SIEMENS

S7-200 SMART G2 系统手册

系统手册

前言

产品概述 1

安全性信息 2

入门指南 3

安装 4

PLC 概念 5

编程概念 6

PLC 设备组态 7

Web 服务器 8

程序指令 9

通信 10

库 11

调试和故障排除 12

PID 回路和整定 13

运动控制 (PLCopen) 14

开环运动控制（原有） 15

技术规范 A

计算功率预算 B

错误代码 C

特殊存储器 (SM) 和系统符号名称 D

参考 E

订购信息 F

03/2026

A5E55110929-AB

法律资讯

警告提示系统

为了您的人身安全以及避免财产损失，必须注意本手册中的提示。人身安全的提示用一个警告三角表示，仅与财产损失有关的提示不带警告三角。警告提示根据危险等级由高到低如下表示。

⚠ 危险
表示如果不采取相应的小心措施，**将会**导致死亡或者严重的人身伤害。

⚠ 警告
表示如果不采取相应的小心措施，**可能**导致死亡或者严重的人身伤害。

⚠ 小心
表示如果不采取相应的小心措施，可能导致轻微的人身伤害。

注意
表示如果不采取相应的小心措施，可能导致财产损失。

当出现多个危险等级的情况下，每次总是使用最高等级的警告提示。如果在某个警告提示中带有警告可能导致人身伤害的警告三角，则可能在该警告提示中另外还附带有可能导致财产损失的警告。

合格的专业人员

本文件所属的产品/系统只允许由符合各项工作要求的**合格人员**进行操作。其操作必须遵照各自附带的文件说明，特别是其中的安全及警告提示。 专业人员是指基于其专业培训及实践经验，熟悉产品的安装、装配、调试、运行、停用和拆卸作业，并具备风险识别与潜在危险规避能力的技术人员。

按规定使用 Siemens 产品

请注意下列说明：

⚠ 警告
西门子产品仅限用于产品样本以及相应使用说明中描述的预定用途。如果要使用其他公司的产品和组件，必须得到 Siemens 推荐和允许。正确的运输、储存、组装、装配、安装、调试、操作和维护是产品安全、正常运行的前提。必须保证允许的环境条件。必须注意相关文件中的提示。

商标

所有带有标记符号 ® 的都是 Siemens Aktiengesellschaft 的注册商标。本印刷品中的其他符号可能是一些其他商标。若第三方出于自身目的使用这些商标，将侵害其所有者的权利。

责任免除

我们已对印刷品中所述内容与硬件和软件的一致性作过检查。然而不排除存在偏差的可能性，因此我们不保证印刷品中所述内容与硬件和软件完全一致。印刷品中的数据都按规定经过检测，必要的修正值包含在下一版本中。

Siemens Aktiengesellschaft
Digital Industries
Postfach 48 48
90026 NÜRNBERG
德国

A5E55110929-AB
Ⓟ 03/2026 本公司保留更改的权利

Copyright © Siemens 2026.
保留所有权利

前言

前言

本文档用途

S7-200 SMART G2 系列包括多种微型可编程逻辑控制器 (Micro PLC, Micro Programmable Logic Controller)，这些控制器可以控制各种自动化应用。S7-200 SMART G2 结构紧凑、成本低廉且具有功能强大的指令集，这使其成为控制小型应用的完美解决方案。S7-200 SMART G2 产品多种多样且提供基于 Windows 的编程工具，支持用户灵活地解决各种自动化问题。

为满足多样化的应用需求，S7200 SMART G2 CPU 系列将提供多种型号，每种型号均具备独特的特性与功能，旨在助力用户构建高效解决方案，后续将陆续推出。

本手册提供了有关 S7-200 SMART G2 CPU 的安装和编程信息，适用于具备可编程逻辑控制器基本知识的工程师、编程人员、安装人员和电气人员。

所需的基本知识

要理解本文档的内容，需要具备自动化和可编程逻辑控制器的基本知识。

本文档的适用范围

本文档适用于 S7-200 SMART G2 CPU 系列和 扩展模块。

有关 S7-200 SMART G2 系列中的各个 CPU 和模块，参见技术规范 (页 1214)。

数字铭牌的 ID 链接

ID 链接是符合 IEC 61406-1 标准的全球唯一标识符，在产品上显示为二维码。

这张图显示了 CPU 的 ID 链接示例。

可以通过右下角带黑色框角的方框来识别 ID 链接。通过 ID 链接可访问产品的数字铭牌。

使用智能手机摄像头、条形码扫描仪或阅读器应用程序扫描产品或包装标签上的二维码，即可访问 ID 链接。

通过数字铭牌，可以找到产品数据、手册、符合性声明、证书和有关产品的其它有用信息。

服务和支持

证书和认证有关证书和认证的详细信息，参见技术规范。

如需要回答任何技术问题、培训或订购 S7 产品方面的帮助，请与西门子经销商或销售部联系。 因为西门子销售代表都经过技术培训并掌握有关操作、过程和工业以及有关您使用的各种西门子产品的最具体的知识，所以他们能够最快最高效地回答您可能遇到的任何问题。

证书和认证

有关证书和认证的详细信息，参见技术规范 (页 1214)。

目录

前言 3

1 产品概述 25

1.1 S7-200 SMART G2 CPU 26

1.2 有哪些新增内容？ 28
1.2.1 S7-200 SMART V3.2 中的新增功能 28
1.2.2 S7-200 SMART V3.1 中的新增功能 30

1.3 S7-200 SMART G2 扩展模块 31

1.4 适用于 S7-200 SMART G2 的 HMI 设备 33

1.5 通信选项 34

1.6 编程软件 35

2 安全性信息 37

2.1 安全信息 37
2.1.1 本文档中的警告 37
2.1.2 安全相关符号 37
2.1.2.1 无防爆保护的设备 37
2.1.2.2 具有防爆保护的设备 38
2.1.3 按预期使用 39
2.1.4 设备和备件的更改 39
2.1.5 知识与个人资质 40
2.1.6 个人防护设备 40
2.1.7 开源软件 40
2.1.8 安全操作 41
2.1.8.1 操作电气部件 41
2.1.9 紧急情况时的行为 42
2.1.10 残余风险 42
2.1.10.1 残余风险 42
2.1.10.2 带电部件 42
2.1.10.3 模块的外部电源 43
2.1.10.4 由于存在允许的电位差，触摸会有触电危险 43
2.1.10.5 导电污染物 44
2.1.10.6 过热 44
2.1.10.7 不受控运动 44
2.1.10.8 不安全的运行状态 45
2.1.11 材料损坏 45
2.1.11.1 运输与储存 45
2.1.11.2 安装和连接 45

2.2 工业网络安全 46
2.2.1 网络安全信息 46
2.2.2 安全说明 46
2.2.3 安全概览 48
2.2.4 网络安全 49
2.2.5 进行以太网通信时使用的通信协议和端口号 51
2.2.6 对通信进行组态 55
2.2.7 CPU 访问保护 57
2.2.8 组态 PLC 安全 57
2.2.8.1 PLC 保护（原有） 59
2.2.8.2 启用用户管理 63
2.2.8.3 通信设置 67
2.2.9 保护项目 68
2.2.9.1 设置项目密码 68
2.2.9.2 POU/数据页面密码保护 69
2.2.10 安全停用 73
2.2.10.1 停用简介 73
2.2.10.2 安全删除数据 73
2.2.10.3 回收和处置 74

3 入门指南 75

3.1 连接到 CPU 75
3.1.1 组态 CPU 以进行通信 76
3.1.1.1 概述 76
3.1.1.2 建立以太网硬件通信连接 77
3.1.1.3 与 CPU 建立以太网通信 77

3.2 创建示例程序 80
3.2.1 程序段 1： 启动定时器 82
3.2.2 程序段 2： 接通输出 83
3.2.3 程序段 3： 复位定时器 84
3.2.4 为项目设置 CPU 的类型和版本 84
3.2.5 保存示例项目 85

3.3 下载示例程序 86

3.4 更改 CPU 的工作模式 86

4 安装 89

4.1 S7-200 SMART G2 设备安装准则 89

4.2 功率预算 93

4.3 安装和拆卸操作步骤 94
4.3.1 S7-200 SMART G2 设备安装尺寸 94
4.3.2 安装和拆卸 CPU 95
4.3.3 安装和拆卸信号板 98

4.3.4 安装和拆卸扩展模块 99
4.3.5 标注 S7200 SMART G2 101
4.3.5.1 贴上参考标识标签 101

4.4 接线准则 102
4.4.1 先决条件 102
4.4.2 I/O 端子接线 110

4.5 维护 111
4.5.1 更换端子排 111

5 PLC 概念 115

5.1 控制逻辑的执行 115
5.1.1 读取输入和写入输出 116
5.1.2 立即读取或写入 I/O 117
5.1.3 执行用户程序 117

5.2 访问数据 119
5.2.1 访问存储区 120
5.2.2 实数格式 127
5.2.3 字符串格式 128
5.2.4 分配指令的常数值 128
5.2.5 对本地 I/O 和扩展 I/O 进行寻址 129
5.2.6 使用指针进行间接寻址 130
5.2.7 指针示例 132

5.3 保存和恢复数据 133
5.3.1 下载项目组件 133
5.3.2 上传项目组件 136
5.3.3 存储类型 138
5.3.4 使用存储卡 138
5.3.5 在标准 CPU 中插入存储卡 140
5.3.6 通过存储卡传送程序 141
5.3.7 上电后恢复数据 145

5.4 更改 CPU 的工作模式 145

5.5 状态 LED 145

6 编程概念 149

6.1 设计 PLC 系统的指南 149

6.2 用户程序的元素 151

6.3 创建用户程序 153
6.3.1 早期版本的 STEP 7-Micro/WIN 项目 153
6.3.2 使用 STEP 7-Micro/WIN SMART 用户界面 157
6.3.3 使用 STEP 7-Micro/WIN SMART 创建程序 160
6.3.4 借助向导创建控制程序 162
6.3.5 梯形图编辑器 162

6.3.6 功能块图编辑器 163
6.3.7 语句表编辑器 164

6.4 数据块 (DB) 编辑器 165

6.5 全局变量表 167

6.6 局部变量表 171

6.7 用户自定义数据类型 175
6.7.1 UDT 表 177

6.8 PLC 错误响应 180
6.8.1 系统信息 181
6.8.1.1 系统 181
6.8.1.2 CPU 182
6.8.1.3 PROFINET 设备 183
6.8.2 事件日志 (Event Log) 184
6.8.3 PROFINET 报警 186
6.8.4 扫描速率 187
6.8.5 非致命错误和 I/O 错误 187
6.8.6 致命错误 189

6.9 在 RUN 模式下执行程序编辑 189

6.10 用于调试程序的功能 192

6.11 轨迹 192
6.11.1 简介 192
6.11.2 用户界面 194
6.11.3 轨迹时序图界面 196
6.11.4 组态轨迹 199
6.11.4.1 新建轨迹 200
6.11.4.2 组态信号表 200
6.11.4.3 组态采样条件 202
6.11.5 管理工程中的轨迹 206
6.11.5.1 打开轨迹 206
6.11.5.2 导入/导出轨迹 207
6.11.6 管理 PLC 中的轨迹 207
6.11.6.1 下载/上传轨迹 207
6.11.6.2 激活/停止轨迹 208
6.11.6.3 轨迹状态表 209
6.11.6.4 删除 PLC 中的轨迹配置和数据 210

7 PLC 设备组态 211

7.1 组态 PLC 系统的运行 211
7.1.1 系统块 211
7.1.2 对通信进行组态 214
7.1.3 组态数字量输入 217

7.1.4 组态数字量输出 220
7.1.5 组态保持范围 221
7.1.6 组态启动选项 223
7.1.7 运动控制 224
7.1.8 组态模拟量输入 225
7.1.9 组态模拟量输出 228
7.1.10 组态 RTD 模拟量输入 231
7.1.11 组态 TC 模拟量输入 235
7.1.12 组态 RS485/RS232 CM01 通信信号板 238
7.1.13 清除 PLC 存储区 239
7.1.14 创建复位为出厂默认存储卡 241

7.2 高速 I/O 241

8 Web 服务器 243

8.1 Web 服务器概述 243

8.2 下载并安装 CA 证书 245
8.2.1 获取 CA 证书 245
8.2.2 安装用于 Windows 的证书 246
8.2.3 安装用于 Mac OS 的证书 247
8.2.4 安装用于移动设备的证书 247

8.3 从 PC 或移动设备访问 Web 服务器 247

8.4 标准 Web 页面 249
8.4.1 Web 页面的布局 249
8.4.2 简介 250
8.4.3 起始页面 251
8.4.4 事件日志 252
8.4.5 状态图 253
8.4.6 监控表 253
8.4.7 数据日志 254
8.4.8 设置时钟 255
8.4.9 模块信息 256

8.5 用户自定义 Web 页面 257
8.5.1 下载用户自定义 Web 资源文件 257

8.6 Web API（应用程序编程接口） 259

9 程序指令 261

9.1 位逻辑 261
9.1.1 标准输入 261
9.1.2 立即输入 263
9.1.3 逻辑堆栈概述 264
9.1.4 STL 逻辑堆栈指令 266
9.1.5 NOT 268

9.1.6 正跳变和负跳变检测器 269
9.1.7 线圈: 输出和立即输出指令 270
9.1.8 置位、复位、立即置位和立即复位功能 271
9.1.9 置位和复位优先双稳态触发器 272
9.1.10 NOP（空操作）指令 273
9.1.11 交替输出和脉冲交替输出 274
9.1.12 沿信号检测器 275
9.1.13 位逻辑输入示例 276
9.1.14 位逻辑输出示例 277

9.2 时钟 280
9.2.1 读取和设置实时时钟 280
9.2.2 读取和设置扩展实时时钟 282

9.3 通信 286
9.3.1 GET 和 PUT（以太网） 286
9.3.2 发送和接收（RS485/RS232 为自由端口） 294
9.3.3 获取端口地址和设置端口地址（RS485/RS232 上的 PPI 协议） 305
9.3.4 获取 IP 地址和设置 IP 地址（以太网） 306
9.3.5 开放式用户通信 309
9.3.5.1 OUC 指令 309
9.3.5.2 OUC 指令错误代码 320

9.4 比较 321
9.4.1 比较数值 321
9.4.2 比较字符串 325

9.5 转换 326
9.5.1 标准转换指令 326
9.5.2 ASCII 字符数组转换 330
9.5.3 数值转换为 ASCII 字符串 336
9.5.4 ASCII 子字符串转换为数值 340
9.5.5 编码和解码 343
9.5.6 量程变换 345

9.6 计数器 348
9.6.1 计数器指令 348
9.6.2 计数器指令(IEC) 352
9.6.3 高速计数器指令 356
9.6.4 高速计数器汇总 358
9.6.5 高速输入降噪 359
9.6.6 高速计数器编程 360
9.6.7 高速计数模式的初始化顺序示例 371
9.6.8 高速频率模式的初始化顺序示例 377

9.7 脉冲输出 381
9.7.1 脉冲输出指令 (PLS) 381
9.7.2 脉冲串输出 (PTO) 382

9.7.3 脉宽调制 (PWM) 385
9.7.4 使用 SM 位置组态和控制 PTO/PWM 操作 385
9.7.5 计算包络表值 389

9.8 数学运算 393
9.8.1 加法、减法、乘法和除法 393
9.8.2 产生双整数的整数乘法和带余数的整数除法 397
9.8.3 三角函数、自然对数/自然指数和平方根 399
9.8.4 递增和递减 402
9.8.5 取最大值或最小值 404
9.8.6 取随机值 405
9.8.7 数组进行排序 407
9.8.8 取数组的平均值 408
9.8.9 取绝对值 410
9.8.10 低通滤波器 411
9.8.11 PID 回路指令 412

9.9 中断 413
9.9.1 中断指令 413
9.9.2 中断例程概述和 CPU 型号事件支持 415
9.9.3 中断编程准则 418
9.9.4 S7-200 SMART CPU 支持的中断事件类型 419
9.9.5 中断优先级、排队和示例程序 420

9.10 逻辑运算 430
9.10.1 取反 430
9.10.2 AND、OR、XOR、NAND、NOR 和 NXOR 431

9.11 传送 436
9.11.1 传送 436
9.11.2 字节、字、双字或实数传送 437
9.11.3 块传送 438
9.11.4 交换字节 439
9.11.5 字节立即传送（读取和写入） 440

9.12 程序控制 441
9.12.1 FOR-NEXT 循环 441
9.12.2 JMP（跳转至标号） 442
9.12.3 SCR（顺控继电器） 444
9.12.4 END、STOP 和 WDR（看门狗定时器复位） 451
9.12.5 GET_ERROR（获取非致命错误代码） 453

9.13 移位和循环移位 454
9.13.1 移位和循环移位 454
9.13.2 移位寄存器位 459

9.14 字符串 461
9.14.1 字符串（获取长度、复制和连接） 461
9.14.2 从字符串中复制子字符串 463

9.14.3 在字符串中查找字符串和第一个字符 465

9.15 表 468
9.15.1 添表 468
9.15.2 先进先出和后进先出 469
9.15.3 存储器填充 471
9.15.4 查表 472

9.16 定时器 476
9.16.1 定时器指令 476
9.16.2 定时器编程提示和示例 479
9.16.3 定时器指令(IEC) 486
9.16.4 定时器编程提示和示例(IEC) 487
9.16.5 时间间隔定时器 490

9.17 子例程 491
9.17.1 CALL（子例程）和 RET（有条件返回） 491
9.17.2 子例程调用参数和返回示例 493

9.18 功能块 497
9.18.1 CALL（功能块）和 RET（有条件返回） 497
9.18.2 功能块调用参数 500

9.19 PROFINET 504
9.19.1 编程指令 "PROFINET" 的特性 504
9.19.2 读取和写入数据记录 504
9.19.2.1 RDREC 和 WRREC 指令的输入和输出接口 504
9.19.2.2 指令 RDREC 和 WRREC 的系统定义错误代码 507
9.19.3 在物理 PROFINET 和存储器地址之间读取和写入多个字节 508
9.19.3.1 BLKMOV_BIR 和 BLKMOV_BIW 的输入和输出接口 508
9.19.3.2 指令 BLKMOV_BIR 和 BLKMOV_BIW 的错误代码 509

10 通信 511

10.1 概述 511

10.2 CPU 通信连接 512

10.3 CPU 通信端口 513

10.4 以太网 513
10.4.1 本地/伙伴连接 513
10.4.2 以太网网络组态示例 514
10.4.3 分配 Internet 协议 (IP) 地址 515
10.4.3.1 为编程设备和网络设备分配 IP 地址 515
10.4.3.2 为项目中的 CPU 或设备组态或更改 IP 地址 516
10.4.3.3 搜索以太网网络上的 CPU 和设备 523
10.4.4 查找 CPU 上的以太网 (MAC) 地址 524
10.4.5 HMI 与 CPU 通信 525
10.4.6 开放式用户通信 527

10.4.6.1 协议 527
10.4.6.2 连接 528
10.4.6.3 端口和 TSAP 528
10.4.7 PROFINET 通信 530
10.4.7.1 简介 530
10.4.7.2 智能设备 531
10.4.7.3 XML 中的设备数据库文件：GSDML 533
10.4.7.4 PROFINET 设备命名 534
10.4.7.5 PROFINET 网络的 LED 状态指示灯 534
10.4.7.6 组态 534

10.5 RS485 552
10.5.1 通信选择 552
10.5.1.1 概述 552
10.5.1.2 PPI 协议 552
10.5.2 通信基础 553
10.5.2.1 S7-200 SMART 网络通信基础 553
10.5.2.2 波特率和网络地址定义 553
10.5.2.3 为 S7-200 SMART CPU 设置波特率和网络地址 554
10.5.3 RS485 网络组态示例 556
10.5.3.1 单主站 PPI 网络 556
10.5.3.2 多主站和多从站 PPI 网络 557
10.5.4 分配 RS485 地址 558
10.5.4.1 为项目中的 CPU 或设备组态或更改 RS485 地址 558
10.5.5 构建网络 561
10.5.5.1 通用准则 561
10.5.5.2 构建网络 561
10.5.5.3 确定网络的距离、传输率和电缆长度 561
10.5.5.4 网络中的中继器 562
10.5.5.5 RS485 电缆规范 563
10.5.5.6 连接器引脚分配 563
10.5.5.7 偏置和端接网络电缆 565
10.5.5.8 偏置和端接 CM01 信号板 565
10.5.6 自由端口模式 566
10.5.6.1 使用自由端口模式创建用户定义的协议 566
10.5.6.2 对 RS232 设备使用 RS232/PPI 多主站电缆和自由端口模式 568
10.5.6.3 RS232/PPI 多主站电缆开关 570

10.6 RS232 572

11 库 575

11.1 库类型（Siemens 及用户定义） 575

11.2 Modbus 通信概述 577
11.2.1 Modbus 寻址 577
11.2.2 Modbus 读取和写入功能 580

11.3 Modbus RTU 库指令 580
11.3.1 Modbus RTU 通信概述 580
11.3.1.1 Modbus RTU 库功能 580
11.3.1.2 使用 Modbus 指令的要求 581
11.3.1.3 Modbus 协议的初始化和执行时间 583
11.3.2 Modbus RTU 主站 583
11.3.2.1 使用 Modbus RTU 主站指令 583
11.3.2.2 MBUS_CTRL/MB_CTRL2/MB_CTRL3 指令（初始化主站） 585
11.3.2.3 MBUS_MSG/MB_MSG2/MB_MSG3 指令 587
11.3.2.4 Modbus RTU 主站执行错误代码 590
11.3.3 Modbus RTU 从站 592
11.3.3.1 使用 Modbus RTU 从站指令 592
11.3.3.2 MBUS_INIT 指令（初始化从站） 594
11.3.3.3 MBUS_SLAVE 指令 596
11.3.3.4 Modbus RTU 从站执行错误代码 597
11.3.4 Modbus RTU 主站示例程序 598
11.3.5 Modbus RTU 高级用户信息 600

11.4 Modbus TCP 库 602
11.4.1 Modbus TCP 库功能 602
11.4.2 Modbus TCP 客户端 604
11.4.2.1 MBUS_CLIENT 指令 604
11.4.2.2 Modbus TCP 客户端执行错误代码 608
11.4.3 Modbus TCP 服务器 609
11.4.3.1 MBUS_SERVER 指令 609
11.4.3.2 Modbus TCP 服务器执行错误代码 611
11.4.4 示例：Modbus TCP 应用程序 611
11.4.5 Modbus TCP 高级用户信息 617
11.4.6 Modbus TCP 常规异常代码 620
11.4.7 Modbus TCP 常规通信异常代码 620

11.5 开放式用户通信库 622
11.5.1 概述 622
11.5.2 OUC 库指令共用的参数 622
11.5.3 开放式用户通信库指令 625
11.5.3.1 TCP_CONNECT 指令 625
11.5.3.2 ISO_CONNECT 指令 628
11.5.3.3 UDP_CONNECT 指令 631
11.5.3.4 TCP_SEND 指令 633
11.5.3.5 TCP_RECV 指令 636
11.5.3.6 UDP_SEND 指令 638
11.5.3.7 UDP_RECV 指令 641
11.5.3.8 DISCONNECT 指令 643
11.5.4 开放式用户通信库指令错误代码 645
11.5.5 开放式用户通信库示例 647

11.5.5.1 主动伙伴（客户端）.....647
11.5.5.2 CheckErrors 子例程.....656
11.5.5.3 主动伙伴变量表.....657
11.5.5.4 被动伙伴（服务器）.....658
11.5.5.5 CheckErrors 子例程.....665
11.5.5.6 被动伙伴变量表.....665

11.6 PN Read Write Record 库.....666
11.6.1 PN Read Write Record 特性.....666
11.6.2 PN Read Write Record 库的输入和输出接口.....667
11.6.3 为输入信号“STATUS”定义参数.....669
11.6.4 PN Read Write Record 库的系统定义错误代码.....669

11.7 USS 协议库指令.....670
11.7.1 USS 通信概述.....670
11.7.1.1 USS 协议概述.....670
11.7.1.2 使用 USS 协议的要求.....671
11.7.1.3 计算与驱动器通信所需的时间.....672
11.7.2 USS 程序指令.....673
11.7.2.1 使用 USS 协议指令.....673
11.7.2.2 USS_INIT 指令.....674
11.7.2.3 USS_CTRL 指令.....676
11.7.2.4 USS_RPM_x 指令.....679
11.7.2.5 USS_WPM_x 指令.....682
11.7.2.6 USS 协议示例程序.....684
11.7.3 USS 协议执行错误代码.....686

11.8 SINAMICS 库.....687
11.8.1 SINAMICS 库特性.....687
11.8.2 SINA_POS 指令.....689
11.8.2.1 使用 SINA_POS 指令的前提.....689
11.8.2.2 SINA_POS 指令的输入和输出接口.....691
11.8.2.3 带 SINA_POS 指令的 SINAMICS 模式选择.....700
11.8.2.4 相对运动.....701
11.8.2.5 绝对运动.....705
11.8.2.6 恒速模式.....709
11.8.2.7 主动回零.....713
11.8.2.8 设置参考点.....716
11.8.2.9 运行程序段.....719
11.8.2.10 点动.....722
11.8.2.11 增量点动.....725
11.8.2.12 编码器调整（仅限 112 报文）.....729
11.8.3 SINA_SPEED 指令.....730
11.8.3.1 使用 SINA_SPEED 指令的前提.....730
11.8.3.2 SINA_SPEED 指令的输入和输出接口.....732
11.8.3.3 “ConfigAxis”参数的定义.....734

11.8.3.4 SINA_SPEED 指令示例 735
11.8.4 SINA_PARA_S 指令 737
11.8.4.1 使用 SINA_PARA_S 指令的前提 737
11.8.4.2 SINA_PARA_S 指令的输入和输出接口 739
11.8.4.3 SINA_PARA_S 指令示例 744

11.9 用户定义的库编程提示 747

11.10 如何重新加载库 754

11.11 用户定义库示例 755
11.11.1 用于缩放模拟值的库示例 755
11.11.2 子例程 S_ITR：整数转换为实数 755
11.11.3 子例程 S_RTI：实数转换为整数 758
11.11.4 子例程 S_RTR： 实数转换为实数 761
11.11.5 创建用户定义库 763
11.11.6 在项目中使用库 767
11.11.7 更改库版本 770

12 调试和故障排除 771

12.1 调试程序 771
12.1.1 书签功能 771
12.1.2 交叉引用表 771
12.1.3 字节使用表 772

12.2 显示程序状态 773
12.2.1 程序编辑器如何显示状态 773
12.2.2 组态 STL 状态选项 776

12.3 使用状态图来监视程序 777

12.4 强制特定值 779

12.5 在 STOP 模式下写入和强制输出 780

12.6 如何执行有限次数的扫描 781

12.7 硬件故障排除指南 782

13 PID 回路和整定 785

13.1 PID 自整定功能介绍 785

13.2 预调节 786

13.3 精确调节 788

13.4 PID 回路定义表 790

13.5 先决条件 796

13.6 自整定序列 796

13.7 例外情况 798

13.8 关于过程变量超限的说明（结果代码 3） 799
13.9 PID 整定控制面板指令 800
14 运动控制 (PLCopen) 805
14.1 基本概念 806
14.2 运动控制资源 808
14.3 运动控制概述 810
14.3.1 控制模式 810
14.3.1.1 位置控制模式（闭环） 810
14.3.1.2 速度控制模式 812
14.3.1.3 单轴转矩控制 821
14.3.2 任务机制 823
14.3.3 过程响应 825
14.3.4 兼容性 826
14.3.4.1 V3.1 和 V3.2 运动控制变更 826
14.3.4.2 全新运动控制全面取代旧版运动向导 828
14.3.4.3 转换为 STEP 7-Micro/WIN SMART V3 (PLCopen)运动控制 828
14.4 组态 831
14.4.1 设定步骤 831
14.4.2 硬件网络拓扑视图 833
14.4.2.1 硬件网络拓扑 833
14.4.2.2 拓扑视图 835
14.4.2.3 设备视图 836
14.4.3 编辑设备和网络 838
14.4.3.1 将设备添加到 PROFINET 网络 838
14.4.3.2 转至在线 839
14.4.3.3 设备连接 840
14.4.3.4 复制/粘贴/删除设备 841
14.4.3.5 移动/交换/插入设备 842
14.4.4 组态设备 843
14.4.4.1 组态控制器 843
14.4.4.2 组态 IRT 设备 844
14.4.4.3 组态 RT 设备 847
14.4.4.4 添加/删除/移动 RT 模块 849
14.4.5 拓扑比较 851
14.4.6 运动控制组件组态窗口 853
14.4.7 组态轴 855
14.4.7.1 组态虚拟轴 856
14.4.7.2 组态 PTO 轴 861
14.4.7.3 组态编码器轴 877
14.4.7.4 组态 PN IRT 轴 884
14.4.8 组态轴组 906
14.4.8.1 轴组基本设置 906

14.4.8.2 诊断信息 908
14.4.9 组态凸轮 908
14.4.9.1 组态凸轮表 908
14.4.9.2 设置参数 911
14.4.9.3 查看曲线 912
14.4.9.4 导入/导出凸轮表 913

14.5 调试（仅适用于 PTO 轴 和 PN IRT 轴） 915
14.5.1 控制面板 915
14.5.2 参数优化 918

14.6 通用机制 922
14.6.1 指令重启 923
14.6.2 状态机 927
14.6.2.1 单轴状态机 927
14.6.2.2 轴组状态机 929
14.6.3 缓冲模式 930
14.6.3.1 单轴命令缓冲模式 930
14.6.3.2 轴组命令缓冲模式 934
14.6.4 变量概览 939
14.6.5 变量一览表 941
14.6.6 使用数组实现多轴控制 951

14.7 运动控制指令 952
14.7.1 基本指令 952
14.7.1.1 MC_Power 952
14.7.1.2 MC_TouchProbe 955
14.7.1.3 MC_AbortTrigger 959
14.7.1.4 MC_Reset 961
14.7.1.5 MC_Home 962
14.7.1.6 MC_Stop 968
14.7.1.7 MC_Halt 971
14.7.1.8 MC_MoveAbsolute 977
14.7.1.9 MC_MoveRelative 982
14.7.1.10 MC_MoveVelocity 989
14.7.1.11 MC_ImmediateStop 994
14.7.1.12 MC_MoveJog 996
14.7.1.13 MC_TorqueControl 1001
14.7.1.14 MC_TorqueRange 1010
14.7.1.15 MC_TorqueLimiting 1016
14.7.2 凸轮和齿轮指令 1020
14.7.2.1 MC_CamIn 1020
14.7.2.2 MC_CamOut 1035
14.7.2.3 MC_GearIn 1040
14.7.2.4 MC_GearOut 1044
14.7.2.5 MC_PhasingRelative 1048

14.7.2.6 MC_GenerateCamTable 1051
14.7.3 轴组指令 1057
14.7.3.1 MC_GroupEnable 1057
14.7.3.2 MC_GroupDisable 1058
14.7.3.3 MC_GroupReset 1059
14.7.3.4 MC_GroupStop 1060
14.7.3.5 MC_MoveLinearAbsolute 1061
14.7.3.6 MC_MoveLinearRelative 1064

14.8 运动控制资源不足时的故障排除与优化 1066

14.9 错误代码 1068
14.9.1 单轴运动的错误代码 1068
14.9.2 电子凸轮和齿轮的错误代码 1074
14.9.3 轴组的错误代码 1076

14.10 示例程序 1077
14.10.1 运动控制示例 1077
14.10.2 单轴示例 1083
14.10.3 轴组示例 1086
14.10.4 凸轮示例 1089

15 开环运动控制（原有） 1095

15.1 使用 PWM 输出 1096
15.1.1 使用 PWM（脉宽调制）输出 1096
15.1.2 组态 PWM 输出 1097
15.1.3 PWMx_RUN 子例程 1098

15.2 使用运动控制 1098
15.2.1 最大速度和启动/停止速度 1099
15.2.2 输入加速和减速时间 1100
15.2.3 组态运动包络 1100

15.3 运动控制的特点 1102

15.4 使用运动轴 1104
15.4.1 编程运动轴 1104
15.4.2 组态运动轴 1106
15.4.3 运动向导为运动轴创建的子程序 1121
15.4.3.1 运动控制子例程使用准则 1122
15.4.3.2 AXISx_CTRL 子例程 1123
15.4.3.3 AXISx_MAN 子例程 1124
15.4.3.4 AXISx_GOTO 子例程 1126
15.4.3.5 AXISx_RUN 子例程 1128
15.4.3.6 AXISx_RSEEK 子例程 1129
15.4.3.7 AXISx_LDOFF 子例程 1130
15.4.3.8 AXISx_LDPOS 子例程 1131
15.4.3.9 AXISx_SRATE 子例程 1132

15.4.3.10 AXISx_DIS 子例程 1133
15.4.3.11 AXISx_CFG 子例程 1134
15.4.3.12 AXISx_CACHE 子例程 1135
15.4.3.13 AXISx_RDPOS 子例程 1136
15.4.3.14 AXISx_ABSPOS 子例程 1137
15.4.4 使用 AXISx_ABSPOS 子程序从 SINAMICS 伺服驱动器读取绝对位置 1138
15.4.4.1 AXISx_ABSPOS 和 AXISx_LDPOS 子程序应用示例 1139
15.4.4.2 互连 1140
15.4.4.3 调试 1141
15.4.4.4 重要事项须知 1144
15.4.5 运动轴示例程序 1145
15.4.5.1 运动轴简单相对移动（定长截断应用）示例 1145
15.4.5.2 运动轴 AXISx_CTRL、AXISx_RUN、AXISx_SEEK 和 AXISx_MAN 示例 1147
15.4.6 监视运动轴 1153
15.4.6.1 显示和控制运动轴的操作 1153
15.4.6.2 显示和修改运动轴的组态 1158
15.4.6.3 显示运动轴的曲线组态 1158
15.4.6.4 运动轴错误代码（SMW620、SMW670、SMW720 或 SMW770 的 WORD） 1159
15.4.6.5 运动轴指令的错误代码（SMB634、SMB684、SMB734 或 SMB784 的七个 LS 位） ... 1161
15.4.7 高级主题 1162
15.4.7.1 理解运动轴的组态/曲线表 1162
15.4.7.2 运动轴的特殊存储器 (SM) 位置 1172
15.4.8 了解运动轴的 RP 搜索模式 1175
15.4.8.1 选择工作区位置以消除反冲 1178

15.5 使用运动轴组 1179
15.5.1 为运动轴组编程 1179
15.5.2 组态运动轴组 1180
15.5.3 运动向导为运动轴组创建的子程序 1186
15.5.3.1 运动控制子例程使用准则 1187
15.5.3.2 GRPx_2/3D_MOVELINEAR 子例程 1187
15.5.3.3 GRPx_MOVEPATH 子例程 1190
15.5.3.4 GRPx_RESET 子例程 1192
15.5.3.5 GRPx_STOP 子例程 1193
15.5.4 运动轴组示例程序 1195
15.5.4.1 使用 GPRx_STOP 子例程停止轴组运动的示例 1203

15.6 停止轴组运动 1205

15.7 运动轴组错误代码（SMW820 或 SMW890 的 WORD） 1205

15.8 运动轴组指令的错误代码（SMB823 或 SMB893 的七个 LS 位） 1206

15.9 轴和轴组之间的交互 1207

A 技术规范 1209

A.1 常规技术规范 1209

A.2 S7-200 SMART CPU 1214
A.2.1 CPU ST32 1214
A.2.1.1 常规规范和特性 1214
A.2.1.2 数字量输入和输出 1220
A.2.1.3 接线图 1222

A.3 数字量输入和输出扩展模块 (EM) 1224
A.3.1 EM DE16 数字量输入规范 1224
A.3.2 EM QT16 数字量输出规范 1225

A.4 模拟量输入和输出扩展模块 (EM) 1227
A.4.1 EM AE04 和 EM AE08 模拟量输入规范 1227
A.4.2 EM AQ04 模拟量输出模块规范 1230
A.4.3 EM AM06 模拟量输入/输出模块规范 1233
A.4.4 模拟量输入的阶跃响应 1236
A.4.5 模拟量输入的采样时间和更新时间 1237
A.4.6 模拟量输入的电压和电流测量范围（SB 和 SM） 1237
A.4.7 模拟量输出的电压和电流测量范围（SB 和 EM） 1238

A.5 热电偶扩展模块 (EM) 1240
A.5.1 EM AT08 热电偶规范 1240

A.6 RTD 扩展模块(EM) 1244
A.6.1 EM AR04 RTD 规范 1244

A.7 RS485/RS232 信号板 1249
A.7.1 SB RS485/RS232 规范 1249

B 计算功率预算 1253

B.1 功率预算 1253

B.2 功率要求计算示例 1254

B.3 计算功率要求 1255

C 错误代码 1257

C.1 时间戳不匹配 1257

C.2 PLC 非致命错误代码 1258

C.3 PLC 非致命错误 SM 标志 1261

C.4 PLC 致命错误代码 1262

C.5 PROFINET 报警 1264

D 特殊存储器 (SM) 和系统符号名称 1265

D.1 SM（特殊存储器）概述 1265

D.2 SMB0： 系统状态 1268

D.3 SMB1： 指令执行状态 1269

D.4 SMB2： 自由端口接收字符 1269
D.5 SMB3： 自由端口字符错误 1270
D.6 SMB4、SM20.0： 中断队列溢出、运行时程序错误、中断启用、自由端口发送器空闲和强制值 1270
D.7 SMB5： I/O 错误状态 1272
D.8 SMB6-SMB7： CPU ID、错误状态和数字量 I/O 点 1272
D.9 SMB8-SMB19、SMB131-SMB134、SMB2030-SMB2045：I/O 模块 ID 和错误 1273
D.10 SMW22-SMW26：扫描时间 1275
D.11 SMB28-SMB29、SMB32-SMB33：信号板 ID 和错误 1276
D.12 SMB30：（端口 0）、SMB130：（端口 1）和 SMB31：（端口 2） 1277
D.13 SMB34-SMB35： 定时中断的时间间隔 1279
D.14 SMB36-SMB45 (HSC0)、SMB46-SMB55 (HSC1)、SMB56-SMB65 (HSC2)、SMB136-SMB145 (HSC3)、SMB146-SMB155 (HSC4)、SMB156-SMB165 (HSC5)、SMB360-SMB369 (HSC6)、SMB370-SMB379 (HSC7)：高速计数器 1279
D.15 SMB66-SMB85 (PTO0/PWM0, PTO1/PWM1)、SMB166-SMB169 (PTO0)、SMB176-SMB179 (PTO1)、SMB566-SMB579 (PTO2/PWM2)、SMB580-SMB593 (PTO3/PWM3)、SMB2000-SMB2013 (PTO4/PWM4)：高速输出 1282
D.16 SMB86-SMB94、SMB186-SMB194 和 SMB405-SMB413：接收信息控制 1290
D.17 SMW98：扩展 I/O 总线通信错误 1295
D.18 SMW100-SMW120、SMW2014-SMW2028 系统报警 1296
D.19 SMB195-SMB210：HSC 类型和频率测量采样周期 1298
D.20 SMB220-SMB235：PID 控制错误代码 1298
D.21 SMB380-SMB404：HSC 用户自定义引脚映射启用 1299
D.22 SMB480-SMB515： 数据日志状态 1300
D.23 SMB600-SMB799：轴（0、1、2 和 3）开环运动控制（适用于运动控制 (V2.x)） 1301
D.24 SMB600-SMB649：轴 0 开环运动控制 1303
D.25 SMB650-SMB699：轴 1 开环运动控制 1303
D.26 SMB700-SMB749：轴 2 开环运动控制 1303
D.27 SMB750-SMB799：轴 3 开环运动控制 1304
D.28 SMB800-SMB858：轴组 0 开环运动控制 1304
D.29 SMB870-SMB928：轴组 1 开环运动控制 1305
D.30 SMB929-SMB936：轨迹控制和轨迹状态 1306
D.31 SMB1000-SMB1049： CPU 硬件/固件 ID 1306

D.32 SMB950-SMB998、SMB1050-SMB1099：SB（信号板）硬件/固件 ID 1307
D.33 SMB1100-SMB1399、SMB250-SMB349、SMB2050-SMB2449：EM（扩展模块）硬件/固件 ID 1308
D.34 SMB1400-SMB1799、SMB2450-2999：EM（扩展模块）模块特定的数据 1317
D.35 SMB1800-SMB1939：PROFINET 设备状态 1318
D.36 SMB1940-SMB1946：Web 服务器状态 1319
E 参考 1321
E.1 常用特殊存储器位 1321
E.2 按优先级别顺序排列的中断事件 1322
E.3 高速计数器汇总 1325
E.4 STL 指令 1325
E.5 CPU 存储器范围和特性 1334
F 订购信息 1339
F.1 CPU 模块 1339
F.2 扩展模块（EMs）和信号板（SBs） 1339
F.3 编程软件 1339
F.4 备件和其它硬件 1340
索引 1341

产品概述

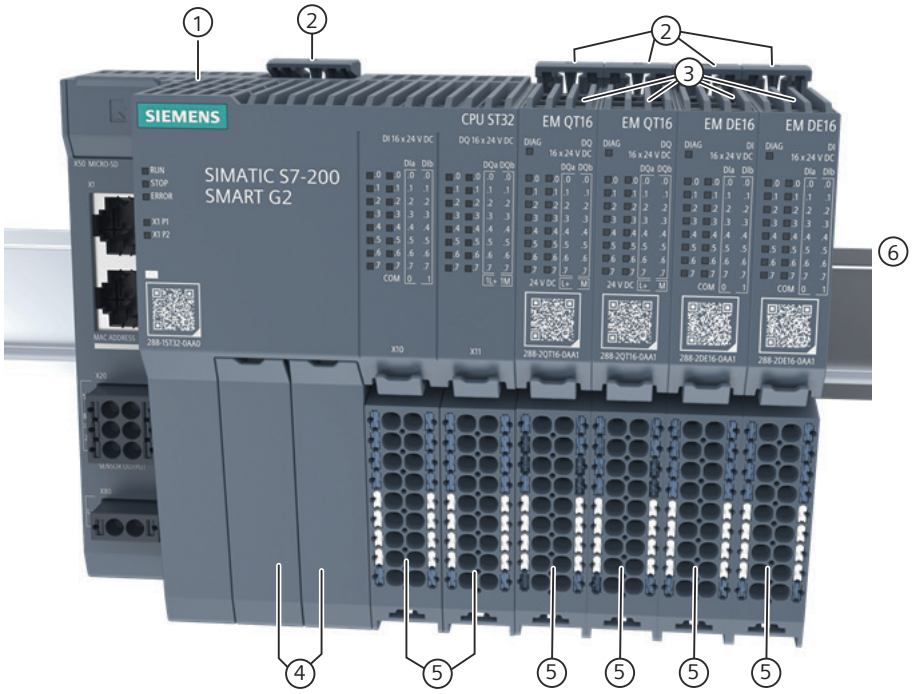
1

S7-200 SMART G2 系列是 S7-200 SMART 的第二代，其占用空间比第一代窄得多。CPU 和扩展模块具有高度的灵活性，可支持多样化的自动化应用场景。

CPU 根据用户程序控制逻辑监视输入并更改输出状态，用户程序可以包含布尔逻辑、计数、定时、复杂数学运算以及与其它智能设备的通信。 S7-200 SMART G2 结构紧凑、组态灵活且具有功能强大的指令集，这些优势的组合使它成为控制各种应用的完美解决方案。

S7-200 SMART G2 系统

S7-200 SMART G2 模块可安装在一根安装导轨上，最多可在安装导轨上安装 19 个模块（CPU，2 个信号板和 16 个 I/O 模块）。



① CPU 模块

② 安装导轨释放按钮

③ I/O 模块

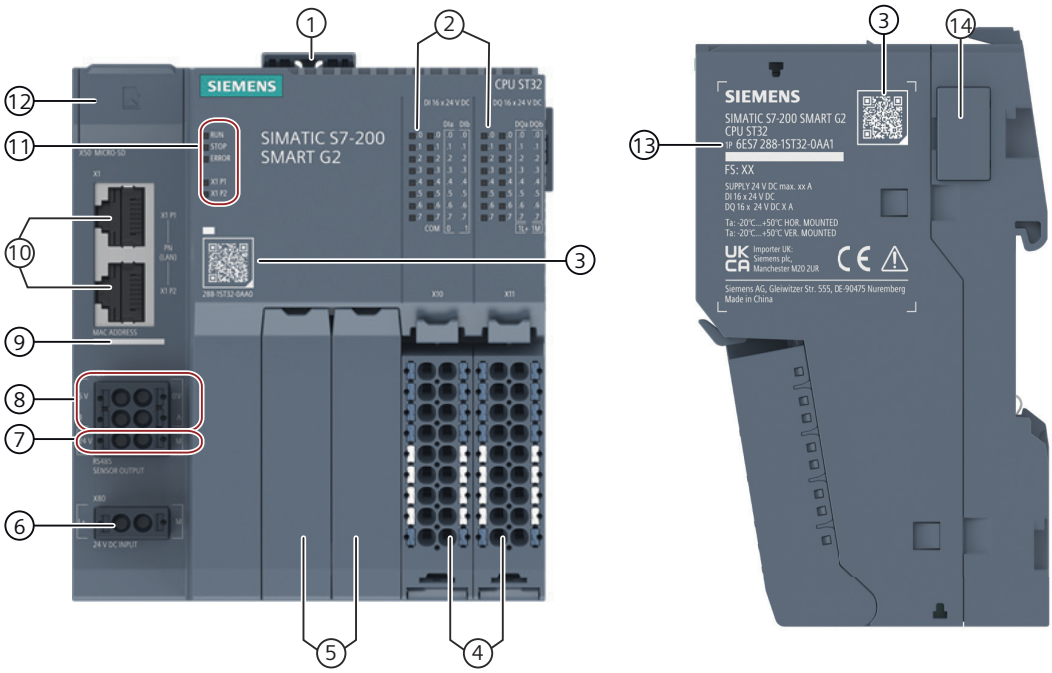
④ 可选信号板

⑤ 端子连接器

⑥ 安装导轨

1.1 S7-200 SMART G2 CPU

CPU 将微处理器、集成电源、输入电路和输出电路组合到一个结构紧凑的外壳中，形成一个功能强大的 Micro PLC。下载用户程序后，CPU 中将包含监控应用程序中输入和输出设备所需的逻辑。



①	安装导轨释放按钮	⑧	RS485 接口
②	板载 I/O 的状态 LED	⑨	MAC 地址
③	ID 链接	⑩	双端口以太网接口
④	端子连接器	⑪	CPU 的状态 LED 指示灯
⑤	可选信号板	⑫	存储卡插槽（盖子下方）
⑥	电源输入连接器	⑬	订货号
⑦	24 V DC 传感器电源输出	⑭	I/O 总线连接器盖

CPU 具有不同型号，它们提供了各种各样的特征和功能，这些特征和功能可帮助用户针对不同的应用创建有效的解决方案。有关特定 CPU 的详细信息，请参见 “技术规范 (页 1214)”。

继电器输出型 CPU 将于后续发布。

有关 CPU 和扩展模块的功率要求信息，请参见 “技术规范 (页 1214)”。使用 “附录 B 计算功率预算 (页 1253)”中的工作表，计算功率预算 (页 1253)。

S7-200 SMART G2 CPU

	ST32
标准，可扩展	✓
继电器输出	
晶体管输出 (DC)	✓
I/O 点（内置）	32

标准型可扩展 CPU

特性		CPU ST32
尺寸：W x H x D (mm)		100 x 118 x 74.5
用户存储器	程序	120 KB[1]
	用户数据	100 KB
	保持性	最大 40 KB[2]
	非绑定全局变量和 FB 实例的存储器 (DB2)	1 MB
板载数字量 I/O	• 输入 • 输出	• 16 DI • 16 DQ
扩展模块		最多 16 个
信号板		2 个
高速计数器（总共 8 个）	单相/双相	8 个，200 kHz
	A/B 相	8 个，200 kHz 4x 频率支持，最高 800 kHz
脉冲输出 [3]		5 个，200 kHz
PID 回路		16
实时时钟，备用时间 7 天		✓

[1] CPU ST32(V3.1)的用户程序大小为 100 KB。

[2] 可组态 V 存储器、M 存储器、C 存储器（当前值）、DB2 的存储区，以及 T 存储器（保持性定时器上的当前值）中要保持的部分，最高至最大指定量。

[3] 指定的最大脉冲频率仅适用于带晶体管输出的 CPU 型号。对于带有继电器输出的 CPU 型号，不建议进行脉冲输出操作。

1.2 有哪些新增内容？

1.2.1 S7-200 SMART V3.2 中的新增功能

只有固件版本为 V3.2 的以下 CPU 型号支持本章所述的新功能：

表格 1-1 受固件 V3.2 影响的 CPU 型号

CPU 型号	订货号
CPU ST32，DC/DC/DC	6ES7288-1ST32-0AA0

说明

固件版本为 V3.2 的 CPU 型号不能降级到 V3.1 以及任何以前的版本。

运动控制（PLCOpen）

全新的运动控制带来总线级实时同步与更高精度的多轴插补能力，同时新增扭矩控制相关指令完善力矩管理能力。

- 新增 IRT 运动控制
  - 新增 IRT 轴
  - 支持最小 1 毫秒 的 PROFINET 总线运动控制周期
  - 支持总线的配置，调试，诊断功能
  - 支持单轴，凸轮，齿轮运动
  - 支持多种控制模式：位置模式、速度模式和扭矩模式
- 新增运动指令：
  - MC_TorqueControl（页 1001）
  - MC_TorqueRange (页 1010)
  - MC_TorqueLimiting (页 1016)

硬件网络拓扑

新增硬件网络拓扑，用于配置 PROFINET 网络

- 拓扑视图 (页 835)： 组态 PROFINET 网络拓扑
- 设备视图 (页 836)： 允许您管理和配置 RT 设备
- 拓扑比较： 能够比较当前在线网络与配置网络的差异

当您使用 STEP 7-Micro/WIN SMART （3. 2 版）打开一个在 之前 版本中创建的、包含 PROFINET RT 配置的项目时，这些 PRONFINET 向导配置将自动迁移并集成到 "硬件网络拓扑"中。

轨迹

支持 IRT 采样周期。

编程

- UDT 、FB 和变量表的导入与导出。

新增指令

- R_TRIG (页 275)
- T_TRIG (页 275)
- CTU_DINT (页 352)
- CTD_DINT (页 352)
- CTUD_DINT (页 352)
- IEC_TON (页 486)
- IEC_TOF (页 486)
- IEC_TONR (页 486)

项目打包

在“文件”菜单新增了“项目打包”按钮，您可以将工程文件以及以下内容打包成一个 .zip 压缩包：

- 项目使用的 GSDML 文件（默认包含）
- Trace 轨迹（可选）

PLC 信息

在扫描速率 (页 187)页面新增了 MC_USER_TASK 的扫描速率，以及运动任务的最大/最小和超时等监控。

SmartBot

说明

SmartBot 仅限在中国大陆地区使用。

STEP 7- Micro/WIN SMART V3 （3.2 版）新增 AI 助手 "SmartBot"，用于常见问题解答。

删除向导

- PROFINETSTEP 7-Micro/WIN SMART （3.2 版）中不再提供旧版 RPOFINET 向导。
- **文本显示**：STEP 7-Micro/WIN SMART （3.2 版）中不再提供用于组态 TD400 的**文本显示**向导。
  当您使用 STEP 7-Micro/WIN SMART （3.2 版）打开一个在 之前 版本中创建的、包含**文本显示**配置的项目时，程序仍然可用，但是不可新增或修改原有配置。

1.2.2 S7-200 SMART V3.1 中的新增功能

全面支持 S7-200 SMART G2 模块。

ST32 最多支持连接 16 个扩展模块。

模块类型	模块描述	订货号
CPU	CPU ST32（DC/DC/DC）	6ES7288-1ST32-0AA0
信号板	SB RS485/RS232 (SB CM01)	6ES7288-5CM01-0AA1
扩展模块	EM 16 点数字量输入 (EM DE16)	6ES7288-2DE16-0AA1
	EM 16 点晶体管型数字量输出 (EM QT16)	6ES7288-2QT16-0AA1
	EM 4 点模拟量输入 (EM AE04)	6ES7288-3AE04-0AA1
	EM 4 点模拟量输出 (EM AQ04)	6ES7288-3AQ04-0AA1
	EM 8 点模拟量输入 (EM AE08)	6ES7288-3AE08-0AA1
	EM 4 点模拟量输入/2 点模拟量输出 (EM AM06)	6ES7288-3AM06-0AA1
	EM RTD 4 x 16 位 (EM AR04)	6ES7288-3AR04-0AA1
	EM TC 8 x 16 位 (EM AT08)	6ES7288-3AT08-0AA1

迁移（Migrate）

迁移 (页 153)：STEP 7-Micro/WIN SMART V3 （3.1 版）允许您将 STEP 7-Micro/WIN SMART（2.8 版）或 STEP 7-Micro/WIN SMART V3（3.0 版）的工程及组态迁移至 S7-200 SMART G2 系列。

在项目迁移过程中，原工程中使用的 CPU 将默认迁移至 ST32；信号板（SB）将迁移至 S7-200 SMART G2 系列中同名的信号板，若无同名信号板，则其配置将被清除；扩展模块（EM）将根据匹配规则替换为 S7-200 SMART G2 扩展模块（EM）。

轨迹（Trace）

轨迹功能 (页 192)用于实时监测和记录程序中的变量值。

一旦轨迹激活，它将开始记录带有时间戳的变量值。这些记录的数据可以通过 STEP 7-Micro/WIN SMART V3（ 3.1 版）进行持续监测，为您调试提供依据。轨迹激活后，轨迹功能将独立于主程序执行。

轨迹数据是临时存储在随机存取存储器（RAM）中的，为了保留轨迹记录，您必须将其上传至 STEP 7-Micro/WIN SMART V3（ 3.1 版）中后导出保存。

编程

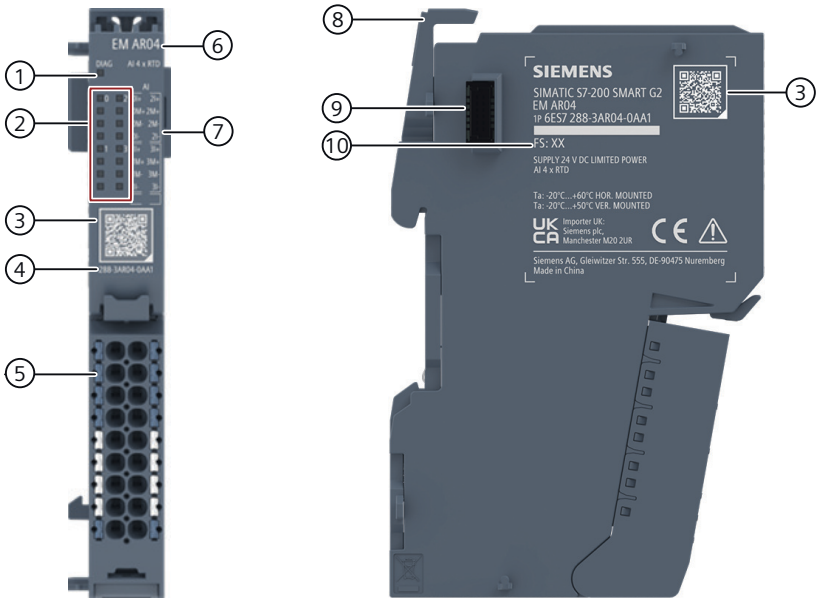
- 用户定义库 (页 747)支持用户自定义数据类型 (UDT) 和功能块 (FB)。
- 用户定义库 (页 747)支持使用 DB2。
- 在创建用户定义库 (页 763)时，允许您定义命名空间，配置库信息。
- 更新参数，当块的输入/输出参数声明被修改时，支持在调用位置更新块参数 (包括子程序 SBR 和功能块 FB)。

1.3 S7-200 SMART G2 扩展模块

为更好的满足应用需求，S7-200 SMART G2 系列包括诸多扩展模块和信号板。可以将这些扩展模块与标准 CPU 型号 (ST32) 一起使用，以扩展 CPU 的功能。

扩展模块 (EM)

ST32 CPU 最多支持 16 个扩展模块。



①	用于诊断的 LED 指示灯	⑥	模块类型和名字
②	通道状态 LED 指示灯	⑦	端子口定义
③	ID 链接	⑧	安装导轨释放按钮
④	订货号	⑨	IO 总线连接器
⑤	端子连接器	⑩	FS 号

信号板(SB)

可以添加信号板（例如 CM01）来增强 CPU 的通信能力。

- ST32 CPU 支持两个插入式扩展板。

在下图中，突出显示了 插入式扩展板， 并且标记了其主要特征：

下表列出了当前提供的扩展模块。有关特定模块的详细信息，请参见技术规范 (页 1209)。

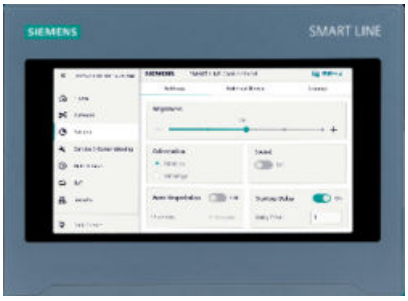
表格 1-2　扩展模块和信号板

类型	仅输入	仅输出	输入/输出组合	其他
数字扩展模块	• 16 个直流输入	• 16 个直流输出		
模拟量扩展模块	• 4 个模拟量输入 • 8 个模拟量输入 • 4 个 RTD 输入 • 8 个热电偶输入	• 4 个模拟量输出	• 4 个模拟量输入/2 个模拟量输出	
信号板				• RS485/RS232

1.4 适用于 S7-200 SMART G2 的 HMI 设备

S7-200 SMART G2 支持 Comfort HMI、SMART HMI、Basic HMI 和 Micro HMI。以下显示 SMART LINE 触摸面板。

表格 1-3　HMI 设备

	**SMART HMI：**SMART LINE 触摸面板可为小型机器和工厂提供操作和监视功能。组态和调试时间短、在 WinCC flexible(ASIA 版本)中组态以及具备双端口 Ethernet/RS485 接口，共同构成这些 HMI 的亮点。

可从 Siemens 客户支持网站 (http://www.siemens.com/automation)下载 SIMATIC 文本显示 (TD) 用户手册。

1.5 通信选项

S7-200 SMART 可实现 CPU、编程设备和 HMI 之间的多种通信：

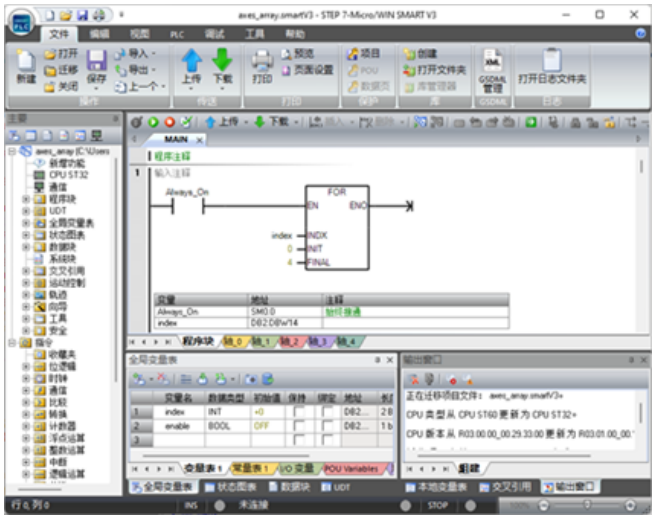
- 以太网：
  - 编程设备到 CPU 的数据交换
  - HMI 与 CPU 间的数据交换
  - S7 与其他 S7-200 SMART CPU 的对等通信
  - 与其他具有以太网功能的设备间的开放式用户通信 (OUC)
  - 使用 PROFINET 设备的 PROFINET 通信
- RS485：
  - 使用 USB-PPI 电缆时，提供一个适用于编程的 STEP 7-Micro/WIN SMART V3 连接

    说明

    S7-200 SMART G2 CPU 不支持直接连接 USB-PPI 线缆。若需使用此线缆连接 S7-200 SMART G2 CPU，则需要自行制作 RS485 连接线。

  - 总共支持 126 个可寻址设备（每个程序段 32 个设备）
  - 支持 PPI（点对点接口）协议
  - HMI 与 CPU 间的数据交换
  - 使用自由端口在设备与 CPU 之间交换数据（XMT/RCV 指令）
- RS232：
  - 支持与一台设备的点对点连接
  - 支持 PPI 协议
  - HMI 与 CPU 间的数据交换
  - 使用自由端口在设备与 CPU 之间交换数据（XMT/RCV 指令）

1.6 编程软件



STEP7-Micro/WIN SMART V3 提供了一个用户友好的环境，用于开发、编辑和监视控制用户应用所需的逻辑。

顶部是常见任务的快速访问工具栏，其后是所有公用功能的菜单。左边是用于对组件和指令进行便捷访问的项目树和导航栏。打开的程序编辑器和其他组件占据用户界面的剩余部分。

STEP7-Micro/WIN SMART V3 提供了三个程序编辑器（LAD、FBD 和 STL），用于方便高效地开发适合用户应用的控制程序。

为了帮助用户找到所需信息，STEP7-Micro/WIN SMART V3 还提供了一个内容丰富的在线帮助系统。

计算机要求

STEP 7-Micro/WIN SMART 在个人计算机上运行。计算机应满足以下最低要求：

- 操作系统：Windows 10（32 位和 64 位两种版本）或 Windows 11
- 至少 500M 字节的空闲硬盘空间
- 鼠标（推荐）

说明

Windows 补丁要求

务必使用 Microsoft 的最新安全补丁！

安全性信息 2

2.1 安全信息

2.1.1 本文档中的警告

有关本文档中所使用的警告的说明，请参见“法律信息”部分。

2.1.2 安全相关符号

2.1.2.1 无防爆保护的设备

下表列出了 SIMATIC 设备、设备包装或随附文档中的符号的说明。

符号	含义
	常规警告符号**小心/注意** 必须阅读产品文档。产品文档包含各种潜在风险的信息，可帮助用户识别风险、采取应对措施。
	阅读产品文档中的相关信息。 ISO 7010 M002
	确保设备仅由电气专业人员进行安装。 IEC 60417 No. 6182
CABLE SPEC.	注意，连接的电源线必须根据预期的最低和最高环境温度进行设计。
EMC	注意，设备的构造和连接必须符合 EMC 规定。
230V MODULES	注意，230 V 设备可能会暴露于危险电压下。 ANSI Z535.2
III 24V MODULES	注意，防护等级为 III 级的设备只能使用符合 SELV/PELV 标准的保护性低电压。 IEC 60417-1-5180“III 级设备”

符号	含义
INDOOR USE ONLY INDUSTRIAL USE ONLY	注意，设备仅获准用于工业领域，且仅限室内使用。
	注意，安装设备时，需要使用外壳。可以考虑使用以下外壳： • 立式开关柜 • 串行开关柜 • 接线盒 • 壁挂外壳

2.1.2.2 具有防爆保护的设备

下表列出了 SIMATIC 设备、设备包装或随附文档中的符号的说明。

符号	含义
ONLY EX MODULES	指定的安全符号适用于**具有防爆认证**的设备。 必须阅读产品文档。产品文档包含各种潜在风险的信息，可帮助用户识别风险、采取应对措施。
	阅读产品文档中的相关信息。 ISO 7010 M002
	确保设备仅由电气专业人员进行安装。 IEC 60417 No. 6182
F<2N DISPLAY F<4N HOUSING	遵循设备的机械额定值。
CABLE SPEC.	注意，连接的电源线必须根据预期的最低和最高环境温度进行设计。
EMC	注意，设备的构造和连接必须符合 EMC 规定。
U = 0V	注意，设备处于欠压状态时，不得进行安装/拆卸，也不得进行插/拔。

符号	含义
230V MODULES	注意，230 V 设备可能会暴露于危险电压下。 ANSI Z535.2
24V MODULES	注意，防护等级为 III 级的设备只能使用符合 SELV/PELV 标准的保护性低电压。 IEC 60417-1-5180“III 级设备”
INDOOR USE ONLY INDUSTRIAL USE ONLY	注意，设备仅获准用于工业领域，且仅限室内使用。
ZONE 2 INSIDE CABINET IP54	注意，在区域 2 这样的易爆气体环境中，设备只有安装到防护等级 ≥ IP54 的外壳中，才可使用。
ZONE 22 INSIDE CABINET IP6x	注意，在区域 22 这样的易爆气体环境中，设备只有安装到防护等级 ≥ IP6x 的外壳中，才可使用。

2.1.3 按预期使用

系统用于控制机器和设备。

按预期使用还包括遵循本文档，尤其是遵循安全说明和使用条件。请参见“技术规格”部分。

⚠ 警告
通过 ID 链接下载文档 本手册通过电子媒介交付。可以通过 ID 链接查看和下载文档。

2.1.4 设备和备件的更改

对设备进行修改，可能会损害其安全和功能：

- 不得对设备进行任何更改或修改。
- 不得移除或覆盖设备上的安全说明。
- 不得用胶带粘贴通风槽、遮盖通风槽或另行安装通风槽。
- 只能使用原厂备件和附件。
- 只能使用西门子提供的软件。

2.1.5 知识与个人资质

使用本设备的所有人员需要具备以下知识：

- 本文档以及随附文档的内容
- （按照说明）操作设备
- 相关标准与法规
- 事故预防指南

以下活动只能由专业人员完成：

操作电气部件

只有以下人员才能操作电气部件：

- 专业电工
- 接受过电气工程培训并在专业人员的指导和监督下工作的人员

调试和组态

调试和组态需要具备自动化技术领域的常规知识。

2.1.6 个人防护设备

个人防护设备由操作员根据活动而定。

为了避免在包装、拆包和安装活动中损坏材料，我们建议：

- 戴 ESD 腕带
- 穿 ESD 鞋

2.1.7 开源软件

本产品、解决方案或服务（“产品”）包含第三方软件组件。这些组件是依据开源倡议组织 (https://opensource.com/)认可的许可证或西门子定义为等效的许可证授权的开源软件（“OSS”）和/或商业软件或免费软件。对于 OSS 组件，适用的 OSS 许可证条件优先于涉及产品的任何其它条款和条件。本产品的 OSS 部分免版税，可免费使用。

如果西门子已将产品的某些组件与依据 GNU LGPL 版本 2 或更高版本授权的 OSS 组件进行组合或链接（按照适用许可证的定义），并且使用相应目标文件受到限制（“LGPL 授权模块”，其中 LGPL 授权模块及与其组合或链接的组件构成“组合产品”），则当满足相关 LGPL 许可证标准时，以下附加权利适用：(i) 用户有权出于个人使用目的修改组合产品，包括但不限于有权修改组合产品来重新链接 LGPL 授权模块已修改的版本，(ii) 可对组合产品进行逆

向工程，但仅限调试修改目的。修改权不包括这些修改的分发权，且用户必须对通过组合产品逆向工程而获取的任何信息予以保密。

某些 OSS 许可证（例如 GNU General Public License、GNU Lesser General Public License 和 Mozilla Public License）要求西门子提供源代码。如果此类许可证适用，且本产品未随附所需源代码，在适用 OSS 许可证规定的期限内，任何收到本信息的人，均可通过联系以下地址获取该源代码的副本。

Siemens Aktiengesellschaft
LC DI FA SL
Werner-von-Siemens Str.60
91052 Erlangen
Germany

关键字：开源请求（适用时，请指定产品名称和版本）

西门子为满足这一请求，可能收取最高 5 欧元的手续费。

关于开源软件进一步使用的保修条款

用户与西门子签订的协议，明确阐述了西门子需承担的保修义务。如果用户对本产品或 OSS 组件进行修改或者不按西门子的规定使用本产品或 OSS 组件，则保修将失效且不再提供技术支持。以下许可条款可能包括责任限制条例，这些条例适用于用户和相应许可方。未免生疑义，西门子公司不代表任何第三方许可方作出任何保修承诺，此类承诺亦不对第三方许可方具有约束力。产品中使用的开源软件以及与该软件相关的许可协议，请参见 Readme_OSS。

2.1.8 安全操作

2.1.8.1 操作电气部件

- 只有专业人员才可操作电气部件（请参见“知识与个人资质 (页 40)”部分）。
- 务必遵守国家/地区特定的安全规则。
- 通知所有将受该程序影响的人员。
- 遵守 DIN EN 50110-1 (VDE 0105-1) 标准中规定的 5 条安全规则：

1. 断开连接
2. 采取防护措施，防止重新连接
3. 验证是否与电源（所有电极）安全隔离
4. 接地和短路
5. 在相邻带电部件周围加上隔板，或将其盖住

在开始操作电气系统之前，必须按照上述顺序执行 5 条安全规则。完成相关工作后，请按相反顺序进行操作。

假设每个电工都熟悉这 5 条规则。

2.1.9 紧急情况时的行为

- 强制紧急切断。

恢复安全运行状态后，请执行以下操作：

- 解锁紧急切断机制。
- 负责系统的人员确保系统以可控且明确的方式启动。

2.1.10 残余风险

2.1.10.1 残余风险

尽管已采取各种技术和流程手段降低风险，但并非所有危险都能避免。

以下章节将介绍这些残余风险以及规避措施。

2.1.10.2 带电部件

接触带电部件可能触电，造成危及生命的伤害（死亡）。

操作

- 请勿拆开设备。
- 不得使用损坏的设备。
- 拔插头，切勿拽电缆。

安装和连接

- 只有专业人员才可操作电气部件（请参见“知识与个人资质 (页 40)”部分）。
- 遵守防护措施安全操作电气部件（请参见“操作电气部件 (页 41)”部分）
- 遵循适用标准，根据所用电缆的载流能力熔断连接导线。

- 在开关柜中安装设备。外壳、机柜或电气设备室必须提供相应的保护，以避免电击和火势蔓延。
- 遵循适用规定为设备接地。
- 对于 24 V DC 电源 (SELV/PELV)，只能 使用符合 IEC 61131-2 或 IEC 61010-2-201 标准、可提供安全超低电压的电源装置。

维护

- 只有专业人员才可操作电气部件（请参见“知识与个人资质 (页 40)”部分）。
- 遵守防护措施安全操作电气部件（请参见“操作电气部件 (页 41)”部分）

2.1.10.3 模块的外部电源

即使设备断开主电源，外部供电的模块也可继续带电。接触带电部件可能触电，导致重伤甚至死亡。

对于继电器模块等特殊 SIMATIC 模块，可连接或切换超出 24 V DC 的电压，例如 230 V AC。在某些应用中，这些模块由系统外电源（外部电源）供电。如果使用外部电源，即使系统断开连接，这些模块上仍可能存有危险电压。

维护

开始操作前，需确保遵循以下规则：

- 设备完全断电。
- 所有外部供电的设备均已断电

2.1.10.4 由于存在允许的电位差，触摸会有触电危险

由于存在允许的电位差，模块输入端可能存在危险电压。接触带电部件可能触电，导致重伤甚至死亡。

模拟量输入模块等特殊 SIMATIC 模块设有隔离通道。此时，传感器电位和模块电位之间没有连接。如果传感器上存在允许电位差范围内的危险电压，则模块测量通道上也可能存在该电压。

有关允许电位差的信息，请参见模块产品手册的“技术规格”部分。

维护

开始操作前，需确保遵循以下规则：

- 设备完全断电。
- 所有外部供电的设备均已断电

2.1.10.5 导电污染物

通过导电污染物向人体传导电能的过程中可能发生触电，造成危及生命的伤害（死亡）。

- 在开关柜中安装设备。
- 确保开关柜上没有导电污染物。

2.1.10.6 过热

设备和线路过热可能引发冒烟起火，致使烧伤，甚至危及生命（死亡）。

要避免过热，应遵循以下原则：

- 确保安装位置正确。
- 确保空气供应充足（例如，不要遮盖通风槽，保持安装间隙）。
- 只能使用未损坏的电缆。

安装和连接

- 遵循安装位置说明。
- 遵循指定通风间隙。
- 根据电缆横截面熔断连接电缆。

维护

- 定期检查插入式连接和电缆是否损坏。

2.1.10.7 不受控运动

受驱动的机器或系统部件在调试、运行、维护和维修期间不受控运动。

- 遵循功能手册中所述的安全说明。

2.1.10.8 不安全的运行状态

不安全的运行状态可能导致未知程度的人身伤害。
触发因素如下：

- 软件遭到恶意操控，例如遭到病毒、木马或蠕虫攻击。

软件遭到恶意操控，例如遭到病毒、木马或蠕虫攻击。

- 采取相应防护措施，防止软件遭到篡改（请参见“工业网络安全”部分）。
- 及时安装可用更新。
- 通过相应防护措施（例如，病毒扫描程序）保护可移动介质上存储的文件，防止遭受恶意软件攻击。
- 设置 CPU 的访问保护。

2.1.11 材料损坏

2.1.11.1 运输与储存

- 只能使用产品原包装或其它适当材料（例如，导电泡沫橡胶或铝膜）包装、储存、运输电子元件、模块或设备。
- 遵循运输和储存期间的限制。请参见“技术规格”部分。

2.1.11.2 安装和连接

- 我们建议，只有元件、模块和设备通过以下某种措施接地时，才可触摸。
  - 戴上 ESD 腕带。
  - 在地面导电的 ESD 区域，穿上 ESD 鞋或戴上 ESD 接地腕带。
- 只能将电子元件、模块和设备置于导电表面（例如，带有 ESD 涂层的桌子、导电 ESD 塑料泡沫、ESD 包装袋和 ESD 运输容器）。
- 确保有充足的过电压保护。
- 通电时，请勿安装/拆卸以下元件：
  - 前连接器
  - I/O 模块
  - CPU

2.2 工业网络安全

2.2.1 网络安全信息

西门子为其产品及解决方案提供了工业网络安全功能，以支持工厂、系统、机器和网络的安全运行。

为了防止工厂、系统、机器和网络受到网络攻击，需要实施并持续维护先进且全面的工业网络安全保护机制。西门子的产品和解决方案构成此类概念的其中一个要素。

客户负责防止其工厂、系统、机器和网络受到未经授权的访问。只有在有必要连接时并仅在采取适当安全措施（例如，防火墙和/或网络分段）的情况下，才能将该等系统、机器和组件连接到企业网络或 互联网。关于可采取的工业网络安全措施的更多信息，请访问 https://www.siemens.com/cybersecurity-industry 。

西门子 不断对产品和解决方案进行开发和完善以提高安全性。西门子 强烈建议您及时更新产品并始终使用最新产品版本。如果使用的产品版本不再受支持，或者未能应用最新的更新程序，客户遭受网络攻击的风险会增加。

要及时了解有关产品更新的信息，请订阅西门子工业网络安全 RSS 源，网址为 https://www.siemens.com/cert 。

2.2.2 安全说明

免责声明

请注意，以下建议的风险最小化安全措施列表并非详尽无遗。因此，请咨询您的安全专家以进行最终评估和组态。此外，“工业安全 (https://www.siemens.com/industrialsecurity)”中已有相应介绍。

请注意 (i) 您有责任防止未经授权访问您的工厂、系统、机器和网络，以及 (ii) 只有在有必要连接时并仅在采取适当安全措施（例如，防火墙和/或网络分段）的情况下，才能将此类系统、机器和组件连接到企业网络或 Internet。由于您全权负责整体的最先进安全概念的构思、实施和维护，以保护您的企业、工厂、系统、机器和网络（包括产品）免受网络威胁，因此您对因未采取或未充分采取安全措施而造成的任何损害负责。

数据保护

西门子遵守数据保护准则，尤其是数据最小化的相关要求（从设计着手保护隐私）。对于该产品，这表示：该产品不处理或保存任何个人信息，但仅处理或保存技术功能数据（如时间戳）。如果用户将此数据关联到其它数据（例如，轮班计划），或者，如果用户将个人信息保存在同一介质（例如，硬盘）上，并因此在此过程中建立身份识别信息，则用户必须确保符合有关数据保护的准则。

主机 PC 保护

用户负责保护自己的主机 PC，防止其受到未经授权的访问。

西门子建议采取以下措施来保护主机 PC：

- 采取适当措施保护对主机 PC 的网络访问。如果需要与主机 PC 进行远程网络连接，西门子建议采取适当措施（例如使用 VPN 连接）保护与 PC 的通信。
- 离开计算机时启用屏幕保护程序并锁定屏幕。
- 在主机 PC 上安装兼容的防病毒软件。请确保病毒扫描程序及其数据库始终为最新版本。
- 在主机 PC 上安装操作系统和软件的更新和补丁，以确保其为最新版本。
- 为了保持适当的安全等级，西门子强烈建议禁用未使用的接口，如 USB 端口或以太网端口。

说明

U 盘使用说明

- 只能使用专用的 U 盘
- 使用 U 盘前对其进行恶意软件扫描
- 只能在需要使用时插入 U 盘

管理员帐户保护说明

具有管理员权限的用户在系统中有广泛的访问权限，可进行多种操作。

因此，请确保有足够的防护措施来保护管理员帐户，以防止未经授权的更改。为此，请使用安全密码和标准用户帐户进行正常操作。应根据需要采取其它措施，例如使用安全策略。

注意
防止未授权用户的访问 未授权用户可能会错误地操作 PC。未授权用户的操作行为会影响运行可靠性。 通过 Windows 用户帐户限制用户权限。

使用说明

说明

为防止 STEP 7-Micro/WIN SMART V3 和 Smart Web Editor 在 PC 遭受来自 Internet 的恶意攻击时被意外操控，西门子强烈建议在 PC 上安装允许列表工具。然后使用此工具管理 PC 上安装的软件。

说明

确保仅使用从官方渠道获取且经合法授权的软件安装程序，例如从官方销售渠道购买或从西门子官网下载的 USB 安装程序。

说明

为了避免密码泄露，西门子建议定期更改 PLC 中使用的密码。

2.2.3 安全概览

概览

本章介绍 S7-200 SMART 上的以下安全功能：

安全功能	说明
网络安全	为了保护网络通信，S7-200 SMART 3.0（及更高版本）CPU 使用 HTTPS 进行以下连接： • Web API • Web 浏览器 如果网络中需要进行身份验证、加密或完整性保护，西门子建议采取适当措施保护网络以及对 S7-200 SMART 设备的物理访问。
CPU 访问保护	可通过 CPU 保护方式，防止对 CPU 的未经授权访问。 有关更多信息，请参见“组态 PLC 安全 (页 57)”。
通信协议和端口 (页 51)	本节概述了通过 PN/IE 接口通信时所支持的协议以及使用的端口号。
SNMP 激活和禁用	在组态 PLC 安全 (页 57)中，可启用 SNMP。
POU 访问保护	POU 保护方式有助于防止 POU 泄露。 有关更多信息，请参见“POU/数据页面密码保护 (页 69)”。

安全功能	说明
项目访问保护	项目访问保护方式有助于防止带密码的 STEP 7-Micro/WIN SMART V3 项目遭受非法访问。 有关更多信息，请参见“设置项目密码 (页 68)”。
证书	证书管理向导用于组态 Web 服务器的 HTTPS 证书。 有关更多信息，请参见“下载并安装 CA 证书 (页 245)”。

说明

在 STEP 7-Micro/WIN SMART V3 中更改组态后，必须将新组态下载到 CPU 3.0 中，组态才能生效。

警告

防止对设备进行未经授权的访问

未经授权的用户可能会错误地操作设备、读/写数据以及通过重新启动设备绕过登录过程。

未经授权的人员所执行的操作会影响运行可靠性。

需通过限制物理访问，对这些形式的通信进行保护。西门子建议将 S7-200 SMART 设备锁在机柜中。

说明

S7-200 SMART 设备的通信协议适用于可信环境，允许对设备进行未经授权的访问。因此，西门子强烈建议采取相应措施保护对 S7-200 SMART 设备的网络访问。例如，将该设备锁在机柜中。

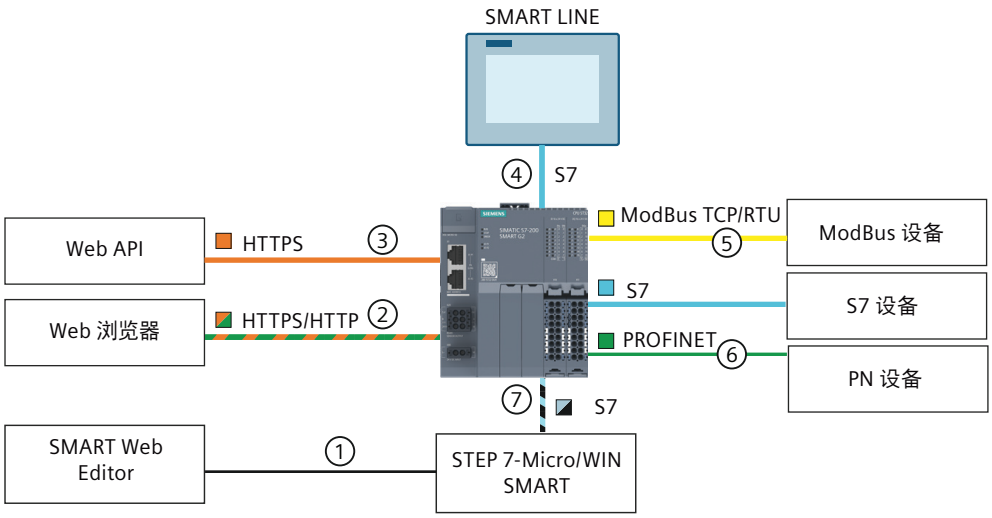
参见

对通信进行组态 (页 55)

2.2.4 网络安全

为 S7-200 SMART 3.0（及更高版本）设备增强网络安全措施后，可通过本地局域网或远程局域网访问 S7-200 SMART CPU 3.0（及更高版本）。

不同的设备/软件使用不同的通道连接到 S7-200 SMART CPU。



连接	支持的 APP/设备	端口	协议	备注
①	SMART Web Editor			虚拟离线数据传输通道
②	Web 浏览器	TCP 443 TCP 80	HTTPS/HTTP	HTTP over TCP 端口 80 仅用于开源组件声明。
③	Web API	TCP 443	HTTPS	
④	SMART LINE S7 设备	TCP 102	S7	
⑤	Modbus 设备	TCP 502	Modbus TCP/RTU	默认端口为 TCP 502，可根据实际业务需要设置要使用的端口。
⑥	PN 设备	UDP 34962/34963/34964	PROFINET	PROFINET 的 UDP 端口始终打开，不能禁用。
⑦	STEP 7-Micro/WIN SMART V3	TCP 102	S7	

为了保持适当的安全等级，西门子强烈建议只在安全网络中通过防火墙打开端口。上表列出了 S7200 SMART 支持的应用程序的所有端口信息。

⚠ 警告

通过 Web 服务器对 S7-200 SMART CPU 进行未经授权的访问

未经授权访问 S7-200 SMART CPU 或将 S7-200 SMART 变量更改为无效值可能会中断过程操作并可能导致死亡、严重人身伤害和/或财产损失。由于启用 Web 服务器后授权用户可执行操作模式更改、写入 S7-200 SMART 数据以及系统时间等操作，西门子建议遵照以下安全实践：

- 使用强密码保护 PLC 用户 ID。保管好密码并经常更改密码。
- 对程序逻辑中的变量执行错误检查和范围检查，因为用户可将 PLC 变量更改为无效值。
- 使用 HTTP 协议时，请使用安全的虚拟专用网络 (VPN) 连接 SMART Web 服务器。

⚠ 警告

通过非安全通道（对于 Modbus，为 TCP 502；对于 S7，为 TCP 102；对于 HTTP，为 TCP 80；对于 PROFINET，为 UDP 34962/34963/34964）对 S7-200 SMART 进行未经授权的访问

未经授权访问 S7-200 SMART 设备或将 S7-200 SMART 变量更改为无效值可能会中断过程操作并可能导致严重人身伤害和/或财产损失。

由于启用非安全端口后，未经授权的用户可执行操作模式更改、写入 S7-200 SMART 数据以及固件更新等操作，西门子建议只在安全网络中通过防火墙打开端口。

2.2.5 进行以太网通信时使用的通信协议和端口号

本节概述了通过 PN/IE 接口通信时所支持的协议以及使用的端口号。每个协议都会指定地址参数、相应通信层以及通信角色和通信方向。

有了这些信息，便可根据所用协议为自动化系统匹配合适的安全防护措施（例如防火墙）。由于安全措施限制为以太网或 PROFINET 网络，表格不包含 PROFIBUS 协议。

说明

使用的端口号

指定的端口号是 S7-200 SMART CPU 使用的标准端口号。很多通信协议和实现都支持使用其它端口号。

以下表格列出了 S7-200 SMART CPU 和 S7-200 SMART 通信模块中使用的不同层和协议。

S7-200 SMART CPU 和软件控制器的层和协议（通过 CPU 的 PROFINET 接口实现）

下表列出了 S7-200 SMART CPU 支持的协议。S7-200 SMART 软件控制器分配到的以太网接口同样支持下表中列出的协议。

表格 2-1　S7-200 SMART CPU 和软件控制器的层和协议（通过 CPU 的 PROFINET 接口实现）

协议/角色	端口号	(2) 链路层 (4) 传输层	说明/功能	默认设置/说明
PROFINET 协议				
DCP	不相关	(2) Ethertype 0x8892 (PROFINET)	PROFINET Discovery and Basic Configuration Protocol. DCP 用于确定 PROFINET 设备并启用基本设置。	默认设置：最高启用 FW 版本 V3.0。
LLDP	不相关	(2) Ethertype 0x88CC (LLDP)	PROFINET Link Layer Discovery Protocol. LLDP 用于确定和管理 PROFINET 设备之间的邻接关系。	默认设置：激活。 可通过 CPU 属性中的 Boundary “结束拓扑发现”(End of topology discovery) 禁用发送功能；准备接收功能保持激活状态。 LLDP 使用特殊多播 MAC 地址：01-80-C2-00-00-0E.
PROFINET IO 数据	不相关	(2) Ethertype 0x8892 (PROFINET)	PROFINET Cyclic IO Data Transfer IO 数据以 PROFINET IO 报文形式通过以太网在 PROFINET IO 控制器和 IO 设备之间循环传输。	默认设置：禁用。 只有出现 PROFINET IO 数据流量时，才激活该协议。
PROFINET 上下文管理器	34964	(4) UDP	无 RPC 的 PROFINET 连接。 管理 IO 控制器和 IO 设备之间的应用和通信关系。	默认设置：启用（UDP 端口打开）。 该功能无法禁用。
面向连接的通信协议				
HTTP 服务器	80	(4) TCP	Hypertext Transfer Protocol. HTTPS 用于与 CPU 内部 Web 服务器进行通信。	默认设置：激活。

协议/角色	端口号	(2) 链路层 (4) 传输层	说明/功能	默认设置/说明
HTTPS 服务器	443	(4) TCP	Hypertext Transfer Protocol Secure. HTTPS 用于通过安全套接层 (SSL) 与 CPU 内部 Web 服务器进行通信。	默认设置：禁用。 可在 CPU 属性中启用。 要求：已在 CPU 属性中启用 Web 服务器。
MODBUS TCP 服务器/客户端	502	(4) TCP	MODBUS Transmission Control Protocol. MODBUS/TCP 供用户程序中的 MB_CLIENT/MB_SERVER 指令使用。	默认设置：禁用。 可通过用户程序中的 Modbus 指令激活。
OUC[1]	1 ... 1999 可在有限程度上使用 [2]	(4) TCP (4) UDP (4) ISO-on-TCP（端口：102）	Open User Communication (TCP/UDP)。 借助 OUC 指令，可通过用户程序建立连接、终止连接和传输数据。	默认设置：禁用。 在用户程序中使用相应开放式用户通信指令或在网络视图中组态连接，可激活相应协议。
	2000 ... 5000 推荐			
	5001 ... 49151 可在有限程度上使用 [2]			
SNMP 代理	161 162（陷阱）	(4) UDP	Simple Network Management Protocol. SNMP 用于供 SNMP 管理器读取和设置网络管理数据（SNMP 管理对象）。	默认设置：禁用。 可通过系统块中的组态激活。

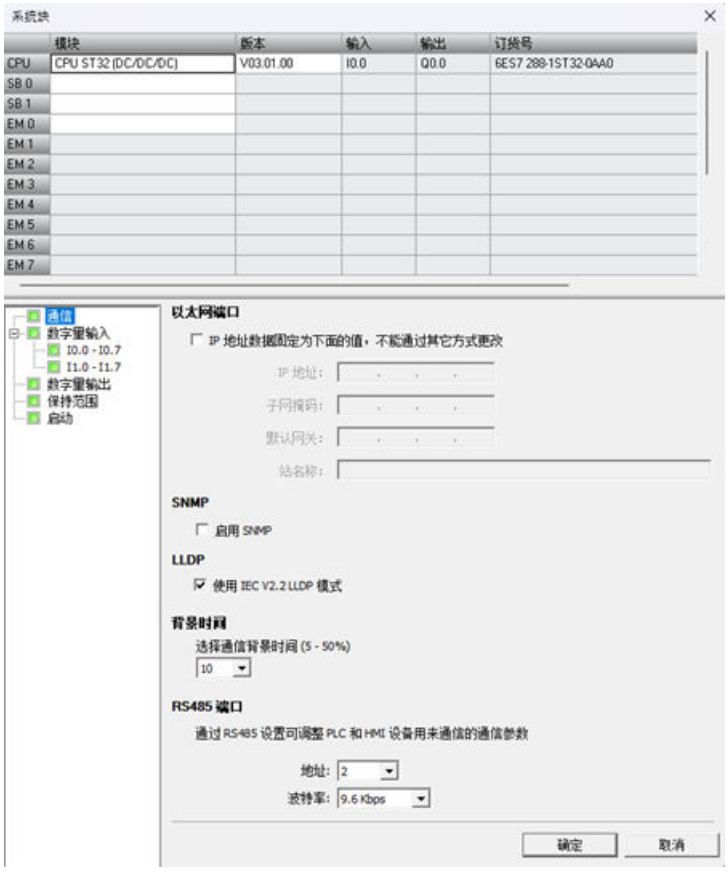
协议/角色	端口号	(2) 链路层 (4) 传输层	说明/功能	默认设置/说明
ISO-on-TCP 服务器	102	4) TCP	S7 protocol（遵循 RFC 1006）。S7 协议使用 ISO-on-TCP 遵循 RFC 1006 与 STEP 7-Micro/WIN SMART V3 进行 PG/HMI 通信。	默认设置：激活。 该功能无法禁用。
Put/Get 服务器	102	4) TCP	Put/Get 服务器，使用 S7 协议与 Put/Get 客户端进行通信。	默认设置：禁用。 可以在通信设置 (页 67)中启用 Put/Get。新组态启用后，必须下载到 PLC 中，端口才能起作用。

[1] 注：借助 OUC（开放式通信），可直接访问 UDP 和 TCP 协议。必须考虑 IANA (Internet Assigned Numbers Authority) 端口限制和定义。

[2] 请勿为 OUC 使用已被其它协议使用的端口。

2.2.6 对通信进行组态

单击“系统块 (页 211)”(System Block) 对话框的“通信”(Communication) 节点组态以太网端口、SNMP、背景时间和 RS485 端口。



以太网端口

若要使 CPU 从项目中获取其以太网网络端口的相关信息，则请单击“IP 地址数据固定为下面的值，不能通过其它方式更改”(IP address data is fixed to the values below and cannot be changed by other means) 复选框。然后便可输入以下以太网信息：

- “IP 地址”(IP Address)：每个设备必须有一个 Internet 协议 (IP) 地址。设备使用此地址在更加复杂的路由网络中传送数据。
- “子网掩码”(Subnet Mask)：子网是已连接的网络设备的逻辑分组。在局域网 (LAN) 中，子网中的节点彼此之间的物理位置通常相对接近。子网掩码定义 IP 子网的边界。子网掩码 255.255.255.0 通常适用于本地网络。

- “默认网关”(Default Gateway)：网关（或 IP 路由器）是 LAN 之间的链路。LAN 中的计算机可使用网关向其他网络发送消息，这些网络可能还隐含着其他 LAN。如果数据的目的地不在 LAN 内，网关会将数据转发给可将数据传送到其目的地的另一个网络或网络组。网关依靠 IP 地址来传送和接收数据包。
- “站名称”(Station Name)：站名称是在网络上定义的 CPU 名称。在“通信”(Communications) 对话框中，请使用有助于识别 CPU 的名称。

说明

站名称遵守标准 DNS（域名系统）命名规范。S7-200 SMART CPU 将站名称限制为最多 63 个字符，其中包括小写字母 a 到 z、数字 0 到 9、连字符（减号）和句号。

CPU 禁用某些名称：

- 站名称不能有 n.n.n.n 格式，其中 n 取 0 到 999 中的值。
- 站名不能以字符串 port-nnn 或字符串 port-nnn-nnnnn 开始，其中 n 是 0 到 9 的数字。例如，port-123 和 port-123-45678 为无效站名。站名称不能以连字符或句号开始或结束。

SNMP

如果要使用 SNMP 对网络设备进行监视和管理，请勾选“启用 SNMP”(Enable SNMP) 前的复选框，启用 SNMP。

使用 IEC V2.2 LLDP 模式

支持 IEC V2.3 模式的设备的默认值：

- “使用 IEC V2.2 LLDP 模式”(Use IEC V2.2 LLDP mode) 选项已禁用（无复选标记）。PROFINET 接口设置为 IECV2.3 模式 (IEC 61158 V2.3)。
- 选择此选项以将 PROFINET 接口设置为 IEC V2 2 模式。将系统拆分为多个相关项目时（例如，在智能设备组态的情况下），可能需要进行此设置。

不支持 IEC V2.3 模式的设备的默认值：

- “使用 IEC V2.2 LLDP 模式”(Use IEC V2.2 LLDP mode) 选项已选中（有复选标记）且无法更改。
- 如果当前项目中所有带 PROFINET 接口的设备都已完整组态，且参数已分配，则保留默认设置。

背景时间

可组态专门用于处理通信请求的扫描周期时间百分比。增加专门用于处理通信请求的时间百分比时，亦会增加扫描时间，从而减慢控制过程的运行速度。扫描时间仅在过程通信请求需要处理时增加。

专门用于处理通信请求的默认扫描时间百分比被设为 10%。该设置在处理编译/状态监控操作和尽量减小对控制过程的影响之间进行了合理的折衷。您可以调整该设置，每次增加 5%，最大为 50%。

随着 S7-200 SMART CPU 通信伙伴的增多，将需要更多的后台时间来处理这些伙伴的请求。GET 和 PUT 指令需要额外资源来创建并保持与其他设备间的连接。如果有 HMI 设备或其他的 CPU 通过 EM DP01 与 S7-200 SMART CPU 通信，则 EM DP01 PROFIBUS DP 模块需要额外的后台通信时间。开放式用户通信 (OUC) 还会给 CPU 增加额外负荷，并可能需要额外的后台时间。

RS485 端口

使用以下设置对板载 RS485 端口调整系统协议通信参数。连接编程设备或 HMI 设备时使用系统协议：

- RS485 端口地址：单击滚动按钮输入所需 CPU 地址 (1-126)。默认端口地址为 2。
- 波特率：从下拉列表中选择所需数据波特率（9.6 Kbps、19.2 Kbps 或 187.5 Kbps）。

说明

对于 S7-200 SMART CPU，可执行以下 RS485 通信连接：

- 使用 USB-PPI 电缆通过任意串行端口（包括 RS485 端口、信号板端口和 DP01 PROFIBUS 端口）对所有型号的 CPU 进行编程。
- 使用 RS485 和 RS232 端口进行 HMI 访问（数据读/写）和自由端口通信。

说明

S7-200 SMART G2 CPU 不支持直接连接 USB-PPI 线缆。若需使用此线缆连接 S7-200 SMART G2 CPU，则需要自行制作 RS485 连接线。

2.2.7 CPU 访问保护

西门子强烈建议使用密码保护功能来防止对 CPU 进行未经授权的操作。

关于设置和更改 CPU 密码的更多信息，请参见章节“组态 PLC 安全 (页 57)”。

2.2.8 组态 PLC 安全

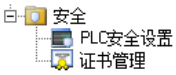
为提高 PLC 的安全性和管理密码，STEP 7-Micro/WIN SMART V3 提供了“PLC 安全设置”(PLC Security Setting) 面板。

打开“PLC 安全设置”(PLC Security Setting) 面板

采用以下任意一种方式打开“PLC 安全设置”(PLC Security Setting) 面板：

- 在“工具”(Tools) 菜单功能区的“安全”(Settings) 区域，单击“PLC 安全设置”(PLC security setting) 按钮。

- 打开项目树中的“安全”(Security) 文件夹；选择“PLC 安全设置”(PLC Security Setting) 节点，然后按 Enter；或者双击“PLC 安全设置”(PLC Security Setting) 节点。



设置 PLC 访问控制

“PLC 安全设置”(PLC Security Setting) 面板提供以下用户访问控制和密码管理选项：

- 禁用 PLC 访问控制
- 启用 PLC 访问控制
  - 启用 PLC 保护
  - 启用用户管理

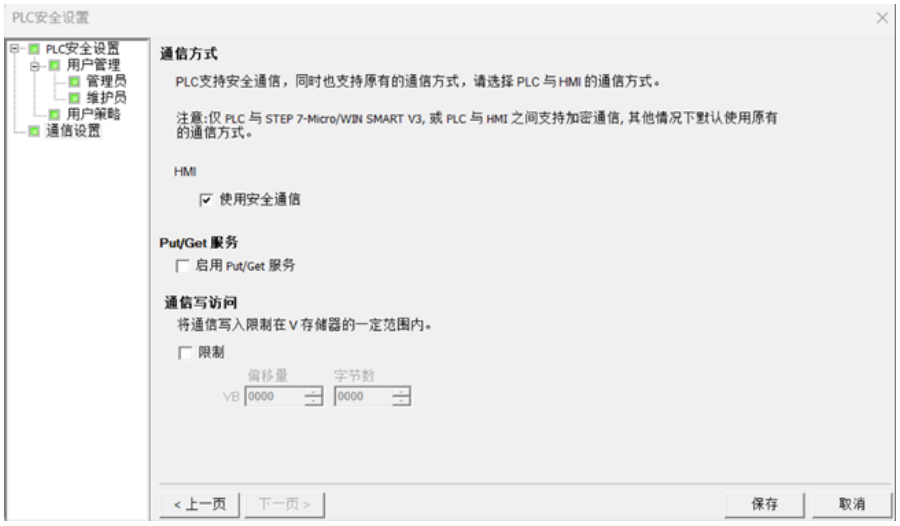
S7-200 SMART 的默认和建议条件为“启用用户管理”(Enable User Management)。

设置通信模式

S7-200 SMART 与 STEP 7-Micro/WIN SMART V3 和 HMI 进行通信时，可启用安全通信。

说明

加密通信仅适用于 PLC 与 STEP 7-Micro/WIN SMART V3 之间，或 PLC 与 HMI 之间。



2.2.8.1 PLC 保护（原有）

密码保护权限级别

CPU 提供四级密码保护，“禁用 PLC 访问控制”（1 级）提供无限制访问，“不允许上传”（4 级）提供最受限制的访问。

CPU 密码授权访问 CPU 功能和存储器。未下载 CPU 密码（“禁用 PLC 访问控制”（1 级））情况下，S7-200 SMART CPU 允许无限制访问。如果已组态比“禁用 PLC 访问控制”（1 级）级别更高的访问权限并下载 CPU 密码，则 S7-200 SMART CPU 要求输入密码以访问下表定义的 CPU 操作。

即使密码已知，“不允许上传”（4 级）密码限制也对用户程序（知识产权）进行保护。4 级权限无法实现上传，只有在 CPU 没有用户程序时才能更改权限级别。因此，即使有人发现密码，您也始终能够保护用户程序。

表格 2-2　S7-200 SMART CPU 密码保护权限级别

操作说明	禁用 PLC 访问控制（1 级）	读取权限（2 级）	最低权限（3 级）	不允许上传（4 级）
读取和写入用户数据	允许	允许	允许	允许
CPU 的运行、停止以及热重启	允许	有限制	有限制	有限制
读取日时钟	允许	允许	允许	允许
写入日时钟	允许	有限制	有限制	有限制
上传用户程序、数据和 CPU 组态	允许	允许	有限制	不允许
下载项目	允许	有限制	有限制	有限制
复位为出厂默认值	允许	有限制	有限制	有限制
删除项目	允许	有限制	有限制	有限制
将程序块、数据块或系统数据块复制到存储卡	允许	有限制	有限制	有限制
强制和取消强制	允许	有限制	有限制	有限制
执行单次或多次扫描操作。	允许	有限制	有限制	有限制
在 STOP 模式下写入输出。	允许	有限制	有限制	有限制
复位 PLC 信息中的扫描速率	允许	有限制	有限制	有限制
程序状态	允许	有限制	有限制	不允许
项目比较	允许	允许	有限制	不允许
下载轨迹配置	允许	有限制	有限制	有限制
删除轨迹配置	允许	有限制	有限制	有限制
上传轨迹	允许	允许	有限制	有限制
激活/停用 轨迹	允许	有限制	有限制	有限制
轨迹状态轮询	允许	允许	允许	允许

串行端口模式更改和日时钟 (TOD) 写入

无需密码也可通过串行端口（如果 CPU 型号支持，内置 RS485 和 RS485/RS232 信号板都可使用）允许 CPU 模式更改（go-to-RUN 和 go-to-STOP）和 TOD 写入。为此，在“串行端口”部分选中“允许”复选框。

此复选框可向下兼容不提示这些功能的密码的旧版 HMI。下列选项可用：

- 如果已选中此复选框且 CPU 受密码保护，则可使用这些旧版 HMI 更改操作模式和进行 TOD 写入。
- 如果未选中此复选框且 CPU 受密码保护，无法使用这些旧版 HMI 更改操作模式和进行 TOD 写入。
- 如果 CPU 不受密码保护，无论是否选中复选框，都可使用这些旧版 HMI 更改操作模式和进行 TOD 写入。

访问受密码保护的 CPU

说明

始终在断开电缆之前退出 STEP 7-Micro/WIN SMART，以防另一位用户未经授权擅自访问。

通过网络输入密码并不影响 S7-200 SMART CPU 的密码保护。如果一位授权用户通过网络访问受限功能，则不授权其他用户访问这些功能。在某一时刻，只允许一位用户无限制访问 S7-200 SMART CPU。

禁用密码

可通过选择“禁用 PLC 访问控制”(Disable PLC Access Control) 禁用密码，因为其允许所有不受限制的 CPU 访问。

说明

如果权限级别为“不允许上传”，存在有效用户程序时无法通过新的密码级别下载新的系统块。必须首先删除该用户程序，然后才能下载更新的系统块。

忘记密码怎么办

如果忘记密码，必须将 PLC 复位为出厂默认状态。（有关详细信息，请参见清除 PLC 存储器。）

禁用 PLC 访问控制

可从 STEP 7-Micro/WIN SMART V3 向 S7-200 SMART 授予“完全权限”（1 级）。

要禁用访问控制，请按以下步骤操作：

1. 在“PLC 安全设置”(PLC Security Setting) 面板中选择“禁用 PLC 访问控制”(Disable PLC Access Control)。
2. 单击“**确定**”(OK)。

启用 PLC 保护

可组态 S7-200 SMART 的安全设置和密码。

要启用 PLC 保护，请按以下步骤操作：

1. 在“PLC 安全设置”(PLC Security Setting) 面板中选择“启用 PLC 访问控制 - 启用 PLC 保护（传统访问控制）”(Enable PLC Access Control - Enable PLC Protection (legacy access control))。然后单击“**下一步**”(Next)。
2. 设置 PLC 保护：
   – 选择受密码保护的权限级别，并设置密码。
   – 选择“允许”(Allow)，无需密码即可进行串行端口通信。然后单击“**下一步**”(Next)。

3. 无需密码也可通过串行端口（如果 CPU 型号支持，内置 RS485 和 RS485/RS232 信号板都可使用）允许 CPU 模式更改（go-to-RUN 和 go-to-STOP）和 TOD 写入。为此，在“串行端口”部分选中“允许”复选框。
   此复选框可向下兼容不提示这些功能的密码的旧版 HMI。下列选项可用：
   – 如果已选中此复选框且 CPU 受密码保护，则可使用这些旧版 HMI 更改操作模式和进行 TOD 写入。
   – 如果未选中此复选框且 CPU 受密码保护，无法使用这些旧版 HMI 更改操作模式和进行 TOD 写入。
   – 如果 CPU 不受密码保护，无论是否选中复选框，都可使用这些旧版 HMI 更改操作模式和进行 TOD 写入。
4. 设置通信模式 (页 67)：

2.2.8.2 启用用户管理

可组态 S7-200 SMART 的用户访问设置。

要启用用户管理，请按下列步骤操作：

1. 在“PLC 安全设置”(PLC Security Setting) 面板中选择“启用 PLC 访问控制 - 启用用户管理”(Enable PLC Access Control - Enable User Management)。然后单击“**下一步**”(Next)。
2. 检查“用户角色说明”(User Roles Description)。然后单击“**下一步**”(Next)。
3. 单击“**添加**”(Add) 或“**删除**”(Delete) 以组态管理员用户。然后单击“**下一步**”(Next)。
   注：表中有一个名为 Admin 的默认用户。Admin 的默认密码是 `Changeme_123`。
4. 单击“**添加**”(Add) 或“**删除**”(Delete) 以组态维护员用户。然后单击“**下一步**”(Next)。
5. 为管理员和维护员组态用户策略。然后单击“**下一步**”(Next)。
6. 组态“安全通信”(Secure Communication)，确保与 HMI 的通信安全。然后单击**“保存”(Save)**。

用户角色权限

用户可以管理员、维护员或访客的身份登录。每个用户角色都被授予不同的操作权限。

以不同用户角色登录会授权对 CPU 功能和存储器的访问。S7-200 SMART 要求登录以访问下表中定义的操作。

操作说明	管理员	维护员	访客
读用户数据	允许	允许	允许
读取 PLC 信息（PLC 信息和 PLC 日志）	允许	允许	允许
CPU 的运行、停止以及热重启	允许	允许	有限制
读取日时钟	允许	允许	允许
写入日时钟	允许	允许	有限制

操作说明	管理员	维护员	访客
上传用户程序、数据、数据日志和 CPU 组态	允许	允许	有限制
上传用户管理（安全组态）	允许	有限制	有限制
下载项目	允许	有限制	有限制
下载/上传用户管理块	允许	有限制	有限制
下载传输层安全 (TLS) 证书块	允许	有限制	有限制
复位为出厂默认值	允许	有限制	有限制
删除项目	允许	有限制	有限制
升级固件、用户自定义 Web 资源	允许	有限制	有限制
运动轴控制	允许	允许	有限制
存储卡	允许	有限制	有限制
强制和取消强制	允许	有限制	有限制
执行单次或多次扫描操作。	允许	允许	有限制
复位 PLC 信息中的扫描速率	允许	允许	有限制
程序状态	允许	允许	有限制
项目比较	允许	允许	有限制
读写 Web API 和状态图表	允许	允许	有限制
写入用户数据 (V.....)	允许	允许	有限制
更改当前用户密码	允许	允许	有限制
闪烁 LED	允许	允许	允许
下载轨迹配置	允许	有限制	有限制
删除轨迹配置	允许	有限制	有限制
上传轨迹	允许	允许	有限制
激活/停用 轨迹	允许	允许	有限制
轨迹状态轮询	允许	允许	有限制

管理员表

可在管理员表中管理和组态管理员访问权限。表中有一个名为 Admin 的默认用户。

- **名称**：双击可设置此用户角色的名称。用户名必须唯一。
- **密码**：双击可设置此用户角色的密码。

- **访问终端：**设为“全部”(ALL)，因为管理员用户拥有最高权限。
- **注释：**可为用户角色添加注释。不需要。

说明

- 强烈建议在收到设备后更改默认用户“Admin”的密码。
- 用户名必须以字母开头，并且只能包含字母、数字和下划线。长度必须在 5 到 20 个字符之间。
- 密码只能同时包含数字、小写和大写字符以及特殊字符。长度必须在 10 到 63 个字符之间。

维护员表

可在维护员表中管理和组态维护员访问权限。默认不添加维护员。

- **名称：**双击可设置此用户角色的名称。
- **密码：**双击可设置此用户角色的密码。
- **访问终端：**选择或取消选择维护员的访问权限。

- **注释：**可为用户角色添加注释。不需要。

说明

- 用户名必须以字母开头，并且只能包含字母、数字和下划线。长度必须在 5 到 20 个字符之间。
- 密码只能同时包含数字、小写和大写字符以及特殊字符。长度必须在 10 到 63 个字符之间。

用户策略

可同时为管理员和维护员组态用户访问策略，并进行应用。

- **允许访客向 PLC 写入用户数据**：即使用户未登录（访客模式），用户也可将数据写入 PLC 并修改系统时间。
- **允许同一用户从多个终端访问 PLC**：用户可从多个终端登录 PLC，如 STEP 7-Micro/WIN SMART V3、HMI 或 Web 服务器。
  - 如果未选择此选项，则同一用户只允许登录到一个终端。例如，如果用户登录到 STEP 7-Micro/WIN SMART V3，然后登录到 Web 服务器，STEP 7-Micro/WIN SMART V3 将自动退出。
  - “活动时间”(Active Time) 用于设置登录用户可以访问 PLC 的时长，当超时后，终端上登录的用户将被踢出 PLC。当“活动时间”机制在每个终端上独立运行时，定时器仅在操作 PLC 时复位（定时器不受程序在线状态的影响）。
- **不允许上传**：防止任何用户（管理员或维护员）上传数据/程序。

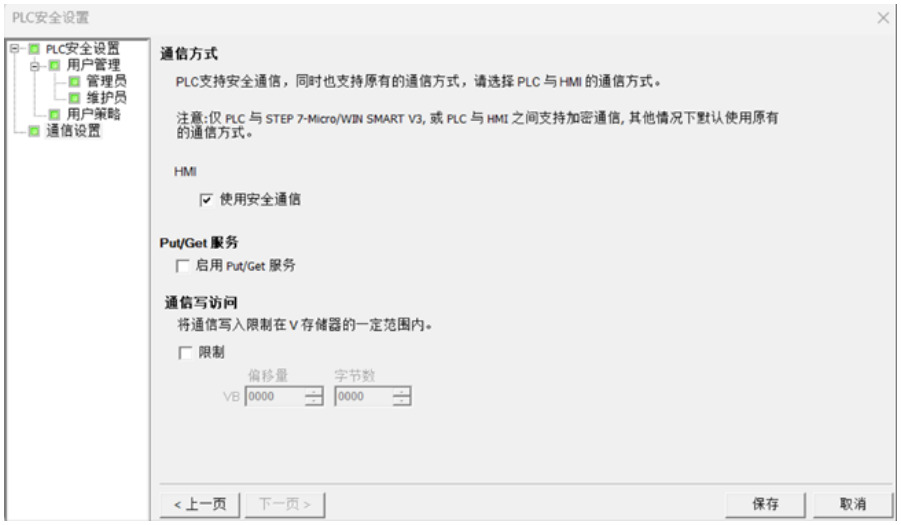
通信设置

设置通信设置 (页 67)

2.2.8.3 通信设置

在此页面中，可为 PLC 组态以下安全设置，以与其他设备进行通信：

- 选择“安全通信”(Secure Communication)，确保与 STEP 7-Micro/Win SMART V3 和 HMI 的通信安全。
- 选择“启用 Put/Get 服务器”(Enable Put/Get Server)，启用通过以太网连接在 S7-200 SMART CPU 之间进行的通信。
- 选择“限制”(Restrict) 以限制通信写入。然后单击“**保存**”(Save)。



Put/Get 服务器

可通过以太网连接启用或禁用 S7-200 SMART CPU 之间的 PLC 通信。

- 此 PLC 用作服务器时，如果启用此通信，则其可由其他客户端访问；如果禁用此通信，则其他客户端将无法进行访问。
- 此 PLC 用作客户端时，此设置不会影响其操作。

Put/Get 服务器默认禁用。

说明

启用 Put/Get 服务器功能会引入潜在的网络安全风险。请务必确保您局域网（LAN）环境的安全性。

通信写入限制

可对 V 存储器特定范围的通信写入进行限制，禁止对其他存储区进行通信写入（I、Q、AQ 和 M）。要对 V 存储器特定范围的通信写入进行限制，选中“限制”复选框，以字节为单位组态 V 存储器范围。

此区域可小到没有字节，大到整个 V 存储器。

使用此功能，用户程序可先验证写入此存储器子集的数据，然后再在应用程序中使用数据，以获得更好的安全性。请注意，这些限制只适用于通信写入（例如来自 HMI、STEP 7-Micro/WIN SMART、PC Access 和其他 CPU PUT 指令的写入），不适用于用户程序写入。

说明

如果限制对 V 存储器特定范围的写访问，确保“文本显示”模块或 HMI 只在 V 存储器的可写范围内写入。此外，如果使用 PID 向导、PID 控制面板、运动控制向导或运动控制面板，确保这些向导或面板使用的 V 存储器在其可写范围内。

禁用此项限制时，可写入存储区的全部范围，包括 I、Q、M、V、AQ 和 DB2。

2.2.9 保护项目

CPU 中的用户程序加密存储在文件系统中。在获得授权登录后，这些用户程序可以使用导入/上传/运行系统导出并验证。建议安全存储这些文件并注意安全风险。

说明

未加密的项目文件仅限具有客户管理权限的用户访问。

2.2.9.1 设置项目密码

要用密码保护整个项目，请按以下步骤操作：

1. 在“文件”(File) 菜单的“保护”(Protection) 部分中单击“项目”(Project) 按钮。

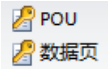
2. 选中“对此项目设置密码保护”(Password-protect this project) 复选框。
3. 输入并验证密码。

为项目设置密码可以防止未经授权的用户对 STEP 7-Micro/WIN SMART V3 项目的任何部分进行读写操作。

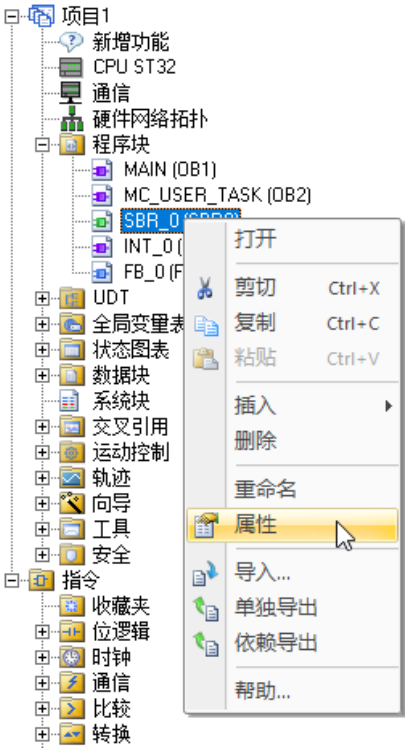
2.2.9.2 POU/数据页面密码保护

要访问 POU 或数据块的属性，请执行以下步骤之一：

- 在“视图”(View) 菜单功能区的“保护”(Protection) 区域，单击 POU 或“数据页”(Data Page) 按钮，以显示当前程序编辑器中的 POU 属性或数据块中的页。



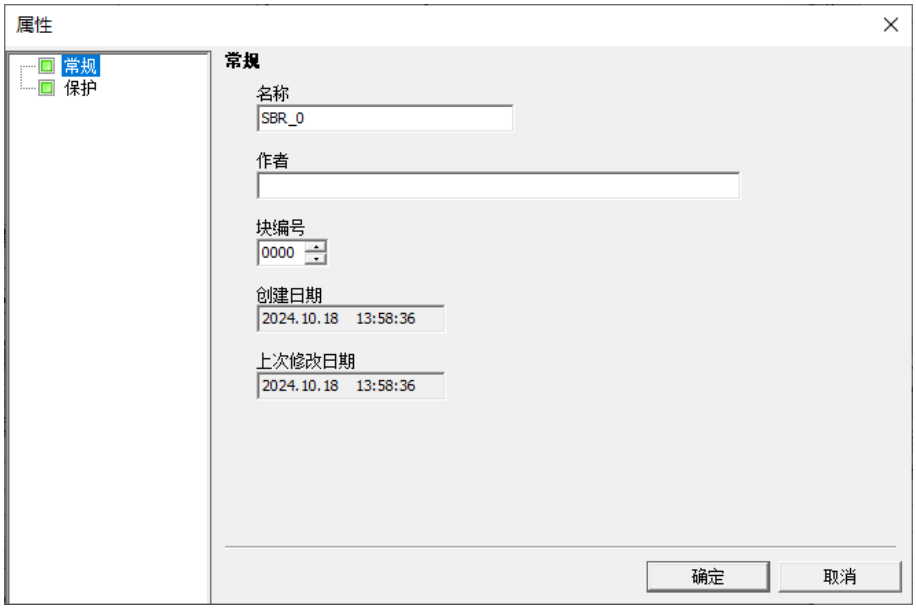
- 在项目树 (页 157)中右键单击 POU (页 153) 或数据块，然后从上下文菜单中选择**“属性”(Properties)**。



- 在编辑器窗口中右键单击 POU (页 153) 或数据块页面选项卡，然后单击“属性”(Properties) 按钮。

POU“属性”(Properties) 对话框包含两个页面：

- 常规 (General)
- 保护 (Protection)



POU 属性常规页面

“常规”(General) 页可用来重新命名主程序块、子程序、中断例程、功能块和数据块页以及指定它们的编号。也可为项目指定作者。

如果为 POU 或数据块指定全局变量名，则全局变量名会显示在用户的程序代码中，即使未启用 “符号寻址”(Symbolic Addressing) 视图也依然显示。全局变量表显示一个“POU 变量”(POU Variables) 选项卡，其中列出所有变量名分配。只能查看该选项卡，无法编辑全局变量表中的条目。要更改分配，必须编辑相应 POU 的属性。

说明

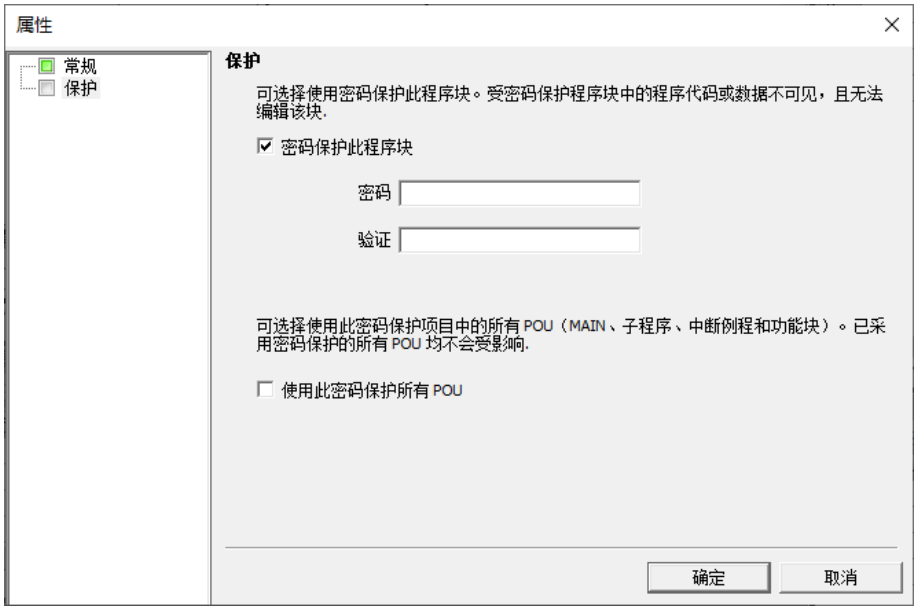
无法使用默认名称（STEP 7-Micro/WIN SMART V3 分配的 POU 地址，例如为子程序分配的 SBR_1、为中断程序分配的 INT_1 或者为功能块分配的 FB_1）作为另外一个 POU 的全局变量名，否则会导致重复分配。如果违反全局变量名分配的准则，在尝试编译程序时 STEP 7-Micro/WIN SMART V3 会报错。

POU 属性保护页面

“保护”(Protection) 页面用于为 POU 或数据块设置密码保护，使其对其他用户不可见并在下载时加密。必须为受保护的 POU 提供密码才能打开程序块以用于正常编辑、在 RUN 模式编辑和程序状态操作。

要加密保护 POU，请按以下步骤操作：

1. 右键单击“主 POU”(MAIN POU)，然后选择“属性”(Properties)。
2. 单击“保护”(Protection) 节点。
3. 选中“密码保护此程序块”(Password-protect this POU) 复选框。输入并验证密码。



4. 还可以选中“密码保护所有 POU”(Password-protect All POU) 复选框，以保护项目中的所有 POU（主程序、子程序、功能块和中断程序）。

说明

强密码安全性更高。强密码长度必须符合以下要求：

- 10-32 个字符之间
- 必须包含数字、大写字母和小写字母以及特殊字符 !@#$%^&*()-+~_={}[]|\:;,.<>
- 不能是可在字典上找到的词，
- 不能是可从个人信息中得到的名称或标识符。

保管好密码并经常更改密码。

说明

增强了 POU 和数据块（数据页）密码的安全性

与之前版本相比，STEP 7-Micro/WIN SMART V2.2 增强了密码的安全性。如果您正在使用之前版本 STEP 7-Micro/WIN SMART 创建的项目，则要重新输入密码才能激活增强的安全性。

POU 块加密

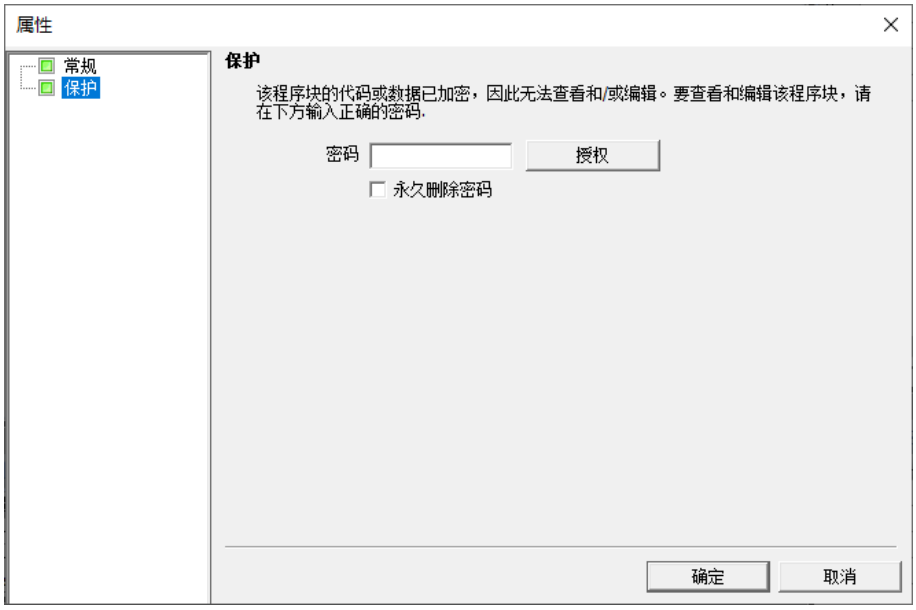
当 POU 或数据块受到保护时，其在项目树中显示为锁定图标（如下所示）：

解锁 POU

必须为受保护的 POU 提供密码才能打开块（用于正常编辑、在 RUN 模式编辑和程序状态操作）。

1. 打开 POU 属性，然后单击“保护”(Protection) 节点。
2. 输入密码。
3. 默认情况下，**“永久删除密码”(Permanently remove password)** 复选框未选中
   - **未选中：**如果您在未选中此框的情况下“授权”POU 进行编辑，则您将获得一个用于当前编辑会话的临时授权，但未来编辑会话的密码保护仍然有效。
   - **选中：**如果选中“永久删除密码”(Permanently remove password) 复选框：并“授权”POU，则将删除密码保护。
4. 单击“授权”(Authorize)。

对于您选择要解锁的每个受保护 POU，必须重复执行此过程。



另请参见

项目密码保护 (页 68)

设置项目密码 (页 68)

应用程序用户参考手册 (页 157)

入门指南 (页 68)

参见

组态 PLC 安全 (页 57)

入门指南 (页 75)

2.2.10 安全停用

2.2.10.1 停用简介

在本章节中，将介绍有关如何正确停用自动化系统中各组件的信息。当组件已达到使用寿命时，则需进行停用。

停用处理包括采用环境无害方式处置存储介质和安全移除存储介质和电子元件中的所有数字数据。

2.2.10.2 安全删除数据

在处置自动化系统组件前，应安全删除这些组件存储介质中的所有数据。下文中，将介绍如何从设备中安全删除数据，确保无法恢复。

注意
非安全删除数据导致的数据滥用 未安全删除数据存储器中的数据或删除不完全，可能导致这些数据被第三方滥用。 因此，在处置这些产品前，需确保所有存储介质中的数据已安全删除。

要从 S7-200 SMART CPU 的数据存储器中删除所有数据，可以按照以下链接中的步骤清除 PLC。该功能将删除模块内部保存的所有信息。

- 清除 PLC 存储区 (页 239)
- 创建复位为出厂默认存储卡 (页 241)

2.2.10.3 回收和处置

S7-200 SMART CPU 为低污染设备，可完全回收利用。为确保旧设备的回收和处置符合环保要求，请联系经认证的电子废料处置公司，并根据所在国家/地区的相关规定进行回收处置。

入门指南 3

借助 STEP 7-Micro/WIN SMART V3，可轻松对 CPU 进行编程。只需一个简单示例和几个简短步骤，即可学会用户程序的创建方法，然后可以下载该程序并在 CPU 中运行。

该示例需要以太网或 RS485 通信电缆、CPU 和运行 STEP 7-Micro/WIN SMART V3 编程软件的编程设备。

3.1 连接到 CPU

连接 CPU 十分容易。在本例中，只需将电源与 CPU 相连，然后通过以太网或通信电缆将编程设备与 CPU 相连。

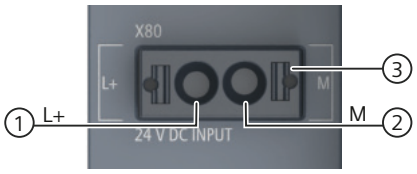
电源装置

只能使用具有安全电气隔离功能超低电压 (≤ 28.8 VDC) 的 SELV/PELV 型电源装置。

将电源连接到 CPU (X80)

> ⚠ **警告**
>
> **安装、接线或拆卸设备前，请确保电源关闭**
>
> 在安装或拆卸任何电气设备之前，请确保已切断该设备的电源。
>
> 如果在通电的情况下尝试安装 CPU 或相关设备或者对它们进行接线，则可能会触电或导致设备错误运行。如果在安装或拆卸过程中未切断 CPU 和相关设备的所有电源，则可能导致人员死亡、重伤或设备损坏。
>
> 在安装或拆卸 CPU 或相关设备之前，必须采取合适的安全预防措施并确保切断该 CPU 的电源。

下图列出了 24 V DC 电源电压的信号名称。



① 电源电压 +24 V DC

② 电源电压接地

③ 弹簧式脱扣器

有关最大导线横截面，请遵循接线规则 (页 102)表中的信息。

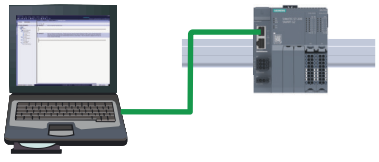
断开电线

使用螺丝刀，尽可能向下按压弹簧释放装置并拉出导线。

3.1.1 组态 CPU 以进行通信

3.1.1.1 概述

CPU 可通信网络中的 STEP 7-Micro/WIN SMART 编程设备进行通信：



CPU 可与以太网中的 STEP 7-Micro/WIN SMART 编程设备进行通信。

说明

- 支持在同一台计算机上安装 STEP 7-Micro/WIN SMART（2.8 版）和 STEP 7-Micro/WIN SMART V3。
- STEP 7-Micro/WIN SMART （2.8 版）能够连接到 2.8 及更早版本的 CPU，STEP 7-Micro/WIN SMART V3 只能连接到 3.0 或更高版本的 CPU。
- 只能同时将一个 STEP 7-Micro/WIN SMART 连接到一台 CPU。

在 CPU 和编程设备之间建立以太网通信时请考虑以下几点：

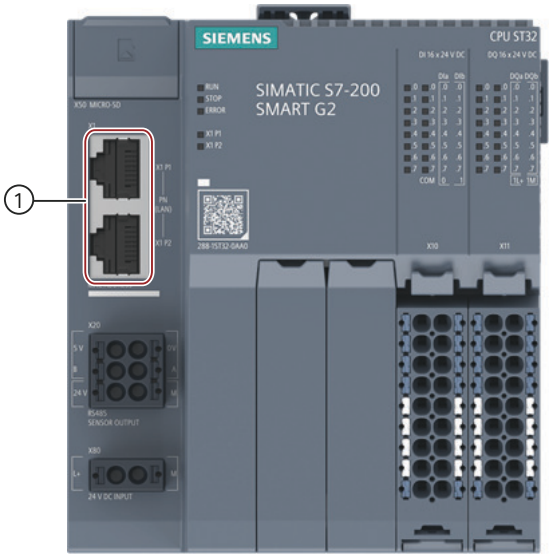
- 组态/设置：单个 CPU 不需要硬件配置。如果想要在同一个网络中安装多个 CPU，则必须将默认 IP 地址更改为新的唯一的 IP 地址。
- 双网络端口，多个端口可采用菊花链形式

3.1.1.2 建立以太网硬件通信连接

以太网接口可在编程设备和 CPU 之间建立物理连接。由于 CPU 内置了自动跨接功能，所以对该接口既可以使用标准以太网电缆，又可以使用跨接以太网电缆。将编程设备直接连接到 CPU 时不需要以太网交换机。

要在编程设备和 CPU 之间创建硬件连接，请按以下步骤操作：

1. 安装 CPU。
2. 将以太网电缆插入 CPU 左上方的以太网端口，如下图所示。
3. 将以太网电缆连接到编程设备上。



① PROFINET (LAN) 端口

3.1.1.3 与 CPU 建立以太网通信

在 STEP 7-Micro/WIN SMART V3 中，使用以下方法之一显示以太网“通信”(Communications) 对话框，从而组态与 CPU 的通信。

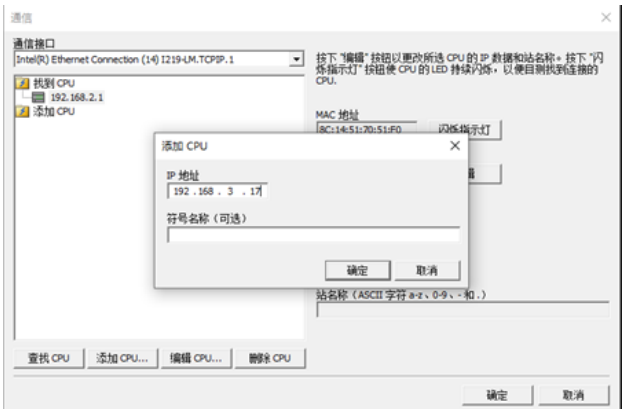
- 在项目树中，双击“通信”(Communications) 节点。
- 单击导航栏中的“通信”(Communications) 按钮 。
- 在“视图”(View) 菜单功能区的“窗口”(Windows) 区域内，从“组件”(Component) 下拉列表中选择“通信”(Communications)。

“通信”(Communication) 对话框提供了两种方法选择需要访问的 CPU：

- 单击“查找 CPU”(Find CPU) 按钮以使 STEP 7-Micro/WIN SMART V3 在本地网络中搜索 CPU。在网络上找到的各个 CPU 的 IP 地址将在“找到 CPU”(Found CPU) 下列出。
- 单击“添加 CPU”(Add CPU) 按钮以手动输入所要访问的 CPU 的访问信息（IP 地址等）。通过此方法手动添加的各 CPU 的 IP 地址将在“添加 CPU”(Added CPU) 中列出并保留。

说明

STEP 7-Micro/WIN SMART V3 只能搜索版本 3.0 或更高版本的 CPU。

	对于“已发现 CPU”（CPU 位于本地网络），可使用“通信”(Communications) 对话框与您的 CPU 建立连接： • 为通信接口选择 TCP/IP。 • 单击“查找 CPU”(Find CPU) 按钮，将显示本地以太网网络中所有可操作 CPU（“找到 CPU”）。所有 CPU 都有默认 IP 地址。请参见下方的“注”。 • 高亮显示 CPU，然后单击“确定”(OK)。
	对于“已添加 CPU”（CPU 位于本地网络或远程网络），可使用“通信”(Communications) 对话框与您的 CPU 建立连接： • 为通信接口选择 TCP/IP。 • 单击“添加 CPU”(Add CPU) 按钮，执行以下任意一项操作： – 输入编程设备可访问但不属于本地网络的 CPU 的 IP 地址。 – 直接输入位于本地网络中的 CPU 的 IP 地址。 所有 CPU 都有默认 IP 地址。请参见下方的“注”。 • 高亮显示 CPU，然后单击“确定”(OK)。
	与 CPU 建立通信之后，即可创建和下载示例程序。 要下载所有项目组件，在“文件”(File) 或 PLC 菜单功能区的“传输”(Transfer) 区域单击“下载”(Download) 按钮，也可按快捷键组合 CTRL+D。 如果 STEP 7-Micro/WIN SMART 未找到您的 CPU，请检查通信参数设置并重复以上步骤。

说明

CPU 列表将显示所有 CPU，而不管以太网网络类别和子网。

要建立与 CPU 的连接，通信接口（用于以太网和网络接口卡 (NIC)）和 CPU 的网络类别和子网必须相同。可以设置网络接口卡与 CPU 的默认 IP 地址匹配，也可以更改 CPU 的 IP 地址与网络接口卡的网络类别和子网匹配。

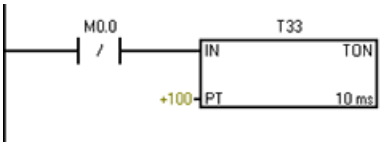
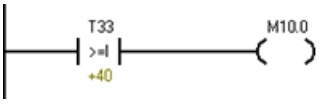
有关如何完成此任务的信息，请参见“为项目中的 CPU 或设备组态或更改 IP 地址”(页 516)。

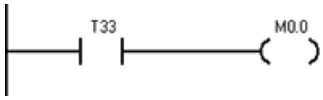
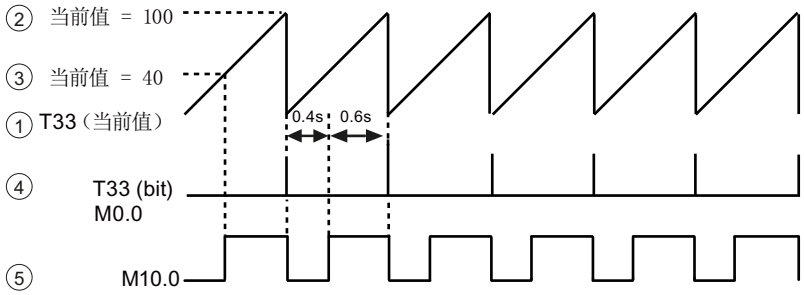
3.2 创建示例程序

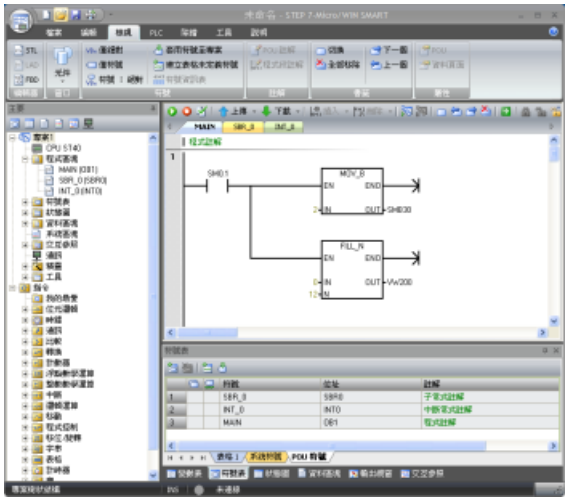
这个控制程序示例可帮助您理解使用 STEP 7-Micro/WIN SMART 有多容易。 该程序在三个程序段中使用 6 条指令创建了一个非常简单的自启动、自复位定时器。

在本例中，使用梯形图 (LAD) 编辑器输入程序指令。 下面的示例以 LAD 和语句表 (STL) 形式显示了整个程序。 描述列说明每个程序段的逻辑。 时序图显示了程序的运行。 STL 程序中没有程序段注释。

表格 3-1 STEP 7-Micro/WIN SMART 使用入门的示例程序

LAD/FBD	STL	说明
	**Network 1** **LDN M0.0** **TON T33, +100**	10 ms 定时器 T33 在 (100 x 10 ms = 1 s) 后超时 M0.0 脉冲速度过快，无法用状态视图监视。
	**Network 2** **LDW>= T33, +40** **= M10.0**	以状态视图可见的速率运行时，比较结果为真。在 (40 x 10 ms = 0.4 s) 之后，M10.0 接通，信号波形 40% 为低电平，60% 为高电平。

LAD/FBD	STL	说明
	**Network 3** **LD T33** **= M0.0**	T33（位）脉冲速度过快，无法用状态视图监视。 在 (100 x 10 ms = 1 s) 时间段之后，通过 M0.0 复位定时器。
		**时序图：** • ① T33（当前值） • ② 当前值 = 100 • ③ 当前值 = 40 • ④ T33（位）和 M0.0 • ⑤ M10.0



输入并保存程序之后，可以将程序下载到 CPU。

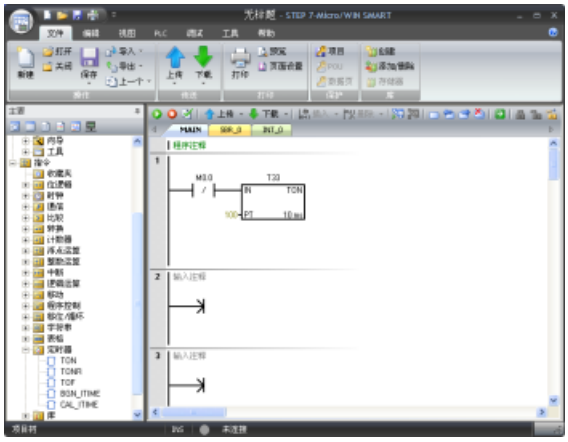
请注意项目树和程序编辑器。 通过项目树将指令插入到程序编辑器的程序段中，方法是将项目树“指令”(Instructions) 部分中的指令拖放到程序段中。

程序中的所有块均保存在项目树的程序块文件夹中。

程序编辑器工具栏中的图标提供 PLC 命令和编程操作的快捷方式。

3.2.1 程序段 1： 启动定时器

程序段 1：启动定时器



当 M0.0 处于断开状态 (0) 时，该触点接通并提供能流启动定时器。

要输入触点 M0.0：

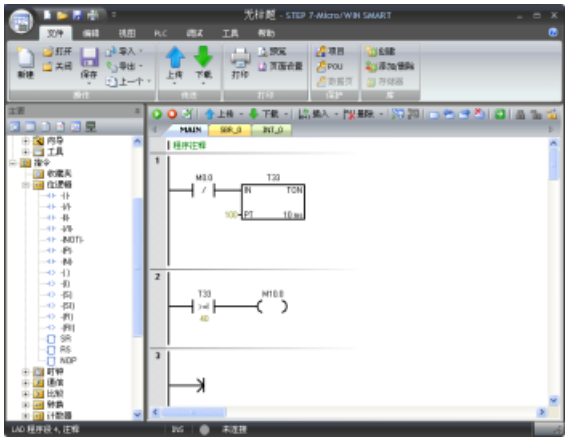
1. 双击“位逻辑”(Bit Logic) 图标或单击加号 (+) 以显示位逻辑指令。
2. 选择“常闭”触点。
3. 按住鼠标左键并将触点拖到第一个程序段中。
4. 为触点输入以下地址： M0.0
5. 按回车键即输入该触点地址。

要输入定时器指令 T33：

1. 双击“定时器”(Timers) 图标以显示定时器指令。
2. 选择“TON”（接通延时定时器）指令。
3. 按住鼠标左键并将定时器拖到第一个程序段中。
4. 为定时器输入以下定时器编号： T33
5. 按回车键即输入定时器编号，光标将移动到预设时间 (PT) 参数。
6. 为预设时间输入以下值： +100.
7. 按回车键即输入该值。

3.2.2 程序段 2： 接通输出

程序段 2： 接通输出



当T33 的定时器值大于或等于 40（40 * 10 毫秒，即 0.4 秒）时，该触点将提供能流接通 CPU 的输出 M10.0。

要输入比较指令：

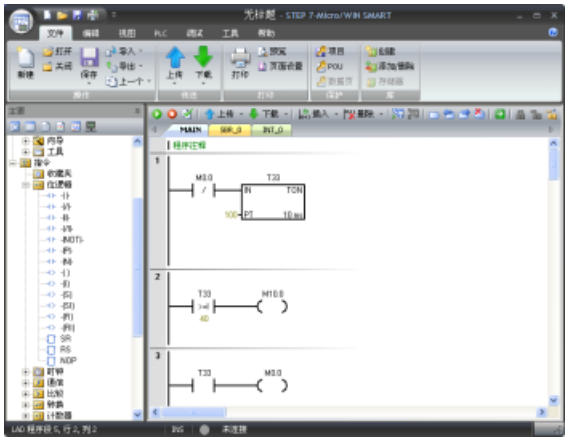
1. 双击“比较”(Compare) 图标以显示比较指令。 选择”>=I”指令（大于或等于整数）。
2. 按住鼠标左键并将比较指令拖到第二个程序段中。
3. 单击触点上方的“???”，然后输入以下定时器地址值： T33
4. 按回车键即输入定时器编号，光标将移动到将与定时器值进行比较的其它值。
5. 输入要与定时器数值比较的以下值： +40
6. 按回车键即输入该值。

要输入用于接通输出 M10.0 的指令：

1. 双击“位逻辑”(Bit Logic) 图标以显示位逻辑指令并选择输出线圈。
2. 按住鼠标左键并将线圈拖到第二个程序段中。
3. 单击线圈上方的“???”，然后输入以下地址： M10.0
4. 按回车键即输入该线圈地址。

3.2.3 程序段 3： 复位定时器

程序段 3： 复位定时器



定时器达到预设值 (100) 时，定时器位将接通，T33 的触点也将接通。 该触点的能流会接通 M0.0 存储单元。 由于定时器由常闭触点 M0.0 使能，所以 M0.0 的状态由断开 (0) 变为接通 (1) 将复位定时器。

要输入 T33 的定时器位触点：

1. 从位逻辑指令中选择“常开”触点。
2. 按住鼠标左键并将触点拖到第三个程序段中。
3. 单击触点上方的“???”，然后输入定时器位的地址： T33
4. 按回车键即输入该触点地址。

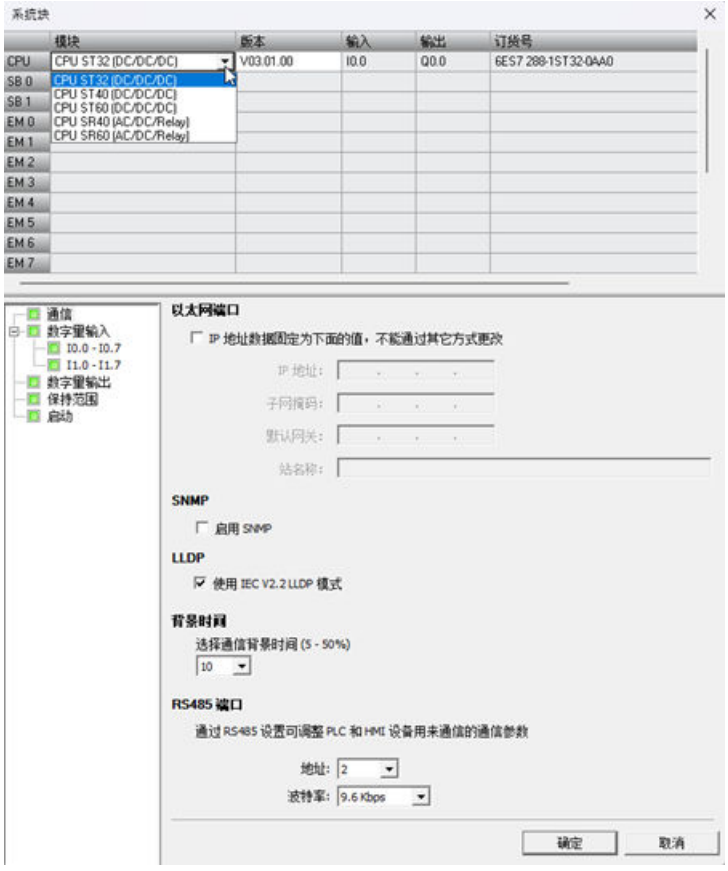
要输入用于接通 M0.0 的线圈：

1. 从位逻辑指令中选择输出线圈。
2. 按住鼠标左键并将输出线圈拖到第三个程序段中。
3. 单击线圈上方的“???”，然后输入以下地址： M0.0
4. 按回车键即输入该线圈地址。

3.2.4 为项目设置 CPU 的类型和版本

组态项目，使 CPU 和版本与物理 CPU 相匹配。如果项目组态所使用的 CPU 及 CPU 版本不正确，则将可能导致下载失败或程序无法运行。

如需选择 CPU，则请单击“模块”(Module) 列下的“CPU”字段，将显示下拉列表按钮，从下拉列表中选择所需 CPU。执行相同的步骤，在“版本”(Version) 列中选择 CPU 版本。



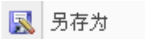
3.2.5 保存示例项目

保存示例项目

输入以上三个指令程序段后，即已完成程序的输入。程序保存后，即创建了一个含 CPU 类型和其他参数的项目。 要以指定的文件名在指定的位置保存项目：

1. 在“文件”(File) 菜单功能区的“操作”(Operations) 区域，单击“保存”(Save) 按钮下的向下箭头以显示“另存为”(Save As) 按钮。

2. 单击“另存为”(Save As) 按钮，然后为保存项目提供文件名。



3. 在“另存为”(Save As) 对话框中输入项目名称。
4. 浏览到想要保存项目的位置。
5. 点击“保存”(Save) 以保存项目。

保存项目后，可下载程序到 CPU。

3.3 下载示例程序

首先确保以太网 (页 77)（仅限标准型 CPU）或 RS485 (页 558) 通信的网络硬件和 PLC 连接器电缆正常运行并且 PLC 通信正常运行。

要下载所有项目组件，在“文件”(File) 或 PLC 菜单功能区的“传送”(Transfer) 区域单击“下载”(Download) 按钮，也可按快捷键组合“CTRL+D”。

单击“下载”(Download) 对话框中的“下载”(Download) 按钮。

STEP 7-Micro/WIN SMART V3 将完整程序或所选择的程序组件复制到 CPU。

如果 CPU 处于 RUN 模式，将弹出一个对话提示您将 CPU 置于 STOP 模式。单击“是”(Yes) 可将 CPU 置于 STOP 模式。

说明

每个项目都与 CPU 类型相关联。如果项目类型与所连接的 CPU 不匹配，STEP 7-Micro/WIN SMART V3 将指示不匹配并提示要采取措施。

3.4 更改 CPU 的工作模式

CPU 有以下两种工作模式：STOP 模式和 RUN 模式。CPU 正面的状态 LED 指示当前工作模式。在 STOP 模式下，CPU 不执行程序，用户可以下载程序。在 RUN 模式下，CPU 执行程序；但用户仍可下载程序。

将 CPU 置于 RUN 模式

1. 在 PLC 菜单功能区或程序编辑器工具栏中单击 “运行” (RUN) 按钮：
2. 收到提示后，单击“确定”(OK) 更改 CPU 的运行模式。

可监视 STEP 7-Micro/WIN SMART V3 中的程序，方法是在“调试”(Debug) 菜单功能区或程序编辑器工具栏中单击“程序状态”(Program Status) 按钮。STEP 7-Micro/WIN SMART V3 将显示指令值。

将 CPU 置于 STOP 模式

若要停止程序，需单击“停止”(STOP) 按钮，并确认有关将 CPU 置于 STOP 模式的提示。也可在程序逻辑中包括 STOP 指令 (页 451)，以将 CPU 置于 STOP 模式。

安装 4

4.1 S7-200 SMART G2 设备安装准则

S7-200 SMART G2 设备设计得易于安装。S7-200 SMART G2 可采用水平或垂直方式安装在标准 DIN 导轨上。S7-200 SMART G2 体积小，用户能更有效地利用空间。

⚠ 警告
S7-200 SMART G2 PLC 安装的安全要求 S7-200 SMART PLC 是敞开式控制器。必须将 PLC 安装在机柜、控制柜或电控室内。敞开式控制器必须安装在外壳内，以保护您免受机械伤害、电击和火灾等危险。仅限获得授权的相关人员可以打开机柜、控制柜或进入电控室。 不遵守这些安装要求可能导致人员死亡或重伤和/或设备损坏。 安装 PLC 时务必遵守这些要求。

安装 S7-200 SMART G2 时，应提供干燥的环境。因为 SELV/PELV 电路可以在干燥的环境中提供防触电保护。

根据适用的电气和建筑规范，此安装应提供适当的机械强度、易燃性保护以及稳定性保护，并且经过批准适用于特定位置类别中的开放式设备。

由灰尘、湿气和空气污染引起的导电污染可能会导致 PLC 出现运行故障和电气故障。

如果将 PLC 放置在可能存在导电污染的区域中，则必须使用具有适当防护等级的外壳对 PLC 进行保护。IP54 防护等级通常用于污染环境中的电子设备外壳，这种防护等级可能适合您的应用。

⚠ 警告
不正确安装 S7-200 SMART G2 可能会导致电气故障或机器意外运行。 这种电气故障或机器意外运行可能会造成死亡、人员重伤和/或财产损失。 为了确保设备安全运行，必须遵守所有有关安装和维护正确运行环境的说明。

稳固连接

在环境振动较大或进行垂直安装的应用场景中，建议在系统两侧安装 DIN 导轨端盖（8WA1 805）以确保模块保持连接状态。

将设备与热源、高压和电气噪声隔离开

作为布置系统中各种设备的基本规则，必须将产生高压和高电噪声的设备与 PLC 等低压逻辑型设备隔离开。

在面板上配置 PLC 的布局时，应注意发热设备并将电子型设备安装在控制柜中温度较低的区域内。少暴露在高温环境中可延长所有电子设备的使用寿命。

还要考虑面板中设备的布线。避免将低压信号线和通信电缆铺设在具有交流电源线和高能量快速开关直流线的槽中。

为散热和布线留出足够的空间

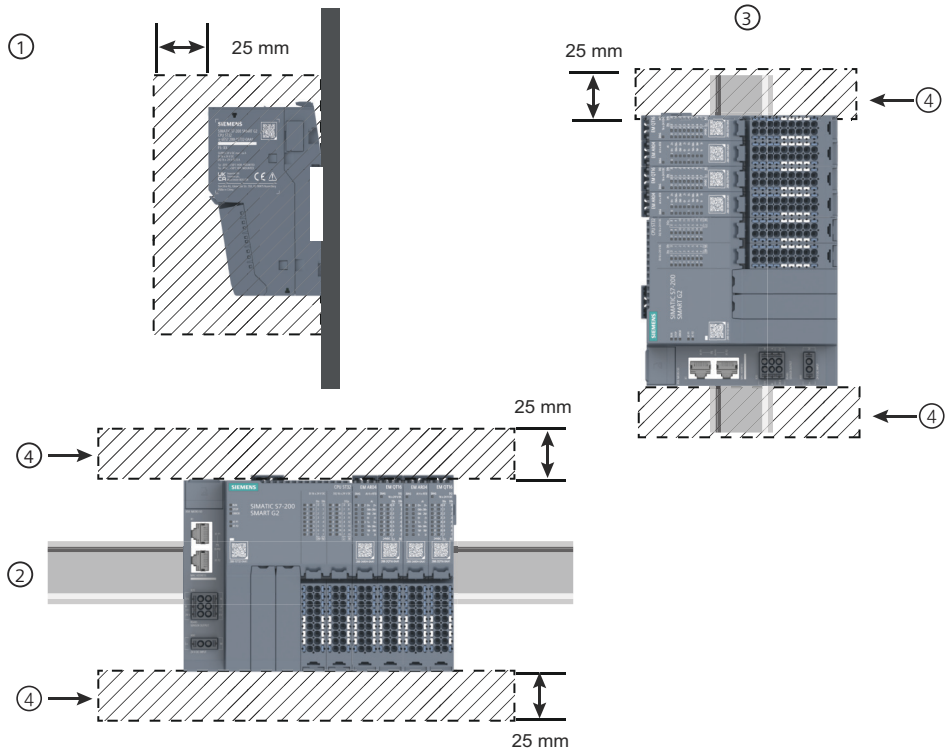
S7-200 SMART G2 设备设计成通过自然对流冷却。为保证适当冷却，必须在设备上方和下方留出至少 25 mm 的间隙。此外，模块前端与机柜内壁间至少应留出 25 mm 的深度。

⚠ 小心

温度相关注意事项

垂直安装时，允许的最高环境温度将降低 10 摄氏度。户外操作时，温度变化过大可能会导致过程操作不稳定或轻微人身伤害。

如果安装中包含扩展模块，建议将 CPU 安装于所有扩展模块下方，如下图所示。模块安装请遵循规定的相关指南，以确保适当冷却。



① 侧视图
② 水平安装
③ 垂直安装
④ 空隙区域

规划 PLC 的布局时，应留出足够的空间以方便进行接线和通信电缆连接。

降低热负荷的安装规则

以下规则有助于降低控制柜中 S7200 SMART G2 系统的热应力。

- 应优先采用水平安装方式
- 垂直安装时，将高功耗模块插到顶部，低功耗模块插到底部
- 将带有高功耗模块的 CPU 站安装在控制柜的下部区域
- 对于多层配置，将高功耗模块插到两侧，以便废热能够不受阻碍地向上散逸

在安装或拆卸任何电气设备之前，请确保已关闭相应设备的电源。同时，还要确保已关闭所有相关设备的电源。

⚠ 警告

安装或拆卸设备时存在触电风险

对 S7-200 SMART G2 或相关设备执行带电安装或接线作业，可能会触电或意外触发设备操作。

务必遵守适当的安全预防措施，确保在安装或拆卸任何设备前断开 S7-200 SMART G2 或相关设备的电源。

如果在安装或拆卸 S7-200 SMART G2 或相关设备的过程中没有断开全部电源，可能会由于触电或意外触发设备操作而导致死亡、人员重伤和/或财产损失。

务必确保无论何时更换或安装 S7-200 SMART G2 设备，都使用正确的模块或同等设备。

务必遵循接地指南 (页 102)。

⚠ 警告

更换 S7-200 SMART G2 设备相关的风险

S7-200 SMART G2 模块安装不当可能导致 S7-200 SMART G2 中的程序工作异常。

更换 S7-200 SMART G2 设备时，使用相同的型号、方向和顺序。

否则，可能导致设备意外运行，从而造成死亡、严重人身伤害和/或财产损失。

⚠ 警告

在易燃或可燃环境中安装或拆除设备

在易燃或可燃环境中断开设备可能引起火灾或爆炸。

请勿在易燃或可燃环境中断开设备；在易燃或可燃环境中工作时，请遵循适当的安全预防措施。

火灾或爆炸则可能导致死亡、人员重伤和/或财产损失。

注意

静电放电可能带来的风险

静电放电可能会损坏存储卡或 CPU 上的卡槽。如果损坏，卡槽或存储卡可能会出现故障或不可运行。

要对存储卡和卡槽进行静电放电保护，请进行以下操作：

- 在操控存储卡时，请先接触接地传导垫或佩戴接地腕带。
- 将存储卡存放在导电容器内。

卡槽或存储卡故障可能导致财产损失。

4.2 功率预算

CPU 有一个内部电源，用于为 CPU、扩展模块以及信号板供电，并可满足其它 24 V DC 用户的电源要求。请使用以下信息作为指导，确定 CPU 可为组态提供多少电能（或电流）。

请参见具体 CPU (页 26) 的技术规范确定 24 V DC 传感器供电预算。请参考计算功率预算 (页 1253)，确定 CPU 能为您的组态提供多大功率。

CPU 可为系统中的任何扩展模块提供所需的 5 V DC 逻辑电源。

说明

如果超出 CPU 功率预算，则可能无法连接 CPU 允许的最大数量模块。

CPU 还提供 24 V DC 传感器电源，可以为输入点、扩展模块上的继电器线圈电源或其它要求供给 24 V DC。如果您的电源要求超出该传感器电源的预算，则必须给系统增加外部 24 V DC 电源。

如果需要外部 24 V DC 电源，请确保该电源不要与 CPU 的传感器电源并联。为提高电气噪声保护能力，建议将不同电源的公共端 (M) 连接在一起。

警告

安全电源连接

将外部 24 V DC 电源与 CPU 的 24 V DC 传感器电源并联会导致这两个电源之间有冲突，因为每个电源都试图建立自己首选的输出电压电平。

该冲突可能导致其中一个电源或两个电源的寿命缩短或立即发生故障，从而导致 PLC 系统意外运行。意外运行可能导致人员死亡、重伤和/或设备损坏。

CPU 的直流传感器电源和任何外部电源应给不同点供电。允许将多个公共端连接到一起。

S7-200 SMART G2 系统中的一些 24 V DC 电源输入端口是互连的，并且通过一个公共逻辑电路连接多个 M 端子。例如，在数据表中指定为“非隔离”时，以下电路是互连的：CPU 的 24 V DC 电源、EM 的继电器线圈的电源输入或非隔离模拟量输入的电源。所有非隔离的 M 端必须连接到同一个外部参考电位。

警告

防止意外电流

将非隔离的 M 端子连接到不同参考电位将导致意外的电流，该电流可能导致 PLC 和任何连接设备损坏或运行不确定。

不遵守这些准则可能会导致设备损坏或运行不确定，而后者可能导致死亡、人员重伤和/或财产损失。

务必确保 S7-200 SMART G2 系统中的所有非隔离 M 端子都连接到同一个参考电位。

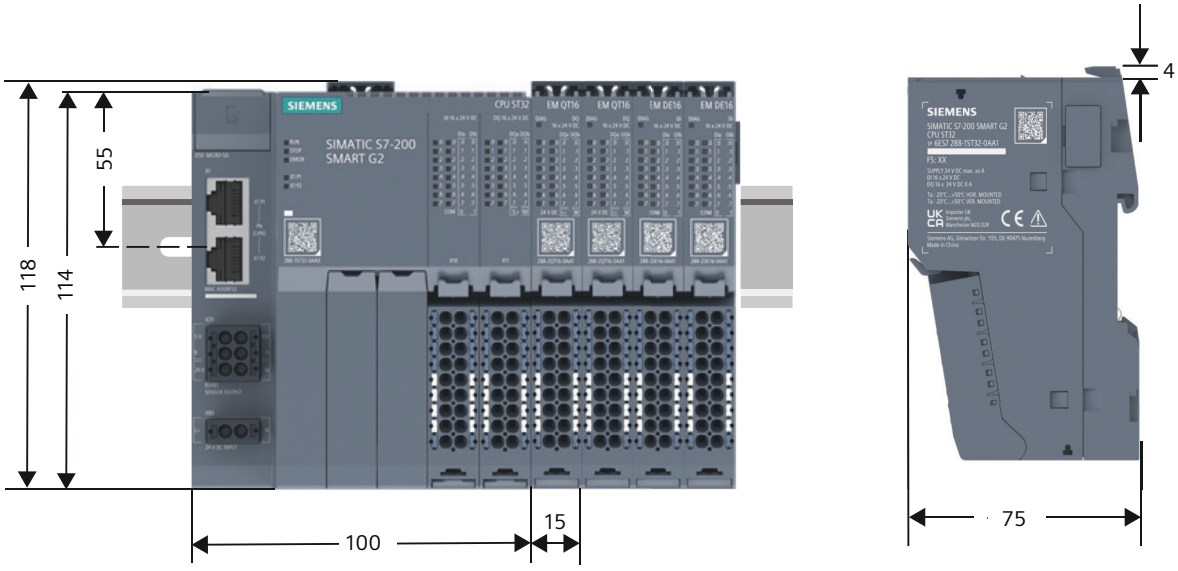
请参见具体 CPU (页 26) 的技术规范确定 24 V DC 传感器供电预算。

4.3 安装和拆卸操作步骤

4.3.1 S7-200 SMART G2 设备安装尺寸

使用以下安装尺寸来确保设备正确安装。所有安装尺寸均以毫米 (mm) 为单位。

CPU ST32



表格 4-1　安装尺寸 (mm)

S7-200 SMART G2 设备		高度 A (mm)	宽度 B (mm)
CPU	CPU ST32，DC/DC/DC	118	100
信号模块	CM01	30	15

S7-200 SMART G2 设备		高度 A (mm)	宽度 B (mm)
扩展模块	DE16 (16DI)	118	15
	QT16 (16DQ)	118	15
	AE04 (4AI)	118	15
	AQ04 (4AQ)	118	15
	AE08 (8AI)	118	15
	AM06 (4AI 2AQ)	118	15
	AR04 (4RTD)	118	15
	AT08 (8TC)	118	15

每个 CPU 和 EM 都支持安装在 DIN 导轨上。使用模块上的 DIN 导轨卡夹将设备固定到导轨上。这些卡夹还能掰到一个伸出位置以提供将设备直接安装到面板上的螺钉安装位置。

务必遵循接地指南 (页 102)。

4.3.2 安装和拆卸 CPU

CPU 可以很方便地安装到标准 DIN 导轨上。

在安装或拆卸任何电气设备之前，请确保已切断该设备的电源。 同时，还要确保已切断所有相关设备的电源。

⚠ 警告

安装或拆卸设备前，请切断 PLC 电源

如果在通电的情况下尝试安装或拆卸 CPU 或相关设备，则可能会触电或导致设备错误运行。

如果在安装或拆卸过程中未切断 PLC 和相关设备的所有电源，则可能导致人员死亡、重伤或设备损坏。

在安装或拆卸 CPU 或相关设备之前，必须采取合适的安全预防措施并确保切断该 PLC 的电源。

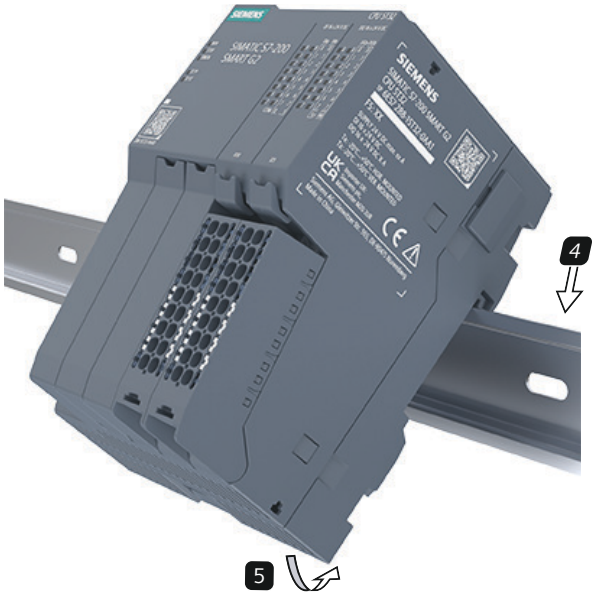
务必确保在更换或安装设备时使用正确的模块或同等设备。

⚠ 警告
模块更换 如果安装不正确的模块，CPU 中的程序将异常运行。 如果未使用同型号设备、未在相同方向或以相同顺序替换设备，则可能导致人员死亡、重伤和/或设备损坏。 使用同型号设备更换设备并确保安装的方向和位置正确。

将 CPU 安装在 DIN 导轨上

按照下面的步骤在 DIN 导轨上安装 CPU。

1. 安装 DIN 导轨；每隔 75 mm 将导轨固定到安装板上。
2. 确保 CPU 和所有 S7-200 SMART G2 设备都已断电。确认 CPU 的安装导轨释放按钮未被按下。
3. 将 CPU 挂到 DIN 导轨上方。
4. 向下转动 CPU ，直至听到"咔嗒"一声，表明安装导轨释放按钮已锁定到位。



从 DIN 导轨上拆卸 CPU

按照下面的步骤从 DIN 导轨上拆卸 CPU。

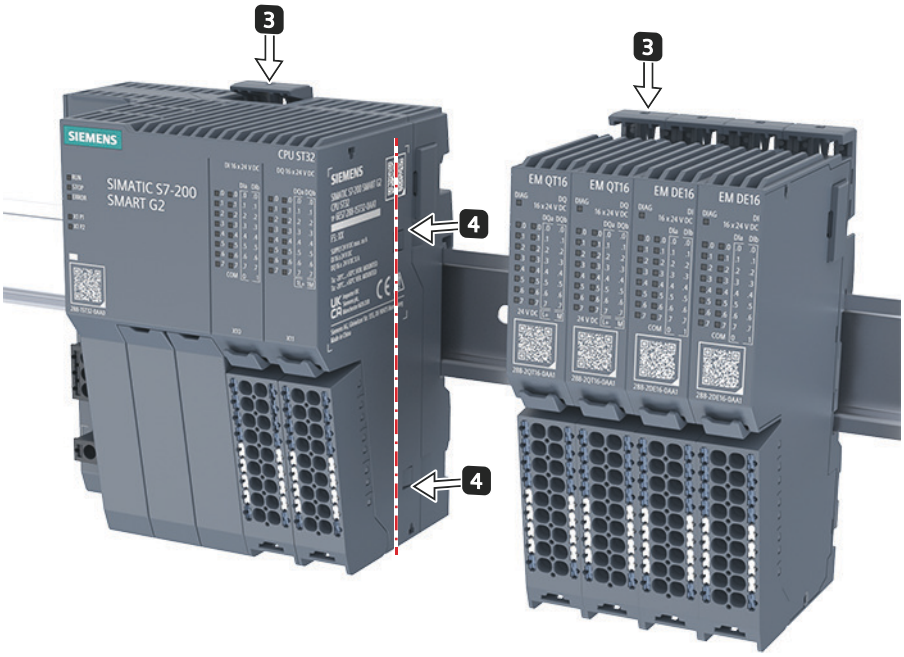
1. 切断 CPU 和连接的所有扩展模块的电源。
2. 断开连接到 CPU 的所有线缆。

3. 按下 CPU 及其相邻模块的安装导轨释放按钮， 以松开他们之间的连接。

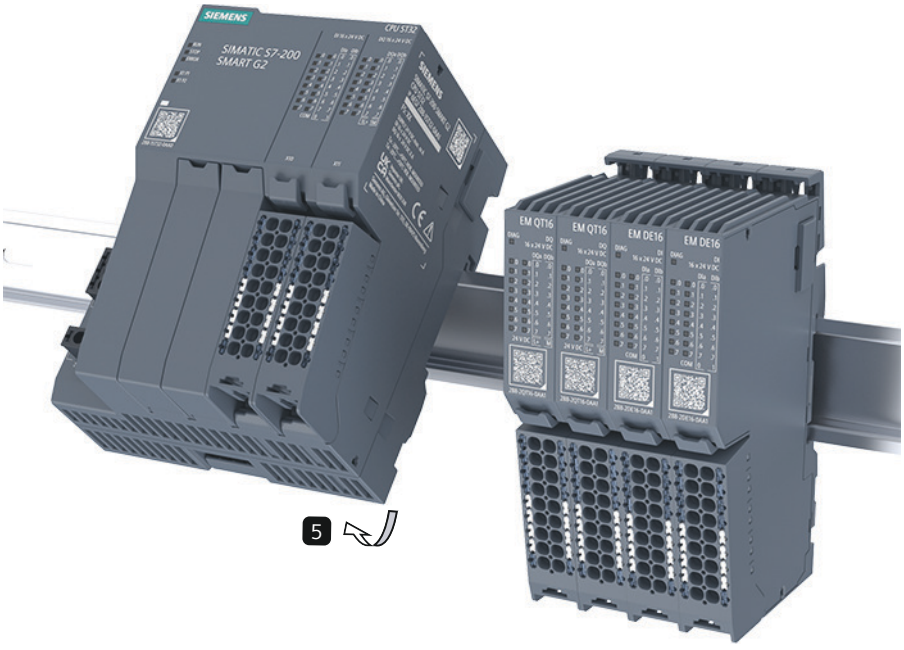
说明

安装导轨释放按钮位于 CPU 或扩展模块上方。

4. 如果连接了扩展模块，则向左滑动 CPU，将其从扩展模块连接器脱离。



5. 向上转动 CPU 使其脱离导轨，然后从系统中卸下 CPU。

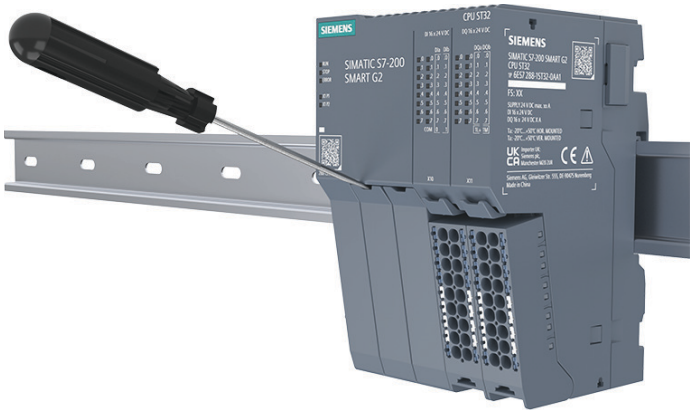


4.3.3 安装和拆卸信号板

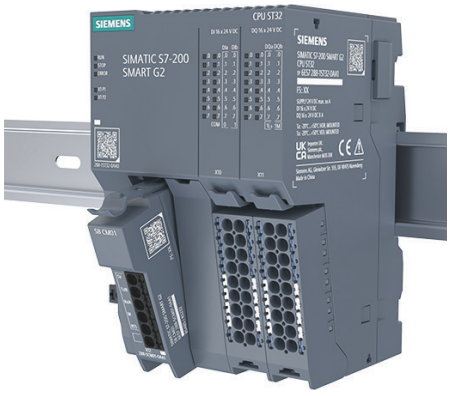
在 CPU 中安装信号板

请按以下步骤安装信号板

1. 确保 CPU 和所有 S7-200 SMART G2 设备与电源断开连接。
2. 断开连接到 CPU 的所有线缆。
3. 将 M3 螺丝刀插入信号板上方的预留槽中，然后撬开空白板盖板， 然后将其从 CPU 上拉出以将其卸下。收好板盖以备再次使用。



4. 将信号板下端挂到 CPU 扩展槽下端 ，然后向下按压直至其卡入到位。



拆下 CPU 中的信号板

请按以下步骤拆卸信号板

1. 确保 CPU 和所有 S7-200 SMART G2 设备与电源断开连接。
2. 将模块直接从 CPU 上部的安装位置中取出。请参考更换端子排 (页 111)
3. 将盖板重新装到 CPU 上。

4.3.4 安装和拆卸扩展模块

在安装 CPU 之后单独安装扩展模块。

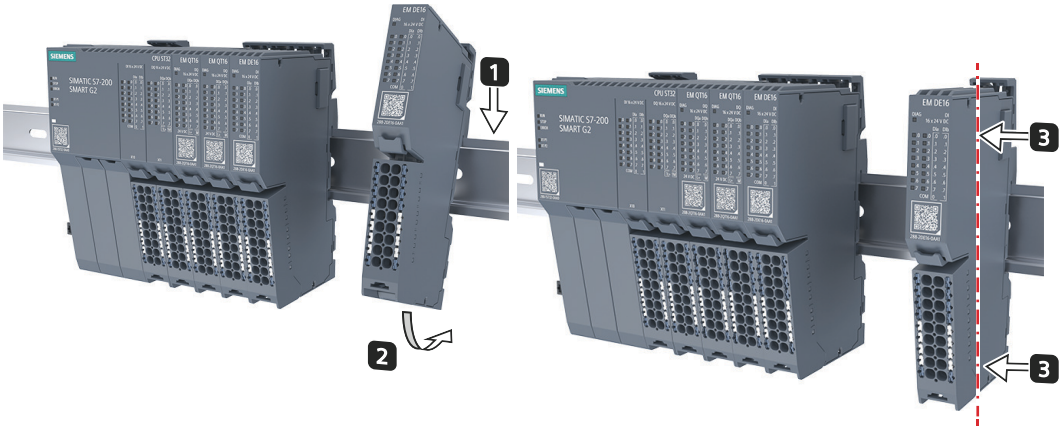
安装扩展模块

按照下面的步骤安装扩展模块：

1. 确保 CPU 和所有 S7-200 SMART G2 设备与电源断开连接。
2. 卸下 CPU 或扩展模块右侧的 I/O 总线连接器盖。将螺丝刀插入盖子上方的槽口，轻轻撬动以松开并取下盖子。收好盖以备再次使用。

将 扩展模块连接至 CPU。

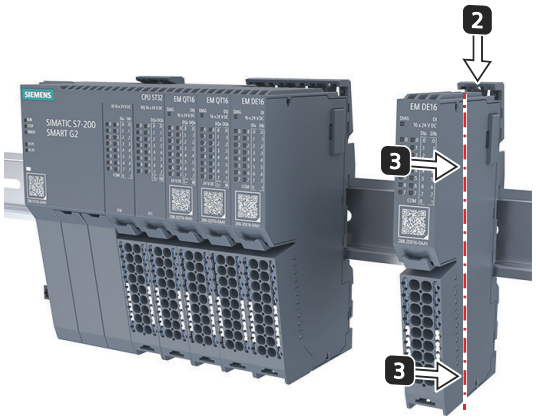
1. 在安装导轨上安装扩展模块。将扩展模块放置在 CPU 右侧。确认扩展模块的安装导轨释放按钮未被按下。
2. 向下转动扩展模块 ，直至听到"咔嗒"一声，表明安装导轨释放按钮已锁定到位。
3. 将模块的 I/O 总线连接器柱脚与 CPU 或模块的孔对齐，然后牢牢按压组件，直到柱脚卡入到位。



拆卸扩展模块

按照下面的步骤拆卸扩展模块：

1. 确保 CPU 和所有 S7-200 SMART G2 设备与电源断开连接。
2. 在要卸下的扩展模块上，用力按下安装导轨释放按钮。



3. 将 扩展模块向右平移。
4. 将扩展模块向上旋转，并从导轨上取下扩展模块。

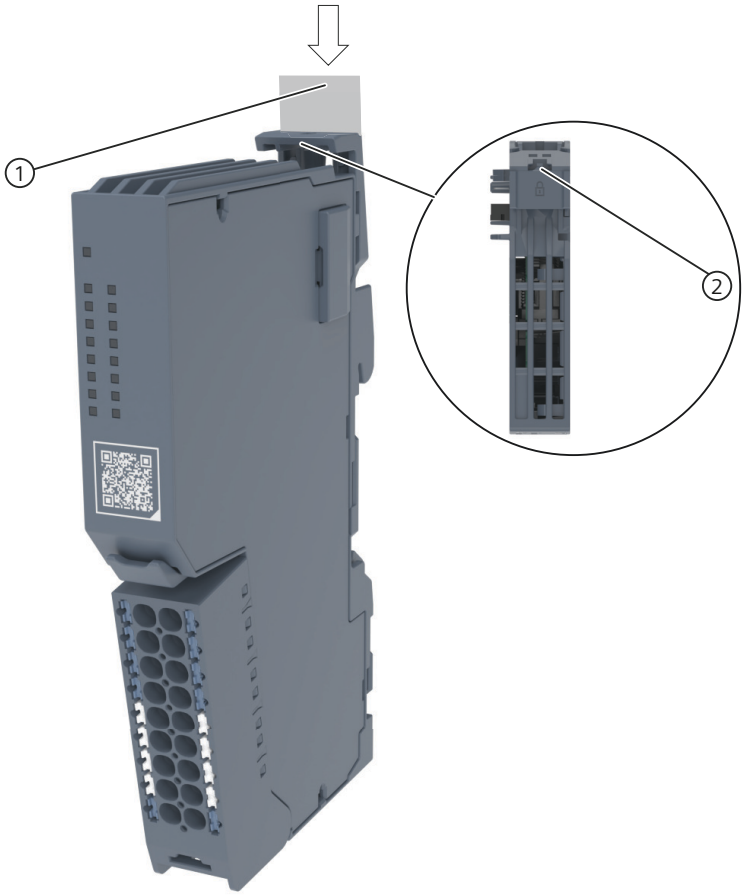
4.3.5 标注 S7200 SMART G2

4.3.5.1 贴上参考标识标签

操作步骤

要安装参考标识标签，请按以下步骤操作：

1. 从标签纸上撕下参考标识标签。
2. 将参考标识标签贴在 CPU 和扩展模块上。此插入口位于 CPU 或扩展模块顶部。



① 参考标识标签

② 标签插入口

说明

参考标识标签

参考标识标签的印刷面必须朝上。

4.4 接线准则

4.4.1 先决条件

所有电气设备的正确接地和接线非常重要，因为这有助于确保实现最佳系统运行以及为应用和 S7-200 SMART G2 提供更好的电噪声防护。有关 S7-200 SMART G2 接线图，请参见技术规范 (页 1214)。

先决条件

在对任何电气设备接地或安装接线之前，请确保该设备的电源已关闭。同时，还要确保所有相关设备的电源都已关闭。

> ⚠ **警告**
>
> **安装或拆卸设备时存在触电风险**
>
> 对 S7-200 SMART G2 或相关设备执行带电安装或接线作业，可能会触电或意外触发设备操作。
>
> 务必遵守适当的安全预防措施，确保在安装或拆卸任何设备前断开 S7-200 SMART G2 或相关设备的电源。
>
> 如果在安装或拆卸 S7-200 SMART G2 或相关设备的过程中没有断开全部电源，可能会由于触电或意外触发设备操作而导致死亡、人员重伤和/或财产损失。

设计 S7-200 SMART G2 系统的接地和接线时，始终要考虑安全性。在对 S7-200 SMART G2 和相关设备接线时遵守所有适用的电气规程。请根据所有适用的国家和地方标准来安装和操作所有设备。请联系当地的管理机构确定哪些规范和标准适用于您的具体情况。

> ⚠ **警告**
>
> **设备意外运行的风险**
>
> 控制装置可能在不安全的条件下出现故障，导致设备意外运行。
>
> 应使用紧急停止功能、机电超控功能或其他独立于 S7-200 SMART G2 的冗余安全功能。
>
> 意外操作可能导致死亡、人员重伤和/或财产损失。

S7-200 SMART G2 交流电源和 I/O 与交流电路的边界可以在交流线路电压与低压电路之间实现安全隔离。根据各种适用的标准，这些边界包括双重或加强绝缘，或者基本绝缘加辅助绝缘。跨过这些边界的组件（例如，光耦合器、电容器、变压器和继电器）已通过安全隔离认证。仅采用交流线路电压的电路才与其他电路实现安全隔离。24 V DC 电路间的隔离边界仅起一定作用，不应依赖于这些边界提供安全性。

根据 EN 61131-2，集成有交流电源的 S7-200 SMART G2 的传感器电源输出、通信电路和内部逻辑电路属于 SELV（安全超低电压）电路。

为了确保安全操作，为以下项目供电时，必须使用符合 SELV/PELV、2 类、限电压或限功率要求的认证电源：

- 通信端口
- 模拟电路
- 24 V DC 标称电源
- 数字量 I/O 电路

⚠ 警告

使用非隔离或单层绝缘电源存在触电风险

使用非隔离或单层绝缘电源可能会导致通常可以安全触摸的电路（如通信电路和低压传感器线路）带有危险的电压。

请勿使用非隔离或单层绝缘电源从交流线路为低压电路供电。

只应当使用合格的高压转低压整流器作为可安全接触的限压电路的供电电源。

使用非隔离或单层绝缘电源可能导致意外电击，从而造成人员死亡、重伤和/或财产损失。

⚠ 警告

安全使用电压转换器

如果使用非隔离或单绝缘电源通过交流线路给低压电路供电，可能会导致本来应当可以安全触摸的电路上出现危险电压，例如，通信电路和低压传感器线路。

这种意外的高压可能会导致人员死亡、重伤或设备损坏。

只能使用合格的高压转低压转换器作为可安全接触的限压电路的供电电源。

当 S7-200 SMART G2 安装到 DIN 导轨时，设备使用弹簧与 DIN 导轨正确接地连接。确保 DIN 导轨正确接地至系统的单点接地。

确定接地点时，应考虑安全接地要求和保护性中断装置的正常运行。

规划 S7-200 SMART G2 的接线时，应提供一个可同时切断 S7-200 SMART G2 CPU 电源、所有输入电路和所有输出电路电力供应的隔离开关。请提供过流保护（例如，熔断器或断路器）以限制电源线中的故障电流。考虑在各输出电路中安装熔断器或其他电流限制器提供额外保护。

CPU 晶体管输出端未集成过流保护功能。当输出端发生短路故障或负载电流超出其额定承受能力时，若无外部过流保护措施，容易导致 MOS 管损坏。为此，建议在 Q 点供电回路中增设外部过流保护。

为所有可能遭雷电冲击的线路安装合适的浪涌抑制设备。

避免将低压信号线和通信电缆铺设在具有交流线和高能量快速开关直流线的槽中。始终成对布线，中性线或公共线与火线或信号线成对。

使用尽可能短的电线并确保线径适合承载所需电流。使用适当尺寸的铜质终端套管将电线终端连接到连接器。

为了减少电噪声，请使用屏蔽线。

除了使用屏蔽线外，还可以采取以下措施进一步降低电气噪声：

- 将电缆屏蔽层的两端接地。
- 围绕屏蔽层使用夹子或铜带来提供较大的接地点连接表面，将其他电缆屏蔽层接地。

在给通过外部电源供电的输入电路接线时，应在电路中安装过流保护装置。由 S7-200 SMART G2 的 24 V DC 传感器电源供电的电路不需要外部保护，因为该传感器电源的电流已经受到限制。

所有 S7-200 SMART G2 模块都有供用户接线的可拆卸连接器。要防止连接器松动，请确保连接器妥善固定，锁定到位并且导线被牢固地安装到连接器中。

有关所提供的绝缘程度和绝缘边界位置的信息，请参见绝缘准则 (页 102)和技术规范 (页 1214)。

下表总结了 S7-200 SMART G2 CPU、EM 和 SB 的接线规则。

表格 4-2　S7-200 SMART G2 CPU、EM 和 SB 的接线规则

适用的接线规则...		CPU		SB	EM
		RS485(2x3)	Power in(2x1)		
实心电缆的电缆横截面积（铜缆）		IEC: 0.2 到 1.5 mm^2			
		UL: 24 到 14 AWG*			
柔性电缆的电缆横截面积（铜缆）	不带末端套管	IEC: 0.2 到 1.5 mm^2**			
		UL: 24 到 14 AWG*			
	带末端套管（带塑料套管）*** 符合 46228-4 or UL486F	IEC: 0.2 mm 到 1.5 mm^2			
		UL: 24 到 16 AWG*			
	带 TWIN 末端套管*** 符合 UL486F	IEC: 0.2 mm 到 0.5 mm^2	IEC:0.2 mm 到 0.75 mm^2	IEC: 0.2 mm 到 0.5 mm^2	IEC: 0.2 mm 到 0.75 mm^2
		UL: 24 到 20 AWG*	UL: 24 到 18 AWG*	UL: 24 到 20 AWG*	UL: 24 到 18 AWG*
导线的剥线长度		10 mm			

适用的接线规则...		CPU		SB	EM
		RS485(2x3)	Power in(2x1)		
末端套管，带塑料套管***	0.2 mm²≤ 电缆横截面积 ≤0.5 mm²	8 到 10 mm 长			
	0.75 mm²≤ 电缆横截面积 ≤1.5 mm²	10 mm 长			
TWIN 末端套管		12 mm 长			

末端套管，符合 DIN 46228 和 UL486F，带塑料套管***				最大电缆剥线长度（mm）
导体截面		压接管长度 (mm)	压接管类型	
mm²	AWG*			
0.14	26	8	AI 0.25-8	10
0.25	24	8	AI 0.25-8	10
		10	AI 0.25-10	10
0.34	22	8	AI 0.34-8	10
		10	AI 0.34-10	10
0.5	20	8	AI 0.5-8	10
		10	AI 0.5-10	10
0.75	18	10	AI 0.75-10	13
1		10	AI 1-10	13
1.5	16	10	AI 1.5-10	13

* AWG：美国线缆规格

** 末端套管，不带塑料套管：0.25 到 2.5 mm²/AWG：24 到 13

*** 请参见末端套管的注释

⚠ 危险

使用带有端子连接器的铜线

在所有通过端子连接到设备的电源线上使用铜（Cu）电缆， 例如，连接到 24 V DC 电源连接器的 24 V DC 电源电缆。

Utilisez des câbles en cuivre sur les connexions à bornes

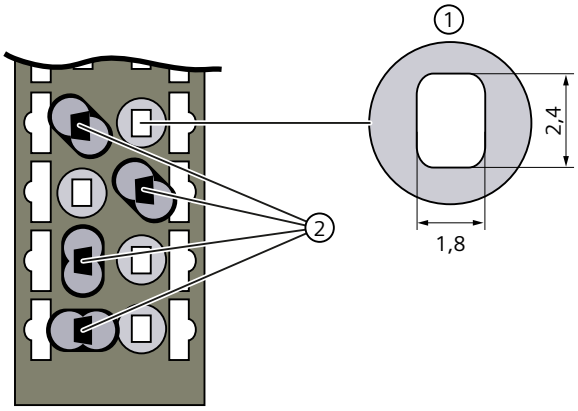
Utiliserz des câbles en cuivre (Cu) pour tous les câbles d'alimentation qui sont raccordés à l'appereil par des bornes, par example les câbles d'alimentation 24 V CC sur le connecteur d'alimentation 24 V CC.

说明

在标准铜导线上使用铜质终端套管可降低散丝导致短路的风险。长度超过建议剥线长度的端头必须包括一个绝缘环以避免因导线侧向移动而导致的短路。裸导线的横截面积限制也适用于端头。

I/O 模块直插式端子的电缆的 TWIN 末端套管

由于横截面积为 0.75 mm² 的 TWIN 末端套管需要占用一定的空间，因此压接 TWIN 末端套管时必须确保以合适的角度排列电缆，以便达到最佳排列效果。



① 端子隔断的横截面积

② 以合适的角度压接 TWIN 末端套管

图 4-1 Twin 末端套管

电缆温度测量阈值

说明

电缆温度测量阈值

在选择电缆时，请注意电缆的操作温度最多可以比 S7-200 SMART G2 系统的环境温度高 45 °C。（例如，环境温度为 60 °C 时，连接导线的温度范围至少要求为 90 °C）。

用户应根据所用电路的电气特征以及安装环境，指定其它连接类型和要求的材料。

灯负载的使用准则

钨丝灯接通时浪涌电流通常是稳态电流的 10 到 15 倍。对于需要频繁开关操作的灯负载，建议使用可更换的插入式继电器或浪涌限制器。

感性负载使用准则

将抑制电路与感性负载配合使用，以在控制输出断开时限制电压升高。抑制电路可保护输出，防止通过感性负载的电流中断时产生的高压瞬变导致其过早损坏。

此外，抑制电路还能限制感性负载开闭时产生的电噪声。未采用有效抑制措施的感性负载产生的高频噪声会中断 PLC 的运行。配备一个外部抑制电路，使其从电路上跨接在负载两端并且在位置上接近负载，这样对降低电气噪声最为有效。

S7-200 SMART G2 的直流输出包含内部抑制电路，该电路足以满足大多数应用对感性负载的要求。由于 S7-200 SMART G2 继电器输出触点可用于开关直流或交流负载，所以未提供内部保护。

针对感性负载（如接触器）的电磁干扰抑制，推荐采用以下方案：优先选用设备制造商已集成抑制电路的产品，或配置其提供的专用抑制模块作为选装组件。但是，一些制造商提供的抑制电路可能不适合您的应用。为获得最佳的噪声消减和触点寿命，可能还需要额外的抑制电路。

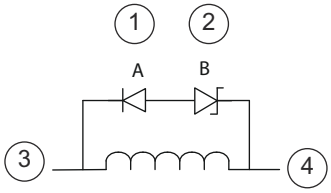
对于交流负载，可将金属氧化物变阻器 (MOV) 或其它电压钳制设备与并联 RC 电路配合使用，但不如单独使用有效。不带并联 RC 电路的 MOV 抑制器通常会导致出现高达钳位电压的显著高频噪声。

良好的受控关断瞬变的振铃频率不超过 10kHz，最好小于 1kHz。交流线路的峰值电压对地应在 +/-1200V 的范围内。使用 PLC 内部抑制的直流负载的负峰值电压比 24 V DC 电源电压低大约 40 V。外部抑制应将瞬变限制在 36V 电源范围内，以卸载内部抑制。

说明

抑制电路的有效性取决于具体应用，必须验证其是否适合您的具体应用。确保所有组件的额定值均正确，并使用示波器观察关断瞬变。

用于开关直流感性负载的直流或继电器输出的典型抑制电路



① 1N4001 二极管或同等元件

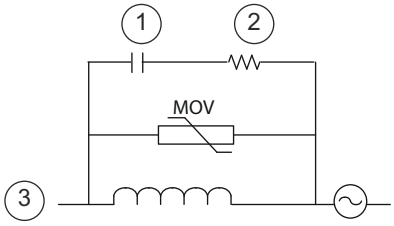
② 8.2 V 稳压二极管（直流输出），36 V 稳压二极管（继电器输出）

③ 输出点

④ M，24 V 参考

在大多数应用中，在直流感性负载两端增加一个二极管 (A) 就可以了，但如果您的应用要求更快的关闭时间，则建议再增加一个稳压二极管 (B)。请确保正确选择稳压二极管，以适合输出电路中的电流量。

用于开关交流感性负载的继电器输出的典型抑制电路



① 关于 C 值，请参见表格

② 关于 R 值，请参见表格

③ 输出点

请确保金属氧化物变阻器 (MOV) 的工作电压至少比额定线电压高出 20%。

选择为脉冲应用推荐的脉冲级非感性电阻和电容（通常为金属薄膜型）。确认元件满足平均功率、峰值功率和峰值电压要求。

如果自行设计抑制电路，下表给出了一系列交流负载的建议电阻值和电容值。这些值是理想元件参数下的计算结果。表中的 I rms 指满载时负载的稳态电流。

表格 4-3　交流抑制电路电阻和电容值

电感负载			抑制值		
I rms	230 V AC	120 V AC	电阻		电容
A	VA	VA	Ω	W（功率额定值）	nF
0.02	4.6	2.4	15000	0.1	15
0.05	11.5	6	5600	0.25	470
0.1	23	12	2700	0.5	100
0.2	46	24	1500	1	150
0.5	115	60	560	2.5	470
1	230	120	270	5	1000
2	460	240	150	10	1500

表中的值满足的条件：

最大关断瞬变阶跃 < 500 V
电阻峰值电压 < 500 V
电容峰值电压 < 1250 V
抑制电流 < 负载电流的 8% (50 Hz)
抑制电流 < 负载电流的 11% (60 Hz)
电容 dV/dt < 2 V/μs
电容脉冲功耗：$\int(dv/dt)^2dt < 10000\ V^2/\mu s$
谐振频率 < 300 Hz
电阻功率对应于 2 Hz 最大开关频率
假设典型感性负载的功率因数为 0.3

警告

正确放置外部电阻/电容噪声抑制电路

当使用继电器扩展模块切换交流感应负载时，必须在交流负载两端并联外部电阻/电容噪声抑制电路，以防止意外的机器或过程操作。意外机器或过程操作将可能导致严重人身伤害，甚至死亡。

放置外部电阻/电容噪声抑制电路时，请务必确保遵循相关指南。

4.4.2 I/O 端子接线

介绍

说明

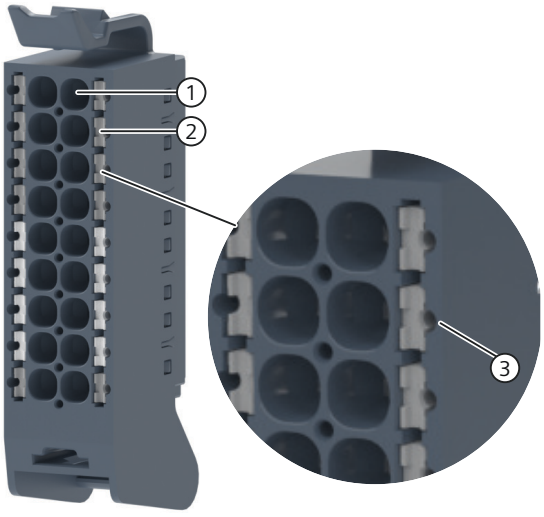
扩展模块的引脚分配取决于所使用的扩展模块。您可以在技术规范 (页 1224)中找到有关相应扩展模块的信息。

要求

- 电源电压已关闭
- 3 到 3.5 mm 一字螺丝刀

推入式端子

下图显示了推入式端子的视图。



① 直插式端子

② 弹簧式脱扣器

③ 测试探头开口（适用的测试探头：直径 1 mm，长度 ≥ 10 mm 且符合允许的电压类别）

电缆连接无需使用工具：不带导线端头的单线电缆，带导线端头或使用超声波密封的多线电缆

要连接不带端头的导线，请按以下步骤操作：

1. 将 8 到 10 mm 的电线剥开。
2. 仅标准导线（横截面积为 2.5 mm² 的导线除外）：
   使用末端套管密封或压接导线。
3. 将电缆尽可能深地插入直插式端子中。

电缆连接：多股电缆，不带末端套管，粗加工

要连接不带端头的导线，请按以下步骤操作：

1. 将 8 到 10 mm 的电线剥开。
2. 使用螺丝刀将电线推入弹簧式脱扣器中。
3. 将电缆尽可能深地插入直插式端子中。
4. 将螺丝刀从弹簧式脱扣器中拔出。

移除接线

使用螺丝刀，尽可能向下按压端子的弹簧式脱扣器并拉出导线。

说明

在按压弹簧式脱扣器时，请勿同时拉动导线/电缆，防止端子损坏。

4.5 维护

4.5.1 更换端子排

介绍

端子排是 CPU 和扩展模块的一个组件。如有需要，您可以更换端子排。为此，您无需拆下 CPU 扩展模块。更换端子排时，电位组的负载电流供电总线不会中断。

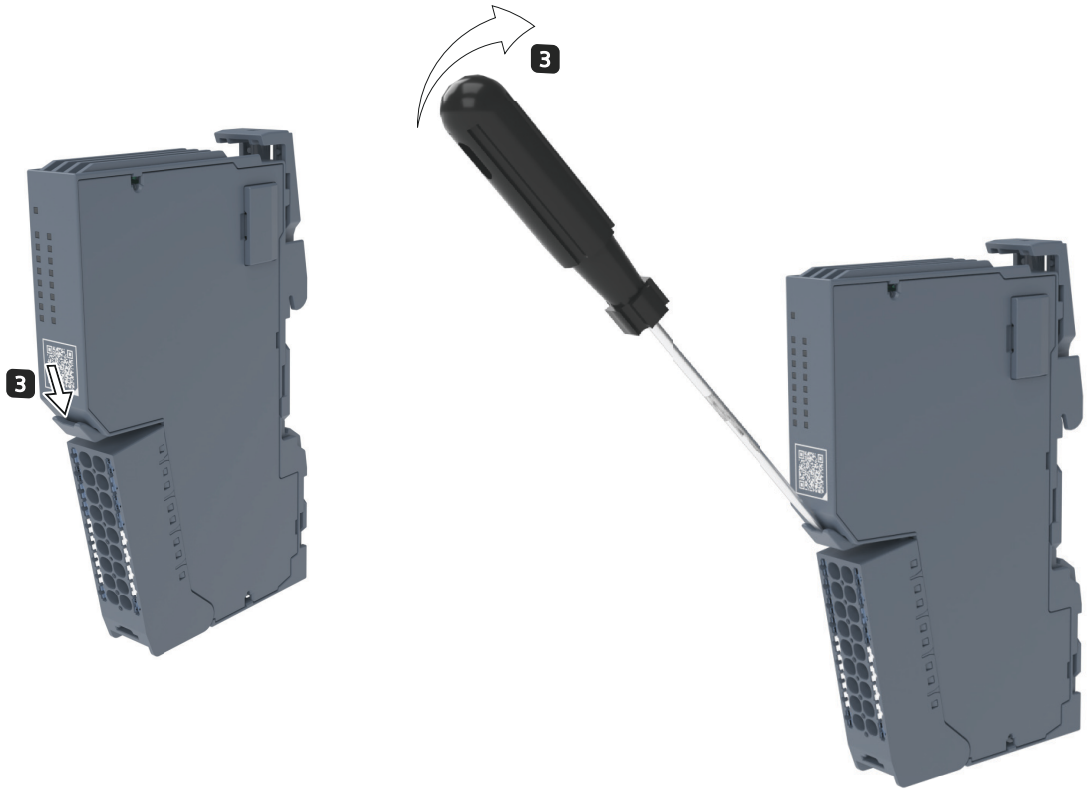
要求

- 只有在关闭电源电压的情况下才能更换端子排。
- CPU 或扩展模块已安装并接线。

操作步骤

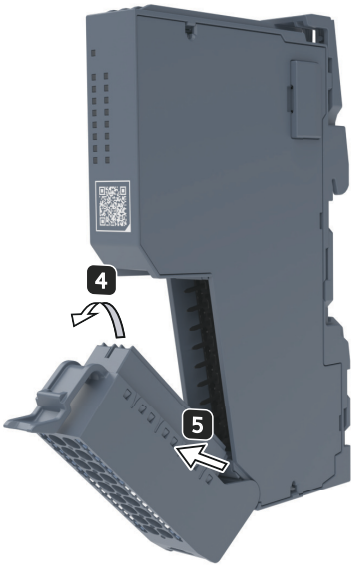
要更换端子排，请按以下步骤操作：

1. 断开模块上的电源电压。
2. 断开端子排上的接线。
3. 用手指按下释放机构。端子排的释放机构位于端子排的顶部。

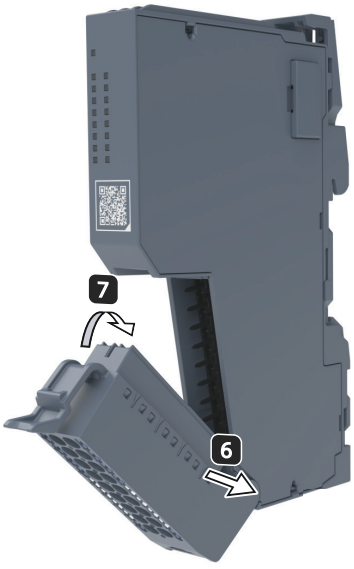


4. 向前倾斜已松开的端子排。

5. 将端子排从扩展模块中取出。



6. 将新的端子排插入扩展模块。
7. 将新的端子排向后旋转插入扩展模块，直到端子排卡入到位。
8. 接线端子排。



9. 接通扩展模块上所有现有的供电电压。

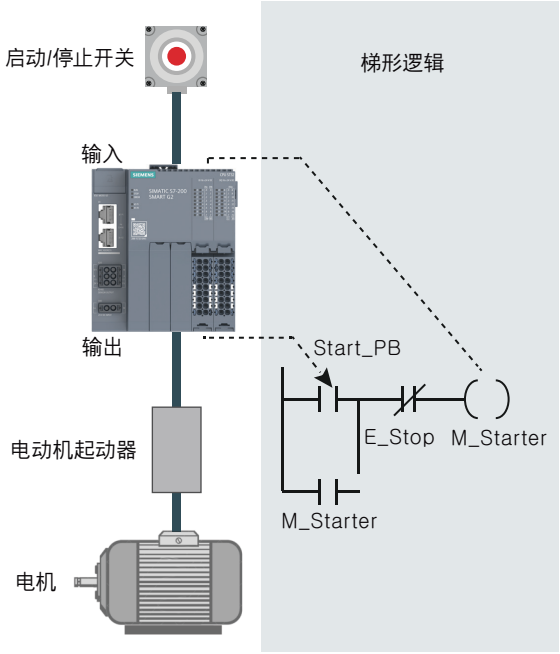
PLC 概念

5

5.1 控制逻辑的执行

CPU 连续执行程序中的控制逻辑和读写数据。基本操作非常简单：

- CPU 读取输入状态。
- 存储在 CPU 中的程序使用这些输入评估控制逻辑。
- 程序运行时，CPU 更新数据。
- CPU 将数据写入输出。

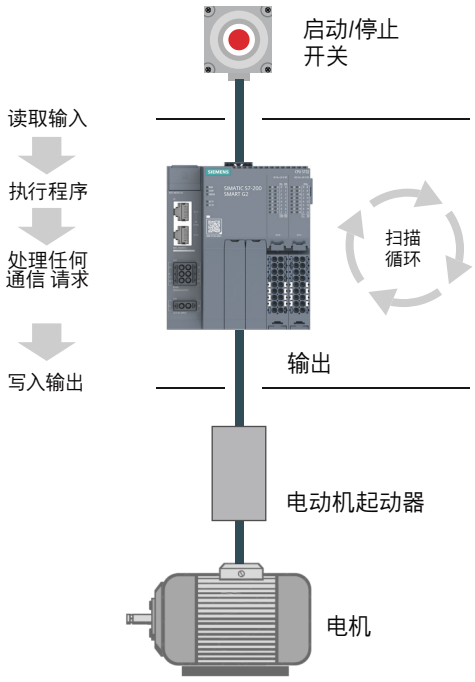


此图显示了电气继电器图与 CPU 关系的简图。在本例中，用于启动电机的开关的状态与其它输入的状态相结合。这些状态的计算结果决定用于控制电机启动执行器的输出的状态。

扫描周期中的任务

CPU 反复执行一系列任务。这种任务循环执行称为扫描周期。用户程序的执行与否取决于 CPU 是处于 STOP 模式还是 RUN 模式。在 RUN 模式下，执行程序；在 STOP 模式下，不执行程序。

表格 5-1　　CPU 在扫描周期中执行任务

扫描周期	说明
	读取输入：CPU 将物理输入的状态复制到过程映像输入寄存器。
	执行程序中的控制逻辑：CPU 执行程序指令，并将值存储到不同存储区。
	处理任何通信请求：CPU 执行通信所需的所有任务。
	执行 CPU 自检诊断：CPU 确保固件、程序存储器和所有扩展模块正确工作。
	写入输出：将存储在过程映像输入寄存器的数值写入到物理输出。

5.1.1　　读取输入和写入输出

读取输入

数字量输入： 每个扫描周期开始时，会读取数字量输入的电流值，然后将该值写入到过程映像输入寄存器。

模拟量输入：CPU 在正常扫描周期中不会读取模拟量输入值。而当程序访问模拟量输入时，将立即从设备中读取模拟量值。

写入输出

数字量输出：扫描周期结束时，CPU 将存储在过程映像输出寄存器的值写入数字量输出。

模拟量输出：CPU 在正常扫描周期中不会写入模拟量输出值。而当程序访问模拟量输出值时，将立即写入模拟量输出。

5.1.2 立即读取或写入 I/O

CPU 指令集提供立即读取或写入物理 I/O 的指令。这些立即 I/O 指令可用来直接访问实际输出或输入点，即使映像寄存器通常用作 I/O 访问的源地址或目的地址。使用立即指令来访问输入点时，不改变相应过程映像输入寄存器单元。使用立即指令来访问输出点时，将同时更新相应过程映像输出寄存器单元。

说明

读取模拟量输入时，可立即读取到相应的值。向模拟量输出写入值时，会立即更新该输出。

在程序执行期间，使用过程映像寄存器比直接访问输入或输出点更有优势。使用映像寄存器共有三个原因：

- 在扫描开始时对所有输入进行采样可在扫描周期的程序执行阶段同步和冻结输入值。程序执行完成后，使用映像寄存器中的值更新输出。这样会使系统更稳定。
- 程序访问映像寄存器的速度比访问 I/O 点的速度快得多，从而可以更快地执行程序。
- I/O 点是位实体，必须以位或字节的形式访问，但可以采用位、字节、字或双字的形式访问映像寄存器。因此，映像寄存器更为灵活。

5.1.3 执行用户程序

在扫描周期的执行阶段，CPU 执行主程序，从第一条指令开始并继续执行到最后一个指令。在主程序或中断例程的执行过程中，使用立即 I/O 指令可立即访问输入和输出。

如果在程序中使用子例程，则子例程作为程序的一部分进行存储。主程序、另一个子程序、功能块或中断例程调用子程序时，执行子程序。从主程序调用时子程序或功能块的嵌套深度是 8 级，从中断例程调用时嵌套深度是 4 级。

如果在程序中使用中断，则与中断事件相关的中断例程将作为程序的一部分进行存储。在正常扫描周期中并不一定执行中断例程，而是当发生中断事件时才执行中断例程（可以是扫描周期内的任何时间）。

为 14 个实体中的每一个保留局部存储器：主程序、八个子程序或功能块嵌套级别（从主程序启动时）、一个中断例程和四个子程序嵌套级别（从中断例程启动时）。局部存储器有一个局部范围，局部存储器仅在相关程序实体内可用，其它程序实体无法访问。有关局部存储器的详细信息，请参见本章中的局部存储区：L。

下图描述了一个典型的扫描流程，该流程包括局部存储器使用和两个中断事件（一个事件发生在程序执行阶段，另一个事件发生在扫描周期的通信阶段）。子例程由下一个较高级别调用，并在调用时执行。没有调用中断例程；发生相关中断事件时才调用中断例程。

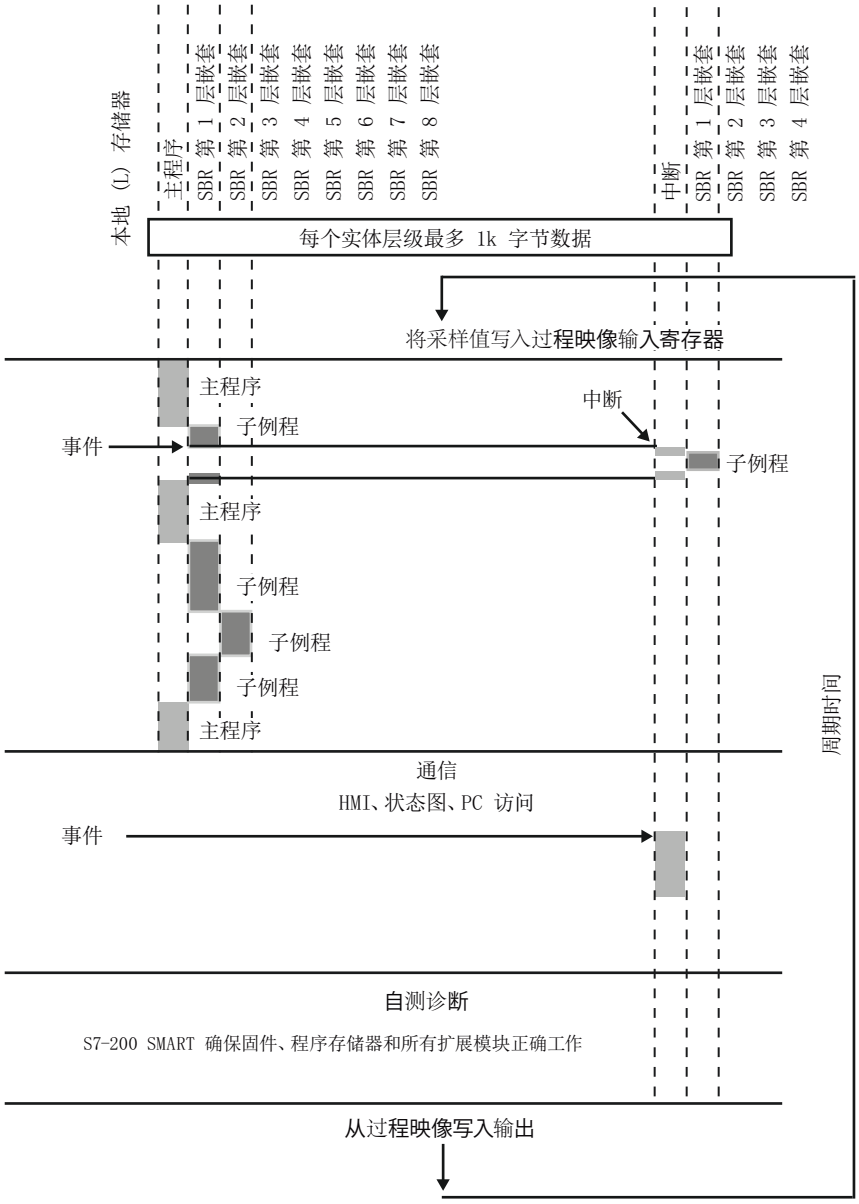


图 5-1 Typical_Scan_Flow

说明

FB 的调用方式与 SBR 类似。

5.2 访问数据

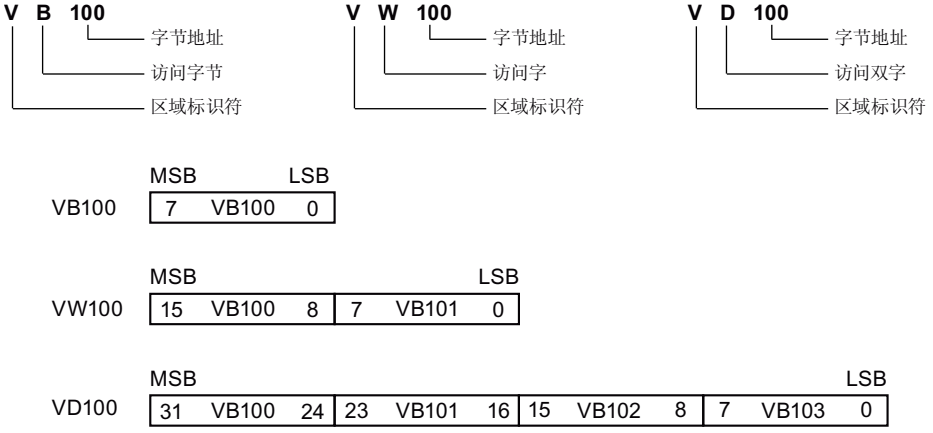
CPU 将信息存储在不同存储单元，每个位置均具有唯一的地址。可以显式标识要访问的存储器地址。这样程序将直接访问该信息。要访问存储区中的位，必须指定地址，该地址包括存储器标识符、字节地址和位号（也称为“字节.位”寻址）。

表格 5-2　位寻址

位地址元素	说明	
M 3 . 4 Ⓐ Ⓑ Ⓒ Ⓓ 0 1 2 3 Ⓔ 4 5 7 6 5 4 3 2 1 0 Ⓕ	A	存储区标识符
	B	字节地址：字节 3
	C	分隔符（“字节.位”）
	D	位在字节中的位置（位 4，共 8 位，编号 7 到 0 的位）
	E	存储区的字节
	F	选定字节的位

在此示例中，存储区和字节地址（“M3”）代表 M 存储器的第 3 个字节，用句点（“.”）与位地址（位 4）分开。

要按字节、字或双字访问存储器中的数据，必须采用类似于指定位地址的方法指定地址。使用“字节地址”格式可按字节、字或双字访问多数存储区（V、I、Q、M、S、L 和 SM）中的数据。这包括区域标识符、数据大小标识和字节、字或双字值的起始字节地址，如下图所示。



下表给出了不同数据长度可表示的整数值范围。

表格 5-3　不同数据长度表示的十进制和十六进制数范围

表示方式	字节 (B)	字 (W)	双字 (D)
无符号整数	0 到 255 16#00 到 16#FF	0 到 65,535 16#0000 到 16#FFFF	0 到 4,294,967,295 16#00000000 到 16#FFFFFFFF
有符号整数	-128 到 +127 16#80 到 16#7F	-32,768 到 +32,767 16#8000 到 16#7FFF	-2,147,483,648 到 +2,147,483,647 16#8000 0000 到 16#7FFF FFFF
实数（IEEE 32 位浮点数）	*不适用*	*不适用*	+1.175495E-38 到 +3.402823E+38（正数） -1.175495E-38 到 -3.402823E+38（负数）

使用包括区域标识符和设备编号的地址格式来访问其它 CPU 存储区（如 T、C、HC 和累加器）中的数据。

5.2.1 访问存储区

I（过程映像输入）

CPU 在每次扫描周期开始时对物理输入点采样，然后将采样值写入过程映像输入寄存器。用户可以按位、字节、字或双字来访问过程映像输入寄存器：

表格 5-4　I 存储器的绝对寻址

位：	I*[字节地址].[位地址]*	I0.1
字节、字或双字：	I*[大小][起始字节地址]*	IB4、 IW7、 ID20

Q（过程映像输出）

扫描周期结束时，CPU 将存储在过程映像输出寄存器的值复制到物理输出点。用户可以按位、字节、字或双字来访问过程映像输出寄存器：

表格 5-5　Q 存储器的绝对寻址

位：	Q[*字节地址*].[*位地址*]	Q1.1
字节、字或双字：	Q[*大小*][*起始字节地址*]	QB5、 QW14、 QD28

可以使用变量存储器存储程序中的控制逻辑所执行操作的中间结果。

V（变量存储器）

也可以使用变量存储器存储与过程或任务相关的其它数据。可以按位、字节、字或双字访问变量存储区：

表格 5-6　V 存储器的绝对寻址

位：	V[*字节地址*].[*位地址*]	V10.2
字节、字或双字：	V[*大小*][*起始字节地址*]	VB16 VW100 VD2136

DB2（变量存储器）

在 PLC 编程中，只能使用符号寻址来访问 DB2。但在与 HMI 通信时，必须使用绝对寻址来访问 DB2 区域。

表格 5-7　DB2 存储器的绝对寻址

位：	DB2.DBX[字节地址].[位地址]	DB2.DBX10.2
字节、字或双字：	DB2.DB[大小][起始字节地址]	DB2.DBB2 DB2.DBW4 DB2.DBD6

M（标志存储器）

可以将标志存储区（M 存储器）用作内部控制继电器来存储操作的中间状态或其它控制信息。 可以按位、字节、字或双字访问标志存储区：

表格 5-8　M 存储器的绝对寻址

位：	M*[字节地址].[位地址]*	M26.7
字节、字或双字：	M*[大小][起始字节地址]*	MB0、MW11、MD20

T（定时器存储器）

CPU 提供的定时器能够以 1 ms、10 ms 或 100 ms 的精度（时基增量）累计时间。定时器有两个变量：

- 当前值： 该 16 位有符号整数可存储定时器计数的时间量。
- 定时器位： 比较当前值和预设值后，可置位或清除该位。 预设值是定时器指令的一部分。

可以使用定时器地址（T + 定时器编号）访问这两个变量。 访问定时器位还是当前值取决于所使用的指令： 带位操作数的指令会访问定时器位，而带字操作数的指令则访问当前值。 如下图所示，“常开触点”指令访问的是定时器位，而“移动字”指令访问的是定时器的当前值。

表格 5-9　T 存储器的绝对寻址

定时器：	T*[定时器编号]*	T24

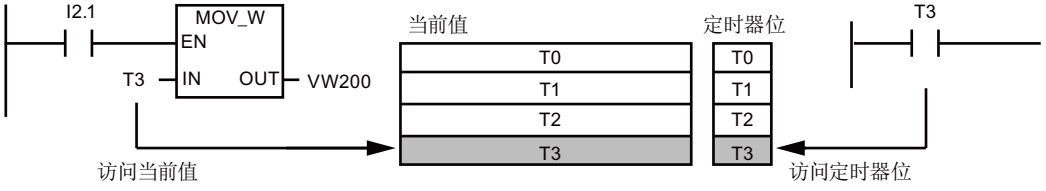


图 5-2　访问定时器位或定时器的当前值

C（计数器存储器）

CPU 提供三种类型的计数器，对计数器输入上的每一个由低到高的跳变事件进行计数： 一种类型仅向上计数，一种仅向下计数，还有一种可向上和向下计数。 有两个与计数器相关的变量：

- 当前值： 该 16 位有符号整数用于存储累加的计数值。
- 计数器位： 比较当前值和预设值后，可置位或清除该位。 预设值是计数器指令的一部分。

可以使用计数器地址（C + 计数器编号）访问这两个变量。 访问计数器位还是当前值取决于所使用的指令： 带位操作数的指令会访问计数器位，而带字操作数的指令则访问当前值。 如下图所示，“常开触点”指令访问的是计数器位，而“移动字”指令访问的是计数器的当前值。

表格 5-10　C 存储器的绝对寻址

计数器	C*[计数器编号]*	C24

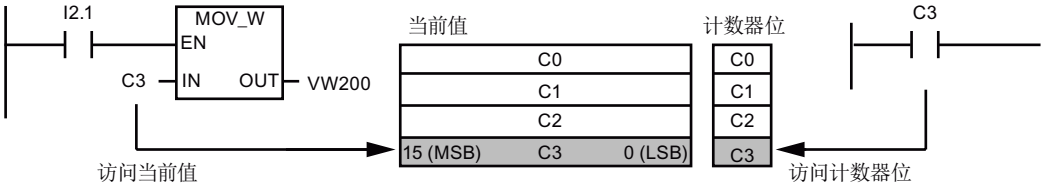


图 5-3　访问计数器位或计数器的当前值

HC（高速计数器）

高速计数器独立于 CPU 的扫描周期对高速事件进行计数。 高速计数器有一个有符号 32 位整数计数值（或当前值）。 要访问高速计数器的计数值，您需要利用存储器类型 (HC) 和计数器编号指定高速计数器的地址。 高速计数器的当前值是只读值，仅可作为双字（32 位）来寻址。

表格 5-11　HC 存储器的绝对寻址

高速计数器	HC*[高速计数器编号]*	HC1

AC（累加器）

累加器是可以像存储器一样使用的读/写器件。例如，可以使用累加器向子例程传递参数或从子例程返回参数，并可存储计算中使用的中间值。CPU 提供了四个 32 位累加器（AC0、AC1、AC2 和 AC3）。可以按字节、字或双字访问累加器中的数据。

被访问的数据大小取决于访问累加器时所使用的指令。如下图所示，当以字节或字的形式访问累加器时，使用的是数值的低 8 位或低 16 位。当以双字的形式访问累加器时，使用全部 32 位。

有关如何在中断子例程中使用累加器的信息，请参见中断指令 (页 413)。

表格 5-12　AC 存储器的绝对寻址

累加器	AC[*累加器编号*]	AC0

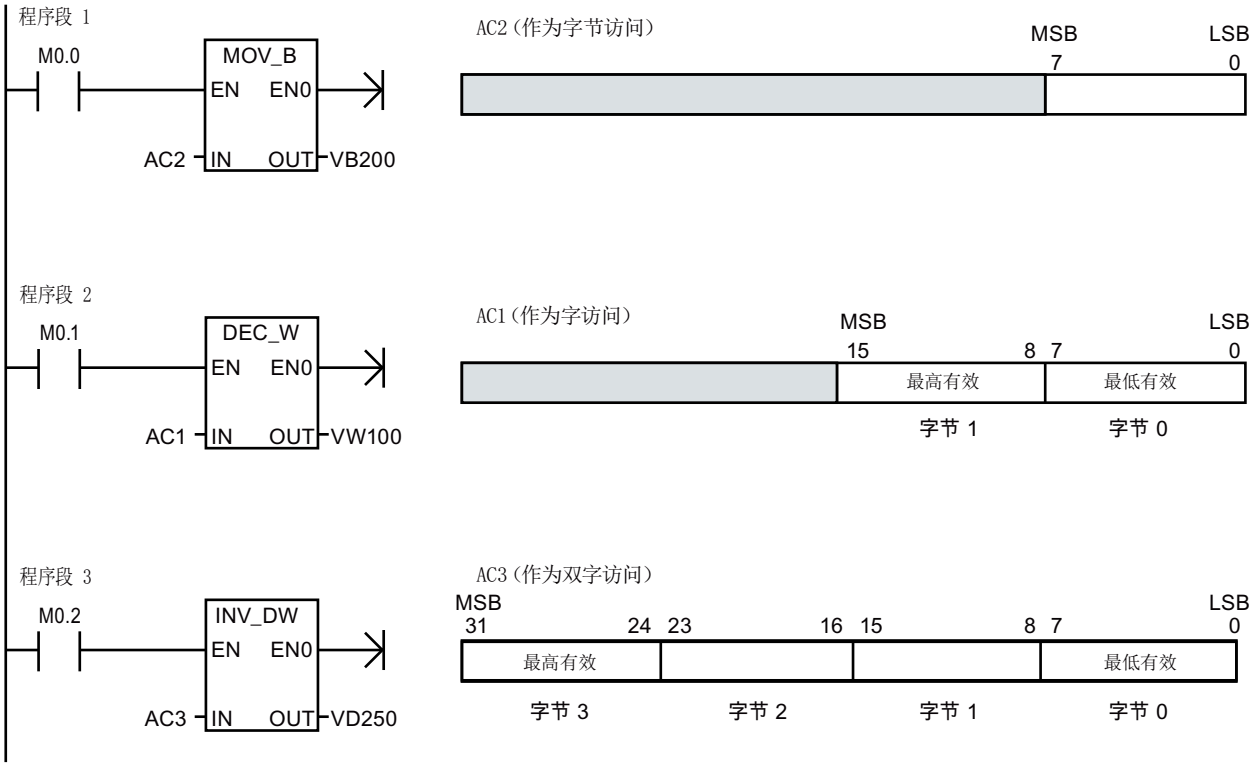


图 5-4　访问累加器

SM（特殊存储器）

SM 位提供了在 CPU 和用户程序之间传递信息的一种方法。可以使用这些位来选择和控制 CPU 的某些特殊功能，例如：在第一个扫描周期接通的位、以固定速率切换的位或显示数学或运算指令状态的位。可以按位、字节、字或双字访问 SM 位：

表格 5-13　SM 存储器的绝对寻址

位：	SM*[字节地址].[位地址]*	SM0.1
字节、字或双字：	SM*[大小][起始字节地址]*	SMB86、SMW300、SMD1000

更多相关信息，请参见 SM 位 (页 1265)的说明。

L（局部存储区）

在局部存储器栈中，CPU 为每个 POU （program organizational unit，程序组织单元）提供 64 个字节的 L 存储器。POU 相关的 L 存储器地址仅可由当前执行的 POU（主程序、子例程或中断例程）进行访问。当使用中断例程和子例程时，L 存储器栈用于保留暂停执行的 POU 的 L 存储器值，这样另一个 POU 就可以执行。之后，暂停的 POU 可通过在为其它 POU 提供执行控制之前就存在的 L 存储器的值恢复执行。

L 存储器栈最大嵌套层数限制：

- 当从主程序开始时为八个子例程嵌套层
- 当从中断例程开始时为四个子例程嵌套层

嵌套限制允许在程序中有 14 层的执行栈。例如，主程序（第 1 层）有八个嵌套子例程（第 2 层到第 9 层）。在执行第 9 层的子例程时，会发生中断（第 10 层）。中断例程包括四个嵌套的子例程（第 11 层到第 14 层）。

L 存储器规则：

- 可将 L 存储器用于所有类型 POU（主程序、子例程和中断例程）中的局部临时“TEMP”变量。
- 只有子例程可将 L 存储器用于传递到子例程或从子例程中传出的“IN”、“IN_OUT”和“OUT”类型的变量。
- 无论是以 LAD 还是以 FBD 编写子例程，TEMP、IN、IN_OUT 和 OUT 变量只能占 60 个字节。STEP 7-Micro/WIN SMART V3 使用局部存储器的最后四个字节。

局部存储器符号、变量类型和数据类型会在“变量”表中进行分配，当在程序编辑器中打开相关的 POU 时此表可用。当成功编译了 POU 时会自动分配 L 存储器的绝对地址。

在大多数情况下，在程序逻辑中使用 L 存储器符号名称引用，因为在成功编译整个 POU 之前，L 存储器的所有绝对地址均未知。然而，可以使用下表中列出的 L 存储器的绝对地址。

表格 5-14　L 存储器的绝对寻址

位：	L*[字节地址].[位地址]*	L0.0
字节、字或双字：	L*[大小] [起始字节地址]*	LB33、 LW5、 LD20

本地存储器和全局 V 存储器使用相似的地址语法，但 V 存储器在全局范围有效，而 L 存储器只在局部范围有效。全局范围表示任何 POU 均可访问同一存储器地址。局部范围是指 L 存储器分配与特定的 POU 相关，其它程序单元无法访问。

当全局符号和局部符号使用相同的名称时，L 存储器的局部范围还会影响符号的使用。如果程序逻辑引用此符号名称，CPU 会忽略全局符号并处理分配给局部存储器符号的地址。

说明

局部存储器的值分配不会为连续执行 POU 始终保留

当前嵌套的序列完成后，L 存储器地址会供下一个执行序列重复使用。根据 POU 在执行栈中的层级和上一次执行 POU 时完成的 L 存储器分配，上一次执行时完成的 POU 的 L 存储器分配会被意外值覆盖。

请牢记，在程序逻辑中，为 L 存储器变量重新分配正确的值。在处理所有 TEMP 值之前重新对其进行初始化，确保所有输出值（OUT 和 IN_OUT）都正确无误。

AI（模拟量输入）

CPU 将模拟量值（如温度或电压）转换为一个字长度（16 位）的数字值。 可以通过区域标识符 (AI)、数据大小 (W) 以及起始字节地址访问这些值。 由于模拟量输入为字，并且总是从偶数字节（例如 0、2 或 4）开始，所以必须使用偶数字节地址（例如 AIW0、AIW2 或 AIW4）访问这些值。 模拟量输入值为只读值。

表格 5-15　AI 存储器的绝对寻址

模拟量输入	AIW*[起始字节地址]*	AIW4

AQ（模拟量输出）

CPU 将一个字长度（16 位）的数字值按比例转换为电流或电压。 可以通过区域标识符 (AI)、数据大小 (W) 以及起始字节地址写入这些值。 由于模拟量输出为字，并且总是从偶数字节（例如 0、2 或 4）开始，所以必须使用偶数字节地址（如 AQW0、AQW2 或 AQW4）写入这些值。 模拟量输出值为只写值。

表格 5-16　AQ 存储器的绝对寻址

模拟量输出	AQW*[起始字节地址]*	AQW4

S（顺序控制继电器）

S 位与 SCR 关联，可用于将机器或步骤组织到等效的程序段中。 可使用 SCR 实现控制程序的逻辑分段。 可以按位、字节、字或双字访问 S 存储器。

表格 5-17　S 存储器的绝对寻址

位：	S*[字节地址].[位地址]*	S3.1
字节、字或双字：	S*[大小][起始字节地址]*	SB4、SW7、SD14

5.2.2　实数格式

实数（或浮点数）以 32 位单精度数表示，其格式为 ANSI/IEEE 754-1985 标准中所描述的形式。 实数按双字长度访问。

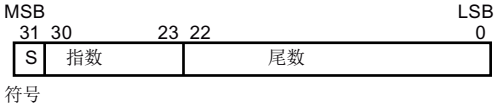


图 5-5　实数格式

说明

浮点数精确到小数点后第 6 位。 因此输入浮点常数时，最多只能指定 6 位小数。

计算涉及到包含非常大和非常小数字的一长串数值时，计算结果可能不准确。 如果数值相差 10 的 *x* 次方（其中 *x* > 6），则会发生上述情况。例如： 100 000 000 + 1 = 100 000 000

5.2.3 字符串格式

字符串是一个字符序列，其中的每个字符都以字节的形式存储。 字符串的第一个字节定义字符串的长度，即字符数。 下图显示了字符串的格式。 字符串的长度可以是 0 到 254 个字符，再加上长度字节，因此字符串的最大长度为 255 个字节。 字符串常数限制为 126 个字节。

图 5-6 字符串格式

5.2.4 分配指令的常数值

在许多编程指令中都可以使用常数值。 常数可以是字节、字或双字。 CPU 以二进制数的形式存储所有常数，随后可用十进制、十六进制、ASCII 或实数（浮点）格式表示这些常数。

表格 5-18 常数值的表示方式

表示方式	格式	示例
十进制	[十进制值]	20047
十六进制	16#[十六进制值]	16#4E4F
二进制	2#[二进制数]	2#1010_0101_1010_0101
ASCII	'[ASCII 文本]'	'ABCD'
实数	ANSI/IEEE 754-1985	+1.175495E-38（正数） -1.175495E-38（负数）
字符串	"[stringtext]"	"ABCDE"

说明

CPU 不支持“数据输入”或数据检查（如指定常数存储为整数、有符号整数或双整数形式）。 例如，加法指令可将 VW100 中的值用作有符号的整数值，而异或指令则可将 VW100 中的同一值用作无符号二进制值。

5.2.5 对本地 I/O 和扩展 I/O 进行寻址

CPU 提供的本地 I/O 具有固定的 I/O 地址。您可以通过在 CPU 的右侧连接扩展 I/O 模块，或通过安装信号板来增加 I/O 点。模块点的地址取决于 I/O 类型和模块在 I/O 链中的位置。举例来说，输出模块不会影响输入模块上的点地址，反之亦然。类似地，模拟量模块不会影响数字量模块的寻址，反之亦然。

说明

数字量 I/O 的过程映像寄存器空间总是以八位（一个字节）递增方式保留。如果模块没有为每个保留字节中的每一位提供相应的物理点，那些未使用的位就无法分配给 I/O 链中的后续模块。对于输入模块，这些未使用的位会在每个输入更新周期中被清零。

模拟量 I/O 点总是以两点递增的方式分配。如果模块没有为这些点分配相应的物理 I/O，则这些 I/O 点将丢失，并且不能够分配给 I/O 链中的后续模块。

下表提供固定映射惯例的示例（由 STEP 7 Micro/WIN SMART 建立，并作为系统块中 I/O 组态的一部分下载）。

表格 5-19 CPU 映射惯例

	CPU	**信号板 0**	**信号板 1**	**扩展模块 0**	**扩展模块 1**	**扩展模块 2**	**扩展模块 3**	**扩展模块 4**	**扩展模块 5**	**扩展模块 6**
起始地址	I0.0 Q0.0	I7.0 Q7.0 AIW12 AQW12	I72.0 Q72.0 AIW528 AQW528	I8.0 Q8.0 AIW16 AQW16	I12.0 Q12.0 AIW32 AQW32	I16.0 Q16.0 AIW48 AQW48	I20.0 Q20.0 AIW64 AQW64	I24.0 Q24.0 AIW80 AQW80	I28.0 Q28.0 AIW96 AQW96	I32.0 QI32.0 AIW112 AQW112

	扩展模块 7	**扩展模块 8**	**扩展模块 9**	**扩展模块 10**	**扩展模块 11**	**扩展模块 12**	**扩展模块 13**	**扩展模块 14**	**扩展模块 15**
起始地址	I36.0 Q36.0 AIW128 AQW128	I40.0 Q40.0 AIW144 AQW144	I44.0 Q44.0 AIW160 AQW160	I48.0 Q48.0 AIW176 AQW176	I52.0 Q52.0 AIW192 AQW192	I56.0 Q56.0 AIW208 AQW208	I60.0 Q60.0 AIW224 AQW224	I64.0 Q64.0 AIW240 AQW240	I68.0 Q68.0 AIW256 AQW256

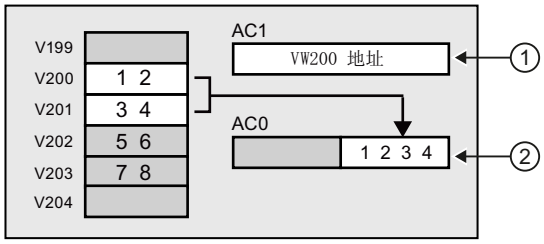
5.2.6 使用指针进行间接寻址

间接寻址使用指针访问存储器中的数据。指针是包含另一个存储单元地址的双字存储单元。只能将 V 存储单元、L 存储单元或累加器寄存器（AC1、AC2、AC3）用作指针。要创建指针，必须使用“移动双字”指令，将间接寻址的存储单元地址移至指针位置。指针还可以作为参数传递至子例程。

S7-200 SMART CPU 允许指针访问下列存储区：I、Q、V、M、S、AI、AQ、SM、T（仅限当前值）和 C（仅限当前值）。不能使用间接寻址访问单个位或访问 HC、L 或累加器存储区。

要间接访问存储器地址中的数据，通过输入一个“和”符号 (&) 和要寻址的存储单元的第一个字节，创建一个该位置的指针。指令的输入操作数前必须有一个“和”符号 (&)，表示存储单元的地址（而非其内容）将被移到在指令输出操作数中标识的位置（指针）。

在指令操作数前面输入一个星号 (*) 可指定该操作数是一个指针。如下图所示，输入 *AC1 表示 AC1 存储指向“移动字”(MOVW) 指令引用的字长度值的指针。在该示例中，在 VB200 和 VB201 中存储的值被移至累加器 AC0。



① MOVD &VB200, AC1

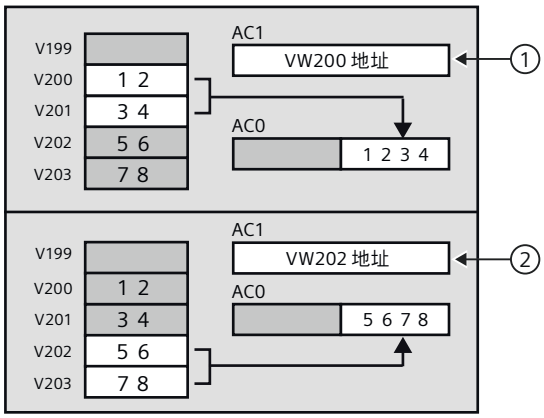
将 VB200（VW200 的初始字节）中的地址传送至 AC1 以创建指针

② MOVW *AC1, AC0

移动 AC1 中的指针引用的字值

图 5-7 创建和使用指针

如下图所示，可以更改指针的值。由于指针是 32 位值，请使用双字指令修改指针值。可使用简单数学运算（例如加或递增）修改指针值。



① MOVD &VB200, AC1

将 VB200（VW200 的初始字节）中的地址传送至 AC1 以创建指针

MOVW *AC1, AC0

移动 AC1 中的指针引用的字值

② +D +2, AC1

向累加器加 2 以指向下一个字位置

MOVW *AC1, AC0

移动 AC1 中的指针引用的字值

图 5-8 修改指针

说明

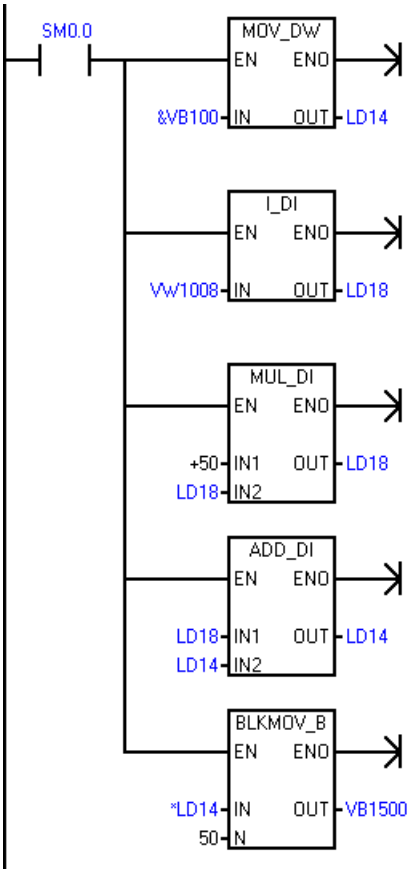
修改指针的值时，请记住调整所访问数据的大小：访问字节时，指针值加 1；访问定时器或计数器的字或当前值时，指针值加 2；访问双字时，指针值加 4。

5.2.7 指针示例

使用指针访问表中数据

该示例使用 LD14 作为指向存储在配方表中的配方的指针，配方表的起始地址为 VB100。 在本例中，VW1008 用于存储特定配方在表中的索引。 如果表中每一个配方的长度都是 50 字节，将该索引乘以 50 即可得到该特定配方的起始地址偏移量。 用指针加上该偏移量，即可访问表中的单独配方。 在本例中，配方会被复制到从 VB1500 开始的 50 个字节中。

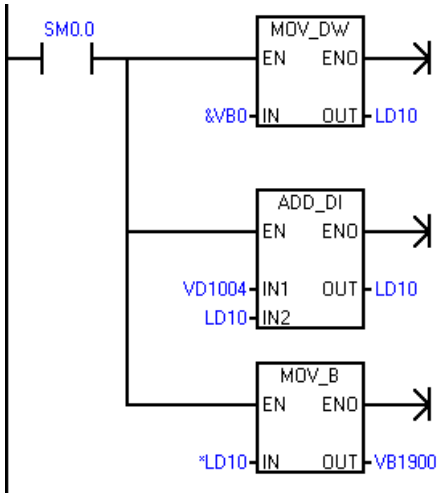
表格 5-20　示例： 使用指针访问表中数据

LAD		STL
	要传送配方表中的配方： • 每个配方的长度都是 50 字节。 • 索引参数 (VW1008) 标识要加载的配方。 创建指向配方表起始地址的指针。 将配方索引转换为双字值。 乘以偏移量，以容纳每个配方的大小。 将调整后的偏移量添加到指针。 将选定配方传送到 VB1500 至 VB1549	**Network 1** **LD SM0.0** **MOVD &VB100, LD14** **ITD VW1008, LD18** ***D +50, LD18** **+D LD18, LD14** **BMB *LD14, VB1500, 50**

使用偏移量访问数据

该示例将 LD10 用作指向地址 VB0 的指针。 然后，将指针增大 VD1004 中存储的偏移量。 LD10 随后将指向 V 存储器中的另一地址（VB0 + 偏移量）。 之后，LD10 指向的 V 存储器地址中的值将被复制到 VB1900。 通过更改 VD1004 中的值，您可以访问任意 V 存储单元。

表格 5-21　示例： 使用偏移量读取任意 V 存储单元的值

LAD		STL
	将 V 存储器的起始地址加载到指针。 将偏移量值添加到指针中。 将 V 存储单元中的值（偏移量）复制到 VB1900	**Network 1** **LD SM0.0** **MOVD &VB0, LD10** **+D VD1004, LD10** **MOVB *LD10, VB1900**

5.3 保存和恢复数据

5.3.1 下载项目组件

说明

下载时工程版本与 CPU 版本之间的兼容性

- 允许将 ST32（V3.1）的工程下载到 ST32（V3.2）的固件中，下载后运行 V3.2 的行为。
- 不允许将 ST32（V3.2）的工程下载到 ST32 （V3.1）的固件中。
- 不允许将 ST40/60、SR40/60 的工程下载到 G2 的 CPU 中；也不能将 G2 的工程下载到 ST40/60、SR40/60 的 CPU 中。

说明

将程序下载到 CPU 会彻底覆盖 CPU 中之前存在的任何程序。执行下载前，确定是要覆盖该程序。

要将项目组件从 STEP 7-Micro/WIN SMART V3 下载到 CPU，请按以下步骤操作：

1. 确保以太网 (页 77)或 RS485 (页 558) 通信的通信接口和 PLC 连接器电缆正常工作，并且 PLC 通信正常运行。
2. 登录到要下载其项目的 PLC。
   如果未登录，则在单击“下载”(Download) 按钮按钮时系统会发出通知。
3. 将 CPU 置于 STOP 模式 (页 86)。
4. 要下载所有项目组件，在“文件”(File) 或 PLC 菜单功能区的“传输”(Transfer) 区域单击“下载”(Download) 按钮，也可按快捷键组合 **CTRL+D**。

5. 要下载选定的项目组件，单击“下载”(Download) 按钮下的向下箭头，然后从下拉列表中选择待下载的特定项目组件（“项目”(Project) 或“用户自定义 Web (页 257)”(User Defined Web)）。
6. 单击“下载”(Download) 按钮后，如果弹出“通信”(Communications) 对话框，请选择要下载到 PLC 的通信接口和以太网 IP 地址或 RS485 网络地址。
7. 在“下载”(Download) 对话框中，设置块的下载选项，以及在 CPU 从 RUN 模式转换为 STOP 模式 (页 86)和从 STOP 模式转换为 RUN 模式 (页 86)时是否希望收到提示。
8. 或者，如果想要对话框在成功下载后自动关闭，请单击“成功后关闭对话框”(Close dialog on success) 复选框。
9. 单击“下载”(Download) 按钮。

说明

图中标记的功能仅对 V 和 DB2 保持范围有效。

STEP 7-Micro/WIN SMART V3 将完整程序和随附的程序组件复制到 CPU。状态图标指示信息性消息，或下载时是否出现潜在问题或错误。状态消息提供操作的特定结果。

说明

使用 STEP 7 Micro/WIN V3 创建的程序，仅支持下载到 S7-200 SMART 3.0 或更高版本的 CPU。

使用 STEP 7 Micro/WIN（2.8 版）或更早版本创建的程序，仅支持下载到 S7-200 SMART CPU V2.8 或更早版本的 CPU。

在 S7-200 SMART V3.0 和 G2 的 CPU 上，用户最多可下载 4 MB 的项目信息，如果超过此限值，系统会提示项目过大并阻止下载。

STEP 7-Micro/WIN SMART V3 还支持在 RUN 模式下执行程序编辑和下载 (页 189)。

下载过程

下载时，STEP 7-Micro/WIN SMART V3 和 CPU 对项目组件依次执行以下任务：

步骤	操作	相关主题和更多说明
1.	基于所选择的下载对象，程序编辑器中的项目组件充当下载操作的输入。程序编辑器可以包含输入的新程序数据、保存并打开的 .smartV3 项目或上传的 ASCII 导入文件。	打开文件 范围检查 项目文件 I/O 错误 程序编辑器错误
2.	STEP 7-Micro/WIN SMART V3 编译 编译或下载命令启动编译器。如果编译顺利通过，程序控制移交至下一步；如果未通过，退出编译或下载操作。	所有 STEP 7-Micro/WIN SMART V3 编译器错误都列在“输出窗口”。双击错误，编辑器将滚动至错误位置。编译成功后显示生成的程序和数据块大小。
3.	通过通信网络将块发送到 CPU 进行 PLC 编译。	通信错误 要下载（编辑器至 PLC）或上载（PLC 至编辑器），PLC 通信必须正常运行。确保网络硬件和 PLC 连接电缆正常操作。

步骤	操作	相关主题和更多说明
4.	PLC 编译 如果 PLC 编译成功，程序控制移交至下一步；如果失败，退出下载并报错。	PLC 编译器会验证 PLC 硬件支持全部程序指令、范围和结构。 在 PLC 菜单的“信息”(Information) 区域中，单击 PLC 按钮查看找到的第一个编译错误
5.	程序位于 CPU 永久存储器中，随时可在 RUN 模式下执行。	致命错误 (页 1262)和非致命运行错误 (页 1258)可从 PLC 菜单的“信息”(Information) 区域访问。

如果下载尝试生成编译器错误或下载错误，则更正错误，然后重新尝试下载。

另请参见

上传项目组件 (页 136)

5.3.2 上传项目组件

要将项目组件从 PLC 上传到 STEP 7-Micro/WIN SMART V3 程序编辑器，请按以下步骤操作：

1. 确保网络硬件和 PLC 连接器电缆（以太网或 RS485）运行正常，并确保 PLC 通信运行正常 (页 782)。
2. 登录到要上传其项目的 PLC。
   如果未登录，则在单击“上传”(Upload) 按钮时系统会发出通知。
3. 要上传所有项目组件，在“文件”(File) 或 PLC 菜单功能区的“传输”(Transfer) 部分单击“上传”(Upload) 按钮，或按快捷键组合 **CTRL+U**。

4. 要上传所选项目组件，单击“上传”(Upload) 按钮下的向下箭头，然后选择具体要上传的项目组件（项目、数据日志）。
5. 如果弹出“通信”(Communications) 对话框，请选择要上传的 PLC 通信接口和以太网 IP 地址或 RS485 网络地址。

6. （可选）如果要对话框在成功上传后自动关闭，单击“成功后关闭对话框”(Close dialog on success) 复选框。
7. 单击“上传”(Upload) 按钮以开始上传。

STEP 7-Micro/WIN SMART V3 将完整程序和随附的程序组件从 PLC 复制到当前打开的项目。状态图标指示信息性消息，或上传时是否出现潜在问题或错误。状态消息提供操作的特定结果。

如果上传成功，可保存上传的程序，或进行进一步更改。

说明

不能使用早期版本的 STEP 7-Micro/WIN SMART 将项目上传到版本 3.0 CPU。

说明

上传到新项目是捕获程序块、系统块和/或数据块信息的保险方法。由于项目空白，不会意外损坏数据。

如果要覆盖在下载 (页 133)到 PLC 后对程序进行的全部修改，上传到现有项目这一操作很有用。但是，上传到现有项目会覆盖对项目进行的任何添加或修改。只有在要使用存储在 PLC 中的项目彻底覆盖 STEP 7-Micro/WIN SMART V3 项目时，才使用此选项。

STEP 7-Micro/WIN SMART V3 不会上传注释，但是如果当前在程序编辑器中打开带有注释的程序，则保留这些注释。注意上传是否会覆盖现有项目，并且仅当项目类似时才使用此方法。

说明

如果您将 V3.1 的工程下载到 V3.2 的固件中，上传后会自动更新运动控制指令和 TO 到 V3.2 版本。

5.3.3 存储类型

CPU 提供了多种功能来确保用户程序和数据能够被正确保留。

- 保持性存储器：在一次上电循环中保持不变的可选择存储区。可在系统数据块中组态保持性存储器。在所有存储区中，只有 V、M 和定时器与计数器的当前值存储区能组态为保持性存储区。
- 永久存储器：用于存储程序块、数据块、系统块、强制值以及组态为保持性的值的存储器。
- 存储卡：用于标准 CPU 的可拆卸 microSDHC 卡，可用于以下用途：
  - 用于作为程序传送卡 (页 141)存储项目块
  - 作为恢复为出厂默认设置的卡 (页 241)完全擦除 PLC
  - 作为固件更新卡 (页 138)更新 PLC 和扩展模块固件

5.3.4 使用存储卡

使用存储卡

标准 S7-200 SMART CPU 支持使用 microSDHC 卡进行以下操作：

- 用户程序传送 (页 141)
- 将 CPU 重置为出厂默认状态 (页 241)
- 支持 CPU 和连接的扩展模块的固件更新

可使用任何容量最高为 128GB 的标准型商业 microSD 卡。

以下 CPU 行为是共同的，而无论存储卡的用法：

1. 在 RUN 模式下将存储卡插入 CPU 导致 CPU 自动转换到 STOP 模式。
2. 如果插入了存储卡，则 CPU 不可前进到 RUN 模式。
3. 仅在 CPU 上电或暖启动后执行存储卡评估。因此，只能在 CPU 上电或暖启动后进行程序传送和固件更新。
4. 存储卡可用于存储与程序传送和固件更新使用不相关的文件和文件夹，只要其名称不与用于程序传送和固件更新使用的文件和文件夹名称冲突。

⚠ 警告

安装存储卡之前，请验证 CPU 当前并未运行任何进程。

安装存储卡将导致 CPU 进入 STOP 模式，这可能会影响在线过程或机器的操作。意外的过程操作或机器操作可能会导致死亡、人身伤害和/或财产损失。

在插入存储卡前，请务必确保 CPU 处于离线模式且处于安全状态。

程序传送卡

存储卡可用于将用户程序内容传送到 CPU 永久存储器中，完全或部分替换已在装载存储器中的内容。

要用于程序传送目的，按以下方式组织存储卡：

表格 5-22　用于程序传送卡的存储卡

在卡的根级别	
文件：S7_JOB.S7S	包含字 TO_ILM 的文本文件
文件夹：SIMATIC.S7S	包含要传送到 CPU 的用户程序文件的文件夹

重置为出厂默认设置的卡

存储卡可用于擦除所有保留数据，将 CPU 重置为出厂默认状态。

要用于复位为出厂默认 (页 241)目的，请按以下方式组织存储卡：

表格 5-23　用于复位为出厂默认设置的存储卡

在卡的根级别	
文件：S7_JOB.S7S	包含字 RESET_TO_FACTORY 的文本文件

固件更新卡

存储卡可用于更新 CPU 和任何连接的扩展模块中的固件。

固件更新存储卡的文件和文件夹结构如下所示：

表格 5-24　用于固件更新目的的存储卡

在卡的根级别	
文件：S7_JOB.S7S	包含字 FWUPDATE 的文本文件
文件夹：FWUPDATE.S7S	包含要更新的每个设备的更新文件 (.upd) 的文件夹

上电后，如果 CPU 检测到存在存储卡，则其在该卡上找到并打开 S7_JOB.SYS 文件。如果 CPU 在该文件中发现 FWUPDATE 字符串，则 CPU 进入固件更新序列。

CPU 检查 FWUPDATE.S7S 文件夹中的每个更新文件 (.upd)，如果更新文件文件名中包含的顺序 ID 与连接的设备（CPU、扩展模块或信号板）的顺序 ID (MLFB) 匹配，则 CPU 会用更新文件内包含的固件内容更新该设备的固件。

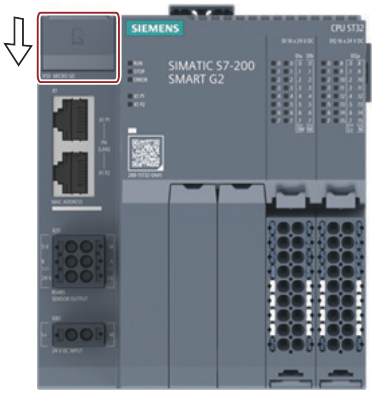
说明

通过 STEP 7-Micro/WIN SMART V3 执行固件更新

也可以通过 STEP 7-Micro/WIN SMART V3 使用 RS485 端口来执行固件更新。对于无存储卡的 CPU 型号，此方法尤为适用。相关说明，请参见 STEP 7-Micro/WIN SMART V3 在线帮助中的 PLC 菜单部分。

5.3.5 在标准 CPU 中插入存储卡

表格 5-25 在标准 CPU 中插入及移除存储卡

任务	步骤
	按照下面的步骤将 microSDHC 存储卡插入 CPU 中。 1. 拉下左上方的端子块连接器盖。 2. 将 microSDHC 存储卡插入位于端子块连接器上方的存储卡插槽（标记为 X50）。 3. 在插入卡后重新拉上端子块连接器盖，以确保该卡牢固。 按照下面的步骤从 CPU 中取下 microSDHC 存储卡。 1. 拉下左上方的端子块连接器盖。 2. 抓住 CPU 中的 microSDHC 存储卡并将其拉出卡插槽（标记为 Micro-SD X50）。 3. 重新拉上端子块连接器盖

5.3.6 通过存储卡传送程序

标准 S7-200 SMART CPU 型号使用 FAT32 文件系统格式支持容量处于 4 GB 到 128 GB 范围内的标准商用 microSD/microSDHC/microSDXC 卡。可将 microSD/microSDHC/microSDXC 卡用作程序传送卡，实现程序和项目数据的便携式存储。

说明

卡要求：

- 必须使用第三方工具将大于 32 GB 的卡格式化为 FAT32，因为 Windows 操作系统不支持将大于 32 GB 的卡格式化为 FAT32。
- 卡的速度等级 (speed class) <= Class10/U1（即：10MB/s）

警告

插入存储卡之前，请检查并确认 CPU 当前未执行任何进程。

在 RUN 模式下将存储卡插入 CPU 导致 CPU 自动切换到 STOP 模式。

将存储卡插入正在运行的 CPU 可导致过程操作中断，可能引起人员死亡或严重伤害。

插入存储卡前，务必确保 CPU 处于 STOP 模式 (页 145)。

在 PLC 上创建程序传送存储卡

说明

将程序传送到存储卡之前，STEP 7-Micro/WIN SMART V3 会首先删除卡上的所有 SIMATIC 内容。使用读卡器和 Windows 资源管理器在卡上存储的任何其他数据均不受干扰。

另请注意，如果读卡器已插入，则无法将 CPU 更改至 RUN 模式。

要将存储卡编程为程序传送卡，按以下步骤操作：

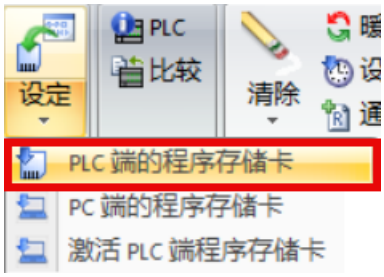
1. 确保网络硬件和 PLC 连接电缆正常工作，CPU 已上电并处于 STOP 模式且 PLC 通信正常运行 (页 77)。
2. 如果尚未插入，将 microSDHC 存储卡插入 CPU。可在 CPU 通电时插拔存储卡。
3. 如果尚未下载，将程序下载 (页 86)到 PLC。

   说明

   下载时工程版本与 CPU 版本之间的兼容性

   - 允许将 ST32（V3.1）的工程下载到 ST32（V3.2）的固件中，下载后运行 V3.2 的行为。
   - 不允许将 ST32（V3.2）的工程下载到 ST32 （V3.1）的固件中。
   - 不允许将 ST40/60、SR40/60 的工程下载到 G2 的 CPU 中；也不能将 G2 的工程下载到 ST40/60、SR40/60 的 CPU 中。

4. 单击 PLC 菜单功能区“设定”(Setting) 按钮下的向下箭头，然后选择“PLC 端的程序存储卡”(Program Memory Card in PLC)。

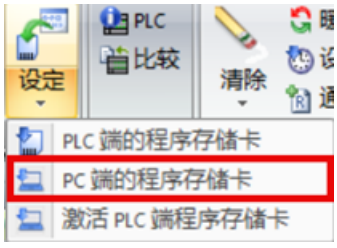


5. 如果要设置用于激活存储卡的密码，请选择“启用数据保护”(Protect Data)。
6. 单击“设置”(Setting) 按钮。
7. 如果需要密码才能对存储卡进行编程，输入密码 (页 59)。

在 PC 上创建程序传送存储卡

要将程序保存至计算机，请执行以下步骤：

1. 单击“设定”(Setting) 按钮下的向下箭头，然后选择“PC 端的程序存储卡”(Program Memory Card in PC)。



2. 选择将以下哪些（或全部）块存储于存储卡：
   – 项目
   – 内部证书

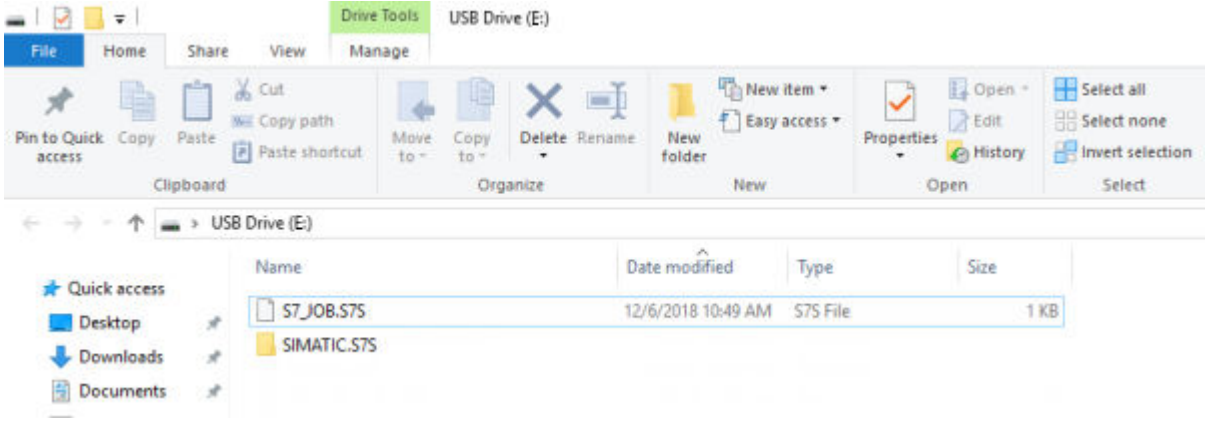
3. 单击“浏览”(Browse) 按钮，然后导航至 SD 卡的根文件夹。

4. 如果要设置用于激活存储卡的密码，请选择“启用数据保护”(Protect Data)。

说明

如果选择“启用数据保护”(Protect Data)，则需稍后激活存储卡。

5. 单击“保存”(Save) 按钮，在 SD 卡中生成两个文件，如下所示：



说明

PLC 可能无法通过“PC 端的程序存储卡”(Program Memory Card in PC) 编译从 SD 卡恢复的程序。为确保您的程序/组态有效，Siemens 建议您连接到 PLC 并至少下载一次程序。

说明

PC 中创建的存储卡的不包含程序中的任何强制组态。

在 PLC 中激活存储卡

如果在创建卡时选择“启用数据保护”(Protect Data)，请按以下步骤激活卡。

1. 将存储卡插入 PLC。
2. 单击“设定”(Setting) 按钮下的向下箭头，然后选择“激活 PLC 端程序存储卡”(Activate Memory card in PLC)。
3. 输入创建卡片时设置的密码。

从程序传送存储卡恢复程序

要将程序传送卡的内容复制到 PLC，必须在插入程序传送卡的情况下对 CPU 循环上电。然后 CPU 执行以下任务：

1. 清空 RAM
2. 将用户程序、系统块（PLC 组态）以及数据块从存储卡复制到 CPU 永久存储器。

复制操作进行过程中，S7-200 SMART CPU 上的 STOP 和 RUN LED 交替闪烁。S7-200 SMART CPU 完成复制操作后，LED 停止闪烁。

说明

程序传送卡兼容性

恢复在不同 CPU 型号上创建的程序传送卡可能会因型号不同而失败。恢复过程中，CPU 验证存储于存储卡的程序内容的以下特性：

- 程序块大小
- 在数据块中指定的 V 存储器大小
- 在系统块 (页 211)中组态的板载数字量 I/O 数量
- 在系统块组态的每个保持范围
- 系统块中的扩展模块和信号板组态
- 系统块中的运动轴组态
- 强制的存储器位置

说明

传送由 STEP 7-Micro/WIN SMART V3 或更高版本创建的程序时，请注意以下几点：

- 在 STEP 7-Micro/WIN SMART V3 中创建的 Web 服务器和内部证书组态不能传送到版本低于 3.0 的 CPU 中。
- 如果选择了内部证书，则将在目标 CPU 中生成内部证书。自证书生成之日起，有效期为一年。
- 如果停用 Web 服务器，则内部证书为灰色，且无法存储。

说明

除了将存储卡用作程序传送卡外，还可创建复位为出厂默认存储卡 (页 241)。

5.3.7 上电后恢复数据

循环上电后 CPU 执行以下操作：

- 从永久存储器中恢复程序块和系统块
- 恢复保持性存储器分配
- 根据永久存储器中的数据块内容来恢复 V 存储器的非保持性部分
- 清空其它存储区的非保持性部分

5.4 更改 CPU 的工作模式

CPU 有以下两种工作模式：STOP 模式和 RUN 模式。CPU 正面的状态 LED 指示当前工作模式。在 STOP 模式下，CPU 不执行程序，用户可以下载程序。在 RUN 模式下，CPU 执行程序；但用户仍可下载程序。

将 CPU 置于 RUN 模式

1. 在 PLC 菜单功能区或程序编辑器工具栏中单击 “运行” (RUN) 按钮：
2. 收到提示后，单击 “确定”(OK) 更改 CPU 的运行模式。

可监视 STEP 7-Micro/WIN SMART V3 中的程序，方法是在 “调试”(Debug) 菜单功能区或程序编辑器工具栏中单击 “程序状态”(Program Status) 按钮。STEP 7-Micro/WIN SMART V3 将显示指令值。

将 CPU 置于 STOP 模式

若要停止程序，需单击 “停止”(STOP) 按钮，并确认有关将 CPU 置于 STOP 模式的提示。也可在程序逻辑中包括 STOP 指令 (页 451)，以将 CPU 置于 STOP 模式。

5.5 状态 LED

CPU 和 EM 使用 LED 提供有关运行状态的信息。

CPU 状态 LED

CPU 提供以下 LED 状态指示灯：

状态	LED 状态	说明
STOP	STOP：开 RUN、ERROR：灭	当 CPU 处于 STOP 模式时适用
STOP 带有强制值	RUN：灭 STOP：以 1 Hz 的频率闪烁 ERROR：灭	当 CPU 处于 STOP 模式且值被强制时适用
STOP，带 PROFINET 功能	STOP：开 RUN：灭 ERROR：以 1 Hz 的频率闪烁	CPU 处于 STOP 模式且任何已组态的 PROFINET 设备失去连接或收到警告时适用。
RUN	RUN：开 STOP、ERROR：灭	当 CPU 处于 RUN 模式时适用
RUN 带有强制值	RUN：开 STOP：以 1 Hz 的频率闪烁 ERROR：灭	当 CPU 处于 RUN 模式且值被强制时适用
RUN，带 PROFINET 功能	STOP：灭 RUN：开 ERROR：以 1 Hz 的频率闪烁	CPU 处于 RUN 模式且任何已组态的 PROFINET 设备失去连接或收到警告时适用。
Busy	STOP、RUN：以 2 Hz 的频率异相闪烁 ERROR：灭	当接电或重启过程中完成卡评估后，正在处理存储卡或正在重启时适用
已插入存储卡	STOP：以 2 Hz 的频率闪烁 RUN、ERROR：灭	将存储卡插入接电的 CPU 时适用
存储卡正常	STOP：以 2 Hz 的频率闪烁 RUN、ERROR：灭	当接电或重启过程中完成存储卡评估后，成功完成存储卡操作时适用。
存储卡错误	STOP、ERROR：以 2 Hz 的频率同相闪烁 RUN：灭	当接电或重启过程中完成存储卡评估后，存储卡操作因出现错误而终止时适用。

状态	LED 状态	说明
故障	STOP、ERROR：开 RUN：灭	当 CPU 处于故障模式时适用
Ping	STOP、RUN：以 1 Hz 的频率异相闪烁 ERROR：与 RUN 指示灯同相闪烁	当 CPU 接收到信号 DCP 控制请求（闪烁的 LED 指示灯）时适用

EM 状态 LED

扩展模块 (EM) 提供以下 LED 状态指示灯：

各数字量 EM 提供一个 DIAG LED，用于显示模块的状态：

- 绿色指示模块处于运行状态
- 红色指示模块有故障或处于非运行状态

各模拟量 EM 为每个模拟量输入和输出提供一个 I/O 通道（Channel） LED。

- 绿色指示通道已组态且处于激活状态

此外，各模拟量 EM 还提供 DIAG LED，可指示模块的状态：

- 绿色指示模块处于运行状态
- 红色指示模块有故障或处于非运行状态

EM 可检测模块的通断电情况（必要时，还可检测现场侧电源）。

表格 5-26　模拟量模块 (EM) 的 LED

说明	DIAG （红色/绿色）	I/O 通道 （Channel） （绿色）
现场侧电源关闭	呈红色闪烁	灭
没有组态，更新在进行中或断线检测	呈绿色闪烁	灭
模块已组态且没有错误	亮（绿色）	亮（绿色）
错误状态	呈红色闪烁	-
I/O 错误（启用诊断时）	-	灭
I/O 错误（禁用诊断时）	-	亮（绿色）

表格 5-27　数字量模块 (EM) 的 LED

说明	DIAG （红色/绿色）	I/O 通道 （Channel） （绿色）
没有组态，更新在进行中	呈绿色闪烁	• 逻辑 1： 亮 • 逻辑 0： 灭
模块已组态且没有错误	亮（绿色）	
错误状态	呈红色闪烁	

表格 5-28　热电偶和 RTD 模块 (EM) 的 LED

说明	DIAG （红色/绿色）	I/O 通道 （Channel） （绿色）
没有组态，更新在进行中或断线检测	呈绿色闪烁	灭
模块已组态且没有错误	亮（绿色）	亮（绿色）
错误状态	呈红色闪烁	-
I/O 错误（启用诊断时）	-	灭
I/O 错误（禁用诊断时）	-	亮（绿色）

编程概念 6

6.1 设计 PLC 系统的指南

设计 PLC 系统有很多种方法。 以下这些通用的指南适用于许多设计项目。 当然，您还必须遵守您所在公司的规程以及您在培训中和现场积累的实践经验。

分解过程或机器

将您的过程或者机器分解成相互独立的部分。 这些独立部分决定了控制器之间的界限，并将影响功能描述规范和资源的分配。

创建功能规范

写出过程或者机器每一部分的操作描述。 包括下列主题： I/O 点、操作的功能描述、允许每个执行器（例如螺线管、电机和驱动器）动作之前必须达到的状态、操作员界面的描述以及与过程或机器其它部分相连的任何接口的描述。

设计安全电路

出于安全考虑，应识别出需要硬接线逻辑的设备。 控制设备若发生故障可能出现不安全状况，造成机器意外启动或运行变化。 若是意外或错误的机械运转可能导致人员身体受伤或重大财产损失，应考虑使用独立于 CPU 运行的机电超驰装置，以防止不安全的运行。

安全电路的设计中应包含以下任务：

- 确定可能造成危险的不正确或意外的执行器操作。
- 确定可确保操作不危险的条件，并确定如何独立于 CPU 检测这些条件。
- 确定上电和断电时 CPU 和 I/O 如何影响过程，并确定检测错误的时间。 此信息仅用于设计正常和可预期的异常操作，不能用于保障安全的目的。
- 设计独立于 CPU 的手动或机电安全超驰来阻止危险的操作。
- 向 CPU 提供独立电路的相应状态信息，便于程序和任何操作员界面都获得必需的信息。
- 标识其它与过程安全操作相关的安全要求。

指定操作员站

根据功能规范的要求创建操作站的组态图。 包括以下几项：

- 显示与过程或者机器有关的每个操作站的位置总览图
- 操作站中设备（如显示器、开关和灯）的机械布局
- 包含 CPU 或扩展模块中相关 I/O 的电气图

创建组态图

根据功能规范的要求创建控制设备的组态图。 包括以下几项：

- 显示与过程或机器相关的每个 CPU 的位置总览图
- CPU 和扩展 I/O 模块的机械布局（包括机柜和其它设备）
- 每个 CPU 和扩展 I/O 模块的电气图（包括设备模型号、通信地址和 I/O 地址）

创建符号名称列表（可选）

如果选择使用符号名称进行寻址，需要对绝对地址创建一个符号名称列表。 不仅要包含物理 I/O 信号，也要包含程序中要用到的其它元素。

6.2 用户程序的元素

程序组织单元 (POU) 由可执行代码和注释组成。可执行代码由主程序、若干功能块和若干子程序或中断程序组成。代码已编译并下载到 CPU 中。可以使用程序组织单元（主程序、子程序、功能块和中断程序）来组织用户程序的结构。

- 用户程序主体包括控制应用的指令。CPU 将按顺序执行这些指令，每个扫描周期执行一次。
- 子例程是只有在调用时才执的程序的可选元素：由主程序、中断例程或另一子例程执行。当您希望重复执行某种功能时，子例程是非常有用的与其在主程序中每个需要使用该功能的位置多次写入相同的程序代码，不如将这段逻辑写在子例程中，然后根据需要在主程序中调用该子例程。子例程具有以下优点：
  - 使用子例程可以减小程序的大小。
  - 由于已将代码移出主程序，因而使用子例程可以缩短扫描时间。CPU 在每个扫描周期都会评估主程序中的代码，不管代码是否执行，而 CPU 仅在调用子例程时评估其代码，如果扫描时不调用子例程，CPU 不会评估其代码。
  - 使用子例程创建的代码是可移植的。您可以在一个子例程中完成一个独立的功能，然后将该子例程复制到另其它程序中，无需进行重复工作。

  说明

  使用 V 存储器地址会限制子例程的可移植性，因为一个程序对于 V 存储器地址的分配有可能与另一个程序对它的分配有冲突。相比之下，在子例程中为全部地址分配使用局部变量表（L 存储器）会使子例程具有极高的可移植性，因为当子例程使用局部变量时，子例程与程序的其它部分之间就不会有地址冲突。

- 功能块为可重复使用的代码块，配有关联的存储区。对于未在一个扫描周期内完成运行的任务或设备，用户通常使用 FB 控制其运行。用户程序中的每一个 FB 都设有一个或多个 FB 实例来存储运行参数，以便从一次扫描切换到下一次扫描时，可快速访问运行参数。调用 FB 时，还需要指定包含块参数以及该调用或“FB 实例”的静态局部数据的 FB 实例。FB 执行完毕后，FB 实例将保留这些值。通过设计通用控制任务 FB，可为多个设备重复使用 FB，方法是：为 FB 的不同调用选择不同的 FB 实例。一个 FB 会将输入、输出、输入/输出和静态参数存储到一个 FB 实例中。

- 中断例程是程序的可选元素，发生特定中断事件时，中断例程会进行响应。您可以设计一个中断例程来处理预先定义好的中断事件。当指定事件发生时，CPU 会执行该中断例程。

  中断例程不会被主程序调用。只有当中断例程与一个中断事件相关联，并且在该中断事件发生时，CPU 才会执行中断例程中的指令。

  说明

  由于无法预测 CPU 何时会产生中断，所以应考虑尽量限制中断例程和程序中其它部分所共用的变量个数。

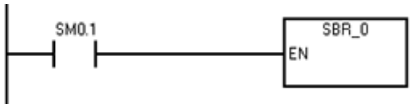
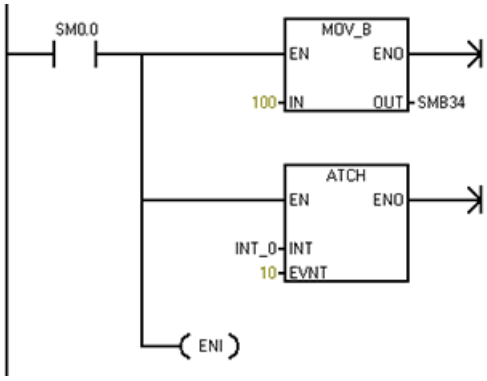
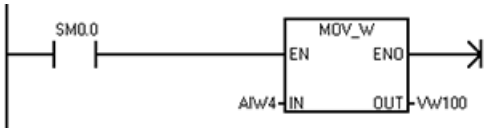
  使用中断例程的局部变量表可确保中断例程仅使用临时存储器，从而不会覆盖程序其它位置使用的数据。

  为了保证主程序与中断例程正确地共享数据，您可以使用许多编程技巧。请参见中断指令(页 413)的说明。

- 其它块中包含 CPU 的信息。
  - 系统块：系统块允许您为 CPU 组态不同的硬件选项。
  - 数据块：DB 用于存储程序使用的不同变量的初始值。

下例中给出了一段包含子例程和中断例程的程序。此示例程序使用定时中断，每 100 ms 读取一次模拟量输入值。

表格 6-1　包含子例程和中断例程的示例程序

主程序		**Network 1** **LD SM0.1** **CALL SBR_0**	首次扫描时，调用子例程 0。
SBR 0		**Network 1** **LD SM0.0** **MOVB 100, SMB34** **ATCH INT_0, 10** **ENI**	将定时中断的时间间隔设置为 100 ms。 启用中断 0。
INT 0		**Network 1** **LD SM0.0** **MOVW AIW4,VW100**	对模拟量输入 AI4 的值进行采样。

6.3 创建用户程序

STEP 7-MicroWIN SMART V3 用户界面提供了一个便捷的工作环境来创建用户项目程序。（STEP 7-MicroWIN SMART V3 项目是带有 .smartV3 文件扩展名的文件。）

要打开用户界面，请双击 STEP 7-MicroWIN SMART V3 图标，或者从 “开始”(Start) 菜单的 “西门子自动化”(Siemens Automation) 组件中选择“STEP 7-MicroWIN SMART V3”。

6.3.1 早期版本的 STEP 7-Micro/WIN 项目

S7-200 SMART G2 作为新一代产品，在硬件层面进行了全面升级，包括 CPU、扩展模块和信号板。其中，S7-200 SMART G2 CPU 显著提升了扩展能力，现在可支持多达 16 个扩展模块。

由于 S7-200 SMART G2 CPU 的型号及其支持的 I/O 点数与旧版本存在差异，您在将现有工程迁移至 S7200 SMART G2 CPU 时，必须对系统块中配置的扩展模块和信号板进行替换，以确保其与 S7200 SMART G2 兼容。

STEP 7-Micro/WIN SMART V3 （3.2 版）允许您将 STEP 7-Micro/WIN SMART （2.8 版）或 STEP 7-Micro/WIN SMART V3（3.0 版）的工程及组态迁移至 S7-200 SMART G2 系列。在迁移过程中，原版本工程中所使用的 CPU、信号板（SB）、扩展模块（EM）将自动被替换为 S7-200 SMART G2 对应的 CPU、信号板（SB）、扩展模块（EM）。如果在迁移后发现 I/O 点数不足，系统会自动添加额外的 S7-200 SMART G2 扩展模块来补足所需的 I/O 容量。

迁移后，STEP 7-Micro/WIN SMART V3 能够准确读取并显示其模块的 PLC 信息、报警和错误等信息。迁移后工程文件保存后，为确保兼容性，仅支持 STEP 7-Micro/WIN SMART V3（3.1 版）及以上版本软件打开。

在 STEP 7-Micro/WIN SMART V3（3.2 版）中使用 STEP 7-Micro/WIN SMART （2.8 版）或 STEP 7-Micro/WIN SMART V3（3.0 版）中创建的工程，请按以下步骤操作：

- 在 “文件”(File) 菜单功能区的 “操作”(Operations) 部分，单击 “迁移”(Migrate) 按钮，或者按下 Ctrl+M 快捷键组合。

迁移完成后，CPU 版本将更新至 S7-200 SMART G2。

说明

只有使用 STEP 7-Micro/WIN SMART（2.8 版）或 STEP 7-Micro/WIN SMART V3（3.0 版）创建的项目才能迁移到 3.2 版。对于在 STEP 7-Micro/WIN SMART（2.8 版）之前版本中创建的项目，需要在 2.8 版 中打开并保存，然后将其迁移到 STEP 7-Micro/WIN SMART V3（3.2 版）。

说明

STEP 7-Micro/WIN SMART V3 在迁移后尽量保持原程序的正常功能。注意输出窗口中的消息

说明

重命名与新添加保留关键字冲突的变量名的符号名称

如果迁移项目的符号表中的符号名称或者变量表中的变量名称与 STEP 7-Micro/WIN SMART V3 引入的关键字冲突（包括新增的系统 POU 名称、新的指令名称以及新增的预留关键字），这些变量将变为红色，表示迁移过程中出现错误。请及时对这些名称进行重命名，以免发生问题。

POU

名称

- 如果迁移项目中的 SBR/INT 名称与 STEP 7-Micro/WIN SMART V3 新增系统 POU 名称冲突，则迁移后，将原名称后添加"_SBR"，重命名为"OriginalName_SBR"。如果重命名后仍然存在重复，则将它们重命名为"OriginalName_SBR_ID##"。

局部存储器

- 原始项目中的 LB 60 到 LB 64 映射到迁移项目中的 LB 1020 到 LB 1024。

指令

在 STEP 7-Micro/WIN SMART V3 中，下列指令在 STL 中支持三个参数，原指令名称改为"原名 (V2)"，在原名称前添加 (V2)。

- 加法、减法、乘法和除法 (页 393)
- 产生双整数的整数乘法和带余数的整数除法 (页 397)
- AND、OR、XOR、NAND、NOR 和 NXOR (页 431)
- 移位和循环移位 (页 454)

原有的二参数指令位于项目树中相应文件夹的末尾，命名为"原有"。

安全设置

关于迁移后的安全设置，请参见下表。

STEP 7-Micro/WIN SMART 2.8 版		STEP 7-Micro/WIN SMART V3	
系统块 > 安全	• 完全权限	PLC 安全设置	禁用 PLC 访问控制
	• 读取权限 • 最低权限 • 不允许上传		启用 PLC 访问控制 • 读取权限 • 最低权限 • 不允许上传
无		PLC 安全设置	通信设置 • 默认情况下启用 Put/Get • 默认情况下禁用通过 HMI 的安全通信

请注意以下几点：

- 迁移后，“系统块 > 通信 > LLDP”(System Block > Communication > LLDP) 将会启用。
- 迁移后，“系统块 > 通信 > SNMP”(System Block > Communication > SNMP) 将会启用。

将符号表迁移到全局变量表

将在 STEP 7-Micro/WIN SMART（2.8 版） 中创建的项目迁移到 STEP 7-Micro/WIN SMART V3（3.2 版） 时，将传送符号、常数和变量，如下所述：

- 具有常量符号的符号表分为两个表，常量置于常量表中，其余内容置于全局变量表中。这意味着，如果项目的符号表“Table_1”中有常量符号需要迁移，则在 STEP 7-Micro/WIN SMART V3 中添加一个名为“Table_1_Const”的新表。随后，“Table_1”中的常量符号会移至“Table_1_Const”。
- 迁移后，系统符号表的名称不变。
- 迁移后的系统变量表仅包含 STEP 7-Micro/WIN SMART V3 版本中新增的系统变量。可使用功能“更新系统变量表”恢复丢失的系统符号。
- 如果 DB 定义的存储器（有效 DB 行）和全局变量表中的变量（有效变量地址）之间存在重叠或完全匹配，并且如果全局变量的初始值为 0，则禁用该变量的初始值单元格。在这种情况下，用户只能在 DB 中设置变量的初始值。
- 迁移前，表属性和可编辑字段具有一致的可编辑属性。
- 变量的数据类型为空。

迁移后，将传送符号表，如下所示：

STEP 7-Micro/WIN SMART（2.8 版）中的 Symbol table_xx	STEP 7-Micro/WIN SMART V3 中的全局变量表	
符号名称	变量名	symbol table_xx
地址	绑定	
注释	变量的注释	
常数符号	常数名称	constant table_xx[1]

[1] 如果原始程序中没有常量，将创建一个默认常量表。

说明

- 变量的初始值为 0。仅对于用户级变量表和库级变量表，默认值才会显示并可编辑。
- 编辑 DB 和变量时，系统将检查二者之间是否有任何地址重叠，若有，则变量单元无法编辑。
- 将符号迁移到变量后，ValidFlag 保持不变（加下划线或高亮显示）。

FB 实例表

迁移后，将创建默认 FB 实例表。

局部变量

- 如果原始项目的 SBR 的局部变量表包含字符串变量，则此类变量迁移为 STRING<254>。调用此 SBR 时，如果与字符串变量对应的操作数是一个变量，则该变量必须为 STRING<254>。

向导

- **运动**：STEP 7-Micro/WIN SMART V3 可将 2.8 版的项目迁移到 STEP 7-Micro/WIN SMART V3，并可使用运动向导或全新运动控制对项目进行修改。有关详细信息，请参见“运动控制 (PLCopen) (页 805)”。
- **HSC**：在 STEP 7-Micro/WIN SMART（2.8 版）中创建的带有 HSC 的项目可像以前一样直接在 STEP 7-Micro/WIN SMART V3 中打开和使用。

- **PROFINET**
  - 迁移后， PROFINET 网络结构会转换到硬件网络拓扑结构。
  - 迁移后，智能设备将支持两个以太网端口。
  - 如果在智能设备中选中了“PROFINET 接口- 上级控制器分配的参数”(PROFINET Interface- Parameters Allocated by Upper-Level- Controller) 复选框，请注意以下事项：Profinet 控制器中组态的 IO 设备版本必须与实际连接的智能设备版本匹配。
  - 如果原始程序组态为智能设备，智能设备描述将更新至 S7-200 SMART G2。
- **Web 服务器**

说明

迁移后 Web 服务器用户丢失

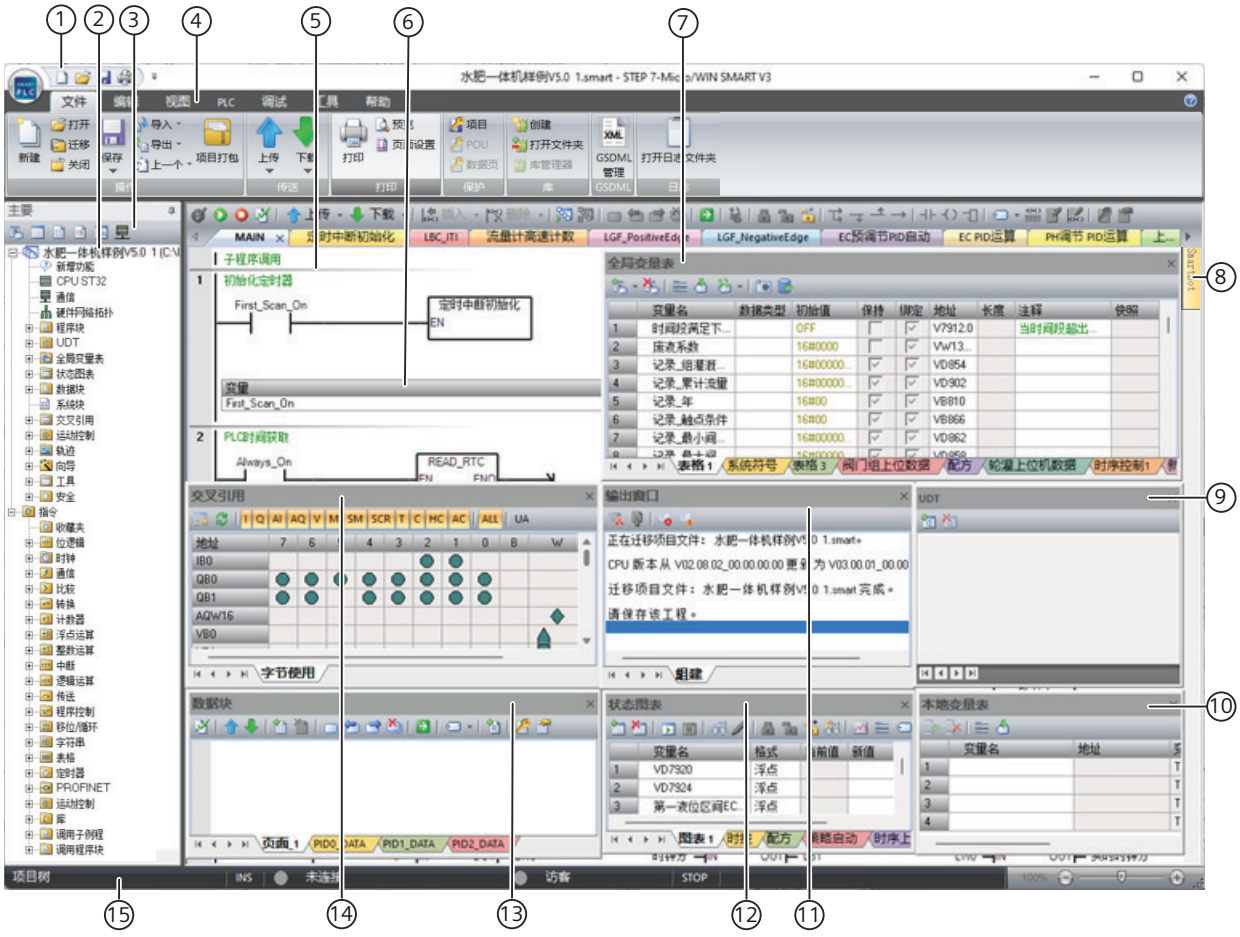
如果 Micro/WIN SMART（2.8 版）中的项目组态了 Web 服务器用户，在迁移后，用户将丢失。

6.3.2 使用 STEP 7-Micro/WIN SMART 用户界面

STEP 7-Micro/WIN SMART V3 是 S7-200 SMART CPU 固件版本 V3.0 (ST/SR40,ST/SR60) 和 V3.1/V3.2 (ST32) 的组态、编程和操作软件。一次可将 STEP 7-Micro/WIN SMART V3 的一个实例连接至一个 S7-200 SMART CPU。

屏幕区域和元素

STEP 7-Micro/WIN SMART V3 用户界面提供多个窗口，可用来排列、编程和监视。请注意，每个窗口均可按您所选择的方式停放或浮动以及排列在屏幕上。您可单独显示每个窗口，也可合并多个窗口以从单独选项卡访问各窗口。



① 快速访问工具栏 (页 160)

② 项目树 (页 160)

③ 导航栏 (页 160)

④ 菜单 (页 160)

⑤ 程序编辑器 (页 160)

⑥ 变量信息表

⑦ 全局变量表 (页 167)

⑧ SmartBot

⑨ UDT (页 177)

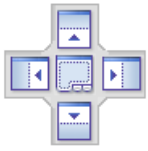
⑩ 本地变量表 (页 171)

⑪ 输出窗口 (页 160)
⑫ 状态图表 (页 777)
⑬ 数据块 (页 165)
⑭ 交叉引用 (页 771)
⑮ 状态栏 (页 160)

菜单功能区

STEP 7-Micro/WIN SMART V3 显示每个菜单的菜单功能区。可通过右键单击菜单功能区并选择“最小化功能区”(Minimize the Ribbon) 的方式最小化菜单功能区，以节省空间。

移动窗口



要移动窗口，请单击窗口框架，然后开始拖动。移动时，会看到定位器符号，它能帮助您将窗口停靠在当前窗口的左侧、右侧、顶部或底部。还可以在 STEP 7-Micro/WIN SMART V3 用户界面内使用左、右、上、下定位器符号。可以使用这些符号将窗口停放在 STEP 7-Micro/WIN SMART V3 四个区域中的一个区域内。通过定位器符号移动窗口时，STEP 7-Micro/WIN SMART V3 会以阴影指示将其拖放到该位置时该窗口的位置。也可将窗口单独拖放到任何浮动位置。

以选项卡形式显示多个窗口

要以单个窗口的选项卡形式显示多个窗口，应将窗口拖动到另一窗口之上。STEP 7-Micro/WIN SMART V3 然后合并这两个窗口，每个窗口各对应一个选项卡。以下示例显示以下窗口组合的选项卡：

- 变量表
- 交叉引用表
- 输出窗口

说明

STEP 7-Micro/WIN SMART V3 会记住上一会话的布局，并恢复已在上一位置打开的窗口。

6.3.3 使用 STEP 7-Micro/WIN SMART 创建程序

快速访问工具栏

快速访问工具栏显示在菜单选项卡上方。通过快速访问文件按钮，可快速方便地访问“文件”(File) 菜单的大部分功能，并可访问最近打开的文档。快速访问工具栏上的其它按钮对应于文件功能“新建”(New)、“打开”(Open)、“保存”(Save) 和“打印”(Print)。

项目树

项目树显示所有的项目对象和创建控制程序需要的指令。您可以将单个指令从树中拖放到程序中，也可以双击指令，将其插入项目编辑器中的当前光标位置。

项目树对项目进行组织：

- 右键单击项目，设置项目密码或项目选项
- 右键单击“程序块”(Program Block) 文件夹，插入新的子程序、功能块和中断程序。
- 打开“程序块”(Program Block) 文件夹，然后右键单击 POU 可打开 POU、编辑其属性、用密码对其进行保护或重命名。
- 右键单击“状态图”(Status Chart) 或“全局表”(Global Table) 文件夹，插入新图或新表。
- 打开“状态图”(Status Chart) 或“全局表”(Global Table) 文件夹，在指令树中右键单击相应图标，或双击相应的 POU 选项卡对其执行打开、重命名或删除操作。

说明

提高了项目、POU 和数据块（数据页）密码的安全性

与之前版本相比，STEP 7-Micro/WIN SMART V2.3 增强了密码的安全性。如果正在使用之前版本 STEP 7-Micro/WIN SMART V3 创建的项目，则要重新输入密码才能激活增强安全性。

导航栏

导航栏显示在项目树上方，可快速访问项目树上的对象。单击一个导航栏按钮相当于展开项目树并单击同一选择内容。导航栏有几组图标，用于访问 STEP 7-Micro/WIN SMART V3 的不同编程功能。

菜单功能区

STEP 7-Micro/WIN SMART V3 显示每个菜单的菜单功能区。可通过右键单击菜单功能区并选择“最小化功能区”(Minimize the Ribbon) 的方式最小化菜单功能区，以节省空间。

程序编辑器

程序编辑器包含程序逻辑和变量表，您可在该表中为临时程序变量分配符号名称。子程序、功能块和中断程序以选项卡的形式显示在程序编辑器窗口顶部。单击这些选项卡可以在子例程、中断和主程序之间切换。

STEP 7-Micro/WIN SMART V3 提供了三个编辑器来创建程序：

- 梯形图 (LAD)
- 语句表 (STL)
- 功能块图 (FBD)

尽管有一定限制，但是用任何一种程序编辑器编写的程序都可以用其它程序编辑器进行浏览和编辑。

可以在“视图”(View) 菜单功能区的“编辑器”(Editor) 部分将编辑器更改为 LAD、FBD 或 STL。通过“工具”(Tools) 菜单功能区“设置”(Settings) 区域内的“选项”(Options) 按钮，可组态启动时的默认编辑器。

状态栏

状态栏位于主窗口底部，用于提供在 STEP 7-Micro/WIN SMART V3 中执行的操作的编辑模式或在线状态的相关信息。

输出窗口

“输出窗口”显示最近编译的 POU（第 843 页）和在编译过程中出现的错误的清单。如果已打开“程序编辑器”窗口和“输出窗口”，可双击“输出窗口”中的错误信息使程序自动滚动到错误所在的程序段。

6.3.4 借助向导创建控制程序

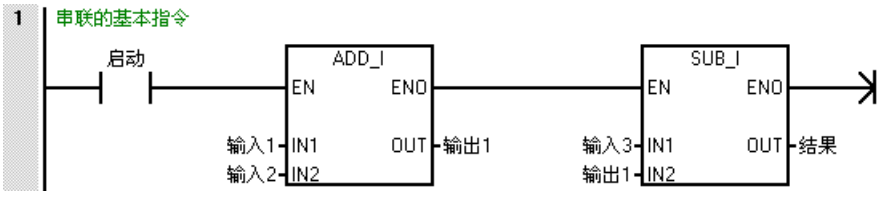
STEP 7-Micro/WIN SMART V3 提供了以下向导，让编程的各环节更简单、更自动化：

- 高速计数器
- 运动 (V2.x)
- PID
- PWM（脉宽调制）
- Get/Put
- 数据日志（仅限标准 CPU）
- Web 服务器

要启动向导，请在 STEP 7-Micro/WIN SMART V3 的“工具”(Tools) 菜单功能区或在项目树中的向导节点中选择相应向导。打开向导后按下 F1，便可从在线帮助系统中获取有关向导的详细信息。

6.3.5 梯形图编辑器

STEP 7-Micro/WIN SMART V3 梯形图 (LAD) 编辑器允许建立与电子线路图相似的程序。梯形图编程是很多 PLC 程序员和维护人员选用的方法；它是为新程序员设计的优秀语言。基本上，梯形图程序允许 CPU 仿真来自一个动力源的电流，通过一系列逻辑输入条件，然后启用逻辑输出条件。逻辑分解为程序段。程序根据指示执行，每次执行一个程序段，顺序为从左至右，然后从顶部至底部。一旦 CPU 到达程序的结尾，又回到程序的顶部重新开始。



各种指令通过图形符号表示，包括三个基本形式。

	触点	表示逻辑输入条件，用于模拟开关、按钮、内部条件等。
	线圈	通常表示逻辑输出结果，用于模拟灯、电机启动器、干预继电器、内部输出条件及其他输出。
	方框	表示附加指令，例如定时器、计数器或数学指令。

可在梯形逻辑中建立的程序段从简单到极复杂。您可以创建带有中线输出的程序段；您甚至可以串联多个方框指令。可串联的方框指令通过“使能输出”(ENO) 线标记。如果方框在 EN

输入处具有能流且执行无错误，则 ENO 输出将能流传递到下一个元素。ENO 可用作使能位，指示指令成功完成。ENO 位用于堆栈顶端，影响用于后续指令执行的能流。

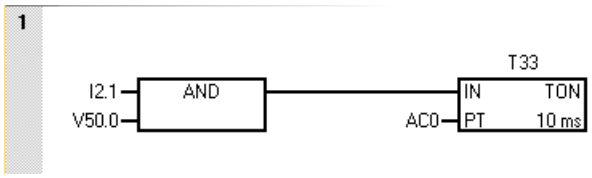
要点

选择 LAD 编辑器时要考虑的要点包括：

- 梯形逻辑便于新程序员使用。
- 图形显示通常易于理解，在全世界通行。
- 始终可以使用 FBD 编辑器 (页 163) 或 STL 编辑器 (页 164)显示使用 LAD 编辑器创建的程序。

6.3.6 功能块图编辑器

STEP 7-Micro/WIN SMART 功能块图 (FBD) 编辑器允许您以逻辑方框（与通用逻辑门图相似）形式查看指令。 FBD 中没有 LAD 编辑器中的触点和线圈，但有相等的指令，以方框指令的形式显示。 程序逻辑从这些方框指令之间的连接派生。 来自一条指令的输出（例如 AND（与）方框）可用于启用另一条指令（例如计时器），以创建必要的控制逻辑。 这一连接概念允许您象使用其他编辑器一样，很方便地解决各种逻辑问题。



如果方框在 EN 输入处具有能流且执行无错误，则 ENO 输出将能流传递到下一个元素。 ENO 可用作使能位，指示指令成功完成。 ENO 位用于堆栈顶端，影响用于后续指令执行的能流。

要点

选择 FBD 编辑器时要考虑的要点包括：

- 图形逻辑门表示样式对跟随程序流有益。
- 可使用 LAD 编辑器 (页 162) 或 STL 编辑器 (页 164)显示使用 FBD 编辑器创建的程序。
- 可扩充 AND/OR（与/或）方框便于绘制复杂的输入组合。

6.3.7 语句表编辑器

STEP 7-Micro/WIN SMART 语句表 (STL) 编辑器允许您通过使用文本形式的 STL 指令助记符和参数来创建控制程序。通常，STL 编辑器对熟悉 PLC 和逻辑编程的经验丰富的程序员更合适。STL 编辑器还允许您创建无法通过梯形逻辑或功能块图编辑器创建的程序。这是因为您是用 CPU 的本机语言在编程，而不是在图形编辑器中编程，在编辑器中必须应用一些限制以便正确编译图形。

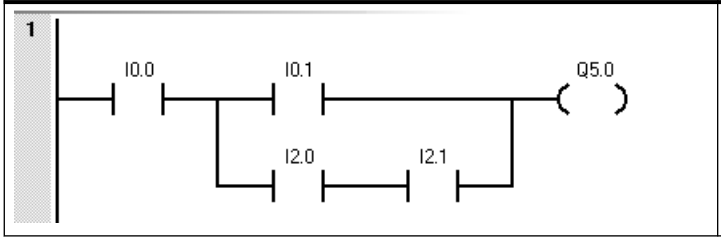
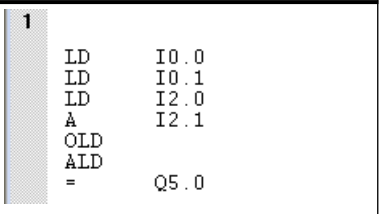
语句表程序示例

```
1
    LD      I0.0
    LD      I0.1
    LD      I2.0
    A       I2.1
    OLD
    ALD
    =       Q5.0
```

基于文本的语句与汇编语言编程相似。CPU 从上到下执行列表中的每条语句。所有程序段处理完成后，将从列表顶部重新开始执行。S7-200 SMART CPU 使用逻辑堆栈来解析 AND/OR 控制逻辑。LAD 和 FBD 编辑器将 AND/OR 图逻辑自动转换为 STL 指令，STL 指令使用逻辑堆栈执行等效逻辑。在 STL 编程中，必须使用堆栈指令并亲自处理堆栈操作。

所有 LAD 和 FBD 程序均必须成功转换为 STL 程序，因为实际下载到 PLC 中的就是 LAD 或 FBD 程序的 STL 版本。

下图显示一个 LAD 中的简单程序和 STL 中的对应程序。

LAD	STL
	

下表显示在 32 位逻辑堆栈中发生的情况：

表格 6-2　指令

堆栈	LD I0.0	LD I0.1	LD I2.0	A	OLD	ALD
S0	I0.0	I0.1	I2.0	I2.0 AND I2.1	(I2.0 AND I2.1) OR I0.1	I0.0 AND [(I2.0 AND I2.1) OR I0.1]
S1		I0.0	I0.1	I0.1	I0.0	

堆栈	LD I0.0	LD I0.1	LD I2.0	A	OLD	ALD
S2			I0.0	I0.0		
…						
S31						

要点

选择 STL 编辑器时要考虑的要点包括：

- STL 对经验丰富的程序员最适合。
- STL 有时可以解决无法用 LAD 或 FBD 编辑器轻易解决的问题。
- 通过 STL 编辑器创建的程序可能导致程序大小小于在 LAD 或 FBD 编辑器中所创建的等效控制逻辑的程序大小。
- 始终可以使用 STL 编辑器查看或编辑通过 LAD 编辑器 (页 162)或 FBD 编辑器 (页 163)建立的程序，反之则并不一定正确。无法始终使用 LAD 编辑器或 FBD 编辑器显示通过 STL 编辑器编写的程序。

6.4 数据块 (DB) 编辑器

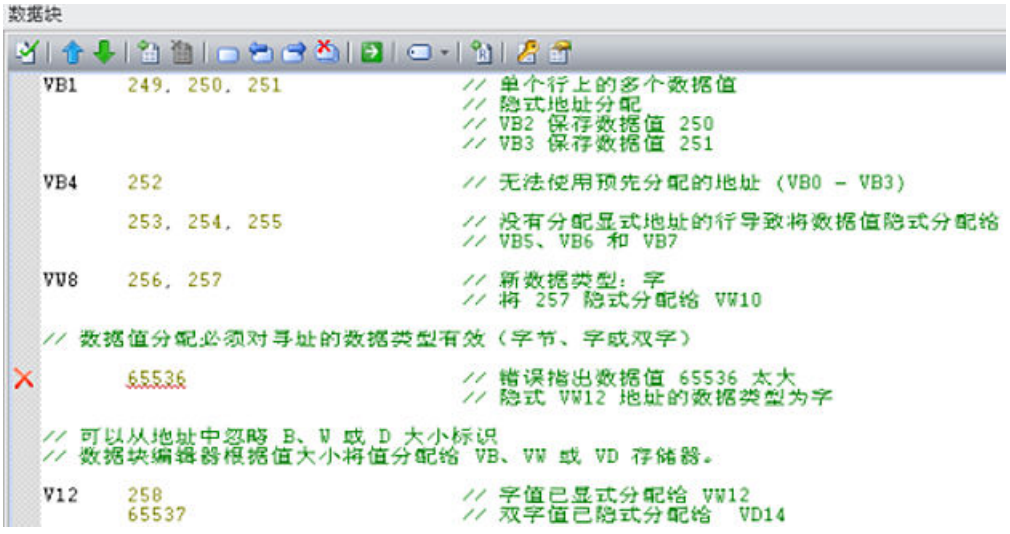
数据块允许您向 V 存储器的特定位置分配常数 (页 128)（数字值或字符串）。您可以对 V 存储区的字节 (V 或 VB)、字 (VW) 或双字 (VD) 地址赋值。还可以输入可选注释，前面带双正斜线 *//*。

- 数据块的第一行必须分配显式地址。可使用存储器地址（绝对地址）或全局变量表 (页 167)中以前分配给地址的变量名。
- 后续行可分配显式地址或隐式地址。当您在单个地址分配后键入多个数据值时，或键入仅包含数据值的一行时，编辑器会自动进行隐性地址分配。编辑器根据先前的地址分配及数据值大小（字节、字或双字），指定适当数量的 V 存储区。
- 数据块编辑器是一种自由格式文本编辑器；但是，它预期地址或变量名出现在第一个位置。如果继续输入一个隐式数据值条目，输入隐式赋值前在地址位置输入至少一个空格。键入一行后，按 ENTER 键，数据块编辑器格式化该行（对齐地址列、数据和注释；大写 V 存储区地址）并重新显示行。数据块编辑器接受大小写字母，并允许使用逗号、制表符或空格作为地址和数据值之间的分隔符。
- 完成一个赋值行后按 CTRL–ENTER，将地址自动增加至下一个可用地址。

说明

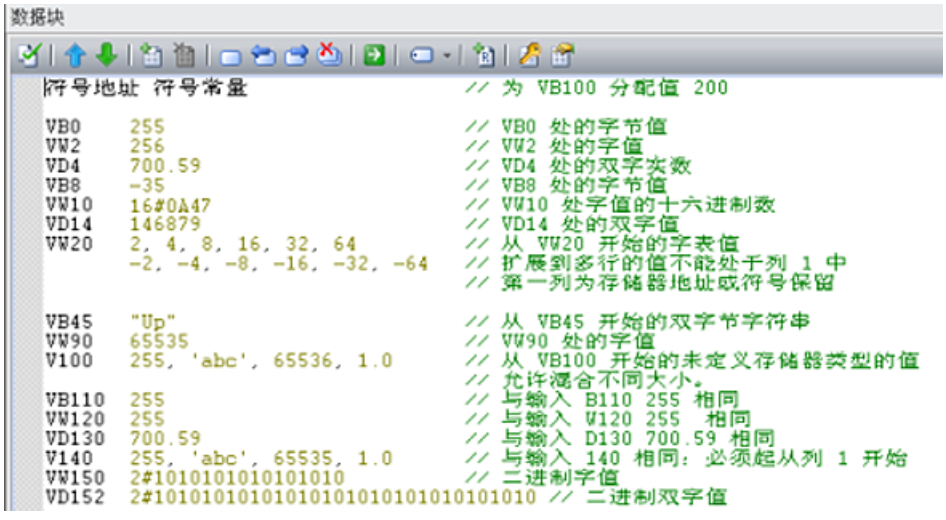
如果 DB 中使用了有数据类型的变量，必须保证 DB 中的值与数据类型一致。DB 不能使用复杂结构变量。

示例：数据块页面



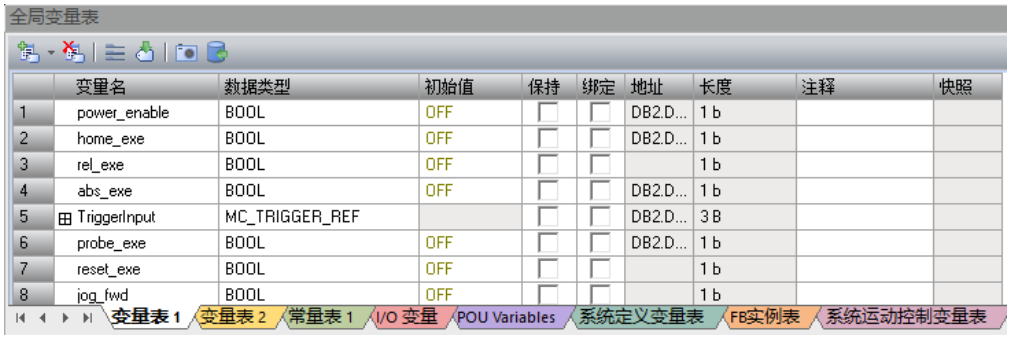
注：在输入非显性地址的行上的数据值前输入一个空格。

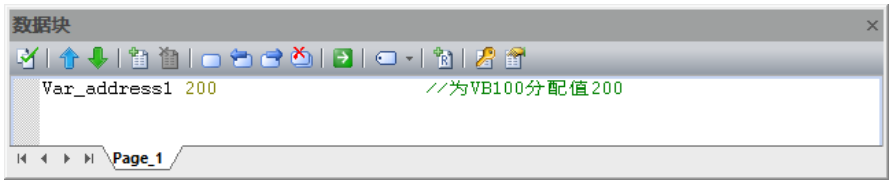
示例：直接地址和数字值



示例：全局变量地址和全局变量赋值

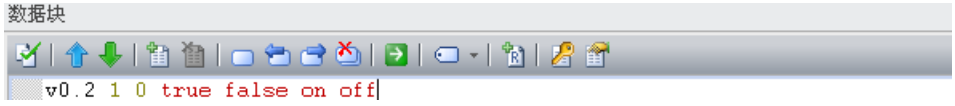
DB 中不能使用复杂结构变量。

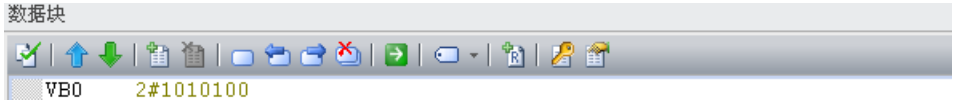




示例：另一种二进制输入方法和得到的二进制分配

可为二进制分配输入数 1 或 0，或 "true"、"false"，"on" 或 "off"（小写，大写，或大小写混合）。数据块编辑器解释输入并显示得到的二进制分配。





6.5 全局变量表

全局变量是可为存储器地址或常量指定的变量名。可为下列存储器类型创建全局变量名：I、Q、M、SM、AI、AQ、V, S、C、T、HC。在全局变量表中定义的变量适用于全局。已定义的全局变量可在程序的所有程序组织单元 (页 151) (POU) 中使用。如果在局部变量表 (页 171) 中指定变量名称，则该变量适用于局部范围。它仅适用于定义时所在的 POU。变量可在创建程序逻辑之前或之后进行定义。

打开全局变量表

要打开 STEP 7-Micro/WIN SMART V3 中的全局变量表，可使用以下方法之一：

- 在“视图”(View) 菜单的“窗口”(Windows) 区域中，从“组件”(Component) 下拉列表中选择“全局变量表”(Global Variable Table)。
- 在项目树中打开“全局变量表”(Global Variable Table) 文件夹。单击“插入全局变量表”(Insert Global Variable Table) 按钮添加新表。

在全局变量表中分配全局变量

要将全局变量分配给地址或常量值，请按以下步骤操作：

1. 打开变量表。
2. 在“名称”(Name) 列中键入变量名（例如，Input1）。变量名可包含的最大字符数为 128 个单字节字符。

   说明

   如果已选择同时显示项目操作数的变量视图和绝对视图，则程序编辑器中较长的变量名将以波浪号 (~) 截断。您可将鼠标光标放在被截断的名称上，以查看在工具提示中显示的全名。

3. 在“初始值”(Initial Value) 列中键入初始值（例如 123）。请注意，在为全局变量分配字符串常数时，需要用双引号将该字符串常量括起来。

	变量名	数据类型	初始值	保持	绑定	地址	长度	注释
1	index	INT	+0	☐	☐	DB2.D...	2 B	
2	enable	BOOL	OFF	☐	☐	DB2.D...	1 b	
3	Earlyvalue	REAL	123.0	☐	☑	VD100	4 B	
4				☐	☐			

4. 可选中“绑定”(Bind) 选项，并在地址栏中输入 PLC 的具体地址。
   注：如果未选中“绑定”(Bind) 选项，系统将在编译过程中为该变量分配一个 DB2 中的地址。
5. 可选中“保持”(Retain) 选项以保持变量。

   说明

   非绑定和绑定变量的保持

   - 如果变量未绑定到特定地址，并且选择了“保持”(Retain) 选项，则系统将为变量分配一个 DB2 地址，并确保在断电后保持数据。
   - 如果变量绑定到特定地址，保持的效果取决于系统块中的保持组态。如果系统块中只有一些地址被设置为保持，则非保持地址中的数据将在断电后丢失，从而影响保持功能的有效性。

6. 也可以键入最长为 79 个字符的注释。

可根据需要在变量表编辑器中调整列宽。

说明

可创建多个变量表；但是，在进行全局变量分配时，不可多次使用同一变量名。

语法规则和错误指示

STEP 7-Micro/WIN SMART V3 通过彩色和波浪下划线来指示错误或不完整的变量分配：

	变量名	数据类型	初始值	保持	绑定	地址	长度	注释	快照
1	2name	BOOL	OFF	☐	☑	I0.0			
2				☐	☑	VBB0			
3	Begin	BOOL	OFF	☐	☑	@			
4				☐	☐				

红色文本表示语法无效。
变量不能以数字开头。
"VBB0" 为无效地址。
Begin 为预留的字，是无效的变量名。

	变量名	数据类型	初始值	保持	绑定	地址	长度	注释	快照
1	Pump1	BOOL	OFF	☐	☑	I0.0	1 b		
2	Pump1_1	BOOL	OFF	☐	☑	I0.0	1 b		
3	Pump1_2	BOOL	OFF	☐	☑	I1.0	1 b		

绿色波浪下划线表示用法无效。

“_number”后缀会自动添加到重复的变量名中。

"Pump1" 是重复的名称。

I0.0 是重复的地址。

定义变量时应遵守以下语法规则：

- 首字符要求：变量名必须以字母或扩展字符（ASCII 128 至 ASCII 255）开头，不能以数字开头 。
- 允许字符：变量名可包含字母、数字、下划线（_）以及扩展字符（ASCII 128–255），但不允许使用特殊符号如空格、冒号、引号等非法字符。
- 长度限制：
  - 全局变量和局部变量的最大长度为 128 个字符。
  - 局部变量名称最多支持 23 个字母数字字符和下划线。
- 禁止使用关键字：不能使用编程语言中的保留字或指令名称作为变量名，例如“Begin”是预留字，属于无效变量名。
- 不能使用 “Aa+数字”， “aa+数字”，“aA+数字” 或“AA+数字 格式”。这些格式的变量名可能会被系统错误地识别为 AQ 地址。
- 唯一性要求：在同一项目中，不可重复使用相同的变量名进行全局分配，否则会导致编译错误；若出现重复名称，系统会自动添加“_number”后缀加以区分。

此外，在特定应用场景下还有额外限制：

- 轴名称最多允许 23 个字母数字字符和下划线，并且第一个字符只能是字母或下划线，也不能与现有指令名称或关键字相同。
- Web 服务器监控表的名称长度应少于 254 个字符或 127 个汉字，且不能包含特殊字符。

说明

在更正错误的变量名或地址后，按下 TAB 键、ENTER 键或箭头键来完成已编辑的更正。

间接寻址

在程序编辑器中引用全局变量时，可以像直接地址一样对全局变量名使用间接记号（& 和 *）。有关间接寻址的详细信息，请参见直接和间接寻址的相关主题。

插入附加行

使用以下方法之一可在全局变量/常量表中插入附加行：

- 右键单击全局变量/常量表中的单元格，从上下文菜单中选择“插入 > 行”(Insert > Row)。STEP 7-Micro/WIN SMART V3 将新行插入到当前位置上方。
- 在“编辑”(Edit) 菜单功能区的“插入”(Insert) 区域中，选择“行”(Row)。STEP 7-Micro/WIN SMART V3 将新行插入到全局变量/常量表中光标所在位置上方。

删除变量/常量

使用以下方法之一可在全局变量/常量表中删除附加行：

- 右键单击全局变量/常量表中的单元格，从上下文菜单中选择“删除 > 行”(Delete > Row)。STEP 7-Micro/WIN SMART V3 将删除当前位置的行。
- 在“编辑”(Edit) 菜单功能区的“删除”(Delete) 区域中，选择“行”(Row)。STEP 7-Micro/WIN SMART V3 将删除全局变量/常量表中光标所在位置的行。

对变量/常量表进行排序

可基于“名称”(Name) 或“地址”(Address) 列按字母升序或降序对变量表进行排序。在“地址”(Address) 列中，数字常量排在字符串常量之上，字符串常量又在地址之上。

要对列进行排序，可单击“名称”(Name) 或“地址”(Address) 列标题来按相应的值进行排序。要颠倒排序顺序，可再次单击该列。STEP 7-Micro/WIN SMART V3 在排序的列旁边显示一个向上或向下箭头，用于指示排序选择。

说明

可从“文件”(File) 菜单功能区的“打印”(Print) 区域打印变量表。

可通过显示变量信息表来按网络查看变量。

6.6 局部变量表

通过局部变量表，可定义对特定 POU 局部有效的局部变量。在以下情况下使用局部变量：

- 要创建不引用绝对地址或全局变量的可移值子例程/FB。
- 您要使用临时变量（声明为 TEMP 的局部变量）进行计算，以便释放 PLC 存储器。
- 要为子例程/FB 定义输入和输出。

如果以上描述对用户的具体情况不适用，则无需使用局部变量；可在全局变量表 (页 167)中定义变量值，从而将其全部设置为全局变量。

了解局部变量

可使用程序编辑器的局部变量表指定对个别子例程、功能块或中断子例程唯一的局部变量。

局部变量可用作传递至子例程/FB 的参数，并可用于增加子例程/FB 的迁移性或重新使用子例程/FB。

程序中的每个 POU (页 151) 都有自身的局部变量表，并占 L 存储器的 1k 个字节（如果在 LAD 或 FBD 中编程，则占 1008 个字节）。借助局部变量表，可对特定范围内的局部变量进行定义：局部变量仅在创建时所处的 POU 内部有效。相反，在每个 POU 中均有效的全局变量只能在全局变量表中定义。

在局部变量表中进行分配时，指定声明类型（TEMP、IN、IN_OUT、OUT 或 STATIC）和数据类型，但不要指定存储器地址；程序编辑器自动在 L 存储器中为所有局部变量分配存储器位置。

- 对于子例程，局部变量表地址分配将局部变量名称与 L 存储器地址相关联。对于 FB，仅 TEMP 与 L 存储器地址相关联。
- 主程序、子例程和中断例程的局部变量表不支持将默认值直接赋给局部变量。
- 除 TEMP 外，功能块的局部变量表支持将值直接赋给局部变量。

说明

对于子例程，PLC 不会将局部数据值初始化为零。必须在程序逻辑中初始化所用局部变量。

局部变量的声明类型

可进行的局部变量分配类型取决于在其中进行分配的 POU。主程序、中断例程、功能块和子例程可使用临时 (TEMP) 变量。只有在执行块时，临时变量才可用，块执行完成后，临时变量可被覆盖。

数据值可以作为参数与子例程/FB 间进行传递，具体如下所述：

- 如果要将数据值传递至子例程/FB，则在子例程/FB 的局部变量表中创建一个局部变量，并将其声明类型指定为 IN。
- 如果要将子例程/FB 中建立的数据值传回至调用例程，则在子例程/FB 的局部变量表中创建一个局部变量，并将其声明类型指定为 OUT。
- 如果要将初始数据值传递至子例程/FB，则执行一项可修改数据值的操作，并将修改后的结果传回至调用例程，然后在子例程/FB 局部变量表中创建一个局部变量，并将其声明类型指定为 IN_OUT。

声明类型	说明
IN	调用 POU 提供的输入参数。
OUT	返回到调用 POU 的输出参数。
IN_OUT	参数，其值由调用 POU 提供、由子例程修改，然后返回到调用 POU。
TEMP	临时保存在局部数据堆栈中的临时变量。一旦 POU 完全执行，临时变量值不再可用。在两次 POU 执行之间，临时变量不保持其值。
STATIC	仅限 FB，STATIC 用于在 FB 实例中存储静态中间结果。STATIC 只能在 FB 内部用于存储静态数据（可保持最后一个周期的值）。

局部变量的数据类型检查

将局部变量作为子例程/FB 参数传递时，在该子例程/FB 局部变量表中指定的数据类型必须与调用 POU 中值的数据类型相匹配。

示例

从主程序调用 SBR0，将称为 INPUT1 的全局变量用作子例程的输入参数。

在 SBR0 的局部变量表中，您已经将一个称为 FIRST 的局部变量定义为输入参数。

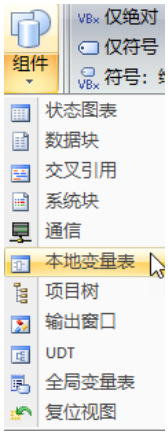
当主程序调用 SBR0 时，INPUT1 的值被传递至 FIRST。

INPUT1 和 FIRST 的数据类型必须匹配。

如果 INPUT1 是实数，FIRST 也是实数，则数据类型匹配。如果 INPUT1 是实数，但 FIRST 是整数，则数据类型不匹配，只有纠正了这一错误，程序才能编译。

查看局部变量表

要查看在程序编辑器中选择的 POU 的局部变量表，在“视图”(View) 菜单的“窗口”(Windows) 区域中，从“组件”(Component) 下拉列表中选择“局部变量表”(Local variable table)。



说明

可将局部变量表放在快速访问工具栏上以便于访问。

在局部变量表中赋值

说明

在程序中使用局部变量**之前**，先在局部变量表中赋值。在程序中使用变量名时，程序编辑器首先检查相应 POU 的局部变量表，然后检查全局变量表。如果变量名在这两处均未定义，程序编辑器则将之视为未定义的全局变量；此类变量用绿色波浪下划线加以指示。程序编辑器不会自动重新读取局部变量表并对程序逻辑做出修正。如果以后对该局部变量名的数据类型分配进行定义（在局部变量表中），必须在名称前手动插入一个井号 (#)，例如：#UndefinedLocalVar（在程序逻辑中）。因此，在使用之前声明局部变量可将编程工作量降至最低。

每个子例程调用的输入/输出参数的最大限制是 64。如果尝试下载一个超出此项限制的程序，STEP 7-Micro/WIN SMART V3 返回错误。

要在局部变量表中赋值，按以下步骤操作。

1. 确保正确的 POU 在程序编辑器窗口中显示（如有必要，通过单击所需 POU 的选项卡）。（由于每个 POU 都有自己的局部变量表，所以需要确保对正确的 POU 赋值。）
2. 如果局部变量表尚不可见，则将其显示出来，方法是在“视图”(View) 菜单的“窗口”(Windows) 区域内，从“组件”(Component) 下拉列表中选择“局部变量表”(Local variable Table)。在局部变量表中，请勿在名称前加上磅符号。井号只用在程序代码中的局部变量前。
3. 选择变量类型与要定义的局部变量类型相符的行，然后在“名称”(Name) 字段输入局部变量名。
   – 如果在主程序或中断例程中赋值，局部变量表只含 TEMP 变量。
   – 如果在子例程中赋值，局部变量表包含 IN、IN_OUT、OUT 和 TEMP 变量。
   – 如果在功能块中赋值，局部变量表包含 IN、IN_OUT、OUT、STATIC 和 TEMP 变量。

   说明

   局部变量名称最多可包含 23 个字母数字字符和下划线。也允许包含扩展字符（ASCII 128 至 ASCII 255）。第一个字符仅限使用字母和扩充字符。不允许使用关键字作为变量名，也不允许使用以数字开头的名称，或者包含非字母数字或扩展字符集中的字符的名称。

   局部变量名称下载到 CPU 存储器并存储在其中。使用较长的变量名称可能会降低可用于存储程序的存储器。

4. 在“数据类型”(Data Type) 字段中单击鼠标指针，并使用列表框为局部变量选择适当的数据类型。

   说明

   将局部变量指定为子例程/FB 参数时，必须确保分配给局部变量的数据类型不与子例程/FB 调用中正在使用的操作数发生冲突。

5. 对于功能块，可指定默认值并选择保持状态。
6. 也可提供注释，描述局部变量。
7. 如果使用 FB 局部变量表，可以设置默认值并保持变量。

输入附加的局部变量

局部变量表显示固定数目的局部变量行。要在表中添加更多行数，需在变量类型表中选择要添加的行，然后单击局部变量表窗口中的“插入”(Insert) 按钮。系统将自动在所选行的上方生成新行，其变量类型与所选变量类型相同。

还可右键单击现有行，然后从上下文菜单中选择**“插入 > 行”(Insert > Row)** 或**“插入 > 下方的行”(Insert > Row Below)** 来添加行。

删除局部变量

要删除局部变量，需在局部变量表中选中此变量，然后单击“删除”(Delete) 按钮。也可删除一行，方法是右键单击该行，然后从上下文菜单中选择**“删除 > 行”(Delete > Row)**。

折叠/展开局部变量表

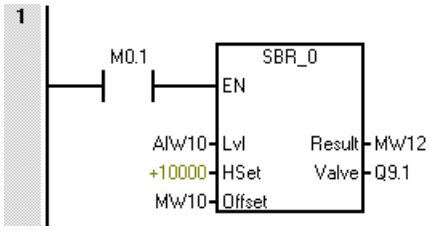
“折叠/展开”(Collapse/expand) 按钮允许折叠或展开局部变量表中的数组/UDT 类型变量。通过单击，可折叠或展开数组/UDT 变量，从而方便地查看其内容。

局部变量表示例

下例显示 SBR_0 的典型局部变量表，以及通过另一程序块对 SBR_0 的调用。

变量表

	变量名	地址	变量类型	数据类型	注释
1	EN		IN	BOOL	
2	Lv1	L0.0	IN	BOOL	Tank trans...
3	HSet	L0.1	IN	BOOL	High setpoint
4	Offset	L0.2	IN	BOOL	Offset
5			IN		
6			IN_OUT		
7	Result	L0.3	OUT	BOOL	Result value
8	Valve	L0.4	OUT	BOOL	Control value
9			OUT		
10			TEMP		



6.7 用户自定义数据类型

描述

用户自定义数据类型 (UDT) 用于变量声明。其表示由不同数据类型的多个元素组成的数据结构。因此，嵌套深度限于 4 层。

可集中更改用户自定义数据类型 (UDT)，并在程序中进行重复使用。所有使用位置都会自动更新。可在全局变量表、变量表或 FB 实例表中使用 UDT。

用户自定义数据类型的优势：

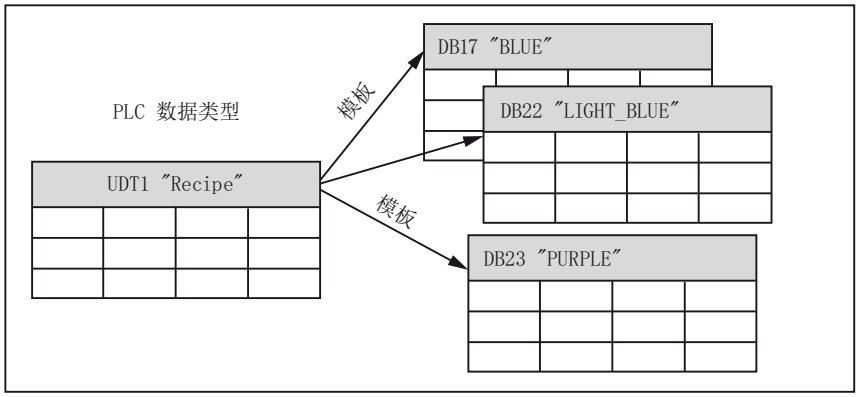
- 根据过程控制对数据进行分组
- 将参数作为数据单元传送

使用用户自定义数据类型

创建变量时，可将用户自定义数据类型声明为类型。基于此类型，可创建许多变量，所有变量都具有相同的数据结构。可通过为相应的任务输入不同的实际值来调整这些数据块。

例如，为混合涂料的配方创建用户自定义数据类型。然后，可将此数据类型分配给多个变量，每个变量都包含不同的数量信息。

下图显示了此应用程序：



用户自定义数据类型可用作模板，以创建具有相同数据结构的全局数据块。

用户自定义数据类型的嵌套深度和潜在数量

用户自定义数据类型可嵌套到深度为 4 的层级。此嵌套深度与使用的 CPU 无关。

系统定义的 UDT

为了满足特定需求，提高开发效率，系统预定义了一系列的 UDT。这些类型扩展了系统的基本数据类型，能够更灵活、更准确地表示和操作复杂的数据结构。用这些 UDT，可以简化代码编写过程，提高代码的可读性和可维护性，确保数据的一致性和完整性。

6.7.1 UDT 表

STEP 7-Micro/WIN SMART V3 预定义了多个 UDT 表，以便用户使用。还可创建自己的 UDT 表并在此表中定义 UDT。预定义 UDT 表名称以 SYS_ 开头，无法编辑。

打开 UDT 表

要打开 UDT 表，可使用以下方法之一：

- 在“视图”(View) 菜单的“窗口”(Windows) 区域中，从“组件”(Component) 下拉列表中选择“UDT 表”(UDT Table)。
- 在项目树中打开“UDT 表”(UDT Table) 文件夹，选择一个表名称；然后按下“Enter”或者双击表名称。

新增 UDT 表

要添加 UDT 表，可使用以下方法之一：

1. 在项目树中选择“UDT 表”(UDT Table)
2. 从快捷菜单中选择“插入表”(Insert Table)。
3. 在弹出式窗口中输入表名，然后单击“是”(Yes)。

或者

1. 打开 UDT 表。
2. 单击“添加”(Add) 按钮，在弹出式窗口中输入表名，然后单击“是”(Yes)。

添加 UDT 成员

1. 打开 UDT 表。
2. 添加 UDT 成员名（例如 Input1）和数据类型。
3. 也可以键入最长为 79 个字符的注释。

插入附加行

使用以下方法之一可在 UDT 表中插入附加行：

- 右键单击 UDT 表中的单元格，从上下文菜单中选择“插入 > 行”(Insert > Row)。
  STEP 7-Micro/WIN SMART 将新行插入到当前位置上方。

删除 UDT 成员

要删除 UDT 成员，可右键单击该成员，然后从快捷菜单中选择 **“删除 > 行”(Delete > Row)**。

删除 UDT 表。

要删除 UDT 表，可使用以下方法之一：

- 在 UDT 视图中选择现有 UDT 选项卡，然后使用以下方法之一将其删除：
  - 在“编辑”(Edit) 菜单功能区的“删除”(Delete) 区域，单击“对象”(Object) 下方的下拉箭头，然后从下拉菜单中选择“UDT”。
  - 单击 UDT 工具栏上的“删除”(Delete) 按钮 。
- 在项目树中单击“UDT”下的一个 UDT，然后按“删除”(Delete) 按钮。
- 在项目树中右键单击“UDT”下的一个 UDT，然后从快捷菜单中选择“删除”(Delete)。

示例

将 IO 通信的 I 点输入到变量，将变量输出到 Q 点

1. 为 I 点和 Q 点创建 UDT。应根据通信对象设置元素变量的排列和类型。

	名称	数据类型	注释
1	温度	REAL	
2	湿度	REAL	
3	浓度	REAL	
4	量	REAL	
5	次数	INT	
6			

图 6-1　I_Point

	名称	数据类型	注释
1	阀_1	REAL	
2	阀_2	REAL	
3	阀_3	REAL	
4	阀_4	REAL	
5	开始	BOOL	
6	停止	BOOL	
7			

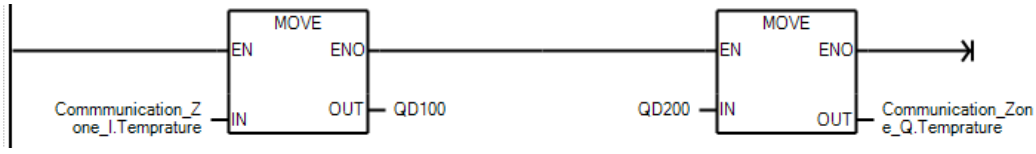
图 6-2　Q_Point

2. 使用变量表中的 UDT。

	变量名	数据类型	初始值	保持	绑定	地址	长度	注释	快照
1	日	通信区_I			✔	IB100			
2	温度	REAL			✔	ID100			
3	湿度	REAL			✔	ID104			
4	浓度	REAL			✔	ID108			
5	量	REAL			✔	ID112			
6	次数	INT			✔	IW116			
7	日	通信区_Q			✔	QB100			
8	阀_1	REAL			✔	QD100			
9	阀_2	REAL			✔	QD104			
10	阀_3	REAL			✔	QD108			
11	阀_4	REAL			✔	QD112			
12	开始	BOOL			✔	Q116.0			
13	停止	BOOL			✔	Q116.1			
14									

变量表 1　常量表 1　I/O 变量　POU Variables　系统定义变量表　FB实

3. 写入 MOVE 程序。



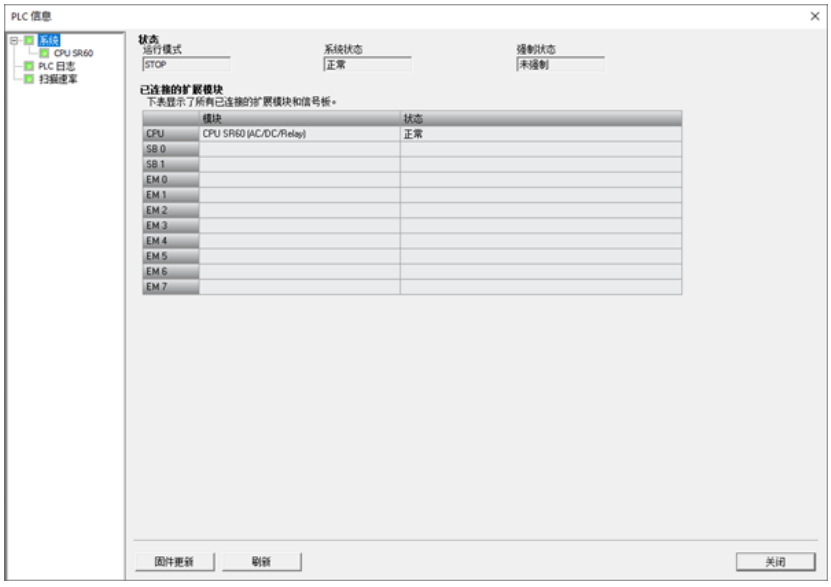
6.8 PLC 错误响应

要查看 PLC 信息或执行固件更新，可在 PLC 菜单功能区的 “信息”(Information) 区域单击 PLC 按钮。

“PLC 信息”(PLC Information) 对话框显示以下信息：

- 系统信息
  - 所连 CPU 状态
  - 组态的 PROFINET 设备状态
- PLC 日志
- PROFINET 报警
- 扫描速率

STEP 7-Micro/WIN SMART V3 显示“PLC 信息”(PLC Information) 对话框：



说明

“刷新”(Refresh) 按钮用于更新信息。只要单击“刷新”(Refresh) 按钮，就会更新所有 PLC 信息。

“固件更新”(Firmware Update) 按钮用于更新固件。

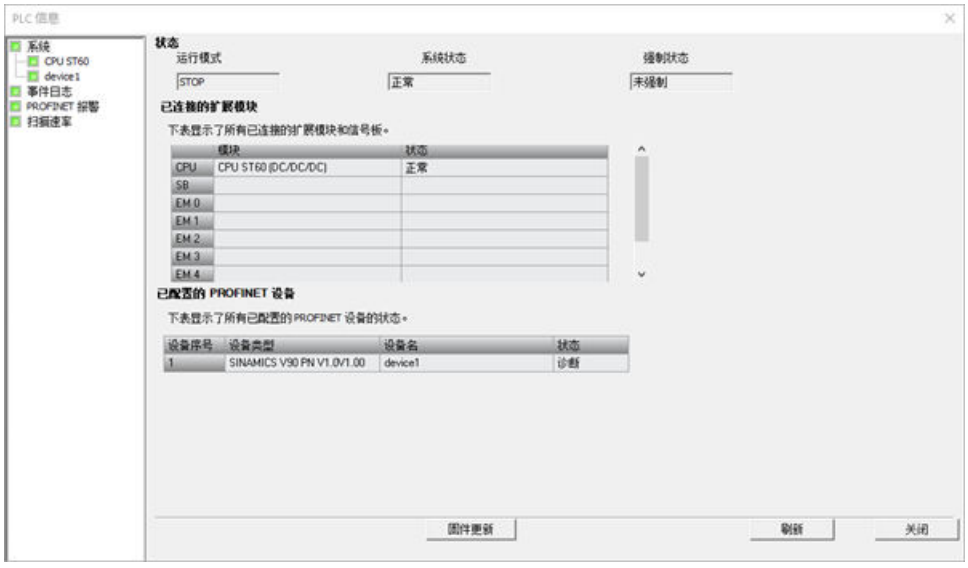
请注意以下信息：

- PLC 为编程的错误反应提供 SM 位。请参见 SM 位 (页 1265)的列表。
- GET_ERROR（获取非致命错误代码）程序指令将返回 PLC 当前的非致命错误代码并清除 PLC 中锁存的非致命错误信息。有关详细信息，请参见 GET_ERROR 指令 (页 453)。

6.8.1 系统信息

6.8.1.1 系统

要查看系统信息，请单击“系统”(System) 按钮。“PLC 信息”(PLC Information) 对话框显示系统状态、已连接扩展模块的状态以及已组态 PROFINET 设备的状态。



“系统”(System) 对话框显示以下信息：

- 状态 (Status)：系统状态
  - 操作模式：PLC 操作模式（RUN 或 STOP）
  - 系统状态：系统状态（正常或错误）
  - 强制状态：变量是否为强制
- 已连接的扩展模块 (Connected Extend Modules)：扩展模块和 CPU 信号板的状态
- 已组态的 PROFINET 设备 (Configured PROFINET Devices)：PROFINET 设备的状态
  状态如下：
  - 未连接 (Not connected)：控制器无法连接设备。
  - 正常 (OK)
  - 诊断 (Diagnosis)：报告警告。

6.8.1.2 CPU

要查看特定设备的信息，在“系统”(System) 下单击相应设备。

PLC 信息对话框将显示设备标识信息、设备状态和模块状态，如下所示：

- 连接的 CPU：CPU 的型号
  该对话框列出以下 CPU 信息：
  – 订货号 (Order Number)
  – 硬件版本 (Hardware Revision)
  – 序列号 (Serial number)
  – 固件版本 (Firmware version)
- 错误：错误信息。要确定具体错误，请参见“错误代码 (页 1262)”。
  – 当前致命错误 (Current Fatal Error)：最新的致命错误
  – 上一个致命错误 (Last Fatal Error)：“上一个致命错误”(Last Fatal Error) 字段显示 CPU 生成的上一个致命错误代码。重启后会保留该值。只要清除了 CPU 的所有存储器，就会清除此位置。
  – 当前非致命错误 (Current Non-Fatal Error)：最新的非致命错误
  – 当前 I/O 错误 (Current I/O Error)：最新的 I/O 错误
- 智能设备识别 (I-Device Identification)：智能设备信息。

  说明

  如果 CPU 用作智能设备，则 CPU 对话框会显示智能设备识别信息。

  – 上位控制器的 IP 地址 (IP Address of Higher-level Controller)：如果上位控制器与智能设备未连接，则在此显示“--.--.--.--”。
  – 与上位控制器的连接状态 (Connection Status with Higher-level Controller)：
    未连接 (Not connected)：智能设备未连接上位控制器。
    诊断 (Diagnosis)：智能设备已连接上位控制器，但该智能设备的组态与上位控制器的组态不匹配。
    正常 (OK)
  – 与上位控制器的 IO 状态 (IO Status with Higher-level Controller)：
    未连接 (Not connected)：智能设备未连接上位控制器。
    IO 数据错误 (IO data error)
    正常 (OK)

6.8.1.3 PROFINET 设备

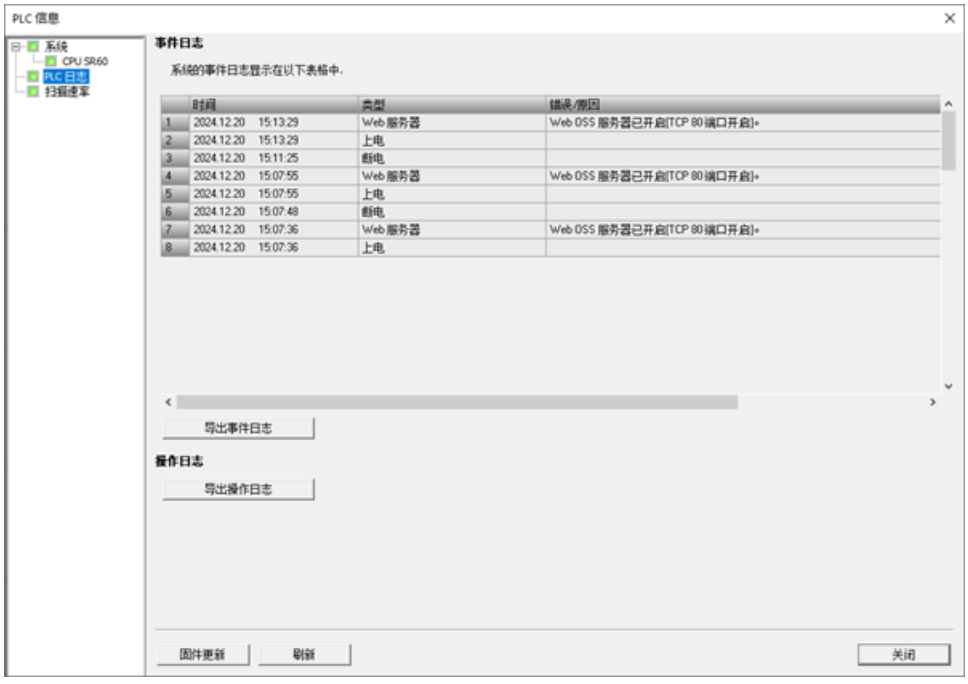
要查看特定设备的信息，在“系统”(System) 下单击相应设备。

PLC 信息对话框将显示设备标识信息、设备状态和模块状态，如下所示：

“模块状态”(Module Status) 字段显示模块状态。如果在任何模块中出现错误，则会在右侧显示详细错误信息，包括插槽和通道编号、错误描述和解决方案描述。

6.8.2 事件日志 (Event Log)

要查看 PLC 的事件日志和操作日志，请单击“PLC 日志”(PLC Log)。“PLC 信息”(PLC Information) 对话框显示带时间戳的事件表。该表列出了时间、事件类型以及错误或原因信息。



显示的事件日志的最大数量为 128。

导出日志

如果要导出日志，请单击“导出事件日志”(Export Event Log) 或“导出操作日志”(Export Operation Log) 按钮，然后浏览到要用于保存日志的文件夹。

操作日志的导出数据格式如下，其中包括时间戳、用户名、访问方法、错误代码、执行的操作和参数。

①时间戳 ②用户名 ③访问方法 ④错误代码 ⑤执行的操作 ⑥ID

说明

操作日志保存最近 512 KB 的操作日志。操作日志超过 512 KB 时，事件会记录在事件日志中。如果审计或其他目的需要这些事件，则应留意并保存这些事件。

注意
及时导出日志 为了确保故障排查的有效性和数据的完整性，请务必及时并定期导出系统日志。这有助于在需要时提供关键信息，并支持您的系统维护工作。

6.8.3 PROFINET 报警

要查看 PROFINET 网络的 PROFINET 报警，请单击“PROFINET 报警”(PROFINET Alarm)。“PLC 信息”(PLC Information) 对话框将显示包含时间戳的报警表。表中列出了设备编号、设备名称、槽地址、子槽地址和报警描述。

6.8.4 扫描速率

要查看扫描速率统计信息，请单击“扫描速率”(Scan Rates)。

“MAIN” 显示主程序的扫描速率

“MOTION” 显示运动控制的扫描速率

“MC_USER_TASK” 显示用户任务的扫描速率

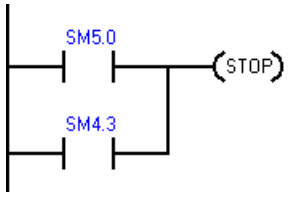
如果单击“重置”(Reset) 按钮，则会清除 PLC 中的扫描速率信息。然后，可单击“刷新”(Refresh) 按钮，以显示更新的扫描速率信息。

6.8.5 非致命错误和 I/O 错误

检测到非致命错误时，CPU 不会切换为 STOP 模式。它仅仅是把事件记录到 SM 存储器中，之后便会继续执行程序。但是，如果用户希望在发生非致命错误时强制将 CPU 切换为 STOP 模式，也可以通过编程实现。

下面的示例程序显示一个程序段，用于监视两个全局非致命错误位，并在这两个位中的任何一个位 = 1 时将 CPU 将切换为 STOP 模式。

表格 6-3　用于检测非致命错误条件的逻辑示例

LAD		STL
	发生 I/O 错误或运行错误时，切换为 STOP 模式	**Network 1** **LD SM5.0** **O SM4.3** **STOP**

非致命错误是指用户程序结构问题或用户程序中某些指令执行问题。I/O 错误是指 CPU、信号板和扩展模块的 I/O 问题。可使用 STEP 7-Micro/WIN SMART V3 查看非致命错误和 I/O 错误生成的错误代码。

在 PLC 菜单功能区的“信息”(Information) 部分单击 PLC 按钮，可查看连接到 STEP 7-Micro/WIN SMART V3 的 PLC 的当前错误状态。

表格 6-4　非致命错误类型

	说明
CPU 中的程序编译错误	下载程序时，CPU 会对程序进行编译。如果 CPU 检测到程序违反编译规则，将中止下载，并生成一个错误代码。（已经下载到 CPU 的程序仍然存在于永久存储器中，不会丢失。）可以在修正错误后再次下载程序。
I/O 设备错误	上电和下载系统块后，CPU 验证系统块中存储的 I/O 组态与 CPU、信号板和实际存在的扩展模块是否匹配。任何不匹配将导致生成设备的组态错误。在运行时，设备检测到的其他 I/O 问题（如缺少用户电源或输入值超出限制）可生成 I/O 错误。 CPU 将模块状态信息存储在特殊存储器 (SM) 位中。程序可以监视和评估这些位。SM5.0 是全局 I/O 错误位，当存在任何 I/O 错误条件时，它将保持置位。
程序执行错误	程序在执行过程中可能产生错误。这些错误的原因可能是指令使用不正确或指令处理的数据无效。例如，如果程序执行期间更改了指针，则程序编译时有效的间接地址指针可能会指向非法地址。更改指针指向无效地址为运行时间编程问题的一个示例。发生运行时编程故障时，CPU 会将 SM4.3 置位。SM4.3 会在 CPU 处于 RUN 模式期间一直保持置位。 程序可通过执行 GET_ERROR 指令 (页 453)获取当前非致命错误代码并将 SM4.3 复位为 OFF。

有关违反编译规则和运行时编程问题的描述，请参见非致命错误代码列表 (页 1258)。

有关用于报告 I/O 和程序执行错误的 SM 位的详细信息，请参见 SM 位 (页 1265)的描述。

6.8.6 致命错误

致命错误导致 PLC 停止执行程序。根据致命错误的严重程度的不同，致命错误可能导致 PLC 无法执行任一或全部功能。处理致命错误的目的是使 PLC 进入安全状态，这样 PLC 能对现有错误条件的询问做出响应。

检测到致命错误时，PLC 将切换到 STOP 模式、点亮 STOP 和 ERROR LED、覆盖输出表并切断输出。PLC 一直处于该状态，直到致命错误条件得到纠正。

在进行更改以纠正致命错误条件后，使用下列方法之一重新启动 PLC：

- 关闭 PLC 的电源后再重新接通。
- 使用 STEP 7-Micro/WIN SMART V3，在 PLC 菜单功能区的“修改”(Modify) 区域单击“暖启动”(Warm Start) 按钮。这会强制 PLC 重新启动并清除所有致命错误。

重新启动 PLC 将清除致命错误条件并执行上电诊断测试以验证致命错误是否已纠正。如果发现其它致命错误条件，则 PLC 将再次点亮 ERROR LED，指示仍然存在错误。否则，PLC 将开始正常运行。

有些错误条件可能会导致 PLC 无法进行通信。在这些情况下，无法查看 PLC 中的错误代码。这些类型的错误表明硬件发生故障，需要维修 PLC；更改程序或清空 PLC 存储器无法修复这些错误。

有关详细信息，请参见致命错误代码列表 (页 1262)。

6.9 在 RUN 模式下执行程序编辑

⚠ 警告

在 RUN 模式下下载程序的风险

在 RUN 模式下将程序变更下载至 PLC 时，相关变更将立即对程序运行产生影响。因此根本没有防范错误的余地；编程编辑中的错误将可能导致严重人身伤害甚至死亡和 / 或设备损坏。仅符合条件的相关人员可以执行 RUN 模式下的程序编辑。

概述

借助“在 RUN 模式下执行程序编辑”功能，无需将 PLC 切换为 STOP 模式即可对程序进行修改，并将相关变更下载至 PLC。

- 无需停机即可对当前程序进行细微修改。
  示例：更改参数值。
- 借助此功能，可更快速地执行程序调试。
  示例：对常开或常闭开关进行取反逻辑操作。

如果将相关更改下载至实际过程（在仿真过程中，可能会程序调试期间进行），下载之前，请务必全面考虑可能会对机器和机器操作员造成的安全后果。

在 RUN 模式下执行程序编辑的过程中，只能下载程序块（OB1、子程序、功能块和中断）。在 RUN 模式下执行程序编辑的过程中，无法下载系统块或数据块。

在 RUN 模式下执行编辑的前提条件

若要在 RUN 模式下将程序编辑下载至 PLC，必须满足以下前提条件：

- 程序编译必须成功。
- 必须在运行 STEP 7-Micro/WIN SMART V3 的计算机与 PLC 之间成功建立通信。
- 必须登录 PLC。
- 必须为受保护的 POU 提供密码才能打开块（用于正常编辑、在 RUN 模式编辑和程序状态操作）。

如果在 RUN 模式下执行程序编辑的过程中将 PLC 切换为 STOP 模式，则 PLC 将中止编辑会话。

可能发生的问题

为了帮助您确定是在 RUN 模式还是在 STOP 模式下将程序相关变更下载至 PLC，需考量在 RUN 模式下执行编辑时各种类型的程序变更所产生的影响：

- 如果删除输出的控制逻辑，则在下次重新上电或切换为 STOP 模式之前，输出将始终保持为其最后的状态。
- 如果删除在 RUN 模式下执行编辑时已运行的 HSC、Motion、或 PLS 函数，则 HSC、Motion、或 PLS 函数将继续运行，直至下一次上电循环或切换到 STOP 模式。
- 如果在 RUN 模式下执行编辑的过程中删除 ATCH 或 DTCH 指令却未删除相应中断例程，则在下次重新上电或切换为 STOP 模式之前，无论何时只要发生控制事件，仍会继续执行中断例程。

- 如果添加以首次扫描标志为条件的 ATCH 指令，则在下次重新上电或发生 STOP-to-RUN 模式转换之前，CPU 不会使能相关事件。
- 如果删除 ENI 或 DISI 指令，则在下次重新上电或发生 RUN-STOP 模式转换之前，激活的中断例程仍将继续运行。
- 如果在 RUN 模式下执行编辑的过程中修改 RCV 指令表的地址，且 RCV 指令处于激活状态，则 PLC 会将接收到的数据写入旧的表地址中。完成当前（旧地址）接收请求之后，PLC 才会使用新地址。由于已完成程序编辑，如果程序在新地址中查找数据，会发现其中不存在数据。GET 与 PUT 指令的功能类似。
- 在重新上电或从 STOP 转换为 RUN 模式之前，PLC 不会执行以首次扫描标志为条件的相关逻辑。完成 RUN 模式下的编辑之后，启动修改后的程序不会设置首次扫描标志。

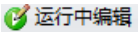
处理正负跳变

在 RUN 模式下执行编辑的过程中，为最大程度减小程序中正跳变 (EU) 和负跳变 (ED) 指令再定位的相关变更对过程的影响，STEP 7-Micro/WIN SMART V3 会为程序中所含的每条跳变指令分配一个临时的编号。对于在 RUN 模式编辑过程中添加到程序中的各跳变指令，必须为其分配唯一的标识号。为帮助用户选择尚未使用的编号，STEP 7-Micro/WIN SMART V3 的交叉引用窗口中提供了边沿使用选项卡，当激活在 RUN 模式下激活程序编辑功能后，该选项卡即可用。该表会列出当前程序中所使用的所有 EU/ED 指令，这样用户即可参照此列表执行程序变更。

在 RUN 模式下执行程序编辑和下载

要在 RUN 模式下启动程序编辑，请按以下步骤操作：

1. 在“调试”(Debug) 菜单功能区的“设置”(Settings) 区中，单击“在 RUN 模式下编辑”(Edit In Run) 按钮。

   说明

   如果尚未对程序编辑器中的当前程序进行保存，STEP 7-Micro/WIN SMART V3 会提示保存项目。项目保存可使用相同名称或者也可更改名称。

2. 单击警告对话框中的“继续”(Continue) 按钮，确认您希望继续执行 RUN 模式下的程序编辑。STEP 7-Micro/WIN SMART V3 会上传当前存储在 CPU 中的程序并在程序编辑器中显示此程序，然后，用户可根据需要在编辑器执行更改。

完成所需变更后，必须将相应变更下载至 CPU，这样变更才会生效。下载启动后，在其结束之前将无法执行 STEP 7-Micro/WIN SMART V3 中的其他任务。

检查输出窗口查看是否存在任何编译错误（例如，EU 或 ED 编号重复）。双击错误信息，即可在程序编辑器中编辑出错的程序段。

指定 CPU 分配（后台时间）

在 RUN 模式下执行程序编辑的过程中，CPU 除继续执行当前加载程序外，还需要一些时间在后台编译已修改的程序。用户可在系统块 (页 211)中组态编译可用的后台时间量。请注意，仅当 CPU 处于 STOP 模式时，才可下载系统块。

6.10 用于调试程序的功能

STEP 7-Micro/WIN SMART V3 提供了下列功能来帮助调试程序：

- 在程序中添加书签可以使您在较长程序中很方便地前后移动到特定行
- 通过交叉引用表 (页 771)跟踪程序中的引用
- 使用状态图 (页 777)显示 PLC 数据值和状态
- 显示程序编辑器中的状态 (页 773)

有关调试程序的详细信息，请参见诊断和故障排除 (页 771)章节。

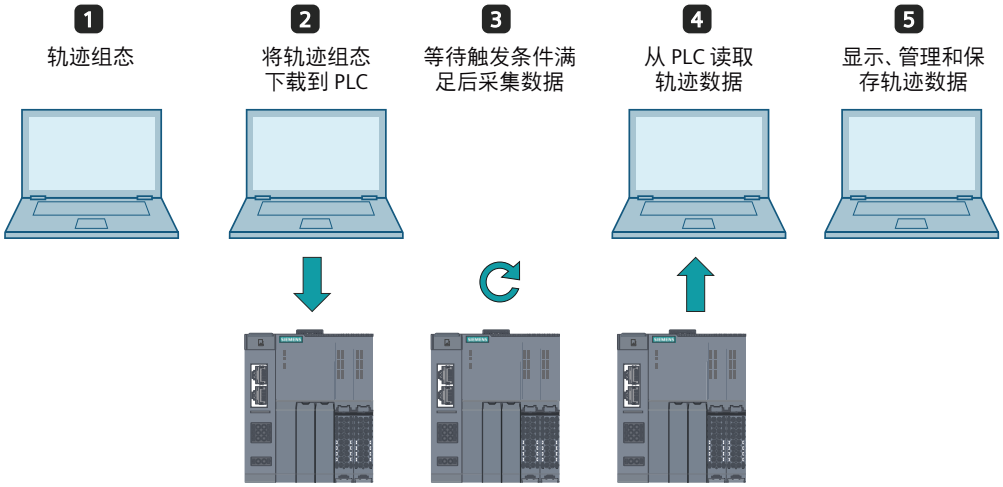
6.11 轨迹

6.11.1 简介

轨迹（Trace）能够帮助您直观且精确地捕获并追踪变量的变化。通过对这些数据的捕获、追踪和对比分析，能够更有效地判断程序的运行状态。尤其对于肉眼难以察觉的瞬态数据变化，轨迹曲线提供了强大的监控与分析能力。例如，在运动控制系统中，可用于对速度和位置变化的监控与分析；在通信过程中，可用于对数据变化、完成位或错误位状态的监控与分析；以及对程序中上升沿或下降沿事件的监控与分析等多种应用场景。

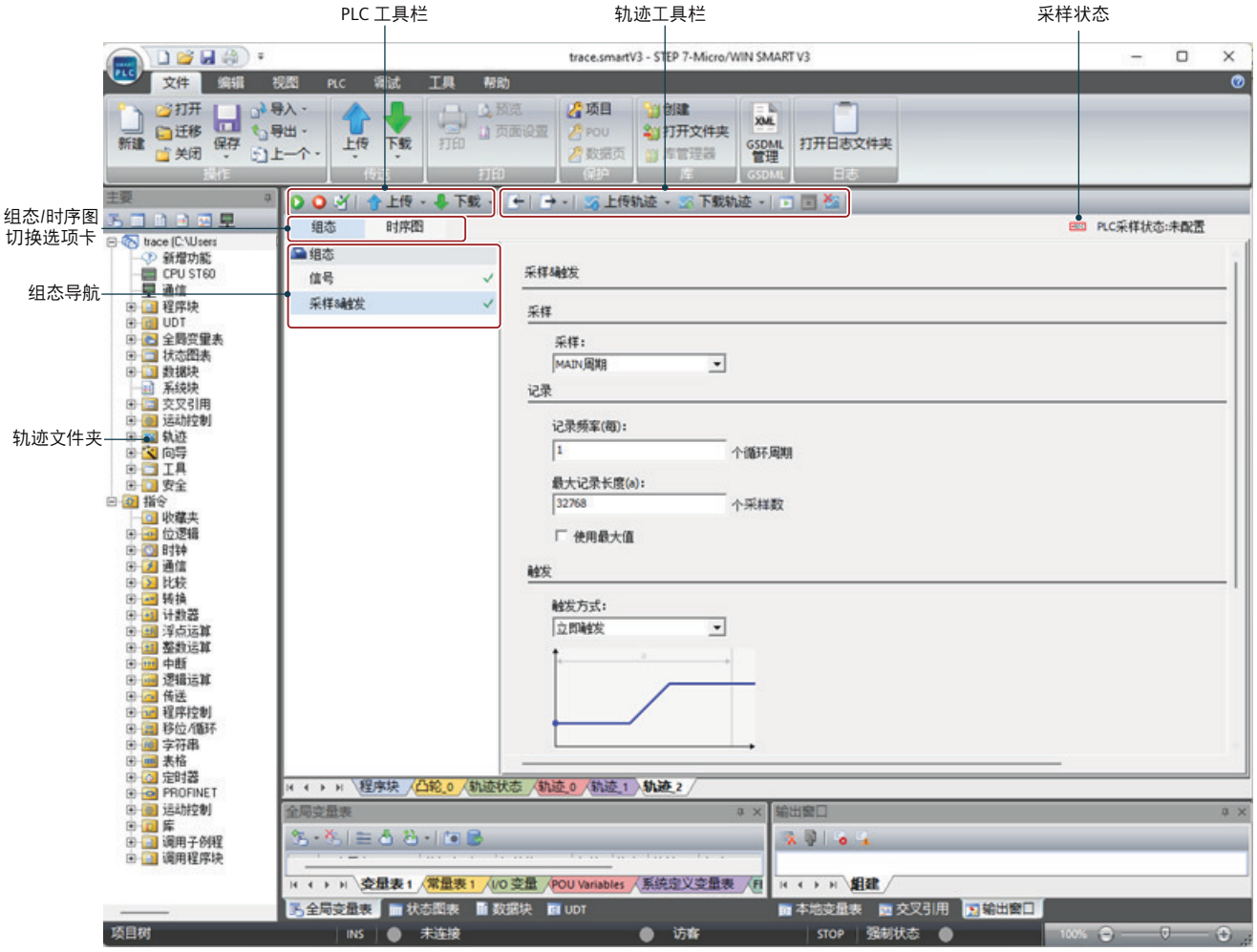
S7-200 SMART G2 支持装载 4 个轨迹组态，每个轨迹最多可以组态 32 个信号。

轨迹功能的运行模式



		描述
①	在 STEP7-Micro/WIN SMART V3 编程设备上组态轨迹	• 选择要采集的信号 • 定义采样和触发条件
②	将轨迹组态从编程设备下载到 PLC	当建立在线连接时，可将轨迹组态以任务的形式下载到 PLC。
③	等待触发条件满足后采集数据	轨迹组态激活后，数据采集将独立于编程设备进行。当满足触发条件时，采集会立即开始。PLC 中的采集数据不具有断电保持性，但可永久保存在项目中。可以在 STEP7-Micro/WIN SMART V3 项目中最多保存 128 条轨迹。
④	从 PLC 读取轨迹数据	使用上传功能可以将 PLC 中的轨迹上传至 STEP7-Micro/WIN SMART V3。轨迹任务包含轨迹组态和采集数据。每个已采集的轨迹任务都可上传保存在项目中。可以离线查看轨迹。
⑤	显示、管理和保存轨迹数据	可通过多种方式来评估曲线图和信号表中采集的数据。可将轨迹数据作为一个 CSV 文件导出。

6.11.2 用户界面



PLC 工具栏

轨迹工作区中的 PLC 工具栏提供 PLC 菜单中的可用功能子集。

轨迹工具栏

轨迹工作区中的轨迹工作栏提供轨迹传送和控制功能。

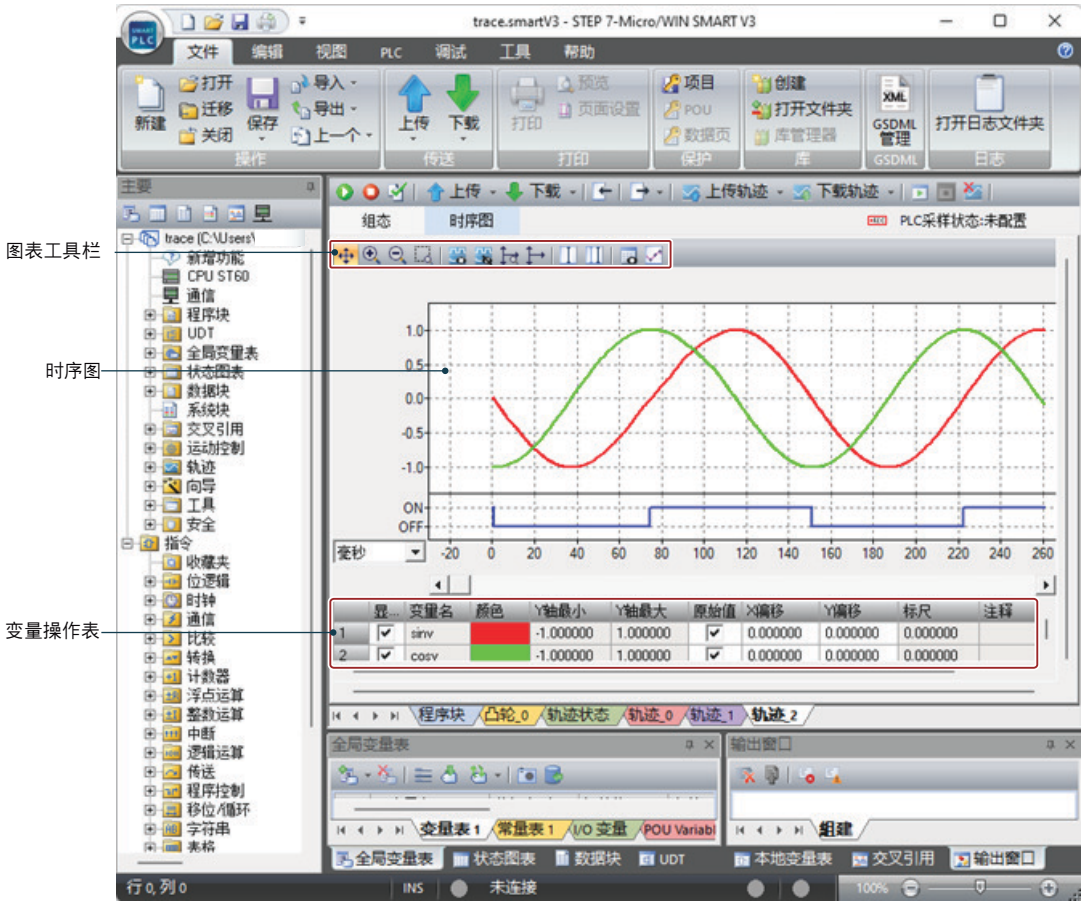
	导入轨迹 (页 207)	从文件中导入轨迹组态，此操作将覆盖当前的现有组态。
	导出轨迹 (页 207) 导出数据 (页 207)	导出轨迹组态，文件扩展名为"*.xml"; 导出轨迹数据，文件扩展名为".csv"
上传轨迹	上传轨迹 (页 207)	从 PLC 中上传指定任务的轨迹（即组态或包含数据的组态），并覆盖当前组态。
下载轨迹	下载轨迹 (页 207)	将轨迹组态以任务的形式下载到 PLC。
	激活轨迹 (页 208)	激活 PLC 上的轨迹任务。
	停止轨迹 (页 208)	停止 PLC 上的轨迹任务。
	删除轨迹 (页 210)	删除 PLC 上的轨迹。

PLC 采样状态

	该轨迹尚未配置。
	该轨迹任务已配置，但尚未激活。
	该轨迹任务已配置并已激活，但尚未满足触发条件。
	轨迹任务已激活，且触发条件已满足，数据采集已开始。
	轨迹数据采集停止： • 采集条数已达上限。 • 因 CPU STOP 导致停止。 • 取消激活。

您也可在轨迹特殊存储区 (页 1306)查看 PLC 采样状态。

6.11.3 轨迹时序图界面



图表工具栏

可使用工具栏中的工具查看轨迹曲线。

	移动视图	在此模式下，您可以通过拖动鼠标来移动时序图。
	放大	以画布中心位置放大视图，每次增加 10%。每次按下该按钮，时间轴和数值轴的范围都会缩小，曲线显示变得更大。
	缩小	以画布中心位置缩小视图，每次缩小 10%。每次按下该按钮，时间轴和数值轴的范围都会放大，曲线显示变得更小。

	框选放大	在时序图区域内，按住鼠标左键并拖动，从左上角向右下角绘制一个矩形选择框。释放鼠标左键后，系统将自动放大所框选的区域。
	显示所有变量	在时序图中显示所有变量的曲线。
	隐藏所有变量	在时序图中隐藏所有变量的曲线。
	还原缩放	轨迹任务采集完成后点击还原，将还原所有缩放级别并按根据下面的情况显示： • 如果存在双标尺，将双标尺放在中心，显示双标尺内所有数据，和标尺前后 100 个点。 • 如果存在单标尺，将单标尺放在中心，显示最多 1000 个点。 • 如果存在单双标尺，将双标尺放在中心，显示双标尺内所有数据，和标尺前后 100 个点。 • 如果不存在标尺，将 x 轴放在开头，显示最多 1000 个点。
	还原所有	还原显示当前所有非隐藏的曲线的 X 轴 和 Y 轴所有数据范围。
	显示单标尺	单标尺工具可帮助您精确地定位曲线上的坐标。在变量操作表中显示 Y 轴的测量值。
	显示双标尺	标尺工具用于计算曲线上任意两点之间的测量值。可通过鼠标移动两个测量光标的垂直位置。 在时序图中显示相关测量值和两个测量光标的位置差。
	显示/隐藏变量操作表	
	显示实际的采样点	

时序图

说明

在时序图中，模拟量和数字量会分开显示。

如果采集的变量同时包含模拟量和数字量，系统将生成两张独立的时序图。模拟量时序图将显示在上方，数字量时序图将显示在下方。

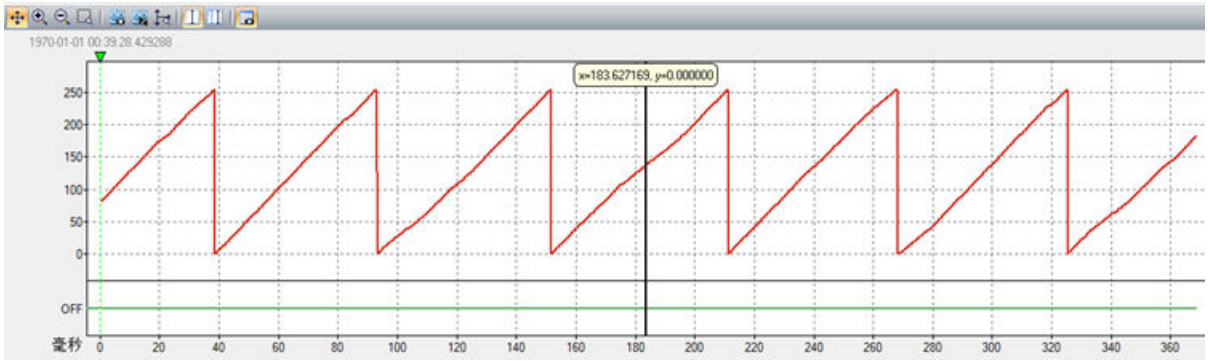
如果采集的变量仅为模拟量或数字量，系统将生成对应的时序图。

坐标轴

- 横坐标轴 (X 轴): 模拟量和数字量共用同一横坐标轴。
- 纵坐标轴 (Y 轴)
  - 模拟量： 显示浮点数值。
  - 数字量： 仅显示 ON/OFF 的离散状态。

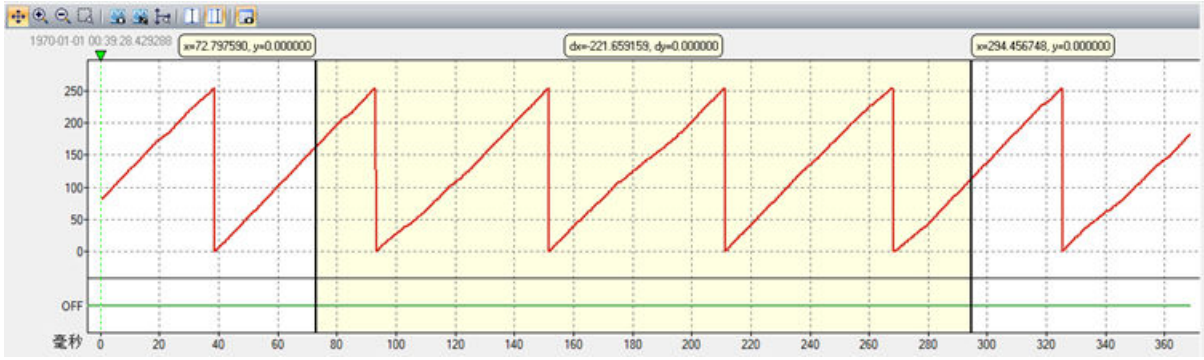
标尺显示

- 单标尺



  - 当鼠标指针悬停在标尺上时，指针会变为水平拖动样式（通常为双向箭头）。此时，按住鼠标左键并左右拖动，即可在时序图上移动标尺。
  - 标尺上显示当前选定曲线（高亮加粗显示的曲线）在标尺位置的精确横纵坐标值。
  - 在拖动标尺的过程中，显示的坐标值会实时同步更新。

- 双标尺



  - 双标尺由左、右两条独立的垂直标尺构成，它们同时作用于时序图中的模拟量和数字量数据。
  - 当鼠标指针悬停在任一标尺上时，光标会变为水平拖动样式（通常显示为双向箭头）。此时按住鼠标左键并拖动，即可在时序图上移动该标尺。
  - 标尺上显示当前选定曲线（高亮加粗显示的曲线）在标尺位置的精确横纵坐标值。
  - 在两个标尺之间的上方区域，系统会显示当前高亮曲线在两个标尺点之间的横向增量（dx）和纵向增量（dy）。
  - 两个标尺之间的区域会以不同的背景颜色填充，以清晰指示所选的数据范围。
  - 拖动任一标尺时，屏幕上显示的坐标值和增量值会同步刷新。
  - 在画布处于平移模式时，如果在两个标尺定义的范围内进行拖动操作，将不会平移整个画布，而是会整体平移双标尺及其选定范围。

变量操作表

此表允许您对变量操作表中的数据曲线进行显示属性和行为的自定义配置。

	显示	变量名	颜色	Y轴最小	Y轴最大	原...	X偏移	Y偏移	标尺	注释
1	✔	test		0.000000	255.000000	✔	0.000000	0.000000	0.000000	
2	✔	test1		0.000000	0.000000	✔	0.000000	0.000000	0.000000	
3	✔	test3		0.000000	0.000000	✔	0.000000	0.000000	0.000000	

显示/隐藏　通过勾选此列的复选框，可以在时序图中显示或隐藏对应数据曲线。

变量名　数据曲线所关联的变量名称，不可编辑。

颜色　显示数据曲线的颜色，此颜色不可编辑。

原始值
- 勾选此选项，数据曲线将以其原始的 Y 轴范围显示，不进行任何缩放。
- 取消勾选后，用户可以通过下方“Y 轴最大”和“Y 轴最小”字段来自定义曲线的缩放比例。

注意： 布尔变量不支持 Y 轴缩放功能。

Y 轴最大　当“原始值”选项未勾选时，用户可在此处手动输入数据曲线的 Y 轴最大显示值。系统将根据用户设定的最大值和最小值对曲线进行等比例缩放。

Y 轴最小　当“原始值”选项未勾选时，用户可在此处手动输入数据曲线的 Y 轴最小显示值。系统将根据用户设定的最大值和最小值对曲线进行等比例缩放。

X 偏移　用于调整数据曲线在 X 轴方向上的显示偏移量（平移）。

Y 偏移　用于调整数据曲线在 Y 轴方向上的显示偏移量（平移）。

标尺　此列显示当单标尺处于某个位置时，对应数据曲线在该位置的 Y 轴值。此值不可编辑。

为确保数据解读的准确性，标尺上显示的值始终是实际采样值。此数值不受当前视图中应用的 X 轴或 Y 轴偏移（平移）以及 Y 轴缩放比例设置的影响。

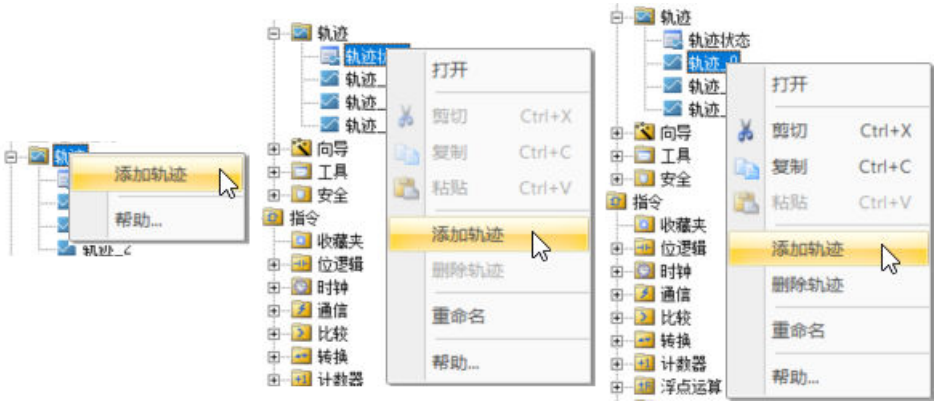
注释　显示与数据曲线相关的注释信息。此信息不可编辑。

6.11.4 组态轨迹

“轨迹”是指在 STEP 7-Micro/WIN SMART V3 中轨迹组态和采集数据的集合。轨迹组态用于规定如何从 PLC 采集数据，这包括信号选择、采样触发条件等。轨迹采样可在时序图中进行实时或离线查看。

6.11.4.1 新建轨迹

1. 添加轨迹。可以使用下列选项：
   – 右键单击项目树中的“轨迹”(Trace) 文件夹，然后选择“添加轨迹”。
   – 打开轨迹文件夹，右键单击“轨迹状态”或已有轨迹， 然后选择“添加轨迹”。



结果将在项目树中创建一个新轨迹组态，并在工作区中打开组态窗口。

说明

每个项目最多支持创建 128 个轨迹。

6.11.4.2 组态信号表

数据类型支持与限制

- 信号表可以添加要采集的变量或地址作为信号。一个轨迹最多可以配置 32 个变量信号，最少需配置一个信号。
- 轨迹信号使用的变量必须与用户工程中的变量保持一致。 手动输入变量名不存在或地址无效时，系统将以飘红方式显示错误。
- 支持的地址类型： I/Q/V/M/SM/S/T/C/HC/AQ。

  说明

  STEP 7-Micro/WIN SMART V3 (3.2) 的轨迹功能不支持 AI 地址类型。

- 仅支持纯数据类型（BOOL、BYTE、WORD、INT、DWORD、DINT、REAL）。不支持 UDT/Struct/Array/String 等复杂数据类型。然而，如果复杂数据类型中的某个成员（节点）是纯数据类型，则用户可以选择该纯数据类型的成员。
- 变量名（包括复杂数据类型展开后的完整路径）长度不得超过 128 个字符。

准备

- 您已创建并打开轨迹。
- “组态”(Configuration) 选项卡中，“信号”(Signals) 区域已打开。

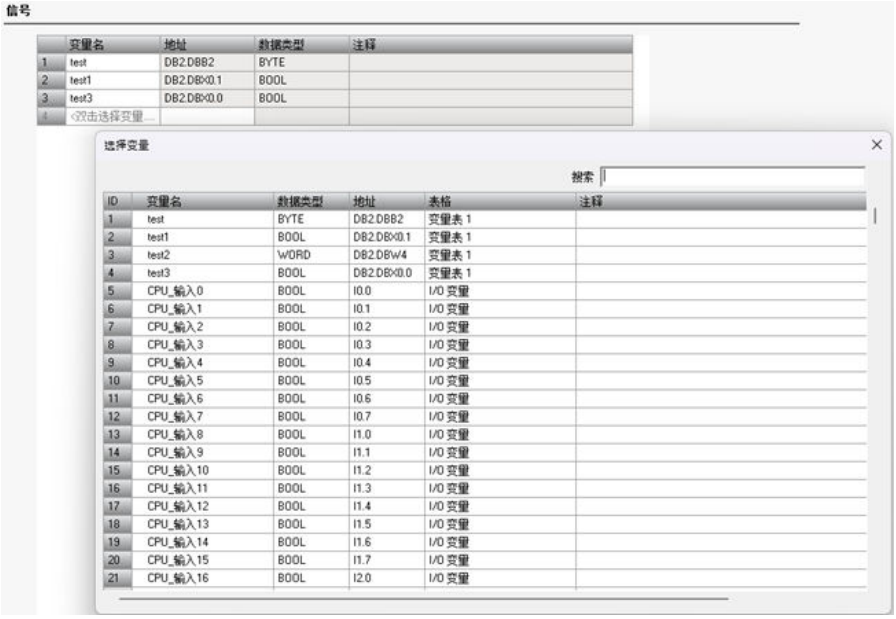
添加信号

要添加待采集的信号，请执行以下操作步骤：

- 从变量表选择：
  1. 在“变量名”列中双击空白格，或者信号表区域通过右键菜单打开“选择变量”窗口。
  2. 双击变量列表中的变量，可将其添加至信号列表。

  变量搜索：也可通过搜索框输入关键词进行查找，此搜索功能支持结构体成员变量搜索，并高亮显示匹配字段。

  变量选择后，将同步显示其地址、数据类型和注释。



- 手动输入:
  - 在“变量名”列的单元格中，输入变量的名称。输入变量名后将同步显示其地址、数据类型和注释。
  - 在“地址”(Address) 列中，直接输入地址。输入地址后将同步显示其数据类型。

重复上述操作，直至将所有待采集信号输入到表格中。

编辑信号表

删除变量

删除信号表中的变量，可采用以下方式：

- 删除单行：
  - 选中某行后按 Del 键。
  - 在信号表区域，通过右键菜单可删除变量。
- 删除多行：选中多行后按 Del 键。

重新选择变量

在信号表区域，通过右键菜单可重新选择变量。

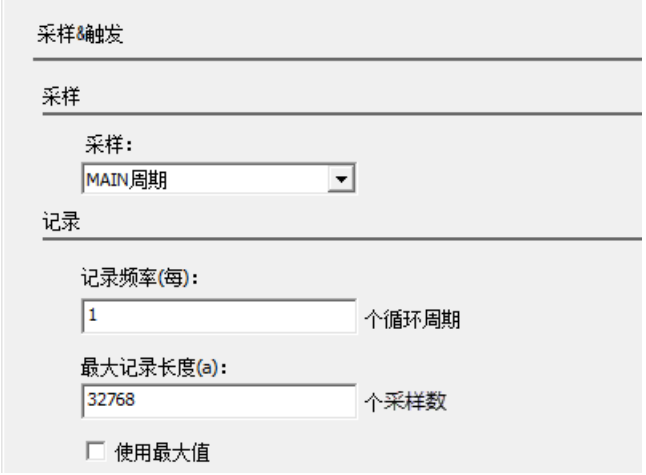
下载/激活时校验

在轨迹任务下载或激活时，系统会对所有变量名进行统一校验：

- 变量存在： 若变量在工程中存在，系统将刷新其地址、类型等信息。
- 变量不存在： 若变量在工程中不存在或无效，系统将报错提示。

6.11.4.3 组态采样条件

采样和记录条件配置如下图所示



采样

- MAIN 周期： 配置为 MAIN 周期后，轨迹将在扫描周期结束时记录轨迹信号的值。

  说明

  由于扫描周期大小随用户工程和程序实际运行情况波动，因此轨迹变量并不是固定时间间隔记录值。

- IRT 周期：配置为 IRT 周期后，轨迹将按照 IRT 周期记录轨迹信号的值。
  当配置为 IRT 周期时，配置界面会显示采样时间，即完成指定数量的采样点所需的总时长，单位为秒。
  换算公式：采样时间= IRT 周期 × 采样频率 × 采样点数

  说明

  要选用 IRT 周期，请确保在工程项目中已添加并正确组态了 IRT 轴。

  说明

  记录变量对 IRT 的影响

  轨迹将会在 IRT 的驱动线程结束时执行采样。当配置的轨迹变量数量过多时，这可能对系统的 IRT 功能产生影响。具体而言，过长的采集时间可能导致 IRT 任务无法在一个 IRT 周期内完成，从而损害系统的实时性。

记录频率

记录频率参数用于设定轨迹记录的执行间隔，即每隔多少个周期执行一次。

最大记录长度

一个信号一次采样作为一个测量点，设置范围最少 1 个点，最多 32768 个点。

要求

- 创建并打开轨迹。
- “组态”(Configuration) 选项卡中，“采样&触发”(Sampling & Trigger) 区域已打开。

操作步骤

要组态记录的周期和时长，请执行以下操作步骤：

1. 在“采样”的下拉列表中选择采样周期。
2. 在“记录频率”(Record frequency) 输入栏中，输入记录数据的循环周期间隔。
3. 指定记录长度。可以使用下列选项：
   – 在“最大记录长度”输入栏中输入采样数。
     记录频率的范围为: [1, 32768]。
   – 勾选“使用最大值”复选框后，最大记录长度将显示为 32768 且不可更改。取消勾选后，最大记录长度将恢复为上次的值。

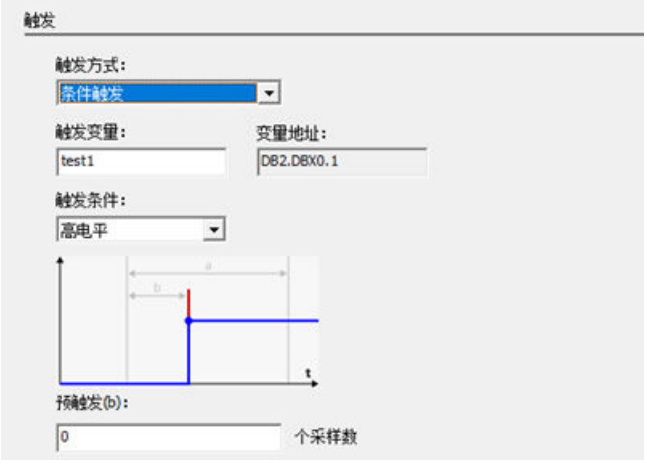
组态触发条件

在触发配置界面，您可以定义启动轨迹数据记录的条件。触发配置是轨迹采样的起始条件。

说明

要使触发配置生效，必须满足以下任一条件：

- 用户在轨迹控制面板中点击“激活轨迹”
- 将轨迹控制特殊存储器区 (页 1306)的值设置为 1



触发模式

触发模式	说明
立即触发	激活轨迹后，轨迹立即开始记录，无需配置其它参数。
条件触发	激活记录后，轨迹处于等待记录中，当触发变量满足设置条件后，轨迹才开始记录。

触发变量

变量触发条件请参考下表:

变量触发条件	数据类型	说明
=TRUE	BOOL	当触发器状态为 TRUE 时，记录开始
=FALSE	BOOL	当触发器状态为 FALSE 时，记录开始
Rising edge	BOOL	当触发器状态从 FALSE 变为 TRUE 时，记录开始
Falling edge	BOOL	当触发器状态从 TRUE 变为 FALSE 时，记录开始
= value	BYTE、WORD、INT、DWORD、DINT、REAL	当触发值等于为此事件配置的数值时，记录开始
<> value	BYTE、WORD、INT、DWORD、DINT、REAL	当触发值不等于为此事件配置的数值时，记录开始
值改变	所有类型	当触发值和记录激活时该值不同时，记录开始

预触发

预触发为轨迹在触发条件满足前所记录的测量点。例如，若最大记录长度为 1000 个测量点，预触发点数设置为 10 个测量点，则在触发条件满足后，将记录 990 个测量点。

要求

- “采样&触发”(Sampling & Trigger) 区域已在“组态”(Configuration) 选项卡中打开。

“立即触发”触发条件

要激活后立即启动采集，请在“触发方式”(Trigger mode) 下拉列表中选择条目“立即触发”(Immediate Trigger)。

“条件触发”(Conditional Trigger) 触发方式

要根据条件启动采集，请执行以下操作步骤：

1. 在“触发方式”(Trigger mode) 下拉列表中选择“条件触发”(Conditional Trigger) 条目。
2. 选择一个变量。可以使用下列选项：
   – 在“触发变量”栏中，双击空白格打开变量表并选择一个变量。
   – 直接在触发变量的输入栏中输入变量符号名或地址。

3. 在“触发条件”(Trigger Conditions) 下拉列表中选择触发条件。
   – 对于非 BOOL 型变量， 当选择“等于”或“不等于”选项时，将显示一个数值输入框。其取值范围取决于当前数据类型。
   – 对于 BOOL 类型，可以选择“高电平”、“低电平”、“上升沿”、“下降沿”、“值变化”。
4. 在“预触发”(Pretrigger) 输入框中输入预触发的采样数。
   为采集触发事件之前的周期，需在预触发输入栏中输入大于 0 的数值。

6.11.5 管理工程中的轨迹

6.11.5.1 打开轨迹

当打开工程时，系统会首先检查工程中是否保存了轨迹文件的路径。根据检查结果，系统将执行以下操作：

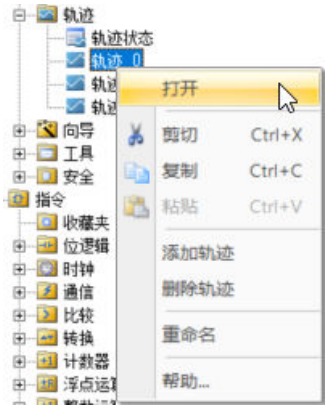
- 如果成功找到轨迹文件，系统将读取其配置信息，并据此生成相应的轨迹数据配置。
- 如果未找到指定的轨迹文件，工程的打开过程将不受影响，但系统会提示轨迹文件丢失。

最多显示 10 个丢失的轨迹名称。超出部分将以 “...” 指示。

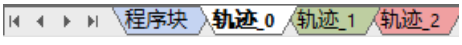
在此情况下保存工程，并在后续重新打开时，将不再报告相关错误。此时，保存的工程将仅包含首次成功加载的轨迹配置。

打开轨迹

1. 打开轨迹。可以使用下列选项：
   – 打开项目树中的“轨迹”(Trace) 文件夹，然后双击轨迹名或通过右键菜单打开您想打开的轨迹。



   – 点击程序编辑器下方的选项卡



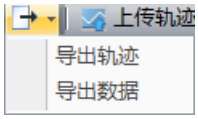
您想编辑的轨迹组态在工作区中打开。

6.11.5.2 导入/导出轨迹

导出配置&数据

您可以通过点击工具栏中的“导出”按钮，将当前的系统配置和相关数据导出为一个文件。

导出轨迹



- 将数据和配置导出为 XML 格式文件，支持后续导入轨迹。配置和数据均不会丢失。

导出数据

- 将当前采集到的数据（不包含配置）导出成 CSV。

导入轨迹

您可以通过点击工具栏中的“导入”按钮，从指定的 XML 文件中导入配置和数据。

6.11.6 管理 PLC 中的轨迹

6.11.6.1 下载/上传轨迹

“任务”是指已组态的轨迹被下载到 PLC 后的可执行实例。它代表了 PLC 上一个具体的、待执行的数据采集。每个 PLC 最多可承载 4 个任务。

说明

- 无论 PLC 处于运行（Run）状态还是停止（Stop）状态，均可执行下载/上传轨迹操作。
- 在轨迹激活过程中，不支持执行下载/上传轨迹操作。

将组态下载到 PLC

- 点击工具栏中 下载轨迹 “下载轨迹” (Download trace) 按钮旁的倒三角图标，从弹出的任务列表中选择您想要下载的任务。

说明

轨迹配置的下载独立于 STEP7-Micro/WIN SMART V3 工程的下载其具体行为如下：

- 下载轨迹配置到 PLC 中的指定任务时，系统会进行配置检查。常见的下载失败原因包括：
  - PLC 程序与 STEP7-Micro/WIN SMART V3 工程不一致
  - 未配置任何变量
  - 变量不存在或无效
  - 当前任务正在采集数据
- 若该轨迹已下载至其他任务，系统将提示您删除原有轨迹
- 下载成功后，将覆盖 PLC 中当前任务的轨迹数据。同时，轨迹状态特殊寄存器区的值将被设置为 1

将轨迹上传到 PLC

- 点击工具栏中 上传轨迹 “上传轨迹” (Upload trace) 按钮旁的倒三角图标，从弹出的任务列表中选择您想要上传的任务。
  上传时系统将弹出对话框，提示您上传操作会覆盖当前轨迹。确认上传后，系统将执行轨迹上传。

6.11.6.2 激活/停止轨迹

准备

- 轨迹组态已下载至 PLC。

激活轨迹

可以使用下列两种方式激活轨迹。

- 在组态工作区中，点击工具栏“激活” 按键完成激活。
  激活后，视图自动切换至时序图界面，该界面显示采集到的数据。时序图上的数据显示是非实时的。数据将分批加载，待当前批次显示完成后，再加载下一批。
- 轨迹特殊存储器区 (页 1306)激活轨迹：可在数据采集完成后，通过上传将数据上传至 STEP 7-Micro/WIN SMART V3，即可在时序图界面进行查看。

说明

仅在 PLC 运行状态下可激活。若 PLC 处于 STOP 状态，系统将提示您启动 PLC。确认后，PLC 将启动，并自动激活轨迹。

说明

- 时序图仅显示非实时曲线数据；采集状态请查看状态栏，状态栏更新周期 5 秒。
- 采集期间，轨迹组态不可修改。
- 采集期间，不可对轨迹执行上传、下载或删除操作。

暂停操作

暂停

在数据采集过程中，界面会实时滚动显示数据，您可以通过执行以下操作来暂停数据采集。

- 拖动画布： 通过鼠标拖动操作移动显示区域。
- 缩放操作： 包括使用鼠标滚轮进行放大/缩小，或通过其他缩放工具 (页 196)调整视图。
- 切换单/双标尺： 启用或禁用单双标尺显示。

退出暂停状态

您可以使用还原 或还原所有 功能来退出暂停状态并恢复界面刷新。

取消激活

可以使用下列两种方式取消激活轨迹。

- 在采集过程中，点击停止按钮 停止采集
- 轨迹特殊存储器区停止轨迹。

说明

若 PLC 在轨迹采集期间进入 STOP 模式，轨迹将自动停止。

6.11.6.3 轨迹状态表

“轨迹状态” 用于在 STEP 7-Micro/WIN SMART V3 中实现对 PLC 轨迹任务的观察、上传与删除。

要求

- 已和 STEP 7-Micro/WIN SMART V3 建立连接，并以相应用户身份登录 PLC

打开轨迹状态表

1. 轨迹状态位于项目树的“轨迹”文件夹下。双击可打开轨迹状态界面。

结果：轨迹状态在工作区中打开。

轨迹任务状态

轨迹状态表仅反映 PLC 中的轨迹实际运行状态，不与当前工程或本地项目进行一致性比较。因此，表中显示的状态可能与您本地工程中的组态不直接关联。

任务号	轨迹状态	轨迹名称	变量数		
0	已停止	轨迹_0	4		
1	已停止	轨迹_1	3		
2					
3	已停止	轨迹_3	1		

	描述
任务	当前任务在并发任务列表中的序号。PLC 最多可同时运行 4 个轨迹任务。
轨迹状态	当前任务中正在运行的轨迹状态。
轨迹名称	当前任务中正在运行的轨迹的名称。
变量数	当前任务中正在运行的轨迹所包含的变量数。
	将当前任务中正在运行的轨迹上传至 STEP 7-Micro/WIN SMART V3。 在执行上传操作时，您需要明确选择是新建一个轨迹，还是覆盖一个已存在的轨迹。 • 新建：将 PLC 该任务下的轨迹上传并生成新的轨迹组态 • 覆盖：将 PLC 该任务下的轨迹上传并覆盖当前 STEP 7-Micro/WIN SMART V3 中同名的轨迹组态。
	删除 PLC 中的轨迹

6.11.6.4 删除 PLC 中的轨迹配置和数据

您可以通过以下任一方式删除 PLC 中的轨迹配置及其采集的数据：

- 恢复出厂设置 (Reset to Factory)
- 执行 CPU 清除操作 (CPU Clear)
- 点击工具栏“删除”按扭。无论 PLC 处于运行（Run）状态还是停止（Stop）状态，均可使用该方式删除轨迹

PLC 设备组态 7

7.1 组态 PLC 系统的运行

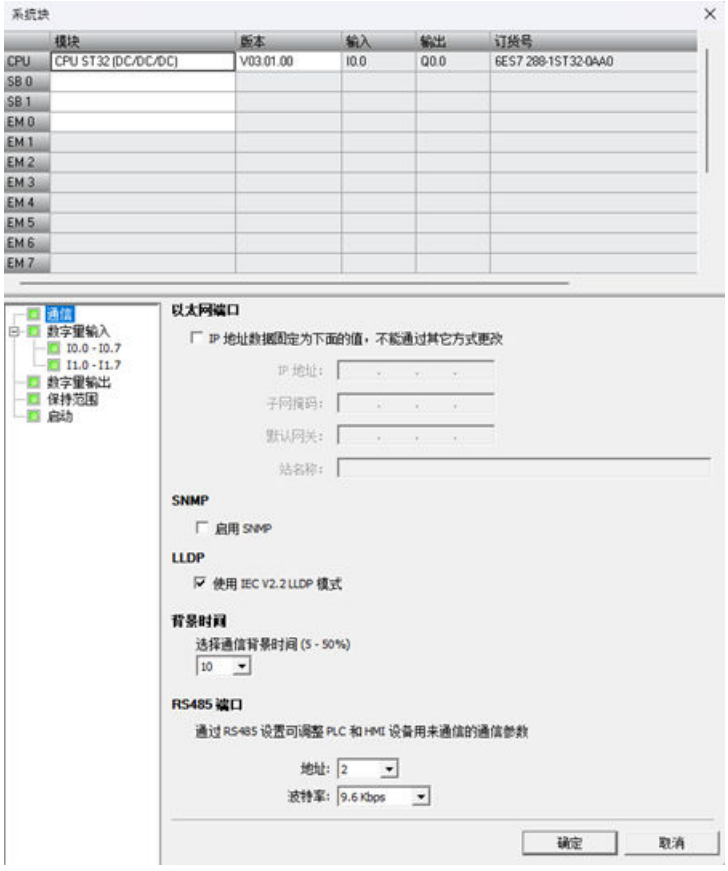
7.1.1 系统块

系统块提供 S7-200 SMART CPU、信号板和扩展模块的组态。

使用以下方法之一查看和编辑系统块以设置 CPU 选项：

- 单击导航栏上的“系统块”(System Block) 按钮。
- 在“视图”(View) 菜单功能区的“窗口”(Windows) 区域内，从“组件”(Component) 下拉列表中选择“系统块”(System Block)。
- 选择“系统块”(System Block) 节点，然后按 Enter 键，或双击项目树中的“系统块”(System Block) 节点。

STEP 7-Micro/WIN SMART 打开系统块，并显示适用于 CPU 类型的组态选项。



硬件配置

“系统块”(System Block) 对话框的顶部显示已经组态的模块，并允许您添加或删除模块。使用下拉列表更改、添加或删除 CPU 型号、信号板和扩展模块。添加模块时，输入列和输出列显示已分配的输入地址和输出地址。

说明

最好选择系统块中的 CPU 型号和固件版本作为真正要使用的 CPU 型号和固件版本。下载项目时，如果项目中的 CPU 型号或固件版本与所连接的 CPU 型号或固件版本不匹配，STEP 7-Micro/WIN SMART 将发出警告消息。您可继续下载，但如果连接的 CPU 不支持项目需要的资源和功能，将发生下载错误。

修改系统

CPU 模块切换

- 从 S7-200 SMART 3.0 CPU 切换至 S7-200 SMART G2 CPU：
  - 如未配置 SB 或 EM 模块，可直接切换。
  - 如配置了 SB 或 EM 模块，系统将提示不兼容的 SB/EM 配置将被清除。点击“否”，取消切换，或点击“是”，切换 CPU 并清除所有 SB/EM 配置。
- 从 S7-200 SMART G2 CPU 切换至 S7-200 SMART 3.0 CPU：
  - 如未配置 SB 或 EM 模块，可直接切换。
  - 如配置了 SB 或 EM 模块，系统将提示您需手动清除相关配置后方可进行切换。

说明

在系统块中切换 CPU 的类型或者版本：

- 将 CPU 类型从 SR40/60 或 ST40/60 切到 ST32, 或者将 ST32 的版本从 V3.1 切到 V3.2：
  系统会提示：“CPU 类型或者版本切换后，同时需要更新运动控制的指令和结构，是否切换？”
  - 选择【否】： 切换失败，运动控制指令和 TO (页 941) 保持 V3.1 版本。
  - 选择【是】： 切换成功，运动控制指令和 TO 更新为 V3.2 版本。
- 将 CPU 类型从 ST32（V3.2）切到 SR40/60 或 ST40/60, 或者将 ST32 的版本从 V3.2 切到 V3.1：
  - 如果配置了 PN IRT 轴： 切换失败，系统提示：“当前 CPU 版本不支持 PN IRT 轴，请删除后重试。”
  - 如果没有配置 PN IRT 轴： 系统会提示：“CPU 类型或者版本切换后，同时需要更新运动控制的指令和结构，是否切换？”
    - 选择【否】： 切换失败。
    - 选择【是】： 切换成功，运动控制指令和 TO 更新为 V3.1 版本。

EM 模块修改

EM 修改后，根据当前槽位，重新设置当前模块的默认地址，如果地址与其它模块的地址不重叠，其它模块的地址保持；如果当前模块的地址与其它模块的地址有重叠，重叠的模块的地址设置成该槽位对应的默认地址。

模块选项

系统块对话框底部显示在顶部选择的模块选项。单击组态选项树中的任意节点均可修改所选模块的项目组态。

系统块包括 CPU 模块的以下组态选项：

- 通信 (页 214)
- 数字量输入和脉冲捕捉位 (页 217)
- 数字量输出 (页 220)

- 保持范围 (页 221)
- 启动 (页 223)

其他设备（如模拟量输入 (页 225)、模拟量输出 (页 228)、RTD 模拟量输入 (页 231)、热电偶 (TC) 模拟量输入 (页 235)、RS485/RS232 CM01 通信信号板 (页 238)、电池 BA01 信号板以及附加数字量输入和输出）的特定组态选项可在添加这些模块时从系统块进行访问。

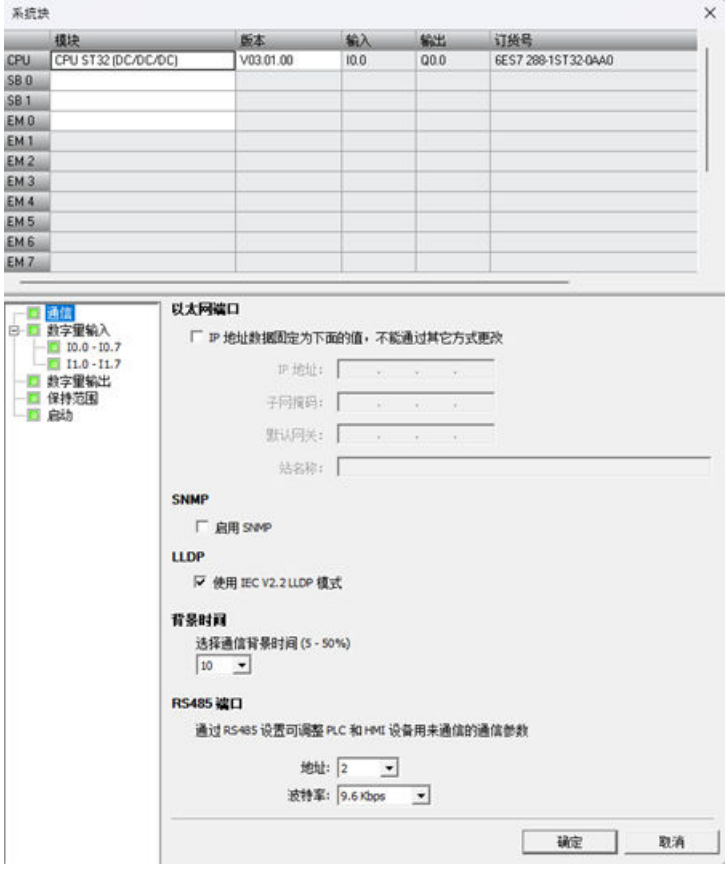
在下载或上传系统块之前，必须在 STEP 7-Micro/WIN SMART 与 CPU 之间建立通信。

然后即可下载一个修改的系统块，以便为 CPU 提供新系统组态。您所输入的新属性在将修改内容下载到 CPU 时生效。

您也可以从 CPU 上传一个现有系统块，以使 STEP 7-Micro/WIN SMART 项目组态与 CPU 组态相匹配。

7.1.2 对通信进行组态

单击“系统块”(System Block) 对话框的“通信”(Communication) 节点组态以太网端口、SNMP、背景时间和 RS485 端口。



以太网端口

若要使 CPU 从项目中获取其以太网网络端口的相关信息，则请单击“IP 地址数据固定为下面的值，不能通过其它方式更改”(IP address data is fixed to the values below and cannot be changed by other means) 复选框。然后便可输入以下以太网信息：

- “IP 地址”(IP Address)：每个设备必须有一个 Internet 协议 (IP) 地址。设备使用此地址在更加复杂的路由网络中传送数据。
- “子网掩码”(Subnet Mask)：子网是已连接的网络设备的逻辑分组。在局域网 (LAN) 中，子网中的节点彼此之间的物理位置通常相对接近。子网掩码定义 IP 子网的边界。子网掩码 255.255.255.0 通常适用于本地网络。
- “默认网关”(Default Gateway)：网关（或 IP 路由器）是 LAN 之间的链路。LAN 中的计算机可使用网关向其他网络发送消息，这些网络可能还隐含着其他 LAN。如果数据的目的地不在 LAN 内，网关会将数据转发给可将数据传送到其目的地的另一个网络或网络组。网关依靠 IP 地址来传送和接收数据包。
- “站名称”(Station Name)：站名称是在网络上定义的 CPU 名称。在“通信”(Communications) 对话框中，请使用有助于识别 CPU 的名称。

说明

站名称遵守标准 DNS（域名系统）命名规范。S7-200 SMART CPU 将站名称限制为最多 63 个字符，其中包括小写字母 a 到 z、数字 0 到 9、连字符（减号）和句号。

CPU 禁用某些名称：

- 站名称不能有 n.n.n.n 格式，其中 n 取 0 到 999 中的值。
- 站名不能以字符串 port-nnn 或字符串 port-nnn-nnnnn 开始，其中 n 是 0 到 9 的数字。例如，port-123 和 port-123-45678 为无效站名。站名称不能以连字符或句号开始或结束。

SNMP

如果要使用 SNMP 对网络设备进行监视和管理，请勾选“启用 SNMP”(Enable SNMP) 前的复选框，启用 SNMP。

使用 IEC V2.2 LLDP 模式

支持 IEC V2.3 模式的设备的默认值：

- “使用 IEC V2.2 LLDP 模式”(Use IEC V2.2 LLDP mode) 选项已禁用（无复选标记）。PROFINET 接口设置为 IECV2.3 模式 (IEC 61158 V2.3)。
- 选择此选项以将 PROFINET 接口设置为 IEC V2 2 模式。将系统拆分为多个相关项目时（例如，在智能设备组态的情况下），可能需要进行此设置。

不支持 IEC V2.3 模式的设备的默认值：

- “使用 IEC V2.2 LLDP 模式”(Use IEC V2.2 LLDP mode) 选项已选中（有复选标记）且无法更改。
- 如果当前项目中所有带 PROFINET 接口的设备都已完整组态，且参数已分配，则保留默认设置。

背景时间

可组态专门用于处理通信请求的扫描周期时间百分比。增加专门用于处理通信请求的时间百分比时，亦会增加扫描时间，从而减慢控制过程的运行速度。扫描时间仅在过程通信请求需要处理时增加。

专门用于处理通信请求的默认扫描时间百分比被设为 10%。该设置在处理编译/状态监控操作和尽量减小对控制过程的影响之间进行了合理的折衷。您可以调整该设置，每次增加 5%，最大为 50%。

随着 S7-200 SMART CPU 通信伙伴的增多，将需要更多的后台时间来处理这些伙伴的请求。GET 和 PUT 指令需要额外资源来创建并保持与其他设备间的连接。如果有 HMI 设备或其他的 CPU 通过 EM DP01 与 S7-200 SMART CPU 通信，则 EM DP01 PROFIBUS DP 模块需要额外的后台通信时间。开放式用户通信 (OUC) 还会给 CPU 增加额外负荷，并可能需要额外的后台时间。

RS485 端口

使用以下设置对板载 RS485 端口调整系统协议通信参数。连接编程设备或 HMI 设备时使用系统协议：

- RS485 端口地址：单击滚动按钮输入所需 CPU 地址 (1-126)。默认端口地址为 2。
- 波特率：从下拉列表中选择所需数据波特率（9.6 Kbps、19.2 Kbps 或 187.5 Kbps）。

说明

对于 S7-200 SMART CPU，可执行以下 RS485 通信连接：

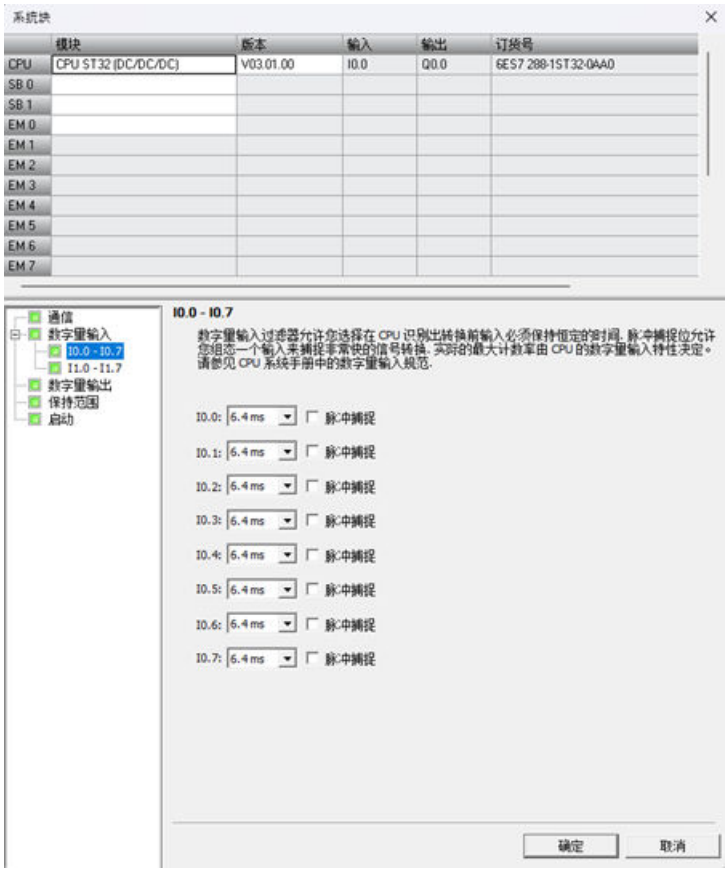
- 使用 USB-PPI 电缆通过任意串行端口（包括 RS485 端口、信号板端口和 DP01 PROFIBUS 端口）对所有型号的 CPU 进行编程。
- 使用 RS485 和 RS232 端口进行 HMI 访问（数据读/写）和自由端口通信。

说明

S7-200 SMART G2 CPU 不支持直接连接 USB-PPI 线缆。若需使用此线缆连接 S7-200 SMART G2 CPU，则需要自行制作 RS485 连接线。

7.1.3 组态数字量输入

单击“系统块”(System Block) 对话框的“数字量输入”(Digital Inputs) 节点组态数字量输入滤波器和脉冲捕捉位。



数字量输入滤波器

通过设置输入延时，您可以过滤数字量输入信号。该延迟帮助过滤输入接线上可能对输入状态造成不良改动的噪音。输入状态改变时，输入必须在时延期限内保持在新状态，才能被认为有效。滤波器会消除噪音脉冲，并强制输入线在数据被接受之前稳定下来。

使用 S7-200 SMART CPU，用户可为其所有数字量输入点选择一个输入延迟。可用输入点数取决于 CPU 型号。

前十六个输入点（I0.0 到 I0.7 以及 I1.0 到 I1.7）支持延迟时间选项的扩展设置（可在 0.2 ms 至 12.8 ms 范围内的七个设置中任选其一，或在 0.2 μs 至 12.8 μs 范围内的七个设置中任选其一）。

所有输入点的默认滤波时间均为 6.4 ms。

要设置输入延迟，请按以下步骤操作：

1. 从一个或多个输入旁的下拉列表中选择延迟时间。
2. 单击“确定”(OK) 按钮，输入选项。

警告

更改数字量输入通道的滤波时间存在的风险

如果数字量输入通道的滤波时间更改自以前的设置，则新的“0”电平输入值可能需要保持长达 12.8 ms 的累积时间，然后滤波器才会完全响应新输入。在此时间段内，持续时间小于 12.8 ms 的短“0”脉冲事件可能无法检测到或无法计数。

滤波时间的这种更改会引发意外的机械或过程操作，这可能会导致人员死亡、重伤和/或设备损坏。

为了确保新的滤波时间立即生效，必须关闭 CPU 电源后再开启。

脉冲捕捉位

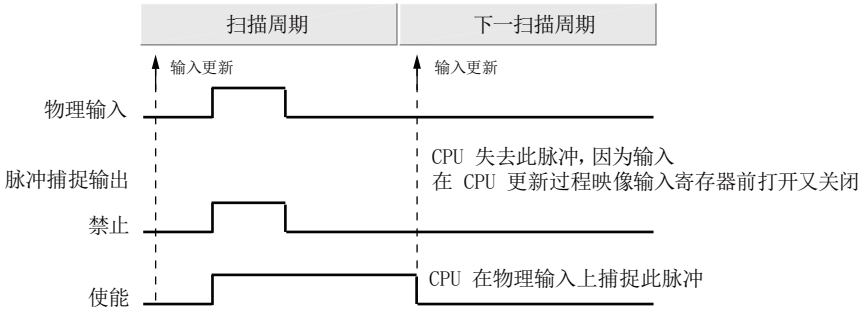
S7-200 SMART CPU 为数字量输入点提供脉冲捕捉功能。通过脉冲捕捉功能可以捕捉高电平脉冲或低电平脉冲。此类脉冲出现的时间极短，CPU 在扫描周期开始读取数字量输入时，可能无法始终看到此类脉冲。

当为某一输入点启用脉冲捕捉时，输入状态的改变被锁定，并保持至下一次输入循环更新。这样可确保延续时间很短的脉冲被捕捉，并保持至 S7-200 SMART CPU 读取输入。

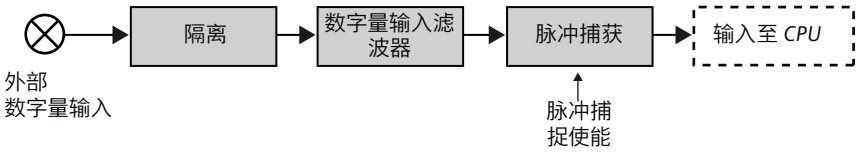
可根据 CPU 型号单独启用前十六个数字量输入点（I0.0 至 I0.7 以及 I1.0 至 I1.7）的脉冲捕捉操作。

如果组态中包含 SB DT04，则可启用此信号板上提供的两个附加数字量输入点的脉冲捕捉操作。

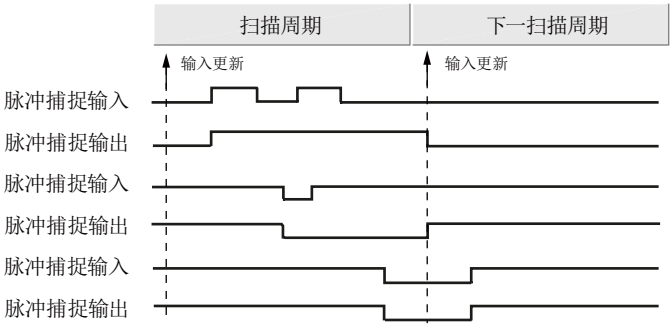
下图显示 S7-200 SMART CPU（脉冲捕捉启用和未启用）的基本操作状况：



由于脉冲捕捉功能在输入通过输入滤波器后对输入进行操作，您必须调整输入滤波时间，以防滤波器过滤掉脉冲。下图显示数字量输入电路方框图：

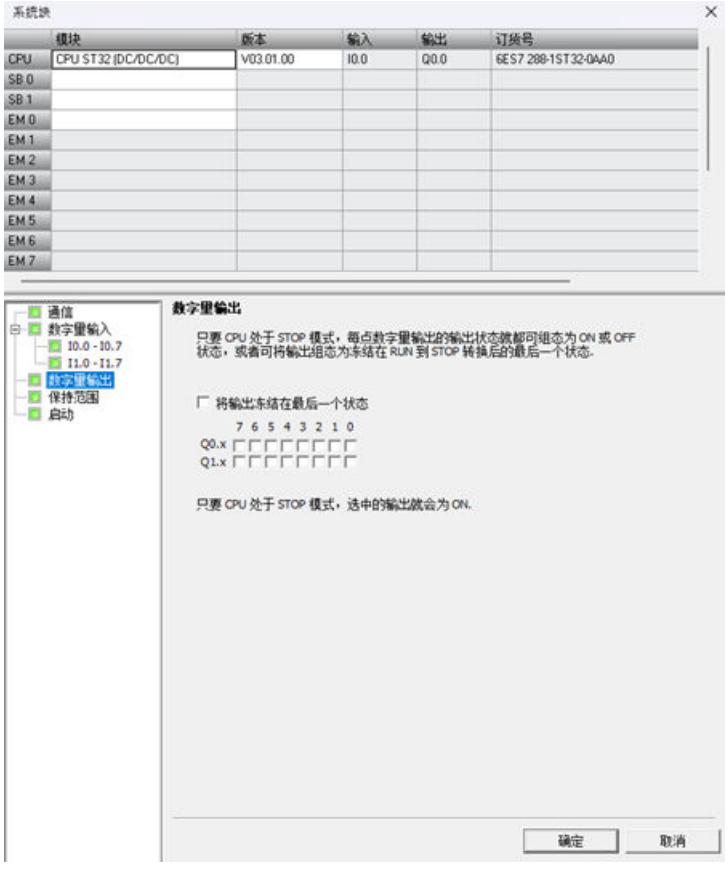


下图显示启用脉冲捕捉功能时对各种不同输入条件的响应。如果在某一特定扫描中存在一个以上脉冲，仅读取第一个脉冲。如果在某一特定扫描中有多个脉冲，则应当使用上升/下降沿中断事件：



7.1.4 组态数字量输出

单击“系统块”(System Block) 的“数字量输出”(Digital Outputs) 节点组态所选模块的数字量输出选项。



当 CPU 处于 STOP 模式时，可将数字量输出点设置为特定值，或者保持在切换到 STOP 模式之前存在的输出状态。

STOP 模式下，有两种方法可用于设置数字量输出行为：

- **“将输出冻结在最后状态”(Freeze Outputs in last state)：** 单击此复选框，就可在进行 RUN 到 STOP 转换时将所有数字量输出冻结在其最后的状态。
- **“替换值”(Substitute value)：** 如果“将输出冻结在最后状态”(Freeze Outputs in last state) 复选框未选中，只要 CPU 处于 STOP 模式，此表就允许选择每个输出所需状态。 单击要设置为 ON (1) 的每个输出的复选框。 数字量输出的默认替换值为 OFF (0)。

7.1.5 组态保持范围

单击“系统块”(System Block) 对话框的“保持范围”(Retentive Ranges) 节点组态在循环上电后保留下来的存储器范围。

选择要在上电循环期间保持的存储区。为 V、M、T 或 C 存储器输入新值。

您可将下列存储区中的地址范围定义为保持：V、M、T 或 C。对于定时器，只能保持保持性定时器 (TONR)，而对于定时器和计数器，只能保持当前值（每次上电时都将定时器和计数器位清零）。

默认情况下，CPU 无定义的保持存储区，但可组态保持范围：

- S7-200 SMART CPU 型号 CPU SR40、CPU ST40、CPU SR60、CPU ST60 和 ST32 的非易失存储器容量最大为 40 KB，其中包括 DB2 和 V、M、T、C。
- DB2 区域只能在全局变量表中组态。

CPU 断电后的数据保持

CPU 在断电和上电时对保持性存储器执行以下操作：

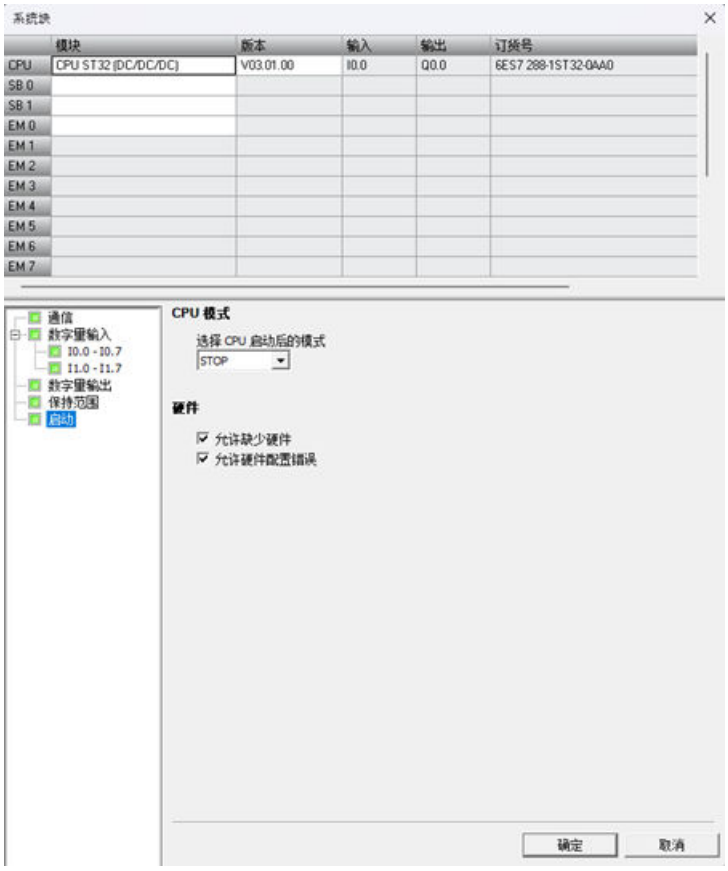
- **断电时：**
  CPU 将指定的保持性存储器范围保存到永久存储器。
- **上电时：**
  CPU 先将 V、M、C、T 或 DB2 存储器清零，将所有初始值都从数据块复制到 V 存储器，然后将保存的保持值从永久存储器复制到 RAM。

S7-200 SMART CPU 存储器地址保持范围

数据类型	描述	CPU SR40 CPU ST40	CPU ST32 CPU SR60 CPU ST60
V	数据存储器	VB0-VB81919	VB0-VB102399
T	定时器	T0-T31、T64-T95、T256 - T261、T268 - T291、T316 - T413	
C	计数器	C0-C511	
M	标志位	MB0-MB63	

7.1.6 组态启动选项

单击“系统块”(System Block) 对话框的“启动”(Startup) 节点组态 PLC 的启动选项。



CPU 模式

可从此对话框选择 CPU 启动后的模式。可以选择以下三种模式之一：

- STOP
  CPU 在上电或重启后始终应该进入 STOP 模式（默认选项）。
- RUN
  CPU 在上电或重启后始终应该进入 RUN 模式。对于多数应用，特别是对 CPU 独立运行而不连接 STEP 7-Micro/WIN SMART 的应用，RUN 启动模式选项是正确选择。
- LAST
  CPU 应进入上一次上电或重启前存在的工作模式。此选项可用于程序开发或调试。要注意运行中的 CPU 会因为很多原因进入 STOP 模式，例如扩展模块故障、扫描看门狗超时事件、存储卡插入或不规则上电事件。CPU 进入 STOP 模式后，每次上电时 CPU 都会继续进入 STOP 模式。必须通过 STEP 7-Micro/WIN SMART 将 CPU 恢复到 RUN 模式。

硬件选项

还可组态 CPU 以允许在以下硬件条件下以 RUN 模式运行：

- 缺少在 CPU 中存储的硬件配置内指定的一台或多台设备。
- CPU 中存储的硬件配置与实际存在的设备之间存在差别，导致配置错误（例如，离散输入模块取代了组态的离散输出模块）。

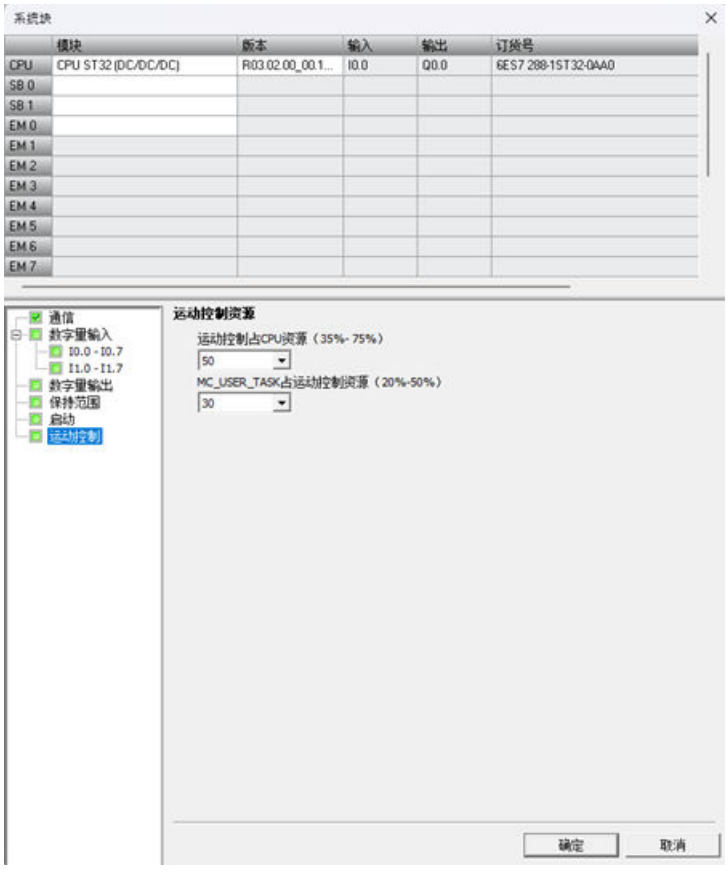
如果不选择以上选项之一或全部并有任一禁止条件为真，则禁止 CPU 进入 RUN 模式。

7.1.7 运动控制

单击“系统块”对话框的“运动控制”节点组态 PLC 的运动控制资源。

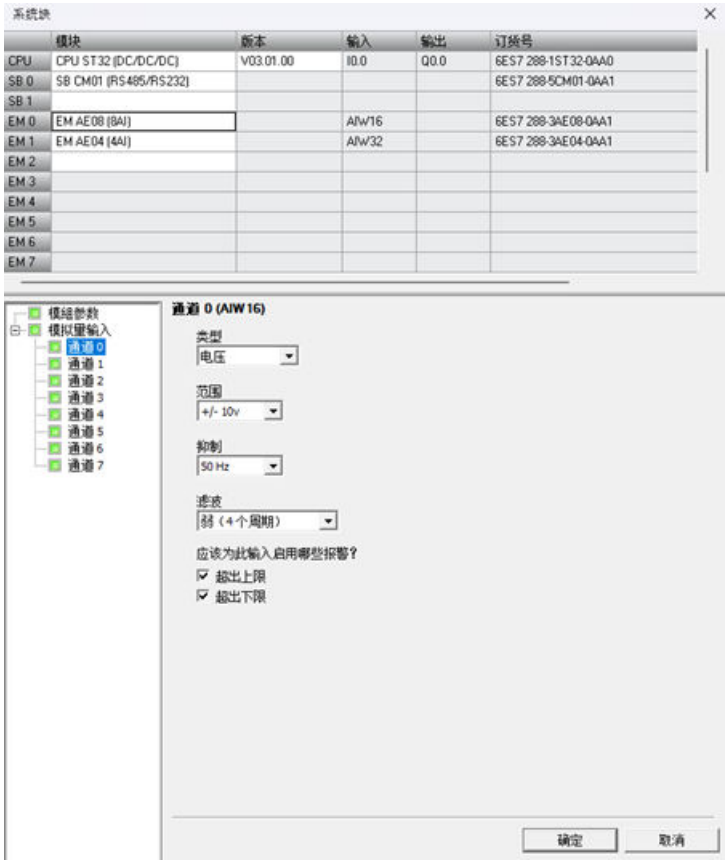
您可以根据实际应用需求，调整 CPU 资源在不同任务之间的分配。

- 运动控制任务所占用的 CPU 资源比例
- ME_USER_TASK 占用运动控制的资源比例



7.1.8 组态模拟量输入

单击“系统块”(System Block) 对话框的“模拟量输入”(Analog Inputs) 节点为在顶部选择的模拟量输入模块组态选项。



模拟量类型组态

对于每条模拟量输入通道，都将类型组态为电压或电流。为偶数通道选择的类型也适用于奇数通道：为通道 0 选择的类型也适用于通道 1，为通道 2 选择的类型也适用于通道 3。

范围

然后组态通道的电压范围或电流范围。可选择以下取值范围之一：

- +/- 2.5v
- +/- 5v
- +/- 10v
- 0 - 20ma

“抑制”(Rejection)

传感器的响应时间或传送模拟量信号至模块的信号线的长度和状况，也会引起模拟量输入值的波动。这种情况下，可能会因波动值变化太快而导致程序逻辑无法有效响应。用户可组态模块对信号进行抑制，进而消除或最小化以下频率点的噪声：

- 10 Hz
- 50 Hz
- 60 Hz
- 400 Hz

“平滑”(Smoothing)

可组态模块在组态的周期数内平滑模拟量输入信号，从而将一个平均值传送给程序逻辑。有四种平滑算法可供选择：

- 无（无平滑）
- 弱
- 中
- 强

报警组态

可为所选模块的所选通道选择是启用还是禁用以下报警：

- 超出上限（值 > 32511）
- 超出下限（值 < -32512）
- 用户电源（在系统块的“模块参数”(Module Parameters) 节点下组态，参见下图。）

模拟量输入技术规范参考

有关模拟量输入组态选项的更多信息，请参见以下技术规范：

- 范围：“模拟量输入的电压和电流测量范围（SB 和 SM）”(页 1237)
- 抑制：“模拟量输入的采样时间和更新时间”(页 1237)
- 滤波：“模拟量输入的阶跃响应”(页 1236)

7.1.9 组态模拟量输出

单击“系统块”(System Block) 对话框的“模拟量输出”(Analog Outputs) 节点为在顶部选择的模拟量输出模块组态选项。

模拟量类型组态

对于每条模拟量输出通道，都将类型组态为电压或电流。

范围

然后组态通道的电压范围或电流范围。可选择以下取值范围之一：

- +/- 10 V
- 0 - 20 mA

STOP 模式下的输出行为

当 CPU 处于 STOP 模式时，可将模拟量输出点设置为特定值，或者保持在切换到 STOP 模式之前存在的输出状态。

STOP 模式下，有两种方法可用于设置模拟量输出行为：

- “将输出冻结在最后状态”(Freeze outputs in last state)：单击此复选框，就可在 PLC 进行 RUN 到 STOP 转换时将所有模拟量输出冻结在其最后值。
- “替换值”(Substitute value)：如果“将输出冻结在最后状态”(Freeze outputs in last state) 复选框未选中，只要 CPU 处于 STOP 模式就可输入应用于输出的值（-32512 到 32511）。默认替换值为 0。

报警组态

可为所选模块的所选通道选择是启用还是禁用以下报警：

- 超出上限（值 > 32511）
- 超出下限（值 < -32512）
- “断路”(Wire break)（仅限电流通道）

- “短路”(Short circuit)（仅限电压通道）
- 用户电源（在系统块“模块参数”(Module Parameters) 节点组态，参见下图。）

模拟量输出技术规范参考

有关模拟量输出范围组态的更多信息，请参见“模拟量输出的电压和电流测量范围（SB 和 SM）” (页 1238)技术规范

7.1.10 组态 RTD 模拟量输入

在“系统块”(System Block) 对话框中，单击 RTD 模拟量输入节点，对顶部所选 RTD 模拟量输入模块的相关选项进行组态。

RTD 模拟量输入模块可提供端子 I+ 和 I- 电流，用于电阻测量。电流流经电阻，以测量其电压。电流电缆必须直接接线到电阻温度计/电阻。

针对 4 线制或 3 线制编程的测量可补偿线路阻抗，并返回相当高精度的测量结果（与 2 线制比较）。

RTD 类型组态

选择以下任一选项，组态各 RTD 输入通道的类型：

- 电阻，4 线制
- 电阻，3 线制
- 电阻，2 线制
- 热敏电阻，4 线制

- 热敏电阻，3 线制
- 热敏电阻，2 线制

电阻

根据所选 RTD 类型，可为通道组态以下 RTD 电阻：

表格 7-1 RTD 类型及可用电阻

RTD 类型	RTD 电阻	
• 电阻，4 线制 • 电阻，3 线制 • 电阻，2 线制 注：对于这些 RTD 类型和电阻，无法组态其温度系数和温度标定。	• 150 欧姆 • 300 欧姆 • 600 欧姆 • 3000 欧姆	
• 热敏电阻，4 线制 • 热敏电阻，3 线制 • 热敏电阻，2 线制	• Pt 10 • Pt 50 • Pt 100 • Pt 200 • Pt 500 • Pt 1000 • LG-Ni 1000	• Ni 100 • Ni 120 • Ni 200 • Ni 500 • Ni 1000 • Cu 10 • Cu 50 • Cu 100

系数

根据所选 RTD 电阻，可为通道组态以下 RTD 温度系数：

RTD 电阻	RTD 温度系数
• 150 欧姆 • 300 欧姆 • 600 欧姆 • 3000 欧姆	注：对于这些 RTD 电阻，无法组态其温度系数和温度标定。
• Pt 10 • Pt 50	• Pt 0.00385055 • Pt 0.003910

RTD 电阻	RTD 温度系数
• Pt 100 • Pt 500	• Pt 0.00385055 • Pt 0.003916 • Pt 0.003902 • Pt 0.003920 • Pt 0.003910
• Pt 200 • Pt 1000	• Pt 0.00385055 • Pt 0.003916 • Pt 0.003902 • Pt 0.003920
• Ni 100	• Ni 0.006170 • Ni 0.006180 • Ni 0.006720
• Ni 120 • Ni 200 • Ni 500 • Ni 1000	• Ni 0.006180 • Ni 0.006720
• Cu 10	• Cu 0.00426 • Cu 0.00428 • Cu 0.00427
• Cu 50 • Cu 100	• Cu 0.00426 • Cu 0.00428
• LG-Ni 1000	• LG-Ni 0.005000

标定

选择以下任一选项，组态通道的温度标定：

- 摄氏度
- 华氏

说明

对于“电阻，4 线制”，“电阻，3 线制”和“电阻，2 线制”RTD 类型和相关电阻，无法其组态温度系数和温度标定。

抑制

传感器的响应时间或负责向模块传送 RTD 模拟量信号的线缆的长度和状况，也会引起 RTD 模拟量输入值的波动。这种情况下，可能会因波动值变化太快而导致程序逻辑无法有效响应。用户可组态模块对信号进行抑制，进而消除或最小化以下频率点的噪声：

- 10 Hz
- 50 Hz
- 60 Hz
- 400 Hz

平滑化

用户可对模块进行组态，在组态的周期数内平滑 RTD 模拟量输入信号，然后将平均值传送至程序逻辑。有四种平滑算法可供选择：

- 无
- 弱
- 中
- 强

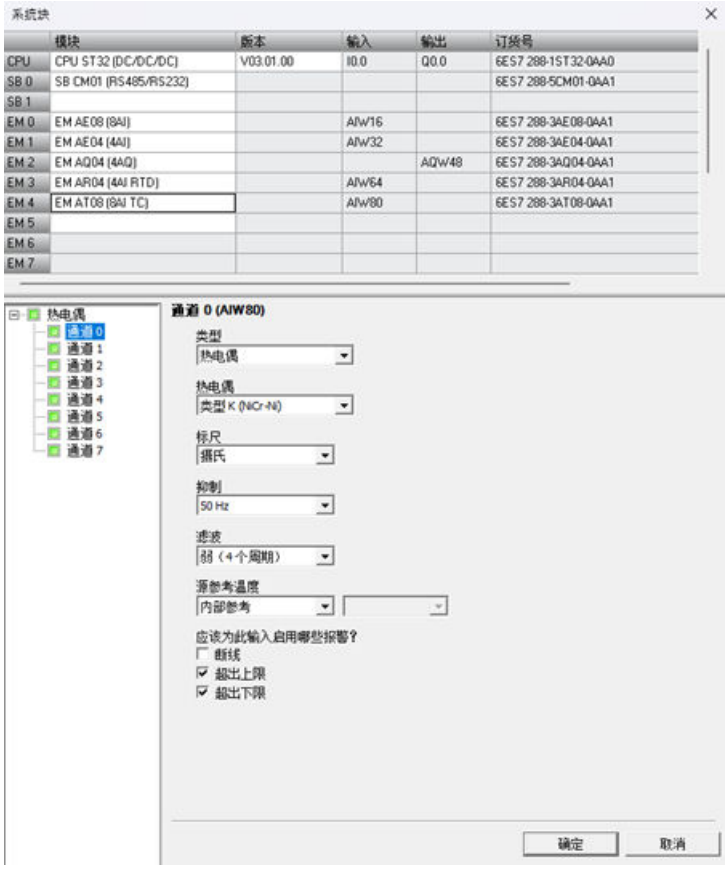
报警组态

可针对所选 RTD 模块的选定通道，选择启用或禁用下列报警：

- 断路
- 超出上限
- 超出下限

7.1.11 组态 TC 模拟量输入

在“系统块”(System Block) 对话框中，单击 TC（Thermocouple，热电偶）模拟量输入节点，对顶部所选 TC 模拟量输入模块的相关选项进行组态。



TC 模拟量扩展模块可测量连接到模块输入的电压值。

热电偶类型组态

选择以下任一选项，组态各 TC 模拟量输入模块通道的类型：

- 热电偶
- 电压

热电偶

根据所选热电偶类型，可为通道组态以下热电偶：

- B 型 (PtRh-PtRh)
- N 型 (NiCrSi-NiSi)

- E 型 (NiCr-CuNi)
- R 型 (PtRh-Pt)
- S 型 (PtRh-Pt)
- J 型 (Fe-CuNi)
- T 型 (Cu-CuNi)
- K 型 (NiCr-Ni)
- C 型 (W5Re-W26Re)
- TXK/XK (TXK/XK(L))

标定

选择以下任一选项，组态通道的温度标定：

- 摄氏度
- 华氏

抑制

传感器的响应时间或负责向模块传送热电偶模拟量信号的线缆的长度和状况，也会引起热电偶模拟量输入值的波动。这种情况下，可能会因波动值变化太快而导致程序逻辑无法有效响应。用户可组态 TC 模拟量输入模块对信号进行抑制，进而消除或最小化以下频率点的噪声：

- 10 Hz
- 50 Hz
- 60 Hz
- 400 Hz

平滑化

用户可对模块进行组态，在组态的周期数内平滑热电偶模拟量输入信号，然后将平均值传送至程序逻辑。有四种平滑算法可供选择：

- 无
- 弱
- 中
- 强

源参考温度

选择以下任一选项，组态各 TC 模拟量输入模块通道的源参考温度：

- 由参数设定
- 内部参比

报警组态

可针对所选 TC 模拟量输入模块的选定通道，选择启用或禁用下列报警：

- 断路
- 超出上限
- 超出下限

热电偶的基本操作

两种不同的金属彼此之间存在电气连接时，便会形成热电偶。热电偶产生的电压与结点温度成正比。电压很小；一微伏能表示很多度。测量热电偶产生的电压，对额外的结点进行补偿，然后将测量结果线性化，这些是使用热电偶测量温度的基础。

将热电偶连接至 TC 模拟量输入模块时，需将两条不同的金属线连接至模块的信号连接器上。这两条不同的金属线互相连接的位置即形成了传感器热电偶。

在这两条不同的金属线与信号连接器相连的位置，构成了另外二个热电偶。连接器温度会引起一定的电压，该电压将添加到传感器热电偶产生的电压中。如果不对该电压进行修正，结果报告的温度将偏离传感器温度。

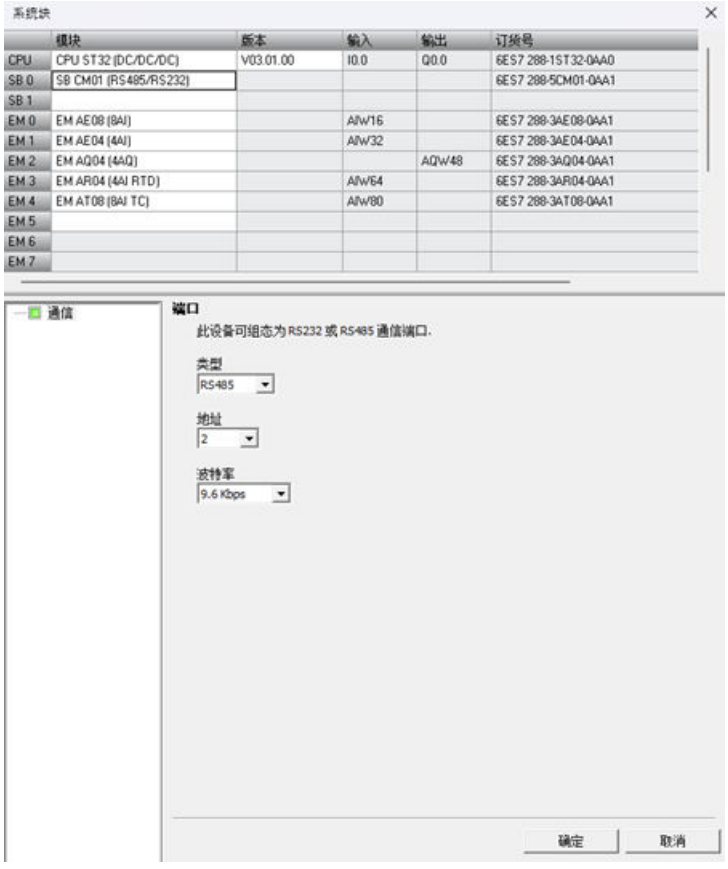
冷端补偿便是用于对连接器热电偶进行补偿。热电偶表是基于参比端温度（通常是零摄氏度）得来的。冷端补偿用于将连接器温度修正为零摄氏度。冷端补偿可消除连接器热电偶增加的电压。模块的温度在内部测量，然后转换为数值并添加到传感器换算中。之后是使用热电偶表对修正后的传感器换算值进行线性化。

为使冷端补偿取得最佳效果，必须将热电偶模块安装在温度稳定的环境中。符合模块规范的模块环境温度的缓慢变化（低于 0.1°C/分钟）能够被正确补偿。穿过模块的空气流动也会引起冷端补偿误差。

如果需要更佳的冷端误差补偿效果，则可使用外部 iso 热端子块。热电偶模块可以使用 0°C 基准值或 50°C 基准值端子块。

7.1.12 组态 RS485/RS232 CM01 通信信号板

在“系统块”(System Block) 对话框中，单击 CM01 通信信号板节点，对顶部所选 RS485/RS232 CM01 通信信号板的相关选项进行组态。最多可组态两个信号板。



CM01 信号板类型组态

从下拉列表中选择以下任一选项，组态 CM01 信号板的类型：

- RS485
- RS232

地址

单击滚动按钮，为 RS485 或 RS232 端口输入所需端口地址 (1-126)：默认端口地址为 2。

波特率

从下拉列表中选择所需数据波特率：

- 9.6 Kbps
- 19.2 Kbps
- 187.5 Kbps

7.1.13 清除 PLC 存储区

要清除指定的 PLC 存储区，按以下步骤操作：

1. 确保 PLC 处于 STOP 模式。
2. 在 PLC 菜单功能区的“修改”(Modify) 区域单击“清除”(Clear) 按钮。

警告

清除 PLC 存储区对输出的影响

清除 PLC 存储区影响数字量和模拟量输出的状态。默认设置是让数字量和模拟量输出使用替代值 0。如果已为数字量或模拟量输出定义不为 0 的替代值或选择“冻结”(Freeze)，则在删除系统块时需要注意，将删除替换值和冻结信息，因此输出将返回默认值 0。此外，如果执行选择性清除以保留系统块而删除程序块，则模拟量输出冻结在其当前值。直至下载新的程序块，对模拟量输出状态进行更改的唯一方法是使用状态图。

清除 PLC 存储区时，如果 S7-200 SMART PLC 与设备相连，对数字量输出状态的更改可发送到该设备。如果清除了 PLC 存储区，却没有仔细考虑对数字量和模拟量输出的影响，设备操作可能出现无法预料的状况，这可能导致人员死亡或严重伤害和/或设备损坏。

请始终采用适当的安全预防措施，并在清除 PLC 存储区之前确保进程处于安全状态。

3. 选择要清除的内容 - 项目、证书、轨迹和和用户自定义 Web，或选择“复位为出厂默认值”(Reset to factory defaults)。
4. 单击“清除”(Clear) 按钮。

清除 PLC 存储器要求 PLC 处于 STOP 模式，然后根据选择删除所选块或将 PLC 复位为出厂默认值。清除操作并不清除 IP 地址和站名称，也不复位日时钟。

执行后，“复位为出厂默认值”(Reset to factory defaults) 删除所有块，将所有用户存储器都复位为初始上电状态，并将所有“特殊存储器” (页 1265)都复位为初始值。

忘记 PLC 密码怎么办

如果忘记 PLC 密码 (页 62)，可以从专为此目的而设计的“复位为出厂默认存储卡”(reset-to-factory-defaults memory card) (页 241) 清除 PLC 存储器（标准 CPU 型号）。

使用复位为出厂默认值存储卡清除 PLC

对于标准 CPU，可使用先前的复位为出厂默认值存储卡清除 PLC。

⚠ 警告

将存储卡插入 CPU

在 RUN 模式下将存储卡插入 CPU 导致 CPU 自动切换到 STOP 模式。如果已插入存储卡，无法将 CPU 更改为 RUN 模式。

将存储卡插入正在运行的 CPU 可导致过程操作中断，可能引起人员死亡或严重伤害。

插入存储卡前，务必确保 CPU 处于 STOP 模式 (页 145)。

要使用此卡清除 PLC，请按以下步骤操作：

1. 插入复位为出厂默认存储卡。CPU 切换到 STOP 模式并且 STOP LED 闪烁。
2. 对 CPU 循环上电。CPU 使 RUN/STOP LED 闪烁，直到复位完成（大约一秒），然后 STOP LED 闪烁，表示复位结束。
3. 卸下存储卡。
4. 对 CPU 循环上电。CPU 复位为出厂默认值。之前的 IP 地址和波特率设置都均已清除，但日时钟不受影响。

CPU 复位后，可分配一个新密码并开始编程，也可从硬盘或从另一个程序传送存储卡加载程序 (页 141)。

说明

如果从存储卡或硬盘上的文件加载密码保护程序，必须输入密码才能访问保护区域。没有密码不能访问密码保护程序组件，不输入密码也不能清除分配的密码。

7.1.14 创建复位为出厂默认存储卡

可创建一个将标准 S7-200 SMART CPU 返回到出厂默认状态的存储卡。如果要清除标准 CPU 的内容，可使用此复位为出厂默认存储卡。要创建复位为出厂默认存储卡，按以下步骤操作：

1. 使用读卡器和 Windows 资源管理器删除 microSDHC 卡中的所有内容。
2. 使用 Notepad 等编辑器创建一个包含一行字符串“RESET_TO_FACTORY”的简单文本文件。（不要输入引号。）
3. 将此文件以文件名“S7_JOB.S7S”保存到 microSDHC 卡根级别。
4. 贴上卡标签，将卡保存在安全位置以供日后使用。

7.2 高速 I/O

高速计数器

CPU 集成了高速计数器功能，可对高速外部事件进行计数而不会降低 CPU 的性能。有关 CPU 支持的速率的信息，请参见“产品概述” (页 25)章节。存在专用于时钟、方向控制和复位功能的输入，这些功能均受支持。可选择单相、双相或 AB 正交相以改变计数速率。有关详细信息，请参见高速计数器指令 (页 356)说明。

高速脉冲输出

标准 CPU 型号支持高速脉冲输出，可在某些输出上生成一个高速脉冲串输出 (PTO) 或脉宽调制 (PWM) 信号。有关 CPU 支持的数量和速率的信息，请参见“产品概述”(页 25)章节。

PTO 函数以指定脉冲数（从 1 到 2,147,483,647 个脉冲）和指定频率 (Hz) 提供一个方波（50% 负载循环）输出。可编写 PTO 函数以产生一个脉冲串或包含多个脉冲串的一个脉冲包络。例如，可使用一个脉冲包络通过一个简单的斜升、运行和斜降顺序或更复杂的顺序控制步进电机。

PWM 功能实现周期时间固定、占空比可变的输出，周期时间和脉冲宽度以微秒或毫秒为增量进行指定。当脉冲持续时间等于循环时间，负载循环为 100%，该输出持续打开。当脉冲持续时间为 0，负载循环为 0%，该输出关闭。

更多相关信息，请参见脉冲输出指令 (页 381)。有关使用 PWM (页 1095) 的详细信息，请参见开环运动控制章节。

开环运动控制

标准 CPU 型号支持开环运动控制功能。运动曲线可以进行构成并执行，可在用户程序控制下执行交互式移动，并可使用若干内置参考点搜索序列。

根据组态的不同，CPU 中要支持开环运动，需要使用某些 CPU 资源，如高速输出、高速计数器和沿中断。

有关 CPU 支持的运动轴数量和脉冲速率的信息，请参见“产品概述”(页 25)章节。

有关 CPU 中各运动功能的完整介绍，请参见开环运动控制 (页 1095)的相关章节。

Web 服务器 8

8.1 Web 服务器概述

S7-200 SMART 有一个内置的 Web 服务器，可用于从 PC 或移动设备操作 S7-200 SMART。

通过这种方法，可使用所连设备（传统 PC、平板电脑或带网络浏览功能的智能手机）输入 CPU 的 IP 地址或 URL 来访问 S7-200 SMART CPU。

通过 Web 服务器，可以使用鼠标指针或触摸屏（取决于所使用的设备）在虚拟化的 S7-200 SMART 上执行快速、简单的操作。

S7-200 SMART 支持 **HTTPS** 用于 Web 服务器通信。

支持的 Web 浏览器

Web 服务器支持以下 Web 浏览器：

对于 PC：

- Google Chrome 63.0 或以上版本
- Mozilla Firefox 67.0 或以上版本
- Microsoft Edge 18.0 或以上版本
- MacOS Safari 12.0 或以上版本

对于移动设备：

- IOS Safari 13.0 或以上版本
- Android Chrome 80.0 或以上版本
- Android Firefox 68.0 或以上版本

说明

有关 Web 浏览器的注意事项：

- 最新版本的 Google Chrome、Microsoft Edge、Mozilla Firefox 和 Apple Safari 提供了更好的可用性和安全性。
- 用户的隐私数据（例如用户名、密码、Cookie、临时文件、历史记录和 Web 表单信息）由浏览器进行管理。应在浏览器设置中组态适当的安全保护策略。

组态 Web 服务器

在使用 Web 服务器之前，需要完成以下任务：

- 使用 Web 服务器向导组态 Web 服务器
- 使用证书管理为 Web 服务器组态 HTTPS 证书

具体步骤，请参阅 STEP 7-Micro/WIN SMART V3 在线帮助中的“工具”>“向导”>“Web 服务器向导”。

标准 Web 页面

S7-200 SMART 包含可通过 PC 的 Web 浏览器进行访问或通过移动设备访问的标准 Web 页面：

- 简介：Web 页面的进入点
- 起始页面：有关 CPU 的常规信息
- 模块信息：有关本地和远程模块的信息
- 监控表：允许用户查看先前在 Web 服务器向导中组态的监控表
- 数据日志：允许用户查看 CPU 上所有数据日志的列表，将数据日志从 CPU 下载到计算机，从 CPU 清除数据日志以及从 CPU 检索或清除数据日志
- 事件日志：允许用户查看 CPU 上所有事件日志的列表，或将数据日志从 CPU 下载到计算机
- 状态图：可以编辑图表行以添加想要通过 Web 服务器监视的 CPU 数据地址
- 设置时钟：设置 CPU 时间

说明

如果设备存在漏洞风险，西门子建议清除证书 (页 239)，并为 CPU 重新生成证书。

用户自定义 Web 页面

S7-200 SMART 支持访问用户自定义 Web 页面和获取 CPU 运行信息。要创建用户自定义 Web 页面，可使用 SMART Web Editor (SWE) 工具创建项目，并将用户自定义 Web 资源文件下载到连接的 CPU；也可使用自选的其它 HTML 编写软件。具体信息，请参见用户自定义 Web 页面 (页 257)一章。

有关此工具的详细信息，请参见 *SMART Web Editor 在线帮助*。

要访问用户自定义 Web 页面，可在 PC 或移动设备上通过已连接 CPU 的 IP 地址或 URL 访问，也可通过标准 Web 页面上的用户自定义主页选项卡访问。

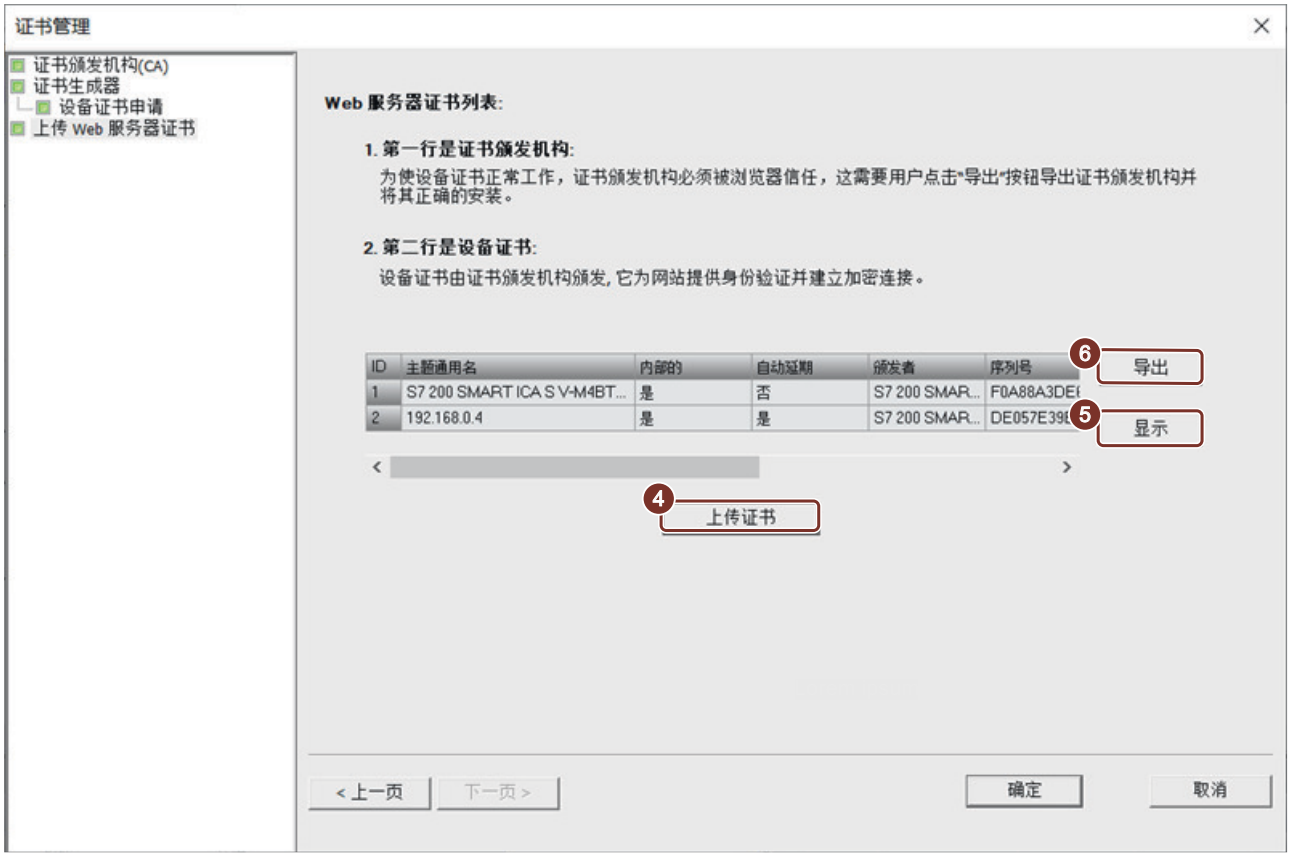
8.2 下载并安装 CA 证书

要从浏览器访问 Web 服务器，需要在 Internet 选项中安装证书颁发机构 (CA) 证书。

8.2.1 获取 CA 证书

可通过以下方式获取证书颁发机构的证书：

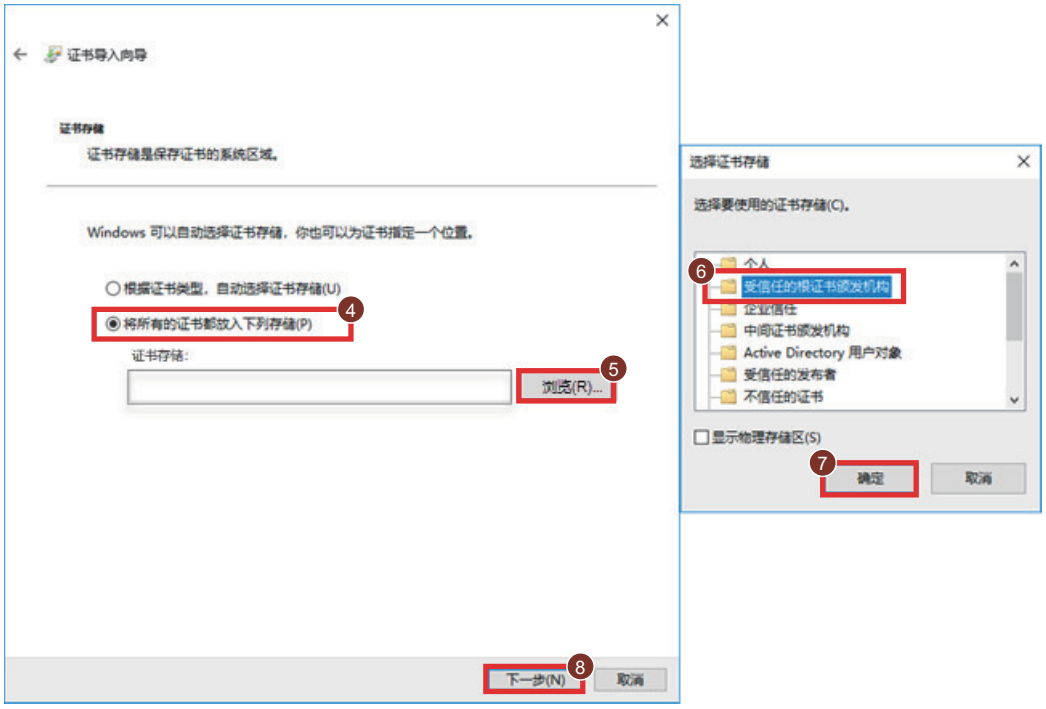
1. 与 CPU 建立以太网通信。
2. 打开证书管理向导。
3. 在导航窗格中选择“上传 Web 服务器证书”(Upload Web server certificates)。
4. 单击“上传证书”(Upload Certificate)。
   CPU 上的证书将上传到 Web 服务器证书表中：第一行显示 CA 证书。
5. 要显示证书，请在表格中将其选中，然后单击“显示”(Show)。
6. 要导出证书，请在表格中将其选中，然后单击“导出”(Export)，再导航到目标文件夹，之后单击“保存”(Save)。



8.2.2 安装用于 Windows 的证书

安装用于 Windows 的证书

1. 双击证书开始安装。
2. 在弹出窗口中检查认证信息，然后单击“安装证书”(Install Certificate) 以继续操作。
3. 在证书导入向导欢迎页面中，选择“存储位置”(Store location)，然后单击“下一步”(Next)。
   – 如果选择“当前用户”(Current User)，则证书只对当前用户有效。
   – 如果选择“本地计算机”(Local Machine)，则证书对此 PC 上的所有用户都有效。只有管理员才能将证书安装至“本地计算机”(Local Machine)。
4. 选择复选框 ④，然后单击 ⑤ 存储证书。
5. 在弹出窗口中选择 ⑥ 信任证书，然后单击 ⑦。
6. 单击 ⑧ 继续操作。



7. 在“证书导入向导”(Certificate import wizard) 窗口中，单击“完成”(Finish) 以确认选择。
8. 在安全警告窗口中，单击“是”(Yes) 确认安装。

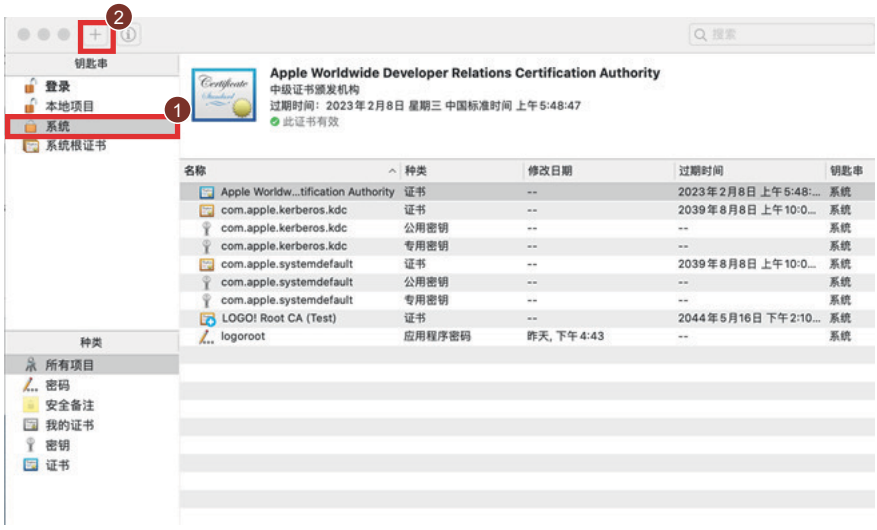
将证书导入 Firefox

如果安装证书后 Firefox 仍然无法信任证书，则在系统管理员的指导下按照 Firefox 中的说明将证书导入 Firefox。

8.2.3 安装用于 Mac OS 的证书

安装用于 MAC OS 的证书

1. 打开密钥链。
2. 要添加证书，请选择“系统”(System) ①，然后单击“+”②。



3. 选择 CA 的证书，然后单击“打开”(Open) 进行添加。
4. 输入密码，然后单击“修改密钥链”(Modify Keychain)。
5. 双击 CA 的证书以将其打开。
6. 通过单击“信任”(Trust) 并将“安全套接层 (SSL)”(Secure Sockets Layer (SSL)) 设置为“始终信任”(Always trust) 来信任证书。
7. 输入密码，并确认修改。

8.2.4 安装用于移动设备的证书

在系统管理员的指导下按照操作系统和浏览器的要求安装证书。

8.3 从 PC 或移动设备访问 Web 服务器

可以通过 S7-200 SMART CPU 的 IP 地址从 PC 或从移动设备访问 S7-200 SMART Web 服务器。

要从 PC 或移动设备访问 S7-200 SMART Web 服务器，请按以下步骤操作：

1. 请确保 S7-200 SMART 和 PC 或移动设备位于同一个以太网中，或二者之间直接使用标准以太网电缆进行连接。
2. 打开 Web 浏览器，输入 URL“https://ww.xx.yy.zz”，其中“ww.xx.yy.zz”与 S7-200 SMART CPU 的 IP 地址相对应。

Web 浏览器将打开“简介”(Introduction) Web 页面。

通过 URL 访问 Web 页面

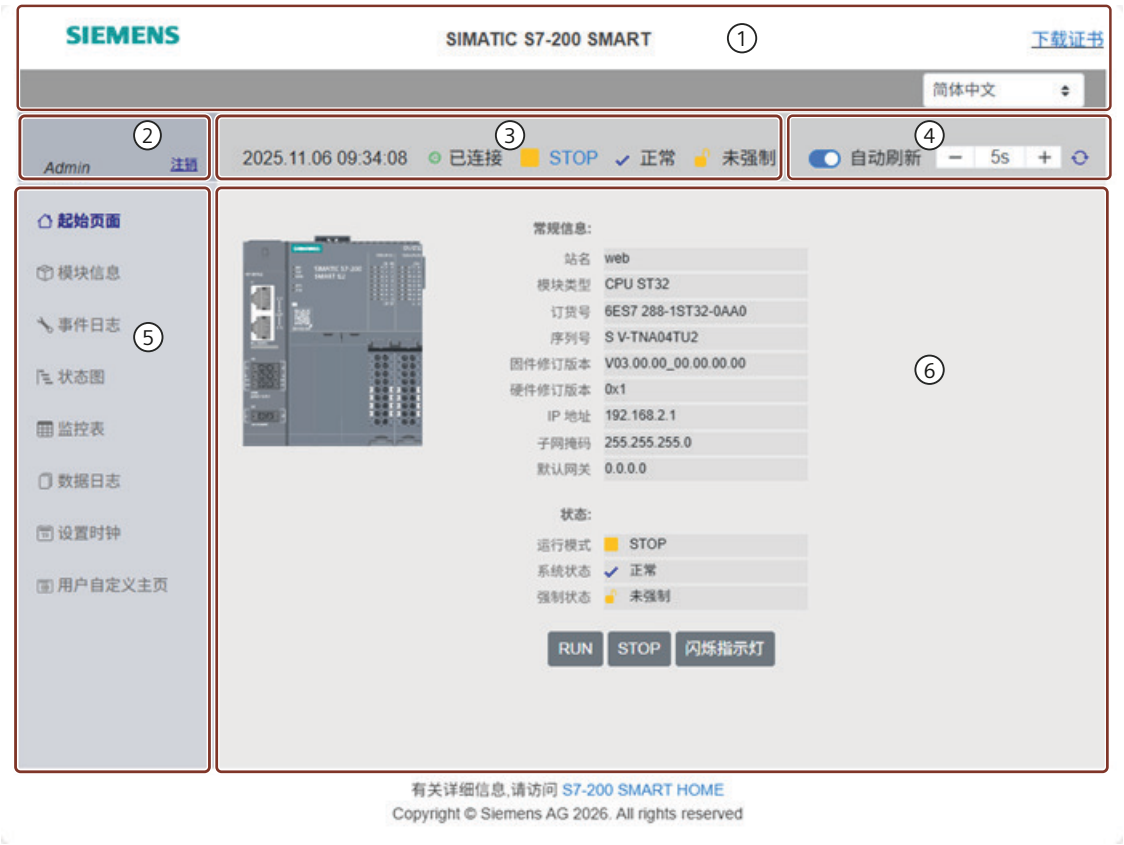
可以通过输入页面 URL 来访问 Web 页面。下表列出了各页面对应的 URL：

Web 页面	**URL**
标准 Web 页面	
简介页面	https://PLCIP
起始页面	https://PLCIP/start
模块信息页面	https://PLCIP/module
事件日志页面	https://PLCIP/eventlog
状态图页面	https://PLCIP/statuschart
监控表页面	https://PLCIP/watchtable
数据日志页面	https://PLCIP/datalog
设置时钟页面	https://PLCIP/setclock
用户自定义 Web 页面	
登录页面	https://PLCIP/user_define/login.html
主页	https://PLCIP/user_define/

8.4 标准 Web 页面

8.4.1 Web 页面的布局

无论是在 PC 还是在移动设备上查看页面，S7-200 SMART Web 服务器的每个页面都具有相同的内容区域，但布局和导航控件会根据屏幕大小和设备分辨率发生变化。在 PC 或大型移动设备上，Web 页面的布局如下：



① 标头：产品名称、证书下载链接和显示语言选择器。

② 登录/注销

③ PLC 状态：显示时间、CPU 操作模式、系统状态、强制状态和连接状态。

切换操作模式：可以通过单击“运行”(RUN) 或“停止”(STOP) 切换操作模式。

④ 刷新按钮

- 自动刷新 (Auto refresh)：如果选择打开“自动刷新”(Auto refresh) 按钮，则默认每五秒钟刷新一次 Web 页面。此外，还可以设置自定义刷新时间周期（以秒为单位）。
- 手动刷新 (Manual refresh)：单击以刷新 Web 页面。

⑤ Web 页面导航

⑥ 特定 Web 页面的详细信息。可使用以下按钮来控制 CPU

- RUN：将 CPU 设置为运行模式。
- STOP：将 CPU 设置为停止模式。
- 启动 LED 闪烁：使 CPU 上的 LED 指示灯闪烁。

说明

用户自定义主页选项卡

用户自定义主页是用户自定义 Web 页面主页的入口点。

8.4.2 简介

输入特定 Web 页面的 URL 并按进入按钮时，将显示简介页面。“简介”(Introduction) 页面是进入 S7-200 SMART Web 页面的欢迎画面。在该页面中，单击“进入”(Enter) 可访问 S7-200 SMART Web 页面。

要登录，请按以下步骤操作：

1. 在“用户名称”(User Name) 字段中输入用户名称。
2. 在“密码”(Password) 字段中输入用户密码。
3. 单击“登录”(Log in)。

说明

如果控制组态中已“禁用 PLC 访问控制”或“启用 PLC 保护”，可使用默认管理员帐户及其初始密码登录 Web 服务器。

如果已启用用户管理，可使用所定义的所有用户帐户及相应密码登录。

如果持续三十分钟没有操作，则登录会话将过期。如果打开自动刷新，则登录会话超时将重置，以防止会话过期。

8.4.3 起始页面

“起始页面”(Start page) 显示所连接 CPU 的常规信息和状态。

可通过单击以下按钮来控制 CPU：

- RUN：将 CPU 转为 RUN 模式
- STOP：将 CPU 置于 STOP 模式
- LED 指示灯闪烁：使 STOP、RUN 和 FAULT 指示灯闪烁

8.4.4 事件日志

“事件日志”(Event Log) 页面显示带有时间戳的事件的表。该表列出了时间、事件类型以及错误或原因信息。

可能的事件日志条目的最大数量为 128。

可以单击 “事件日志”(Event log) 或图标 在 Microsoft Excel 电子表格中下载事件日志。

8.4.5 状态图

“状态图”(Status Chart) 页面可用于监视 PLC 数据值。可以分配存储器地址、格式和新值，以及将更改应用于 CPU。将状态图页面添加为书签后，可以在下次快速访问同一图表。

说明

- 新地址的最大数量为 30。
- 要查看“状态图”(Status Chart) 页面，需要具有“状态图只读”(Status chart read only) 权限。
- 要查看和修改数据值，则需要具有“状态图读写”(Status chart read and write) 权限。

说明

- 以下存储区无法写入：I、AI、SM、HC、T 位和 C 位。
- 如果 CPU 处于停止模式，则不支持在“状态图”(Status Chart) 页面中应用更改。
- 如果状态图中有多个项目发生更改，并且其中一项存在错误代码，则无法成功应用所有更改。

8.4.6 监控表

“监控表”(Watch Table) 页面可用于检查和修改 IO 数据。

说明

必须在 Web 服务器向导中组态监控表。

可从下拉列表中选择一个监控表，然后更改变量格式或为变量分配新值。

可以根据需要清除新值，或单击“全部应用”(Apply All) 按钮以保存所有更改。

说明

- 以下存储区无法写入：I、AI、SM、HC、T 位和 C 位。
- 如果 PLC 处于停止模式，则不支持在“监控表”(Watch Table) 页面中应用更改。
- 如果监控表中有多个项目发生更改，并且其中一项存在错误代码，则无法成功应用所有更改。

8.4.7 数据日志

“数据日志”(Data Log) 页面可用于访问所连接 CPU 的数据日志文件。

在数据日志向导中选择的数据日志上传选项与“数据日志”(Data Log) 页面中的选择一致。例如，如果在数据日志向导中为上传选项选中了“上传时清除数据日志中的所有记录”(Clear all records from the data log on upload) 复选框，则“数据日志”(Data Log) 页面将显示“是”(Yes)。

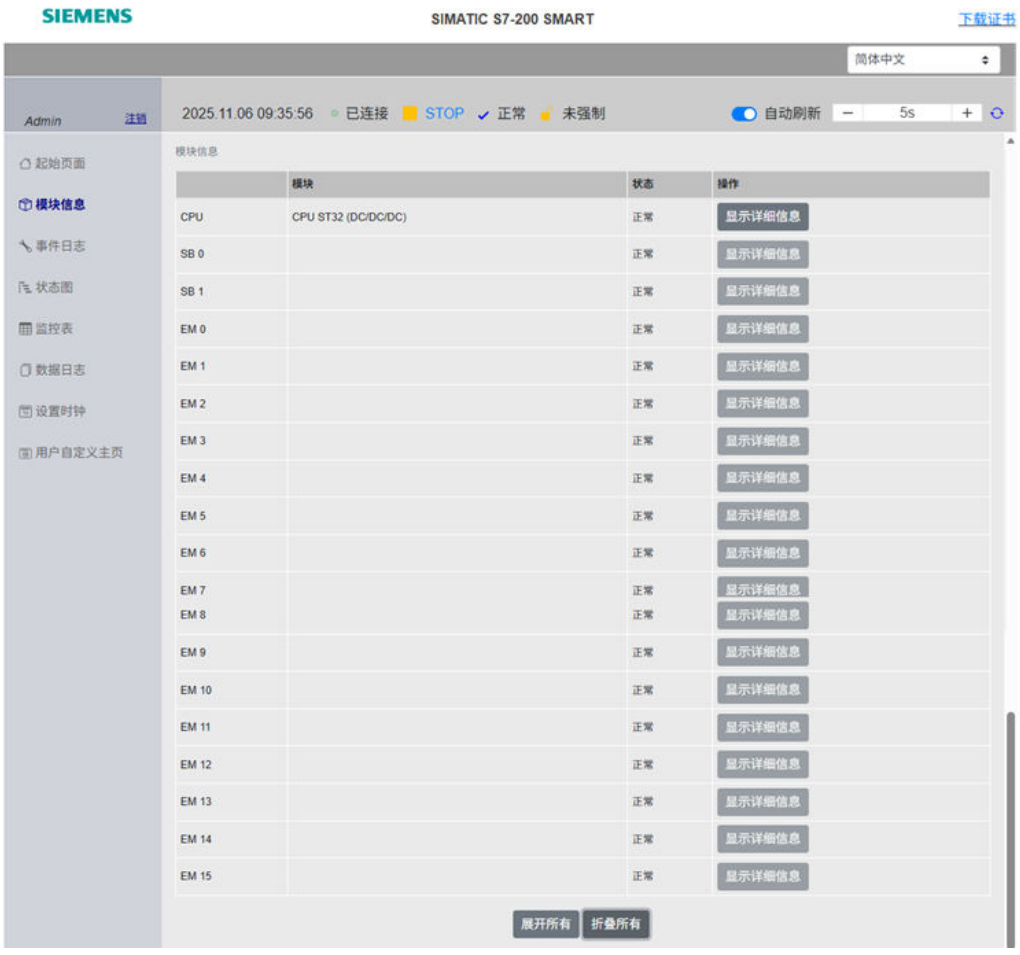
可以单击“数据日志”(Data Log) 名称以 CVS 格式下载数据日志文件。

8.4.8 设置时钟

“设置时钟”(Set Clock) 页面可用于设置时间和日期以及组态夏令时。

8.4.9 模块信息

“模块信息”(Module Information) 页面显示 CPU、扩展模块和 PROFINET 设备信息。



可使用以下选项来检查模块信息：

- 全部展开 (Expand All)：显示每个模块的详细信息。
- 全部折叠 (Collapse All)：隐藏每个模块的详细信息。
- 显示详细信息 (Show Details)：为特定模块显示详细信息。
- 隐藏详细信息 (Hide Details)：为特定模块隐藏详细信息。

8.5 用户自定义 Web 页面

S7-200 SMART 支持用户定制自己的 Web 页面，并可连接 CPU 以获取 CPU 运行信息。该过程涉及以下步骤：

1. 通过 SMART Web Editor 或其它任何 HTML 编辑器工具创建用户自定义 Web 页面资源。
   有关此工具的使用方法以及用户自定义 Web 页面资源的创建方法的详细信息，请参见 *SMART Web Editor 在线帮助*。

   说明

   Bin 文件大小。

   通过用户自定义 Web 页面资源生成的 bin 文件的大小不能超过 2 M，否则资源无法成功下载到 CPU。

2. 将用户自定义 Web 资源文件下载到连接的 CPU (页 257)。
3. 使用 Web 服务器向导组态 Web 服务器。
4. 使用证书管理为 Web 服务器组态 HTTPS 证书。
5. 访问用户自定义 Web 页面。
   可以通过 URL 访问用户自定义 Web 页面，也可以使用标准 Web 页面上的用户自定义主页选项卡作为入口点。

⚠ 警告

通过用户自定义 Web 页面对 CPU 进行未经授权的访问

用户自定义 Web 页面中的不安全代码会引入跨站脚本 (XSS)、代码注入等安全漏洞。

按照“操作准则”中介绍的安全方式安装 S7-200 SMART CPU 可防止未经授权的访问，这一准则可在工业安全网站 (http://www.siemens.com/industrialsecurity)上找到。

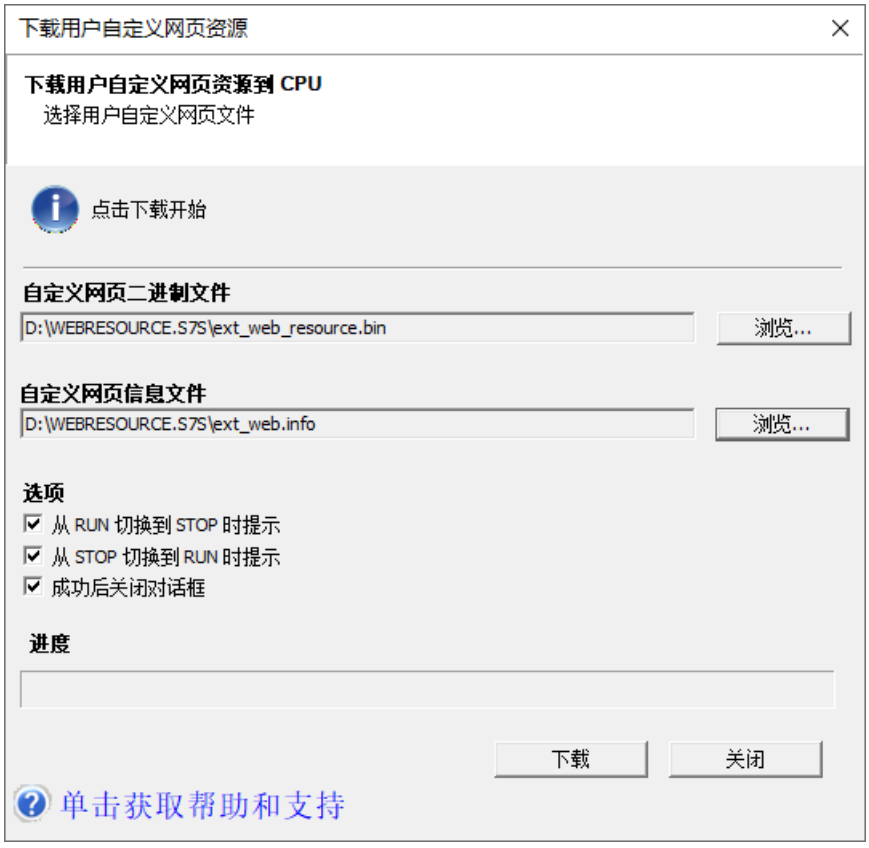
8.5.1 下载用户自定义 Web 资源文件

可通过两种方法将用户自定义 Web 资源文件下载到连接的 CPU

- 通过 Step 7-Micro/WIN SMART V3 将 Web 资源文件下载到连接的 CPU
- 通过 microSDHC 卡将 Web 资源文件传输到连接的 CPU。

通过 Step 7-Micro/WIN SMART V3 下载

1. 单击“**文件**”(File) 或“**PLC**”菜单“**传输**”(Transfer) 部分中的“**下载**”(Download) 下拉列表按钮，然后选择“**用户自定义 Web**”(User Defined Web)。将出现“下载用户自定义 Web 资源”(Download User Defined Web Resource) 窗口。
2. 单击“**浏览**”(Browse) 按钮分别在弹出的下载窗口中上传网页资源二进制文件和信息文件。



3. 单击“**下载**”(Download) 按钮开始下载过程。

通过 microSDHC 卡传送

将 Web 资源文件放入存储卡的根目录，且卡的层级结构如下：

- 名为 WEBRESOURCE.S7S 的文件夹，包含 Web 资源文件：ext_web_resource.bin; ext_web.info
- 名为 S7_JOB.S7S 的文本文件，包含字 EXT_WEB

8.6 Web API（应用程序编程接口）

CPU 提供基于 Web 的 API (Web API)，用作读取和写入 CPU 数据的接口。
通过 Web API，可以：

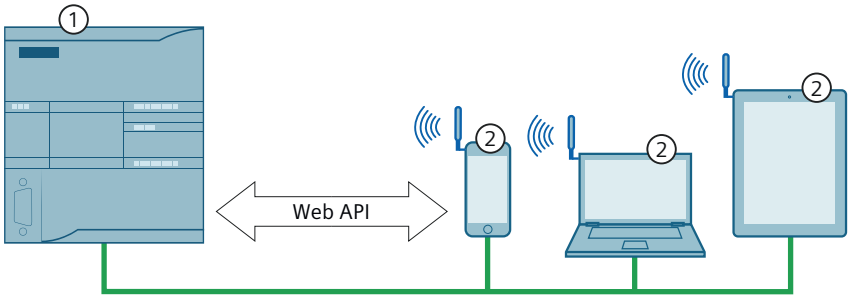
- 实施基于最新、最先进技术的 Web 应用程序。
- 通过脚本和编程语言与 CPU 的 Web 服务器进行通信。
- 创建同时连接多个 CPU 的 Web 应用程序，例如，创建显示多个 CPU 状态的仪表板等。

说明

Web API 仅可用于固件版本不低于 V2.7 的 CPU。

Web API 连接

下图给出了 CPU 与终端设备之间的 Web API 示例。



① CPU

② 终端设备

CPU 通过 PROFINET 端口支持 Web API。

Web API 功能

Web API 可用于远程访问 CPU 数据。组态用户权限后，可读取或写入所连接 CPU 的数据。

Web API 方法

Web API 支持以下方法。有关这些指令和编程模板的详细信息，请参见《*S7-200 SMART Web API 开发手册*》。

指令	说明
login	登录 CPU
logout	注销 CPU
get_permission	获取授权操作列表
read	读取 CPU 的过程数据
write	将过程数据写入 CPU
browse	浏览 CPU 支持的 Web API 列表

程序指令

9

9.1 位逻辑

9.1.1 标准输入

LAD	FBD	STL	说明
bit	AND OR	LD bit A bit O bit	测试存储器（M、SM、T、C、V、S、L）或过程映像寄存器（I 或 Q）中的位值。 **LAD**：常开和常闭开关通过触点符号进行表示。如果能流位于左侧且触点闭合，则能流将通过触点流向右侧的连接器，流至下一连接元件。 • 常开 (N.O.)位值为 1 时，LAD 触点闭合 (ON)。 • 常闭 (N.C.)位值为 0 时，LAD 触点闭合 (ON)。 **FBD**：常开指令通过 AND/OR 功能框进行表示。功能框指令可用于评估布尔信号，评估方式与梯形图触点程序段相同。常闭指令也通过功能框进行表示。在二进制输入信号连接器上放置取反圆圈 AND，即可创建常闭指令。AND/OR 功能框输入的数量最多可扩展至 31 个。 **STL**：常开触点通过 LD（装载）、AN（“与”运算）和 ON（“或”运算）指令进行表示。这些指令使用逻辑堆栈顶部位的值对寻址位的值执行装载、“与”运算或者“或”运算。常闭触点通过 LDN（取反后装载）、AN（“与非”运算）和 ON（“或非”运算）指令进行表示。这些指令使用逻辑堆栈顶部位的值对寻址位值的逻辑非运算值执行装载、“与”运算或者“或”运算。
bit /	AND OR	LDN bit AN bit ON bit	

输入/输出	数据类型	操作数
位（LAD、STL）	BOOL	I、Q、V、M、SM、S、T、C、L
输入 (FBD)	BOOL	I、Q、V、M、SM、S、T、C、L、逻辑流
输出 (FBD)	BOOL	I、Q、V、M、SM、S、T、C、L、逻辑流

FBD AND/OR 输入分配

仅当选中 FBD 功能框光标内的输入短线且短线为红色时，下表所述的编辑器功能才处于激活状态。

输入选项	光标放置	工具按钮	快捷键
添加输入	在功能框上		+
移除输入	在功能框和底部输入上		-

另请参见

位逻辑输入示例 (页 276)

逻辑堆栈概述 (页 264)

9.1.2 立即输入

LAD	FBD	STL	说明
bit I	AND OR	LDI bit AI bit OI bit	在该立即指令执行时，该指令获取物理输入值，但不更新过程映像寄存器。立即触点不会等待 PLC 扫描周期进行更新，而是会立即更新。 物理输入点（位）状态为 1 时，常开立即触点闭合（接通）。 物理输入点（位）状态为 0 时，常闭立即触点闭合（接通）。 **LAD**：常开和常闭立即指令通过触点进行表示。 **FBD**：输入连接前面垂直的立即指示符 AND 即代表立即常开指令。 输入连接前面的立即指示符和取反圆圈 AND 则代表立即常闭指令。 使用逻辑流连接而不是物理输入 (I) 位地址时，不能使用立即指示符。 FBD 功能框指令可用于评估物理信号，评估方式与梯形图触点相同。AND/OR 功能框输入的数量最多可扩展至 31 个。 **STL**：常开立即触点通过 LDI（立即装载）、AI（立即与）和 OI（立即或）指令进行表示。这些指令使用逻辑堆栈顶部的值对物理输入值执行装载、“与”运算或者“或”运算。 常闭立即触点通过 LDNI（取反后立即装载）、ANI（取反后立即与）和 ONI（取反后立即或）指令进行表示。这些指令使用逻辑堆栈顶部的值对物理输入值的逻辑非运算值执行立即装载、“与”运算或者“或”运算。
bit /I	AND OR	LDNI bit ANI bit ONI bit	

输入/输出	数据类型	操作数
位（LAD、STL）	BOOL	I
输入 (FBD)	BOOL	I

FBD 编辑器输入分配

仅当选中 FBD 功能框光标内的输入短线且短线为红色时，下表所述的编辑器功能才处于激活状态。

输入选项	光标放置	工具按钮	快捷键
添加输入	在功能框上		+
移除输入	在功能框和底部输入上		-
切换取反输入	在功能框和输入上		F11
切换立即输入	在功能框和输入上		CTRL F11

另请参见

位逻辑输入示例 (页 276)

逻辑堆栈概述 (页 264)

9.1.3 逻辑堆栈概述

STEP 7-Micro/WIN SMART V3 程序编译器使用逻辑堆栈将 LAD 和 FBD 程序的图形 I/O 程序段转换为 STL（语句表）程序。得出的 STL 程序在逻辑上与原始 LAD 或 FBD 图形程序段相同，并且可作为程序表执行。所有成功编译的 LAD 和 FBD 程序均已生成基本 STL 程序，并可被视为 LAD、FBD 或 STL。

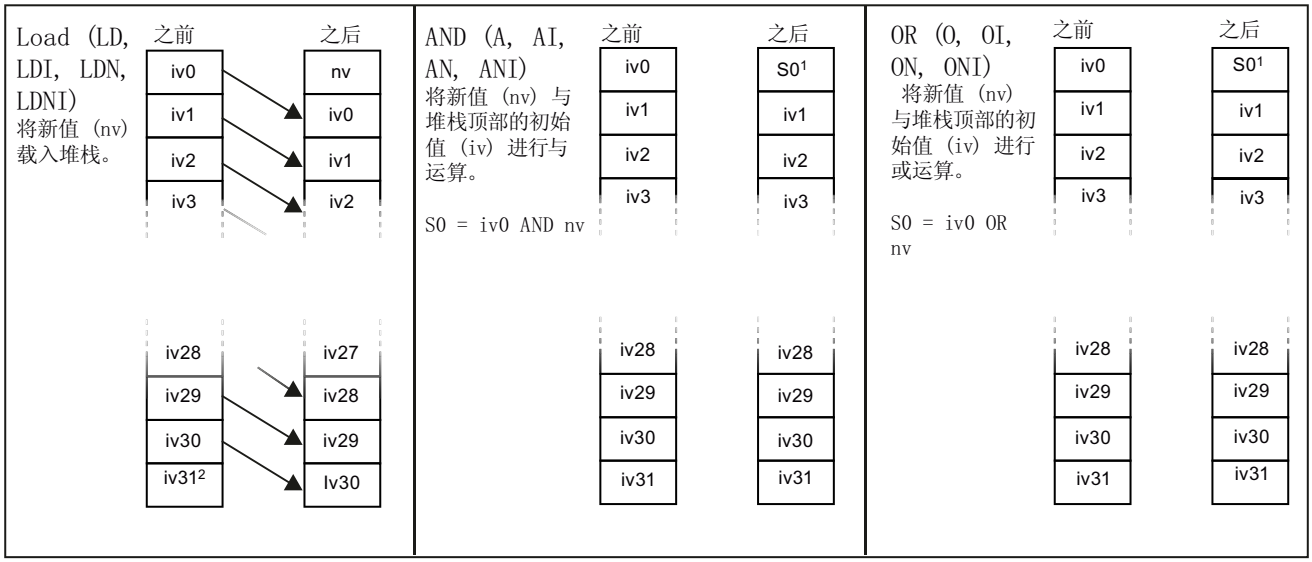
对于 LAD 和 FBD 编辑，会自动生成 STL 逻辑堆栈指令，并且程序员不需要使用逻辑堆栈指令。

还可使用 STL 编辑器直接创建 STL 程序。STL 程序员可直接用逻辑堆栈指令。可在 STL 编辑器中创建组合逻辑，该组合逻辑过于复杂，无法在 LAD 或 FBD 编辑器中查看，但某些特殊应用可能必须使用该逻辑。

所有成功编译的 LAD 和 FBD 程序均可在 STL 中查看，但并不是所有成功编译的 STL 程序均可在 LAD 或 FBD 中查看。

输入程序段和逻辑堆栈

如下图所示，CPU 使用逻辑堆栈来合并 STL 输入的逻辑状态。在这些示例中，“iv0”至“iv31”用于标识逻辑堆栈层的初始值，“nv”用于标识指令提供的新值，“S0”用于标识存储在逻辑堆栈中的计算值。



[1] S0 用于标识存储在逻辑堆栈中的计算值。
[2] 执行装载后，值 iv31 丢失。

输出程序段和逻辑堆栈

ENO 是 LAD 和 FBD 中功能框的二进制输出。如果 LAD 功能框的 EN 输入有能流并且无错误执行，则 ENO 输出会将能流传递到下一 LAD 元素。可将用于指示指令成功完成的 ENO 用作使能位。ENO 位用于堆栈顶端，影响用于后续指令执行的能流。STL 指令没有 EN 输入。栈顶值必须为逻辑 1，条件指令才能执行。在 STL 中，没有 ENO 输出。但是，与具有 ENO 输出的 LAD 和 FBD 指令相对应的 STL 指令可置位特殊 ENO 位。可通过 “与 ENO”(AENO) 指令访问该位。

STL	描述
AENO	AENO 在 LAD/FBD 功能框 ENO 位的 STL 表示中使用。AENO 对 ENO 位和栈顶值执行逻辑与运算，产生的效果与 LAD/FBD 功能框的 ENO 位相同。与操作的结果值成为新的栈顶值。

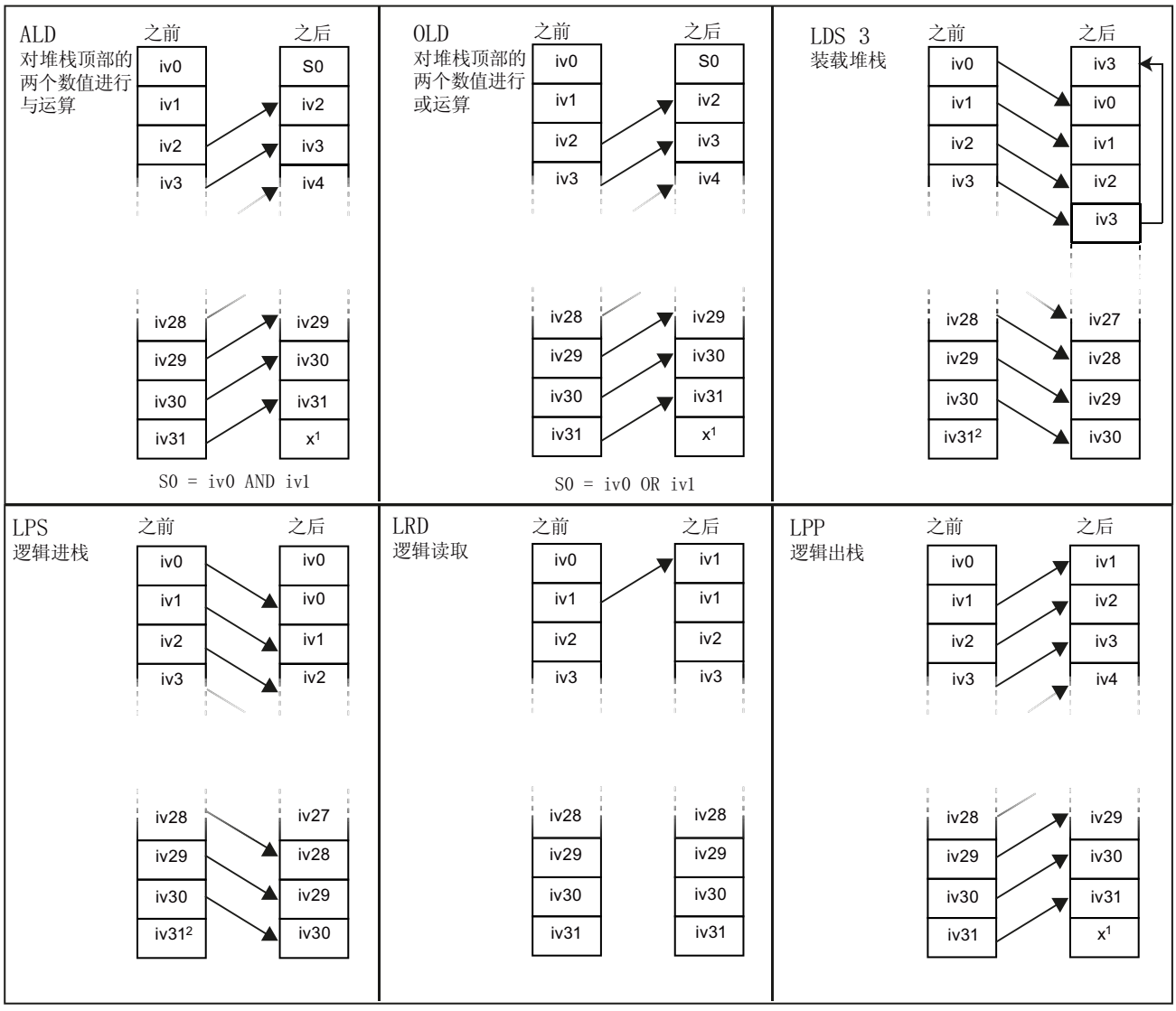
9.1.4 STL 逻辑堆栈指令

STL[1]	说明
ALD	与装载指令 (ALD) 对堆栈第一层和第二层中的值进行逻辑与运算。 结果装载到栈顶。 执行 ALD 后，栈深度减一。
OLD	或装载指令 (OLD) 对堆栈第一层和第二层中的值进行逻辑或运算。 结果装载到栈顶。 执行 OLD 后，栈深度减一。
LPS	逻辑进栈指令 (LPS) 复制堆栈顶值并将该值推入堆栈。栈底值被推出并丢失。
LRD	逻辑读栈指令 (LRD) 将堆栈第二层中的值复制到栈顶。 此时不执行进栈或出栈，但原来的栈顶值被复制值替代。
LPP	逻辑出栈指令 (LPP) 将栈顶值弹出。 堆栈第二层中的值成为新的栈顶值。
LDS N	装载堆栈指令 (LDS) 复制堆栈中的栈位 (N) 值，并将该值置于栈顶。 栈底值被推出并丢失。
AENO	AENO 在 LAD/FBD 功能框 ENO 位的 STL 表示中使用。 AENO 对 ENO 位和栈顶值执行逻辑与运算，产生的效果与 LAD/FBD 功能框的 ENO 位相同。 与操作的结果值成为新的栈顶值。

[1] 不适用于 LAD 或 FBD

LDS（装入堆栈）输入	数据类型	操作数
N	BYTE	常数（0 到 31）

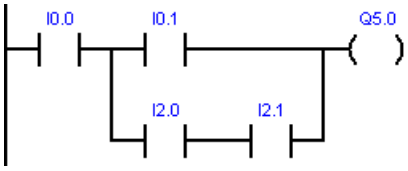
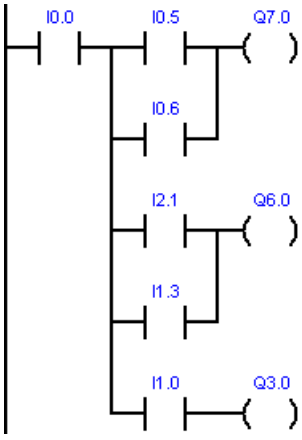
如下图所示，CPU 使用逻辑堆栈来解决控制逻辑。 在这些示例中，“iv0”至“iv31”用于标识逻辑堆栈的初始值，“nv”用于标识指令提供的新值，“S0”用于标识存储在逻辑堆栈中的计算值。



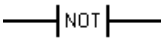
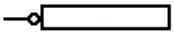
[1] 该值未知（可以是 a 0 或 a 1）。

[2] 执行逻辑进栈或装入堆栈指令后，值 iv31 丢失。

逻辑堆栈示例： 将 LAD 程序段转换为 STL 代码

LAD	STL
	Network 1 LD I0.0 LD I0.1 LD I2.0 A I2.1 OLD ALD = Q5.0
	Network 2 LD I0.0 LPS LD I0.5 O I0.6 ALD = Q7.0 LRD LD I2.1 O I1.3 ALD = Q6.0 LPP A I1.0 = Q3.0

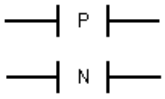
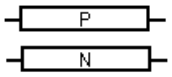
9.1.5 NOT

LAD	FBD	STL	说明
		NOT	取反指令 (NOT) 取反能流输入的状态。 **LAD：** NOT 触点会改变能流输入的状态。 能流到达 NOT 触点时将停止。 没有能流到达 NOT 触点时，该触点会提供能流。 **FBD：** NOT 指令在布尔功能框输入连接器上有取反符号，该指令的作用与逻辑状态取反器相同。 **STL：** NOT 指令会将堆栈顶值从 0 更改为 1 或从 1 更改为 0。

另请参见

位逻辑输入示例 (页 276)

9.1.6 正跳变和负跳变检测器

LAD	FBD	STL	说明
		**EU** **ED**	正跳变触点指令（上升沿）允许能量在每次断开到接通转换后流动一个扫描周期。 负跳变触点指令（下降沿）允许能量在每次接通到断开转换后流动一个扫描周期。 S7-200 SMART CPU 支持在程序中合计（上升和下降）使用 1024 条边缘检测器指令。 **LAD：** 正跳变和负跳变指令通过触点进行表示。 **FBD：** 跳变指令通过 P 和 N 功能框进行表示。 **STL：** EU（上升沿）指令用于检测正跳变。 如果检测到堆栈顶值发生 0 到 1 跳变，则将堆栈顶值设置为 1；否则，将其设置为 0。 ED（下降沿）指令用于检测负跳变。 如果检测到堆栈顶值发生 1 到 0 跳变，则将堆栈顶值设置为 1；否则，将其设置为 0。

输入/输出	数据类型	操作数
IN (FBD)	BOOL	I、Q、V、M、SM、S、T、C、L、逻辑流
OUT (FBD)	BOOL	I、Q、V、M、SM、S、T、C、L、逻辑流

说明

因为正跳变和负跳变指令需要断开到接通或接通到断开转换，所以无法在首次扫描时检测上升沿或下降沿跳变。 首次扫描期间，CPU 会将初始输入状态保存在存储器位中。 在后续扫描中，这些指令会将当前状态与存储器位的状态进行比较以检测是否发生转换。

另请参见

位逻辑输入示例 (页 276)

9.1.7 线圈: 输出和立即输出指令

LAD	FBD	STL	说明
bit	bit =	= bit	该输出指令将输出位的新值写入过程映像寄存器。 **LAD** 和 **FBD**：执行输出指令时，S7-200 SMART 会接通或断开过程映像寄存器中的输出位。分配的位被设置为等于能流状态。 **STL**：堆栈顶值复制到分配的位。
bit I	bit =I	=I bit	该立即输出指令执行时，指令会将新值写入物理输出和相应的过程映像寄存器单元。 **LAD** 和 **FBD**：执行立即输出指令时，物理输出点（位）立即被设置为等于能流状态。“I”表示一个立即地址引用；新值将写入物理输出点和相应的过程映像寄存器地址。这不同于非立即地址引用仅将新值写入过程映像寄存器。 **STL**：该指令立即将栈顶的值复制到所分配的物理输出位和过程映像地址。

输入/输出	数据类型	操作数
位	BOOL	I、Q、V、M、SM、S、T、C、L
位（立即）	BOOL	Q
输入 (LAD)	BOOL	能流
输入 (FBD)	BOOL	I、Q、V、M、SM、S、T、C、L、逻辑流

另请参见

位逻辑输出示例 (页 277)

9.1.8 置位、复位、立即置位和立即复位功能

LAD	FBD	STL	说明
bit S N	bit S N	S bit, N	置位 (S) 和复位 (R) 指令用于置位（接通）或复位（断开）从指定地址（位）开始的一组位 (N)。可以置位或复位 1 至 255 个位。 如果复位指令指定定时器位（T 地址）或计数器位（C 地址），则该指令将对定时器或计数器位进行复位并清零定时器或计数器的当前值。
bit R N	bit R N	R bit, N	
bit SI N	bit SI N	SI bit, N	立即置位和立即复位指令立即置位（接通）或立即复位（断开）从指定地址（位）开始的一组位 (N)。可立即置位或复位 1 至 255 个点。 “I”表示一个立即地址引用；新值将写入物理输出点和相应的过程映像寄存器单元。这不同于非立即地址引用仅将新值写入过程映像寄存器。
bit RI N	bit RI N	RI bit, N	

ENO = 0 时的非致命错误	受影响的 SM 位
• N = 0（零） • 0006H 间接地址 • 0091H 操作数超出范围	无

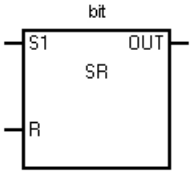
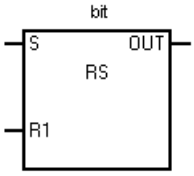
输入/输出	数据类型	操作数
位	BOOL	I、Q、V、M、SM、S、T、C、L
位（立即）	BOOL	Q
N	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、常数、*VD、*AC、*LD
输入 (LAD)	BOOL	能流
输入 (FBD)	BOOL	I、Q、V、M、SM、S、T、C、L、逻辑流

另请参见

位逻辑输入示例 (页 276)

位逻辑输出示例 (页 277)

9.1.9 置位和复位优先双稳态触发器

LAD/FBD [1]	描述
	位参数用于分配要置位或复位的布尔型地址。可选的 OUT 连接反映“位”(Bit) 参数的信号状态。 **SR**（置位优先双稳态触发器）是一种置位优先锁存器。如果置位 (S1) 和复位 (R) 信号均为真，则输出 (OUT) 为真。
	**RS**（复位优先双稳态触发器）是一种复位优先锁存器。如果置位 (S) 和复位 (R1) 信号均为真，则输出 (OUT) 为假。

[1] 不适用于 STL

输入/输出	数据类型	操作数
位	BOOL	I、Q、V、M、S
S1、R (LAD SR)	BOOL	能流
S、R1 (LAD RS)	BOOL	能流
OUT (LAD)	BOOL	能流
S1、R (FBD SR)	BOOL	I、Q、V、M、SM、S、T、C、L、逻辑流
S、R1 (FBD RS)	BOOL	I、Q、V、M、SM、S、T、C、L、逻辑流
OUT (FBD)	BOOL	I、Q、V、M、SM、S、T、C、L、逻辑流

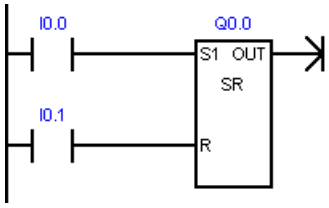
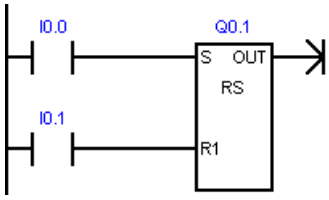
SR 真值表

S1	R	输出（位）
0	0	先前状态
0	1	0
1	0	1
1	1	1

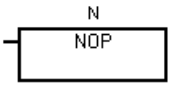
RS 真值表

S	R1	输出（位）
0	0	先前状态
0	1	0
1	0	1
1	1	0

SR 和 RS 示例

LAD	STL
	Network 1 LD I0.0 LD I0.1 NOT A Q0.0 OLD = Q0.0
	Network 2 LD I0.0 LD I0.1 NOT LPS A Q0.1 = Q0.1 LPP ALD O Q0.1 = Q0.1

9.1.10 NOP（空操作）指令

LAD	STL	说明
	NOP N	空操作 (NOP) 指令不影响用户程序的执行。 在 FBD 模式下，该指令不可用。 操作数 N 为 0 至 255 之间的数。

输入/输出	数据类型	操作数
N（LAD、STL）	BYTE	N： 常数（0 到 255）

9.1.11 交替输出和脉冲交替输出

LAD	FBD	STL	描述
bit —(ALT)	bit —[=ALT]	**=ALT bit**	能流有效时，交替输出指令 (ALT) 持续取反输出位的状态。如果不满足此条件，则输出位保持原样。 **LAD/FBD**：能流有效时，ALT 触点指令会更改输出位的状态。电流到达 ALT 触点时，输出位的状态会在每个扫描周期取反，从 0 取反为 1 或从 1 取反为 0。能流未到达 ALT 触点时，输出位保持原样。 **STL**：能流有效时，ALT 指令会将输出位的值保持在 0 和 1 之间切换。
bit —(ALTP)	bit —[=ALTP]	**=ALTP bit**	脉冲交替输出指令 (ALTP) 在每个上升沿取反输出位的状态。 **LAD/FBD**：ALTP 触点在每个上升沿更改输出位的状态。上升沿到达 ALTP 触点时，输出位的状态取反，从 0 取反为 1 或从 1 取反为 0。否则，输出位保持原样。 **STL**：ALTP 指令在每个上升沿将输出位的值在 0 和 1 之间切换。

输入/输出	数据类型	操作数
IN	BOOL	能流
OUT	BOOL	Q、V、M

另请参见

位逻辑输出示例 (页 277)

9.1.12 沿信号检测器

LAD	STL	说明
CLK R_TRIG Q	TRIGR	使用“检测信号上升沿”指令，可以检测输入 能流 CLK 的从“0”到“1”的状态变化。该指令将输入 CLK 的当前值与指定实例中保存的上次查询状态（通过边沿存储位记录）进行比较。 当指令检测到输入 CLK 的状态从“0”变成了“1”，会在输出 Q 中生成一个信号上升沿，输出将在一个循环周期内为 “ON”或“1”，并改变能流状态。 在其它任何情况下，该指令输出的信号状态均为“0”，无能流输出。
CLK F_TRIG Q	TRIGF	使用“检测信号下降沿”指令，可以检测输入 能流 CLK 的从“1”到“0”的状态变化。该指令将输入 CLK 的当前值与指定实例中保存的上次查询状态（通过边沿存储位记录）进行比较。 当指令检测到输入 CLK 的状态从“1”变成了“0”，会在输出 Q 中生成一个信号上升沿，输出将在一个循环周期内为 “ON”或“1”，并改变能流状态。 在其它任何情况下，该指令输出的信号状态均为“0”，无能流输出。

输入/输出	数据类型	操作数
CLK(FBD)	BOOL	I、Q、V、M、SM、S、T、C、L、逻辑流
Q(FBD)	BOOL	I、Q、V、M、SM、S、T、C、L、逻辑流

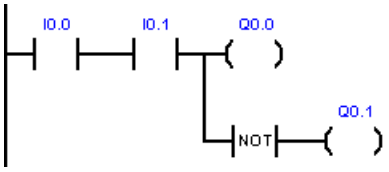
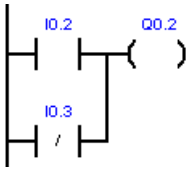
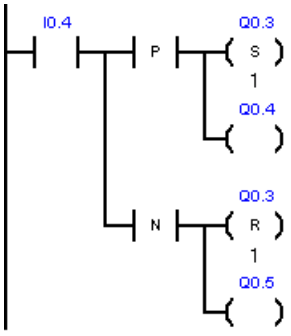
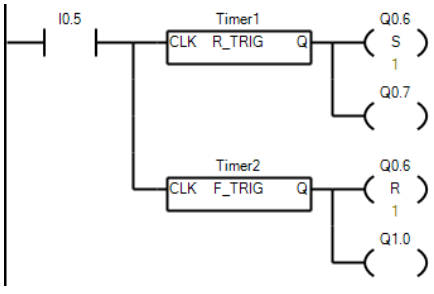
说明

因为正跳变和负跳变指令需要断开到接通或接通到断开转换，所以无法在首次扫描时检测上升沿或下降沿跳变。 首次扫描期间，CPU 会将初始输入状态保存在存储器位中。 在后续扫描中，这些指令会将当前状态与存储器位的状态进行比较以检测是否发生转换。

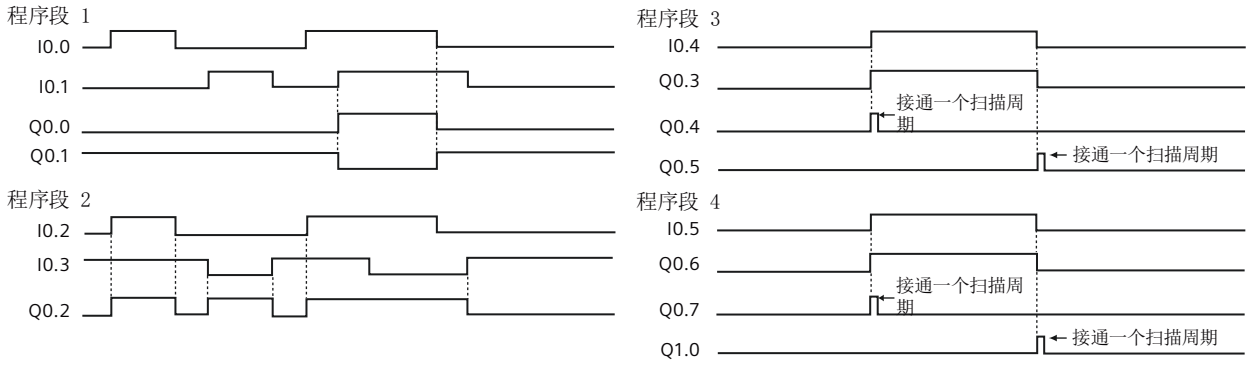
另请参见

位逻辑输入示例 (页 276)

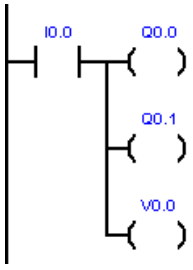
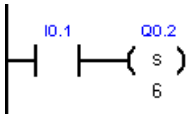
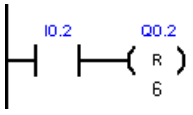
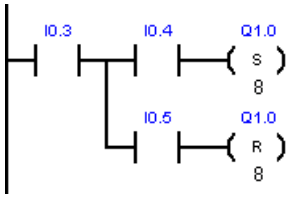
9.1.13 位逻辑输入示例

LAD		STL
	常开触点 I0.0 和 I0.1 必须接通（闭合）才能激活 Q0.0。NOT 指令用作取反器。 在 RUN 模式下，Q0.0 和 Q0.1 的逻辑状态相反。	Network 1 LD I0.0 A I0.1 = Q0.0 NOT = Q0.1
	常开触点 I0.2 必须接通或常闭触点 I0.3 必须断开，才能激活 Q0.2。一个或多个并联 LAD 分支（或逻辑）为真，才能激活输出。	Network 2 LD I0.2 ON I0.3 = Q0.2
	P 触点上出现上升沿输入或 N 触点上出现下降沿输入时会输出一个持续 1 个扫描周期的脉冲。 在 RUN 模式下，Q0.4 和 Q0.5 的脉动状态变化过快，无法在程序状态视图中监视。 置位和复位输出将脉冲状态锁存至 Q0.3，这使得此状态变化在程序状态视图中可见。	Network 3 LD I0.4 LPS EU S Q0.3, 1 = Q0.4 LPP ED R Q0.3, 1 = Q0.5
	R_TRIG 的 CLK 出现上升沿时会输出一个持续 1 个扫描周期的脉冲，F_TRIG 的 CLK 出现下降沿时会输出一个持续 1 个扫描周期的脉冲。 在 RUN 模式下，Q0.7 和 Q1.0 的脉动状态变化过快，无法在程序状态视图中监视。 置位和复位输出将脉冲状态锁存至 Q0.6，这使得此状态变化在程序状态视图中可见。	LD I0.5 LPS TRIGR Timer1 S Q0.6, 1 = Q0.7 LPP TRIGF Timer2 R Q0.6, 1 = Q1.0

运行模式下的输入定时示例

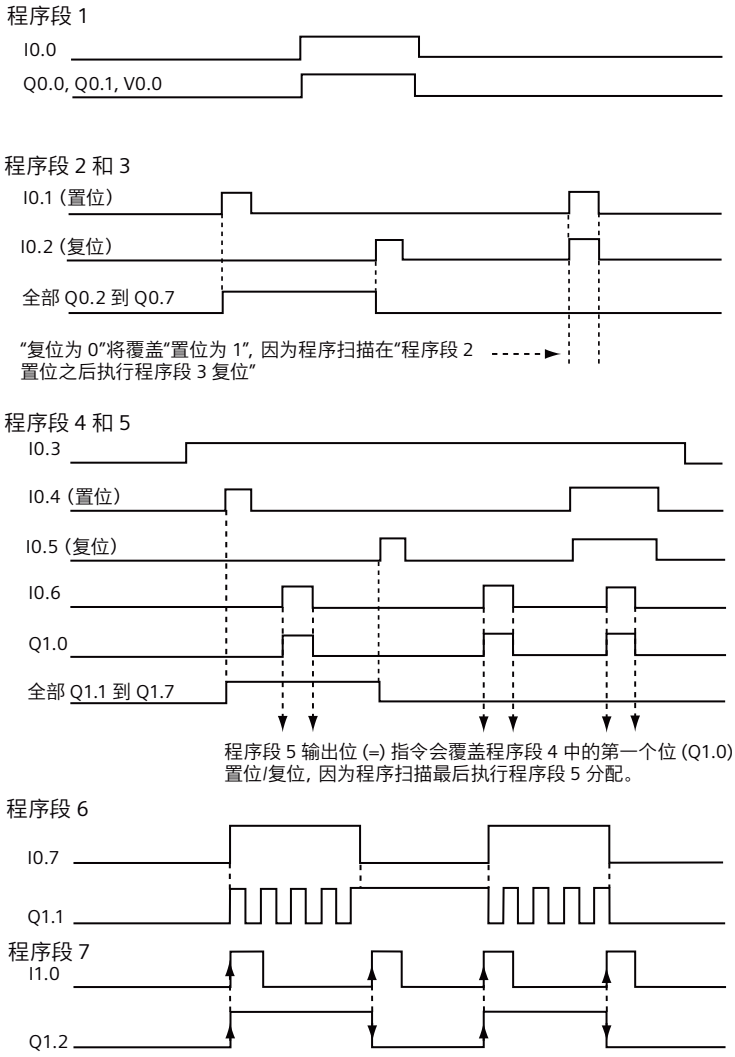


9.1.14 位逻辑输出示例

LAD		STL
	输出指令将位值分配给外部 I/O（I、Q）和内部存储器（M、SM、T、C、V、S、L）。	程序段 1 LD I0.0 = Q0.0 = Q0.1 = V0.0
	将一组连续的 6 个位设置为值 1。指定起始位地址和要置位的位数。第一个位 (Q0.2) 的值为 1 时，“置位”指令的程序状态指示器为 ON。	程序段 2 LD I0.1 S Q0.2, 6
	将一组连续的 6 个位复位为值 0。指定起始位地址和要复位的位数。第一个位 (Q0.2) 的值为 0 时，“复位”指令的程序状态指示器为 ON。	程序段 3 LD I0.2 R Q0.2, 6
	成组置位和复位 8 个输出位（Q1.0 至 Q1.7）。	程序段 4 LD I0.3 LPS A I0.4 S Q1.0, 8 LPP A I0.5 R Q1.0, 8

I0.6 Q1.0	置位和复位指令执行锁存继电器的功能。要隔离置位/复位位，请确保它们不会被其它赋值指令改写。在该示例中，程序段 4 成组置位和复位八个输出位（Q1.0 至 Q1.7）。在 RUN 模式下，程序段 5 会改写 Q1.0 位值并控制程序段 4 中的置位/复位程序状态指示器。	程序段 5 LD I0.6 = Q1.0
I0.7 Q1.1 ALT	脉冲交替输出指令 (ALT) 在每个扫描周期取反输出位的状态。	程序段 6 LD I0.7 =ALT Q1.1
I1.0 Q1.2 ALTP	脉冲交替输出指令 (ALTP) 在每个上升沿取反输出位的状态。	程序段 7 LD I1.0 =ALTP Q1.2

运行模式下的输出定时示例



9.2 时钟

9.2.1 读取和设置实时时钟

LAD/FBD	STL	说明
READ_RTC EN ENO T	**TODR T**	读取实时时钟指令从 CPU 读取当前时间和日期，并将其装载到从字节地址 T 开始的 8 字节时间缓冲区中。
SET_RTC EN ENO T	**TODW T**	设置实时时钟指令通过由 T 分配的 8 字节时间缓冲区数据将新的时间和日期写入到 CPU。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • 0007H T 数据错误	无

输入	数据类型	操作数
T	BYTE	IB、QB、VB、MB、SMB、SB、LB、*VD、*LD、*AC

说明

READ_RTC、SET_RTC 编程提示

这些指令不接受无效日期。例如，如果输入 2 月 30 日，则会发生非致命性日时钟错误 (0007H)。

不要在主程序和中断例程中使用 READ_RTC/SET_RTC 指令。执行另一个 READ_RTC/SET_RTC 指令时，无法执行中断例程中的 READ_RTC/SET_RTC 指令。在这种情况下，CPU 会置位系统标志位 SM4.3，指示尝试同时对日时钟执行二重访问，导致 T 数据错误（非致命错误 0007H）。

CPU 中的日时钟仅使用年份的最后两位数，因此 00 表示为 2000 年。使用年份值的用户程序必须考虑两位数的表示法。

2099 年之前的闰年年份，CPU 都能够正确处理。

8 字节时间缓冲区的格式，从字节地址 T 开始

所有日期和时间值必须采用 BCD 格式分配（例如，16#12 代表 2012 年）。00 至 99 的 BCD 值范围可分配范围为 2000 至 2099 的年份。

T 字节	说明	数据值
0	年	00 至 99（BCD 值）20xx 年：其中，xx 是 T 字节 0 中的两位数 BCD 值
1	月	01 至 12（BCD 值）
2	日	01 至 31（BCD 值）
3	小时	00 至 23（BCD 值）
4	分	00 至 59（BCD 值）
5	秒	00 至 59（BCD 值）
6	保留	始终设置为 00
7	星期几	使用 SET_RTC/TODW 指令写入时会忽略值。 通过 READ_RTC/TODR 指令进行读取时，值会根据当前年/月/日值报告正确的星期几。 1 至 7，1 = 星期日，7 = 星期六（BCD 值）

超出断电时长对 CPU 时钟的影响

有关掉电期间实时时钟可维持正确时间的时长，请参见《S7-200 SMART 系统手册》的附录 A“CPU 规范”。

超出断电时长后，CPU 将初始化为下表所示的时间值。

日期	时间	星期几
2000 年 1 月 1 日	00:00:00	星期六

说明

紧凑型串行 (CRs) CPU 型号没有 RTC（实时时钟）

可使用 READ_RTC 和 SET_RTC 指令设置紧凑型串行 (CRs) CPU 型号中的年份、日期和时间值，但这些值将在下一次 CPU 断电通电循环时丢失。上电时，日期和时间将初始化为 2000 年 1 月 1 日。

9.2.2 读取和设置扩展实时时钟

LAD/FBD	STL	说明
READ_RTCX EN ENO T	**TODRX T**	读取扩展实时时钟指令从 PLC 中读取当前时间、日期和夏令时组态，并将其装载到从 T 所分配地址开始的 19 字节缓冲区中。
SET_RTCX EN ENO T	**TODWX T**	设置实时时钟指令使用字节地址 T 分配的 19 字节时间缓冲区数据将新的时间、日期和夏令时组态写入到 PLC 中。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • 0007H T 数据错误 • 0091H 操作数超出范围	无

输入	数据类型	操作数
T	BYTE	IB、QB、VB、MB、SMB、SB、LB、*VD、*LD、*AC

说明

READ_RTCX、SET_RTCX 编程提示

这些指令不接受无效日期。例如，如果输入 2 月 30 日，则会发生非致命性日时钟错误 (0007H)。

不要在主程序和中断例程中使用 READ_RTCX/SET_RTCX 指令。执行另一个 READ_RTCX/SET_RTCX 指令时，无法执行中断例程中的 READ_RTCX/SET_RTCX 指令。在这种情况下，CPU 会置位系统标志位 SM4.3，指示尝试同时对日时钟执行二重访问，导致 T 数据错误（非致命错误 0007H）。

CPU 中的日时钟仅使用年份的最后两位数，因此 00 表示为 2000 年。使用年份值的用户程序必须考虑两位数的表示法。

2099 年之前的闰年年份，CPU 都能够正确处理。

19 字节时间缓冲区的格式，从字节地址 T 开始

说明

仅当在字节 8 中分配时间修正模式时，才使用 T 字节（9 至 18）或（9 至 20）。否则，将返回由 STEP 7-Micro/WIN SMART 或 SET_RTCX 指令最后写入到字节（9 至 18）或（9 至 20）中的值。

所有日期和时间值必须采用 BCD 格式分配（例如，16#12 代表 2012 年）。00 至 99 的 BCD 值范围可分配范围为 2000 至 2099 的年份。

T 字节	说明	数据值
0	年	00 至 99（BCD 值）20xx 年：其中，xx 是 T 字节 0 中的两位数 BCD 值
1	月	01 至 12（BCD 值）
2	日	01 至 31（BCD 值）
3	小时	00 至 23（BCD 值）
4	分	00 至 59（BCD 值）
5	秒	00 至 59（BCD 值）
6	保留	始终设置为 00
7	星期几	使用 SET_RTCX/TODWX 指令写入时会忽略值。 通过 READ_RTCX/TODRX 指令进行读取时，值会根据当前年/月/日值报告正确的星期几。 1 至 7，1 = 星期日，7 = 星期六（BCD 值）

T 字节	说明	数据值
8	修正模式： 针对夏令时 (DST)	00H = 禁用修正 01H = 欧盟（相对于 UTC 的时区偏移量 = 0 小时）[1] 02H = 欧盟（相对于 UTC 的时区偏移量 = +1 小时）[1] 03H = 欧盟（相对于 UTC 的时区偏移量 = +2 小时）[1] 04H - 07H = 保留 08H = 欧盟（相对于 UTC 的时区偏移量 = -1 小时）[1] 09H - 0FH = 保留 10H = 美国 [2] 11H = 澳大利亚 [3] 12H = 保留 13H = 新西兰 [4] 14H-EDH = 保留 EEH = 用户定义（星期几）（使用字节 9 - 20 中的值） EFH - FDH 保留 FEH = 保留 FFH = 用户定义（月中的某一天）（使用字节 9 - 18 中的值）
以下字节 9-18 仅用于修正模式 = FFH（由以前的用户分配）		
9	DST 修正小时数	0 至 23（BCD 值）
10	DST 修正分钟数	0 至 59（BCD 值）
11	DST 开始月份	1 至 12（BCD 值）
12	DST 开始日	1 至 31（BCD 值）
13	DST 开始小时	0 至 23（BCD 值）
14	DST 开始分钟	0 至 59（BCD 值）
15	DST 结束月份	1 至 12（BCD 值）
16	DST 结束日	1 至 31（BCD 值）
17	DST 结束小时	0 至 23（BCD 值）
18	DST 结束分钟	0 至 59（BCD 值）
以下字节 9-20 仅用于修正模式 = EEH（由扩展用户分配）		
9	DST 修正小时数	0 至 23（BCD 值）
10	DST 修正分钟数	0 至 59（BCD 值）
11	DST 开始月份	1 至 12（BCD 值）
12	DST 开始星期	1 至 5（BCD 值）[5]

T 字节	说明	数据值
13	DST 开始工作日	1 至 7（BCD 值）
14	DST 开始小时	0 至 23（BCD 值）
15	DST 开始分钟	0 至 59（BCD 值）
16	DST 结束月份	1 至 12（BCD 值）
17	DST 结束星期	1 至 5（BCD 值）[5]
18	DST 结束工作日	1 至 7（BCD 值）
19	DST 结束小时	0 至 23（BCD 值）
20	DST 结束分钟	0 至 59（BCD 值）

[1] 欧盟惯例：在三月最后一个星期日的 UTC 时间凌晨一点将时间向前调一小时。在十月最后一个星期日的 UTC 时间凌晨两点将时间往回调一小时。（进行修正时的当地时间取决于相对于 UTC 的时区偏移量。）

[2] 美国惯例：2007 年标准 - 在当地时间三月第二个星期日的凌晨两点将时间向前调一小时。在十一月第一个星期日的当地时间凌晨两点将时间向后调一小时。

[3] 澳大利亚惯例：2007 年标准 - 在十月第一个星期日的当地时间凌晨两点将时间向前调一小时。在四月第一个星期日的当地时间凌晨两点将时间向后调一小时（还适用于澳大利亚 - 塔斯马尼亚）。

[4] 新西兰惯例：2007 年标准 - 在九月最后一个星期日的当地时间凌晨两点将时间向前调一小时。在四月第一个星期日的当地时间凌晨两点将时间向后调一小时。

[5] 要分配某月最后出现的工作日（例如四月的最后一个星期一），设置星期 = 5。

超出断电时长对 CPU 时钟的影响

有关掉电期间实时时钟可维持正确时间的时长，请参见《S7-200 SMART 系统手册》的附录 A“CPU 规范”。

超出断电时长后，CPU 将初始化为下表所示的时间值。

日期	时间	星期几
2000 年 1 月 1 日	00:00:00	星期六

说明

紧凑型串行 (CRs) CPU 型号没有 RTC（实时时钟）

可使用 READ_RTCX 和 SET_RTCX 指令设置紧凑型串行 (CRs) CPU 型号中的年份、日期和时间值，但这些值将在下一次 CPU 断电通电循环时丢失。上电时，日期和时间将初始化为 2000 年 1 月 1 日。

9.3 通信

9.3.1 GET 和 PUT（以太网）

可以使用 GET 和 PUT 指令通过以太网连接在 S7-200 SMART CPU 之间进行通信。

说明

要使用 PUT & GET，必须选择“启用 Put/Get 服务器”(Enable Put/Get Server)。

表格 9-1 GET 和 PUT 指令

LAD/FBD	STL	描述
GET EN ENO TABLE	**GET table**	GET 指令启动以太网端口上的通信操作，从远程设备获取数据（如说明表 (TABLE) 中的定义）。 GET 指令可从远程站读取最多 222 个字节的信息。
PUT EN ENO TABLE	**PUT table**	PUT 指令启动以太网端口上的通信操作，将数据写入远程设备（如说明表 (TABLE) 中的定义）。 PUT 指令可向远程站写入最多 212 个字节的信息。

程序中可以有任意数量的 GET 和 PUT 指令，但在同一时间最多只能激活共 16 个 GET 和 PUT 指令。例如，在给定的 CPU 中可以同时激活八个 GET 和八个 PUT 指令，或六个 GET 和十个 PUT 指令。

当执行 GET 或 PUT 指令时，CPU 与 GET 或 PUT 表中的远程 IP 地址建立以太网连接。该 CPU 可同时保持最多八个连接。连接建立后，该连接将一直保持到在 CPU 进入 STOP 模式为止。

针对所有与同一 IP 地址直接相连的 GET/PUT 指令，CPU 采用单一连接。例如，远程 IP 地址为 192.168.2.10，如果同时启用三个 GET 指令，则会在一个 IP 地址为 192.168.2.10 的以太网连接上按顺序执行这些 GET 指令。

如果您尝试创建第九个连接（第九个 IP 地址），CPU 将在所有连接中搜索，查找处于未激活状态时间最长的一个连接。CPU 将断开该连接，然后再与新的 IP 地址创建连接。

GET 和 PUT 指令处于处理中/激活/繁忙状态或仅保持与其他设备的连接时，会需要额外的后台通信时间（参见“组态通信”）。所需的后台通信时间量取决于处于激活/繁忙状态的 GET 和 PUT 指令数量、GET 和 PUT 指令的执行频率以及当前打开的连接数量。如果通信性能不佳，则应当将后台通信时间调整为更高的值。

表格 9-2　GET 和 PUT 指令的有效操作数

输入/输出	数据类型	操作数
TABLE	BYTE	IB、QB、VB、MB、SMB、SB、*VD、*LD、*AC

设置 ENO = 0 的错误条件：

- 0006（间接地址）
- 函数返回错误，并置位表状态字节的错误位（请参见下图）

下图显示了 TABLE 参数引用的表，下表列出了错误代码。

表格 9-3　GET 和 PUT 指令 TABLE 参数的定义

字节偏移	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
0	D [1]	A [2]	E [3]	0	错误代码			
1	远程							
2	站							
3	IP							
4	地址 [4]							
5	保留 = 0（必须设置为零）							
6	保留 = 0（必须设置为零）							
7	指针，指向数据							
8	区域，位于							
9	远程站							
10	（I、Q、M、V 或 DB1）[5]							
11	数据长度 [6]							

字节偏移	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
12	指针，指向数据区域，位于本地站（此 CPU）（I、Q、M、V 或 DB1）[7]							
13								
14								
15								

[1] D - 完成（函数已完成）

[2] A - 激活（函数已排队）

[3] E - 错误（函数返回错误）

[4] 远程站 IP 地址：将要访问的数据所处 CPU 的地址。

[5] 指向远程站中数据区的指针：指向远程站中将要访问的数据的间接指针。

[6] 数据长度：远程站中将要访问的数据的字节数（PUT 为 1 至 212 字节，GET 为 1 至 222 字节）。

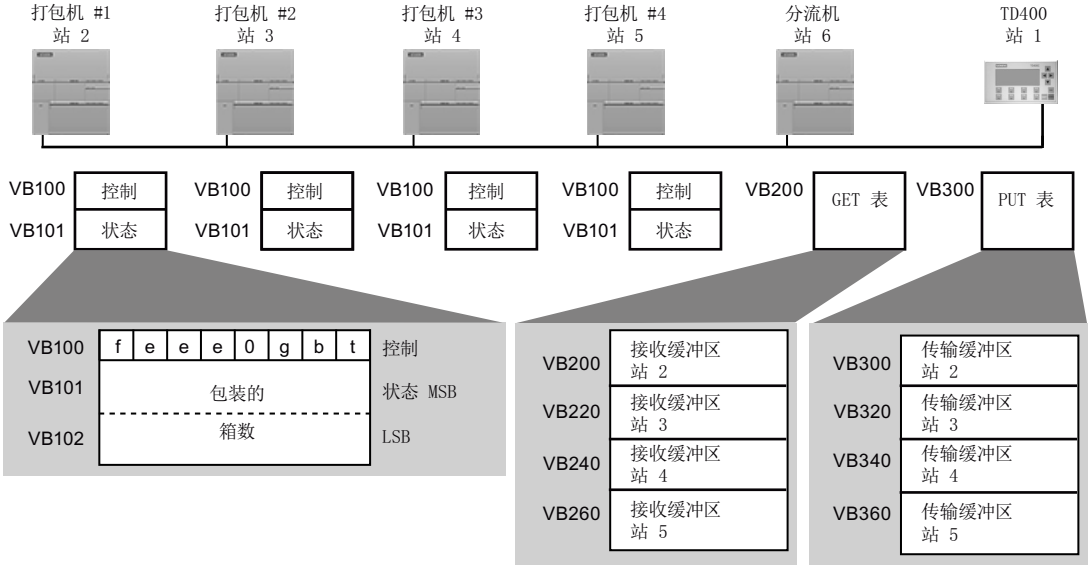
[7] 指向本地站中数据区的指针：指向本地站（此 CPU）中将要访问的数据的间接指针。

表格 9-4　GET 和 PUT 指令 TABLE 参数的错误代码：

代码	定义
0	无错误
1	PUT/GET 表中存在非法参数： • 本地区域不包括 I、Q、M 或 V • 本地区域的大小不足以提供请求的数据长度 • 对于 GET，数据长度为零或大于 222 字节；对于 PUT，数据长度大于 212 字节 • 远程区域不包括 I、Q、M 或 V • 远程 IP 地址是非法的 (0.0.0.0) • 远程 IP 地址为广播地址或组播地址 • 远程 IP 地址与本地 IP 地址相同 • 远程 IP 地址位于不同的子网
2	当前处于活动状态的 PUT/GET 指令过多（仅允许 16 个）
3	无可用连接。当前所有连接都在处理未完成的请求

代码	定义
4	从远程 CPU 返回的错误： • 请求或发送的数据过多 • STOP 模式下不允许对 Q 存储器执行写入操作 • 存储区处于写保护状态（请参见 SDB 组态）
5	与远程 CPU 之间无可用连接： • 远程 CPU 无可用的服务器连接 • 与远程 CPU 之间的连接丢失（CPU 断电、物理断开）
6 至 9、A 至 F	未使用（保留以供将来使用）

下图通过示例说明 GET 和 PUT 指令的功能。本例中，假设一条生产线正在灌装黄油桶，然后传送到四台装箱机（打包机）中的一台。打包机将 8 个黄油桶装入一个纸板箱中。分流机控制黄油桶流向各个打包机。4 个 CPU 控制打包机，具有 TD 400 操作员界面的 CPU 控制分流机。



t　黄油桶不足，无法包装；t=1，黄油桶不足

b　纸箱供应不足；b=1，必须在 30 分钟内增加纸箱

g　胶水供应不足；g=1，必须在 30 分钟内增加胶水

eee　标识遇到的故障类型的错误代码

f　故障指示器；f=1，装相机检测到错误

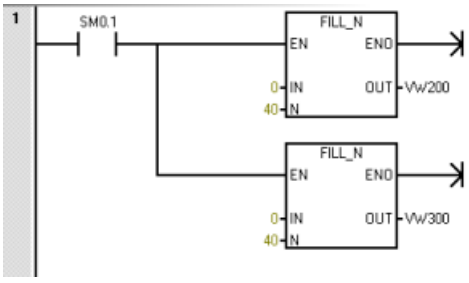
下图显示访问站 2 中数据所用的 GET 表格 (VB200) 和 PUT 表格 (VB300)。分流 CPU 使用 GET 指令连续读取来自每个装箱机的控制和状态信息。每当打包机装完 100 箱时，分流机都会注意到并通过 PUT 指令发送相应消息清除状态字。

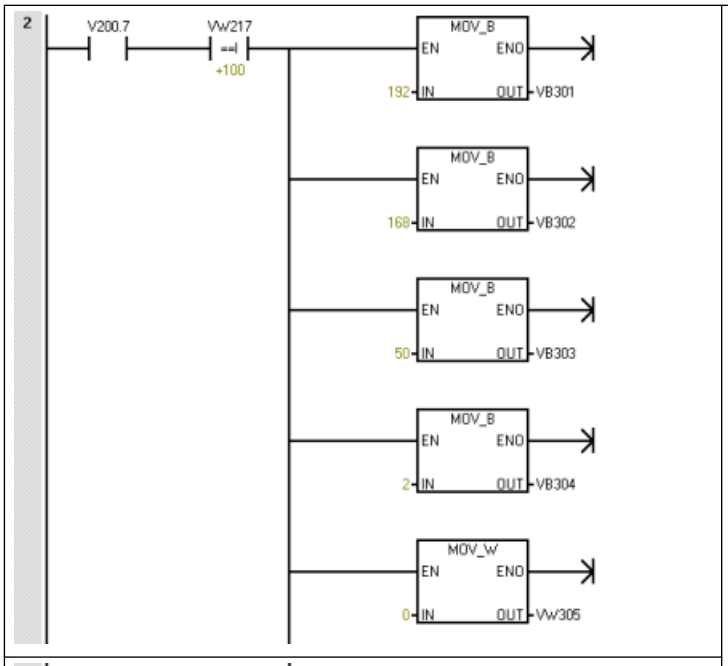
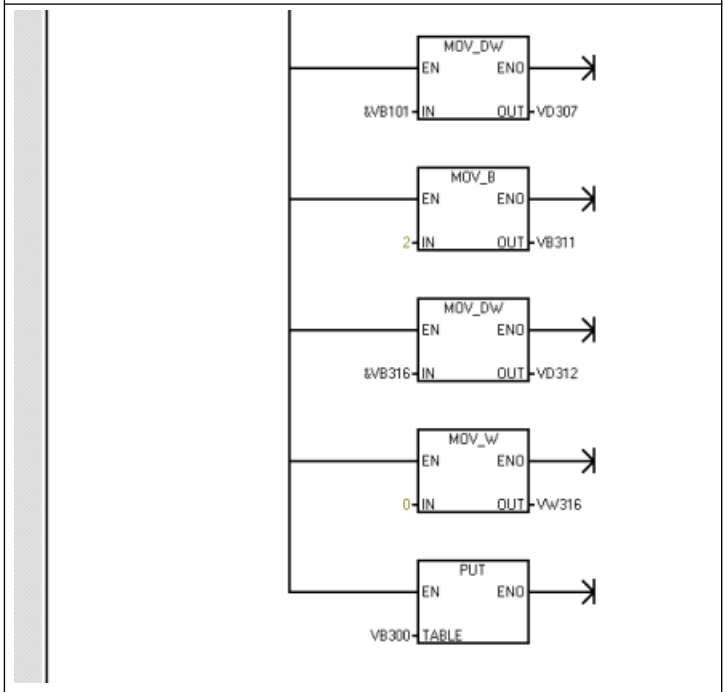
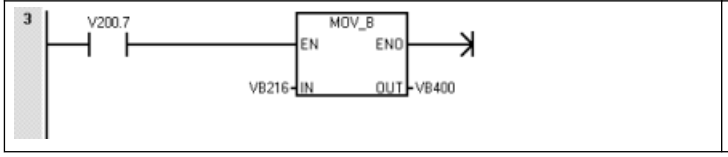
表格 9-5　用于读取和清除打包机 1 计数的 GET 和 PUT 指令缓冲区

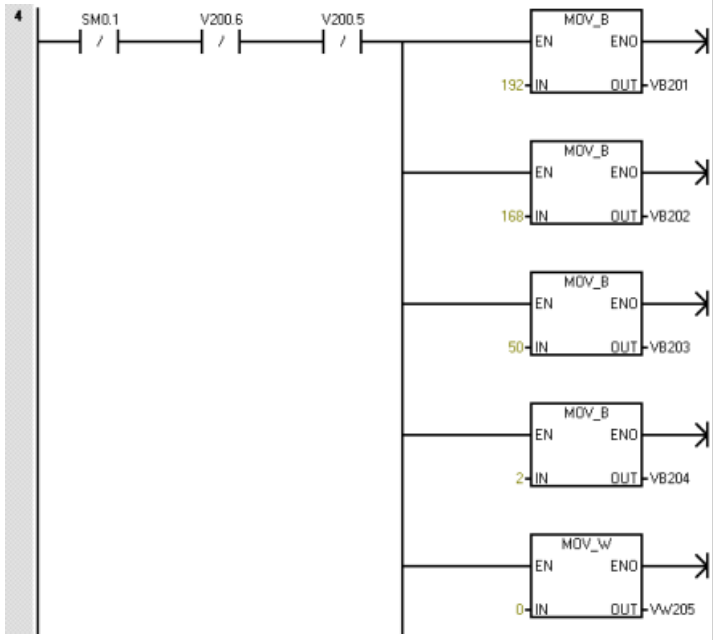
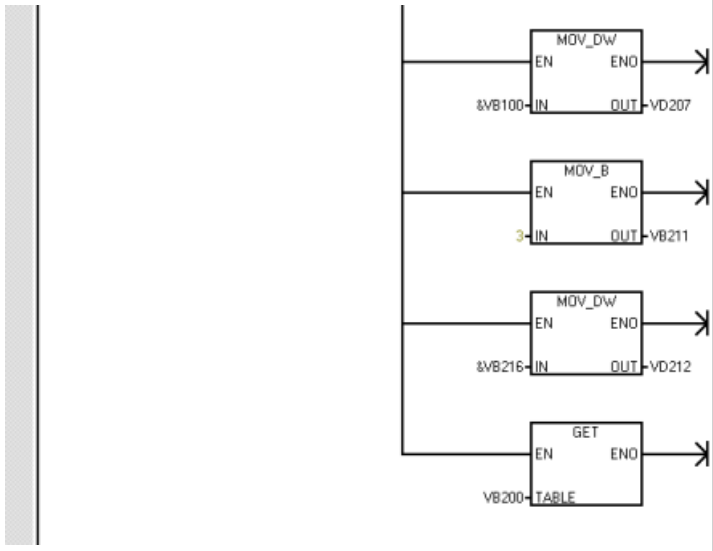
GET_TABLE 缓冲区	**位 7**	**位 6**	**位 5**	**位 4**	**位 3**	**位 2**	**位 1**	**位 0**	**PUT_TABLE 缓冲区**	**位 7**	**位 6**	**位 5**	**位 4**	**位 3**	**位 2**	**位 1**	**位 0**
VB200	D	A	E	0	错误代码				VB300	D	A	E	0	错误代码			
VB201	远程站 IP 地址 = 192.								VB301	远程站 IP 地址 = 192.							
VB202	168.								VB302	168.							
VB203	50.								VB303	50.							
VB204	2								VB304	2							
VB205	保留 = 0（必须设置为零）								VB305	保留 = 0（必须设置为零）							
VB206	保留 = 0（必须设置为零）								VB306	保留 = 0（必须设置为零）							
VB207	指针，指向数据								VB307	指针，指向数据							
VB208	区域，位于								VB308	区域，位于							
VB209	远程站 =								VB309	远程站 =							
VB210	(&VB100)								VB310	(&VB101)							
VB211	数据长度 = 3 个字节								VB311	数据长度 = 2 个字节							
VB212	指针，指向数据								VB312	指针，指向数据							
VB213	区域，位于								VB313	区域，位于							
VB214	本地站（此 CPU） =								VB314	本地站（此 CPU） =							
VB215	(&VB216)								VB315	(&VB316)							
VB216	控制								VB316	0							
VB217	状态 MSB								VB317	0							
VB218	状态 LSB																

在本示例中，数据紧随 PUT 和 GET 表的变化而变化。由于表中本地站的指针指向该数据，因此可将该数据置于 CPU 存储器中的任意位置（例如，VB212 - VB215）。

表格 9-6　　示例：GET 和 PUT 指令

	**Network 1** **LD SM0.1** **FILL +0, VW200, 40** **FILL +0, VW300, 40**	首次扫描时，清空所有接收和发送缓冲区。

 	**Network 2** **LD V200.7** **AW= VW217, +100** **MOVB 192, VB301** **MOVB 168, VB302** **MOVB 50, VB303** **MOVB 2, VB304** **MOVW 0, VW305** **MOVD &VB101, VD307** **MOVB 2, VB311** **MOVD &VB316, VD312** **MOVW 0, VW316** **PUT VB300**	当 GET 指令“完成”位 (V200.7) 置位，已包装完 100 箱时： 1. 装载打包机 1 的站地址。 2. 装载指向远程站中数据的指针。 3. 装载要发送的数据的长度。 4. 装载要发送的数据。 复位由打包机 1 包装的纸箱数
	**Network 3** **LD V200.7** **MOVB VB216, VB400**	当 GET“完成”位置位时，保存打包机 1 中的控制数据。

	**Network 4** **LDN SM0.1** **AN V200.6** **AN V200.5** **MOVB 192, VB201** **MOVB 168, VB202** **MOVB 50, VB203** **MOVB 2, VB204** **MOVW 0, VW205** **MOVD &VB100, VD207** **MOVB 3, VB211** **MOVD &VB216, VD212** **GET VB200**	如果不是第一次扫描且没有出错： 1. 装载打包机 1 的站地址。 2. 装载指向远程站中数据的指针。 3. 装载要接收的数据的长度。 4. 读取打包机 1 中的控制和状态数据。
		

9.3.2 发送和接收（RS485/RS232 为自由端口）

可使用发送 (XMT) 和接收 (RCV) 指令，通过 CPU 串行端口在 S7-200 SMART CPU 和其他设备之间进行通信。每个 S7-200 SMART CPU 都提供集成的 RS485 端口（端口 0）。标准 CPU 额外支持两个可选 CM01 信号板 (SB) RS232/RS485 端口（端口 1 和端口 2）。必须在用户程序中执行通信协议。

LAD/FBD	STL	说明
XMT EN ENO TBL PORT	**XMT TBL, PORT**	发送指令 (XMT) 用于在自由端口模式下通过通信端口发送数据。
RCV EN ENO TBL PORT	**RCV TBL, PORT**	接收指令 (RCV) 可启动或终止接收消息功能。必须为要操作的接收功能框指定开始和结束条件。通过指定端口 (PORT) 接收的消息存储在数据缓冲区 (TBL) 中。数据缓冲区中的第一个条目指定接收的字节数。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • 0009H（在端口 0 上同时发送/接收） • 000BH（在端口 1 上同时发送/接收） • 0x001F（在端口 2 上同时发送/接收） • 0090H 端口号无效 • 接收参数错误置位 SM86.6、SM186.6 或 SM405.6 • CPU 未处于自由端口模式	• SM 86.6 端口 0 终止接收消息 • SM 186.6 端口 1 终止接收消息 • SM 405.6 端口 2 终止接收消息

输入/输出	数据类型	操作数
TBL	BYTE	IB、QB、VB、MB、SMB、SB、*VD、*LD、*AC
PORT	BYTE	常数：0、1 或 2 注：三个可用端口如下： • 集成 RS485 端口（端口 0）， • CM01 信号板 (SB) RS232/RS485 端口（端口 1 和端口 2）

使用自由端口模式控制串行通信端口

可以选择自由端口模式以通过用户程序控制 CPU 的串行通信端口。选择自由端口模式后，程序通过使用接收中断、发送中断、发送指令和接收指令来控制通信端口的操作，并在自由端口模式下完全控制通信协议。使用 SMB30、SMB130 和 SMB31 来选择波特率和奇偶校验。

CPU 向两个物理端口分配三个特殊存储器字节：

- 向集成 RS485 端口（端口 0）分配 SMB30
- 向 CM01 RS232/RS485 信号板 (SB) 端口（端口 1）分配 SMB130
- 向 CM01 RS232/RS485 信号板 (SB) 端口（端口 2）分配 SMB31

CPU 处于 STOP 模式时，会禁用自由端口模式，并会重新建立正常通信（例如，HMI 设备访问）。

在最简单的情况下，可以只使用发送 (XMT) 指令向打印机或显示器发送消息。其他示例包括与条形码阅读器、秤和焊机的连接。在各种情况下，都必须编写程序，以支持在自由端口模式下与 CPU 进行通信的设备所使用的协议。

仅当 CPU 处于 RUN 模式时，才可使用自由端口通信。要启用自由端口模式，请在 SMB30（端口 0）、SMB130（端口 1）或 SMB31（端口 2）的协议选择字段中设置值 01。处于自由端口模式时，无法与同一端口上的 HMI 通信。

将 PPI 通信更改为自由端口模式

SMB30、SMB130 和 SMB31 分别组态通信端口 0、1 和 2 以进行自由端口操作，并提供波特率、奇偶校验和数据位数的选择。下图显示了自由端口控制字节。对于所有组态，都生成一个停止位。

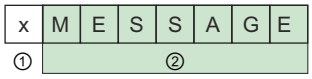
MSB 7							LSB 0
p	p	d	b	b	b	m	m

pp	奇偶校验选择		d	每个字符的数据位数	
	00 =	无奇偶校验		0 =	每个字符 8 位
	01 =	偶校验		1 =	每个字符 7 位
	10 =	无奇偶校验			
	11 =	奇校验			

bbb	自由端口波特率		mm	协议选择	
	000 =	38400		00 =	PPI 从站模式
	001 =	19200		01 =	自由端口模式
	010 =	9600		10 =	保留（默认为 PPI 从站模式）
	011 =	4800		11 =	保留（默认为 PPI 从站模式）
	100 =	2400			
	101 =	1200			
	110 =	115200			
	111 =	57600			

发送数据

发送指令用于对单字符或多字符（最多 255 个字符）缓冲区执行发送操作。下图显示了发送缓冲区的格式。



① 要发送的字节数

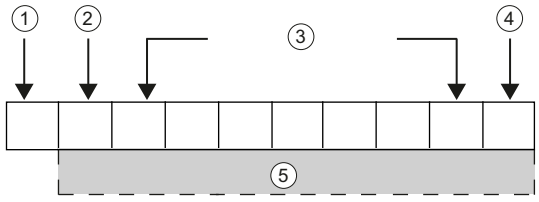
② 消息字符

如果中断例程连接到发送完成事件，CPU 将在发送完缓冲区的最后一个字符后生成中断（对于端口 0 为中断事件 9，对于端口 1 为中断事件 26，对于端口 2 为中断事件 48）。

您可以不使用中断，而通过监视 SM4.5（端口 0）、SM4.6（端口 1）或 SM20.0（端口 2）用信号表示完成发送的时间来发送消息（例如，向打印机发送消息）。

将字符数设为零，然后执行发送指令，这样可产生 BREAK 状态。这样产生的 BREAK 状态，在线上会持续以当前波特率发送 16 位数据所需要的时间。发送 BREAK 的操作与发送任何其他消息的操作是相同的。BREAK 发送完成时，会生成发送中断，并且 SM4.5、SM4.6 或 SM20.0 会指示发送操作的当前状态。

接收数据



接收指令用于对单字符或多字符（最多 255 个字符）缓冲区执行接收操作。下图显示了接收缓冲区的格式。

① 接收到的字节数（字节字段）

② 起始字符

③ 消息

④ 结束字符

⑤ 消息字符

如果中断例程连接到接收消息完成事件，CPU 会在接收完缓冲区的最后一个字符后生成中断（对于端口 0 为中断事件 23，对于端口 1 为中断事件 24，对于端口 2 为中断事件 46）。

可以不使用中断，而通过监视 SMB86（端口 0）、SMB186（端口 1）或 SMB405（端口 2）来接收消息。如果接收指令未激活或已终止，该字节不为零。正在接收时，该字节为零。

如下表所示，接收指令允许您选择消息开始和结束条件，对于端口 0 使用 SMB86 到 SMB94，对于端口 1 使用 SMB186 到 SMB194，对于端口 2 使用 SMB405 到 SMB413。

说明

如果出现组帧错误、奇偶校验错误、超限错误或断开错误，则接收消息功能将自动终止。必须定义开始条件和结束条件（最大字符数），这样接收消息功能才能运行。

接收缓冲区（SMB86 至 SMB94、SMB186 至 SMB194 以及 SMB405 至 SMB413）

端口 0	端口 1	端口 2	说明
SMB86	SMB186	SMB405	接收消息状态字节 MSB 7 … LSB 0 n r e 0 0 t c p n：1 = 接收消息功能终止；用户发出禁用命令。 r：1 = 接收消息功能终止；输入参数错误或缺少开始或结束条件。 e：1 = 收到结束字符。 t：1 = 接收消息功能终止；定时器时间到。 c：1 = 接收消息功能终止；达到最大字符计数。 p：1 = 接收消息功能终止；奇偶校验错误。

端口 0	端口 1	端口 2	说明
SMB87	SMB187	SMB406	接收消息控制字节 MSB 7 … LSB 0 en | sc | ec | il | c/m | tmr | bk | 0 en： 0 = 禁用接收消息功能。 1 = 启用接收消息功能。 每次执行 RCV 指令时，都会检查启用/禁用接收消息位。 sc： 0 = 忽略 SMB88、SMB188 或 SMB407。 1 = 使用 SMB88 或 SMB188 的值检测消息的起始。 ec： 0 = 忽略 SMB89、SMB189 或 SMB408。 1 = 使用 SMB89 或 SMB189 的值检测消息的结束。 il： 0 = 忽略 SMB90、SMB190 或 SMB409。 1 = 使用 SMB90 或 SMB190 的值检测消息的起始。 c/m： 0 = 定时器为字符间定时器。 1 = 定时器为消息定时器。 tmr： 0 = 忽略 SMW92、SMW192 或 SMW411。 1 = 如果超出 SMW92 或 SMW192 中的时间段，则终止接收。 bk： 0 = 忽略断开条件。 1 = 使用断开条件作为消息检测的起始。
SMB88	SMB188	SMB407	消息字符开始。
SMB89	SMB189	SMB408	消息字符结束。
SMW90	SMW190	SMW409	空闲线时间段以毫秒为单位指定。空闲线时间过后接收到的第一个字符为新消息的开始。

端口 0	端口 1	端口 2	说明
SMW 92	SMW 192	SMW 411	字符间/消息定时器超时值以毫秒为单位指定。如果超出该时间段，接收消息功能将终止。
SMB9 4	SMB1 94	SMB4 13	要接收的最大字符数（1 至 255 字节）。即使没有使用字符计数消息终止，此范围也必须设置为预期的最大缓冲区大小。

接收指令的开始和结束条件

接收指令使用接收消息控制字节（SMB87、SMB187 或 SMB406）中的位来定义消息开始和结束条件。

说明

执行接收指令时，如果通信端口上有来自其他设备的通信，则接收消息功能可能会从该字符的中间开始接收字符，从而导致奇偶校验错误或组帧错误以及接收消息功能终止。如果未启用奇偶校验，收到的消息可能包含错误字符。将开始条件指定为特定起始字符或任何字符时，可能会发生这种情况，如下文中的第 2 项和第 6 项所述。

接收指令支持多种消息开始条件。指定与断开或空闲线检测相关的开始条件，并在将字符放入消息缓冲区之前强制接收消息功能将消息开始与字符开始同步，这样可避免出现从字符的中间开始消息的问题。

接收指令支持多种开始条件：

1. *空闲线检测：*空闲线条件定义为传输线路上的安静或空闲时间。当通信线的安静或空闲时间达到在 SMW90、SMW190 或 SMW409 中指定的毫秒数时，便会开始接收。执行程序中的接收指令时，接收消息功能将开始搜索空闲线条件。如果在空闲线时间过期之前接收到任何字符，接收消息功能会忽略这些字符，并会按照 SMW90、SMW190 或 SMW409 中指定的时间重新启动空闲线定时器。请参见下图。空闲线时间过期后，接收消息功能会将接收到的所有后续字符存入消息缓冲区。
   空闲线时间应始终大于以指定波特率传送一个字符（包括起始位、数据位、奇偶校验位和停止位）所需的时间。空闲线时间的典型值为以指定波特率传送三个字符所需要的时间。
   对于二进制协议、没有特定起始字符的协议或指定了消息之间最小时间间隔的协议，可以将空闲线检测用作开始条件。
   设置：il = 1，sc = 0，bk = 0，SMW90/SMW190/SMW409 = 空闲线超时（毫秒）

① 执行接收指令：启动空闲时间

② 重新启动空闲时间

③ 检测到空闲时间：启动接收消息功能

④ 第一个字符放入消息缓冲区中

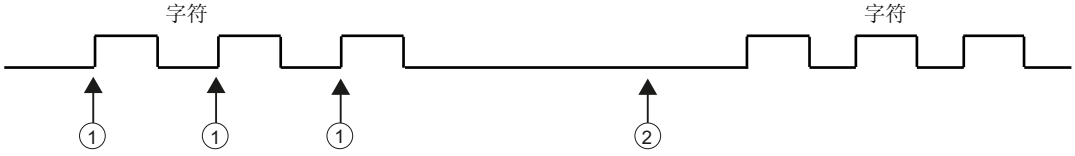
2. *起始字符检测：*起始字符是用作消息第一个字符的任意字符。当收到 SMB88、SMB188 或 SMB407 中指定的起始字符时，消息开始。接收消息功能会将起始字符作为消息的第一个字符存入接收缓冲区。接收消息功能忽略在起始字符之前收到的任何字符。起始字符以及在起始字符之后收到的所有字符都存储在消息缓冲区中。
   通常情况下，对于所有消息均以同一字符开始的 ASCII 协议，可以使用起始字符检测。
   设置：il = 0，sc = 1，bk = 0，SMW90/SMW190/SMW409 = 不相关，SMB88/SMB188/SMB407 = 起始字符

3. *空闲线和起始字符：*接收指令可启动组合了空闲线和起始字符的消息。执行接收指令时，接收消息功能会搜索空闲线条件。找到空闲线条件后，接收消息功能将查找指定的起始字符。如果接收到的字符不是起始字符，接收消息功能将开始重新搜索空闲线条件。所有在满足空闲线条件之前接收到以及在收到起始字符之前接收到的字符都将被忽略。起始字符与所有后续字符一起存入消息缓冲区。
   空闲线时间应始终大于以指定波特率传送一个字符（包括起始位、数据位、奇偶校验位和停止位）所需的时间。空闲线时间的典型值为以指定波特率传送三个字符所需要的时间。
   通常，对于指定消息之间最小时间间隔并且消息的首字符为指定特定设备的地址或其他信息的协议，可以使用这种类型的起始条件。这种方式尤其适用于通信链路上存在多台设备的情况。这种情况下，仅当接收到的消息的起始字符为特定地址或设备时，接收指令才会触发中断。
   设置：il = 1，sc = 1，bk = 0，SMW90/SMW190/SMW409 > 0，SMB88/SMB188/SMB407 = 起始字符

4. *断开检测：*当接收到的数据保持为零的时间大于完整字符传输的时间时，会指示断开。完整字符传输时间定义为传输起始位、数据位、奇偶校验位和停止位的时间总和。如果接收指令组态为接收到断开条件后启动消息，断开条件之后接收到的任意字符都会存储在消息缓冲区中。断开条件之前接收到的任何字符都会被忽略。
   通常，仅当协议需要时才将断开检测用作开始条件。
   设置：il = 0，sc = 0，bk = 1，SMW90/SMW190/SMW409 = 不相关，SMB88/SMB188/SMB407 = 不相关
5. *断开和起始字符：*接收指令可组态为在接收到断开条件后开始接收字符，然后按顺序接收特定起始字符。满足断开条件后，接收消息功能将查找指定的起始字符。如果接收到的字符不是起始字符，接收消息功能将重新搜索断开条件。所有在断开条件满足之前以及在接收到起始字符之前接收的字符都会被忽略。起始字符与所有后续字符一起存入消息缓冲区。
   设置：il = 0，sc = 1，bk = 1，SMW90/SMW190/SMW409 = 不相关，SMB88/SMB188/SMB407 = 起始字符
6. *任意字符：*接收指令可组态为立即开始接收任意字符和所有字符，并将其存入消息缓冲区。这是空闲线检测的一种特殊情况。在这种情况下，空闲线时间（SMW90、SMW190 或 SMW409）设为零。这样会强制接收指令一经执行便开始接收字符。
   设置：il = 1，sc = 0，bk = 0，SMW90/SMW190/SMW409 = 0，SMB88/SMB188/SMB407 = 不相关
   以任意字符开始一条消息允许使用消息定时器监视消息接收是否超时。如果使用自由端口实施协议的主站或主机部分，并且要在指定时间段内从站没有发出任何响应的情况下采用超时处理，这种方法非常有用。由于空闲线时间设为零，接收指令执行时，消息定时器将启动。如果未满足其他结束条件，则消息定时器超时，并会终止接收消息功能。
   设置：il = 1，sc = 0，bk = 0，SMW90/SMW190/SMW409 = 0，SMB88/SMB188/SMB407 = 不相关，c/m = 1，tmr = 1，SMW92/SMW192/SMW411 = 消息超时（毫秒）

接收指令支持多种终止消息的方式。终止消息的方式可以是以下一种方式，也可以是几种方式的组合：

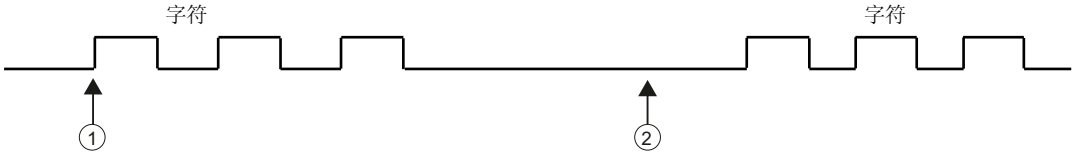
1. *结束字符检测：*结束字符是用于指示消息结束的任意字符。找到开始条件之后，接收指令将检查接收到的每一个字符，并判断其是否与结束字符匹配。接收到结束字符时，会将其存入消息缓冲区，接收终止。
   通常情况下，对于所有消息均以特定字符结束的 ASCII 协议，可以使用结束字符检测。可以将结束字符检测与字符间定时器、消息定时器或最大字符计数相结合，以终止消息。
   设置：ec = 1，SMB89/SMB189/SMB408 = 结束字符
2. *字符间定时器：*字符间时间是指从一个字符结束（停止位）到下一个字符结束（停止位）测得的时间。如果字符间的时间（包括第二个字符）超出 SMW92、SMW192 或 SMW411 中指定的毫秒数，则接收消息功能将终止。接收到每个字符后，字符间定时器重新启动。请参见下图。
   如果协议没有特定的消息结束字符，可以使用字符间定时器终止消息。由于定时器总是包含接收一个完整字符（起始位、数据位、奇偶校验位和停止位）的时间，定时器的值必须设为大于以选定波特率传输一个字符所需的时间。
   可以将字符间定时器与结束字符检测和最大字符计数结合使用，以终止消息。
   设置：c/m = 0，tmr = 1，SMW92/SMW192/SMW411 = 超时（毫秒）



① 重新启动字符间定时器

② 字符间定时器时间到：终止消息并生成接收消息中断

3. *消息定时器：*消息定时器在消息开始后的指定时间终止消息。消息定时器将在接收消息功能的开始条件得到满足后立即启动。经过 SMW92、SMW192 或 SMW411 中指定的毫秒数后，消息定时器时间到期。请参见下图。
   通常，当通信设备不能保证字符之间无时间间隔或使用调制解调器进行通信时，可以使用消息定时器。对于调制解调器，可以使用消息定时器指定一个从消息开始算起的允许接收消息的最大时间。消息定时器的典型值约为在选定波特率下接收最长消息所需时间值的 1.5 倍。
   可以将消息定时器与结束字符检测和最大字符计数相结合，以终止消息。
   设置：c/m = 1，tmr = 1，SMW92/SMW192/SMW411 = 超时（毫秒）



① 消息开始：启动消息定时器

② 消息定时器时间到：终止消息并生成接收消息中断

4. *最大字符计数：*接收指令必须获知要接收的最大字符数（SMB94、SMB194 或 SMB413）。达到或超出该值后，接收消息功能将终止。即使最大字符计数不被专门用作结束条件，接收指令仍要求用户指定最大字符计数。这是因为接收指令需要知道接收消息的最大长度，这样才能保证消息缓冲区之后的用户数据不被覆盖。
   对于消息长度已知并且恒定的协议，可以使用最大字符计数终止消息。最大字符计数总是与结束字符检测、字符间定时器或消息定时器结合在一起使用。
5. *奇偶校验错误：*当硬件发出信号指示奇偶校验错误、组帧错误或超限错误时，或在消息开始后检测到断开条件时，接收指令自动终止。仅当在 SMB30、SMB130 或 SMB31 中启用了奇偶校验后，才会出现奇偶校验错误。仅当停止位不正确时，才会出现组帧错误。仅当字符进入速度过快以致硬件无法处理时，才会出现超限错误。断开条件因与硬件的奇偶校验错误或组帧错误类似的错误而终止消息。无法禁用此功能。
6. *用户终止：*用户程序可以通过执行另一个 SMB87、SMB187 或 SMB406 中的使能位 (EN) 设置为零的接收指令终止接收消息功能。这样可以立即终止接收消息功能。

使用字符中断控制接收数据

为了完全适应对各种协议的支持，您还可以使用字符中断控制来接收数据。接收每个字符时都会产生中断。执行连接到接收字符事件的中断例程之前，接收到的字符存入 SMB2，奇偶校验状态（若已启用）存入 SM3.0。SMB2 是自由端口接收字符缓冲区。自由端口模式下接收到的每一个字符都会存入这一位置，便于用户程序访问。SMB3 用于自由端口模式，包含一个奇偶校验错误位，如果在接收到的字符中检测到奇偶校验错误、组帧错误、超限错误或断开错误，该位将置位。保留该字节的所有其他位。可使用奇偶校验位丢弃消息或向该消息发送否定确认。

以较高波特率（38.4 Kbps 到 115.2 Kbps）使用字符中断时，中断之间的时间间隔会非常短。例如，波特率为 38.4 Kbps 时的字符中断为 260 微秒，57.6 Kbps 时为 173 微秒，115.2 Kbps 时为 86 微秒。确保中断例程足够短，以避免字符丢失，否则请使用接收指令。

说明

SMB2 和 SMB3 可供端口 0、端口 1 和端口 2 共用。在端口 0 上接收字符导致执行连接到该事件（中断事件 8）的中断例程时，SMB2 包含在端口 0 上接收的字符，而 SMB3 则包含该字符的奇偶校验状态。在端口 1 上接收字符导致执行连接到该事件（中断事件 25）的中断例程时，SMB2 包含在端口 1 上接收的字符，而 SMB3 则包含该字符的奇偶校验状态。在端口 2 上接收字符导致执行连接到该事件（中断事件 47）的中断例程时，SMB0 包含在端口 1 上接收的字符，而 SMB3 则包含该字符的奇偶校验状态。

示例：发送和接收指令

MAIN	Network 1	Network 1 //本程序接收字符串，直至接收到换行字符。然后，消息会发送回发送方。	
		LD SM0.1 MOVB 16#09, SMB30	第一次扫描时： 1.初始化空闲端口： - 选择 9600 波特。 - 选择 8 位数据位。 - 选择无奇偶校验。
		MOVB 16#B0, SMB87	2.初始化 RCV 消息控制字节： - 启用 RCV。 - 检测消息结束字符。 -检测是否以线路空闲条件作为消息起始条件。
		MOVB 16#0A, SMB89	3.将消息结束字符设为十六进制 0A（换行）。
		MOVW +5, SMW90	4.将空闲线超时设为 5 ms。
		MOVB 100, SMB94	5.将最大字符数设为 100。
		ATCH INT_0, 23	6.将中断 0 连接到接收完成事件。
		ATCH INT_2, 9	7.将中断 2 连接到发送完成事件。
		ENI	8.启用用户中断。
		RCV VB100, 0	9.启用具有 VB100 缓冲区的接收功能框。

INT 0	Network 1	Network 1	
		LDB= SMB86, 16#20 MOVB 10, SMB34 ATCH INT_1, 10 CRETI NOT RCV VB100, 0	收到完成中断例程： 1.如果接收状态显示接收结束字符，则连接 10 ms 定时器，触发发送并返回。 2.如果因其他原因完成接收，则启动新的接收过程。
INT 1	Network 1	Network 1	
		LD SM0.0 DTCH 10 XMT VB100, 0	10 ms 定时器中断： 1.断开定时器中断。 2.将消息发送回端口上的用户。
INT 2	Network 1	Network 1	
		LD SM0.0 RCV VB100, 0	发送完成中断：启用另一接收。

9.3.3 获取端口地址和设置端口地址（RS485/RS232 上的 PPI 协议）

可使用 GET_ADDR 和 SET_ADDR 指令来读取和设置所选端口的 PPI 网络地址。

LAD/FBD	STL	描述
GET_ADDR EN ENO ADDR PORT	GPA ADDR, PORT	GET_ADDR 指令可读取 PORT 中指定的 CPU 端口的站地址，并将该值放入 ADDR 中指定的地址。
SET_ADDR EN ENO ADDR PORT	SPA ADDR, PORT	SET_ADDR 指令可将端口站地址 (PORT) 设为在 ADDR 中指定的值。新地址不会永久保存。循环上电后，受影响的端口将返回到通过系统块下载的网络地址。

ENO = 0 时的非致命错误条件	受影响的 SM 位
• 006H 间接地址 • 0004H 尝试在中断例程中执行 SET_ADDR 指令 • 0090H 端口号无效 • 0091H 端口地址无效	无

输入/输出	数据类型	操作数
ADDR	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数（常数值仅对“设置端口地址”指令有效。）
PORT	BYTE	常数：0、1 或 2 注：两个可用端口如下： • 集成 RS485 端口（端口 0）， • CM01 信号板 (SB) RS232/RS485 端口（端口 1） • CM01 信号板 (SB) RS232/RS485 端口（端口 2）：

9.3.4 获取 IP 地址和设置 IP 地址（以太网）

可使用 GIP_ADDR 和 SIP_ADDR 指令来读取和设置以太网端口的以太网 IP 地址、子网掩码和网关地址。

LAD/FBD	STL	描述
GIP_ADDR EN ENO ADDR MASK GATE	`GIP ADDR, MASK, GATE`	GIP_ADDR 指令将 CPU 的 IP 地址复制到 ADDR，将 CPU 的子网掩码复制到 MASK，并且将 CPU 的网关复制到 GATE。
SIP_ADDR EN ENO ADDR MASK GATE	`SIP ADDR, MASK, GATE`	SIP_ADDR 指令将 CPU 的 IP 地址设置为 ADDR 中找到的值，将 CPU 的子网掩码设置为 MASK 中找到的值，将 CPU 的网关设置为 GATE 中找到的值。

ENO = 0 时的非致命错误	受影响的 SM 位
• 006H 间接地址 • 0004H 尝试在中断例程中执行 SIP_ADDR 指令 • IP 地址无法更改（参见下方注释） • IP 地址对于当前子网无效	无

输入/输出	数据类型	操作数
ADDR	DWORD	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC、常数（仅限 SIP_ADDR）
MASK	DWORD	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC、常数（仅限 SIP_ADDR）
GATE	DWORD	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC、常数（仅限 SIP_ADDR）

说明

若要使用 SIP_ADDR 指令，请勿选中系统块通信部分中以太网端口的**“IP 地址数据固定为下面的值，不能通过其他方式更改”**(IP address data is fixed to the values below and cannot be changed by other means) 选项。

执行 SIP_ADDR 指令后，CPU 会将 IP 地址、子网掩码和网关值存储到永久性存储器中。

示例

注意 STEP 7-Micro/WIN SMART 会将 GIP_ADDR 指令、ADDR、MASK 和 GATE 的输出以字符串值的形式显示。但是，对于 SIP_ADDR 指令，用户需以十六进制值的形式提供 ADDR、MASK 和 GATE 输入。对于 SIP_ADDR 输入值，会将 IP 地址、MASK 和 GATE 的每个八位字节都视作十六进制数。

对于 SIP_ADDR 指令，IP 地址的八位字节表示为“192.168.2.150”：

八位十进制值	十六进制值
192	C0
168	A8

八位十进制值	十六进制值
2	02
150	96

用户应使用八位十六进制值的组合作为 SIP_ADDR 指令的 ADDR 输入：16#C0A80296。（可将这些数转化为十进制值，但十六进制值为八位字节值的表示形式。）

类似地，子网掩码的八位字节表示为“255.255.255.0”：

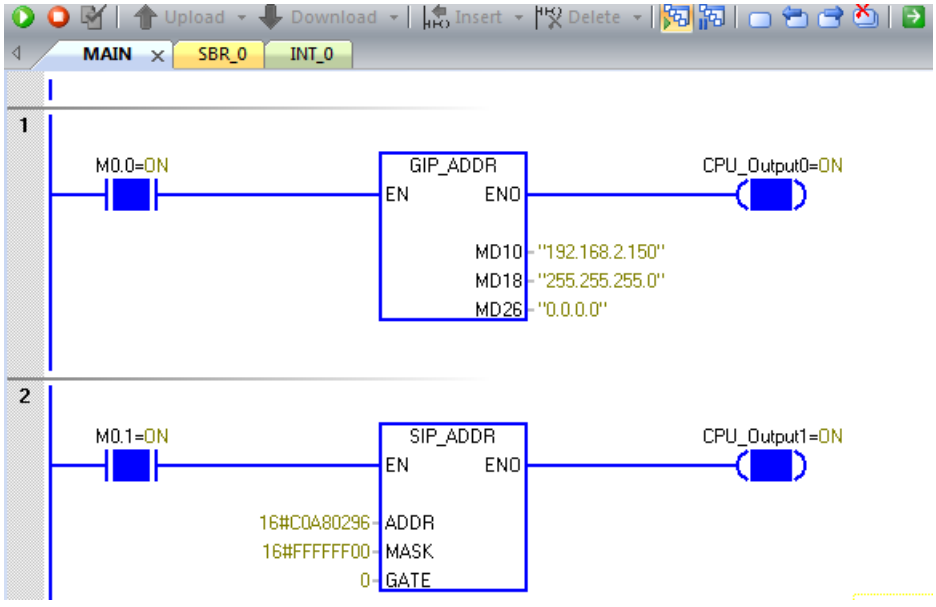
八位十进制值	十六进制值
255	FF
255	FF
255	FF
0	00

用户应使用八位十六进制值的组合作为 SIP_ADDR 指令的 MASK 输入：16#FFFFFF00。也可使用十进制等效值，但非字符串表示形式。

下面的程序状态显示屏显示了两个程序段：

- 程序段 1：GIP_ADDR 读取 IP 地址 192.168.2.150 和子网掩码 255.255.255.0。
- 程序段 2：SIP_ADDR 设置 IP 地址为 192.168.2.150 (16#C0A80296)，设置子网掩码为 255.255.255.0 (16#FFFFFF00)。

注意默认网关值为 0。



9.3.5 开放式用户通信

开放式用户通信 (OUC) 指令可使您的程序通过以太网与另一个支持以太网的设备进行通信。对方以太网设备可以是另一个 S7-200 SMART CPU 或是另一个支持 UDP、TCP、或 ISO-on-TCP 协议的第三方设备。您的程序对通信进行全方位的控制，包括选择协议、发起连接、发送数据、接收数据和终止连接。

9.3.5.1 OUC 指令

控制通信过程的开放式用户通信 (OUC) 指令有四条：

- TCON 打开 S7-200 SMART CPU 和远程设备之间的 UDP、TCP、或 ISO-on-TCP (RFC 1006) 连接。
- TSEND 和 TRCV 发送和接收数据。
- TDCON 关闭连接。

表格 9-7 OUC 指令

LAD/FBD	STL	描述
TCON EN ENO TABLE	**TCON table**	TCON 用于发起从 CPU 到通信伙伴的 UDP、TCP 或 ISO-on-TCP 通信连接。
TSEND EN ENO TABLE	**TSEND table**	TSEND 用于将数据发送到另一个设备。
TRECV EN ENO TABLE	**TRECV table**	TRECV 用于检索通过现有通信连接接收到的数据。
TDCON EN ENO TABLE	**TDCON table**	TDCON 用于终止 UDP、TCP 或 ISO-on-TCP 的通信连接。

OUC 指令能够保持有关连接的信息，这样您的程序就不需要为 OUC 表永久分配 V 存储空间。OUC 指令激活时，表中的数据必须保持不变。

OUC 指令处于处理中/激活/繁忙状态或仅保持与其它设备的连接时，会需要额外的后台通信时间。所需的后台通信时间量取决于处于激活/繁忙状态的 OUC 指令数量、OUC 指令的执行频率以及当前打开的连接数量。如果通信性能不佳，则应当将后台通信时间调整为更高的值。更多信息，请参见“组态通信”。

所有 OUC 指令都使用一个表为指令存储参数。每条指令在表中的内容描述如下。

S7-200 SMART CPU 使用输入表参数确定 OUC 指令的实例。为使 S7-200 SMART CPU 确认特定指令（实例）与前一次扫描中的指令为同一条指令，运行期间表的参数需保持不变。

说明

方便起见，Siemens 还提供开放式用户通信 (OUC) 库指令。OUC 库指令基于库指令输入为您构建该表。库指令还会从该表中检索响应信息，并在库指令的输出中提供这一信息。有关详细信息，请参见“开放式用户通信库”。

表格 9-8　OUC 指令的有效操作数

输入/输出	数据类型	操作数
表	字节	IB、QB、VB、MB、SMB、SB、*VD、*LD、*AC

设置 ENO = 0 的错误条件：

- 0006（间接地址）
- 如果函数返回错误并置位表状态字节的 E 位（请参见下图）

TCON 指令

您可使用 TCON 指令设置和建立通信连接。一旦 CPU 建立连接，它会自动保持和监视该连接。TCON 指令只有 TCON 表地址这一个参数。TCON 表包含连接参数。基于所选的连接协议，TCON 表有两种格式。UDP 和 TCP 共用同一公共表格格式。ISO-on-TCP 使用特殊的 TCON 表格格式。如需了解更多信息，请参见下方的 TCON 指令表。

要发起一个连接，将表中的 REQ 位设为 TRUE。当 TCON 指令激活、连接正在初始化、“激活”(Active) 位为 TRUE 时，CPU 将忽略 REQ 位。CPU 建立连接后，TCON 指令将“完成”(Done) 位置位。如果连接参数出现问题，或 CPU 无法与远程设备建立连接，则将置位“错误”(Error) 位。如果“错误”(Error) 位置位，错误代码会指出连接失败的原因。

TCON 指令是异步指令，可能需要数次扫描才能完成执行。连接操作待决时，将置位“激活”(Active) 位。

TCON 指令可创建主动（客户端）连接或被动（服务器）连接。主动连接是由 CPU 发起与远程设备的连接。被动连接则是 CPU 等待远程设备连接 CPU。

您也可使用 TCON 指令来确定当前连接的状态。如果 TCON 指令的 REQ 位设为 FALSE，则程序调用该指令时 CPU 会报告连接状态：

- 如果 CPU 建立了连接且连接可用，则指令将“完成”(Done) 位（无错误）置位。
- 如果连接仍处于正在连接过程，则指令将“激活”(Active) 位置位。
- 如果无法建立连接，则指令将“完成”(Done) 位和“错误”(Error) 位置位。错误代码将给出连接失败的原因。

表中的 REQ 位为电平触发位。建议在 REQ 输入端放置一个上升沿触发器来发起连接，这样 CPU 只需要建立一次连接。

在连接过程中（调用 TCON 指令），程序给连接分配一个连接 ID。连接 ID 是用户选定并传给 TCON 指令的 16 位数。连接 ID 可以是任何 0 到 65534 之间的数。CPU 不允讲将连接 ID 设为 65535 (0xFFFF)。连接 ID 值是所有 OUC 指令的输入，用以识别给定操作所使用的连接。

您可以根据自己实际情况选择连接 ID 数值，使其更符合逻辑。例如，您可以使用部分 IP 地址作为连接 ID。您可以为与 IP 地址 192.168.2.10（连接 ID 10）之间的连接命名。

请注意，连接关闭后，S7-200 SMART 不会自动尝试重新连接到设备。连接断开后，您的程序必须执行另一个 TCON 指令来重新连接该设备。主动和被动连接皆如此。

TCON 指令表

下表列出了 TCON 指令的格式和定义。有关错误代码列表，请参见“OUC 指令错误代码”(页 320)。有关端口数限制及更多信息，请参见“端口和 TSAP”：

- 状态：表的第一个字节将操作状态返回给用户。作为输入时，OUC 指令忽略状态字节的值。状态字节在返回指令时有效。状态位的定义为：
  - D = 完成（完毕）
  - A = 激活（进行中，换言之，繁忙）
  - E = 错误（完成且有错误）
  - 错误代码

  如果出现错误，则“完成”(Done) 位和“错误”(Error) 位均置位。错误代码列于“OUC 指令错误代码”（页 320)。
- REQ：您可使用 REQ 位发起新的操作。REQ 位为电平触发值。如果需要，程序代码必须提供该单步操作（上升沿接触）。如果操作不繁忙，则当 REQ 值为 TRUE 时将发起一个新的操作。例如：如果当前没有在执行的 TSEND 指令，则 REQ 位为 TRUE 会使程序发起一个新的 TSEND 指令操作。
- 连接 ID：连接 ID 是您选定传递给函数的 16 位值。范围是 0 到 65534（65535 保留）。连接 ID 参数是 OUC 指令的输入。TSEND、TRECV 和 TDCON 指令将您为 TCON 指令选择的连接 ID 作为参考。

表格 9-9　UDP 和 TCP 的 TCON 指令表参数结构定义

字节偏移量	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
0	D	A	E	错误代码（5 位）				
1	A/P [1]							REQ
2	连接 ID（2 字节）							
3								
4	连接类型 [2]							
5	远程 IP 地址 [3]							
6								
7								
8								
9	远程端口 [4]							
10								

字节偏移量	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
11	本地 [5]							
12								

A/P [1] 主动/被动选择（1 = 主动，0 = 被动）

[2] 连接类型：连接类型通知 TCON 指令期望的连接类型：UDP = 19，TCP = 11

[3] 远程 IP 地址：该地址为主动连接中远程设备的 IP 地址。对于 UDP 连接，您应将远程 IP 地址设为 0.0.0.0。IP 地址不能与本地 CPU 的 IP 地址相同，且不能为组播或广播地址。由于 S7-200 Smart 支持路由功能，因此 IP 地址可以和本地 CPU 不在同一个子网中。
如果您为被动（服务器）连接设定了 IP 地址，则 CPU 只接受来自特定 IP 地址的连接。如果您将被动连接的 IP 地址设为 0.0.0.0，则 CPU 可接受来自任何 IP 地址的连接。

[4] 远程端口：远程设备中的端口号。UDP 或被动连接不使用远程端口号，并应将远程端口设为零。

[5] 本地端口：本地 CPU 中连接的端口号。

表格 9-10　ISO-on-TCP 的 TCON 指令表参数结构定义

字节偏移量	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
0	D	A	E	错误代码（5 位）				
1	A/P [1]							REQ
2	连接 ID（2 字节）							
3								
4	连接类型 [2]							
5	远程 IP 地址 [3]							
6								
7								
8								
9	远程 TSAP [4] 2 到 16 字符的字符串（3 到 17 个字节）							
到								
25								

字节偏移量	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
26	本地 TSAP [5]							
到	2 到 16 字符的字符串（3 到 17 个字节）							
42								

A/P [1] 主动/被动选择（1 = 主动，0 = 被动）

[2] 连接类型：连接类型通知 TCON 指令期望的连接类型：ISO-on-TCP = 12

[3] 远程 IP 地址：该地址为主动连接中远程设备的 IP 地址。IP 地址不能与本地 CPU 的 IP 地址相同，且不能为组播或广播地址。由于 S7-200 Smart 支持路由功能，因此 IP 地址可以和本地 CPU 不在同一个子网中。
如果您为被动（服务器）连接设定了 IP 地址，则 CPU 只接受来自特定 IP 地址的连接。如果您将被动连接的 IP 地址设为 0.0.0.0，则 CPU 可接受来自任何 IP 地址的连接。

[4] 远程 TSAP：远程设备的传送服务访问点 (TSAP)。只能将远程 TSAP 用于 ISO-on-TCP 连接。远程 TSAP 为一个 2 到 16 个 ASCII 字符的字符串。

[5] 本地 TSAP：本地 CPU 中连接的传送服务访问点 (TSAP)。只能将本地 TSAP 用于 ISO-on-TCP 连接。本地 TSAP 为一个 2 到 16 个 ASCII 字符的字符串。如果使用两个字符，则 TSAP 必须以一个十六进制字符“E0”开头 ($E0)，后跟另一个十六进制字符（例如，“$E0$01”）。不能使用“SIMATIC”字符串。

TSEND

您可通过现有的通信连接使用 TSEND 指令发送数据。TSEND 表包含连接参数。基于所选的连接协议，TSEND 表有两种格式。TCP 和 ISO-on-TCP 共用同一公共表格格式。UDP 使用特殊的 TSEND 表格格式。如需了解更多信息，请参见下方的 TSEND 和 TRECV 指令表。

如果 REQ 被置位且连接当前未被其它操作占用，则当您的程序调用 TSEND 指令时，TSEND 指令将开始发送特定数量的字节。

REQ 位为电平触发。建议在 REQ 输入端放置一个上升沿触发器来发起连接，这样 CPU 不会意外发起发送操作。当 TSEND 为“激活”(Active) 时，CPU 会忽略 REQ 位。状态位和错误代码会显示每次调用时 TSEND 的状态：

- 完成无错误意为 TSEND 指令完成，且没有错误。
- 激活意为 TSEND 指令仍为繁忙状态。
- 完成但有错误意为 TSEND 出现了问题。错误代码中包含故障原因。

发送操作完成后，会显示每个 TSEND 指令调用的完成/激活/错误状态。此后，TSEND 通过错误代码 24 作出响应，表示无待决操作，前提是您的程序调用指令时将 REQ 设为 FALSE。如果将 REQ 保持置位，则 TSEND 指令将发起另一个发送操作。

您可在一条消息内最多发送 1024 字节的数据。在一个给定的连接中，一次只能有一条 TSEND 处于激活状态。在 REQ 置位情况下执行 TSEND 指令时，程序将数据从用户存储器的发送缓冲区复制到内部缓冲区，这样您可在 TSEND 指令执行后修改发送缓冲区。

TRECV

对于由 CPU 通过现有通信连接接收的数据，您可使用 TRECV 指令进行检索。分配接收区/缓冲区以及接收区最大长度，从而避免出现缓冲区溢出。TRECV 表包含 TRECV 指令所需参数。基于所选的连接协议，TRECV 表有两种格式。TCP 和 ISO-on-TCP 共用同一公共表格格式。UDP 使用特殊的 TRECV 表格格式。如需了解更多信息，请参见下方的 TSEND 和 TRECV 表。

TRECV 指令无 REQ 位。首次执行 TRECV 指令后，状态位显示指令为“激活”(Active)。如果此次连接 CPU 未接收到数据，则所有后续调用 TRECV 指令均显示“激活”(Active) 状态。

成功接收数据后，指令将表中状态字节的“完成”(Done) 位置位，返回的数据长度值是实际接收到的字节数。只有当 TRECV 指令执行且“完成”(Done) 位设为 TRUE 时，TRECV 指令才会将接收到的数据从内部缓冲区复制到您的接收缓冲区。

在一条消息中最多可以接收 1024 字节的数据。由于 TCP 起“流”协议作用，如果未频繁调用 TRECV 指令，则程序可在一条接收消息中采集多个消息。UDP 和 ISO-on-TCP 协议可确保将每条消息单独划分出来。

例如：假设一个 TCP 客户端向 S7-200 SMART 快速、连续地发送四个 20 字节消息，但您的程序未调用 TRECV 指令。如果您的程序是在 CPU 接收所有四条消息后才调用 TRECV 指令的，则程序将认为接收了一条 80 字节的消息。每当一条消息发送时，您的程序负责调用一次 TRECV 指令接收该条消息。

假设客户端和消息与上例相同，ISO-on-TCP 和 UDP 在随后四次调用 TRECV 指令期间发送了四条消息。这些协议将消息进行划分并单独存放在 CPU 中，直到您的程序调用 TRECV 指令对其进行检索。

如果 CPU 接收的字节数超出用户缓冲区的容量，TRECV 指令将复制所允许的最多字节数（表中的数据长度），并放弃其它接收到的字节。在这种情况下，TRECV 指令执行完成后出现错误消息，提醒用户字节被丢弃。

TSEND 和 TRECV 指令表

下表列出了 TSEND 和 TRECV 指令的格式和定义。有关错误代码列表，请参见“OUC 指令错误代码” (页 320)。有关端口数限制及更多信息，请参见“端口和 TSAP”：

- 状态：表的第一个字节将操作状态返回给用户。作为输入时，OUC 指令忽略状态字节的值。状态字节在返回指令时有效。状态位的定义为：
  - D = 完成（完毕）
  - A = 激活（进行中，换言之，繁忙）
  - E = 错误（完成且有错误）
  - 错误代码

  如果出现错误，则“完成”(Done) 位和“错误”(Error) 位均置位。错误代码列于“OUC 指令错误代码” (页 320)。
- REQ：您可使用 REQ 位发起新的操作。REQ 位为电平触发值。如果需要，程序代码必须提供该单步操作（上升沿接触）。如果操作不繁忙，则当 REQ 值为 TRUE 时将发起一个新的操作。例如：如果当前没有在执行的 TSEND 指令，则 REQ 位为 TRUE 会使程序发起一个新的 TSEND 指令操作。
- 连接 ID：连接 ID 是您选定传递给函数的 16 位值。范围是 0 到 65534（65535 保留）。连接 ID 参数是 OUC 指令的输入。TSEND、TRECV 和 TDCON 指令将您为 TCON 指令选择的连接 ID 作为参考。

表格 9-11　TCP 和 ISO-on-TCP 的 TSEND 和 TRECV 指令表参数结构定义

字节偏移量	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
0	D	A	E	错误代码（5 位）				
1								REQ [1]
2	连接 ID（2 字节）							
3								
4	数据长度 [2]							
5								

字节偏移量	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
6	数据指针 [3]							
7								
8								
9								

[1] REQ：您可通过将 REQ 位设为 TRUE 来发起新的 TSEND 指令操作。TRECV 指令忽略 REQ 状态位。REQ 位仅用于 TSEND 指令。
对于 TRECV 指令，“完成”位意为 CPU 接收到数据（新数据准备好），Data_Length 值返回实际接收到的字节数。如果调用时没有可用数据，则 TRECV 指令返回，且“激活”(Active) 标志置位，Data_Length 值为零。如果接收到的字节数超出接收缓冲区的大小（数据长度输入），则程序将最大数目的字节复制到缓冲区，并向 TRECV 指令返回一个错误。

[2] 数据长度：TRECV 指令表中的数据长度既是输入参数也是输出参数。输入值为接收缓冲区的最大容量。输出值为实际接收到的字节数。
数据长度仅作为 TSEND 指令的输入值。

[3] 数据指针：指向本地 CPU 中数据的 S7-200 SMART 指针。

表格 9-12 UDP 的 TSEND 和 TRECV 指令表参数结构定义

字节偏移量	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
0	D	A	E	错误代码（5 位）				
1								REQ [1]
2	连接 ID（2 字节）							
3								
4	数据长度 [2]							
5								
6	数据指针 [3]							
7								
8								
9								

字节偏移量	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
10	远程							
11	IP							
12	地址 [4]							
13								
14	远程端口 [5]							
15								

[1] REQ：您可通过将 REQ 位设为 TRUE 来发起新的 TSEND 指令操作。TRECV 指令忽略 REQ 状态位。REQ 位仅用于 TSEND 指令。
对于 TRECV 指令，“完成”位意为 CPU 接收到数据（新数据准备好），Data_Length 值返回实际接收到的字节数。如果调用时没有可用数据，则 TRECV 指令返回，且“激活”(Active) 标志置位，Data_Length 值为零。如果接收到的字节数超出接收缓冲区的大小（数据长度输入），则程序将最大数目的字节复制到缓冲区，并向 TRECV 指令返回一个错误。

[2] 数据长度：TRECV 指令结构中的数据长度既是输入参数也是输出参数。输入值为接收缓冲区的最大容量。输出值为实际接收到的字节数。
数据长度仅作为 TSEND 指令的输入值。

[3] 指向数据区的数据指针：指向本地 CPU 中数据的 S7-200 SMART 指针。

[4] 远程 IP 地址：该地址为 TSEND 指令的远程设备的 IP 地址。IP 地址不能与本地 CPU 的 IP 地址相同，且不能为组播或广播地址。由于 S7-200 Smart 支持路由功能，因此 IP 地址可以与本地 CPU 不在同一个子网中。（必须为每个 UDP 发送操作提供 IP 地址）
IP 地址为 UDP 接收操作的返回值。IP 地址是 UDP 消息发送方的地址。

[5] 远程端口：为远程设备中的端口号。
远程端口为 UDP 接收操作的返回值。该端口为 UDP 信息发送方的端口号。
UDP 需要每个 TSEND 指令消息的远程端口号。

TDCON

您可使用 TDCON 指令来终止现有的通信连接。当 REQ 置位时，指令终止连接。建议在 REQ 输入端放置一个上升沿触发器。如果您的程序调用了 TDCON 指令，且连接已断开，则指令将通过错误代码 24 作出响应，意为无待决操作。

TDCON 指令表

下表列出了 TDCON 指令的格式和定义。有关错误代码列表，请参见“OUC 指令错误代码” (页 320)。有关端口数限制及更多信息，请参见“端口和 TSAP”：

- 状态：表的第一个字节将操作状态返回给用户。作为输入时，OUC 指令忽略状态字节的值。状态字节在返回指令时有效。状态位的定义为：
  - D = 完成（完毕）
  - A = 激活（进行中，换言之，繁忙）
  - E = 错误（完成且有错误）
  - 错误代码

  如果出现错误，则“完成”(Done) 位和“错误”(Error) 位均置位。错误代码列于“OUC 指令错误代码” (页 320)。
- REQ：您可使用 REQ 位发起新的操作。REQ 位为电平触发值。如果需要，程序代码必须提供该单步操作（上升沿接触）。如果操作不繁忙，则当 REQ 值为 TRUE 时将发起一个新的操作。例如：如果当前没有在执行的 TSEND 指令，则 REQ 位为 TRUE 会使程序发起一个新的 TSEND 指令操作。
- 连接 ID：连接 ID 是您选定传递给函数的 16 位值。范围是 0 到 65534（65535 保留）。连接 ID 参数是 OUC 指令的输入。TSEND、TRECV 和 TDCON 指令将您为 TCON 指令选择的连接 ID 作为参考。

表格 9-13　TDCON 指令表参数结构定义

字节偏移量	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
0	D	A	E	错误代码（5 位）				
1								REQ
2	连接 ID（2 字节）							
3								

9.3.5.2 OUC 指令错误代码

下表列出了开放式用户通信 (OUC) 错误代码：

错误代码	描述	TCON	TSEND	TRECV	TDCON
0	无错误	X	X	X	X
1	数据长度参数大于允许的最大长度（1024 字节）。		X	X	
2	数据缓冲区未处于 I、Q、M 或 V 存储区。		X	X	
3	数据缓冲区不适合存储区。		X	X	
4	表格参数不适合存储区。	X	X	X	X
5	连接在另一上下文中被锁定。您正在试图同时访问背景主程序 (Main) 和中断例程中的同一连接。	X	X	X	X
6	UDP IP 地址或端口错误		X		
7	实例不符：在另一实例中连接为忙，或是当发起请求时，为所请求的连接 ID 保存的数据与输入数据不符。	X	X	X	X
8	由于连接从未创建，所以连接 ID 不存在，或连接按您的要求终止（使用 TDCON 指令）。	X	X	X	X
9	使用此连接 ID 的 TCON 操作正在进行中。		X	X	X
10	使用此连接 ID 的 TDCON 操作正在进行中。	X	X	X	
11	使用此连接 ID 的 TSEND 指令正在进行中。		X		X
12	发生了临时通信错误。此时无法启动连接。请稍后重试。	X	X	X	
13	连接伙伴拒绝或主动断开连接（伙伴将断开与此 CPU 的连接）。	X	X	X	
14	无法连接连接伙伴（连接请求无应答）。	X	X	X	
15	连接因不一致而断开。断开并重新连接以纠正这一情况。	X	X	X	X
16	连接 ID 已与不同的 IP 地址、端口或 TSAP 组合配合使用。	X			
17	没有连接资源可用。所有请求类型（主动/被动）的连接都在使用中。	X			

错误代码	描述	TCON	TSEND	TRECV	TDCON
18	本地或远程端口号被保留，或端口号已用于另一服务器（被动）连接。	X			
19	已发生以下 IP 地址错误之一： • IP 地址无效（例如，地址 0.0.0.0）。 • 该 IP 地址是此 CPU 的 IP 地址。 • 该 CPU 地址为 0.0.0.0。 • IP 地址为广播地址或多播地址。	X			
20	本地或远程 TSAP 错误（仅 ISO-on-TCP）	X			
21	连接 ID 无效（65535 保留）	X			
22	主动/被动错误（UDP 只允许被动）	X			
23	连接类型不在所允许的类型中。	X			
24	没有待决操作，因此没有要报告的状态。		X		X
25	接收缓冲区过小：CPU 接收的字节数超出缓冲区支持的长度。CPU 丢弃额外的字节。			X	
31	未知错误	X	X	X	X

9.4 比较

9.4.1 比较数值

比较指令可以对两个数据类型相同的数值进行比较。您可以比较字节、整数、双整数和实数。

对于 **LAD 和 FBD**：比较结果为 TRUE 时，比较指令将接通触点（LAD 程序段能流）或输出（FBD 逻辑流）。

对于 **STL**：比较结果为 TRUE 时，比较指令可装载 1、将 1 与逻辑栈顶中的值进行“与”运算或者“或”运算。

比较类型

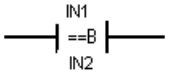
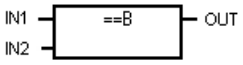
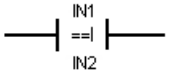
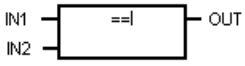
有六种比较类型可用：

比较类型	输出仅在以下条件下为 TRUE
== (LAD/FBD) = (STL)	IN1 等于 IN2
<>	IN1 不等于 IN2
>=	IN1 大于或等于 IN2
<=	IN1 小于或等于 IN2
>	IN1 大于 IN2
<	IN1 小于 IN2

选择要比较的数据类型

所选数据类型标识符决定 IN1 和 IN2 参数所需的数据类型。

数据类型标识符	所需 IN1、IN2 数据类型
B	无符号字节
W	有符号字整数
D	有符号双字整数
R	有符号实数

LAD 触点，FBD 功能框	STL	比较结果
 	**LDB= IN1, IN2** **OB= IN1, IN2** **AB= IN1, IN2**	比较两个无符号字节值： 如果 IN1 = IN2，则结果为 TRUE
 	**LDW= IN1, IN2** **OW= IN1, IN2** **AW= IN1, IN2**	比较两个有符号整数值： 如果 IN1 = IN2，则结果为 TRUE

LAD 触点，FBD 功能框	STL	比较结果
IN1 ==D IN2 IN1 ==D OUT IN2	LDD= IN1, IN2 OD= IN1, IN2 AD= IN1, IN2	比较两个有符号双精度整数值： 如果 IN1 = IN2，则结果为 TRUE
IN1 ==R IN2 IN1 ==R OUT IN2	LDR= IN1, IN2 OR= IN1, IN2 AR= IN1, IN2	比较两个有符号实数值： 如果 IN1 = IN2，则结果为 TRUE

说明

以下条件会导致非致命错误，将能流设置为 OFF（ENO 位 = 0），并且使用值 0 作为比较结果

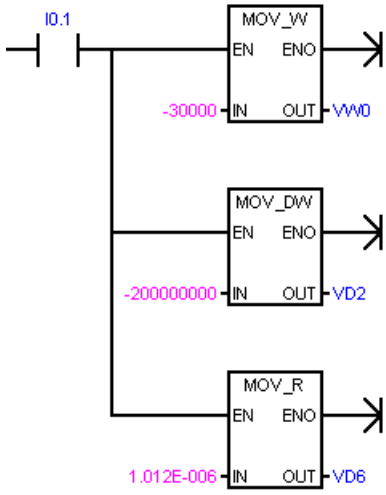
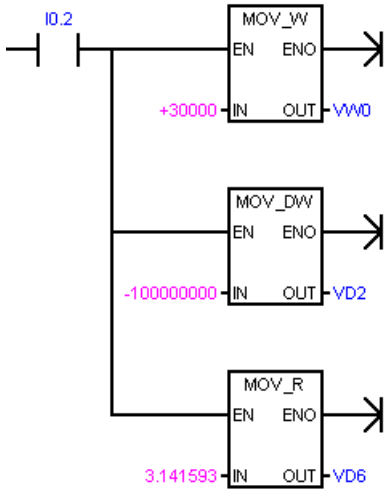
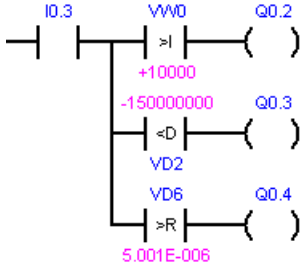
- 遇到非法间接地址（任意比较指令）
- 比较实数指令遇到非法实数（例如 NaN）

为了避免这些情况的发生，首先应确保正确初始化指针以及包含实数的值，然后再执行使用这些值的比较指令。

无论能流的状态如何，都会执行比较指令。

输入/输出	数据类型	操作数
IN1、IN2	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数
	INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*LD、*AC、常数
	DINT	ID、QD、VD、MD、SMD、SD、LD、AC、HC、*VD、*LD、*AC、常数
	REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC、常数
OUT	BOOL	LAD：能流 FBD：I、Q、V、M、SM、S、T、C、L、逻辑流

比较值示例

LAD		STL
	激活 I0.1，以便在 V 存储器地址中装载较小的值，使比较结果为 FALSE，并将状态指示灯设为 OFF。	**Network 1** **LD I0.1** **MOVW -30000, VW0** **MOVD -200000000, VD2** **MOVR 1.012E-006, VD6**
	激活 I0.2 以便在 V 存储器地址中装载较大的值，使比较结果为 TRUE，并将状态指示灯设为 ON。	**Network 2** **LD I0.2** **MOVW +30000, VW0** **MOVD -100000000, VD2** **MOVR 3.141593, VD6**
	激活 I0.3，以执行比较。 整数字比较测试 VW0 > +10000 是否为 TRUE。 还可以比较存储在可变存储器中的两个值，例如 VW0 > VW100。	**Network 3** **LD I0.3** **LPS** **AW> VW0, +10000** **= Q0.2** **LRD** **AD< -150000000, VD2** **= Q0.3** **LPP** **AR> VD6, 5.001E-006** **= Q0.4**

另请参见

常数

9.4.2 比较字符串

比较字符串指令可比较两个 ASCII 字符串。

对于 **LAD** 和 **FBD**： 比较结果为 TRUE 时，比较指令将接通触点 (LAD) 或输出 (FBD)。

对于 **STL**： 比较结果为 TRUE 时，比较指令可装载 1、将 1 与逻辑栈顶中的值进行“与”运算或者“或”运算。

可以在两个变量或一个常数和一个变量之间进行比较。 如果比较中使用了常数，则它必须为顶部参数（LAD 触点/ FBD 功能框）或第一参数 (STL)。

在程序编辑器中，常数字符串参数赋值必须以双引号字符开始和结束。 常数字符串条目的最大长度是 126 个字符（字节）。

相反，变量字符串由初始长度字节的字节地址引用，字符字节存储在下一个字节地址处。 变量字符串的最大长度为 254 个字符（字节），并且可在数据块编辑器进行初始化（前后带双引号字符）。

LAD 触点 FBD 功能框	STL	说明
IN1 ==S IN2 IN1 ==S OUT IN2	**LDS= IN1, IN2** **OS= IN1, IN2** **AS= IN1, IN2**	比较两个 STRING 数据类型的字符串： 如果字符串 IN1 **等于**字符串 IN2，则结果为 TRUE。
IN1 <>S IN2 IN1 <>S OUT IN2	**LDS<> IN1, IN2** **OS<> IN1, IN2** **AS<> IN1, IN2**	比较两个 STRING 数据类型的字符串： 如果字符串 IN1 **不等于**字符串 IN2，则结果为 TRUE。

说明

以下条件会导致非致命错误，能流将设置为 OFF（ENO 位 = 0），并采用值 0 作为比较结果：

- 遇到非法间接地址（任意比较指令）
- 遇到长度大于 254 个字符的变量字符串（比较字符串指令）
- 变量字符串的起始地址和长度使其不适合所指定的存储区（比较字符串指令）

为了避免这些情况的发生，首先应确保正确初始化指针以及用于保留 ASCII 字符串的存储单元，然后再执行使用这些值的比较指令。 确保为 ASCII 字符串预留的缓冲区能够完全放入指定的存储区。

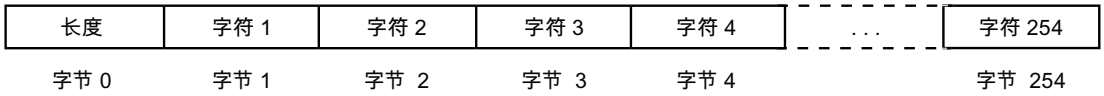
无论能流的状态如何，都会执行比较指令。

输入/输出	数据类型	操作数
IN1	STRING	VB、LB、*VD、*LD、*AC、常数字符串
IN2	STRING	VB、LB、*VD、*LD、*AC
OUT	BOOL	LAD：能流 FBD：I、Q、V、M、SM、S、T、C、L、逻辑流

STRING 数据类型的格式

字符串变量是一个字符序列，其中的每个字符均以字节形式存储。 STRING 数据类型的第一个字节定义字符串的长度，即字符字节数。

下图所示为存储器中以变量形式存储的 STRING 数据类型。 字符串的长度可以是 0 到 254 个字符。 变量字符串的最大存储要求为 255 个字节（长度字节加上 254 个字符）。



如果直接在程序编辑器中输入常数字符串参数（最多 126 个字符），或在数据块编辑器中初始化变量字符串（最多 254 个字符），则字符串赋值必须以双引号字符开始和结束。

另请参见常数

9.5 转换

9.5.1 标准转换指令

这些指令可以将输入值 IN 转换为分配的格式，并将输出值存储在由 OUT 分配的存储单元中。例如，您可以将双精度整数值转换为实数。也可以在整数与 BCD 格式之间进行转换。

标准转换

LAD/FBD	STL	描述
B_I EN ENO IN OUT	**BTI IN, OUT**	**字符到整数：** 将字节值 IN 转换为整数值，并将结果存入分配给 OUT 的地址中。字节是无符号的，因此没有符号扩展位。
I_B EN ENO IN OUT	**ITB IN, OUT**	**整数到字节：** 将字值 IN 转换为字节值，并将结果存入分配给 OUT 的地址中。可转换 0 到 255 之间的值。所有其它值将导致溢出，且输出不受影响。 **注：**要将整数转换为实数，请先执行整数到双精度整数指令，然后执行双精度整数到实数指令。
I_DI EN ENO IN OUT	**ITD IN, OUT**	**整数到双精度整数：** 将整数值 IN 转换为双精度整数值，并将结果存入分配给 OUT 的地址中。符号位扩展到高字节中。
DI_I EN ENO IN OUT	**DTI IN, OUT**	**双精度整数到整数：** 将双精度整数值 IN 转换为整数值，并将结果存入分配给 OUT 的地址处。如果转换的值过大以至于无法在输出中表示，则溢出位将置位，并且输出不受影响。
DI_R EN ENO IN OUT	**DTR IN, OUT**	**双精度整数到实数：** 将 32 位有符号整数 IN 转换为 32 位实数，并将结果存入分配给 OUT 的地址处。
BCD_I EN ENO IN OUT I_BCD EN ENO IN OUT	**BCDI OUT** **IBCD OUT**	**BCD 到整数：** 将二进制编码的十进制 WORD 数据类型值 IN 转换为整数 WORD 数据类型的值，并将结果加载至分配给 OUT 的地址中。IN 的有效范围为 0 到 9999 的 BCD 码。 **整数到 BCD：** 将输入整数 WORD 数据类型值 IN 转换为二进制编码的十进制 WORD 数据类型，并将结果加载至分配给 OUT 的地址中。IN 的有效范围为 0 到 9999 的整数。 对于 STL，IN 和 OUT 参数使用同一地址。

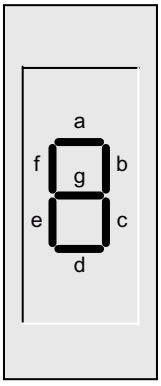
LAD/FBD	STL	描述
ROUND EN ENO IN OUT TRUNC EN ENO IN OUT	ROUND IN, OUT TRUNC IN, OUT	**取整：** 将 32 位实数值 IN 转换为双精度整数值，并将取整后的结果存入分配给 OUT 的地址中。如果小数部分大于或等于 0.5，该实数值将进位。 **截断：** 将 32 位实数值 IN 转换为双精度整数值，并将结果存入分配给 OUT 的地址中。只有转换了实数的整数部分之后，才会丢弃小数部分。 **注：**如果要转换的值不是一个有效实数或由于过大不能在输出中表示，则溢出位置位，但输出不受影响。
SEG EN ENO IN OUT	SEG IN, OUT	**SEG：** 要点亮七段显示中的各个段，可通过“段码”指令转换 IN 指定的字符字节，以生成位模式字节，并将其存入分配给 OUT 的地址中。 点亮的段表示输入字节最低有效位中的字符。
DF_R EN ENO IN OUT	DFTR IN, OUT	**双精度浮点到实数：** 将双精度浮点值 IN 转换为实数值，并将结果存入分配给 OUT 的地址处。如果转换的值过大以至于无法在输出中表示，则溢出位将置位，并且输出不受影响。
BCD_DI EN ENO IN OUT DI_BCD EN ENO IN OUT	BCDDI IN, OUT DIBCD IN, OUT	**BCD 到双精度整数：** 将二进制编码的十进制 DINT 数据类型值 IN 转换为双精度整数 DINT 数据类型的值，并将结果加载至分配给 OUT 的地址中。IN 的有效范围为 0 到 99999999 BCD。 **双精度整数到 BCD：** 将输入双精度整数 DINT 数据类型值 IN 转换为二进制编码的十进制 DINT 数据类型，并将结果加载至分配给 OUT 的地址中。IN 的有效范围为 0 到 99999999（整数）。

ENO = 0 时的非致命错误条件	受影响的 SM 位
• 0006H 间接地址 • SM1.1 溢出 • SM1.6 无效 BCD	• SM1.1 溢出 • SM1.6 无效 BCD

输入/输出	数据类型	操作数
IN	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数
	WORD (BCD_I, I_BCD)、INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AIW、AC、*VD、*LD、*AC、常数
	DINT	ID、QD、VD、MD、SMD、SD、LD、HC、AC、*VD、*LD、*AC、常数
	REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC、常数
OUT	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC
	WORD（BCD_I、I_BCD）	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、*VD、*LD、*AC
	INT（B_I、DI_I）	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AQW、*VD、*LD、*AC
	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC

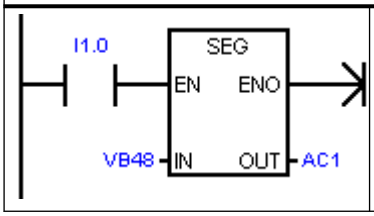
七段显示器的编码

(IN) LSD	分段显示	(OUT) - gfe dcba
0	0	0011 1111
1	1	0000 0110
2	2	0101 1011
3	3	0100 1111
4	4	0110 0110
5	5	0110 1101
6	6	0111 1101
7	7	0000 0111

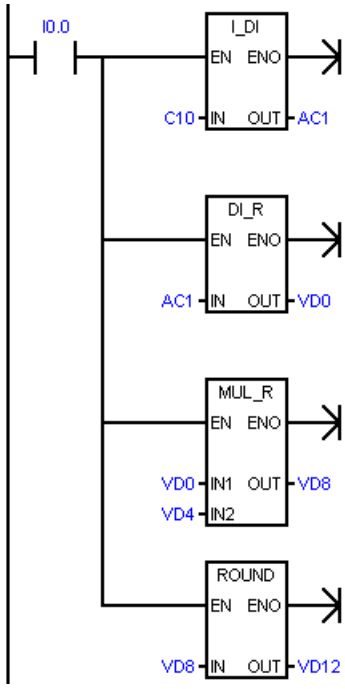
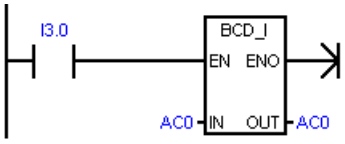


(IN) LSD	分段显示	(OUT) - gfe dcba
8	8	0111 1111
9	9	0110 0111
A	A	0111 0111
B	b	0111 1100
C	C	0011 1001
D	d	0101 1110
E	E	0111 1001
F	F	0111 0001

示例：使用 SEG 在七段显示屏上显示数值 5

LAD	STL
 05 SEG 6D VB48 AC1 5 （显示字符）	程序段 1 **LD I1.0** **SEG VB48, AC1**

示例：I_DI、DI_R 和 BCD_I

LAD		STL
	将英寸转换为厘米： 1. 将计数器值（英寸）载入 AC1（执行 C10=101）。 2. 将该值转换为实数（执行 VD0=101.0）。 3. 乘以 2.54 转换为厘米 (如：VD4=2.54，VD8=256.54)。 4. 将该值转回整数（执行 VD12=257）。	程序段 1 LD I0.0 ITD C10, AC1 DTR AC1, VD0 MOVR VD0, VD8 *R VD4, VD8 ROUND VD8, VD12
	将 BCD 值转换为整数（依次执行 AC0=1234、BCD_I、AC0=04D2）。	程序段 2 LD I0.3 BCDI AC0

另请参见

分配指令的常数值

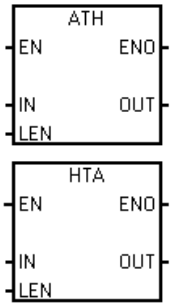
9.5.2 ASCII 字符数组转换

转换或转换为 ASCII 字符字节数组

ASCII 字符数组指令的字符输入输出采用 BYTE 数据类型。ASCII 字符数组为被引用的字节地址序列。

由于未使用长度字节，因此该数组并不是 STRING 数据类型。可使用 ASCII 字符串指令处理 STRING 数据类型的变量。

ASCII 转换为十六进制和十六进制转换为 ASCII

LAD/FBD	STL	说明
	**ATH IN, OUT, LEN** **HTA IN, OUT, LEN**	ATH 可以将长度为 LEN、从 IN 开始的 ASCII 字符转换为从 OUT 开始的十六进制数。可转换的最大 ASCII 字符数为 255 个字符。 HTA 可以将从输入字节 IN 开始的十六进制数转换为从 OUT 开始的 ASCII 字符。由长度 LEN 分配要转换的十六进制数的位数。可以转换的 ASCII 字符或十六进制数的最大数目为 255。 有效的 ASCII 输入字符为字母数字字符 0 到 9（十六进制代码值为 30 到 39）以及大写字符 A 到 F（十六进制代码值为 41 到 46）。

ENO = 0 时的非致命错误条件	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围 • SM1.7 ATH：非法 ASCII 值	• SM1.7 ATH：非法 ASCII 值

输入/输出	数据类型	操作数
IN, OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC
LEN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant

将数字值转换为用 ASCII 字符表示（ITA、DTA 和 RTA）

ASCII 字符输出的数字格式：

- 正值写入输出缓冲区时不带符号。
- 负值写入输出缓冲区时带前导负号 (-)。
- 小数点左侧的前导零会被隐藏，但与小数点相邻的数字除外。
- 数值在输出缓冲区中是右对齐的。
- 实数：小数点右侧的值取整，以与分配的小数点右侧的位数相符。
- 实数：输出缓冲区的大小必须至少比小数点右侧的位数多三个字节。

整数转换为 ASCII

LAD/FBD	STL	说明
ITA EN ENO IN OUT FMT	**ITA IN, OUT, FMT**	整数转换为 ASCII 指令可以将整数值 IN 转换为 ASCII 字符数组。格式参数 FMT 将分配小数点右侧的转换精度，并指定小数点显示为逗号还是句点。得出的转换结果将存入以 OUT 分配的地址开始的 8 个连续字节中。

ENO = 0 时的非致命错误条件	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围 • FMT 字节的 4 个最高有效位的 FMT 位不为零 • nnn > 5	无

输入/输出	数据类型	操作数
IN	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
FMT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC

输出缓冲区的大小始终为 8 个字节。通过 nnn 字段分配输出缓冲区中小数点右侧的位数。nnn 字段的有效范围是 0 到 5。如果分配 0 位数到小数点右侧，则转换后的值无小数点。对于 nnn 值大于 5 的情况，将使用 ASCII 空格字符填充输出缓冲区。c 位指定使用逗号 (c=1) 还是小数点 (c=0) 作为整数部分与小数部分之间的分隔符。4 个最高有效位必须始终为零。

下图中给出了一个数值的例子，其格式为使用小数点 (c=0)，小数点右侧有三位 (nnn=011)。

整数转换为 ASCII (ITA) 指令的 FMT 操作数

FMT

MSB							LSB
7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

c = 逗号 (1) 或小数点 (0)
nnn= 小数点右侧的位数

	输出	输出 +1	输出 +2	输出 +3	输出 +4	输出 +5	输出 +6	输出 +7
in = 12				0	.	0	1	2
in = -123			-	0	.	1	2	3
in = 1234				1	.	2	3	4
in = -12345		-	1	2	.	3	4	5

双整数转换为 ASCII

LAD/FBD	STL	说明
DTA EN ENO IN OUT FMT	DTA IN, OUT, FMT	**双精度整数转换为 ASCII** 指令可将双字 IN 转换为 ASCII 字符数组。格式参数 FMT 指定小数点右侧的转换精度。得出的转换结果将存入以 OUT 开头的 12 个连续字节中。

ENO = 0 时的非致命错误条件	受影响的 SM 位
• 0006H 无效间接地址 • 0091H 操作数超出范围 • FMT 字节的 4 个最高有效位的 FMT 位不为零 • nnn > 5	• 无

输入/输出	数据类型	操作数
IN	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
FMT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC

输出缓冲区的大小始终为 12 个字节。输出缓冲区中小数点右侧的位数由 nnn 字段分配。nnn 字段的有效范围是 0 到 5。如果分配 0 位数到小数点右侧，则转换后的值无小数点。对于 nnn 值大于 5 的情况，将使用 ASCII 空格字符填充输出缓冲区。c 位指定使用逗号 (c=1) 还是小数点 (c=0) 作为整数部分与小数部分之间的分隔符。4 个最高有效位必须始终为零。

下图给出了一个数值作为示例，其格式为使用小数点 (c=0)，小数点右侧有四位 (nnn=100)。

双整数转换为 ASCII (DTA) 指令的 FMT 操作数

FMT

MSB							LSB
7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

c = 逗号 (1) 或小数点 (0)
nnn= 小数点右侧的位数

	输出	输出 +1	输出 +2	输出 +3	输出 +4	输出 +5	输出 +6	输出 +7	输出 +8	输出 +9	输出 +10	输出 +11
输入 = 12						-	0	.	0	0	1	2
输入 = 1234567					1	2	3	.	4	5	6	7

实数转换为 ASCII

LAD/FBD	STL	说明
RTA EN ENO IN OUT FMT	RTA IN, OUT, FMT	**实数转换为 ASCII** 指令可将实数值 IN 转换成 ASCII 字符。格式参数 FMT 会指定小数点右侧的转换精度、小数点显示为逗号还是句点以及输出缓冲区大小。得出的转换结果会存入以 OUT 开头的输出缓冲区中。

ENO = 0 时的非致命错误条件	受影响的 SM 位
• 0006H 无效间接地址 • 0091H 操作数超出范围 • nnn > 5 • ssss < 3 • ssss < OUT 中的字符数	无

输入/输出	数据类型	操作数
IN	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
FMT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC

得出的 ASCII 字符数（或长度）就是输出缓冲区的大小，它的值在 3 到 15 个字节或字符之间。

实数格式最多支持 7 位有效数字。尝试显示 7 位以上的有效数字将导致舍入错误。

下图显示了 RTA 指令的格式操作数 (FMT)。通过 ssss 字段分配输出缓冲区的大小。0、1 或 2 个字节大小无效。输出缓冲区中小数点右侧的位数由 nnn 字段分配。nnn 字段的有效范围是 0 到 5。如果分配 0 位数到小数点右侧，则转换后的值无小数点。如果 nnn 的值大于 5 或者分配的输出缓冲区太小以致无法存储转换后的值，则使用 ASCII 空格填充输出缓冲区。c 位指定使用逗号 (c=1) 还是小数点 (c=0) 作为整数部分与小数部分之间的分隔符。

下图给出了一个数值作为示例，其格式为使用小数点 (c=0)、小数点右侧有一位 (nnn=001)、缓冲区的大小为六个字节 (ssss=0110)。

实数转换为 ASCII (RTA) 指令的 FMT 操作数

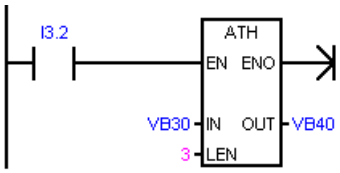
FMT

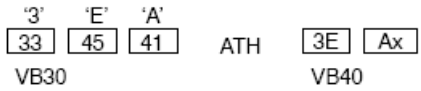
MSB							LSB
7	6	5	4	3	2	1	0
s	s	s	s	c	n	n	n

ssss = 输出缓冲区大小
c = 逗号 (1) 或小数点 (0)
nnn= 小数点右侧的位数

	输出	输出 +1	输出 +2	输出 +3	输出 +4	输出 +5
in = 1234.5	1	2	3	4	.	5
in = -0.0004				0	.	0
in = -3.67526			-	3	.	7
in = 1.95				2	.	0

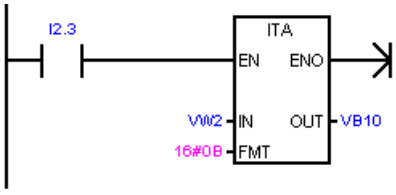
示例：ASCII 转换为十六进制

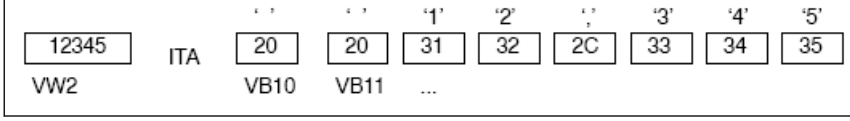
LAD	STL
	**Network 1** **LD I3.2** **ATH VB30, VB40, 3**



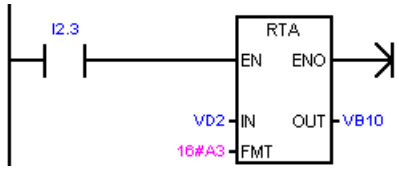
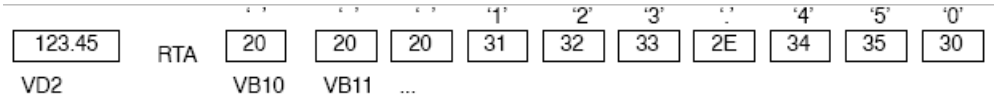
[1] "x"表示"半字节"（字节的一半）不变。

示例：整数转换为 ASCII

LAD	STL
 将 VW2 中的整数值转换为从 VB10 开始的 8 个 ASCII 字符，使用 16#0B 格式（逗号作小数点，保留 3 位小数）。	**Network 1** **LD I2.3** **ITA VW2, VB10, 16#0B**



示例：实数转换为 ASCII

LAD		STL
	将 VD2 中的实数值转换为从 VB10 开始的 10 个 ASCII 字符，使用 16#A3 格式（句点作小数点，保留 3 位小数）。	**Network 1** **LD I2.3** **RTA VD2, VB10, 16#A3**
		

另请参见

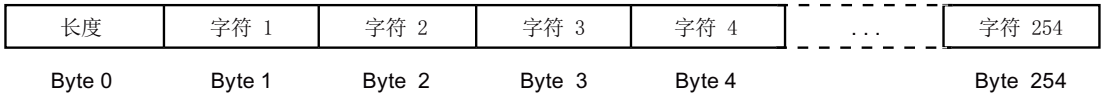
分配指令的常数值

9.5.3 数值转换为 ASCII 字符串

STRING 数据类型的格式

字符串变量是一个字符序列，其中的每个字符均以字节形式存储。STRING 数据类型的第一个字节定义字符串的长度，即字符字节数。

下图所示为存储器中以变量形式存储的 STRING 数据类型。字符串的长度可以是 0 到 254 个字符。变量字符串的最大存储要求为 255 个字节（长度字节加上 254 个字符）。



如果直接在程序编辑器中输入常数字符串参数（最多 126 个字符），或在数据块编辑器中初始化变量字符串（最多 254 个字符），则字符串赋值必须以双引号字符开始和结束。

ASCII 输出数字格式

- 正值写入输出缓冲区时不带符号。
- 负值写入输出缓冲区时带前导负号 (-)。
- 小数点左侧的前导零会被隐藏，但与小数点相邻的数字除外。
- 输出字符串中的值为右对齐。

- 实数：小数点右侧的值被舍入为小数点右侧的指定位数。
- 实数：输出字符串的大小必须比小数点右侧的位数多至少三个字节。

整数到字符串转换

LAD/FBD	STL	说明
I_S EN ENO IN OUT FMT	ITS IN, OUT, FMT	整数转换为字符串的指令会将整数字 IN 转换为长度为 8 个字符的 ASCII 字符串。格式 (FMT) 分配小数点右侧的转换精度，并指定小数点显示为逗号还是句点。结果字符串会写入从 OUT 处开始的 9 个连续字节中。

ENO = 0 时的非致命错误条件	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围 • 格式非法 (nnn > 5) • FMT 字节的四个最高有效位的 FMT 位不为零	无

输入/输出	数据类型	操作数
IN	INT	IW、QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC、常数
FMT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, 常数
OUT	STRING	VB, LB, *VD, *LD, *AC

输出字符串的长度始终为 8 个字符。输出缓冲区中小数点右侧的位数由 nnn 字段分配。nnn 字段的有效范围是 0 到 5。如果分配 0 位数到小数点右侧，则转换后的值无小数点。对于 nnn 大于 5 的值，输出为 8 个 ASCII 空格字符组成的字符串。c 位指定使用逗号 (c=1) 还是小数点 (c=0) 作为整数部分与小数部分之间的分隔符。格式的最高有效 4 位必须是零。

下图还给出了值的示例，其格式为：使用小数点 (c = 0)，小数点右侧有三位数 (nnn = 011)。OUT 处的值为下一字节地址中存储的字符串的长度。

整数转换为字符串指令的 FMT 参数

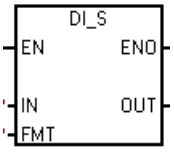
FMT

MSB							LSB
7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

c = 逗号 (1) 或小数点 (0)
nnn= 小数点右侧的位数

	输出	输出 +1	输出 +2	输出 +3	输出 +4	输出 +5	输出 +6	输出 +7	输出 +8
输入 = 12	8				0	.	0	1	2
输入 = -123	8			-	0	.	1	2	3
输入 = = 123	8				1	.	2	3	4
输入 = -12345	8		-	1	2	.	3	4	5

双精度整数到字符串转换

LAD/FBD	STL	说明
	**DTS IN, OUT, FMT**	双整数转换为字符串的指令会将双整数 IN 转换为长度为 12 个字符的 ASCII 字符串。格式 (FMT) 分配小数点右侧的转换精度，并指定小数点显示为逗号还是句点。结果字符串会写入从 OUT 处开始的 13 个连续字节中。

ENO = 0 时的非致命错误条件	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围 • 格式非法 (nnn > 5) • FMT 字节的四个最高有效位的 FMT 位不为零	无

输入/输出	数据类型	操作数
IN	DINT	ID、QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC、常数
FMT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, 常数
OUT	STRING	VB, LB, *VD, *LD, *AC

输出字符串的长度始终为 12 个字符。输出缓冲区中小数点右侧的位数由 nnn 字段指定。nnn 字段的有效范围是 0 到 5。如果分配 0 位数到小数点右侧，则该值不显示小数点。对于 nnn 大于 5 的值，输出为 12 个 ASCII 空格字符组成的字符串。c 位指定使用逗号 (c=1) 还是小数点 (c=0) 作为整数与小数部分之间的分隔符。格式的高 4 位必须是零。

下图还给出了一个值的示例，其格式为：使用小数点 (c = 0)，小数点右侧有四位数 (nnn = 100)。OUT 处的值为下一字节地址中存储的字符串的长度。

双整数转换为字符串指令的 FMT 操作数

FMT

MSB							LSB
7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

c = 逗号 (1) 或小数点 (0)
nnn= 小数点右侧的位数

	输出	输出 +1	输出 +2	输出 +3	输出 +4	输出 +5	输出 +6	输出 +7	输出 +8	输出 +9	输出 +10	输出 +11	输出 +12
输入 = 12	12							0	.	0	0	1	2
输入 = 1234567	12					1	2	3	.	4	5	6	7

实数到字符串转换

LAD/FBD		说明
R_S EN ENO IN OUT FMT	**RTS IN, OUT, FMT**	实数转换为字符串的指令会将实数值 IN 转换为 ASCII 字符串。格式 (FMT) 分配小数点右侧的转换精度、小数点显示为逗号还是句点以及输出字符串的长度。转换结果放置在以 OUT 开头的字符串中。结果字符串的长度在格式中指定，可以是 3 到 15 个字符。

ENO = 0 时的非致命错误条件	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围 • 格式非法 – (nnn > 5) – ssss < 3 – ssss < 所需字符数	无

输入/输出	数据类型	操作数
IN	REAL	ID、QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC、常数
FMT	BYTE	IB、QB、VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC、常数
OUT	STRING	VB, LB, *VD, *LD, *AC

CPU 使用的实数格式最多支持 7 位有效数字。尝试显示 7 位以上有效数字会产生舍入错误。

输出字符串的长度由 ssss 字段指定。0、1 或 2 个字节大小无效。输出缓冲区中小数点右侧的位数由 nnn 字段分配。nnn 字段的有效范围是 0 到 5。如果分配 0 位数到小数点右侧，则该值不显示小数点。如果 nnn 大于 5，或者因分配的输出字符串长度太小而无法存储转换的值，则会用 ASCII 空格字符填充输出字符串。c 位指定使用逗号 (c=1) 还是小数点 (c=0) 作为整数与小数部分之间的分隔符。

下图还给出了一个值的示例，其格式为：小数点 (c = 0)，小数点右侧有一位数 (nnn = 001)，输出字符串的长度为 6 个字符 (ssss = 0110)。OUT 处的值为下一字节地址中存储的字符串的长度。

实数转换为字符串指令的 FMT 操作数

FMT

MSB							LSB
7	6	5	4	3	2	1	0
s	s	s	s	c	n	n	n

ssss = 输出字符串的长度
c = 逗号 (1) 或小数点 (0)
nnn= 小数点右侧的位数

	输出	输出 +1	输出 +2	输出 +3	输出 +4	输出 +5	输出 +6
in = 1234.5	6	1	2	3	4	.	5
in = -0.0004	6				0	.	0
in = -3.67526	6			-	3	.	7
in = 1.95	6				2	.	0

另请参见

分配指令的常数值

9.5.4 ASCII 子字符串转换为数值

LAD/FBD	STL	说明
S_I EN ENO IN OUT INDX	**STI IN, INDX, OUT**	ASCII 子字符串转换为整数值
S_DI EN ENO IN OUT INDX	**STD IN, INDX, OUT**	ASCII 子字符串转换为双整数值
S_R EN ENO IN OUT INDX	**STR IN, INDX, OUT**	ASCII 子字符串转换为实数值

ENO = 0 时的非致命错误条件	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围 • 009BH 索引 = 0 • SM1.1 溢出或非法值	• SM1.1 溢出或非法值

输入/输出	数据类型	操作数
IN	STRING	VB、LB、*VD、*LD、*AC、常数字符串
INDX	BYTE	VB、IB、QB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数
OUT	INT	VW、IW、QW、MW、SMW、SW、T、C、LW、AC、AQW、*VD、*LD、*AC
	DINT、REAL	VD、ID、QD、MD、SMD、SD、LD、AC、*VD、*LD、*AC

S_I（整数）和 S_DI（双整数）的字符串输入格式

[空格] [+ 或 -] [数字 0 - 9]

S_R（实数）的字符串输入格式

[空格] [+ 或 -] [数字 0 - 9] [. 或 ,] [数字 0 - 9]

INDX 参数

INDX 值通常设为 1，从字符串的第一个字符开始转换。INDX 值可设置为其它值，以在字符串中的不同点处开始转换。当输入字符串包含不属于要转换的数字一部分的文本时，可采用此方法。例如，如果输入字符串为“Temperature:77.8”，可将 INDX 设置为 13 来跳过字符串开头的单词“Temperature:”。

子字符串转换为实数的指令不会转换以科学记数法或指数形式表示实数的字符串。该指令不会产生溢出错误 (SM1.1)，但会将字符串转换为指数之前的实数，然后终止转换。例如，字符串“1.234E6”会转换为实数值 1.234，而不会出现错误。

达到字符串结尾或遇到第一个无效字符时，转换将终止。无效字符为非数字 (0 - 9) 的字符或以下字符之一：加号 (+)、减号 (-)、逗号 (,) 或句号 (.)。

当转换产生的整数值对于输出值来说过大时，会置位溢出错误 (SM1.1)。例如，当输入字符串产生的值大于 32767 或小于 -32768 时，子字符串转换为整数的指令会置位溢出错误。

当输入字符串不包含有效值而无法进行转换时，也会置位溢出错误 (SM1.1)。例如，如果输入字符串包含“A123”，则转换指令会置位 SM1.1（溢出），输出值保持不变。

有效和无效输入字符串的示例

有效的整数和双精度整数输入字符串

输入字符串	输出整数
'123'	123
'-00456'	-456
'123.45'	123
'+2345'	2345
'000000123ABCD'	123

有效的实数输入字符串

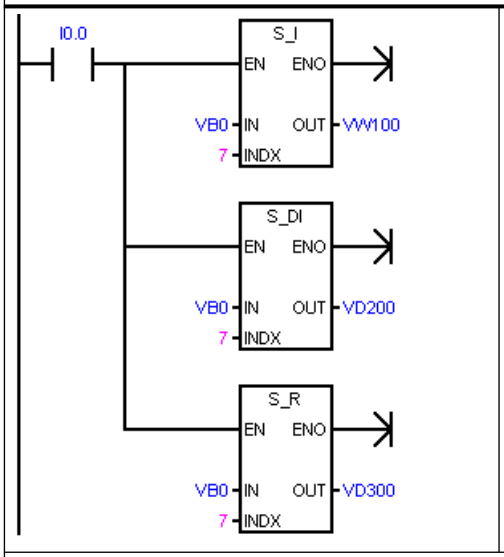
输入字符串	输出实数
'123'	123.0
'-00456'	-456.0
'123.45'	123.45
'+2345'	2345.0
'00.000000123'	0.000000123

有效输入字符串

输入字符串
'A123'
' '
'++123'
'+-123'
'+ 123'

字符串转换示例：子字符串转换为整数、双精度整数和实数

LAD



S_I 将数字字符串转换为整数值。

S_I 将数字字符串转换为双精度整数值。

S_R 将数字字符串转换为实数值。

STL

```
Network 1
LD I0.0
STI VB0, 7, VW100
STD VB0, 7, VD200
STR VB0, 7, VD300
```

VB0 … VB11

11	'T'	'e'	m'	'p'	' '	' '	'9'	'8'	'.'	'6'	'F'

执行该程序段后：
VW100（整数）= 98
VD200（双精度整数）= 98
VD300（实际）= 98.6

另请参见

分配指令的常数值

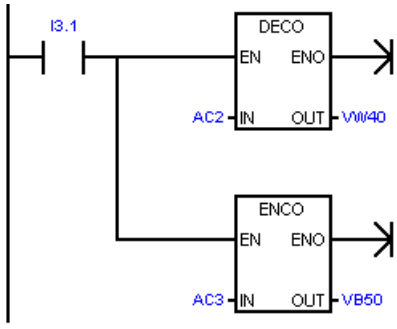
9.5.5 编码和解码

LAD/FBD	STL	说明
ENCO EN ENO IN OUT	**ENCO IN, OUT**	编码指令将输入字 IN 中设置的最低有效位的位编号写入输出字节 OUT 的最低有效“半字节”（4 位）中。
DECO EN ENO IN OUT	**DECO IN, OUT**	解码指令置位输出字 OUT 中与输入字节 IN 的最低有效“半字节”（4 位）表示的位号对应的位。输出字的所有其它位都被设置为 0。

ENO = 0 时的非致命错误条件	受影响的 SM 位
• 0006H 间接地址	无

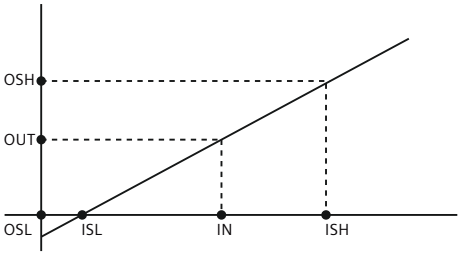
输入/输出	数据类型	操作数
IN	WORD (ENCO)	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*LD、*AC、常数
	BYTE (DECO)	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数
OUT	BYTE (ENCO)	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC
	WORD (DECO)	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AQW、*VD、*LD、*AC

示例：编码和解码

LAD		STL
	如果 AC2 包含错误位： 1. DECO 指令会置位 VW40 中与该错误代码对应的位。 2. ENCO 指令会将最低有效位转换为存储在 VB50 中的错误代码。	**Network 1** **LD I3.1** **DECO AC2, VW40** **ENCO AC3, VB50**
AC2 3 DECO 15 3 0 VW40 0000 0000 0000 1000	AC3 15 9 0 1000 0010 0000 0000 ENCO VB50 9	

9.5.6 量程变换

可使用“量程变换”指令通过将 IN 输入的值映射到指定的值范围来缩放该值。

LAD/FBD	STL	描述	
S_ITR EN ENO IN OUT ISH ISL OSH OSL	**SITR IN, ISH, ISL, OSH, OSL, OUT**	可使用此指令将模拟量输入转换为 S7-200 SMART 的内部数据。执行该指令时，输入 IN 处的整数值被缩放到由参数 OSH 和 OSL 定义的值范围。缩放的结果是实数，该实数存储在输出 OUT 处。	下图显示了如何转换值的示例：  OUT = [(OSH - OSL) * (IN - ISL)/(ISH - ISL)] + OSL • OUT：转换结果 • OSH：转换结果上限 • OSL：转换结果下限 • IN：换算对象 • ISH：转换对象上限 • ISL：转换对象下限
S_RTR EN ENO IN OUT ISH ISL OSH OSL	**SRTR IN, ISH, ISL, OSH, OSL, OUT**	可使用此指令将 REAL 转换为另一个 REAL。执行该指令时，输入 IN 处的实数值被缩放到由参数 OSH 和 OSL 定义的值范围。缩放的结果是实数，该实数存储在输出 OUT 处。	
S_RTI EN ENO IN OUT ISH ISL OSH OSL	**SRTI IN, ISH, ISL, OSH, OSL, OUT**	可使用此指令将 S7-200 SMART 的内部数据转换为模拟量输出。执行该指令时，输入 IN 处的实数值被缩放到由参数 OSH 和 OSL 定义的值范围。缩放的结果是整数，该整数存储在输出 OUT 处。	

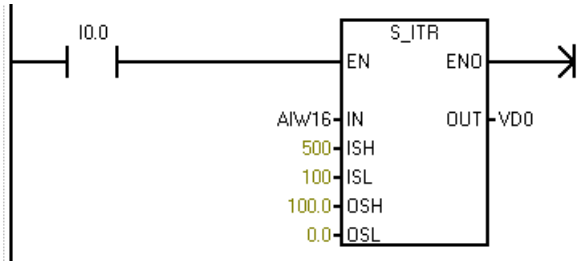
ENO = 0 时的非致命错误

- 0X0090：无效参数（(ISH ≤ ISL) 或 (OSH ≤ OSL) 或者 IN 超出范围 [ISL、ISH]）。

	输入/输出	数据类型	操作数
S_ITR	IN	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	ISH	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	ISL	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	OSH	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	OSL	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	OUT	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
S_RTR	IN	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	ISH	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	ISL	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	OSH	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	OSL	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	OUT	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
S_RTI	IN	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	ISH	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	ISL	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	OSH	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	OSL	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	OUT	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AQW, *VD, *AC, *LD

示例：

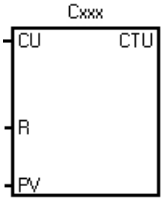
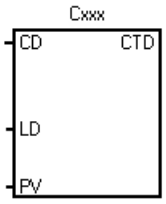
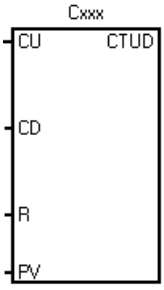
LAD					STL

			如果“I0.0”操作数的信号状态为“1”，则执行此指令。输入“IN”处的值范围为“ISH”和“ISL”，可缩放到“OSH”和“OSL”输入处的值所定义的值范围。结果存储在“OUT”输出中。如果指令执行无误，则 ENO 启用输出的信号状态为“1”，并置位了“OUT”输出。		程序段 1
					`LD I0.0`
					`SITR AIW16, 500, 100, 100.0, 0.0, VD0`

IN	ISH	ISL	OSH	OSL	OUT
350	500	100	100.0	0.0	62.5

9.6 计数器

9.6.1 计数器指令

LAD/FBD	STL	描述
	**CTU Cxxx, PV**	**LAD/FBD：**每次加计数 CU 输入从 OFF 转换为 ON 时，**CTU** 加计数指令就会从当前值开始加计数。当前值 Cxxx 大于或等于预设值 PV 时，计数器位 Cxxx 接通。当复位输入 R 接通或对 Cxxx 地址执行复位指令时，当前计数值会复位。达到最大值 32,767 时，计数器停止计数。 **STL：**R 复位输入为栈顶值。CU 加计数输入加载至第二堆栈层中
	**CTD Cxxx, PV**	**LAD/FBD：**每次 CD 减计数输入从 OFF 转换为 ON 时，**CTD** 减计数指令就会从计数器的当前值开始减计数。当前值 Cxxx 等于 0 时，计数器位 Cxxx 打开。LD 装载输入接通时，计数器复位计数器位 Cxxx 并用预设值 PV 装载当前值。达到零后，计数器停止，计数器位 Cxxx 接通。 **STL：**LD 装载输入为栈顶值。CD 减计数输入值会装载到第二堆栈层中
	**CTUD Cxxx, PV**	**LAD/FBD：**每次 CU 减计数输入从 OFF 转换为 ON 时，**CTUD** 加/减计数指令就会加计数，每次 CD 减计数输入从 OFF 转换为 ON 时，该指令就会减计数。计数器的当前值 Cxxx 保持当前计数值。每次执行计数器指令时，都会将 PV 预设值与当前值进行比较。 达到最大值 32,767 时，向上计数输入的下一个上升沿导致当前计数变成最小值 -32,768。达到最小值 -32,768 时，向下计数输入的下一个上升沿导致当前计数变成最大值 32,767。 当前值 Cxxx 大于或等于 PV 预设值时，计数器位 Cxxx 接通。否则，计数器位关断。当 R 复位输入接通或对 Cxxx 地址执行复位指令时，计数器复位。 **STL：**R 复位输入为栈顶值。CD 减计数输入值会加载至第二堆栈层中。CU 加计数输入值会装载到第三堆栈层中

输入/输出	数据类型	操作数
Cxxx	WORD	常数（C0 到 C511）
CU、CD (LAD)	BOOL	能流
CU、CD (FBD)	BOOL	I、Q、V、M、SM、S、T、C、L、逻辑流
R (LAD)	BOOL	能流
R (FBD)	BOOL	I、Q、V、M、SM、S、T、C、L、逻辑流
LD (LAD)	BOOL	能流
LD (FBD)	BOOL	I、Q、V、M、SM、S、T、C、L、逻辑流
PV	INT	IW、QW、VW、MW、SMW、SW、LW、T、C、AC、AIW、*VD、*LD、*AC、常数

说明

由于每个计数器有一个当前值，因此请勿将同一计数器编号分配给多个计数器。（编号相同的加计数器、加/减计数器和减计数器会访问相同的当前值。）

使用复位指令复位计数器时，计数器位会复位，并且计数器当前值会设为零。计数器编号可同时用于表示该计数器的当前值和计数器位。

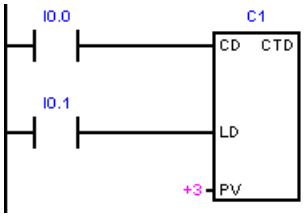
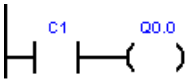
另请参见组态保持范围 - 系统块组态

计数器操作

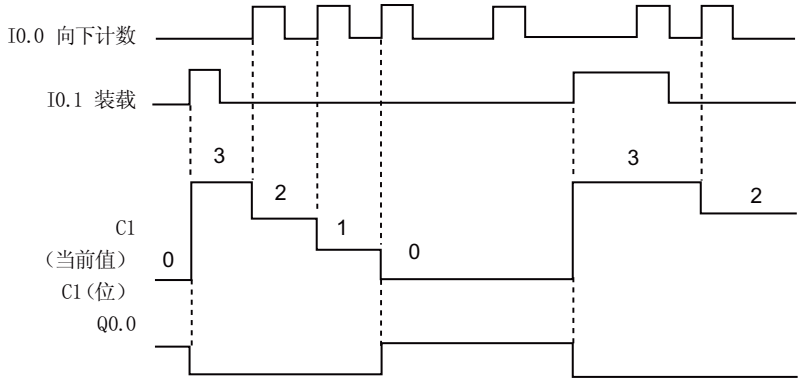
类型	操作	计数器位	上电循环/首次扫描
CTU	• CU 增加当前值。 • 当前值持续增加，直至达到 32,767。	以下情况下，计数器位接通：当前值 >= 预设值	• 计数器位关断。 • 当前值可保留 [1]
CTD	• CD 减少当前值，直至当前值达到 0。	以下情况下，计数器位接通：当前值 = 0	• 计数器位关断。 • 当前值可保留 [1]
CTUD	• CU 增加当前值。 • CD 减少当前值。 • 当前值持续增加或减少，直至计数器复位。	以下情况下，计数器位接通：当前值 >= 预设值	• 计数器位关断。 • 当前值可保留 [1]

[1] 您可以选择使计数器保留的当前值，但不能选择计数器位值。

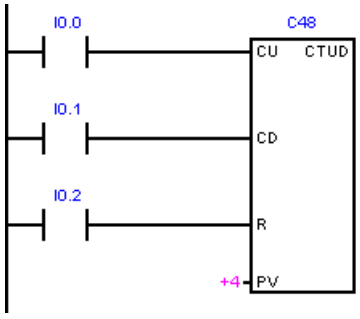
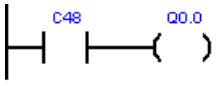
CTD 减计数示例

LAD		STL
	减计数器 C1 当前值从 3 计数至 0 当 I0.1 关断时，I0.0 的上升沿会减少 C1 当前值 I0.1 接通会装载减计数预设值 3	**Network 1** **LD I0.0** **LD I0.1** **CTD C1, +3**
	当计数器 C1 当前值 = 0 时，C1 位接通	**Network 2** **LD C1** **= Q0.0**

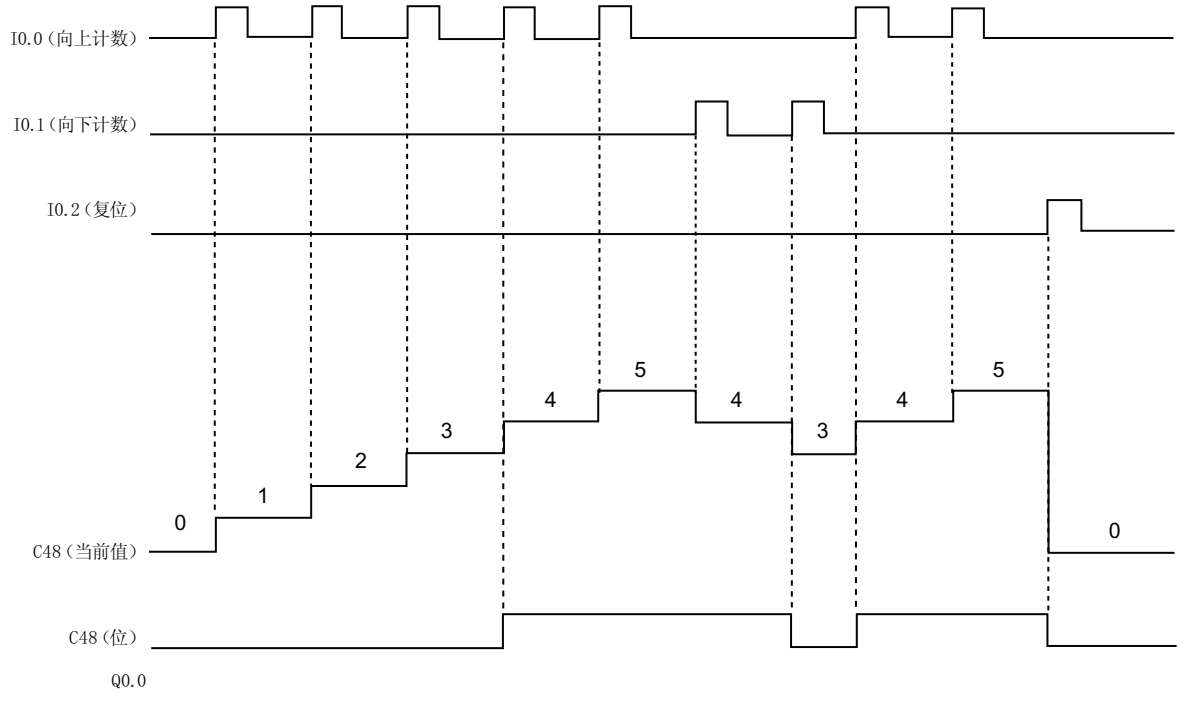
时序图



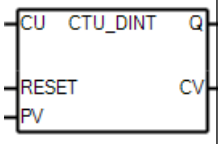
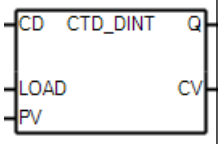
CTUD 加计数/减计数示例

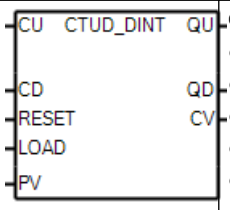
LAD		STL
	I0.0 加计数 I0.1 减计数 I0.2 将当前值复位为 0	**Network 1** **LD I0.0** **LD I0.1** **LD I0.2** **CTUD C48, +4**
	当前值 >= 4 时，加/减计数计数器 C48 接通 C48 位	**Network 2** **LD C48** **= Q0.0**

时序图



9.6.2 计数器指令(IEC)

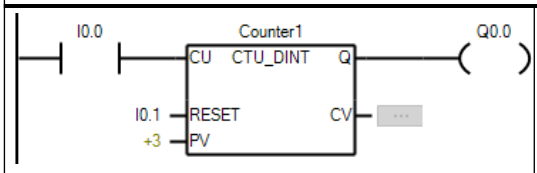
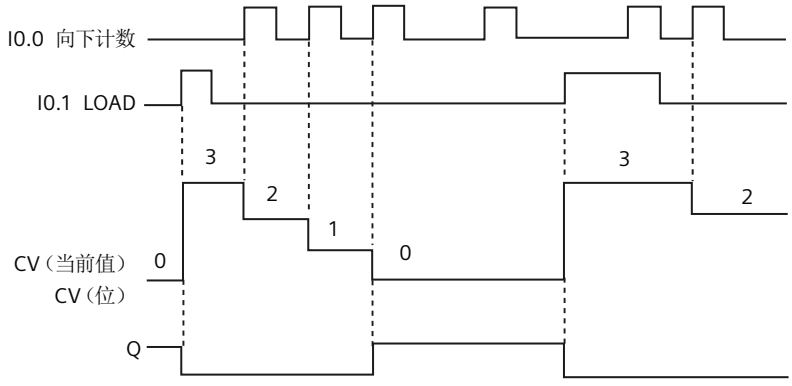
LAD/FBD	STL	描述
	**CTUDINT ctu1, ctu1.RESET, ctu1.PV, ctu1.CV**	**LAD/FBD：** “加计数”指令用于递增 CV 参数的值。如果参数 CU 的信号状态从“0”变为“1”（信号上升沿），则执行该指令，同时参数 CV 的当前计数器值加“1”。每检测到一个信号上升沿，计数器值就会递增，直到达到输出 CV 中所指定数据类型的上限。达到上限时，参数 CU 的信号状态将不再影响该指令。 可以通过参数 Q 查询计数状态。参数 Q 的信号状态由参数 PV 决定。如果当前计数器值大于或等于参数 PV 的值，则参数 Q 的信号状态将置位为“1”。在其它任何情况下，参数 Q 的信号状态均为“0”。 参数 RESET 的信号状态变为“1”时，参数 CV 的值将复位为“0”。只要参数 RESET 的信号状态为“1”，参数 CU 的信号状态就不会影响该指令。 **STL：**RESET 复位输入为栈顶值。CU 加计数输入加载至第二堆栈层中。
	**CTDDINT ctd1, ctd1.LOAD, ctd1.PV, ctd1.CV**	**LAD/FBD：** “减计数”指令用于递减 CV 参数的值。如果参数 CD 的信号状态从“0”变为“1”（信号上升沿），则执行该指令，同时参数 CV 的当前计数器值减 1。每检测到一个信号上升沿，计数器值就会递减 1，直到达到指定数据类型的下限为止，直到达到 0。达到 0 时，参数 CD 的信号状态将不再影响该指令。 可以通过参数 Q 查询计数状态。如果当前计数器值等于“0”，则参数 Q 的信号状态将置位为“1”。在其它任何情况下，参数 Q 的信号状态均为“0”。 当参数 LOAD 的信号状态变为“1”时，参数 CV 的值会设置为参数 PV 的值。只要参数 LOAD 的信号状态为“1”，参数 CD 的信号状态就不会影响该指令。 **STL：**LOAD 装载输入为栈顶值。CD 减计数输入值会装载到第二堆栈层中。

LAD/FBD	STL	描述
	**CTUDDINT ctud1,** **ctud1.CD,** **ctud1.RESET,** **ctud1.LOAD,** **ctud1.PV,** **ctud1.QD,** **ctud1.CV**	**LAD/FBD：**使用“加减计数”指令递增和递减 CV 参数的计数器值。如果参数 CU 的信号状态从“0”变为“1”（信号上升沿），则当前计数器值加 1 并存储在参数 CV 中。如果参数 CD 的信号状态从“0”变为“1”（信号上升沿），则参数 CV 的计数器值减 1。如果在一个程序周期内输入 CU 和 CD 都出现了一个信号上升沿，则参数 CV 的当前计数器值保持不变。 计数器值达到参数 CV 指定数据类型的上限后，停止递增。达到上限后，即使出现信号上升沿，计数器值也不再递增。达到 0 后，计数器值便不再递减。 当参数 LOAD 中的信号状态变为“1”时，参数 CV 的计数器值会设置为参数 PV 的值。只要参数 LOAD 的信号状态为“1”，参数 CU 和 CD 的信号状态就不会影响该指令。 当 RESET 参数的信号状态变为“1”时，计数器值将置位为 0。只要 RESET 参数的信号状态仍为“1”，参数 CU、CD 和 LOAD 信号状态的改变就不会影响“加减计数”指令。 可以在 QU 参数中查询加计数器的状态。如果当前计数器值大于或等于参数 PV 的值，则参数 QU 的信号状态将置位为“1”。在其它任何情况下，参数 QU 的信号状态均为“0”。 可以在 QD 参数中查询减计数器的状态。如果当前计数器值小于或等于“0”，则参数 QD 的信号状态将置位为“1”。在其它任何情况下，参数 QD 的信号状态均为“0”。 **STL：**RESET 复位输入为栈顶值。CD 减计数输入值会加载至第二堆栈层中。CU 加计数输入值会装载到第三堆栈层中。

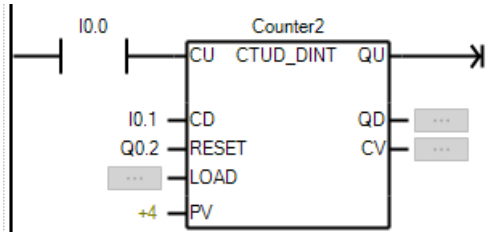
输入/输出	数据类型	操作数
CU	BOOL	能流
CD	BOOL	I、Q、V、M、T、C、SM、S、L、全局变量、FB 成员变量
RESET	BOOL	I、Q、V、M、T、C、SM、S、L、全局变量、FB 成员变量
LOAD	BOOL	I、Q、V、M、T、C、SM、S、L、全局变量、FB 成员变量
PV	DINT	I、Q、V、M、T、C、SM、AC、L、*AC、常数、全局变量、FB 成员变量
Q	BOOL	能流
QU	BOOL	能流

输入/输出	数据类型	操作数
QD	BOOL	I、Q、V、M、T、C、SM、S、L、全局变量，FB 成员变量
CV	DINT	I、Q、V、M、T、C、SM、AC、L、*AC、常数、全局变量，FB 成员变量

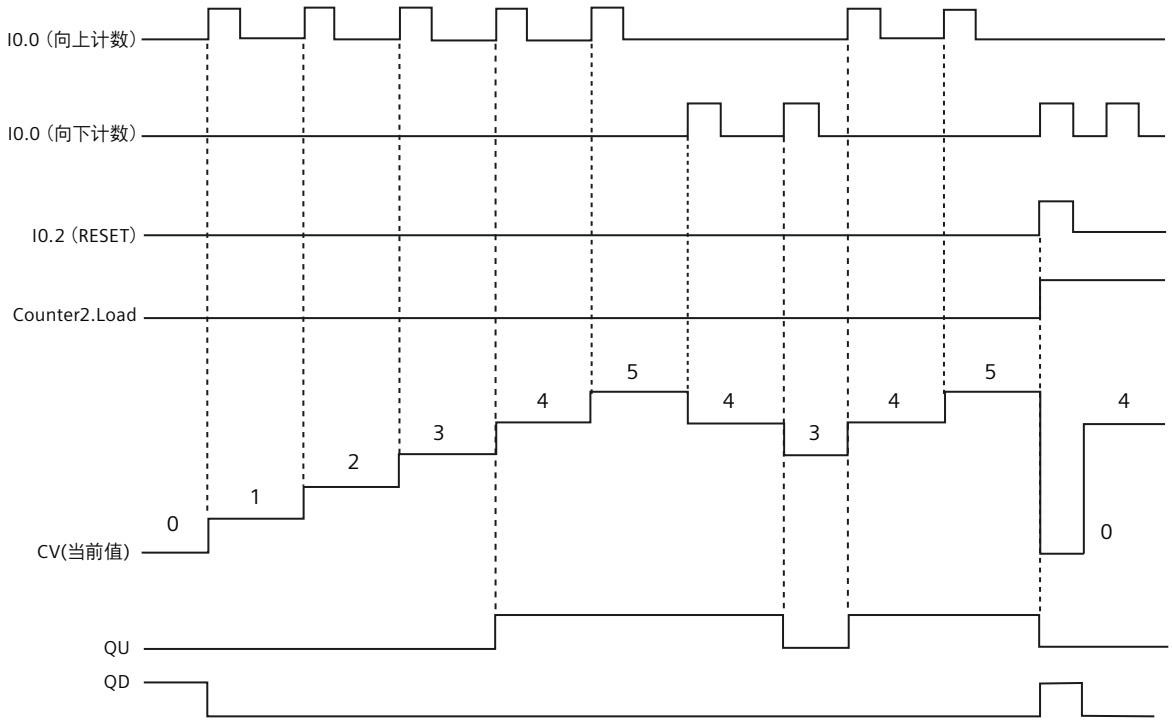
CTD 减计数示例

LAD		STL
	当前值 CV 从 3 计数至 0 当 I0.1 关断时，I0.0 的上升沿会减少当前值 CV I0.1 接通会装载减计数预设值 3	**Network 1** **LD I0.0** **CTUDINT Counter1, I0.1, +3,** **Counter1.CV** **= Q0.0**
时序图 		

CTUD 加计数/减计数示例

LAD		STL
	I0.0 加计数 I0.1 减计数 I0.2 将当前值复位为 0	**Network 1** **LD I0.0** **CTUDDINT Counter2, I0.1, Q0.2, Counter2.LOAD, +4, Counter2.QD, Counter2.CV**

时序图



9.6.3 高速计数器指令

高速计数器可对标准计数器无法控制的高速事件进行计数。标准计数器以受 PLC 扫描时间限制的较低速率运行。您可以使用 HDEF 和 HSC 指令创建自己的 HSC 例程，也可以使用高速计数器向导简化编程任务。

LAD/FBD	STL	描述
HDEF EN ENO HSC MODE	**HDEF HSC, MODE**	高速计数器定义指令 (HDEF) 选择特定高速计数器 (HSC0-7) 的工作模式。模式选择定义高速计数器的时钟、方向和复位功能。 必须为多达八个激活的高速计数器各使用一条高速计数器定义指令。
HSC EN ENO N	**HSC N**	高速计数器 (HSC) 指令根据 HSC 特殊存储器位的状态组态和控制高速计数器。参数 N 指定高速计数器编号。 高速计数器最多可组态为八种不同的工作模式。 每个计数器都有专用于时钟、方向控制、复位的输入，这些功能均受支持。在 AB 正交相，可以选择一倍 (1x) 或四倍 (4x) 的最高计数速率。所有计数器均以最高速率运行，互不干扰。

ENO = 0 时的错误条件		受影响的 SM 位
HDEF： • 0003H 输入点冲突 • 0004H 中断中存在非法指令 • 000AH HSC 重新定义 • 0016H 试图在输入上使用分配给运动功能使用的 HSC 或边缘中断 • 0090H HSC 编号无效	HSC： • 0001H 在 HDEF 之前执行 HSC • 0005H 同时执行 HSC/PLS • 0090H HSC 编号无效	无

输入/输出	数据类型	操作数
HSC	BYTE	HSC 编号常数（0、1、2、3、4、5、6 或 7）
MODE	BYTE	计数器 • 模式编号常数：八种可能的模式（0、1、3、4、6、7、9 或 10） 频率 • 模式编号常数：四个可能的模式（0、3、6、9）
N	WORD	HSC 编号常数（0、1、2、3、4、5、6 或 7）

HSC 运行

高速计数器可用作鼓式定时器的驱动，其中有一个装有增量轴编码器的轴，以恒定速度旋转。该轴编码器每转提供指定数量的计数值以及一个复位脉冲。来自轴编码器的时钟和复位脉冲为高速计数器提供输入。

高速计数器载入几个预设值中的第一个，并在当前计数值小于当前预设值的时间段内激活所需输出。计数器设置为在当前计数值等于预设值和出现复位时产生中断。

每次出现“当前计数值等于预设值”中断事件时，将装载一个新的预设值，同时设置输出的下一状态。当出现复位中断事件时，将设置输出的第一个预设值和第一个输出状态，并重复该循环。

由于程序中断发生的频率远低于高速计数器的计数速率，因此能够在对整个 PLC 扫描周期时间影响相对较小的情况下实现对高速操作的精确控制。通过中断，可在独立的中断例程中执行每次的新预设值装载操作，从而实现简单的状态控制。（此外，也可在单个中断例程中处理所有中断事件。）

HSC 输入分配及功能

工作模式相同时，所有高速计数器的功能均相同。HSC 输入连接（时钟、方向和复位）必须使用 CPU 的集成输入通道，如高速计数器汇总 (页 358)表所示。信号板或扩展模块上的输入通道不能用于高速计数器。

说明

使用高速计数器计数高频信号，必须确保对其输入进行正确接线和滤波。

在 S7-200 SMART CPU 中，所有高速计数器输入均连接至内部输入滤波电路。S7-200 SMART 的默认输入滤波设置为 6.4 ms，这样便将最大计数速率限定为 78 Hz。如需以更高频率计数，必须更改滤波器设置。

有关系统块滤波选项、最大计数频率、屏蔽要求及外部下拉电路的详细信息，请参见“高速输入降噪 (页 359)”。

可用的 HSC 计数器类型

模式	描述	输入		
0	具有内部方向控制功能的单相时钟计数器	时钟		
1		时钟		复位
3	具有外部方向控制功能的单相时钟计数器	时钟	方向	
4		时钟	方向	复位

模式	描述	输入		
6	具有 2 路时钟输入（加时钟和减时钟）的双相时钟计数器	向上计数时钟	向下计数时钟	
7		向上计数时钟	向下计数时钟	复位
9	AB 正交相计数器	A 相时钟	B 相时钟	
10		A 相时钟	B 相时钟	复位

HSC 操作规则

- 使用高速计数器之前，必须执行 HDEF 指令（高速计数器定义）选择计数器模式。使用首次扫描存储器位 SM0.1（首次扫描时，该位为 ON，后续扫描时为 OFF）直接执行 HDEF 指令，或调用包含 HDEF 指令的子例程。
- 可以使用所有计数器类型（带复位输入或不带复位输入）。
- 激活复位输入时，会清除当前值，并在您禁用复位输入之前保持清除状态。

引用信息

更多信息，请参考以下部分：

- 高速计数器编程 (页 360)
- 高速计数器汇总 (页 358)
- 高速计数器的初始化顺序示例 (页 371)
- 高速输入降噪 (页 360)

9.6.4 高速计数器汇总

	时钟 A	方向/时钟 B	复位	单相/双相最大时钟/输入速率	AB 正交相最大时钟/输入速率
HSC0、HSC1、HSC2、HSC3、HSC4、HSC5、HSC6、HSC7	I0.0-I1.7 [1]			• 200 kHz	• 200 kHz • A/B 相位 (4X) 频率支持高达 800 kHZ

[1] 对于同一通道 HSC，时钟 A 和方向/时钟 B 只能在 IB0 (I0.0 ~I0.7) 或 IB1 (I1.0 ~I1.7) 中同时进行选择。

9.6.5 高速输入降噪

使用 HSC 输入对高速脉冲计数

说明

高速输入接线必须使用屏蔽电缆

连接 HSC 输入通道 I0.0-I1.7 时，所用屏蔽电缆的长度不应超过 50 m。

要正确操作高速计数器，可能需要执行以下一项或两项操作。

- 调整 HSC 通道所用输入通道的“系统块”数字量输入滤波时间。在 HSC 通道对脉冲进行计数前 S7-200 SMART CPU 会应用输入滤波。这意味着，如果 HSC 输入脉冲以输入滤波过滤掉的速率发生，则 HSC 不会在输入上检测到任何脉冲。请务必将 HSC 的每路输入的滤波时间组态为允许以应用需要的速率进行计数的值。包括方向和复位输入。下表所示为 HSC 可检测到的各种输入滤波组态的最大输入频率。

输入滤波时间	可检测到的最大频率
0.2 µs	200 kHz
0.4 µs	200 kHz
0.8 µs	200 kHz
1.6 µs	200 kHz
3.2 µs	156 kHz
6.4 µs	78 kHz
12.8 µs	39 kHz
0.2 ms	2.5 kHz
0.4 ms	1.25 kHz
0.8 ms	625 Hz
1.6 ms	312 Hz
3.2 ms	156 Hz
6.4 ms	78 Hz
12.8 ms	39 Hz

- 如果生成 HSC 输入信号的设备未将输入信号驱动为高电平和低电平，则高速时可能出现信号失真。如果设备的输出是集电极开路晶体管，则可能出现这种情况。晶体管关闭时，没有任何因素将信号驱动为低电平状态。这种情况可能导致脉冲丢失。可通过将下拉电阻连接到输入信号的方法避免这种情况，如下图所示。由于 CPU 的输入电压是 24 V DC，因此电阻的额定功率必须为高功率。

说明

信号线与公共线需要成对走线，建议使用双绞线，以获得更佳的信号质量。

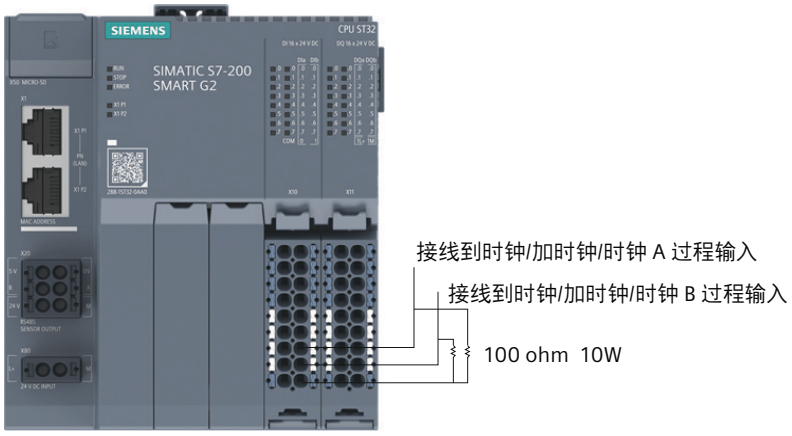


图 9-1　为集电极开路 HSC 输入驱动接线下拉电阻

9.6.6 高速计数器编程

可以使用高速计数器向导简化 HSC 编程任务。该向导可帮助用户选择计数器类型/模式、预设值/当前值以及计数器选项，并生成必要的特殊存储器分配、子例程和中断例程。

说明

使用高速计数器计数高频信号，必须确保对其输入进行正确滤波和接线。

在 S7-200 SMART CPU 中，所有高速计数器输入均连接至内部输入滤波电路。
S7-200 SMART CPU 的默认输入滤波设置为 6.4 ms，这样便将最大计数速率限定为 78 Hz。如需以更高频率计数，必须更改滤波器设置。

有关系统块滤波选项、最大计数频率、屏蔽要求及外部下拉电路的详细信息，请参见“高速输入降噪 (页 359)”。

组态高速计数器

请使用以下操作之一组态高速计数器向导：

- 打开向导：在“工具”(Tools) 菜单功能区的“向导”(Wizards) 区域中选择“高速计数器”(High-Speed Counter)。
- 打开向导：在项目导航的“向导”(Wizards) 文件夹中双击“高速计数器”(High-Speed Counter) 节点。

打开向导后，分配 HSC 设置值。可浏览向导设置页面、修改参数，然后生成新向导程序代码。

要使用高速计数器，程序必须执行以下基本任务：

- 定义计数器和模式（对每个计数器执行一次 HDEF 指令）。
- 在 SM 存储器中设置控制字节。
- 在 SM 存储器中设置当前值（初值）。
- 在 SM 存储器中设置预设值（目标值）。
- 分配并启用相应的中断例程。
- 激活高速计数器（执行 HSC 指令）。

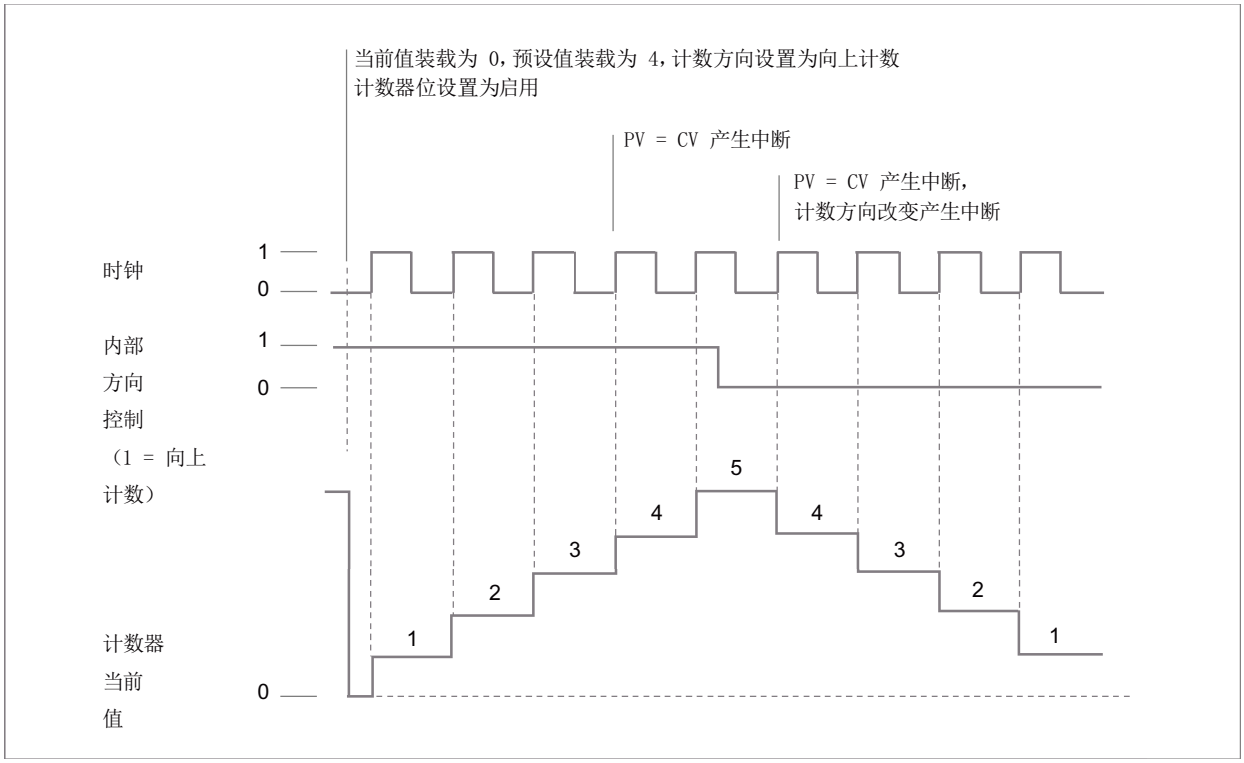
HDEF 指令设置计数模式

HDEF 指令分配 HSC 计数器模式。下表列出了为时钟、方向控制和复位功能分配的物理输入。同一输入无法用于两个不同的功能，但是其高速计数器的当前模式未使用的任何输入均可用于其他用途。例如，如果 HSC0 的当前模式为使用 I0.0 和 I0.4 的模式 1，则可将 I0.1、I0.2 和 I0.3 用于沿中断、HSC3 或运动控制输入。

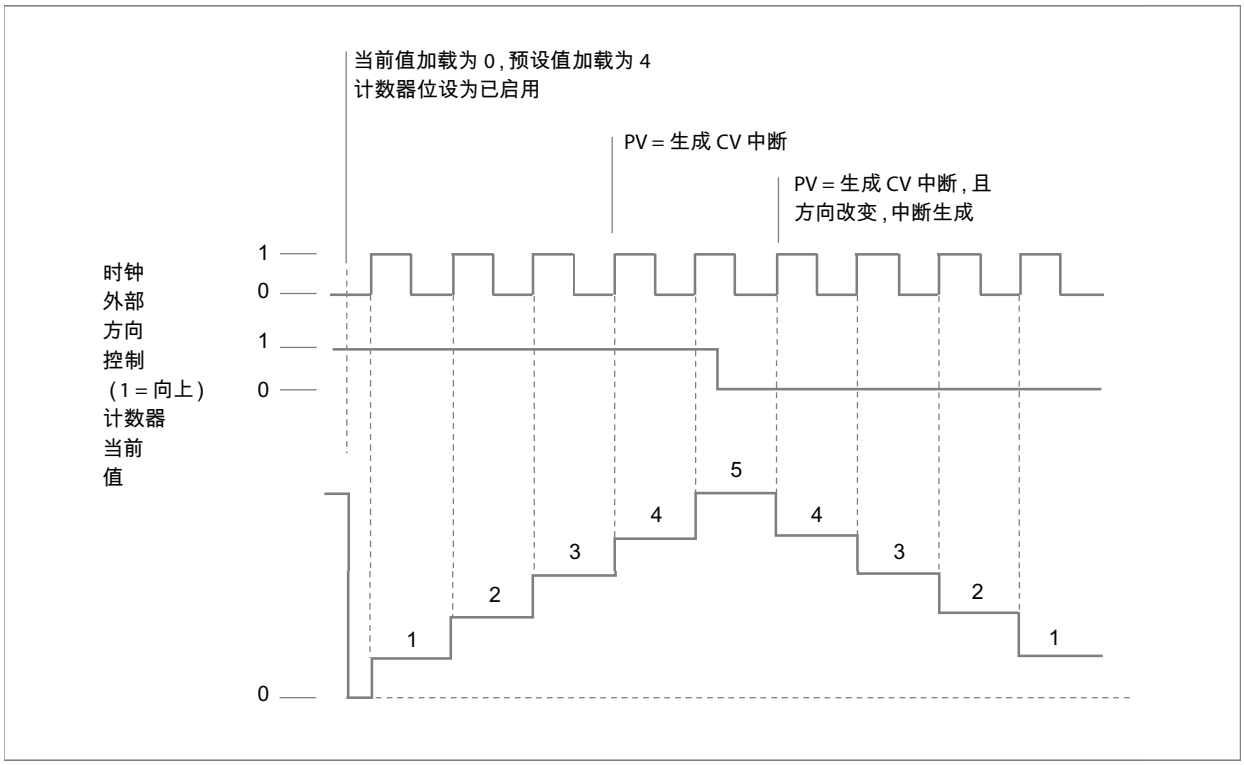
模式	说明	输入分配		
0	具有内部方向控制的单相计数器	时钟		
1		时钟		复位
3	具有外部方向控制的单相计数器	时钟	方向	
4		时钟	方向	复位
6	具有 2 个时钟输入的双相计数器	加时钟	减时钟	
7		加时钟	减时钟	复位
9	AB 正交相计数器	时钟 A	时钟 B	
10		时钟 A	时钟 B	复位

模式选择对计数操作的影响

HSC 模式 0 和 1

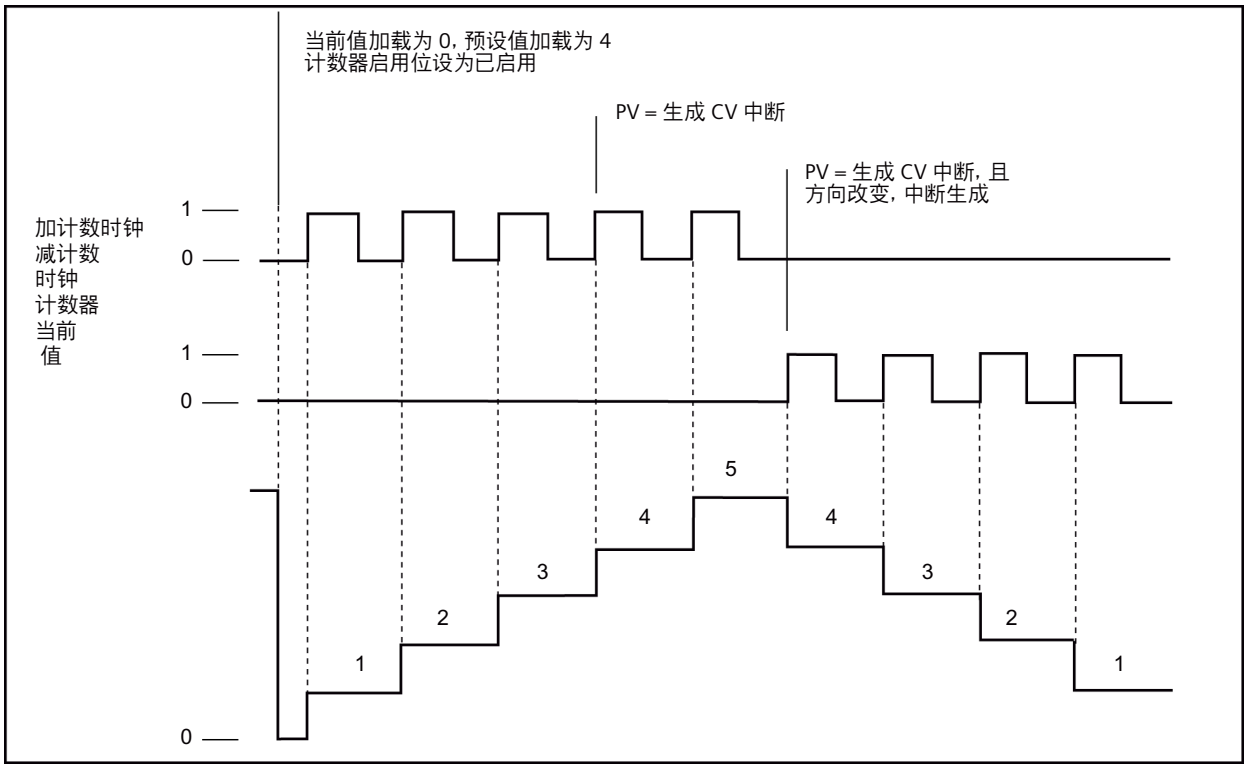


HSC 模式 3 和 4

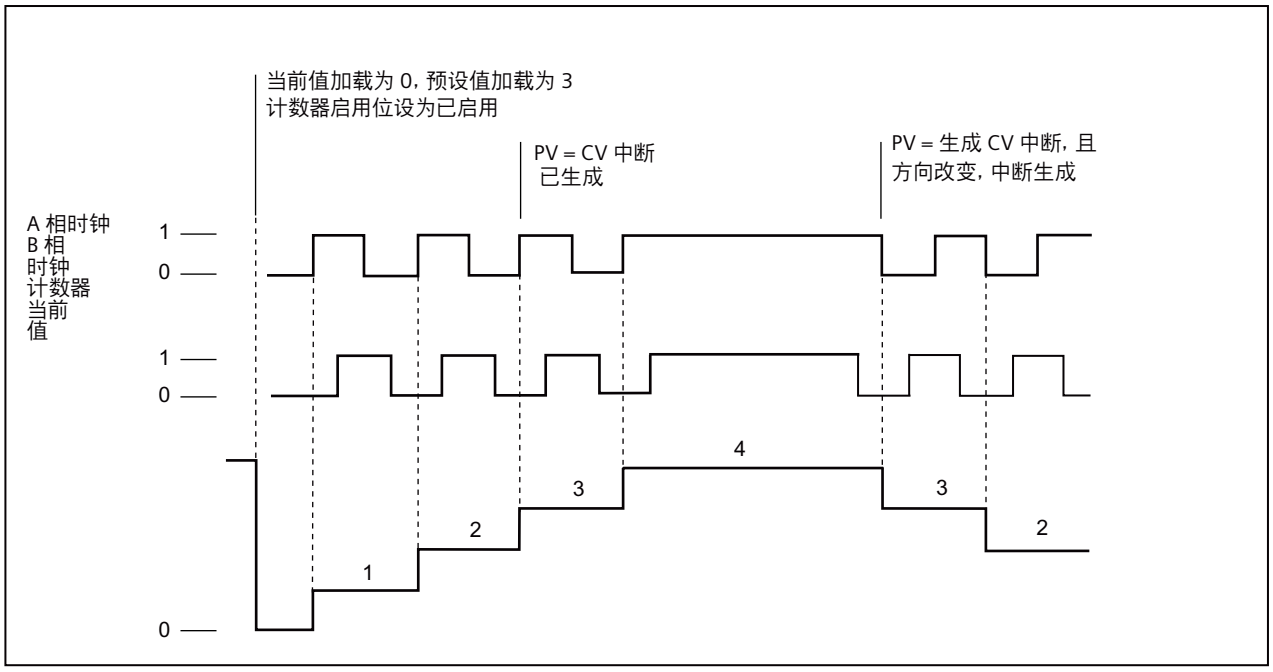


HSC 模式 6 和 7

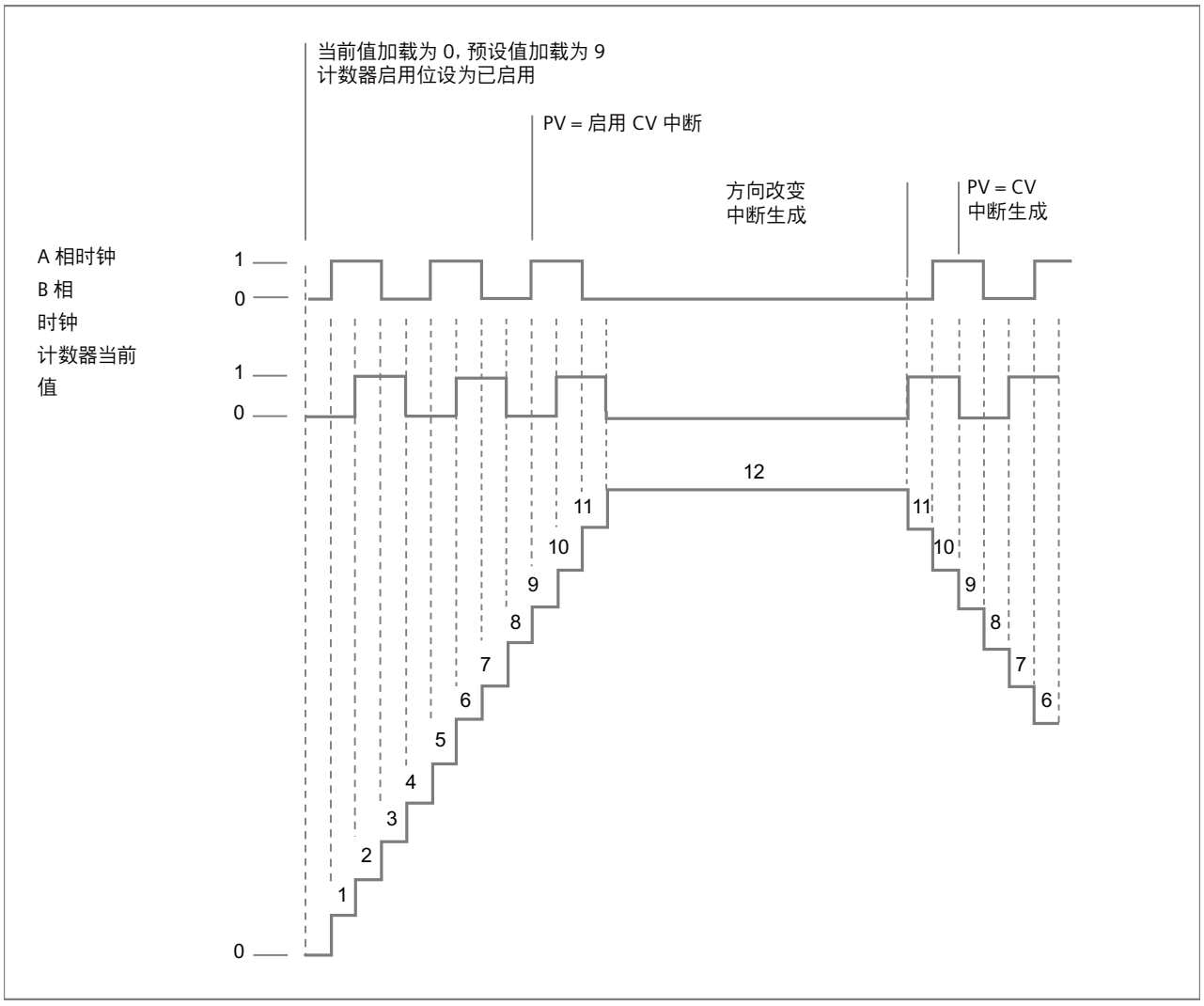
使用计数模式 6 或 7 时，如果加时钟和减时钟输入的上升沿在 0.3 微秒内发生，高速计数器可能认为这些事件同时发生。如果发生这种情况，当前值不改变，而且计数方向不改变。只要加时钟和减时钟输入的上升沿之间的间隔大于该时段，高速计数器就能够单独捕获每个事件。在两种情况下，均不会生成程序错误，而且计数器保持正确计数值。



HSC 模式 9 和 10（AB 正交相位 1X）



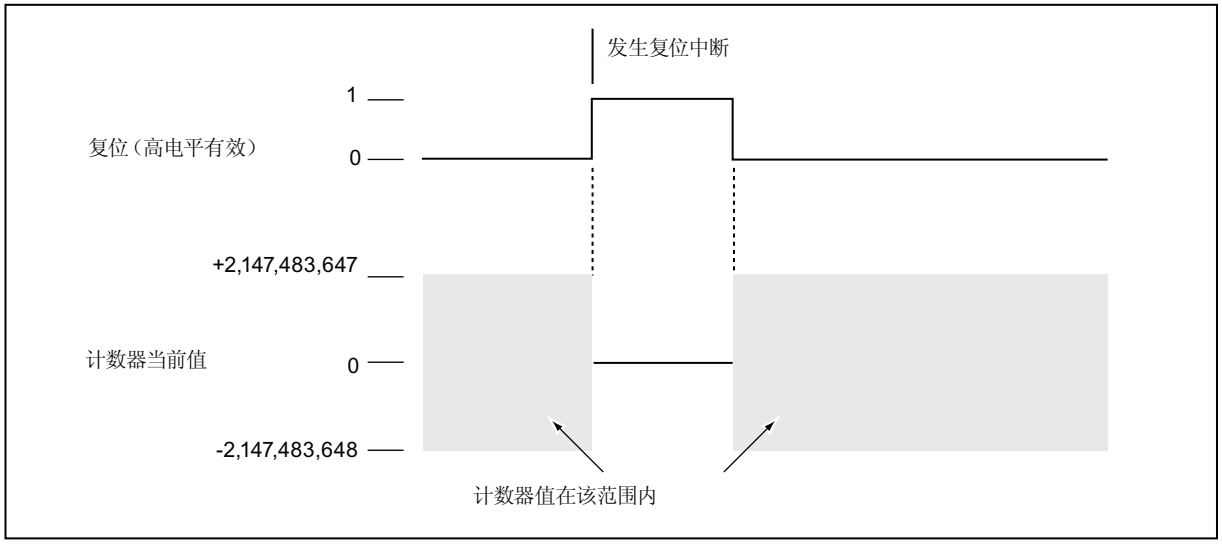
HSC 模式 9 和 10（AB 正交相位 4X）



复位操作

下图显示的复位操作适用于使用复位输入的所有模式。在下图中，显示的复位操作将有效状态分配为高位。

HSC 复位



HDEF 指令设置复位有效电平和计数速率

说明

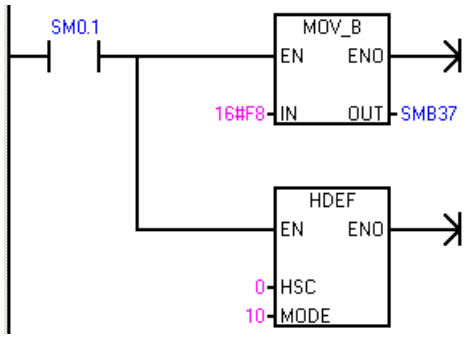
执行 HDEF 指令之前，必须将这两个控制位设置为所需状态。否则，计数器会采用所选计数器模式的默认组态。

执行 HDEF 指令后，将无法再更改计数器设置，除非首先将 CPU 设为 STOP 模式。

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7	描述（仅在执行 HDEF 时使用）
SM37.0	SM47.0	SM57.0	SM137.0	SM147.0	SM157.0	SM361.0	SM371.0	**复位的有效电平控制位：** * • 0 = 高电平激活时复位 • 1 = 低电平激活时复位
SM37.2	SM47.2	SM57.2	SM137.2	SM147.2	SM157.2	SM361.2	SM371.2	**AB 正交相计数器的计数速率选择：** * • 0 = 4X 计数速率 • 1 = 1X 计数速率

* 复位输入的默认设置为高电平有效，AB 正交相计数速率为 4x（或 4 倍输入时钟频率）。

示例：高速计数器定义

LAD			STL
MAIN		第一次扫描时： 1. 将复位输入设为高电平有效并选择 4x 模式。 2. 将 HSC0 组态为具有复位输入的 AB 正交相（模式 10）。	**Network 1** **LD SM0.1** **MOVB 16#F8, SMB37** **HDEF 0, 10**

使用 HSC 指令启用计数器、设置计数方向、载入预设值/当前计数值

HSC 指令在执行期间使用控制字节。分配计数器和计数器模式之后，即可对计数器的动态参数进行编程。每个高速计数器的 SM 存储器内均有一个控制字节，允许执行以下操作：

- 启用或禁用计数器
- 控制方向（仅限模式 0 和模式 1）
- 加载当前值
- 加载预设值

HSC 控制字节

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7	说明
SM37.3	SM47.3	SM57.3	SM137.3	SM147.3	SM157.3	SM361.3	SM371.3	计数方向控制位： • 0 = 减计数 • 1 = 加计数
SM37.4	SM47.4	SM57.4	SM137.4	SM147.4	SM157.4	SM361.4	SM371.4	向 HSC 写入计数方向： • 0 = 不更新 • 1 = 更新方向
SM37.5	SM47.5	SM57.5	SM137.5	SM147.5	SM157.5	SM361.5	SM371.5	向 HSC 写入新预设值： • 0 = 不更新 • 1 = 更新预设值

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7	说明
SM37.6	SM47.6	SM57.6	SM137.6	SM147.6	SM157.6	SM361.6	SM371.6	向 HSC 写入新当前值： • 0 = 不更新 • 1 = 更新当前值
SM37.7	SM47.7	SM57.7	SM137.7	SM147.7	SM157.7	SM361.7	SM371.7	启用 HSC： • 0 = 禁用 HSC • 1 = 启用 HSC
SMB195	SMB197	SMB199	SMB201	SMB203	SMB205	SMB207	SMB209	HSC 类型： • 0 = 计数模式 • 1 = 频率测量模式
SMB196	SMB198	SMB200	SMB202	SMB204	SMB206	SMB208	SMB210	HSC 频率测量采样周期，工作在频率测量模式下： • 0 = 采样周期 1 s • 1 = 采样周期 100 ms • 2 = 采样周期 10 ms

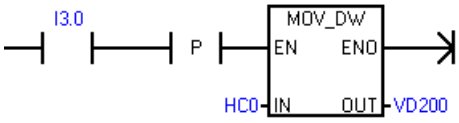
使用程序读取 HSC 当前值

只能使用后面带有计数器标识符编号（0、1、2、3、4、5、6 或 7）的数据类型 HC（高速计数器当前值）读取每个高速计数器的当前值，如下表所示。无论何时想要读取当前值，都可以在状态图表或用户程序中使用 HC 数据类型。HC 数据类型为只读双字值；不能使用 HC 数据类型将新的当前计数值写入高速计数器。

HSC0、HSC1、HSC2、HSC3、HSC4、HSC5、HSC6 和 HSC7 的当前值

要读取的值	HSC0 地址	HSC1 地址	HSC2 地址	HSC3 地址	HSC4 地址	HSC5 地址	HSC6 地址	HSC7 地址
CV（计数器当前值）	HC0	HC1	HC2	HC3	HC4	HC5	HC6	HC7

示例：读取并保存当前计数值

LAD			STL
MAIN		当 I3.0 从 OFF 转换为 ON 时，将 HSC0 的值保存到 VD200 中。	**Network 1** **LD I3.0** **EU** **MOVD HC0, VD200**

使用程序设置当前值和预设值

每个高速计数器内部都存储着一个 32 位当前值 (CV) 和一个 32 位预设值 (PV)。当前值是计数器的实际计数值，而预设值是当前值达到预设值时选择用于触发中断的比较值。可以按照上一部分所述使用 HC 数据类型读取当前值。不能直接读取预设值。要将新的当前值或预设值载入高速计数器，必须对控制字节以及保存所需新当前值和/或新预设值的特殊存储器双字进行设置，同时，必须执行 HSC 指令将新值传送到高速计数器中。下表列出了用于保存所需新当前值和预设值的特殊存储器双字。

使用以下步骤将新当前值和/或新预设值写入高速计数器（可按任一顺序执行步骤 1 和 2）：

1. 加载要写入相应 SM 新当前值和/或新预设值的值（请参见下表）。加载这些新值尚不会影响高速计数器。
2. 设置或清除相应控制字节的相应位，指示是否更新当前值和/或预设值（位 x.5 代表预设值，位 x.6 代表当前值）。调节这些位尚不会影响高速计数器。
3. 执行引用相应高速计数器编号的 HSC 指令。执行该指令可检查控制字节。如果控制字节指定更新当前值、预设值或两者，则会将相应值从 SM 新当前值和/或新预设值位置复制到高速计数器内部寄存器中。

要加载的值	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7
新当前值（新 CV）	SMD38	SMD48	SMD58	SMD138	SMD148	SMD158	SMD362	SMD372
新预设值（新 PV）	SMD42	SMD52	SMD62	SMD142	SMD152	SMD162	SMD366	SMD376

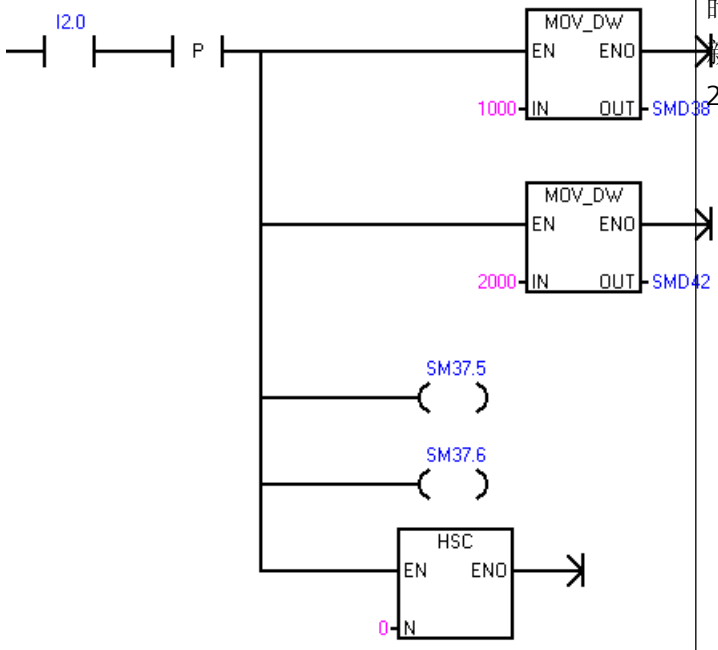
说明

执行相应的 HSC 指令前，更改新当前值和新预设值的控制字节和 SM 位置不会影响高速计数器。

示例：更新当前值和预设值

LAD		STL

MAIN 程序段	当 I2.0 从关断转换为接通时，HSC0 的当前计数值更新为 1000，预设值更新为 2000。	Network 1
		LD I2.0
		EU
		MOVD 1000, SMD38
		MOVD 2000, SMD42
		= SM37.5
		= SM37.6
		HSC 0



在程序中附加 HSC 中断例程

当 HSC 的当前值等于加载的预设值时，所有高速计数器模式都支持中断事件。使用外部复位输入的计数器模式支持在激活外部复位时中断。除模式 0 和模式 1 以外的所有计数器模式均支持计数方向改变时中断。可单独启用或禁用这些中断条件。有关中断使用的完整讨论，请参见中断指令 (页 413)部分。

HSC 状态字节

每个高速计数器的状态字节提供状态存储器位，用于指示当前计数方向以及当前值是否大于或等于预设值。下表定义了每个高速计数器的这些状态位。

说明

只有在执行高速计数器中断例程时，状态位才有效。监视高速计数器状态的目的在于启用对正在执行的操作有重大影响的事件的中断程序。

表格 9-14　HSC0、HSC1、HSC2、HSC3、HSC4、HSC5、HSC6 和 HSC7 的状态位

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7	说明
SM36.5	SM46.5	SM56.5	SM136.5	SM146.5	SM156.5	SM360.5	SM370.5	**当前计数方向状态位：** • 0 = 减计数 • 1 = 加计数
SM36.6	SM46.6	SM56.6	SM136.6	SM146.6	SM156.6	SM360.6	SM370.6	**当前值等于预设值状态位：** • 0 = 不相等 • 1 = 相等
SM36.7	SM46.7	SM56.7	SM136.7	SM146.7	SM156.7	SM360.7	SM370.7	**当前值大于预设值状态位：** • 0 = 小于或等于 • 1 = 大于

引用信息

更多信息，请参考以下部分：

- 高速计数器指令 (页 356)
- 高速计数器汇总 (页 358)
- 高速计数器的初始化顺序示例 (页 371)

9.6.7 高速计数模式的初始化顺序示例

HSC0 在以下初始化和操作顺序说明中被用作计数器。

初始化说明假设刚刚将 CPU 置于 RUN 模式，因此首次扫描存储器位为真。如果不是如此，请记住在进入 RUN 模式后，只能为每台高速计数器执行一次 HDEF 指令。为高速计数器第二次执行 HDEF 会生成运行时错误，并不会更改该计数器首次执行 HDEF 时计数器的设置方式。

说明

虽然以下顺序分别显示如何更改方向、当前值和预设值，但您可以按照相同的顺序更改所有数值或这些数值的任何组合，方法是相应设置 SMB37 的值，然后执行 HSC0 指令。

初始化模式 0 和 1

下列步骤说明如何为带内部方向的单相向上/向下计数器（模式 0 和 1）初始化 HSC0：

1. 使用首次扫描存储器位调用执行初始化操作的子例程。由于使用子例程调用，后续扫描不再调用子例程，因此可减少扫描执行时间并使程序结构更加合理。
2. 在初始化子例程中，根据所需的控制操作加载 SMB195 和 SMB37。
   例如：SMB195=0 表示计数模式
   SMB37 = 16#F8 产生如下结果：
   – 启用计数器
   – 写入新当前值
   – 写入新预设值
   – 将方向设置为加计数
   – 将复位输入设为高电平有效
3. 将 HSC 输入设 0 且 MODE 输入设为下列值之一后执行 HDEF 指令：
   – 模式 0 表示无外部复位
   – 模式 1 表示有外部复位
4. 用所需当前值加载 SMD38（双字大小值）（加载 0 可进行清除）。
5. 用所需预设值加载 SMD42（双字大小值）。
6. 为捕获当前值等于预设值事件，将 CV = PV 中断事件（事件 12）附加于中断例程，编程中断。有关中断处理的完整详细信息，请参见讨论中断指令的部分。
7. 为捕获外部复位事件，将外部复位中断事件（事件 28）附加于中断例程，编程中断。
8. 执行全局中断启用指令 (ENI) 以启用中断。
9. 执行 HSC 指令，使 CPU 对 HSC0 编程。
10. 退出子例程。

初始化模式 3 和 4

下列步骤说明如何为带外部方向控制的单相向上/向下计数器（模式 3 和 4）初始化 HSC0：

1. 使用首次扫描存储器位调用执行初始化操作的子例程。由于使用子例程调用，后续扫描不再调用子例程，因此可减少扫描执行时间并使程序结构更加合理。
2. 在初始化子例程中，根据所需的控制操作加载 SMB195 和 SMB37。
   例如：SMB195=0 表示计数模式
   SMB37 = 16#F8 产生如下结果：
   – 启用计数器
   – 写入新当前值
   – 写入新预设值
   – 将 HSC 的初始方向设置为向上计数
   – 将复位输入设为高电平有效

3. 将 HSC 输入设 0 且 MODE 输入设为下列值之一后执行 HDEF 指令：
   – 模式 3 表示无外部复位
   – 模式 4 表示有外部复位
4. 用所需当前值加载 SMD38（双字大小值）（加载 0 可进行清除）。
5. 用所需预设值加载 SMD42（双字大小值）。
6. 为捕获当前值等于预设值事件，将 CV = PV 中断事件（事件 12）附加于中断例程，编程中断。有关中断处理的完整详细信息，请参见讨论中断指令的部分。
7. 为捕获方向更改，将方向更改中断事件（事件 27）附加于中断例程中，编程中断。
8. 为捕获外部复位事件，将外部复位中断事件（事件 28）附加于中断例程，编程中断。
9. 执行全局中断启用指令 (ENI) 以启用中断。
10. 执行 HSC 指令，使 CPU 对 HSC0 编程。
11. 退出子例程。

初始化模式 6 和 7

下列步骤说明如何为带加/减时钟的双相向上/向下计数器（模式 6 和 7）初始化 HSC0：

1. 使用首次扫描存储器位调用执行初始化操作的子例程。由于使用子例程调用，后续扫描不再调用子例程，因此可减少扫描执行时间并使程序结构更加合理。
2. 在初始化子例程中，根据所需的控制操作加载 SMB195 和 SMB37。
   例如：SMB195=0 表示计数模式
   SMB37 = 16#F8 产生如下结果：
   – 启用计数器
   – 写入新当前值
   – 写入新预设值
   – 将 HSC 的初始方向设置为向上计数
   – 将复位输入设为高电平有效
3. 将 HSC 输入设 0 且 MODE 设为下列值之一后执行 HDEF 指令：
   – 模式 6 表示无外部复位
   – 模式 7 表示有外部复位
4. 用所需当前值加载 SMD38（双字大小值）（加载 0 可进行清除）。
5. 用所需预设值加载 SMD42（双字大小值）。
6. 为捕获当前值等于预设值事件，将 CV = PV 中断事件（事件 12）附加于中断例程，编程中断。请参见关于中断的部分。
7. 为捕获方向更改，将方向更改中断事件（事件 27）附加于中断例程中，编程中断。
8. 为捕获外部复位事件，将外部复位中断事件（事件 28）附加于中断例程，编程中断。
9. 执行全局中断启用指令 (ENI) 以启用中断。
10. 执行 HSC 指令，使 CPU 对 HSC0 编程。
11. 退出子例程。

初始化模式 9 和 10

以下步骤介绍如何将 HSC0 初始化为 AB 正交相计数器（针对模式 9 和 10）：

1. 使用首次扫描存储器位调用执行初始化操作的子例程。由于使用子例程调用，后续扫描不再调用子例程，因此可减少扫描执行时间并使程序结构更加合理。
2. 在初始化子例程中，根据所需的控制操作加载 SMB195 和 SMB37。
   例如（1x 计数模式）：SMB195=0 表示计数模式
   SMB37 = 16#FC 产生如下结果：
   – 启用计数器
   – 写入新当前值
   – 写入新预设值
   – 将 HSC 的初始方向设置为向上计数
   – 将复位输入设为高电平有效
   示例（4x 计数模式）：SMB37 = 16#F8 产生如下结果：
   – 启用计数器
   – 写入新当前值
   – 写入新预设值
   – 将 HSC 的初始方向设置为向上计数
   – 将复位输入设为高电平有效
3. 将 HSC 输入设 0 且 MODE 输入设为下列值之一后执行 HDEF 指令：
   – 模式 9 表示无外部复位
   – 模式 10 表示有外部复位
4. 用所需当前值加载 SMD38（双字大小值）（加载 0 可进行清除）。
5. 用所需预设值加载 SMD42（双字大小值）。
6. 为捕获当前值等于预设值事件，将 CV = PV 中断事件（事件 12）附加于中断例程，编程中断。有关中断处理的完整详细信息，请参见启用中断 (ENI) 的相关部分。
7. 为捕获方向更改，将方向更改中断事件（事件 27）附加于中断例程中，编程中断。
8. 为捕获外部复位事件，将外部复位中断事件（事件 28）附加于中断例程，编程中断。
9. 执行全局中断启用指令 (ENI) 以启用中断。
10. 执行 HSC 指令，使 CPU 对 HSC0 编程。
11. 退出子例程。

更改模式 0 和 1 的方向

下列步骤说明如何组态 HSC0，以更改带内部方向的单相计数器（模式 0 和 1）的方向：

1. 加载 SMB37，以写入所需方向：
   SMB37 = 16#90
   – 启用计数器
   – 将 HSC 的方向设置为减计数
   SMB37 = 16#98
   – 启用计数器
   – 将 HSC 的方向设置为加计数
2. 执行 HSC 指令，使 CPU 对 HSC0 编程。

加载新当前值（任何模式）

以下步骤介绍了如何更改 HSC0 的计数器当前值（任何模式）：

1. 加载 SMB37，以写入所需当前值：
   SMB37 = 16#C0
   – 启用计数器
   – 写入新当前值
2. 用所需当前值加载 SMD38（双字大小值）（加载 0 可进行清除）。
3. 执行 HSC 指令，使 CPU 对 HSC0 编程。

加载新预设值（任何模式）

以下步骤介绍了如何更改 HSC0 的预设值（任何模式）：

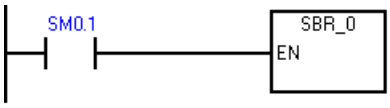
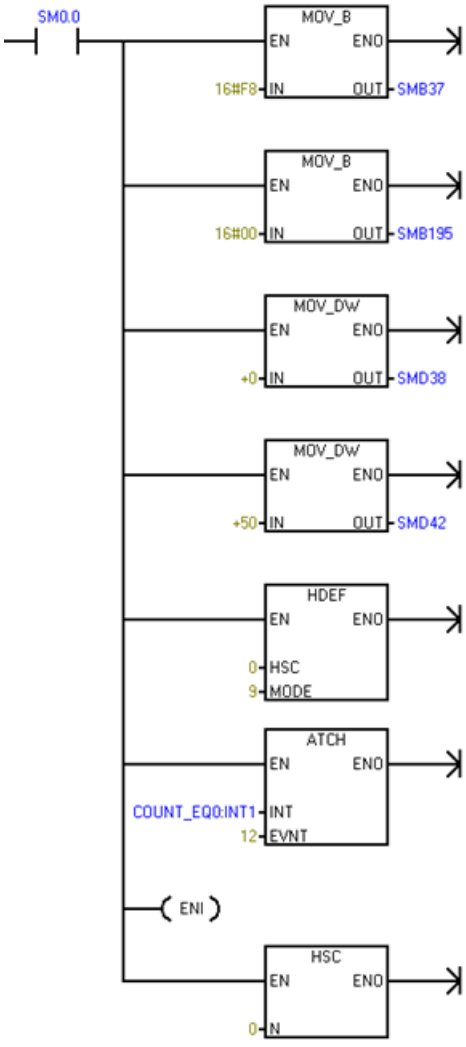
1. 加载 SMB37，以写入所需预设值：
   SMB37 = 16#A0
   – 启用计数器
   – 写入新预设值
2. 用所需预设值加载 SMD42（双字大小值）。
3. 执行 HSC 指令，使 CPU 对 HSC0 编程。

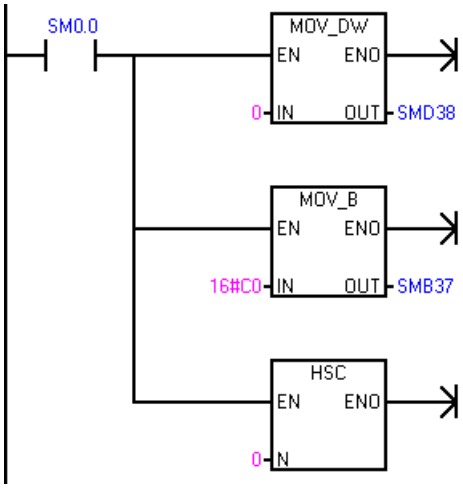
禁用高速计数器（任何模式）

以下步骤介绍了如何禁用 HSC0 高速计数器（任何模式）：

1. 加载 SMB37，以禁用计数器：
   SMB37 = 16#00
   – 禁用计数器
2. 执行 HSC 指令，以禁用计数器。

示例：高速计数器指令

LAD			STL
MAIN		首次扫描时，调用 SBR_0。	**Network 1** **LD SM0.1** **CALL SBR_0**
SBR0		首次扫描时，组态 HSC0： 1. 启用计数器 – 写入新当前值。 – 写入新预设值。 – 将初始方向设置为加计数。 – 选择复位输入高电平有效。 – 选择 4x 模式。 2. 将 HSC0 组态为具有复位输入的 AB 正交相。 3. 清除 HSC0 的当前值。 4. 将 HSC0 预设值设置为 50。 5. 将事件 12 附加到中断例程 INT_0。在 HSC0 当前值＝预设值时将执行该中断 6. 全局中断启用 7. 组态 HSC0。	**Network 1** **LD SM0.0** **MOVB 16#F8, SMB37** **MOVB 16#00, SMB195** **MOVD +0, SMD38** **MOVD +50, SMD42** **HDEF 0, 9** **ATCH** **COUNT_EQ0:INT1, 12** **ENI** **HSC 0**

LAD			STL
INT0		对 HSC0 编程： 1. 清除 HSC0 的当前值。 2. 选择仅写入新当前值，将 HSC0 保持为启用状态。 3. 组态 HSC0。	**Network 1** **LD SM0.0** **MOVD +0, SMD38** **MOVB 16#C0, SMB37** **HSC 0**

引用信息

更多信息，请参考以下部分：

- 高速计数器指令 (页 356)
- 高速计数器汇总 (页 358)
- 高速计数器编程 (页 360)
- 中断指令 (页 413)

9.6.8 高速频率模式的初始化顺序示例

初始化模式 0

下列步骤说明如何为带内部方向的单相向上/向下频率计（模式 0）初始化 HSC0：

1. 使用首次扫描存储器位调用执行初始化操作的子例程。由于使用子例程调用，后续扫描不再调用子例程，因此可减少扫描执行时间并使程序结构更加合理。
2. 在初始化子例程中，根据所需的控制操作加载 SMB195 和 SMB37。
   例如：SMB195 = 1 表示频率测量模式
   SMB37 = 16#F8 产生如下结果：
   – 启用计数器
   – 将方向设置为加计数
3. 在将 HSC 输入设为 0 且 MODE 输入设为 0 后执行 HDEF 指令，无外部复位。
4. 执行 HSC 指令，使 CPU 对 HSC0 编程。
5. 退出子例程。

初始化模式 3

下列步骤说明如何为带外部方向控制的单相向上/向下计数器（模式 3）初始化 HSC0：

1. 使用首次扫描存储器位调用执行初始化操作的子例程。由于使用子例程调用，后续扫描不再调用子例程，因此可减少扫描执行时间并使程序结构更加合理。
2. 在初始化子例程中，根据所需的控制操作加载 SMB195 和 SMB37。
   例如：SMB195 = 1 表示频率测量模式
   SMB37 = 16#F8 产生如下结果：
   – 启用计数器
3. 在将 HSC 输入设为 0 且 MODE 输入设为 3 后执行 HDEF 指令，无外部复位。
4. 为捕获方向更改，将方向更改中断事件（事件 27）附加于中断例程中，编程中断。
5. 执行全局中断启用指令 (ENI) 以启用中断。
6. 执行 HSC 指令，使 CPU 对 HSC0 编程。
7. 退出子例程。

初始化模式 6

下列步骤说明如何为带加/减时钟的双相向上/向下计数器（模式 6）初始化 HSC0：

1. 使用首次扫描存储器位调用执行初始化操作的子例程。由于使用子例程调用，后续扫描不再调用子例程，因此可减少扫描执行时间并使程序结构更加合理。
2. 在初始化子例程中，根据所需的控制操作加载 SMB195 和 SMB37。
   例如：SMB195 = 1 表示频率测量模式
   SMB37 = 16#F8 产生如下结果：
   – 启用计数器
3. 在将 HSC 输入设为 0 且 MODE 设为 6 后执行 HDEF 指令，无外部复位。
4. 为捕获方向更改，将方向更改中断事件（事件 27）附加于中断例程中，编程中断。
5. 执行全局中断启用指令 (ENI) 以启用中断。
6. 执行 HSC 指令，使 CPU 对 HSC0 编程。
7. 退出子例程。

初始化模式 9

以下步骤介绍如何将 HSC0 初始化为 AB 正交相计数器（模式 9）：

1. 使用首次扫描存储器位调用执行初始化操作的子例程。由于使用子例程调用，后续扫描不再调用子例程，因此可减少扫描执行时间并使程序结构更加合理。
2. 在初始化子例程中，根据所需的控制操作加载 SMB195 和 SMB37。
   例如：SMB195 = 1 表示频率测量模式
   SMB37 = 16#FC 产生如下结果：
   – 启用计数器
   示例（4x 计数模式）：SMB37 = 16#F8 产生如下结果：
   – 启用计数器
3. 在将 HSC 输入设为 0 且 MODE 输入设为 9 后执行 HDEF 指令，无外部复位。
4. 为捕获方向更改，将方向更改中断事件（事件 27）附加于中断例程中，编程中断。
5. 执行全局中断启用指令 (ENI) 以启用中断。
6. 执行 HSC 指令，使 CPU 对 HSC0 编程。
7. 退出子例程。

更改模式 0 的方向

下列步骤说明如何组态 HSC0，以更改带内部方向的单相计数器（模式 0）的方向：

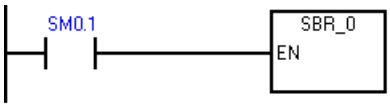
1. 加载 SMB37，以写入所需方向：
   SMB37 = 16#90
   – 启用计数器
   – 将 HSC 的方向设置为减计数
   SMB37 = 16#98
   – 启用计数器
   – 将 HSC 的方向设置为加计数
2. 执行 HSC 指令，使 CPU 对 HSC0 编程。

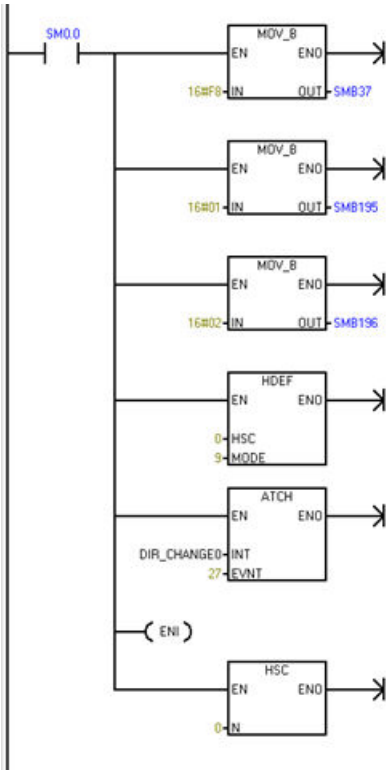
禁用高速计数器（任何模式）

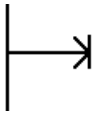
以下步骤介绍了如何禁用 HSC0 高速计数器（任何模式）：

1. 加载 SMB37，以禁用计数器：
   SMB37 = 16#00
   – 禁用计数器
2. 执行 HSC 指令，以禁用计数器。

示例：高速计数器指令

LAD			STL
MAIN		首次扫描时，调用 SBR_0。	程序段 1 LD SM0.1 CALL SBR_0

SBR0		首次扫描时，组态 HSC0：
		1. 启用计数器
		– 选择 4x 模式。
		2. 将模式类型设为“频率”(Frequency)。
		3. 将采样周期设为 10 ms。
		4. 将 HSC0 组态为具有复位输入的 AB 正交相。
		5. 将事件 27 附加到中断例程 DIR_CHANGE0。方向发生更改时执行中断。
		6. 中断启用
		7. 组态 HSC0。
			程序段 1
			LD SM0.0
			MOVB 16#F8, HSC0_Ctrl:SMB37
			MOVB 16#01, SMB195
			MOVB 16#02, SMB196
			HDEF 0, 9
			ATCH DIR_CHANGE0, 27
			ENI
			HSC 0

INT0		用户自定义	

引用信息

更多信息，请参考以下部分：

- 高速计数器指令 (页 356)
- 高速计数器汇总 (页 358)
- 高速计数器编程 (页 360)
- 中断指令 (页 413)

9.7 脉冲输出

9.7.1 脉冲输出指令 (PLS)

脉冲输出 (PLS) 指令控制脉冲串输出 (PTO) 和高速输出（Q0.0、Q0.1、Q0.3、Q0.2 和 Q0.4）上提供的脉宽调制 (PWM) 功能。

若使用 PWM，可通过可选向导来创建 PWM 指令。

LAD/FBD	STL	描述
PLS EN ENO N	PLS N	最多可以使用 PLS 指令创建五个 PTO 或 PWM 操作。PTO 允许用户控制方波（50% 占空比）输出的频率和脉冲数量。PWM 允许用户控制占空比可变的固定循环时间输出。

ENO = 0 时的错误条件	受影响的 SM 位
• 0005H：并行 HSC/PLS • 000DH：尝试在激活时重新定义脉冲输出 • 000EH：PTO 包络段的数量设置为 0 • 0017H：试图为已分配给运动控制的 PTO/PWM 分配资源 • 001BH：尝试在启用的 PWM 上更改时基 • 0090H：N 不为 0、1、2、3 或 4。 • 0091H：范围错误	无

输入/输出	数据类型	操作数
N（通道）	WORD	常数：0 (= Q0.0)、1 (= Q0.1)、2 (= Q0.3)、3 (=Q0.2) 或 4(=Q0.4)

CPU 有五个 PTO/PWM 发生器（PLS0、PLS1、PLS2、PLS3 和 PLS4），用于产生高速脉冲串或脉宽调制波形。PLS0 分配给数字量输出点 Q0.0，PLS1 分配给数字量输出点 Q0.1，PLS2 分配给数字量输出点 Q0.3，PLS3 分配给数字量输出点 Q0.2，PLS4 分配给数字量输出点 Q0.4。指定的特殊存储器 (SM) 单元用于存储每个发生器的以下数据：一个 PTO 状态字节（8 位值）、一个控制字节（8 位值）、一个循环时间或频率（16 位无符号值）、一个脉冲宽度值（16 位无符号值）以及一个脉冲计数值（32 位无符号值）。

PTO/PWM 发生器和过程映像寄存器共用 Q0.0、Q0.1、Q0.3、Q0.2 和 Q0.4。若在 Q0.0、Q0.1、Q0.3、Q0.2 和 Q0.4 上激活 PTO 或 PWM 功能，PTO/PWM 生成器将控制输出，从而禁止输出点的正常用法。输出波形不会受过程映像寄存器状态、输出点强制值或立即输出指令执行的影响。若未激活 PTO/PWM 生成器，则重新交由过程映像寄存器控制输出。过程映像寄存器决定输出波形的初始和最终状态，确定波形是以高电平还是低电平开始和结束。

说明

如果已通过运动控制向导将所选输出点组态为运动控制用途，则无法通过 PLS 指令激活 PTO/PWM。

PTO/PWM 输出的最低负载必须至少为额定负载的 10%，才能实现启用与禁用之间的顺利转换。

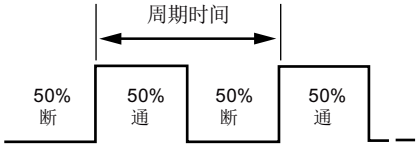
启用 PTO/PWM 操作前，应将 Q0.0、Q0.1、Q0.3、Q0.2 和 Q0.4 的过程映像寄存器的值设为 0。

所有控制位、循环时间/频率、脉冲宽度和脉冲计数值的默认值均为 0。

正在运行的 PLS 指令可由其他 PLS 指令超驰。触发超驰后，CPU 直接将速度或脉冲计数改为目标速度或目标脉冲计数，到达新位置。

9.7.2 脉冲串输出 (PTO)

PTO 以指定频率和指定脉冲数量提供 50% 占空比输出的方波。请参见下图。PTO 可使用脉冲包络生成一个或多个脉冲串。您可以指定脉冲的数量和频率。



- 脉冲数：1 到 2,147,483,647
- 频率：
  - 1 到 200,000 Hz（多段）
  - 1 到 65,535 Hz（单段）

使用以下公式将周期时间转换为频率：

F = 1 / CT

其中：

F	频率 (Hz)
CT	周期时间（秒）

请参见下表了解脉冲计数和频率极限：

表格 9-15　PTO 中的脉冲计数和频率

脉冲计数/频率	响应
频率 < 1 Hz	频率默认为 1 Hz
频率 > 200,000 Hz	频率默认为 200,000 Hz
脉冲计数 = 0	脉冲计数默认为 1 个脉冲
脉冲计数 > 2,147,483,647	脉冲计数默认为 2,147,483,647 个脉冲

说明

使用周期时间非常短的（高频率）PTO 时，应考虑输出点的开关延迟规范以及开关延迟对占空比的影响。有关 CPU 的数字量输出开关延迟的信息，请参见附录 A。

PTO 功能允许脉冲串“链接”或“管道化”。有效脉冲串结束后，新脉冲串的输出会立即开始。这样便可持续输出后续脉冲串。

PTO 功能允许单段“超驰”。新的脉冲串组态将超驰正在输出的脉冲串。

PTO 脉冲的单段管道化

在单段管道化中，您负责更新下一脉冲串的 SM 位置。在初始的 PTO 段开始后，您需要立即使用第二个波形的参数修改 SM 单元。SM 的相应值更新后，再次执行 PLS 指令。PTO 功能在管道中保留第二个脉冲串的属性，直到其完成了第一个脉冲串。PTO 功能在管道中一次只能存储一个条目。在第一个脉冲串完成时，开始输出第二个波形，然后可在管道中存储一个新脉冲串设置。之后可重复此过程，设置下一脉冲串的特性。若在管道仍然填满时试图装载新设置，将导致 PTO 溢出位（SM66.6、SM76.6、SM566.6 或 SM580.6）置位并且指令被忽略。

只有在当前有效的脉冲串在 PLS 指令捕获到新脉冲串设置之前完成，才能在脉冲串之间实现平滑转换。

说明

在单段管道化期间，频率的上限为 65,535 Hz。如果需要更高的频率（最高为 200,000 Hz），则必须使用多段管道化。

PTO 脉冲的多段管道化

在多段管道化期间，S7-200 SMART 从 V 存储器的包络表中自动读取每个脉冲串段的特性。该模式中使用的 SM 单元为控制字节、状态字节和包络表的起始 V 存储器（SMW168、SMW178、SMW578、SMW592 或 SMW2012）的偏移量。执行 PLS 指令将启动多段操作。

每段条目长 12 字节，由 32 位起始频率、32 位结束频率和 32 位脉冲计数值组成。下表给出了 V 存储器中组态的包络表的格式：

PTO 生成器会自动将频率从起始频率线性提高或降低到结束频率。频率以恒定速率提高或降低一个恒定值。在脉冲数量达到指定的脉冲计数时，立即装载下一个 PTO 段。该操作将一直重复到到达包络结束。段持续时间应大于 500 微秒。如果持续时间太短，CPU 可能没有足够的时间计算下一个 PTO 段值 。如果不能及时计算下一个段，则 PTO 管道下溢位（SM66.6、SM76.6、SM566.6、SM580.6 和 SM2000.6）被置“1”，且 PTO 操作终止。

在 PTO 包络作用期间，在 SMB166、SMB176、SMB576、SMB590 或 SMB2010 中提供当前有效段的编号。

表格 9-16　多段 PTO 操作的包络表格式 [1]

字节偏移	段	表格条目的描述
0		段数量：1 到 255[2]
1	#1	起始频率（1 到 200,000 Hz）
5		结束频率（1 到 200,000 Hz）
9		脉冲计数（1 到 2,147,483,647）
13	#2	起始频率（1 到 200,000 Hz）
17		结束频率（1 到 200,000 Hz）
21		脉冲计数（1 到 2,147,483,647）
（依此类推）	#3	（依此类推）

[1] 若输入将包络表的任何部分放到 V 存储器之外的包络偏移量和段数量，将生成非致命错误。该 PTO 功能将不生成 PTO 输出。

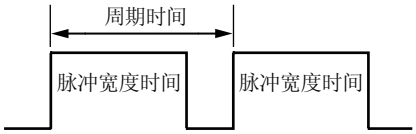
[2] 若段数量输入为 0 值，将生成非致命错误。此时，不会生成 PTO 输出

PTO 脉冲的单段超驰

在单段超驰中，您负责更新超驰脉冲串的 SM 位置。在初始的 PTO 段开始后，您需要立即使用第二个波形的参数修改 SM 单元。SM 的相应值更新后，在您需要的时刻再次执行 PLS 指令即可实现超驰。

9.7.3 脉宽调制 (PWM)

PWM 提供四条通道，这些通道允许占空比可变的固定周期时间输出。请参见下图。可以指定周期时间和脉冲宽度（以微秒或毫秒为增量）：



- 周期时间：5 μs 到 65,535 μs 或 2 ms 到 65,535 ms
- 脉冲宽度时间：0 μs 到 65,535 μs 或 0 ms 到 65,535 ms

如下表所示，将脉冲宽度设置为等于周期时间（占空比为 100%）会使输出一直接通。将脉冲宽度设置为 0（占空比为 0%）会使输出断开。

说明

使用周期时间非常短的 PWM 时，应考虑输出点的开关延迟规范以及开关延迟对脉冲宽度时间的影响。有关 CPU 的数字量输出开关延迟的信息，请参见附录 A。

脉冲宽度时间、周期时间和 PWM 功能的响应

脉冲宽度时间/周期时间	响应
脉冲宽度时间 >= 周期时间值	占空比为 100%：输出一直接通。
脉冲宽度时间 = 0	占空比为 0%：连续关闭输出。
周期时间 < 2 个时间单位	默认情况下，周期时间为两个时间单位。

更改 PWM 波形的特性

只能使用同步更新更改 PWM 波形的特性。执行同步更新时，信号波形特性的更改发生在周期交界处，这样可实现平滑转换。

9.7.4 使用 SM 位置组态和控制 PTO/PWM 操作

PLS 指令读取存储于指定 SM 存储单元的数据，并相应地编程 PTO/PWM 生成器。SMB67 控制 PTO0 或 PWM0，SMB77 控制 PTO1 或 PWM1，SMB567 控制 PTO2 或 PWM2，SMB581 控制 PTO3 或 PWM3，SMB2001 控制 PTO4 或 PWM4。“PTO/PWM 控制寄存器的 SM 单元”表（下面第一个表）介绍了用于控制 PTO/PWM 操作的寄存器。可快速参考“PTO/PWM 控制字节参考”表（下面第二个表）来确定在 PTO/PWM 控制寄存器中放置什么值才能调用想要的操作。

可通过修改 SM 区域（包括控制字节）中的单元，然后执行 PLS 指令，来改变 PTO 或者 PWM 波形的特性。通过在控制字节（SM67.7、SM77.7、SM567.7、SM581.7 或 SM2001.7）的 PTO/PWM 使能位中写入 0 后执行 PLS 指令，可随时禁用生成 PTO 或 PWM 波形。输出点将立即恢复为过程映像寄存器控制。

如果在 PTO 或 PWM 操作正在产生脉冲时被禁止，该脉冲将内在地完成其整个节拍时间。但是，该脉冲不会出现在输出端，因为此时过程映像寄存器重新获得了对输出的控制。只要以下条件为真，您的程序可再次无延迟地启动脉冲发生器。启用与禁用的脉冲模式（PTO 或 PWM）相同。以下情况会导致错误发生：您的程序首先禁用了 PTO，然后又在同一输出通道启用了 PWM，或者您的程序首先禁用了 PWM，然后又启用了 PTO。

状态字节（SM66.7、SM76.7、SM566.7、SM580.7 或 SM2006.7）中的 PTO 空闲位可用来指示编程的脉冲串是否已结束。另外，中断例程可在脉冲串结束后进行调用。（请参见“中断指令 (页 413)”的介绍。）如果是使用单段操作，则在每个 PTO 结束时调用中断例程。例如，如果第二个 PTO 已装载到管道中，PTO 功能在第一个 PTO 结束时调用中断例程，然后在已装载到管道中第二个 PTO 结束时再次调用。若使用多段操作，PTO 功能在包络表完成时调用中断例程。

下列条件将设置状态字节（SMB66、SMB76、SMB566、SMB580 和 SMB2006）的位：

- 如果在导致无效频率值的脉冲生成器中发生“添加错误”，PTO 功能将终止，并且增量计算错误位（SM66.4、SM76.4、SM566.4、SM580.4 和 SM2006.4）将置 1。输出恢复为映像寄存器控制。要纠正该问题，请尝试调整 PTO 包络参数。
- 若手动禁止进行中的 PTO 包络，则 PTO 包络禁用位（SM66.5、SM76.5、SM566.5、SM580.5 或 SM2006.5）置 1。
- 如果以下任一情况发生，PTO/PWM 上溢/下溢位（SM66.6、SM76.6、SM566.6、SM580.6 或 SM2006.6）将置 1：
  - 当管道已满时试图装载管道；这是溢出条件。
  - PTO 包络段太短而导致 CPU 无法计算下一段，以及传送了空管道；这是下溢条件，且输出将恢复为映象寄存器控制。
- 在 PTO/PWM 溢出/下溢位置位后，必须手动将其清零才能检测到后续的溢出事件。切换到 RUN 模式可将该位初始化为 0。

说明

- 确保了解 PTO/PWM 模式选择位（SM67.6、SM77.6、SM567.6、SM581.6 和 SMB2000.6）的定义。该位定义可能与支持脉冲指令的早期产品有所不同。在 S7-200 SMART 中，用户可通过以下定义来选择 PTO 或 PWM 模式：0 = PWM，1 = PTO。
- 当装载节拍时间/频率（SMW68、SMW78、SMW568、SMW582 或 SMW2002）、脉冲宽度（SMW70、SMW80、SMW570、SMW584 或 SMW2004）或脉冲计数（SMD72、SMD82、SMD572、SMD586 或 SMD2006）时，在执行 PLS 指令之前也要设置控制寄存器中相应的更新位。
- 对于多段脉冲串操作，在执行 PLS 指令之前也必须装载包络表的起始偏移量（SMW168、SMW178、SMW578、SMW592 或 SMW2012）和包络表值。
- 如果在 PWM 在执行过程中试图改变 PWM 的时基，则该请求被忽略并产生非致命错误 (0x001B - ILLEGAL PWM TIMEBASE CHG)。

表格 9-17　PTO/PWM 控制寄存器的 SM 单元

Q0.0	Q0.1	Q0.3	Q0.2	Q0.4	状态位
SM66.4	SM76.4	SM566.4	SM580.4	SM2000.4	PTO 增量计算错误（因添加错误导致） • 0 = 无错误 • 1 = 因错误而中止
SM66.5	SM76.5	SM566.5	SM580.5	SM2000.5	PTO 包络被禁用（因用户指令导致）： • 0 = 非手动禁用的包络 • 1 = 用户禁用的包络
SM66.6	SM76.6	SM566.6	SM580.6	SM2000.6	PTO/PWM 管道溢出/下溢： • 0 = 无溢出/下溢 • 1 = 溢出/下溢
SM66.7	SM76.7	SM566.7	SM580.7	SM2000.7	PTO 空闲： • 0 = 进行中 • 1 = PTO 空闲
Q0.0	**Q0.1**	**Q0.3**	**Q0.2**	**Q0.4**	**控制位**
SM67.0	SM77.0	SM567.0	SM581.0	SM2001.0	PTO/PWM 更新频率/节拍时间： • 0 = 不更新 • 1 = 更新频率/节拍时间
SM67.1	SM77.1	SM567.1	SM581.1	SM2001.1	PWM 更新脉冲宽度时间： • 0 = 不更新 • 1 = 更新脉冲宽度

SM67.2	SM77.2	SM567.2	SM581.2	SM2001.2	PTO 更新脉冲计数值： • 0 = 不更新 • 1 = 更新脉冲计数
SM67.3	SM77.3	SM567.3	SM581.3	SM2001.3	PWM 时基： • 0 = 1 μs/时标 • 1 = 1 ms/时标
SM67.4	SM77.4	SM567.4	SM581.4	SM2001.4	PTO 超驰响应： • 0 = 禁用 • 1 = 启用
SM67.5	SM77.5	SM567.5	SM581.5	SM2001.5	PTO 单/多段操作： • 0 = 单段 • 1 = 多段
SM67.6	SM77.6	SM567.6	SM581.6	SM2001.6	PTO/PWM 模式选择： • 0 = PWM • 1 = PTO
SM67.7	SM77.7	SM567.7	SM581.7	SM2001.7	PWM 使能： • 0 = 禁用 • 1 = 启用
Q0.0	**Q0.1**	**Q0.3**	**Q0.2**	**Q0.4**	**其他寄存器**
SMW68	SMW78	SMW568	SMW582	SMW2002	PTO 频率或 PWM 节拍时间值：1 到 65,535 Hz (PTO)，2 到 65,535 (PWM)
SMW70	SMW80	SMW570	SMW584	SMW2004	PWM 脉冲宽度值：0 到 65,535
SMD72	SMD82	SMD572	SMD586	SMD2006	PTO 脉冲计数值：1 到 2,147,483,647
SMB166	SMB176	SMB576	SMB590	SMB2010	进行中段的编号： 仅限多段 PTO 操作
SMW168	SMW178	SMW578	SMW592	SMW2012	包络表的起始单元（相对 V0 的字节偏移）： 仅限多段 PTO 操作

表格 9-18　PTO/PWM 控制字节参考

		PLS 指令的执行结果						
控制寄存器（十六进制值）	启用	选择模式	PTO 段操作	超驰	时基	脉冲计数	脉冲宽度	节拍时间/频率
16#80	是	PWM		否	1 μs/周期			
16#81	是	PWM		否	1 μs/周期			更新节拍时间
16#82	是	PWM		否	1 μs/周期		更新	
16#83	是	PWM		否	1 μs/周期		更新	更新节拍时间
16#88	是	PWM		否	1 ms/周期			
16#89	是	PWM		否	1 ms/周期			更新节拍时间
16#8A	是	PWM		否	1 ms/周期		更新	
16#8B	是	PWM		否	1 ms/周期		更新	更新节拍时间
16#C0	是	PTO	单段	否				
16#C1	是	PTO	单段	否				更新频率
16#C4	是	PTO	单段	否		更新		
16#C5	是	PTO	单段	否		更新		更新频率
16#D0	是	PTO	单段	是				
16#D1	是	PTO	单段	是				更新频率
16#D4	是	PTO	单段	是		更新		
16#D5	是	PTO	单段	是		更新		更新频率
16#E0	是	PTO	多段	否				

9.7.5 计算包络表值

PTO 生成器的多段管道化功能对于许多应用（特别是步进电机控制）都很实用。

例如，可使用带有脉冲包络的 PTO 通过简单的斜升（加速）、运行（不加速）和斜降（减速）顺序来控制步进电机。通过定义脉冲包络可创建更复杂的顺序，脉冲包络最多可由 255 段组成，每段对应一个斜升、运行或斜降操作。

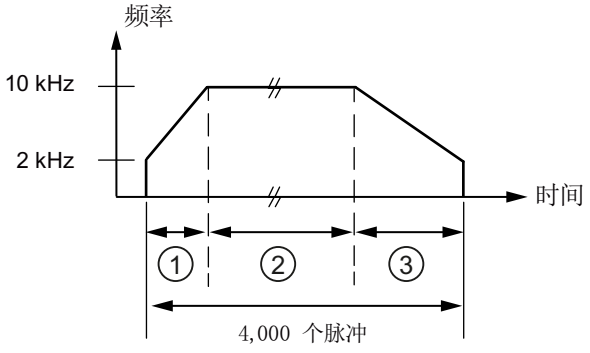
下图说明了生成输出波形所需的采样包络表值：

- 段 1：加速步进电机
- 段 2：以恒定转速运行电机
- 段 3：使电机减速

在本例中，要达到期望的电机转数，PTO 生成器需要以下值：

- 2 kHz 的启动和结束脉冲频率
- 10 kHz 的最大脉冲频率
- 4000 个脉冲

在输出包络的加速部分，大约在 200 脉冲后，输出波形应达到最大脉冲频率。大约在 400 脉冲后，输出波形应完成包络的减速部分。



① 段 1：200 个脉冲

② 段 2：3400 个脉冲

③ 段 3：400 个脉冲

下表列出了用于生成示例波形的值。本例中，包络表位于 V 存储器，起始地址为 VB500。可以使用任意可用于 PTO 包络表的 V 存储器区域。可以在程序中使用指令将这些值装载到 V 存储器中，也可在数据块中定义包络值。

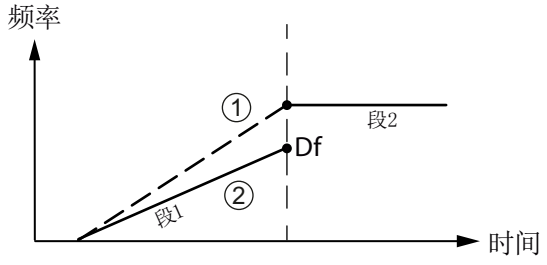
表格 9-19　包络表值

地址	值	说明	
VB500	3	总段数	
VD501	2,000	起始频率 (Hz)	分段 1
VD505	10,000	结束频率 (Hz)	
VD509	200	脉冲数	
VD513	10,000	起始频率 (Hz)	分段 2
VD517	10,000	结束频率 (Hz)	
VD521	3,400	脉冲数	
VD525	10,000	起始频率 (Hz)	分段 3
VD529	2,000	结束频率 (Hz)	
VD533	400	脉冲数	

PTO 生成器开始时先运行段 1。PTO 生成器达到段 1 所需脉冲数后，会自动装载段 2。该操作将持续到最后一段。达到最后一段的脉冲数后，S7-200 SMART CPU 将禁用 PTO 生成器。

对于 PTO 包络的每一段，脉冲串以表中分配的起始频率开始。PTO 生成器以恒定速率提高或降低频率，从而以正确的脉冲数达到结束频率。但是，PTO 生成器将工作频率限制为表中指定的启动和结束频率。

PTO 生成器逐步叠加工作频率，从而使频率随时间呈线性变化。叠加到频率的恒定值的分辨率受到限制。该分辨率限制会在产生的频率中引入截断误差。因此，PTO 生成器无法保证脉冲串频率可以到达为段指定的结束频率。在下图中，可以看到截断误差会影响 PTO 加速频率。应该测量输出，确定该频率是否在可接受的频率范围内。



① 期望的频率曲线图

② 实际的频率曲线图

如果一段结束和下一段开始的频率差 (Δf) 是不可接受，请尝试调整结束频率来对该差值进行补偿。为使得输出位于可接受的频率范围内，可能需要反复进行这种调整。

注意，段参数改变会影响 PTO 完成的时间。可以使用段的持续时间等式（在下文介绍）来了解其对时间的影响。对于给定的段来说，要想获得准确的段持续时间，结束频率值或脉冲数必须具备一定的弹性。

上面的简化示例用于介绍目的，实际应用可能需要更复杂的波形包络。别忘了您只能分配整数形式的 Hz 频率，且必须以恒定速率执行频率更改。S7-200 SMART CPU 可选择该恒定速率，且每一段的恒定速率可以不同。

对于依照周期时间（而非频率）开发的传统项目，可以使用以下公式来进行频率转换：

$$CT_{Final} = CT_{Initial} + (\Delta CT * PC)$$

$$F_{Initial} = 1 / CT_{Initial}$$

$$F_{Final} = 1 / CT_{Final}$$

其中：

$CT_{Initial}$	段启动周期时间 (s)
ΔCT	段增量周期时间 (s)
PC	段内脉冲数量
CT_{Final}	段结束周期时间 (s)
$F_{Initial}$	段起始频率 (Hz)
F_{Final}	段结束频率 (Hz)

给定 PTO 包络段的加速度（或减速度）和持续时间有助于确定正确的包络表值。使用以下公式可计算给定包络的持续时间和加速度：

$$\Delta F = F_{Final} - F_{Initial}$$

$$T_s = PC / (F_{min} + (| \Delta F | / 2))$$

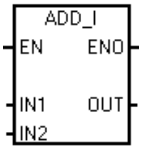
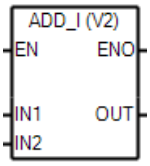
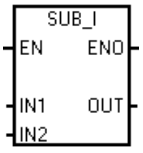
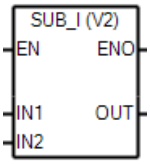
$$A_s = \Delta F / T_s$$

其中：

T_s	段持续时间 (s)
A_s	段频率加速度 (Hz/s)
PC	段内脉冲数量
F_{min}	段最小频率 (Hz)
ΔF	段增量（总变化）频率 (Hz)

9.8 数学运算

9.8.1 加法、减法、乘法和除法

LAD/FBD	STL	说明
 ADD_DI ADD_R	**+I IN1, IN2 OUT** **+D IN1, IN2 OUT** **+R IN1, IN2 OUT**	加整数指令将两个 16 位整数相加，产生一个 16 位结果。加双精度整数指令将两个 32 位整数相加，产生一个 32 位结果。加实数指令将两个 32 位实数相加，产生一个 32 位实数结果。 • LAD、FBD 和 STL：IN1 + IN2 = OUT • 历史 STL： IN1 + OUT = OUT
 ADD_DI ADD_R	历史： **+I IN1, OUT** **+D IN1, OUT** **+R IN1, OUT**	
 SUB_DI SUB_R	**-I IN1, IN2, OUT** **-D IN1, IN2, OUT** **-R IN1, IN2, OUT**	整数减法指令将两个 16 位整数相减，产生一个 16 位结果。双整数减法 (-D) 指令将两个 32 位整数相减，产生一个 32 位结果。实数减法 (-R) 指令将两个 32 位实数相减，产生一个 32 位实数结果。 • LAD、FBD 和 STL：IN1 - IN2 = OUT • 历史 STL：OUT - IN1 = OUT
SUB_DI  SUB_R	历史： **-I IN1, OUT** **-D IN1, OUT** **-R IN1, OUT**	

LAD/FBD	STL	说明

MUL_I
EN ENO
IN1 OUT
IN2
MUL_DI
MUL_R
	`*I IN1, IN2, OUT`
	`*D IN1, IN2, OUT`
	`*R IN1, IN2, OUT`
		整数乘法指令将两个 16 位整数相乘，产生一个 16 位结果。双整数乘法指令将两个 32 位整数相乘，产生一个 32 位结果。实数乘法指令将两个 32 位实数相乘，产生一个 32 位实数结果。
		• LAD、FBD 和 STL：IN1 * IN2 = OUT
		• 历史 STL：IN1 * OUT = OUT

MUL_I (V2)
EN ENO
IN1 OUT
IN2
MUL_DI
MUL_R
	历史：
	`*I IN1, OUT`
	`*D IN1, OUT`
	`*R IN1, OUT`	

DIV_I
EN ENO
IN1 OUT
IN2
DIV_DI
DIV_R
	`/I IN1, IN2, OUT`
	`/D IN1, IN2, OUT`
	`/R IN1, IN2, OUT`
		整数除法指令将两个 16 位整数相除，产生一个 16 位结果。（不保留余数。）双整数除法指令将两个 32 位整数相除，产生一个 32 位结果。（不保留余数。）实数除法 (/R) 指令将两个 32 位实数相除，产生一个 32 位实数结果。
		• LAD、FBD 和 STL：IN1/IN2 = OUT
		• 历史 STL：OUT / IN1 = OUT

DIV_I (V2)
EN ENO
IN1 OUT
IN2
DIV_DI
DIV_R
	历史：
	`/I IN1, OUT`
	`/D IN1, OUT`
	`/R IN1, OUT`	

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • SM1.1 溢出 • SM1.3 除数为零	• SM1.0 运算结果 = 零 • SM1.1 溢出、运算期间生成非法值或非法输入 • SM1.2 负数结果 • SM1.3 除数为零

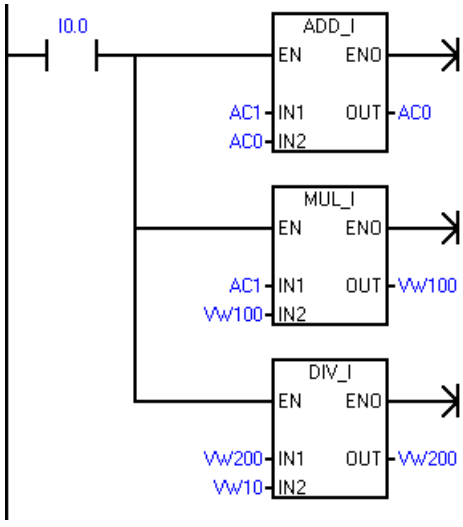
SM1.1 指示溢出错误和非法值。如果 SM1.1 置位，则 SM1.0 和 SM1.2 的状态无效，原始输入操作数不变。如果 SM1.1 和 SM1.3 未置位，则数学运算已完成且结果有效，并且 SM1.0

和 SM1.2 包含有效状态。如果在除法运算过程中 SM1.3 置位，则其他数学运算状态位保持不变。

输入/输出	数据类型	操作数
IN1、IN2	INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*AC、*LD、常数
	DINT	ID、QD、VD、MD、SMD、SD、LD、AC、HC、*VD、*LD、*AC、常数
	REAL[1]	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC、常数
OUT	INT	IW、QW、VW、MW、SMW、SW、LW、T、C、AC、*VD、*AC、*LD
	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC

[1] 实数（或浮点数）使用 ANSI/IEEE 754-1985 标准（单精度）中说明的格式进行表示。有关详细信息，请参见该标准。

示例：整数数学运算指令

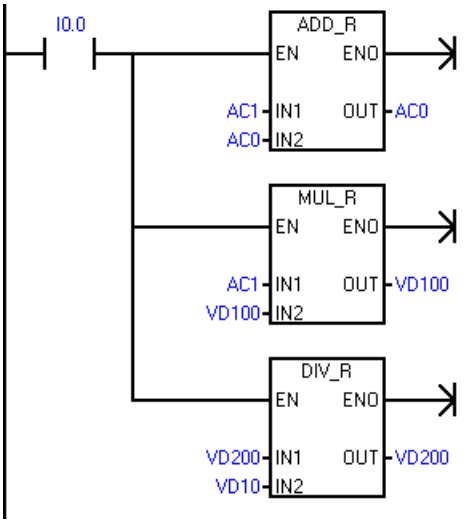
LAD	STL
	Network LD I0.0 +I AC1, AC0, AC0 *I AC1, VW100, VW100 /I VW200, VW10, VW200

LAD 示例中的整数运算

	IN1		IN2		OUT
数据相加	40	+	60	=	100

数据地址	AC1		AC0		AC0
数据相乘	40	*	20	=	800
数据地址	AC1		VW100		VW100
数据相除	4000	/	40	=	100
数据地址	W200		VW10		VW200

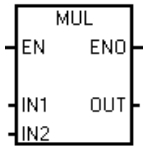
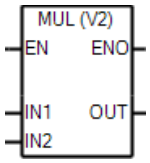
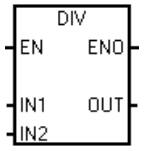
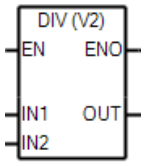
示例：实数数学运算指令

LAD	STL
	**Network 1** **LD I0.0** **+R AC1, AC0, AC0** ***R AC1, VD100, VD100** **/R VD200, VD10, VD200**

LAD 示例中的实数运算

	IN1		IN2		OUT
数据相加	4000.0	+	6000.0	=	10000.0
数据地址	AC1		AC0		AC0
数据相乘	400.0	*	200.0	=	80000.0
数据地址	AC1		VD100		VD100
数据相除	4000.0	/	41.0	=	97.5609
数据地址	VD200		VD10		VD200

9.8.2 产生双整数的整数乘法和带余数的整数除法

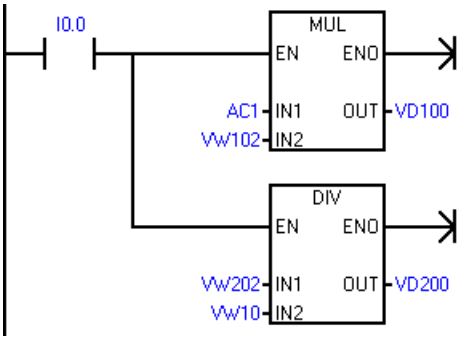
LAD/FBD	STL	说明
	**MUL IN1, IN2, OUT**	两个整数的整数乘法指令将两个 16 位整数相乘，产生一个 32 位乘积。 在 STL 中，32 位 OUT 的最低有效字（16 位）被用作其中一个乘数。 • LAD、FBD 和 STL：IN1 * IN2 = OUT • 历史 STL：IN1 * OUT = OUT
	历史： **MUL IN1, OUT**	
	**DIV IN1, IN2, OUT**	带余数的整数除法指令将两个 16 位整数相除，产生一个 32 位结果，该结果包括一个 16 位的余数（最高有效字）和一个 16 位的商（最低有效字）。 在 STL 中，32 位 OUT 的最低有效字（16 位）用作被除数。 • LAD、FBD 和 STL：IN1/IN2 = OUT • 历史 STL：OUT / IN1 = OUT **注**：此指令的 OUT 没有 DB2 区域支持。
	历史： **DIV IN1, OUT**	

ENO=0 时的非致命错误	受影响的 SM 位 [1]
• 0006H 间接地址 • SM1.1 溢出 • SM1.3 除数为零	• SM1.0 运算结果 = 零 • SM1.1 溢出、运算期间生成非法值或非法输入 • SM1.2 负数结果 • SM1.3 除数为零

[1] 对于以上两条指令，SM 位用于指示错误和非法值。如果在除法运算过程中 SM1.3（除数为零）置位，则其他数学运算状态位保持不变。否则，在数字运算完成时，所有受支持的数学运算状态位均包含有效状态。

输入/输出	数据类型	操作数
IN1、IN2	INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*LD、*AC、常数
OUT	DINT	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC

示例：MUL 和 DIV 指令

LAD	STL
	**Network 1** **LD I0.0** **MUL AC1, VW102, VD100** **DIV VW202, VW10, VD200**

[1] VD100 包含：VW100 和 VW102，VD200 包含：VW200 和 VW202。

LAD 示例中的实数运算

	IN1		IN2		OUT	
数据相乘	400	*	200	=	80000	
数据地址	AC1		VW102		VD100	
					商	余数
数据相除	4000	/	41	=	97	23
数据地址	VW202		VW10		VW200 VD200	VW202

9.8.3 三角函数、自然对数/自然指数和平方根

三角函数和反三角函数说明

LAD/FBD	STL	描述
SIN EN ENO IN OUT	**SIN IN, OUT**	正弦 (SIN)、余弦 (COS) 和正切 (TAN) 指令计算角度值 IN 的三角函数，并在 OUT 中输出结果。 反正弦 (ARC_SIN)、反余弦 (ARC_COS) 和反正切 (ARC_TAN) 指令计算角度值 IN 的反三角函数，并在 OUT 中输出结果。 ARCSIN 和 ARCCOS 的 IN 取值范围为 [-1,1]。 输入角度值以弧度为单位。 • SIN (IN) = OUT • COS (IN) = OUT • TAN (IN) = OUT • ARCSIN (IN) = OUT • ARCCOS (IN) = OUT • ARCTAN (IN) = OUT 要将角度从度转换为弧度：使用 MUL_R (*R) 指令将以度为单位的角度乘以 1.745329E-2（约为 π/180）。 对于数学函数指令，SM1.1 用于指示溢出错误和非法值。如果 SM1.1 置位，则 SM1.0 和 SM1.2 的状态无效，原始输入操作数不变。如果 SM1.1 未置位，则数学运算已完成且结果有效，并且 SM1.0 和 SM1.2 包含有效状态。
COS EN ENO IN OUT	**COS IN, OUT**	
TAN EN ENO IN OUT	**TAN IN, OUT**	
ARC_SIN EN ENO IN OUT	**ARCSIN IN, OUT**	
ARC_COS EN ENO IN OUT	**ARCCOS IN, OUT**	
ARC_TAN EN ENO IN OUT	**ARCTAN IN, OUT**	

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • SM1.1 溢出	• SM1.0 运算结果 = 零 • SM1.1 溢出、运算期间生成非法值或非法输入 • SM1.2 负数结果

输入/输出	数据类型	操作数
IN	REAL[1]	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC、常数
OUT	REAL[1]	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC

[1] 实数（或浮点数）使用 ANSI/IEEE 754-1985 标准（单精度）中说明的格式进行表示。有关详细信息，请参见该标准。

自然对数 (LN) 和自然指数 (EXP) 指令

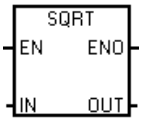
LAD/FBD	STL	描述
LN EN ENO IN OUT	**LN IN, OUT**	自然对数指令 (LN) 对 IN 中的值执行自然对数运算，并在 OUT 中输出结果。 自然指数指令 (EXP) 执行以 e 为底，以 IN 中的值为幂的指数运算，并在 OUT 中输出结果。 • LN (IN) = OUT • EXP (IN)= OUT 要从自然对数获得以 10 为底的对数：将自然对数除以 2.302585（约为 10 的自然对数）。 若要将任意实数作为另一个实数的幂，包括分数指数：组合自然指数指令和自然对数指令。例如，要将 X 作为 Y 的幂，请使用 EXP (Y * LN (X))。
EXP EN ENO IN OUT	**EXP IN, OUT**	

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • SM1.1 溢出	• SM1.0 运算结果 = 零 • SM1.1 溢出、运算期间生成非法值或非法输入 • SM1.2 负数结果

输入/输出	数据类型	操作数
IN	REAL[1]	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC、常数
OUT	REAL[1]	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC

[1] 实数（或浮点数）使用 ANSI/IEEE 754-1985 标准（单精度）中说明的格式进行表示。有关详细信息，请参见该标准。

平方根 (SQRT) 指令

LAD/FBD	STL	描述
	**SQRT IN, OUT**	平方根指令 (SQRT) 计算实数 (IN) 的平方根，产生一个实数结果 OUT。 • SQRT (IN)= OUT 要获得其它根： • 5 的立方 = 5^3 = EXP(3*LN(5)) = 125 • 125 的立方根 = 125^(1/3) = EXP((1/3)*LN(125))= 5 • 5 的立方的平方根 = 5^(3/2) = EXP(3/2*LN(5)) = 11.18034

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • SM1.1 溢出	• SM1.0 运算结果 = 零 • SM1.1 溢出、运算期间生成非法值或非法输入 • SM1.2 负数结果

输入/输出	数据类型	操作数
IN	REAL[1]	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC、常数
OUT	REAL[1]	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC

[1] 实数（或浮点数）使用 ANSI/IEEE 754-1985 标准（单精度）中说明的格式进行表示。有关详细信息，请参见该标准。

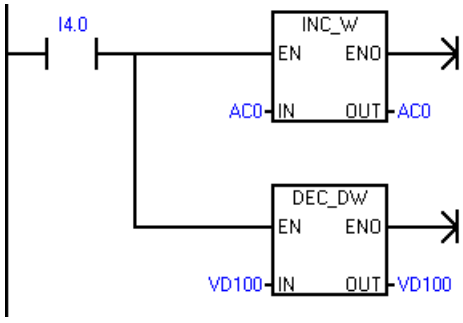
9.8.4 递增和递减

LAD/FBD	STL	说明
INC_B EN ENO IN OUT INC_W INC_DW	**INCB OUT** **INCW OUT** **INCD OUT**	递增指令对输入值 IN 加 1 并将结果输入 OUT 中。 • LAD 和 FBD：IN + 1 = OUT • STL：OUT + 1 = OUT 字节递增 (INC_B) 运算为无符号运算。字递增 (INC_W) 运算为有符号运算。双字递增 (INC_DW) 运算为有符号运算。
DEC_B EN ENO IN OUT DEC_W DEC_DW	**DECB OUT** **DECW OUT** **DECD OUT**	递减指令将输入值 IN 减 1，并在 OUT 中输出结果。 • LAD 和 FBD：IN - 1 = OUT • STL：OUT - 1= OUT 字节递减 (DEC_B) 运算为无符号运算。字递减 (DEC_W) 运算为有符号运算。双字递减 (DEC_D) 运算为有符号运算。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • SM1.1 溢出	• SM1.0 运算结果 = 零 • SM1.1 溢出、运算期间生成非法值或非法输入 • SM1.2 负数结果

输入/输出	数据类型	操作数
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC,*VD, *LD, *AC
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

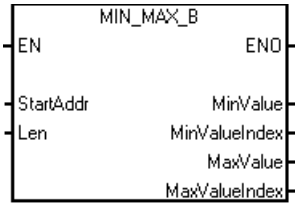
示例：递增和递减

LAD	STL
	**Network 1** **LD I4.0** **INCW AC0** **DECD VD100**

LAD 示例中的递增/递减运算

	IN				OUT
字递增	125	+	1	=	126
数据地址	AC0				AC0
双字递减	128000	-	1	=	127999
数据地址	VD100				VD100

9.8.5 取最大值或最小值

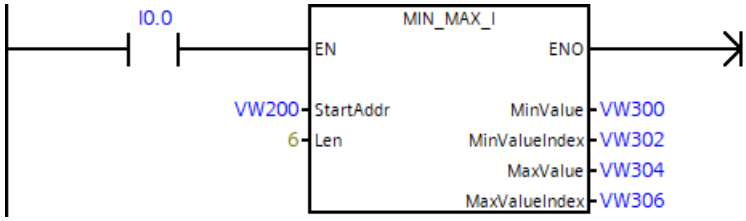
LAD/FBD	STL	描述
 MIN_MAX_I MIN_MAX_DI MIN_MAX_R	**MINMAXB StarAddr, MinValue, MinValueIndex, MaxValue, MaxValueIndex, Len** **MINMAXI StarAddr, MinValue, MinValueIndex, MaxValue, MaxValueIndex, Len** **MINMAXDI StarAddr, MinValue, MinValueIndex, MaxValue, MaxValueIndex, Len** **MINMAXR StarAddr, MinValue, MinValueIndex, MaxValue, MaxValueIndex, Len**	“取最大值或最小值”指令比较可用输入的值，然后将最小值和最大值的值及索引写入 MinValue, MinValueIndex, MaxValue 和 MaxValueIndex 的输出。 如果数组中有多个最小值或最大值，则会输出第一个最小值或最大值的索引号。 数组的索引从 0 开始。

ENO = 0 时的非致命错误

- 0X0006：间接地址，Len 超出 V 区的长度。
- 0X0090：无效参数（Len=0 时）。

输入/输出	数据类型	操作数
StartAddr	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC,
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AIW, *VD, *LD, *AC
	DINT	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
Len	WORD	IW, QW, VW, MW, SMW, SW, LW, AC, Constant, *VD, *LD, *AC
MinValue, MaxValue	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC,*VD, *LD, *AC, AQW
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
MinValueIndex, MaxValueIndex	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, *VD, *LD, *AC

示例：取数组的最小值

LAD	STL
	**程序段 1** **LD I0.0** **MINMAXI VW200, VW300, VW302, VW304, VW306, 6**

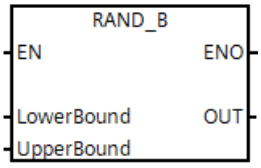
取源 6 字地址序列（VW200 到 VW210）中的 MinValue、MinValueIndex、MaxValue 和 MaxValueIndex，并将其写入到目标字地址序列（VW300 到 VW306）。

源数据值	+98	-522	+100	+0	+6	+10000
源数据地址	VW200	VW202	VW204	VW206	VW208	VW210

如果 I0.0 = 1，则执行 MIN_MAX_I 获取 MinValue、MinValueIndex、MaxValue 和 MaxValueIndex，并将其写入到目标地址

目标数据值	-522	1	+10000	5
目标数据地址	VW300	VW302	VW304	VW306

9.8.6 取随机值

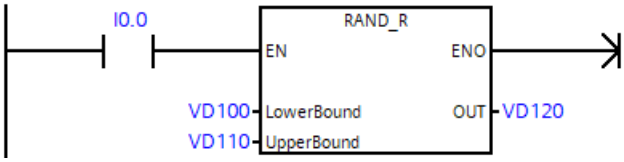
LAD/FBD	STL	描述
 RAND_I RAND_DI RAND_R	**RANDB** **LowerBound** **UpperBound OUT** **RANDI** **LowerBound** **UpperBound OUT** **RANDDI** **LowerBound** **UpperBound OUT** **RANDR** **LowerBound** **UpperBound OUT**	随机指令向 OUT 输出生成所设定范围内的一个随机值。

ENO = 0 时的非致命错误

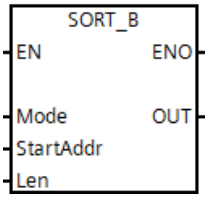
- 0X0090：无效参数（LowerBound > UpperBound 时）。

输入/输出	数据类型	操作数
LowerBound	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD, Constant
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
UpperBound	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD, Constant
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *AC, *LD
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

示例：取随机值

LAD		STL
		程序段 1 LD I0.0 RANDR VD100, VD110, VD120
取 lowerBound VD100 和 Upper Bound VD110 之间的两个封闭区间的随机数。		
源数据值	+10.0	+10000.0
源数据地址	VD100	VD110
如果 I0.0 = 1，则执行 RAND_R 以获取源数据值中的随机值，并将其写入到目标地址		
目标数据值	+1869.342	
目标数据地址	VD120	

9.8.7 数组进行排序

LAD/FBD	STL	描述
 SORT_I SORT_DI SORT_R	**SORTB Mode, StartAddr, OUT, Len** **SORTI Mode, StartAddr, OUT, Len** **SORTDI Mode, StartAddr, OUT, Len** **SORTR Mode, StartAddr, OUT, Len**	Sort 指令排序给定的数组，并将其写入 OUT 输出。

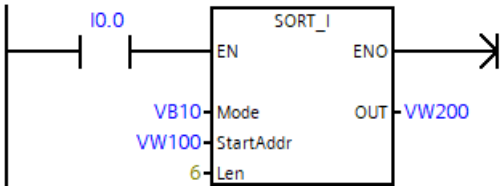
ENO = 0 时的非致命错误

- 0X0006：间接地址，Len 超出 V 区的长度。
- 0X0090：无效参数（Len=0，或模式不为 0 或 1 时）。

输入/输出	数据类型	操作数
Mode [1]	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, Constant
StartAddr	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC,
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AIW, *VD, *LD, *AC
	DINT	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
Len	WORD	IW, QW, VW, MW, SMW, SW, LW, AC, Constant, *VD, *LD, *AC
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *AC, *LD
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, *VD, *LD, *AC
	DINT	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC

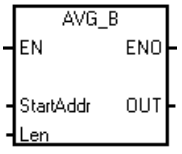
[1] 0：按升序排序；1：按降序排序

示例：数组进行排序

LAD	STL
	程序段 1 LD I0.0 SORTI VB10, VW100, VW200, 6

以字地址为起始，对源 6 字地址序列（VW100 到 VW110）中的数据按升序 (VB10=0) 排序，并将其写入到目标序列 (VW200)。						
源数据值	+23	+123	+98	+200	+300	-2
源数据地址	VW100	VW102	VW104	VW106	VW108	VW110
如果 I0.0 = 1，则执行 SORT_INT 对源数据进行排序并将其写入目标地址						
目标数据值	-2	+23	+98	+123	+200	+300
目标数据地址	VW200	VW202	VW204	VW206	VW208	VW210

9.8.8 取数组的平均值

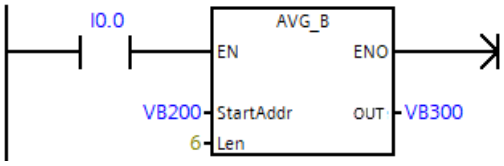
LAD/FBD	STL	描述
 AVG_I AVG_DI AVG_R	**AVGB StartAddr OUT Len** **AVGI StartAddr OUT Len** **AVGDI StartAddr OUT Len** **AVGR StartAddr OUT Len**	Average 指令计算给定数组的平均值，并将其写入到 OUT 输出。

ENO = 0 时的非致命错误

- 0X0006：间接地址，Len 超出 V 区的长度。
- 0X0090：无效参数（Len=0 时）。

输入/输出	数据类型	操作数
StartAddr	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC,
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AIW, *VD, *LD, *AC
	DINT	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
Len	WORD	IW, QW, VW, MW, SMW, SW, LW, AC, Constant, *VD, *LD, *AC
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

示例：取数组的平均值

LAD	STL
	程序段 1 LD I0.0 AVGB VB200, VB300, 6

获取源 6 字节地址序列（VB200 到 VB205）中所设置范围内的平均值，并将其写入到目标字地址 (VB300)。						
源数据值	10	20	30	50	60	70
源数据地址	VB200	VB201	VB202	VB203	VB204	VB205
如果 I0.0 = 1，则执行 AVG_B，以计算源数据的平均值，并将其写入到目标地址						
目标数据值	40					
目标数据地址	VB300					

9.8.9 取绝对值

LAD/FBD	STL	描述
ABS_I EN ENO IN OUT ABS_DI EN ENO IN OUT ABS_R EN ENO IN OUT	**ABSI IN OUT** **ABSDI IN OUT** **ABSR IN OUT**	该指令会对输入 IN 处指定的值取绝对值，并将结果加载到存储器位置 OUT 中。 **注：**该指令不会影响边界值。例如，对于类型 INT，当输入为 -32768 时，则输出会保持 -32768。

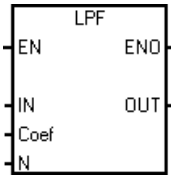
ENO = 0 时的非致命错误	受影响的 SM 位
• 0X0090：invalid parameter（无效参数）	• SM1.0 运算结果 = 零

输入/输出	数据类型	操作数
IN	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
OUT	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *AC, *LD
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

示例：绝对值指令

LAD			STL
I0.0 ABS_I EN ENO VW0 IN OUT VW2	取 VW0 的绝对值。结果放在 VW2 中。		**程序段 1** **LD I0.0** **ABSI VW0 VW2**
	IN	-101	
	OUT	101	

9.8.10 低通滤波器

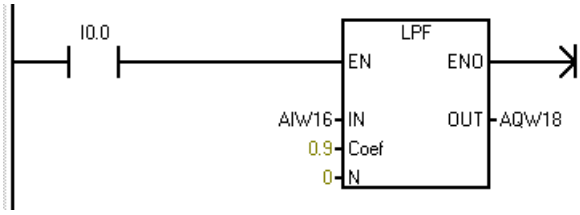
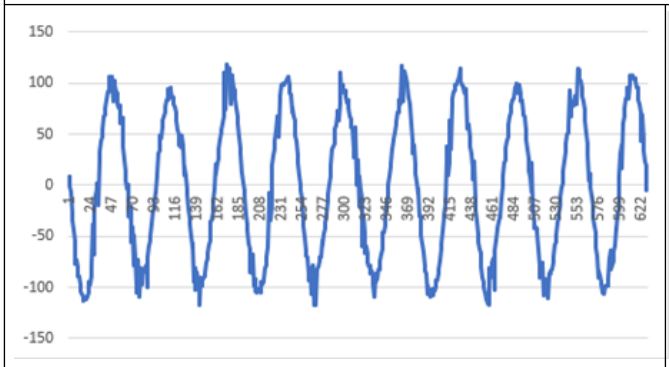
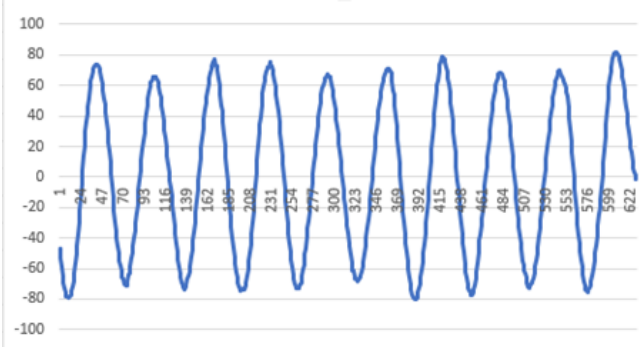
LAD/FBD	STL	描述
	**LPF IN,Coef,N,OUT**	可使用“低通滤波器”指令来衰减高频分量，例如信号中的噪声。使用以下公式计算指令的结果： y(n) = Coef*y(n-1) + （1- Coef）* x(n)； y(n) 为当前滤波器输出，Coef 为滤波器系数，y(n-1) 为上一个滤波器输出，x(n) 为当前采样值。 可指定以下滤波器参数： • Coef：滤波器系数 如果作为命令中的比率参数的滤波器系数被设为 0，则不执行滤波。 • N：目前，S7-200 SMART 支持 64 个一阶低通滤波器，因此在编程时必须指定使用哪个滤波器。 相同编号的低通滤波器将使用历史滤波值和当前采样值一起计算，得到当前滤波值。

ENO = 0 时的非致命错误

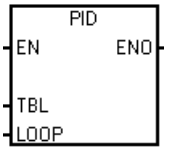
- 0X0090：无效参数（N 和 Coef 的值超出范围时）。

输入/输出	数据类型	操作数	备注
IN	WORD	AIW/VW	
Coef	REAL	ID,QD,VD,MD,SMD,SD,LD,AC,*VD,*LD,*AC, Constant	Coef 必须处于范围 [0, 1) 内
N	BYTE	VB, Constant	N 必须处于范围 [0, 63] 内
OUT	WORD	AQW/VW	

示例：低通滤波器

LAD	STL
选择滤波器 0 以处理滤波器系数为 0.9 的 AIW16 的输入。结果放在 AQW18 中。	
	LD I0.0 LPF AIW16, 0.9, 0, AQW18
 Before_LPF	 After_LPF

9.8.11 PID 回路指令

LAD/FBD	STL	描述
	**PID TBL, LOOP**	PID 回路指令 (PID) 根据输入和表 (TBL) 中的组态信息对引用的 LOOP 执行 PID 回路计算。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0013H PID 回路表非法	• SM1.1 溢出

输入/输出	数据类型	操作数
TBL	BYTE	VB
LOOP	BYTE	常数（0 到 15）

PID 回路指令（比例、积分、微分回路）用于执行 PID 计算。逻辑堆栈栈顶 (TOS) 值必须为 ON（能流），才能启用 PID 计算。该指令有两个操作数：作为回路表起始地址的表地址和取值范围为常数 0 到 15 的回路编号。

可以在程序中使用十六条 PID 指令。如果两条或两条以上的 PID 指令使用同一回路编号（即使它们的表地址不同），这些 PID 计算会互相干扰，输出不可预料。

回路表存储九个用于监控回路运算的参数，这些参数中包含过程变量当前值和先前值、设定值、输出、增益、采样时间、积分时间（复位）、微分时间（速率）以及积分和（偏置）。

要在所需采样速率下执行 PID 计算，必须在定时中断例程或主程序中以受定时器控制的速率执行 PID 指令。必须通过回路表提供采样时间作为 PID 指令的输入。

如果出现错误，将在 SMB220-SMB235：PID 控制错误代码 (页 1298)中进行指示。

PID 指令已集成自整定功能。有关自整定的详细说明，请参见“PID 回路和整定”。“PID 整定控制面板”只能用于通过 PID 向导创建的 PID 回路。

STEP 7-Micro/WIN SMART 提供 PID 向导，指导您为闭环控制过程定义 PID 算法。从“工具”(Tools) 菜单中选择“指令向导”(Instruction Wizard) 命令，然后从“指令向导”(Instruction Wizard) 窗口中选择“PID”。

说明

上限设定值和下限设定值应与过程变量的上下限对应。

9.9 中断

9.9.1 中断指令

切换到 RUN 模式时，中断开始时被禁止。在 RUN 模式下，可通过执行 ENI（中断启用）指令来启用中断处理。执行 DISI（中断禁止）指令将禁止处理中断；但激活的中断事件将继续排队。

ENI、DISI 和 CRETI

LAD	FBD	STL	描述
ENI	ENI	**ENI**	中断启用指令全局性启用对所有连接的中断事件的处理。
DISI	DISI	**DISI**	中断禁止指令全局性禁止对所有中断事件的处理。
RETI	RETI	**CRETI**	从中断有条件返回指令可用于根据前面的程序逻辑的条件从中断返回。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0004H 尝试在中断例程中执行不允许执行的 ENI 或 DISI	无

ATCH、DTCH 和 CEVENT

LAD/FBD	STL	描述
ATCH EN ENO INT EVNT	**ATCH INT, EVNT**	中断连接指令将中断事件 EVNT 与中断例程编号 INT 相关联，并启用中断事件。
DTCH EN ENO EVNT	**DTCH EVNT**	中断分离指令解除中断事件 EVNT 与所有中断例程的关联，并禁用中断事件。
CLR_EVNT EN ENO EVNT	**CEVENT EVNT**	清除中断事件指令从中断队列中移除所有类型为 EVNT 的中断事件。使用该指令可将不需要的中断事件从中断队列中清除。如果该指令用于清除假中断事件，则应在从队列中清除事件之前分离事件。否则，在执行清除事件指令后，将向队列中添加新事件。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0002H 将输入分配给 HSC 时发生冲突 • 0016H 尝试使用已分配给运动控制功能的输入通道上的 HSC 或边缘中断 • 0019H 尝试使用未安装或未组态信号板的信号板函数 • 0090H 操作数无效（事件编号非法）	无

输入/输出	数据类型	操作数
INT	BYTE	常数：中断例程编号（0 到 127）
EVNT	BYTE	常数：中断事件编号

9.9.2 中断例程概述和 CPU 型号事件支持

在调用中断例程之前，必须指定中断事件和要在事件发生时执行的程序段之间的关联。可以使用中断连接指令将中断事件（由中断事件编号指定）与程序段（由中断例程编号指定）相关联。可以将多个中断事件连接到一个中断例程，但不能将一个事件连接到多个中断例程。

连接事件和中断例程时，仅当程序已执行全局 ENI（中断启用）指令且中断事件处理处于激活状态时，新出现此事件才会执行所连接的中断例程。否则，CPU 会将该事件添加到中断事件队列中。如果使用全局 DISI（中断禁止）指令禁止所有中断，每次发生中断事件时 CPU 都会排队，直至使用全局 ENI（中断启用）指令重新启用中断或中断队列溢出。

可以使用中断分离指令取消中断事件与中断例程之间的关联，从而禁用单独的中断事件。分离中断指令使中断返回未激活或被忽略状态。下表列出了不同类型的中断事件。

事件	描述	CPU SR40/ST40 CPU SR60/ST60 CPU ST32
0	I0.0 上升沿	是
1	I0.0 下降沿	是
2	I0.1 上升沿	是
3	I0.1 下降沿	是

事件	描述	CPU SR40/ST40 CPU SR60/ST60 CPU ST32
4	I0.2 上升沿	是
5	I0.2 下降沿	是
6	I0.3 上升沿	是
7	I0.3 下降沿	是
8	端口 0 接收字符	是
9	端口 0 发送完成	是
10	定时中断 0（SMB34 控制时间间隔）	是
11	定时中断 1（SMB35 控制时间间隔）	是
12	HSC0 CV=PV（当前值 = 预设值）	是
13	HSC1 CV=PV（当前值 = 预设值）	是
14	HSC1 方向改变	是
15	HSC1 外部复位	是
16	HSC2 CV=PV（当前值 = 预设值）	是
17	HSC2 方向改变	是
18	HSC2 外部复位	是
19	PTO0 脉冲计数完成	是
20	PTO1 脉冲计数完成	是
21	定时器 T32 CT=PT（当前时间 = 预设时间）	是
22	定时器 T96 CT=PT（当前时间 = 预设时间）	是
23	端口 0 接收消息完成	是
24	端口 1 接收消息完成	是
25	端口 1 接收字符	是
26	端口 1 发送完成	是
27	HSC0 方向改变	是
28	HSC0 外部复位	是
29	HSC4 CV=PV	是
30	HSC4 方向改变	是
31	HSC4 外部复位	是
32	HSC3 CV=PV（当前值 = 预设值）	是
33	HSC5 CV=PV	是

事件	描述	CPU SR40/ST40 CPU SR60/ST60 CPU ST32
34	PTO2 脉冲计数完成	是
35	I7.0 上升沿（信号板插槽 0，输入 0）	是
36	I7.0 下降沿（信号板插槽 0，输入 0）	是
37	I7.1 上升沿（信号板插槽 0，输入 1）	是
38	I7.1 下降沿（信号板插槽 0，输入 1）	是
43	HSC5 方向改变	是
44	HSC5 外部复位	是
45	PTO3 脉冲计数完成	是
46	端口 2 接收消息完成	是
47	端口 2 接收字符	是
48	端口 2 发送完成	是
49	I72.0 上升沿（信号板插槽 1，输入 0）	是
50	I72.0 下降沿（信号板插槽 1，输入 0）	是
51	I72.1 上升沿（信号板插槽 1，输入 1）	是
52	I72.1 下降沿（信号板插槽 1，输入 1）	是
53	HSC3 方向改变	是
54	HSC3 外部复位	是
55	HSC6 CV=PV	是
56	HSC6 方向改变	是
57	HSC6 外部复位	是
58	HSC7 CV=PV	是
59	HSC7 方向改变	是
60	HSC7 外部复位	是
61	定时器 T262 CT=PT（当前时间 = 预设时间）	是
62	定时器 T263 CT=PT（当前时间 = 预设时间）	是
63	定时器 T264 CT=PT（当前时间 = 预设时间）	是
64	定时器 T265 CT=PT（当前时间 = 预设时间）	是
65	定时器 T266 CT=PT（当前时间 = 预设时间）	是
66	定时器 T267 CT=PT（当前时间 = 预设时间）	是
67	PTO4 脉冲计数完成	是

9.9.3 中断编程准则

中断例程执行

执行中断例程执行时会响应关联的内部或外部事件。 执行了中断例程的最后一个指令之后，控制会在中断时返回到扫描周期的断点。 您可以通过执行“从中断有条件返回指令”(CRETI) 退出例程。

中断处理可快速响应特殊内部或外部事件。 可优化中断例程以执行特定任务，然后将控制权返回到扫描周期。

说明

- 中断例程中不能使用中断禁止 (DISI)、中断启用 (ENI)、高速计数器定义 (HDEF) 和结束 (END) 指令。
- 应保持中断例程编程逻辑简短，这样执行速度会更快，其它过程也不会延迟很长时间。 如果不这样做，则可能会出现无法预料的情形，从而导致主程序控制的设备异常运行。

中断的系统支持

由于中断能影响触点、线圈和累加器逻辑，所以系统会保存并重新装载逻辑堆栈、累加器寄存器以及用于指示累加器和指令操作状态的特殊存储器位 (SM)。 这样可避免因进入和退出中断例程而导致用户主程序中断。

从中断例程调用子例程

可从中断例程中调用四个嵌套级别的子例程。 累加器和逻辑堆栈在中断例程和从中断例程调用的四个嵌套级别子例程之间共享

主程序和中断例程共享数据

可在主程序和一个或多个中断例程之间共享数据。 由于无法预测 CPU 何时生成中断，所以最好限制中断例程和程序中的其它位置使用的变量数。 如果在主程序中执行指令时被中断事件中断，中断程序的操作可能会导致共享数据出现一致性问题。 使用中断块“变量表”（块调用接口表）可确保中断例程仅使用临时存储器，从而不会覆盖程序其它位置使用的数据。

确保对单个共享变量的访问

- 对于共享单个变量的 STL 程序： 如果共享数据是单字节、字或双字变量并且程序以 STL 编写，则通过将对共享数据进行运算所得的中间值仅存储在非共享存储单元或累加器可确保正确的共享访问。
- 对于共享单个变量的 LAD 程序： 如果共享数据是单字节、字或双字变量并且程序以 LAD 编写，则通过规定仅使用传送指令（MOVB、MOVW、MOVD、MOVR）访问共享存储单元可确保正确的共享访问。 许多 LAD 指令都是由 STL 指令的可中断序列组成，但这些传送指令却是由单个 STL 指令组成，单个 STL 指令的执行不受中断事件的影响。

确保对多个共享变量的访问

对于共享多个变量的 STL 或 LAD 程序： 如果共享数据由许多相关的字节、字或双字组成，则可使用中断禁用/启用指令（DISI 和 ENI）来控制中断例程的执行。 在主程序中即将对共享存储单元开始操作的点，禁止中断。 所有影响共享位置的操作都完成后，重新启用中断。 在中断禁用期间，无法执行中断例程，因此无法访问共享存储单元；但此方法会导致对中断事件的响应发生延迟。

9.9.4 S7-200 SMART CPU 支持的中断事件类型

通信端口中断

CPU 的串行通信端口可通过程序进行控制。通信端口的这种操作模式称为自由端口模式。在自由端口模式下，程序定义波特率、每个字符的位数、奇偶校验和协议。接收和发送中断可简化程序控制的通信。有关详细信息，请参见发送和接收指令。

I/O 中断

I/O 中断包括上升/下降沿中断、高速计数器中断和脉冲串输出中断。CPU 可以为输入通道 I0.0、I0.1、I0.2 和 I0.3（以及带有可选数字量输入信号板的标准 CPU 的输入通道 I7.0、I7.1、I72.0 和 I72.1）生成输入上升和/或下降沿中断。可对这些输入点中的每一个捕捉上升沿和下降沿事件。这些上升沿/下降沿事件可用于指示在事件发生时必须立即处理的状况。

高速计数器中断使您可以对下列情况做出响应：当前值达到预设值，与轴旋转方向反向相对应的计数方向发生改变或计数器外部复位。这些高速计数器事件均可触发实时执行的操作，以响应在可编程逻辑控制器扫描速度下无法控制的高速事件。

脉冲串输出中断在指定的脉冲数完成输出时立即进行通知。脉冲串输出的典型应用为步进电机控制。

通过将中断例程连接到相关 I/O 事件来启用上述各中断。

基于时间的中断

基于时间的中断包括定时中断、定时器 T32/T96 中断和定时器 T262 - T267 中断。可使用定时中断指定循环执行的操作。循环时间位于 1 ms 到 255 ms 之间，按增量为 1 ms 进行设置。必须在定时中断 0 的 SMB34 和定时中断 1 的 SMB35 中写入循环时间。

每次定时器到时，定时中断事件都会将控制权传递给相应的中断例程。通常，可以使用定时中断来控制模拟量输入的采样或定期执行 PID 回路。

将中断例程连接到定时中断事件时，启用定时中断并且开始定时。连接期间，系统捕捉周期时间值，因此 SMB34 和 SMB35 的后续变化不会影响周期时间。要更改周期时间，必须修改周期时间值，然后将中断例程重新连接到定时中断事件。重新连接时，定时中断功能会清除先前连接的所有累计时间，并开始用新值计时。

定时中断启用后，将连续运行，每个连续时间间隔后，会执行连接的中断例程。如果退出 RUN 模式或分离定时中断，定时中断将禁用。如果执行了全局 DISI（中断禁止）指令，定时中断会继续出现，但是尚未处理所连接的中断例程。每次定时中断出现均排队等候，直至中断启用或队列已满。

使用定时器 T32/T96 中断和定时器 T262 - T267 中断可及时响应指定时间间隔的结束。仅 1 ms 分辨率的接通延时 (TON) 和断开延时 (TOF) 定时器 T32、T96 和 T262 - T267 支持此类中断。否则 T32、T96 和 T262 - T267 定时器正常工作。启用中断后，如果在 CPU 中执行正常的 1 ms 定时器更新期间，激活定时器的当前值等于预设时间值，将执行连接的中断例程。可通过将中断例程连接到 T32（事件 21）、T96（事件 22）和 T262 - T267（事件 61 - 66）中断事件来启用这些中断。

9.9.5 中断优先级、排队和示例程序

中断服务

优先级相同时，CPU 按照先来先处理的原则处理中断。在某一时间仅执行一个用户中断例程。中断例程开始执行后，一直执行直至完成。其他中断例程无法预先清空该例程，即使更高优先级的例程。正在处理另一个中断时发生的中断会进行排队等待处理。下表显示了三种中断队列以及它们能存储的最大中断数。

出现的中断有可能比队列所能容纳的中断更多。因此，队列溢出存储器位（标识已丢失的中断事件类型）由系统进行维护。下表给出了中断队列溢出位。应仅在中断例程中使用这些位，因为当队列清空时，这些位将复位，并且控制权将返回到扫描周期。

如果多个中断事件同时发生，则优先级（组和组内）会确定首先处理哪一个中断事件。处理了优先级最高的中断事件之后，会检查队列，以查找仍在队列中的当前优先级最高的事件，并会执行连接到该事件的中断例程。会继续执行这一步骤，直至队列为空且控制权返回到扫描周期。

每个中断队列的最大条目数

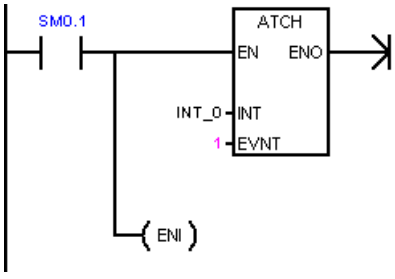
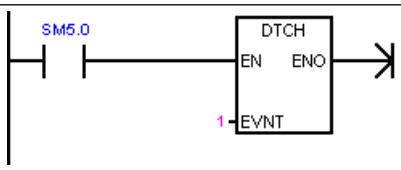
下表给出了所有中断事件及其优先级和分配的事件编号。

队列	所有 S7-200 SMART CPU 型号的队列深度
通信队列	4
I/O 中断队列	16
定时中断队列	8

中断队列溢出位

说明（0 = 无溢出，1 = 溢出）	SM 位
通信队列	SM4.0
I/O 中断队列	SM4.1
定时中断队列	SM4.2

示例 1：输入信号沿检测器中断

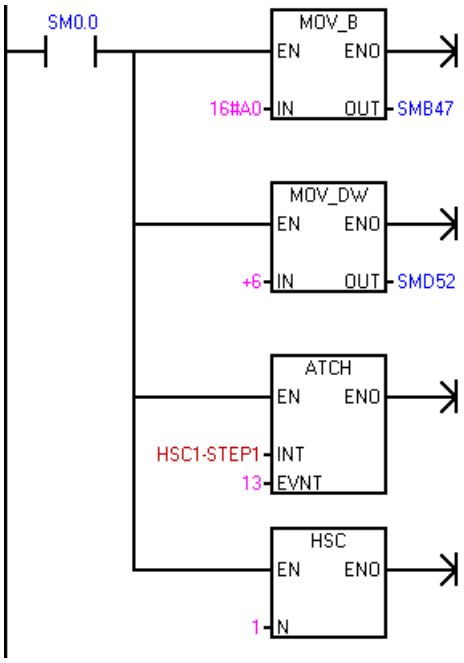
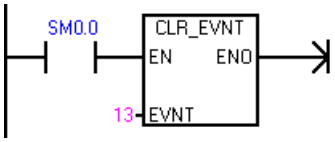
LAD			STL
MAIN 程序段 1		第一次扫描时： 1. 将中断例程 INT_0 定义为 I0.0 的下降沿中断 2. 全局启用中断。	**Network 1** **LD SM0.1** **ATCH INT_0, 1** **ENI**
程序段 2		如果检测到 I/O 错误，则禁用 I0.0 的下降沿中断。 （此程序段可选。）	**Network 2** **LD SM5.0** **DTCH 1**

LAD			STL
程序段 3	M5.0 DISI	M5.0 接通时，会禁用所有中断。禁用时，所连接中断事件将排队，但是不会执行相应的中断例程，直至使用 ENI 指令重新启用中断。	Network 3 LD M5.0 DISI
INT 0 程序段 1	SM5.0 RETI	I0.0 下降沿中断例程：基于 I/O 错误的有条件返回。	Network 1 LD SM5.0 CRETI

示例 2：用于读取模拟量输入值的定时中断

LAD			STL
MAIN 程序段 1	SM0.1 SBR_0 EN	首次扫描时，调用子例程 0。	Network 1 LD SM0.1 CALL SBR_0
SBR 0 程序段 1	SM0.0 MOV_B EN ENO 100 IN OUT SMB34 ATCH EN ENO INT_0 INT 10 EVNT ENI	将定时中断 0 的时间间隔设置为 100 ms。 将定时中断 0（事件 10）连接到 INT_0。 全局中断启用。	Network 1 LD SM0.0 MOVB 100, SMB34 ATCH INT_0, 10 ENI
INT 0 程序段 1	SM0.0 MOV_W EN ENO AIW16 IN OUT VW100	每 100 ms 读取一次 AIW16 的值。	Network 1 LD SM0.0 MOVW AIW16, VW100

示例 3：清除中断事件指令

LAD			STL
SBR 1 程序段 1		HSC 指令向导： 设置控制位，写入预设值。 PV = 6 连接中断 HSC1_STEP1: CV = PV（对于 HC1） 组态 HSC 1。	**Network 1** **LD SM0.0** **MOVB 16#A0, SMB47** **MOVD +6, SMD52** **ATCH HSC1_STEP1, 13**
SBR 1 程序段 2		清除机器振动引起的不必要中断。	**Network 2** **LD SM0.0** **CEVNT 13**

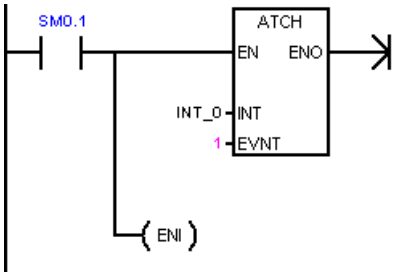
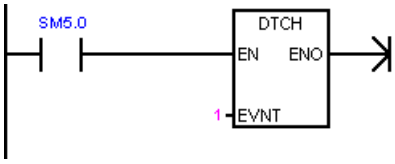
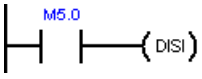
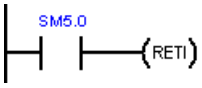
表格 9-20　中断事件的优先级顺序

优先级组	事件	描述
通信 最高优先级	8	端口 0 接收字符
	9	端口 0 发送完成
	23	端口 0 接收消息完成
	24	端口 1 接收消息完成
	25	端口 1 接收字符
	26	端口 1 发送完成
	46	端口 2 接收消息完成
	47	端口 2 接收字符
	48	端口 2 发送完成

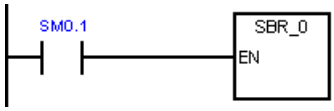
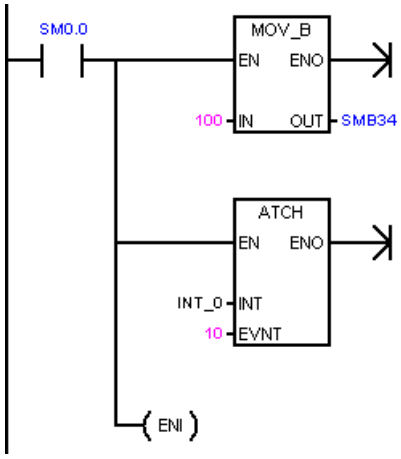
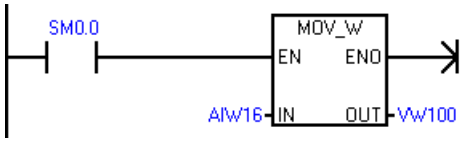
优先级组	事件	描述
离散 中等优先级	19	PTO0 脉冲计数完成
	20	PTO1 脉冲计数完成
	34	PTO2 脉冲计数完成
	45	PTO3 脉冲计数完成
	67	PTO4 脉冲计数完成
	0	I0.0 上升沿
	2	I0.1 上升沿
	4	I0.2 上升沿
	6	I0.3 上升沿
	1	I0.0 下降沿
	3	I0.1 下降沿
	5	I0.2 下降沿
	7	I0.3 下降沿
	35	I7.0 上升沿（信号板 0，输入 0）
	36	I7.0 下降沿（信号板 0，输入 0）
	37	I7.1 上升沿（信号板 0，输入 1）
	38	I7.1 下降沿（信号板 0，输入 1）
	49	I72.0 上升沿（信号板 1，输入 0）
	50	I72.0 下降沿（信号板 1，输入 0）
	51	I72.1 上升沿（信号板 1，输入 1）
	52	I72.1 下降沿（信号板 1，输入 1）
	12	HSC0 CV=PV（当前值 = 预设值）
	27	HSC0 方向改变
	28	HSC0 外部复位
	13	HSC1 CV=PV（当前值 = 预设值）
	14	HSC1 方向改变
	15	HSC1 外部复位/视在阻抗相位
	16	HSC2 CV=PV（当前值 = 预设值）
	17	HSC2 方向改变
	18	HSC2 外部复位
	32	HSC3 CV=PV（当前值 = 预设值）

优先级组	事件	描述
	53	HSC3 方向改变
	54	HSC3 外部复位
	29	HSC4 CV=PV
	30	HSC4 方向改变
	31	HSC4 外部复位
	33	HSC5 CV=PV
	43	HSC5 方向改变
	44	HSC5 外部复位
	55	HSC6 CV=PV
	56	HSC6 方向改变
	57	HSC6 外部复位
	58	HSC7 CV=PV
	59	HSC7 方向改变
	60	HSC7 外部复位
定时 最低优先级	10	定时中断 0 SMB34
	11	定时中断 1 SMB35
	21	定时器 T32 CT = PT 中断
	22	定时器 T96 CT = PT 中断
	61	定时器 T262 CT = PT 中断
	62	定时器 T263 CT = PT 中断
	63	定时器 T264 CT = PT 中断
	64	定时器 T265 CT = PT 中断
	65	定时器 T266 CT = PT 中断
	66	定时器 T267 CT = PT 中断

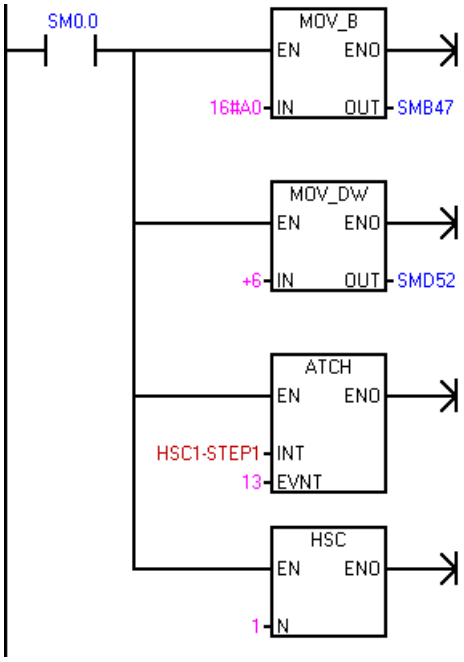
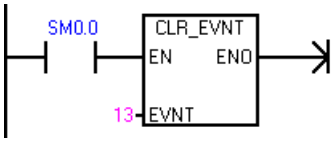
示例 1：输入信号沿检测器中断

LAD			STL
MAIN 程序段 1		第一次扫描时： 1. 将中断例程 INT_0 定义为 I0.0 的下降沿中断 2. 全局启用中断。	**Network 1** **LD SM0.1** **ATCH INT_0, 1** **ENI**
程序段 2		如果检测到 I/O 错误，则禁用 I0.0 的下降沿中断。 （此程序段可选。）	**Network 2** **LD SM5.0** **DTCH 1**
程序段 3		M5.0 接通时，会禁用所有中断。禁用时，所连接中断事件将排队，但是不会执行相应的中断例程，直至使用 ENI 指令重新启用中断。	**Network 3** **LD M5.0** **DISI**
INT 0 程序段 1		I0.0 下降沿中断例程：基于 I/O 错误的有条件返回。	**Network 1** **LD SM5.0** **CRETI**

示例 2：用于读取模拟量输入值的定时中断

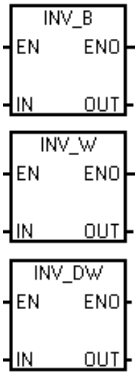
LAD			STL
MAIN 程序段 1		首次扫描时，调用子例程 0。	**Network 1** **LD SM0.1** **CALL SBR_0**
SBR 0 程序段 1		将定时中断 0 的时间间隔设置为 100 ms。 将定时中断 0（事件 10）连接到 INT_0。 全局中断启用。	**Network 1** **LD SM0.0** **MOVB 100, SMB34** **ATCH INT_0, 10** **ENI**
INT 0 程序段 1		每 100 ms 读取一次 AIW16 的值。	**Network 1** **LD SM0.0** **MOVW AIW16, VW100**

示例 3：清除中断事件指令

LAD			STL
SBR 1 程序段 1		HSC 指令向导： 设置控制位，写入预设值。 PV = 6 连接中断 HSC1_STEP1: CV = PV（对于 HC1） 组态 HSC 1。	**Network 1** **LD SM0.0** **MOVB 16#A0, SMB47** **MOVD +6, SMD52** **ATCH HSC1_STEP1, 13**
SBR 1 程序段 2		清除机器振动引起的不必要中断。	**Network 2** **LD SM0.0** **CEVNT 13**

9.10 逻辑运算

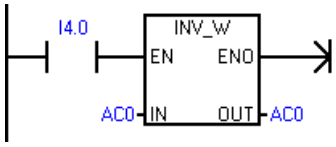
9.10.1 取反

LAD/FBD	STL	说明
	**INVB OUT** **INVW OUT** **INVD OUT**	字节取反、字取反和双字取反指令对输入 IN 执行求补操作，并将结果装载到存储单元 OUT 中

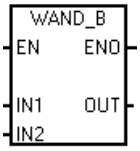
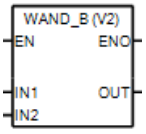
ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址	• SM1.0 运算结果 = 零

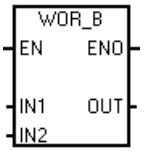
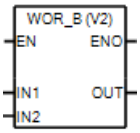
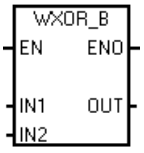
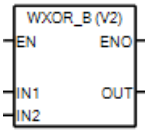
输入/输出	数据类型	操作数
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC,*VD, *LD, *AC
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

示例：取反指令

LAD		STL
	取反 AC0 中的字值。结果放在 AC0 中。 字输入取反 AC0 1101 0111 1001 0101 执行补码取反 字输出取反 AC0 0010 1000 0110 1010	**Network 1** **LD I4.0** **INVW AC0**

9.10.2 AND、OR、XOR、NAND、NOR 和 NXOR

LAD/FBD	STL	说明
 WAND_W WAND_DW	**ANDB** **IN1,IN2,OUT** **ANDW** **IN1,IN2,OUT** **ANDD IN1,** **IN2,OUT**	“与”字节、“与”字以及“与”双字指令，与两个输入值 IN1 和 IN2 的相应位进行逻辑“与”运算，并将结果加载到 OUT 的存储单元中。 • LAD 和 FBD：IN1 AND IN2 = OUT • STL：IN1 AND OUT = OUT
 WAND_W WAND_DW	历史： **MOVB IN1,IN2** **ANDB IN2,OUT** **MOVW IN1,IN2** **ANDW IN2,OUT** **MOVD IN1,IN2** **ANDD IN2,OUT**	

LAD/FBD	STL	说明
 WOR_W WOR_DW	**ORB** **IN1,IN2,OUT** **ORW** **IN1,IN2,OUT** **ORD** **IN1,IN2,OUT**	“或”字节、“或”字以及“或”双字指令，与两个输入值 IN1 和 IN2 的相应位进行逻辑“或”运算，并将计算结果加载到 OUT 的存储单元中。 • LAD 和 FBD：IN1 OR IN2 = OUT • STL：IN1 OR OUT = OUT
 WOR_W WOR_DW	历史： **MOVB IN1,IN2** **ORB IN2,OUT** **MOVW IN1,IN2** **ORW IN2,OUT** **MOVD IN1,IN2** **ORD IN2,OUT**	
 WXOR_W WXOR_DW	**XORB** **IN1,IN2,OUT** **XORW** **IN1,IN2,OUT** **XORD** **IN1,IN2,OUT**	“异或”字节、“异或”字以及“异或”双字指令，与两个输入值 IN1 和 IN2 的相应位进行逻辑“异或”运算，并将计算结果加载到 OUT 的存储单元中。 • LAD 和 FBD：IN1 XOR IN2 = OUT • STL：IN1 XOR OUT = OUT
 WXOR_W WXOR_DW	历史： **MOVB IN1,IN2** **XORB IN2,OUT** **MOVW IN1,IN2** **XORW IN2,OUT** **MOVD IN1,IN2** **XORD IN2,OUT**	

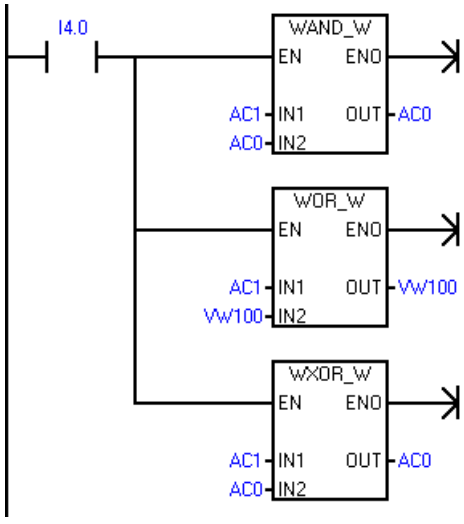
LAD/FBD	STL	说明
NAND_B EN ENO IN1 OUT IN2 NAND_W NAND_DW	NANDB IN1,IN2,OUT **NANDW IN1,IN2,OUT** **NANDD IN1,IN2,OUT**	“与非”字节、“与非”字以及“与非”双字指令，与两个输入值 IN1 和 IN2 的相应位进行逻辑“与非”运算，并将计算结果加载到 OUT 的存储单元中。 • LAD 和 FBD：IN1 NAND IN2 = OUT • STL：IN1 NAND OUT = OUT
NAND_B (V2) EN ENO IN1 OUT IN2 NAND_W NAND_DW	历史： **MOVB IN1,IN2** **NANDB IN2,OUT** **MOVW IN1,IN2** **NANDW IN2,OUT** **MOVD IN1,IN2** **NANDD IN2,OUT**	
NOR_B EN ENO IN1 OUT IN2 NOR_W NOR_DW	**NORB IN1,IN2,OUT** **NORW IN1,IN2,OUT** **NORD IN1,IN2,OUT**	“或非”字节、“或非”字以及“或非”双字指令，与两个输入值 IN1 和 IN2 的相应位进行逻辑“或非”运算，并将计算结果加载到 OUT 的存储单元中。 • LAD 和 FBD：IN1 NOR IN2 = OUT • STL：IN1 NOR OUT = OUT
NOR_B (V2) EN ENO IN1 OUT IN2 NOR_W NOR_DW	历史： **MOVB IN1,IN2** **NORB IN2,OUT** **MOVW IN1,IN2** **NORW IN2,OUT** **MOVD IN1,IN2** **NORD IN2,OUT**	

LAD/FBD	STL	说明
NXOR_B EN ENO IN1 OUT IN2 NXOR_W NXOR_DW	**NXORB IN1,IN2,OUT** **NXORW IN1,IN2,OUT** **NXORD IN1,IN2,OUT**	“异或非”字节、“异或非”字以及“异或非”双字指令，与两个输入值 IN1 和 IN2 的相应位进行逻辑“异或非”运算，并将计算结果加载到存储单元 OUT 中。 • LAD 和 FBD：IN1 NXOR IN2 = OUT • STL：IN1 NXOR OUT = OUT
NXOR_B (V2) EN ENO IN1 OUT IN2 NXOR_W NXOR_DW	历史： **MOVB IN1,IN2** **NXORB IN2,OUT** **MOVW IN1,IN2** **NXORW IN2,OUT** **MOVD IN1,IN2** **NXORD IN2,OUT**	

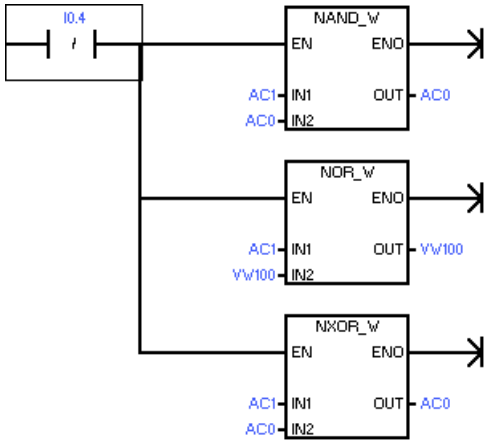
ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址	• SM1.0 运算结果 = 零

输入/输出	数据类型	操作数
IN1, IN2	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *AC, *LD
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

示例：与、或和异或指令

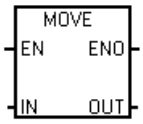
LAD		STL
	AC1 0001 1111 0110 1101 AND AC0 1101 0011 1110 0110 等于 AC0 0001 0011 0110 0100 AC1 0001 1111 0110 1101 OR VW100 1101 0011 1010 0000 等于 VW100 1101 1111 1110 1101 AC1 0001 1111 0110 1101 XOR AC0 0001 0011 0110 0100 等于 AC0 0000 1100 0000 1001	**Network 1** **LD I4.0** **ANDW AC1, AC0, ACO** **ORW AC1, VW100, VW100** **XORW AC1, AC0, AC0**

示例：NAND、NOR 和 NXOR 指令

LAD		STL
	AC1 0001 1111 0110 1101 NAND AC0 1101 0011 1110 0110 等于 AC0 1110 1100 1001 1011 AC1 0001 1111 0110 1101 NOR VW100 1101 0011 1010 0000 等于 VW100 0010 0000 0001 0010 AC1 0001 1111 0110 1101 NXOR AC0 1110 1100 1001 1011 等于 AC0 0000 1100 0000 1001	**Network 1** **LDN I0.4** **NANDW AC1, AC0, AC0** **NORW AC1, VW100, VW100** **NXORW AC1, AC0, AC0**

9.11 传送

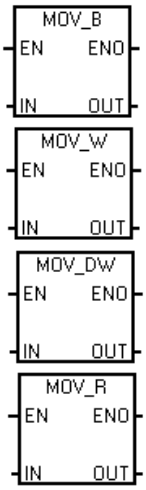
9.11.1 传送

LAD/FBD	STL	描述
	**MOVE IN, OUT**	传送指令将数据值在相同数据类型之间从源（常数或存储单元）IN 传送到新存储单元 OUT，而不会更改源存储单元中存储的值。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址	无

输入/输出	数据类型	操作数
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, Constant
	WORD, INT	IW, QW, VW, MW, SMW, SW, LW, Constant
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, Constant
	REAL	ID, QD, VD, MD, SMD, SD, LD, Constant
	UDT	
	ARRAY	
	STRING	
	FB 实例	
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB,
	WORD, INT	IW, QW, VW, MW, SMW, SW, LW
	DWORD, DINT, REAL	ID, QD, VD, MD, SMD, SD, LD
	UDT	
	ARRAY	
	STRING	
	FB 实例	

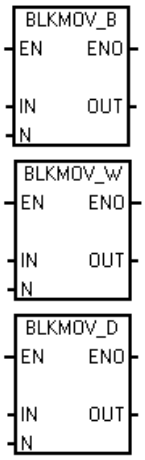
9.11.2 字节、字、双字或实数传送

LAD/FBD	STL	说明
	**MOVB IN, OUT** **MOVW IN, OUT** **MOVD IN, OUT** **MOVR IN, OUT**	字节传送、字传送、双字传送和实数传送指令将数据值从源（常数或存储单元）IN 传送到新存储单元 OUT，而不会更改源存储单元中存储的值。 使用双字传送指令创建指针。 有关详细信息，请参见指针和间接寻址部分。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址	无

输入/输出	数据类型	操作数
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, HC, &VB, &IB, &QB, &MB, &SB, &T, &C, &SMB, &AIW, &AQW, AC, *VD, *LD, *AC, Constant
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AQW, *VD, *LD, *AC
	DWORD, DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

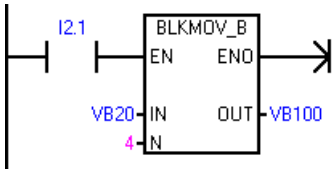
9.11.3 块传送

LAD/FBD	STL	说明
	**BMB IN, OUT, N** **BMW IN, OUT, N** **BMD IN, OUT, N**	字节块传送、字块传送、双字块传送指令将已分配数据值块从源存储单元（起始地址 IN 和连续地址）传送到新存储单元（起始地址 OUT 和连续地址）。参数 N 分配要传送的字节、字或双字数。存储在源单元的数据值块不变。 N 取值范围是 1 到 255。

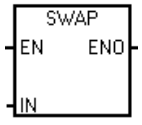
ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围	无

输入/输出	数据类型	操作数
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AIW, *VD, *LD, *AC
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AQW, *VD, *LD, *AC
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, Constant, *VD, *LD, *AC

示例：块传送指令

LAD			STL	
	将源四字节地址序列（VB20 至 VB23）中的数据传送（复制）到目标四字节地址序列（VB100 至 VB103）。		**Network 1** **LD I2.1** **BMB VB20, VB100, 4**	
源数据值	30	31	32	33
源数据地址	VB20	VB21	VB22	VB23
如果 I2.1 = 1，则执行 BLKMOV_B，以便将源数据值传送到目标地址				
目标数据值	30	31	32	33
目标数据地址	VB100	VB101	VB102	VB103

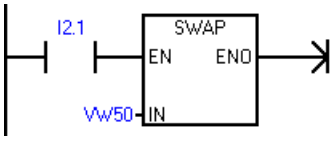
9.11.4 交换字节

LAD/FBD	STL	说明
	**SWAP IN**	字节交换指令用于交换字 IN 的最高有效字节和最低有效字节。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址	无

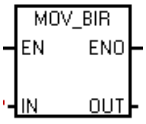
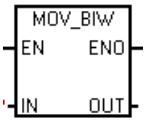
输入/输出	数据类型	操作数
IN	WORD	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、*VD、*LD、*AC

示例： 交换指令

LAD	STL
	**Network 1** **LD I2.1** **SWAP VW50**

十六进制数据值	D6	C3
数据地址	VB50	VB51
如果 I2.1 = 1，则执行 SWAP，以便交换数据字中的字节数据		
十六进制数据值	C3	D6
数据地址	VB50	VB51

9.11.5 字节立即传送（读取和写入）

LAD/FBD	STL	描述
	**BIR IN, OUT**	移动字节立即读取指令读取物理输入 IN 的状态，并将结果写入存储器地址 OUT 中，但不更新过程映像寄存器。
	**BIW IN, OUT**	传送字节立即写入指令从存储器地址 IN 读取数据，并将其写入物理输出 OUT 以及相应的过程映像位置。

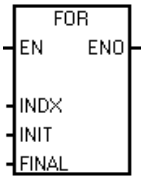
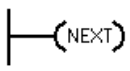
ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • 无法访问扩展模块	无

输入/输出	数据类型	操作数
IN (BIR)	BYTE	IB、*VD、*LD、*AC
IN (BIW)	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC

输入/输出	数据类型	操作数
OUT (BIR)	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC
OUT (BIW)	BYTE	QB、*VD、*LD、*AC

9.12 程序控制

9.12.1 FOR-NEXT 循环

LAD/FBD	STL	说明
	**FOR INDX, INIT, FINAL**	FOR 指令执行 FOR 和 NEXT 指令之间的指令。 需要分配索引值或当前循环计数 INDX、起始循环计数 INIT 和结束循环计数 FINAL。
LAD  FBD 	**NEXT**	NEXT 指令会标记 FOR 循环程序段的结束。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址	无

输入/输出	数据类型	操作数
INDX	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
INIT, FINAL	INT	VW, IW, QW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant

使用 FOR 和 NEXT 指令可在重复执行分配计数的循环中执行程序段。 每条 FOR 指令需要一条 NEXT 指令。 将 FOR-NEXT 循环置于最大嵌套深度为八层的 FOR-NEXT 循环内。

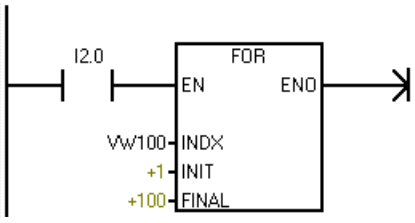
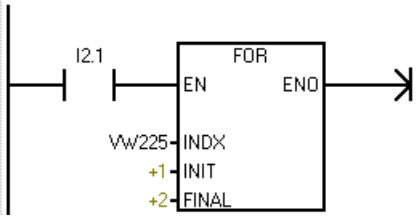
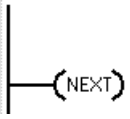
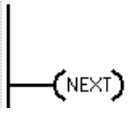
如果启用 FOR-NEXT 循环，则完成迭代操作之前会持续执行循环，除非在循环内部更改 FINAL 值。 在 FOR-NEXT 循环处于循环过程时可更改值。 再次启用循环时，会将 INIT 值复制到 INDX 值（当前循环编号）。

例如，假定 INIT 值为 1，FINAL 值为 10，则 FOR 指令和 NEXT 指令之间的指令将执行 10 次，INDX 值递增： 1, 2, 3, ... 10.

如果 INIT 值大于 FINAL 值，则不执行循环。 每次执行完 FOR 指令和 NEXT 指令之间的指令后，INDX 值递增，并将结果与最终值进行比较。 如果 INDX 大于最终值，则循环执行终止。

对于 STL，如果程序进入 FOR-NEXT 循环时逻辑堆栈的栈顶值为 1，则在程序退出 FOR-NEXT 循环时逻辑堆栈的栈顶值将为 1。

示例： FOR-NEXT 循环

LAD		STL
1 	I2.0 接通时，执行 100 次外部循环 (Network 1 - 4)。	**Network 1** **LD I2.0** **FOR VW100, +1, +100**
2 	I2.1 接通时，每执行一次外部循环会执行两次内部循环 (Network 2 - 3)。	**Network 2** **LD I2.1** **FOR VW225, +1, +2**
3 	内部循环结束	**Network 3** **NEXT**
4 	外部循环结束	**Network 4** **NEXT**

9.12.2 JMP（跳转至标号）

可在主程序、子例程、功能块或中断例程中使用 JMP（跳转）指令。JMP 及其对应的 LBL（标号）指令必须位于与主程序、子例程或中断例程相同的代码段中。

说明

无法从主程序跳转至子例程、功能块或中断例程中的标号。同样，也无法从子例程或中断例程跳转至该子例程或中断例程外的标号。

可在 SCR 程序段中使用跳转指令，但相应的标号指令必须位于同一 SCR 程序段中。

LAD/FBD	STL	描述
LAD n JMP FBD n JMP	**JMP N**	JMP（跳转）指令对程序中的标号 N 执行分支操作。
LAD n LBL FBD n LBL	**LBL N**	LBL（标号）指令用于标记跳转目的地 n 的位置。

输入/输出	数据类型	操作数
n	WORD	常数（0 到 255）

示例：跳转至标号

LAD		STL
SM0.2 / 4 JMP	如果保持性数据未丢失，则跳转至标号 4。	**Network 1** **LDN SM0.2** **JMP 4**
4 LBL	标号 4	**Network 2** **LBL 4**

9.12.3 SCR（顺控继电器）

SCR（顺控继电器）指令为 LAD、FBD 或 STL 程序提供简单但强大的状态控制编程技术。应用程序包含一系列必须重复执行的操作时，可以使用 SCR 来结构化程序，使其直接与应用程序相对应。因此，可以快速、轻松地设计和调试应用程序。

⚠ 警告

POU 中 S 位的用法

请勿在一个以上的 POU 中使用同一 S 位。例如，如果在主程序中使用 S0.1，则不要在子例程中使用它。

多个 POU 访问同一 S 位可能导致无法预料的过程操作，很可能会导致人员死亡或重伤。

请检查程序，确保多个 POU 不会访问的同一 S 位。

说明

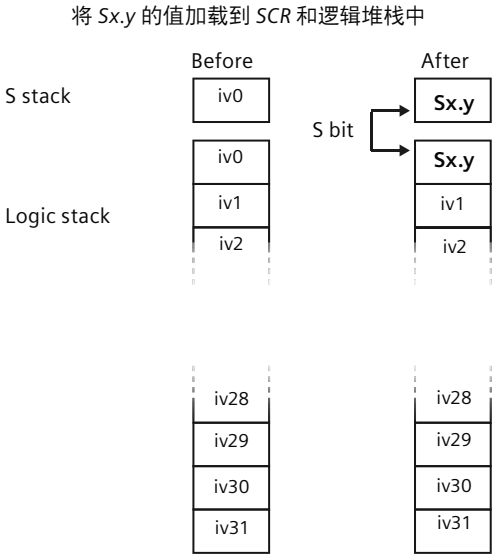
SCR 编程限制

- 无法跳入或跳出 SCR 段；但可以使用跳转和标号指令跳过 SCR 段或在 SCR 段内跳转。
- 不能在 SCR 段中使用结束指令。

LAD	FBD	STL	描述
S_bit SCR	S_bit SCR	**LSCR S_bit**	**SCR** 指令将该指令所引用的 S 位的值装载到 SCR 和逻辑堆栈。 得出的 SCR 堆栈的值会接通或断开 SCR 堆栈。SCR 堆栈的值会被复制到逻辑堆栈的栈顶，以便使 LAD 功能框或输出线圈可直接与左侧电源线相连，无需在前面使用触点指令。
S_bit SCRT	S_bit SCRT	**SCRT S_bit**	
	CSCRE	**CSCRE**	**SCRT** 指令标识要启用的 SCR 位（要设置的下一个 S_bit）。能流进入线圈或 FBD 功能框时，CPU 会开启引用的 S_bit，并会关闭 LSCR 指令（启用此 SCR 段的指令）的 S_bit。
SCRE	SCRE	**SCRE**	对于 STL 和 FBD，启用 **CSCRE**（有条件 SCR 结束）指令后，会终止执行 SCR 段。对于 LAD，置于 **SCRE** 线圈前的有条件触点会执行有条件 SCR 结束功能。 对于 STL 和 FBD，**SCRE**（无条件 SCR 结束）指令终止执行 SCR 段。对于 LAD，直接连接到电源线的 SCRE 线圈执行无条件 SCR 结束功能。

输入/输出	数据类型	操作数
S_bit	BOOL	S

S 堆栈和逻辑堆栈交互



图中显示了 S 堆栈和逻辑堆栈以及执行装载 SCR 指令的影响。

通过 SCR 段控制程序流

主程序由每次扫描 PLC 时都会按顺序执行一次的指令组成。对于许多应用程序，将主程序在逻辑上划分成一系列将步骤映射到受控过程中的操作步骤（例如一系列机器操作）可能比较恰当。

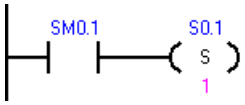
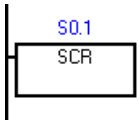
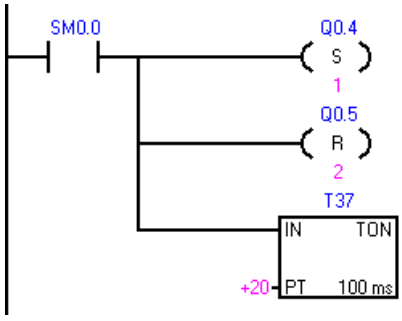
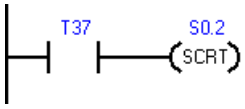
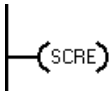
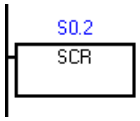
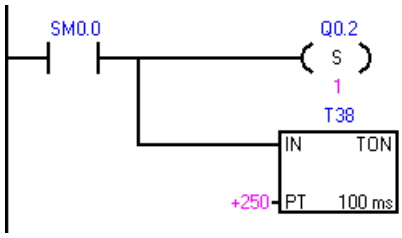
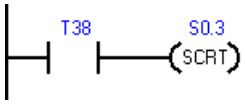
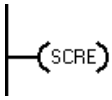
将程序在逻辑上划分为多个步骤的一种方法是使用 SCR 段。SCR 段可以将程序划分为单个顺序步骤流，或划分为可以同时激活的多个流。可以将单个流有条件地分为多个流，并可以将多个流有条件地条件重新合并为一个流。

SCR 操作

- SCR（装载 SCR）标记 SCR 段的开始， SCRE（结束 SCR）标记 SCR 程序段的结束。SCR 和 SCRE 指令之间的所有逻辑是否执行取决于 S 堆栈的值。SCRE 和下一条 SCR 指令之间的逻辑与 S 堆栈的值无关。
- SCRT（SCR 转换）将控制权从激活的 SCR 段转交给另一个 SCR 段。
  SCR 转换指令有能流时，执行该指令将复位当前激活的 SCR 段的 S 位，并会置位所引用段的 S 位。SCR 转换指令执行时，复位激活段的 S 位不会影响 S 堆栈。因此，SCR 段保持接通直至退出该段。
- 仅 STL 指令 CSRE（有条件 SCR 结束）存在激活的 SCR 段，而不在 CSRE 和 SCRE（SCR 结束）指令之间执行指令。有条件 SCR 结束指令不影响任何 S 位，也不会影响 S 堆栈。

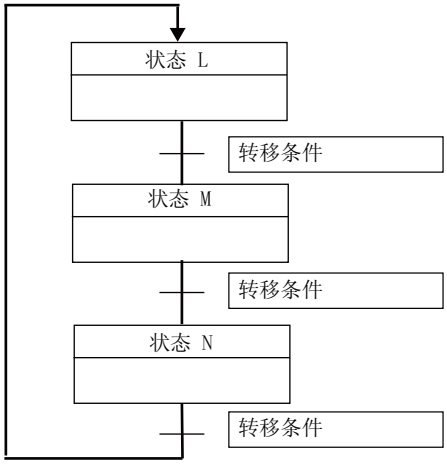
示例：SCR 顺序控制流

在以下示例程序中，首次扫描位 SM0.1 用于置位 S0.1，S0.1 在首次扫描时将是激活状态 1。2 秒延时后，T37 会导致转换为状态 2。该转换会禁用状态 1 SCR (S0.1) 段，并会激活状态 2 SCR (S0.2) 段。

LAD		STL
	首次扫描时启用状态 1。	Network 1 LD SM0.1 S S0.1, 1
	状态 1 控制区域开始。	Network 2 LSCR S0.1
	控制街道 1 的信号： 1. 置位：红灯接通。 2. 复位：黄灯和绿灯断开。 3. 启动 2 秒定时器。	Network 3 LD SM0.0 S Q0.4, 1 R Q0.5, 2 TON T37, +20
	2 秒延时后，转换为状态 2。	Network 4 LD T37 SCRT S0.2
	状态 1 的 SCR 区域结束。	Network 5 SCRE
	状态 2 控制区域开始。	Network 6 LSCR S0.2
	控制街道 2 的信号： 1. 置位：绿灯接通。 2. 启动 25 秒定时器。	Network 7 LD SM0.0 S Q0.2, 1 TON T38, +250
	25 秒延时后，转换为状态 3。	Network 8 LD T38 SCRT S0.3
	状态 2 的 SCR 区域结束。	Network 9 SCRE

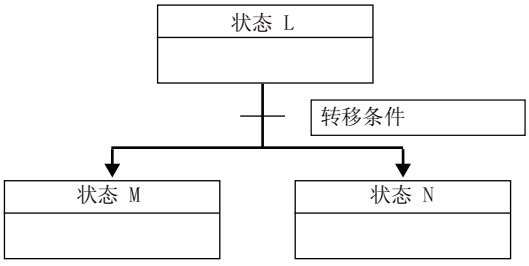
顺序控制流

顺序步骤定义明确的过程易于使用 SCR 段建模。例如，考虑包含 3 个步骤的循环过程，第三步完成时应返回第一步。



分散控制流

在许多应用程序中，一个顺序状态流必须分为两个或多个不同状态流。控制流分为多个控制流时，必须同时激活所有输出流。



如下例所示，可在 SCR 程序中使用由相同转换条件启用的多条 SCRT 指令将控制流划分为多个分支。

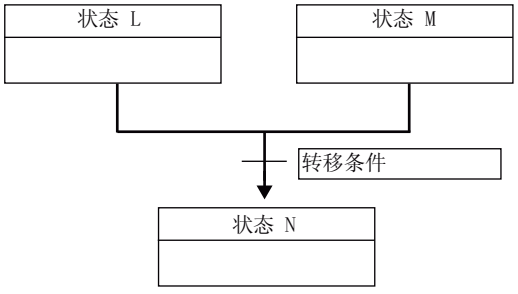
示例：SCR 分散流控制

LAD		STL
S3.4 SCR	状态 L 控制区域开始	Network 1 LSCR S3.4
M2.3 I2.1 S3.5 (SCRT) S6.5 (SCRT)	S3.5：转换为状态 M S6.5：转换为状态 N	Network 2 LD M2.3 A I2.1 SCRT S3.5 SCRT S6.5
(SCRE)	状态 L 的状态区域结束	Network 3 SCRE

合并流控制

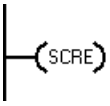
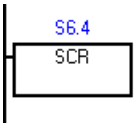
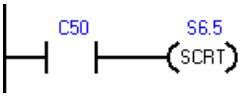
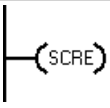
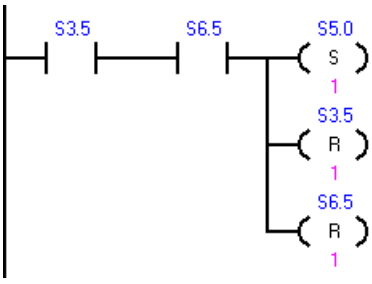
状态流合并时，在执行下一个状态之前，必须完成所有输入流。

可在 SCR 程序中通过从状态 L 转换到 L' 以及通过从状态 M 转换到 M' 来合并控制流。如下例所示，当代表 L' 和 M' 的两个 SCR 位均为真时，启用状态 N。



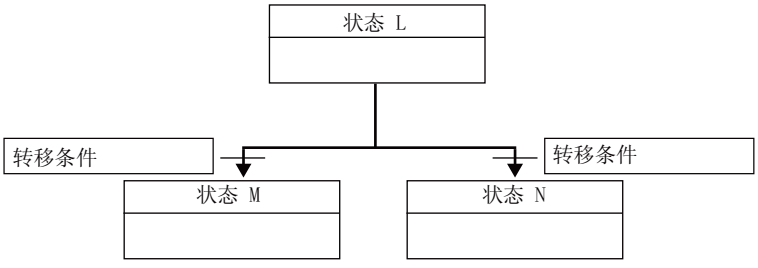
示例：SCR 合并流控制

LAD		STL
S3.4 SCR	状态 L 控制区域开始	Network 1 LSCR S3.4
V100.5 S3.5 (SCRT)	转换为状态 L'	Network 2 LD V100.5 SCRT S3.5

LAD		STL
	状态 L 的 SCR 区域结束	**Network 3** **SCRE**
	状态 M 控制区域开始	**Network 4** **LSCR S6.4**
	转换为状态 M'	**Network 5** **LD C50** **SCRT S6.5**
	状态 M 的 SCR 区域结束	**Network 6** **SCRE**
	状态 L' 和状态 M' 同时激活时： • 启用状态 N (S5.0) • 复位状态 L' (S3.5) • 复位状态 M' (S6.5)	**Network 7** **LD S3.5** **A S6.5** **S S5.0, 1** **R S3.5, 1** **R S6.5, 1**

控制流的分支，取决于转换条件

在其他情况下，控制流可能转到许多可能控制流之一，具体取决于哪个转换条件先变为真。



示例：SCR 分散流控制，具体取决于转换条件

LAD		STL
S3.4 SCR	状态 L 控制区域开始	**Network 1** **LSCR S3.4**
M2.3 S3.5 (SCRT)	转换为状态 M	**Network 2** **LD M2.3** **SCRT S3.5**
I3.3 S6.5 (SCRT)	转换为状态 N	**Network 3** **LD I3.3** **SCRT S6.5**
(SCRE)	状态 L 的 SCR 区域结束	**Network 4** **SCRE**

9.12.4 END、STOP 和 WDR（看门狗定时器复位）

LAD	FBD	STL	描述
(END)	END	**END**	有条件 END 指令基于前一逻辑条件终止当前扫描。可在主程序中使用有条件 END 指令，但不能在子例程、功能块或中断例程中使用。
(STOP)	STOP	**STOP**	有条件 STOP 指令通过将 CPU 从 RUN 模式切换到 STOP 模式来终止程序的执行。 如果在中断例程中执行 STOP 指令，则中断例程将立即终止，所有挂起的中断将被忽略。当前扫描周期中的剩余操作已完成，包括执行主用户程序。从 RUN 到 STOP 模式的转换是在当前扫描结束时进行的。
(WDR)	WDR	**WDR**	看门狗复位指令触发系统看门狗定时器，并将完成扫描的允许时间（看门狗超时错误出现之前）加 500 毫秒。

看门狗定时器操作

CPU 处于 RUN 模式时，默认状态下，主扫描的持续时间限制为 500 毫秒。如果主扫描的持续时间超过 500 毫秒，则 CPU 会自动切换为 STOP 模式，并会发出非致命错误 001AH（扫描看门狗超时）。

可以在程序中执行看门狗复位 (WDR) 指令来延长主扫描的持续时间。每次执行 WDR 指令时，扫描看门狗超时时间都会复位为 500 毫秒。

但是，主扫描的最大绝对持续时间为 5 秒。如果当前扫描持续时间达到 5 秒，CPU 会无条件地切换为 STOP 模式。

示例：STOP、END 和 WDR（看门狗复位）指令

LAD		STL
SM5.0 (STOP)	检测到 I/O 错误时，会强制切换为 STOP 模式。	**Network 1** **LD SM5.0** **STOP**
M5.6 (WDR)	M5.6 接通时，执行看门狗复位指令可将允许的扫描时间延长 500 毫秒。	**Network 1** **LD SM5.6** **WDR**
I0.0 (END)	I0.0 接通时，终止当前扫描。	**Network 1** **LD I.0** **END**

说明

如果预计扫描时间将超过 500 ms，或预计会出现大量可能阻止返回主扫描超过 500 ms 的中断活动，则应使用看门狗复位指令来重新触发看门狗定时器。

请小心使用看门狗复位指令。如果程序执行循环阻止扫描完成或过度延迟扫描的完成，则扫描周期完成之前禁止以下过程。

- 通信（自由端口模式除外）
- I/O 更新（立即 I/O 除外）
- 强制值更新
- SM 位更新（不更新 SM0、SM5 至 SM29）
- 运行时间诊断
- STOP 指令，在中断例程中使用时

9.12.5 GET_ERROR（获取非致命错误代码）

LAD/FBD	STL	说明
GET_ERROR EN ENO ECODE	GERR ECODE	获取非致命错误代码指令将 CPU 的当前非致命错误代码存储在分配给 ECODE 的位置。 而 CPU 中的非致命错误代码将在存储后清除。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址	无

输入/输出	数据类型	操作数
ECODE	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC

非致命运行时错误也会影响某些特殊的存储器错误标志地址，可配合 GET_ERROR 指令对这些地址进行评估，以确定运行时间故障的原因。 如果通用错误标志 SM4.3 = 1（运行时编程问题）激活，则可通过执行 GET_ERROR 标识特定错误。

非致命错误代码 0000H 指示目前不存在实际错误。 如果出现临时运行时间非致命错误，GET_ERROR（ECODE 输出）会生成非零错误值，然后下一次程序扫描会生成零 ECODE 值。

应使用比较逻辑将 ECODE 值保存到另一个存储单元。 之后，程序便可测试保存的错误代码值，并开始编程响应。

说明

ECODE 输出错误代码列在 PLC 非致命错误代码表（请参见以下参考内容）。 错误代码值为十六进制 (16#xxxx)。

另请参见

PLC 非致命错误代码 (页 1258)

PLC 非致命错误 SM 标志 (页 1261)

9.13 移位和循环移位

9.13.1 移位和循环移位

移位指令（仅说明大小为字节的 LAD 功能框，其他功能框类似）

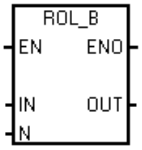
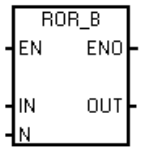
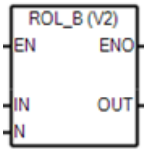
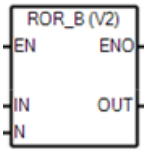
LAD/FBD	STL	移位类型	说明
SHL_B EN ENO IN OUT N SHR_B EN ENO IN OUT N	**SLB IN, N, OUT** **SRB IN, N, OUT**	左移字节 右移字节	移位指令将输入值 IN 的位值右移或左移位位置移位计数 N，然后将结果装载到分配给 OUT 的存储单元中。 对于每一位移出后留下的空位，移位指令会补零。如果移位计数 N 大于或等于允许的最大值（字节操作为 8、字操作为 16、双字操作为 32），则会按相应操作的最大次数对值进行移位。如果移位计数大于 0，则将溢出存储器位 SM1.1 会置位为移出的最后一位的值。如果移位操作的结果为零，则 SM1.0 零存储器位将置位。 字节操作是无符号操作。对于字操作和双字操作，使用有符号数据值时，也对符号位进行移位。
SHL_B (V2) EN ENO IN OUT N SHR_B (V2) EN ENO IN OUT N	历史 **SLB OUT, N** **SRB OUT, N**	左移字节 右移字节	
SHL_W SHR_W	**SLW IN, N, OUT** **SRW IN, N, OUT**	左移字 右移字	
	历史 **SLW OUT, N** **SRW OUT, N**	左移字 右移字	
SHL_DW SHR_DW	**SLD IN, N, OUT** **SRD IN, N, OUT**	左移双字 右移双字	
	历史 **SLD OUT, N** **SRD OUT, N**	左移双字 右移双字	

LAD/FBD	STL	移位类型	说明

ENO=0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址	• SM1.0 运算结果 = 零 • SM1.1 溢出（最后一位移出）

输入/输出	数据类型	操作数
IN	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数
	WORD, INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*LD、*AC、常数
	DWORD, DINT	ID、QD、VD、MD、SMD、SD、LD、AC、HC、*VD、*LD、*AC、常数
OUT	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC
	WORD, INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、*VD、*LD、*AC
	DWORD, DINT	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC
N	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数

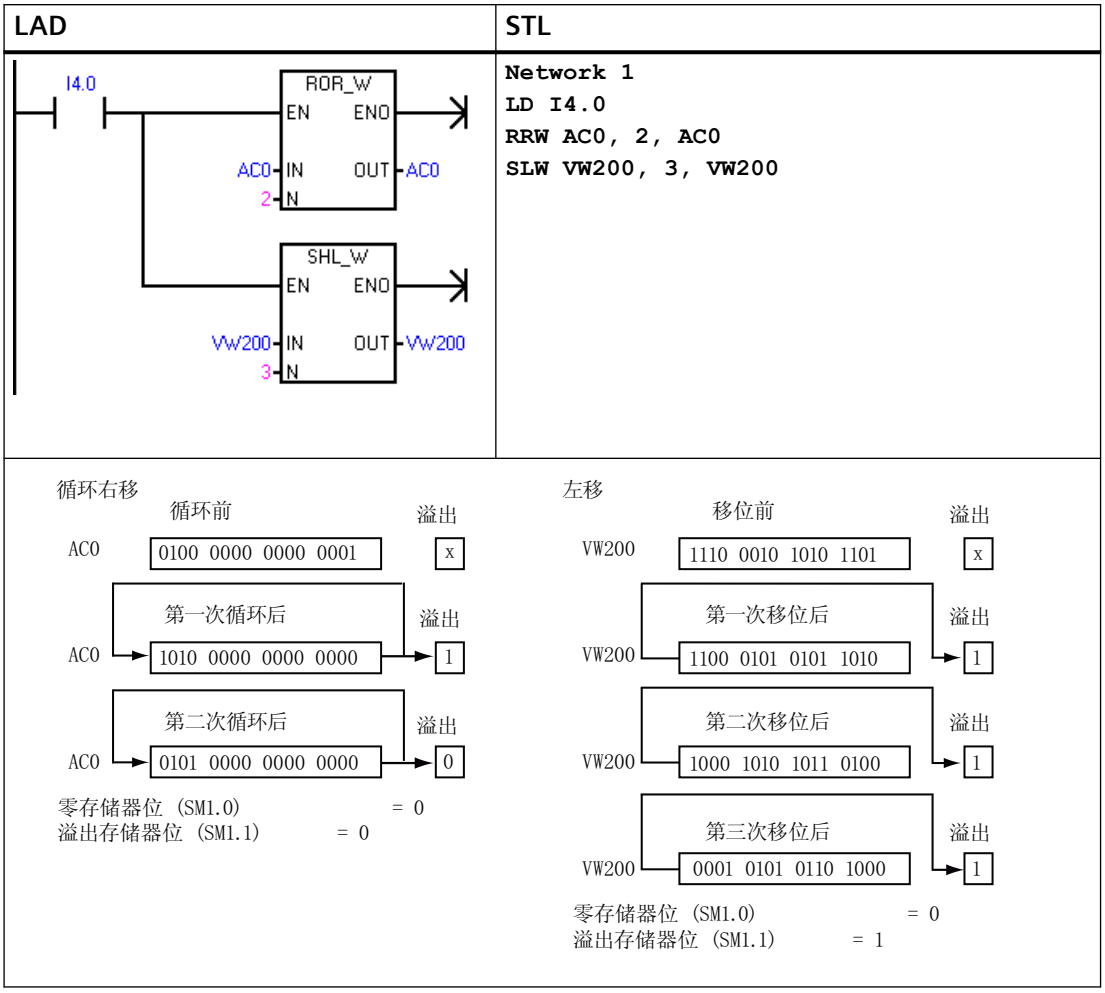
循环移位指令（仅说明大小为字节的 LAD 功能框，其他功能框类似）

LAD/FBD	STL	循环移位类型	说明
 	**RLB IN, N, OUT** **RRB IN, N, OUT**	循环左移字节 循环右移字节	循环移位指令将输入值 IN 的位值循环右移或循环左移位位置循环移位计数 N，然后将结果装载到分配给 OUT 的存储单元中。循环移位操作为循环操作。 如果循环移位计数大于或等于操作的最大值（字节操作为 8、字操作为 16、双字操作为 32），则 CPU 会在执行循环移位前对移位计数执行求模运算以获得有效循环移位计数。该结果为移位计数，字节操作为 0 至 7，字操作为 0 至 15，双字操作为 0 至 31。 如果循环移位计数为 0，则不执行循环移位操作。如果执行循环移位操作，则溢出位 SM1.1 将置位为循环移出的最后一位的值。 如果循环移位计数不是 8 的整倍数（对于字节操作）、16 的整倍数（对于字操作）或 32 的整倍数（对于双字操作），则将循环移出的最后一位的值复制到溢出存储器位 SM1.1。如果要循环移位的值为零，则零存储器位 SM1.0 将置位。 字节操作是无符号操作。对于字操作和双字操作，使用有符号数据类型时，也会对符号位进行循环移位。
 	历史 **RLB OUT, N** **RRB OUT, N**	循环左移字节 循环右移字节	
ROL_W ROR_W	**RLW IN, N, OUT** **RRW IN, N, OUT**	循环左移字 循环右移字	
	历史 **RLW OUT, N** **RRW OUT, N**	循环左移字 循环右移字	
ROL_DW ROR_DW	**RLD IN, N, OUT** **RRD IN, N, OUT**	循环左移双字 循环右移双字	
	历史 **RLD OUT, N** **RRD OUT, N**	循环左移双字 循环右移双字	

ENO=0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址	• SM1.0 运算结果 = 零 • SM1.1 溢出（最后一位移出）

输入/输出	数据类型	操作数
IN	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数
	WORD, INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*LD、*AC、常数
	DWORD, DINT	ID、QD、VD、MD、SMD、SD、LD、AC、HC、*VD、*LD、*AC、常数
OUT	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC
	WORD, INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、*VD、*LD、*AC
	DWORD, DINT	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC
N	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数

示例：移位和循环移位指令



9.13.2 移位寄存器位

移位寄存器位指令将位值移入移位寄存器。该指令提供了排序和控制产品流或数据的简便方法。使用该指令在每次扫描时将整个寄存器移动一位。

LAD/FBD	STL	说明
SHRB EN ENO DATA S_BIT N	**SHRB DATA, S_bit, N**	移位寄存器位指令将 DATA 的位值移入移位寄存器。S_BIT 指定移位寄存器最低有效位的位置。N 指定移位寄存器的长度和移位方向（正向移位 = N，反向移位 = -N）。 将 SHRB 指令移出的每个位值复制到溢出存储器位 SM1.1 中。 移位寄存器位由最低有效位 S_BIT 位置和长度 N 指定的位数定义。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围 • 0092H 计数字段中有错误	• SM1.1 溢出（最后一位移出）

输入/输出	数据类型	操作数
DATA、S_bit	BOOL	I、Q、V、M、SM、S、T、C、L
N	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数

可使用以下公式计算移位寄存器的最高有效位地址 (MSB.b)：

MSB.b = *[(S_BIT 字节) + ([N] - 1 + (S_BIT 位))/8].[除以 8 后的余数]*

例如：如果 S_BIT 为 V33.4，N 为 14，则以下计算的结果将是 MSB.b 为 V35.1。

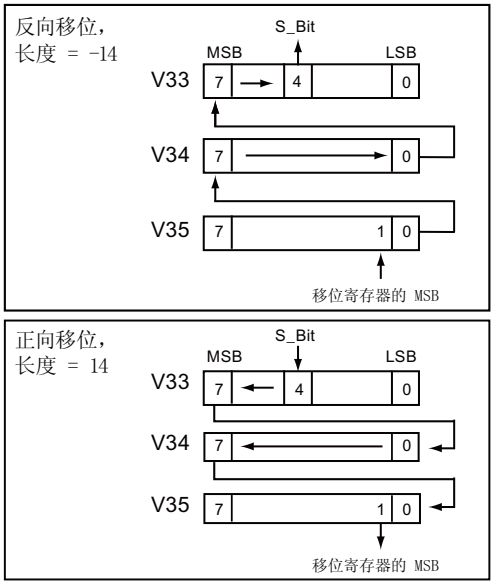
MSB.b = V33 + ([14] - 1 +4)/8

= V33 + 17/8

= V33 + 2，余数为 1

= V35.1

下图显示 N 为负值和正值时的移位情况。



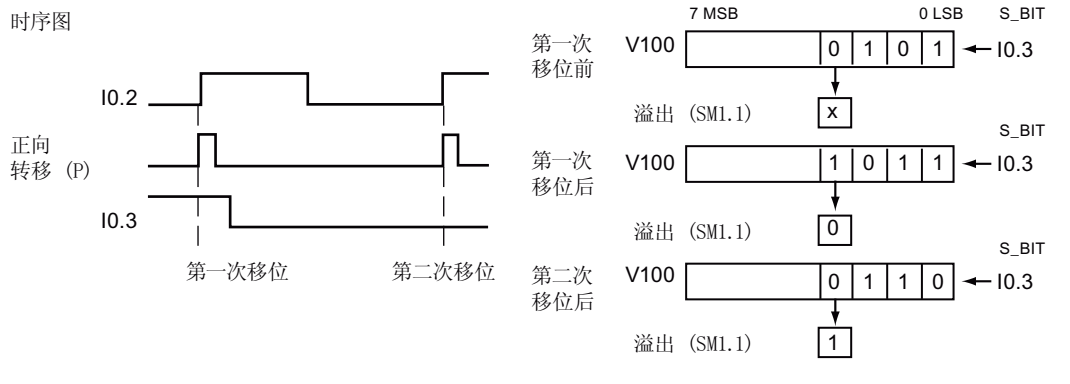
反向移位操作用长度 N 的负值表示。将 DATA 的输入值移入移位寄存器的最高有效位，然后移出由 S_BIT 指定的最低有效位位置。然后将移出的数据放在溢出存储器位 SM1.1 中。

正向移位操作用长度 N 的正值表示。将 DATA 的输入值移入由 S_BIT 指定的最低有效位位置，然后移出移位寄存器的最高有效位。然后将移出的位值放在溢出存储器位 SM1.1 中。

由 N 指定的移位寄存器的最大长度为 64 位（正向或反向）。

示例：SHRB 指令

LAD	STL
I0.2 P SHRB EN ENO I0.3 DATA V100.0 S_BIT +4 N	Network 1 LD I0.2 EU SHRB I0.3, V100.0, +4



9.14 字符串

9.14.1 字符串（获取长度、复制和连接）

SLEN（字符串长度）

LAD/FBD	STL	描述
STR_LEN EN ENO IN OUT	SLEN IN, OUT	字符串长度指令返回由 IN 指定的字符串长度（字节）。 注：因为中文字符并非由单字节表示，STR_LEN 函数不会返回包含中文字符的字符串中的字符数。

ENO = 0 时的错误条件	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围	无

输入/输出	数据类型	操作数
IN	STRING	VB, LB, *VD, *LD, *AC, 常数字符串
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC

SCPY 和 SCAT（字符串复制和字符串连接）

LAD/FBD	STL	描述
STR_CPY EN ENO IN OUT	**SCPY IN, OUT**	字符串复制指令将由 IN 指定的字符串复制到由 OUT 指定的字符串。
STR_CAT EN ENO IN OUT	**SCAT IN, OUT**	字符串连接指令将由 IN 指定的字符串附加到由 OUT 指定的字符串的末尾。

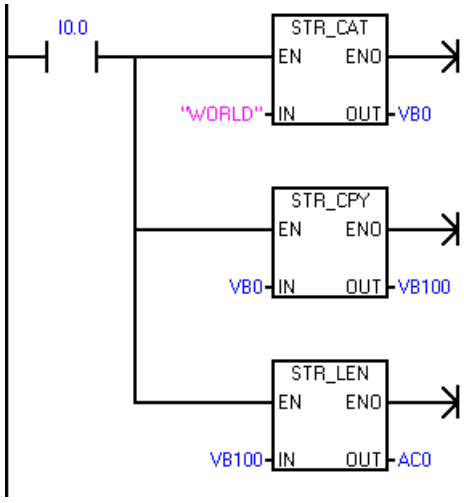
注：STR_CPY 和 STR_CAT 指令作用对象是字节而不是字符。因为中文字符并非由单字节表示，所以 STR_CPY 和 STR_CAT 指令作用于包含中文字符的字符串时，可能出现非预期的结果。如果您知道字符包含的字节数，则可以在使用 STR_CPY 和 STR_CAT 指令时使用正确的字节数。

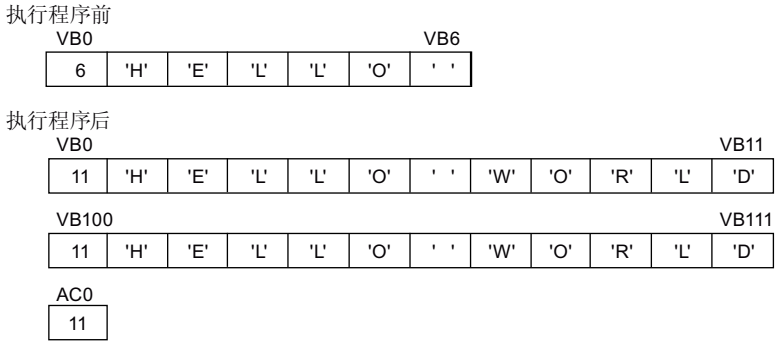
ENO = 0 时的错误条件	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围	无

注：如果输入字符串 (IN) 的长度超出输出变量 (OUT) 的长度，则运行期间会报错误代码 0091H。

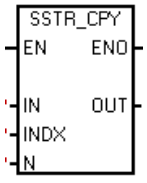
输入/输出	数据类型	操作数
IN	STRING	VB, LB, *VD, *LD, *AC, 常数字符串
OUT	STRING	VB, LB, *VD, *LD, *AC

示例：字符串连接指令、字符串复制指令和字符串长度指令

LAD		STL
	1. 将字符串“WORLD”附加到 VB0 中的字符串。 2. 将 VB0 中的字符串复制到 VB100 中的新字符串。 3. 获取从 VB100 开始的字符串的长度。	**Network 1** **LD I0.0** **SCAT "WORLD", VB0** **SCPY VB0, VB100** **SLEN VB100, AC0**



9.14.2 从字符串中复制子字符串

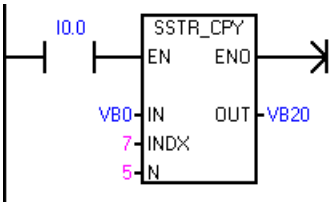
LAD/FBD	STL	描述
	**SSCPY IN, INDX, N, OUT**	从字符串中复制子字符串指令从 IN 指定的字符串中将从索引 INDX 开始的指定数目的 N 个字符复制到 OUT 指定的新字符串中。 注：SSTR_CPY 指令作用对象是字节而不是字符。因为中文字符并非由单字节表示，所以 SSTR_CPY 指令作用于包含中文字符的字符串时，可能出现非预期的结果。如果您知道字符包含的字节数，则可以在使用 SSTR_CPY 指令时使用正确的字节数。

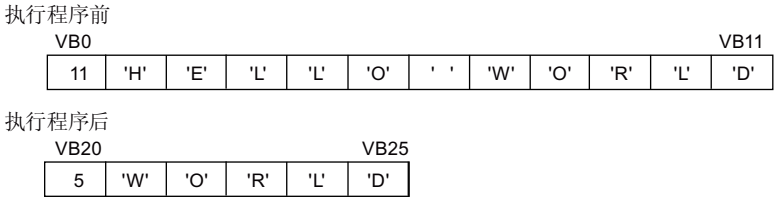
ENO=0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围 • 009BH 索引 = 0	无

注：如果输入字符串 (IN) 的长度超出输出变量 (OUT) 的长度，则运行期间会报错误代码 0091H。

输入/输出	数据类型	操作数
IN	STRING	VB, LB, *VD, *LD, *AC，常数字符串
OUT	STRING	VB, LB, *VD, *LD, *AC
INDX, N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC、常数

示例：复制子字符串指令

LAD		STL
	按字节计数，从 VB0 中字符串的第 7 个字节开始，复制 5 个字符到 VB20 中的新字符串。	**Network 1** **LD I0.0** **SSCPY VB0, 7, 5, VB20**



9.14.3 在字符串中查找字符串和第一个字符

LAD/FBD	STL	说明
STR_FIND EN ENO IN1 OUT IN2	**SFND IN1, IN2, OUT**	STR_FIND 在字符串 IN1 中搜索第一次出现的字符串 IN2。 从 OUT 的初始值指定的起始位置（在执行 STR_FIND 之前，起始位置必须位于 1 至 IN1 字符串长度范围内）开始搜索。 如果找到与字符串 IN2 完全匹配的字符序列，则将字符序列中第一个字符在 IN1 字符串中的位置写入 OUT。 如果在字符串 IN1 中没有找到 IN2 字符串，则将 OUT 设置为 0。
CHR_FIND EN ENO IN1 OUT IN2	**CFND IN1, IN2, OUT**	CHR_FIND 在字符串 IN2 中搜索第一次出现的字符串 IN1 字符集中的任意字符。 从 OUT 的初始值指定的起始位置（在执行 CHR_FIND 之前，起始位置必须位于 1 至 IN1 字符串长度范围内）开始搜索。 如果找到匹配字符，则将字符位置写入 OUT。 如果没有找到匹配字符，OUT 设置为 0。

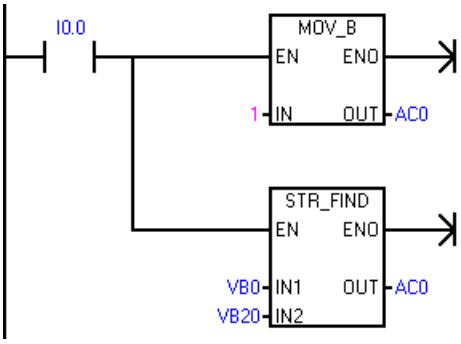
注： 因为中文字符并非由单字节表示，并且字符串指令作用于字节而不是字符，所以 STR_FIND 和 CHR_FIND 指令作用于包含中文字符的字符串时，可能出现非预期的结果。

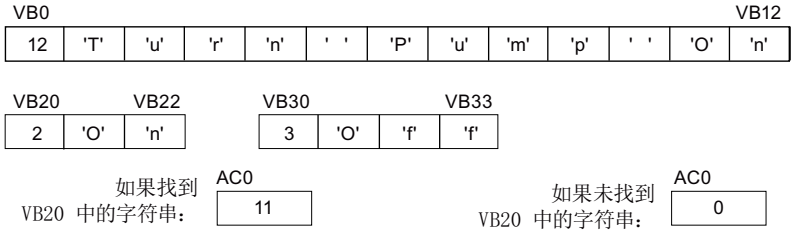
ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围 • 009BH 索引 = 0	无

输入/输出	数据类型	操作数
IN1, IN2	STRING	VB, LB, *VD, *LD, *AC，常数字符串
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC

示例：“在字符串中查找字符串”指令

使用 VB0 中存储的字符串作为泵开/关命令。字符串“On”存储在 VB20 中，字符串“Off”存储在 VB30 中。“在字符串中查找字符串”指令的结果存储在 AC0（OUT 参数）中。如果结果不为 0，则说明在命令字符串中找到字符串“On”(VB12)。

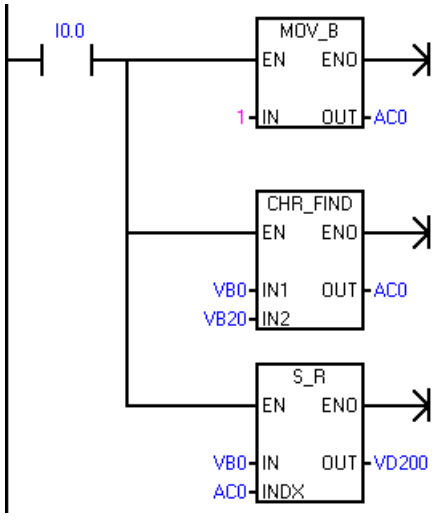
LAD		STL
	1. 将 AC0 设置为 1。（AC0 用作 OUT 参数。） 2. 在 VB0 中的字符串中，从第一个位置 (AC0=1) 开始搜索 VB20 中的字符串（“On”）。	**Network 1** **LD I0.0** **MOVB 1, AC0** **SFND VB0, VB20, AC0**



示例：“在字符串中查找字符”指令

存储在 VB0 中的字符串包含温度。 IN1 中的字符串常数提供可标识字符串中的温度数字的所有数字字符（包括 0-9、+ 和 -）。 执行 CHR_FIND 可找到字符“9”在 VB0 字符串中的起始位置，然后执行 S_R 将实数字符转换为实数值。 VD200 用于存储温度的实数值。

LAD



1. 将 AC0 设置为 1。（AC0 用作 OUT 参数，并指向字符串中的第一个字符位置。）
2. 在 VB0 中存储的字符串中查找第一个数字字符。
3. 将字符串转换为实数值。

STL

```
Network 1
LD I0.0
MOVB 1, AC0
CFND VB0, VB20, AC0
STR VB0, AC0, VD200
```

VB0											VB11
11	'T'	'e'	'm'	'p'	' '	' '	'9'	'8'	'.'	'6'	'F'

VB20												VB32
12	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'	'0'	'+'	'-'

存储在 VB0 中的温度的起始位置： AC0 = 7

温度的实数值： VD200 = 98.6

9.15 表

9.15.1 添表

LAD/FBD	STL	描述
AD_T_TBL EN ENO DATA TBL	**ATT DATA, TBL**	添表指令向表格 TBL 中添加字值 DATA。表格中的第一个值为最大表格长度 TL。第二个值是条目计数 EC，用于存储表格中的条目数，并自动更新。新数据填加到表格中最后一个条目之后。每次向表格中填加新数据时，条目计数将加 1。 一个表格最多可有 100 个数据条目。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围 • SM1.4 表格溢出	• 如果表格溢出，SM1.4 将设置为 1

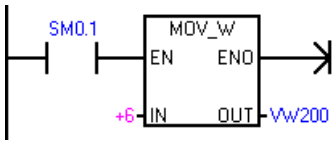
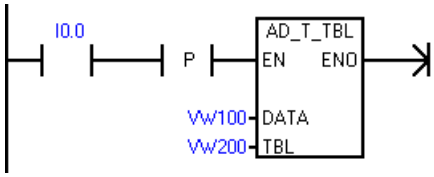
输入/输出	数据类型	操作数
DATA	INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*LD、*AC、常数
TBL	WORD	IW、QW、VW、MW、SMW、SW、T、C、LW、*VD、*LD、*AC **注：**此指令的 TBL 没有 DB2 区域支持。

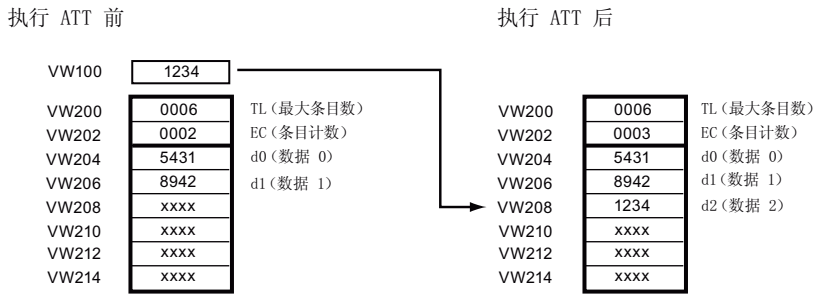
说明

要创建表格，首先创建用于表示最大表格条目数的条目。如果不创建此条目，则不能在表格中创建任何条目。

必须使用沿触发指令激活所有表格读取指令和表格写入指令。

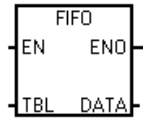
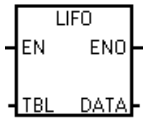
示例：添表指令

LAD		STL
	仅在第一次扫描时，将最大表格长度 6 装载到 VW200。	**Network 1** **LD SM0.1** **MOVW +6, VW200**
	当 I0.0 转换为 1 时，将第三个数据值（VW100 中）添加到 VW200 中的表格。之前已将两个数据条目存储在表格中，该表格最多可容纳六个条目。	**Network 2** **LD I0.0** **ATT VW100, VW200**



9.15.2 先进先出和后进先出

表格 9-21 FIFO 和 LIFO 指令

LAD/FBD	STL	说明
	**FIFO TBL, DATA**	先进先出指令将表中的最早（或第一个）条目移动到输出存储器地址，具体操作是移走指定表格 (TBL) 中的第一个条目并将该值移动到 DATA 指定的位置。 表格中的所有其它条目向上移动一个位置。 每次执行 FIFO 指令时，表中的条目计数值减 1。
	**LIFO TBL, DATA**	后进先出指令将表中的最新（或最后一个）条目移动到输出存储器地址，具体操作是移走表格 (TBL) 中的最后一个条目并将该值移动到 DATA 指定的位置。 每次执行 LIFO 指令时，表中的条目计数值减 1。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围 • SM1.5： 尝试删除空表格中的条目	• SM1.5：尝试删除空表格中的条目

输入/输出	数据类型	操作数
TBL	WORD	IW、QW、VW、MW、SMW、SW、T、C、LW、*VD、*LD、*AC
DATA	INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AQW、*VD、*LD、*AC

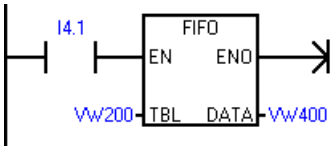
说明

所有表格读取和表格写入指令都必须通过沿触发指令激活。

要创建表格，必须先创建用于表示最大表格条目数的条目，然后才能将其它条目放入表格中。

示例： FIFO 指令

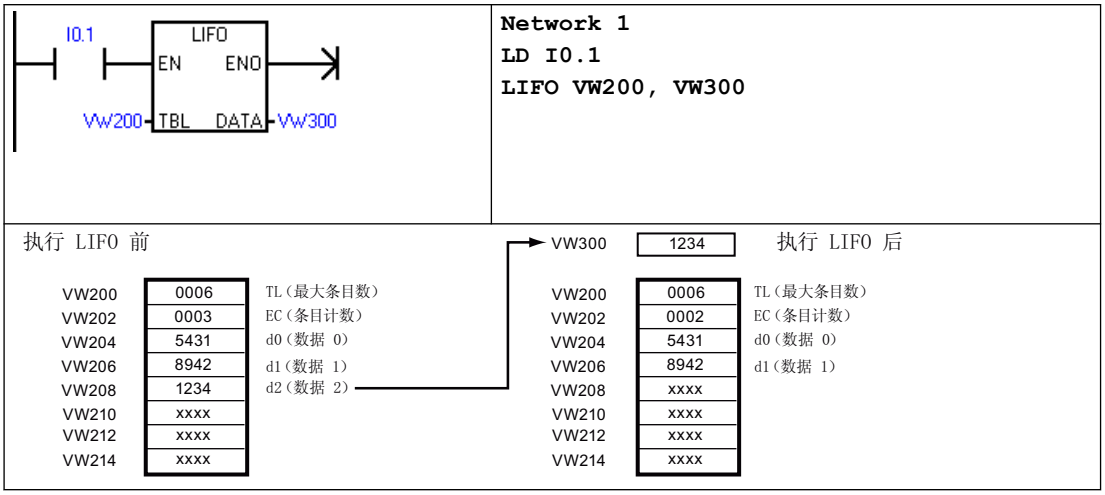
LAD



STL

```
Network 1
LD I4.1
FIFO VW200, VW400
```

示例： LIFO 指令



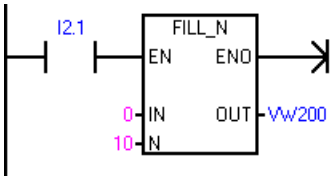
9.15.3 存储器填充

LAD/FBD	STL	说明
FILL_N EN ENO IN OUT N	**FILL IN, OUT, N**	存储器填充指令使用地址 IN 中存储的字值填充从地址 OUT 开始的 N 个连续字。 N 取值范围是 1 到 255。

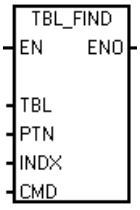
ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围	无

输入/输出	数据类型	操作数
IN	INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*LD、*AC、常数
N	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数
OUT	INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AQW、*VD、*LD、*AC

示例： 存储器填充指令

LAD	STL
	**Network 1** **LD I2.1** **FILL +0, VW200, 10**

9.15.4 查表

LAD/FBD	STL	说明
	**FND= TBL, PTN, INDX** **FND<> TBL, PTN, INDX** **FND< TBL, PTN, INDX** **FND> TBL, PTN, INDX**	查表指令在表格中搜索与搜索条件匹配的数据。查表指令由表格条目 INDX 开始，在表格 TBL 中搜索与 CMD 定义的搜索标准相匹配的数据值或模式 PTN。指令参数 CMD 的 1 到 4 的数字值分别对应于 =、<>、< 和 >。 如果找到匹配条目，INDX 将指向表中的该匹配条目。要查找下一个匹配条目，再次调用查表指令之前，必须先使 INDX 增加 1。如果未找到匹配条目，则 INDX 值等于条目计数。 一个表格最多可有 100 个数据条目。数据条目（搜索区域）编号为 0 到 99（最大值）。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围	无

输入/输出	数据类型	操作数
TBL	WORD	IW、QW、VW、MW、SMW、SW、T、C、LW、*VD、*LD、*AC
PTN	INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*LD、*AC、常数
INDX	WORD	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、*VD、*LD、*AC
CMD	BYTE	常数： • 1 = 相等 (=) • 2 = 不相等 (<>) • 3 = 小于 (<) • 4 = 大于 (>)

说明

查表指令与使用添表指令、后进先出指令和先进先出指令生成的表配合使用时，条目计数和数据条目数直接对应。添表指令、后进先出指或先进先出指令需要表示最大条目数的字，但查表指令不需要表示最大条目数的字。请参见下图。

因此，应将查找指令的 TBL 操作数的地址设置得比相应的添表指令、后进先出指令或先进先出指令的 TBL 操作数高一个字（两个字节）。

ATT、LIFO、FIFO 和 TBL_FIND 指令的表格格式差异

ATT、LIFO 及 FIFO 的表格式

VW200	0006	TL（最大条目数）
VW202	0006	EC（条目计数）
VW204	xxxx	d0（数据 0）
VW206	xxxx	d1（数据 1）
VW208	xxxx	d2（数据 2）
VW210	xxxx	d3（数据 3）
VW212	xxxx	d4（数据 4）
VW214	xxxx	d5（数据 5）

TBL_FIND 的表格式

VW202	0006	EC（条目计数）
VW204	xxxx	d0（数据 0）
VW206	xxxx	d1（数据 1）
VW208	xxxx	d2（数据 2）
VW210	xxxx	d3（数据 3）
VW212	xxxx	d4（数据 4）
VW214	xxxx	d5（数据 5）

示例：查表指令

LAD	STL
I2.1 TBL_FIND EN ENO VW202 TBL 16#3130 PTN AC1 INDX 1 CMD	程序段 1 LD I2.1 FND= VW202, 16#3130, AC1

I2.1 接通时，在表中搜索等于 3130（十六进制）的值。

VW202	0006	EC（条目计数）
VW204	3133	d0（数据 0）
VW206	4142	d1（数据 1）
VW208	3130	d2（数据 2）
VW210	3030	d3（数据 3）
VW212	3130	d4（数据 4）
VW214	4541	d5（数据 5）

如果该表是通过 ATT、LIFO 和 FIFO 指令创建的，
则 VW200 将包含允许的最大条目数，并且“查找”(Find) 指令不需要使用 VW200。

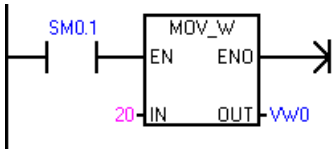
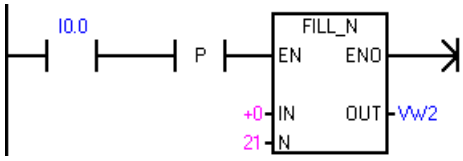
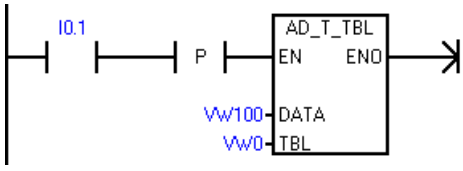
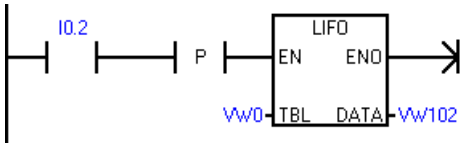
AC1	0	AC1 必须设为 0 才能从表顶部进行搜索。
执行表搜索 AC1	2	AC1 中包含的数据条目编号对应于表中的第一个匹配条目 (d2)。
AC1	3	先将 INDX 值加 1，然后再搜索表中的剩余条目。
执行表搜索 AC1	4	AC1 中包含的数据条目编号对应于表中的第二个匹配条目 (d4)。
AC1	5	先将 INDX 值加 1，然后再搜索表中的剩余条目。
执行表搜索 AC1	6	AC1 包含的值等于条目计数。 已搜索整个表，未发现其它匹配条目。
AC1	0	再次搜索表之前，必须先将 INDX 值复位为 0。

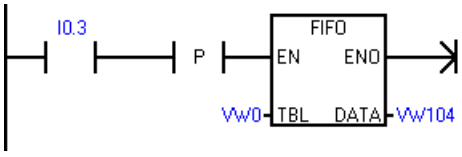
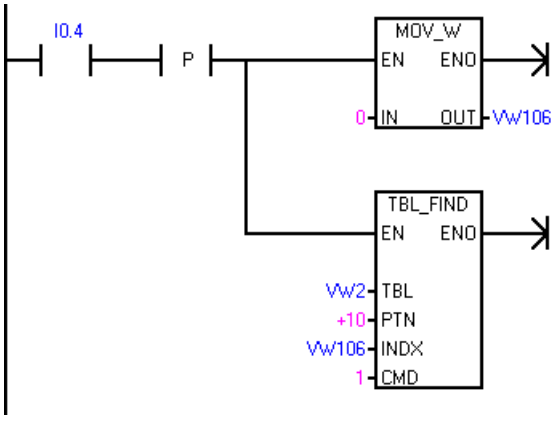
示例：表格

下列程序创建一个包含 20 个条目的表格。表格的第一个存储单元存储表格长度（在本例中为 20）。第二个存储单元存储当前表格条目数。其它位置存储各条目。一个表格最多可有 100 个条目。其中不包括用于定义最大表格长度或实际条目数（在本例中为 VW0 和 VW2）的参数。每次执行指令时，CPU 会自动对表格中的实际条目数（在本例中为 VW2）执行递增或递减操作。

在使用表格之前，必须指定最大表格条目数。否则，您将无法在表中添加条目。此外，还要确保使用边沿触发指令激活所有读取和写入命令。

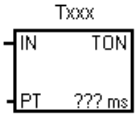
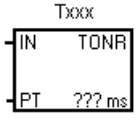
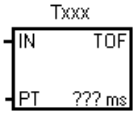
要搜索表格，在执行查找操作之前，必须将索引 (VW106) 设置为 0。如果找到匹配条目，索引存储该表格条目编号；如果未找到匹配条目，则索引为表格的当前条目计数 (VW2)。

LAD		STL
	创建表格，表格包含 20 个条目，从存储器位置 4 开始。 • 在第一次扫描时，定义表格的最大长度。	**Network 1** **LD SM0.1** **MOVW +20, VW0**
	通过输入 I0.0 复位表格。 • 在 I0.0 的上升沿，使用“+0”填充从 VW2 开始的存储单元。	**Network 2** **LD I0.0** **EU** **FILL +0, VW2, 21**
	通过输入 I0.1 将值写入表格。 • 在 I0.1 的上升沿，将存储单元 VW100 的值复制到表格。	**Network 3** **LD I0.1** **EU** **ATT VW100, VW0**
	通过输入 I0.2 读取表中的最后一个值。 • 将表中的最后一个值移动到 VW102 位置。这样会使条目数减少。在 I0.2 的上升沿，将表中的最后一个值移入 VW102。	**Network 4** **LD I0.2** **EU** **LIFO VW0, VW102**

LAD		STL
	通过输入 I0.3 读取表中的第一个值。 • 将表中的第一个值移动到 VW104 位置。这样会使条目数减少。在 I0.3 的上升沿，将表中的第一个值移入 VW104。	**Network 5** **LD I0.3** **EU** **FIFO VW0, VW104**
	在表中搜索值为 10 的第一个位置。 • 在 I0.4 的上升沿，复位索引指针。 • 查找等于 10 的表格条目。	**Network 6** **LD I0.4** **EU** **MOVW +0, VW106** **FND= VW2, +10, VW106**

9.16 定时器

9.16.1 定时器指令

LAD/FBD	STL	说明
	**TON Txxx, PT**	**TON** 接通延时定时器用于测定单独的时间间隔。
	**TONR Txxx, PT**	**TONR** 保持型接通延时定时器用于累积多个定时时间间隔的时间值。
	**TOF Txxx, PT**	**TOF** 断开延时定时器用于在 OFF（或 FALSE）条件之后延长一定时间间隔，例如冷却电机的延时。

输入/输出	数据类型	操作数
Txxx	WORD	定时器编号（T0 至 T511）
IN	BOOL	I、Q、V、M、SM、S、T、C、L、能流
PT	INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*LD、*AC、常数

定时器分辨率

TON、TONR 和 TOF 定时器提供三种分辨率。分辨率由定时器编号确定如下所示。当前值的每个单位均为时基的倍数。例如，使用 10 ms 定时器时，计数 50 表示经过的时间为 500 ms。

Txxx 定时器编号分配决定定时器的分辨率。分配有效的定时器编号后，分辨率会显示在 LAD 或 FBD 定时器功能框中。

定时器编号和分辨率选项

定时器类型	分辨率	最大值	定时器号
TON、TOF	1 ms	32.767 s	T32、T96、T262-T267
	10 ms	327.67 s	T33-T36、T97-T100、T292-T315
	100 ms	3276.7 s	T37-T63、T101-T255、T414-T511
TONR	1 ms	32.767 s	T0、T64、T256-T261
	10 ms	327.67 s	T1-T4、T65-T68、T268-T291
	100 ms	3276.7 s	T5-T31、T69-T95、T316-T413

说明

避免定时器编号冲突

同一个定时器编号不能同时用于 TON 和 TOF 定时器。例如，不能同时使用 TON T32 和 TOF T32。

说明

要确保最小时间间隔，请将预设值 (PV) 增大 1。

例如：使用 100 ms 定时器时，为确保最小时间间隔至少为 2100 ms，则将 PV 设置为 22。

TON 和 TONR 定时器操作

TON 和 TONR 指令在使能输入 IN 接通时开始计时。当前值等于或大于预设时间时，定时器位置为接通。

- 使能输入置为断开时，清除 TON 定时器的当前值。
- 使能输入置为断开时，保持 TONR 定时器的当前值。输入 IN 置为接通时，可以使用 TONR 定时器累积时间。使用复位指令 (R) 可清除 TONR 的当前值。
- 达到预设时间后，TON 和 TONR 定时器继续定时，直到达到最大值 32,767 时才停止定时。

TOF 定时器操作

TOF 指令用于使输出在输入断开后延迟固定的时间再断开。当使能输入接通时，定时器位立即接通，当前值设置为 0。当输入断开时，计时开始，直到当前时间等于预设时间时停止计时。

- 达到预设值时，定时器位断开，当前值停止递增；但是，如果在 TOF 达到预设值之前使能输入再次接通，则定时器位保持接通。
- 要使 TOF 定时器开始定时断开延时时间间隔，使能输入必须进行接通-断开转换。
- 如果 TOF 定时器在 SCR 区域中，并且 SCR 区域处于未激活状态，则当前值设置为 0，定时器位断开且当前值不递增。

类型	当前值 >= 预设值	使能输入 IN 的状态	上电循环/首次扫描
TON	定时器位接通 当前值继续定时到 32,767	ON：当前值 = 定时值 OFF：定时器位断开，当前值 = 0	定时器位 = OFF 当前值 = 0
TONR[1]	定时器位接通 当前值继续定时到 32,767	ON：当前值 = 定时值 OFF：定时器位和当前值保持最后状态和值	定时器位 = OFF 当前值可以保持 [1]

类型	当前值 >= 预设值	使能输入 IN 的状态	上电循环/首次扫描
TOF	定时器位断开 当前值 = 预设值，停止定时	ON：定时器位接通，当前值 = 0 OFF：在接通-断开转换之后，定时器开始定时	定时器位 = OFF 当前值 = 0
[1] 可将保持性定时器的当前值指定为在整个上电循环中可保持。有关详细信息，请参见组态保持范围。			

说明

复位指令与定时器指令配合使用

只能用复位 (R) 指令复位 TONR 定时器。

TON 和 TOF 定时器可通过定时器的使能输入和复位 (R) 指令两种方法复位。

复位指令执行下列操作：

- 定时器位 = OFF
- 定时器当前值 = 0
- 复位后，TOF 定时器在使能输入从接通转换为断开时才会重新启动断开延时定时器。

参见

定时器编程提示和示例 (页 479)

9.16.2 定时器编程提示和示例

定时器类型

您可利用定时器实现时基计数功能。S7-200 SMART 指令集提供三种不同类型的定时器。

- 接通延时定时器 (TON)，用于单间隔定时
- 保持型接通延时定时器 (TONR)，用于累积一定数量的定时间隔
- 断开延时定时器 (TOF)，用于在断开（或 FALSE）条件之后延长一定时间，例如电机关闭后使电机冷却

寻址定时器值

T 编号的含义取决于程序中的上下文。

- 分配给定时器功能框的“T37”标识要使用哪个定时器。
- 分配给常开触点的“T37”寻址布尔型 T37 定时器位。
- 分配给整数操作的“T37”作为数据字寻址 T37 当前时间值。

1 毫秒分辨率

1 毫秒定时器记录自活动 1 毫秒定时器启用以来经过的 1 毫秒定时器时间间隔的数目。执行定时器指令即开始计时；但是，1 毫秒定时器每毫秒更新一次（定时器位及定时器当前值），不与扫描周期同步。换言之，在超过 1 毫秒的扫描过程中，定时器位和定时器当前值将多次更新。

定时器指令用于打开和复位定时器，如果是 TONR 定时器，则用于关闭定时器。

因为可在一毫秒内的任意时刻启动定时器，预设值必须设为比最小所需定时器间隔大的一个时间间隔。例如，使用 1 毫秒定时器时，为了保证时间间隔至少为 56 毫秒，则预设时间值应设为 57。

10 毫秒分辨率

10 毫秒定时器记录自活动 10 毫秒定时器启用以来经过的 10 毫秒定时器时间间隔的数目。执行定时器指令即开始计时；但是，在每次扫描周期开始时更新 10 毫秒定时器（换言之，在整个扫描过程中，定时器当前值及定时器位保持不变）,更新方法是将积累的 10 毫秒间隔数（自前一次扫描开始）加到活动定时器的当前值。

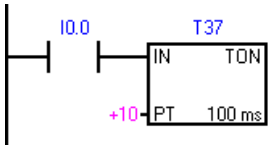
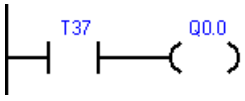
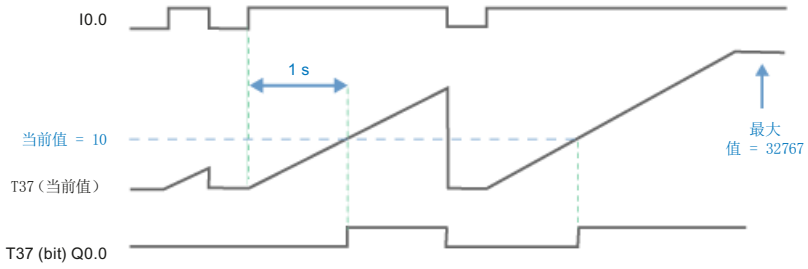
因为可在 10 毫秒内的任意时刻启动定时器，预设值必须设为比最小所需定时器间隔大的一个时间间隔。例如，使用 10 毫秒定时器时，为了保证时间间隔至少为 140 毫秒，则预设时间值应设为 15。

100 毫秒分辨率

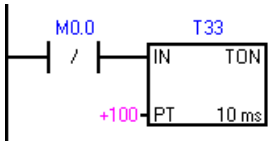
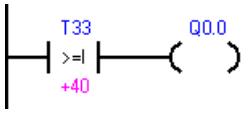
100 毫秒定时器记录自活动 100 毫秒定时器上次更新以来经过的 100 毫秒定时器间隔的数目。通过以下方法更新这种定时器：执行定时器指令时，将累积的 100 毫秒间隔数（自前一次扫描周期起）加到定时器的当前值。

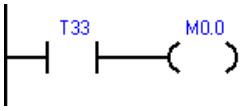
因为可在 100 毫秒内的任意时刻启动定时器，预设值必须设为比最小所需定时器间隔大的一个时间间隔。例如，使用 100 毫秒定时器时，为了保证时间间隔至少为 2100 毫秒，则预设时间值应设为 22。

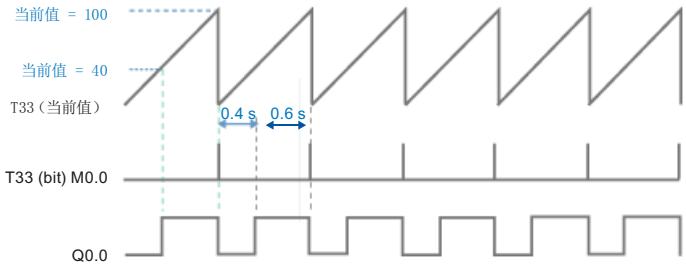
TON 接通延时定时器示例

LAD		STL
	100 ms 定时器 T37 在 1 s (10 x 100 ms) 后超时 • I0.0 ON = T37 使能， • I0.0 OFF = 禁用并复位 T37	**Network 1** **LD I0.0** **TON T37, +10**
	T37 位由定时器 T37 控制	**Network 2** **LD T37** **= Q0.0**
时序图 		

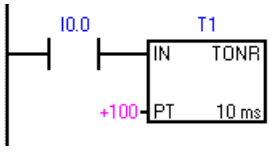
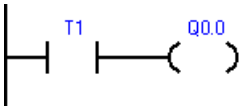
TON 自动复位接通延时定时器示例

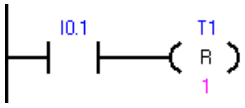
LAD		STL
	10 ms 定时器 T33 在 1 s (100 x 10 ms) 后超时 M0.0 脉冲速度过快，无法用状态视图监视。	**Network 1** **LDN M0.0** **TON T33, +100**
	以状态视图中可见的速率运行时，比较结果为真。 在 (40 x 10 ms) 之后，Q0.0 接通，信号波形 40% 为低电平，60% 为高电平	**Network 2** **LDW>= T33, +40** **= Q0.0**

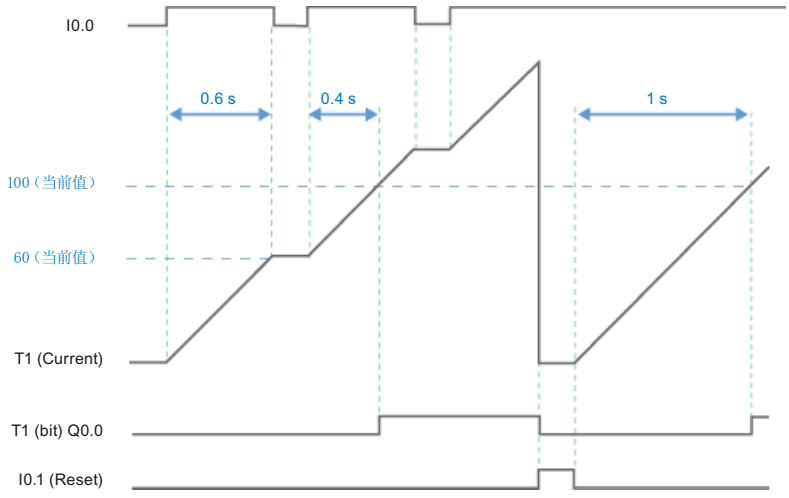
LAD		STL
	T33（位）脉冲速度过快，无法用状态视图监视。 在 (100 x 10 ms) 时间段之后，通过 M0.0 复位定时器。	**Network 3** **LD T33** **= M0.0**
时序图		



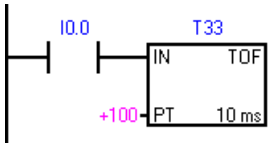
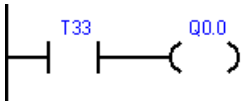
TONR 保持型接通延时定时器示例

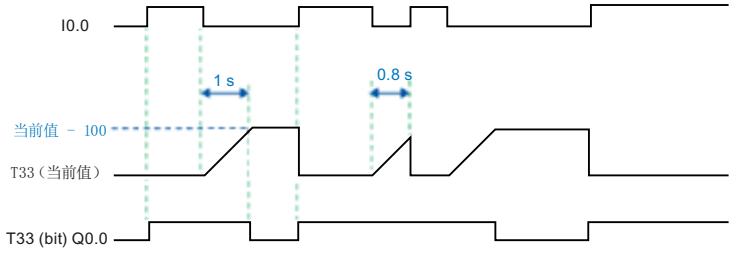
LAD		STL
	10 ms TONR 定时器 T1 在 PT = 1 s (100 x 10 ms) 时超时。	**Network 1** **LD I0.0** **TONR T1, +100**
	T1 位由定时器 T1 控制。 定时器总共累计 1 秒后，Q0.0 接通。	**Network 2** **LD T1** **= Q0.0**

LAD		STL
	TONR 定时器必须由带有 T 地址的复位指令复位。 I0.1 接通时，复位定时器 T1（当前值和定时器位）。	**Network 3** **LD I0.1** **R T1, 1**
时序图		



TOF 关断延时定时器示例

LAD		STL
	10 ms 定时器 T33 在 1 s (100 x 10 ms) 后超时。 • I0.0 上升沿 = T33 使能， • I0.0 下降沿 = 禁用并复位 T33	**Network 1** **LD I0.0** **TOF T33, +100**
	定时器 T33 通过定时器触点 T33 控制 Q0.0。	**Network 2** **LD T33** **= Q0.0**
时序图		



定时器分辨率对定时器位和当前时间值更新时间的影响

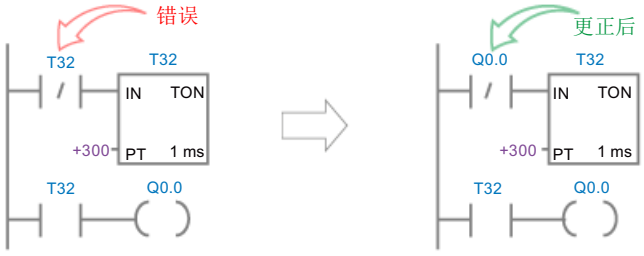
- **1 ms** 定时器：定时器位和当前值的更新与扫描周期不同步。扫描周期大于 1 ms 时，定时器位和当前值在该扫描周期内更新多次。
- **10 ms** 定时器：定时器位和当前值在每个扫描周期开始时更新。定时器位和当前值在整个扫描期间保持不变。扫描期间累积的时间间隔会在每次扫描开始时加到当前值上。
- **100 ms** 定时器：对于分辨率为 100 ms 的定时器，定时器位和当前值在指令执行时更新；因此，确保在每个扫描周期内程序仅执行 100 ms 定时器指令一次，这样才能保证定时器的定时正确。

示例：自动重新触发的单触发定时器

已更正的示例使用常闭触点 Q0.0 代替定时器位作为定时器使能输入。这样可确保输出 Q0.0 在每次定时器达到预设值时接通，并且在一个扫描周期内保持接通。

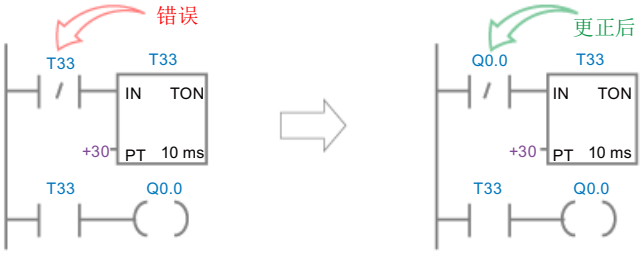
1 ms 定时器

在执行常闭触点 T32 之后以及执行常开触点 T32 之前，只要更新定时器的当前值，Q0.0 就会在一个扫描周期内保持接通。



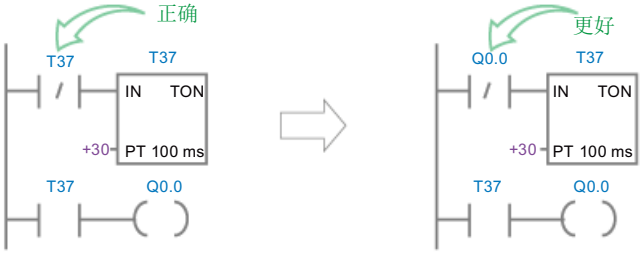
10 ms 定时器

Q0.0 从不接通，因为定时器位 T33 在从扫描开始到执行定时器功能框的时间段内接通。执行定时器功能框后，定时器的当前值及其 T 位均置零。执行常开触点 T33 时，T33 及 Q0.0 均断开。



100 ms 定时器

只要定时器的当前值达到预设值，Q0.0 就会在一个扫描周期内始终接通。



9.16.3 定时器指令(IEC)

LAD/FBD	STL	说明
IN IEC_TON Q PT ET	IECTON xxx, PT, ET	使用“接通延时”指令，可将输出 Q 的置位延迟预设的时间 PT。当输入 IN 的逻辑运算结果 从“0”变为“1”（上升沿）时，启动该指令。指令启动时，ET 即开始计时, 超出时间 PT 之后，输出 Q 的信号状态将变为“1”。只要启动输入仍为“1”，输出 Q 就保持置位。启动输入的信号状态从“1”变为“0”时，将复位输出 Q。在启动输入检测到新的信号上升沿时，该定时器功能将再次启动。 可以在 ET 输出查询当前的时间值。该定时器值从 0 开始，最大能累计到 2147483647。只要输入 IN 的信号状态变为“0”， ET 就复位清零。 IEC_TON 分辨率为 1 ms。
IN IEC_TOF Q PT ET	IECTOF xxx, PT, ET	使用“关断延时”指令，可将输出 Q 的复位延迟预设的时间 PT。当输入 IN 的逻辑运算结果 从“0”变为“1”（上升沿）时，将置位 Q 输出。当输入 IN 的信号状态重新变为“0”（下降沿）时，ET 开始计时。只要 ET 仍在计时，输出 Q 就保持置位。ET 累计达到 PT 时计时结束，将复位输出 Q。如果输入 IN 的信号状态在 ET 计时结束之前变为“1”，则重置定时器, 输出 Q 的信号状态仍将为“1”。 可以在 ET 输出查询当前的时间值。该定时器值从 0 开始，在达到 PT 后结束。当 ET 计时结束后，在输入 IN 变回“1”之前，输出 ET 会保持计时结束时的状态。在 ET 计时结束之前，如果输入 IN 的信号状态切换为“1”，则 ET 将复位为 0 。 IEC_TOF 分辨率为 1 ms。
IN IEC_TONR Q RESET ET PT	IECTONR xxx, RESET, PT, ET	使用"时间累积"指令，可在参数 PT 设定的时间范围内进行时间值累加。当输入 IN 的信号状态从"0"变为"1"（上升沿）时，启动时间测量并且 ET 开始计时。在计时期间，系统将持续累加输入 IN 为"1"状态的时间值，累积时间通过输出 ET 显示并可查询。当 ET 达到设定时间 PT 时，输出 Q 将置位为"1"。即使输入 IN 信号状态由"1"变为"0"（下降沿），Q 仍保持置位状态。 输入 RESET 将强制复位输出 ET 和 Q，不受启动输入信号状态影响。 IEC_TONR 分辨率为 1 ms。

输入/输出	数据类型	操作数
IN	BOOL	能流
PT	DWORD	I、Q、V、M、SM、T、C、L、AC、*AC、常数、全局变量、FB 成员变量
RESET	BOOL	I、Q、V、M、SM、S、T、C、L、全局变量、FB 成员变量

输入/输出	数据类型	操作数
Q	BOOL	能流
ET	DWORD	I、Q、V、M、SM、T、C、L、AC、*AC、全局变量、FB 成员变量

9.16.4 定时器编程提示和示例(IEC)

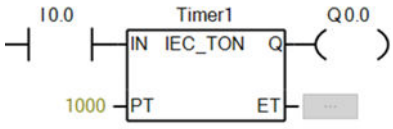
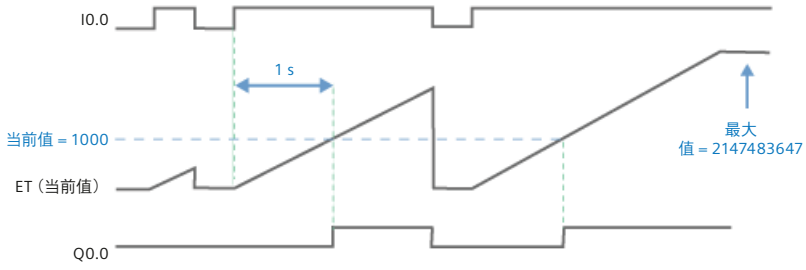
定时器类型

您可利用定时器实现时基计数功能。S7-200 SMART 指令集提供三种不同类型的 IEC 定时器。

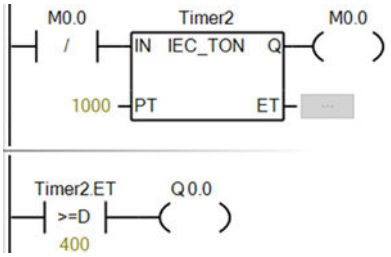
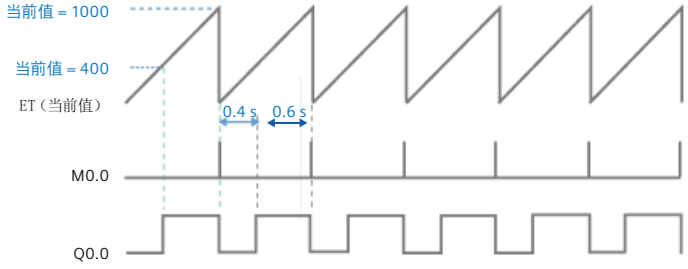
- 接通延时定时器 (IEC_TON)，用于单间隔定时
- 保持型接通延时定时器 (IEC_TONR)，用于累积一定数量的定时间隔
- 断开延时定时器 (IEC_TOF)，用于在断开（或 FALSE）条件之后延长一定时间，例如电机关闭后使电机冷却

示例

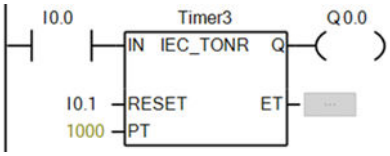
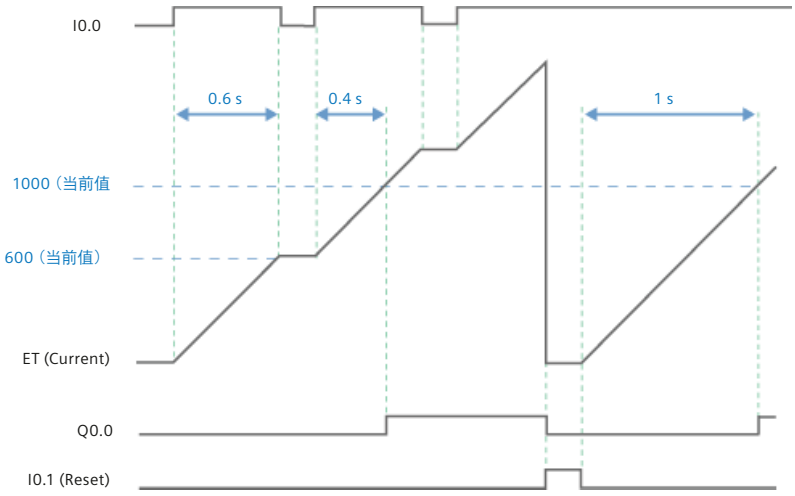
TON 接通延时定时器示例

<table>
<tr><th colspan="2">LAD</th><th>STL</th></tr>
<tr><td> 
</td><td>定时器在 1000ms(1s) 后超时
• I0.0 ON = 定时器使能，
• I0.0 OFF = 禁用并复位定时器</td><td><pre>Network 1
LD IECTON Timer1, Timer1.PT, Timer1.ET
= Q0.0</pre></td></tr>
<tr><td colspan="3">时序图
 
</td></tr>
</table>

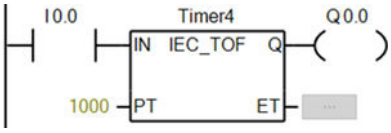
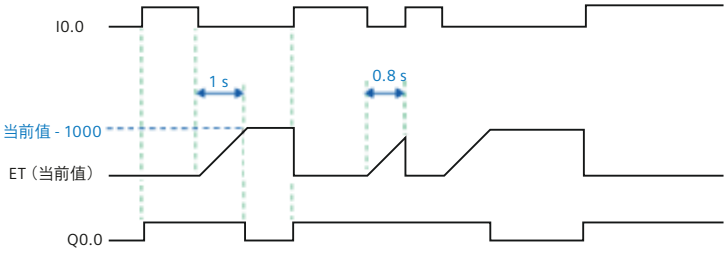
TON 自动复位接通延时定时器示例

LAD		STL
	1000 ms 后,定时器超时,超时后 M0.0 置位,又会复位定时器 M0.0 脉冲速度过快，无法用状态视图监视到。 在 400ms 之后，Q0.0 接通，信号波形 40% 为低电平，60% 为高电平。	**Network 1** **LDN M0.0** **IECTON Timer2, 1000, Timer2.ET** **= M0.0** **Network 2** **LDD>= Timer4.ET, 400** **= Q0.0**
时序图 		

TONR 保持型接通延时定时器示例

LAD		STL
	定时器总共累计 1 秒后，Q0.0 接通。 I0.1 接通时，复位定时器（ET 和 Q）。	**Network 1** **LD I0.0** **IECTONR Timer3, I0.1, 1000, Timer3.ET** **= Q0.0**
时序图 		

TOF 关断延时定时器示例

LAD		STL
	定时器在 1000ms 后超时。 • I0.0 上升沿 = 定时器复位, Q0.0 置位， • I0.0 下降沿 = 定时器开始计时, 超时后 Q0.0 复位。	**Network 1** **LD I0.0** **IECTOF Timer4, 1000, Timer4.ET** **= Q0.0**
时序图 		

9.16.5 时间间隔定时器

LAD/FBD	STL	说明
BGN_ITIME EN ENO OUT	**BITIM OUT**	开始间隔时间指令读取内置 1 毫秒计数器的当前值，并将该值存储在 OUT 中。 双字毫秒值的最大计时间隔为 2 的 32 次方或 49.7 天。
CAL_ITIME EN ENO IN OUT	**CITIM IN, OUT**	计算间隔时间指令计算当前时间与 IN 中提供的时间的时间差，然后将差值存储在 OUT 中。 双字毫秒值的最大计时间隔为 2 的 32 次方或 49.7 天。 根据 BITIM 指令的执行时间，CITIM 指令会自动处理在最大间隔内发生的一毫秒定时器翻转。

ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址	无

输入/输出	数据类型	操作数
IN	DWORD	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
OUT	DWORD	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

示例： 触发和计算时间间隔定时器

LAD		STL
Q0.0 P BGN_ITIME EN ENO OUT VD0 Ex1_Interval_time_net1	捕捉 Q0.0 接通的时刻。	**Network 1** **LD Q0.0** **EU** **BITIM VD0**
Q0.0 CAL_ITIME EN ENO VD0 IN OUT VD4	计算 Q0.0 接通的时长。	**Network 2** **LD Q0.0** **CITIM VD0, VD4**

9.17 子例程

9.17.1 CALL（子例程）和 RET（有条件返回）

要添加新子例程，请选择 **“编辑”(Edit)** 功能区，然后选择 **“插入对象”(Insert Object)** 和 **“子例程”(Subroutine)** 命令。STEP 7-Micro/WIN SMART 自动在每个子例程中添加一个无条件返回。还可以在子例程中添加有条件返回 CRET 指令。

在主程序中，可以嵌套子例程和 FB（在子例程/FB 中调用子例程/FB），最大嵌套深度为八。在中断例程中，子例程/FB 的最大嵌套深度为四。

说明

允许递归调用（子例程调用自己），但在子程序中进行递归调用时应慎重。

LAD/FBD	STL	说明
SBR_n EN x1 x2 x3	**CALL SBR_n, x1, x2, x3**	子例程调用指令将程序控制权转交给子例程 SBR_n。可以使用带参数或不带参数的子例程调用指令。子例程执行完后，控制权返回给子例程调用指令后的下一条指令。 调用参数 x1 (IN)、x2 (IN_OUT) 和 x3 (OUT) 分别表示传入、传入和传出或传出子例程的三个调用参数。调用参数是可选的。可以使用 0 到 63 个调用参数。 调用子例程时，保存整个逻辑堆栈，栈顶值设置为一，堆栈其他位置的值设置为零，控制权交给被调用子例程。该子例程执行完后，堆栈恢复为调用时保存的数值，控制权返回给调用例程。 子例程和调用例程共用累加器。由于子例程使用累加器，所以不对累加器执行保存或恢复操作。 在同一周期内多次调用子例程时，不应使用上升沿、下降沿、定时器和计数器指令。
LAD: RET FBD: RET	**CRET**	从子例程有条件返回指令 (CRET) 根据前面的逻辑终止子例程。

ENO = 0 时的错误条件	受影响的 SM 位
• 0006H 间接地址 • 008H 超出子例程最大嵌套限制	无

输入/输出	数据类型	操作数
SBR_n	WORD	常数：0-127
IN	BOOL	V、I、Q、M、SM[2]、S、T、C、L、能流 (LAD)、逻辑流 (FBD)
	BYTE	VB、IB、QB、MB、SMB[2]、SB、LB、AC、*VD、*LD、*AC[1]、常数
	WORD, INT	VW、T、C、IW、QW、MW、SMW[2]、SW、LW、AC、AIW、*VD、*LD、*AC[1]、常数
	DWORD, DINT	VD、ID、QD、MD、SMD[2]、SD、LD、AC、HC、*VD、*LD、*AC[1]、&VB、&IB、&QB、&MB、&T、&C、&SB、&AI、&AQ、&SMB、常数
	REAL	VD、ID、QD、MD、SMD[2]、SD、LD、AC、*VD、*LD、*AC1、&VB、&IB、&QB、&MB、&T、&C、&SB、&AI、&AQ、&SMB、常数
	STRING	*VD、*LD、*AC[1]、常数
	ARRAY, UDT	
IN_OUT	BOOL	V、I、Q、M、SM[2]、S、T、C、L
	BYTE	VB、IB、QB、MB、SMB[2]、SB、LB、AC、*VD、*LD、*AC[1]
	WORD, INT	VW、T、C、IW、QW、MW、SMW[2]、SW、LW、AC、*VD、*LD、*AC[1]
	DWORD, DINT	VD、ID、QD、MD、SMD[2]、SD、LD、AC、*VD、*LD、*AC[1]
	REAL	VD、ID、QD、MD、SMD[2]、SD、LD、AC、*VD、*LD、*AC1
	ARRAY, UDT	
OUT	BOOL	V、I、Q、M、SM[2]、S、T、C、L
	BYTE	VB、IB、QB、MB、SMB[2]、SB、LB、AC、*VD、*LD、*AC[1]
	WORD, INT	VW、T、C、IW、QW、MW、SMW[2]、SW、LW、AC、AQW、*VD、*LD、*AC[1]
	DWORD, DINT	VD、ID、QD、MD、SMD[2]、SD、LD、AC、*VD、*LD、*AC[1]
	REAL	VD、ID、QD、MD、SMD[2]、SD、LD、AC、*VD、*LD、*AC[1]
	ARRAY, UDT	

[1] 只允许 AC1、AC2 或 AC3（不允许 AC0）

[2] 字节偏移必须在 30 到 999 之间才能进行读/写访问

9.17.2 子例程调用参数和返回示例

带调用参数调用子例程

子例程可选择使用传递参数。参数在子例程的局部变量表中定义。必须为每个参数分配局部变量名称（最多 23 个字符）、变量类型和数据类型。一个子例程最多可以传递十六个参数。变量表中的 VAR_Type 类型字段定义变量是传入子例程 (IN)、传入和传出子例程 (IN_OUT)，还是传出子例程 (OUT)。

要添加新参数行，请将光标置于要添加变量类型 IN、IN_OUT、OUT 或 TEMP 的 Var_Type 字段上。单击鼠标右键打开选择菜单。选择“插入”(Insert) 选项，然后选择“下一行”(Row Below) 选项。所选类型的另一个参数行将出现在当前条目下方。

可在变量表中分配临时 (TEMP) 参数来存储只在子例程执行过程中有效的数据。局部 TEMP 数据不会作为调用参数进行传递。也可在主例程和中断例程中分配 TEMP 参数，但只有子例程可以使用 IN、IN_OUT 和 OUT 调用参数。

子例程的变量表参数类型

参数	描述
IN	参数传入子例程。如果参数是直接地址（例如 VB10），则指定位置的值传入子例程。如果参数是间接地址（例如 *AC1），则指针指代位置的值传入子例程。如果参数是数据常数 (16#1234) 或地址 (&VB100)，则常数或地址值传入子例程。
IN_OUT	指定参数位置的值传入子例程，子例程的结果值返回至同一位置。常数（例如 16#1234）和地址（例如 &VB100）不允许用作输入/输出参数。
OUT	子例程的结果值返回至指定参数位置。常数（例如 16#1234）和地址（例如 &VB100）不允许用作输出参数。由于输出参数并不保留子例程最后一次执行时分配给它的值，所以每次调用子例程时必须给输出参数分配值。
TEMP	没有用于传递参数的任何局部存储器都可在子例程中作为临时存储单元使用。

调用参数允许的数据类型

- 能流：布尔能流仅允许用于位（布尔）输入。此声明将输入参数分配给基于位逻辑指令组合的能流结果。能流输入与 EN 输入相似，都与位逻辑（例如，LAD 触点）相连接，而不连接到直接/间接地址分配。必须在变量表的最上一行（或多行）指定布尔能流输入，然后再指定任何非布尔数据类型。只有输入参数可以这样使用。下例中的使能输入 (EN) 和 IN1 输入使用能流逻辑。
- BOOL：此数据类型用于单个位输入和输出。下例中的 IN3 是分配给直接地址的布尔输入。

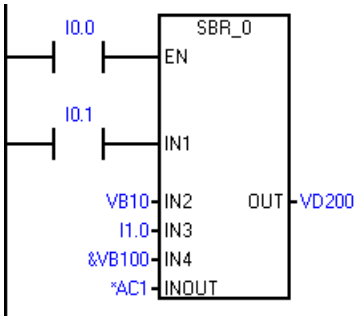
- BYTE、WORD、DWORD：这些数据类型分别标识1、2或4字节的无符号输入或输出参数。
- INT、DINT：这些数据类型分别标识 2 或 4 字节有符号输入或输出参数。
- REAL：此数据类型标识单精度（4 字节）IEEE 浮点值。
- STRING：此数据类型用作指向字符串的四字节指针。

变量表示例

变量表

	变量名	地址	变量类型	数据类型	注释
1	EN		IN	BOOL	
2	Lv1	L0.0	IN	BOOL	Tank trans...
3	HSet	L0.1	IN	BOOL	High setpoint
4	Offset	L0.2	IN	BOOL	Offset
5			IN		
6			IN_OUT		
7	Result	L0.3	OUT	BOOL	Result value
8	Valve	L0.4	OUT	BOOL	Control value
9			OUT		
10			TEMP		

示例：带调用参数的子例程调用

LAD	STL
	仅在 STL 中显示： **Network 1** **LD I0.0** **CALL SBR_0, I0.1, VB10, I1.0, &VB100, *AC1, VD200** 可在 LAD 和 FBD 中正确显示： **Network 1** **LD I0.0** **= L60.0** **LD I0.1** **= L63.7** **LD L60.0** **CALL SBR_0, L63.7, VB10, I1.0, &VB100, *AC1, VD200**

说明

上面提供了两个 STL 示例。STL 程序员可使用第一组只能显示在 STL 编辑器中的简化 STL 指令。这是因为用作 LAD/FBD 能流输入的 BOOL 参数不保存到 L 存储器。

第二组编译器生成的 STL 指令可显示在 LAD、FBD 和 STL 编辑器中，因为程序编译器使用 L 存储器来保存在 LAD/FBD 中指定为能流输入的 BOOL 输入参数的状态。

地址参数（例如，IN4 (&VB100)）传入子例程作为 DWORD（无符号双字）值。对于调用例程中常数值前面有常数描述符的参数，必须为其指定常数参数类型。例如：要传送值为 12,345 的无符号双字常数作为参数，必须将常数参数指定为 DW#12345。如果参数中遗漏了常数描述符，则可将该常数认定为不同类型。

系统不对输入或输出参数自动执行数据类型转换。例如，如果变量表指定参数的数据类型为 REAL，但在调用例程中，为该参数指定双字 (DWORD) 数据类型，则子例程中的参数值将是双字数据类型。

值传递到子例程后，存储在子例程的局部存储器中。变量表的最左列显示各传递参数的局部存储器地址。调用子例程时，输入参数值将复制到子例程的局部存储器中。子例程执行完成时，从子例程的局部存储器将输出参数值复制到指定输出参数地址。

数据元素大小和类型用参数的编码表示。参数值到子例程中的局部存储器的分配如下所述：

- 参数值按照带参数的调用子例程指令指定的顺序分配给局部存储器，起始地址是 L 0.0。
- 一至八个连续位参数值分配给从 Lx.0 到 Lx.7 的单个字节。
- 字节、字和双字值分配给以字节为边界的局部存储器（LBx、LWx 或 LDx）。

在带参数的子例程调用指令中，必须按照一定的顺序排列参数，输入参数在最前面，其次是输入/输出参数，然后是输出参数。

如果使用 STL 编程，则 CALL 指令的格式是：
CALL 子例程编号, 参数 1, 参数 2, ... , 参数 16

示例：子例程和子例程返回指令

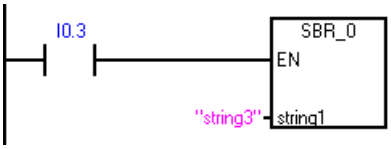
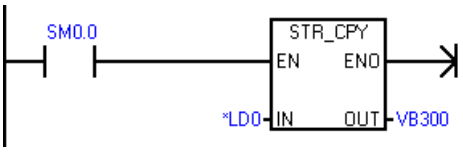
LAD			STL
MAIN	SM0.1 — SBR_0 EN	首次扫描时，调用子例程 0 进行初始化。	Network 1 LD SM0.1 CALL SBR_0
SBR0	M14.3 — (RET)	可以在最后一个程序段前使用有条件返回指令来退出子例程。	Network 1 LD M14.3 CRET
SBR0	SM0.0 — MOV_B EN ENO; 10 IN OUT VB0	如果 M14.3 接通，将跳过此网络。	Network 2 LD SM0.0 MOVB 10, VB0

示例：使用字符串参数的子例程调用

此示例根据给定输入的状态将不同字符串文字复制到唯一地址。此字符串的唯一地址被保存。然后，通过间接地址将字符串地址传入子例程。子例程输入参数的数据类型是字符串。然后，子例程将字符串移到其他位置。

字符串文字也可传入子例程。子例程内的字符串引用始终相同。

LAD		STL
MAIN	I0.0 — STR_CPY EN ENO; "string1" IN OUT VB100 — MOV_DW EN ENO; &VB100 IN OUT VD0	Network 1 LD I0.0 SCPY "string1", VB100 AENO MOVD &VB100, VD0
MAIN	I0.1 — STR_CPY EN ENO; "string2" IN OUT VB200 — MOV_DW EN ENO; &VB200 IN OUT VD0	Network2 LD I0.1 SCPY "string2", VB200 AENO MOVD &VB200, VD0
MAIN	I0.2 — SBR_0 EN; *VD0 string1	Network3 LD I0.2 CALL SBR_0, *VD0

LAD		STL
MAIN		**Network4** **LD I0.3** **CALL SBR_0, "string3"**
SBR0		**Network 1** **LD SM0.0** **SSCPY *LD0, VB300**

9.18 功能块

9.18.1 CALL（功能块）和 RET（有条件返回）

要添加新功能块，请选择 **“编辑”(Edit)** 功能区，然后选择 **“插入对象”(Insert Object)** 和 **“功能块”(function block)** 命令。STEP 7-Micro/WIN SMART 自动在每个功能块中添加一个无条件返回。还可以在功能块中添加有条件返回 CRET 指令。

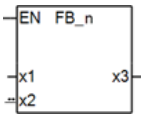
在主程序中，可以嵌套功能块（在功能块中调用功能块），最大嵌套深度为八。

在中断例程中，可嵌套的功能块深度为四。

在主程序中，可以嵌套子例程和 FB（在子例程/FB 中调用子例程/FB），最大嵌套深度为八。在中断例程中，子例程/FB 的最大嵌套深度为四。

说明

不允许递归（功能块调用自身）。

LAD/FBD	STL	说明
	**FBCALL FB_n, FB_n_instance, FB_n_instance.x1, FB_n_instance.x2, FB_n_instance.x3**	功能块调用指令将程序控制权交给功能块 SBR_n。可以使用带参数或不带参数的功能块调用指令。功能块执行完后，控制权返回给功能块调用指令后的下一条指令。 调用参数 x1 (IN)、x2 (IN_OUT) 和 x3 (OUT) 分别表示传入、传入和传出或传出功能块的三个调用参数。调用参数是可选的。可以使用 0 到 62 个调用参数。 调用功能块时，保存整个逻辑堆栈，栈顶值设置为一，堆栈其他位置的值设置为零，控制权交给被调用功能块。该功能块执行完后，堆栈恢复为调用时保存的数值，控制权返回给调用例程。 功能块和调用例程共用累加器。由于功能块使用累加器，所以不对累加器执行保存或恢复操作。 在同一周期内多次调用功能块时，不应使用上升沿、下降沿、定时器、ALT、ALTP 和计数器指令。

ENO = 0 时的错误条件	受影响的 SM 位
• 0006H 间接地址 • 008H 超出功能块最大嵌套限制	无

输入/输出	数据类型	操作数
FB_n	WORD	常数：0-127
IN	BOOL	V、I、Q、M、SM[2]、S、T、C、L、能流 (LAD)、逻辑流 (FBD)
	BYTE	VB、IB、QB、MB、SMB[2]、SB、LB、AC、*VD、*LD、*AC[1]、常数
	WORD, INT	VW、T、C、IW、QW、MW、SMW[2]、SW、LW、AC、AIW、*VD、*LD、*AC[1]、常数
	DWORD, DINT	VD、ID、QD、MD、SMD[2]、SD、LD、AC、HC、*VD、*LD、*AC[1]、&VB、&IB、&QB、&MB、&T、&C、&SB、&AI、&AQ、&SMB、常数
	REAL	VD、ID、QD、MD、SMD[2]、SD、LD、AC、*VD、*LD、*AC[1]、&VB、&IB、&QB、&MB、&T、&C、&SB、&AI、&AQ、&SMB、常数
	STRING	*VD、*LD、*AC[1]、常数
	ARRAY, UDT	

输入/输出	数据类型	操作数
IN_OUT	BOOL	V、I、Q、M、SM[2]、S、T、C、L
	BYTE	VB、IB、QB、MB、SMB[2]、SB、LB、AC、*VD、*LD、*AC[1]
	WORD, INT	VW、T、C、IW、QW、MW、SMW[2]、SW、LW、AC、*VD、*LD、*AC[1]
	DWORD, DINT	VD、ID、QD、MD、SMD[2]、SD、LD、AC、*VD、*LD、*AC[1]
	REAL	VD、ID、QD、MD、SMD[2]、SD、LD、AC、*VD、*LD、*AC[1]
	STRING	*VD、*LD、*AC[1]
	ARRAY, UDT	
OUT	BOOL	V、I、Q、M、SM[2]、S、T、C、L
	BYTE	VB、IB、QB、MB、SMB[2]、SB、LB、AC、*VD、*LD、*AC[1]
	WORD, INT	VW、T、C、IW、QW、MW、SMW[2]、SW、LW、AC、AQW、*VD、*LD、*AC[1]
	DWORD, DINT	VD、ID、QD、MD、SMD[2]、SD、LD、AC、*VD、*LD、*AC[1]
	REAL	VD、ID、QD、MD、SMD[2]、SD、LD、AC、*VD、*LD、*AC[1]
	STRING	*VD、*LD、*AC[1]
	ARRAY, UDT	
Static	BOOL	
	BYTE	
	WORD, INT	
	DWORD, DINT	
	REAL	
	STRING	
	ARRAY, UDT	

[1] 只允许 AC1、AC2 或 AC3（不允许 AC0）

[2] 字节偏移必须在 30 到 999 之间才能进行读/写访问

注：功能块的输入或输出可以是 UDT 或 ARRAY 类型元素。

9.18.2 功能块调用参数

调用带调用参数的功能块

功能块可选择使用传递参数。参数在功能块的局部变量表中定义。必须为每个参数分配局部变量名称（最多 23 个字符）、变量类型和数据类型。一个功能块最多可以传递十六个参数。变量表中的 VAR_Type 类型字段定义变量是传入功能块 (IN)、传入和传出功能块 (IN_OUT)，还是传出功能块 (OUT)。

要添加新参数行，请将光标置于要添加变量类型 IN、IN_OUT、OUT、STATIC 或 TEMP 的 Var_Type 字段上。单击鼠标右键打开选择菜单。选择“插入”(Insert) 选项，然后选择“下一行”(Row Below) 选项。所选类型的另一个参数行将出现在当前条目下方。

可在变量表中分配临时 (TEMP) 参数来存储只在功能块执行过程中有效的数据。局部 TEMP 数据不会作为调用参数进行传递。也可在主例程和中断例程中分配 TEMP 参数，但只有功能块可以使用 IN、IN_OUT 和 OUT 调用参数。

STATIC

功能块的变量表参数类型

参数	说明
IN	参数传入功能块。如果参数是直接地址（例如 VB10），则指定位置的值传入功能块。如果参数是间接地址（例如 *AC1），则指针指代位置的值传入功能块。如果参数是数据常数 (16#1234) 或地址 (&VB100)，则常数或地址值传入功能块。
IN_OUT	指定参数位置的值传入功能块，功能块的结果值返回至同一位置。常数（例如 16#1234）和地址（例如 &VB100）不允许用作输入/输出参数。
OUT	功能块的结果值返回至指定参数位置。常数（例如 16#1234）和地址（例如 &VB100）不允许用作输出参数。由于输出参数并不保留功能块最后一次执行时分配给它的值，所以每次调用功能块时必须给输出参数分配值。
TEMP	没有用于传递参数的任何局部存储器都可在功能块中作为临时存储单元使用。
STATIC	静态变量保存在 FB 实例中，不参与外部参数传递。

调用参数允许的数据类型

- BOOL：此数据类型用于单个位输入和输出。下例中的 IN3 是分配给直接地址的布尔输入。
- BYTE、WORD、DWORD：这些数据类型分别标识 1、2 或 4 字节的无符号输入或输出参数。
- INT、DINT：这些数据类型分别标识 2 或 4 字节有符号输入或输出参数。

- REAL：此数据类型标识单精度（4 字节）IEEE 浮点值。
- STRING：此数据类型用作指向字符串的四字节指针。
- ARRAY 或 UDT：功能块的输入或输出可以是 UDT 或 ARRAY 类型元素。

变量表示例

本地变量表

	变量名	地址	变量类型	数据类型	默认值	掉电保持	注释
1	SetCom		IN	BOOL	OFF	不保持	
2	ResetCom		IN	BOOL	OFF	不保持	
3	Input		IN	INT	+0	不保持	
4			IN_OUT	BOOL	OFF	不保持	
5	Output		OUT	INT	+0	不保持	
6	Out		OUT	BOOL	OFF	不保持	
7	Diff		OUT	INT	+0	不保持	
8	SavedInput		STATIC	INT	+0	不保持	
9	Resetstate		STATIC	BOOL	OFF	不保持	
10							
11			TEMP				

示例：带调用参数的功能块调用

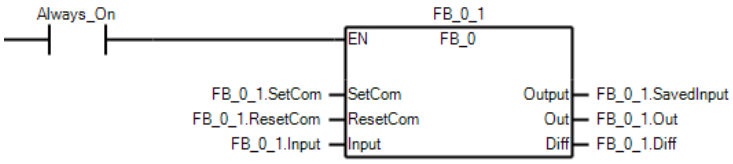
要调用功能块，必须先创建 FB 实例表中的实例。

全局变量表

	变量名	数据类型	初始值	保持	地址	长度	注释
2	SetCom	BOOL	OFF		DB2.DBX8.0		
3	Reset...	BOOL	OFF		DB2.DBX8.1		
4	Input	INT	+0		DB2.DBW10		
5	Output	INT	+0		DB2.DBW12		
6	Out	BOOL	OFF		DB2.DBX1...		
7	Diff	INT	+0		DB2.DBW16		
8	Saved...	INT	+0		DB2.DBW18		
9	Resets...	BOOL	OFF		DB2.DBX2...		
10	FB_0_2	FB_0			DB2.DBB22	14 B	
11	FB_0_3	FB_0			DB2.DBB36	14 B	

Variable Table 1 | Constant Table 1 | I/O Variable | POU Variables | System Defined Variable

全局变量表 | 状态图表 | 数据块 | UDT

LAD	
STL	`LD Always_On` `FBCALL FB_0, FB_0_1, FB_0_1.SetCom, FB_0_1.ResetCom, FB_0_1.Input, FB_0_1.SavedInput, FB_0_1.Out, FB_0_1.Diff`

地址参数（例如，IN4 (&VB100)）传入功能块作为 DWORD（无符号双字）值。对于调用例程中常数值前面有常数描述符的参数，必须为其指定常数参数类型。例如：要传送值为 12,345 的无符号双字常数作为参数，必须将常数参数指定为 DW#12345。如果参数中遗漏了常数描述符，则可将该常数认定为不同类型。

系统不对输入或输出参数自动执行数据类型转换。例如，如果变量表指定参数的数据类型为 REAL，但在调用例程中，为该参数指定双字 (DWORD) 数据类型，则功能块中的参数值将是双字数据类型。

值传递到功能块后，存储在功能块的局部存储器中。变量表的最左列显示各传递参数的局部存储器地址。调用功能块时，输入参数值将复制到功能块的局部存储器中。功能块执行完成时，从功能块的局部存储器将输出参数值复制到指定输出参数地址。

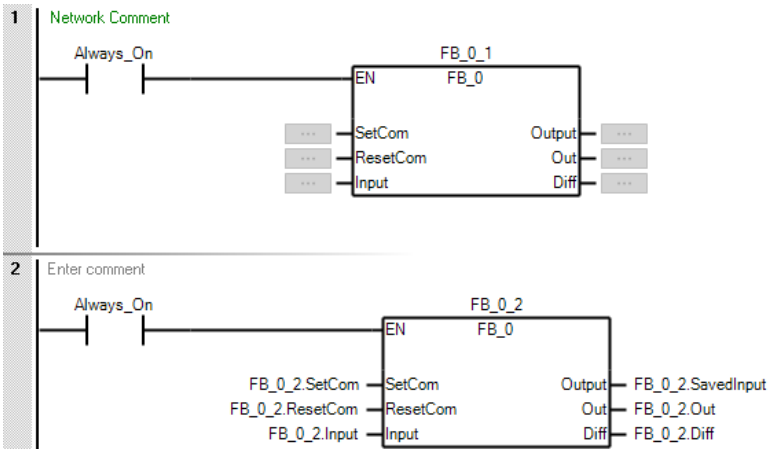
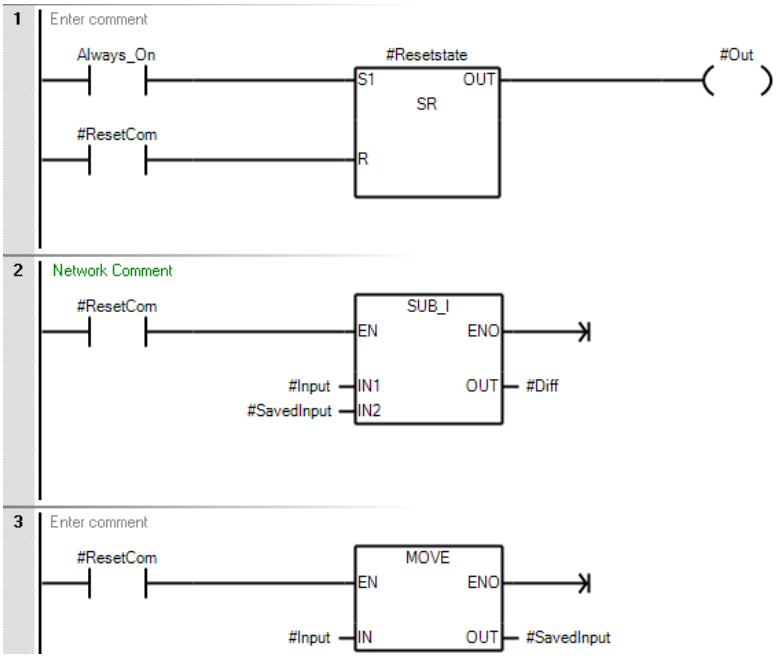
数据元素大小和类型用参数的编码表示。参数值到功能块中的局部存储器的分配如下所述：

- 参数值按照带参数的调用功能块指令指定的顺序分配给局部存储器，起始地址是 L 0.0。
- 一至八个连续位参数值分配给从 Lx.0 到 Lx.7 的单个字节。
- 字节、字和双字值分配给以字节为边界的局部存储器（LBx、LWx 或 LDx）。

在带参数的功能块调用指令中，必须按照一定的顺序排列参数，输入参数在最前面，其次是输入/输出参数，然后是输出参数。

如果使用 STL 编程，则 CALL 指令的格式是：
`CALL 功能块编号, 参数 1, 参数 2, ... , 参数 16`

示例：使用字符串参数的功能块调用

LAD		STL
MAIN[1]		Network 1 LD Always_On FBCALL FB_0, FB_0_1, FB_0_1.SetCom, FB_0_1.ResetCom, FB_0_1.Input, FB_0_1.Output, FB_0_1.Out, FB_0_1.Diff Network 2 LD Always_On FBCALL FB_0, FB_0_2, FB_0_2.SetCom, FB_0_2.ResetCom, FB_0_2.Input, FB_0_2.SavedInput, FB_0_2.Out, FB_0_2.Diff
FB0		Network 1 LD Always_On LD #ResetCom NOT A #Resetstate OLD = #Resetstate = #Out Network 2 LD #ResetCom -I #Input, #SavedInput, #Diff Network 3 LD #ResetCom MOVE #Input, #SavedInput

[1] 调用功能块 (FB) 实例时，如果不为引脚分配变量，系统会自动为引脚分配变量。

9.19 PROFINET

9.19.1 编程指令 "PROFINET" 的特性

“PROFINET”文件夹下有两组编程指令：

- RDREC 和 WRREC (页 504)：从任何连接的 PROFINET 设备读取数据记录或将数据记录写入任何连接的 PROFINET 设备。
- BLKMOV_BIR 和 BLKMOV_BIW (页 508)：立即从 PROFINET 设备读取多个字节或立即将多个字节写入 PROFINET 设备。

说明

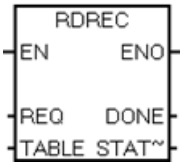
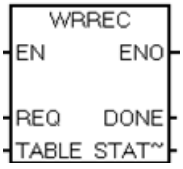
对于任何可访问 I 或 Q 存储区的传统指令，均可访问 PROFINET 网络的 I 或 Q 存储区。

9.19.2 读取和写入数据记录

9.19.2.1 RDREC 和 WRREC 指令的输入和输出接口

RDREC 和 WRREC 指令如下：

表格 9-22 RDREC 和 WRREC

LAD/FBD	STL	描述
	RDREC Req, Table, Done, Status	使用 RDREC 指令从 PROFINET 设备读取数据记录。
	WRREC Req, Table, Done, Status	使用 WRREC 指令将数据记录写入 PROFINET 设备。

RDREC 和 WRREC 指令的参数如下：

表格 9-23　RDREC 和 WRREC 指令的参数

参数和类型		数据类型	操作数	描述
REQ	IN	BOOL	I, Q, V, M, T, C, SM, S, L	REQ=1：传送数据记录
TABLE	IN/OUT	BYTE	Q, V, M, SM, S, L, *AC,*VD,*LD	为数据读/写记录设置的参数表。有关详细信息，请参见 “TABLE”参数的定义 (页 505)。
DONE	OUT	BOOL	I, Q, V, M, SM, S, L	完成指令。
STATUS	OUT	BYTE	I, Q, V, M, SM, S, L	当前操作的状态。有关详细信息，请参见 “STATUS”参数的定义 (页 506)。

说明

支持的最大数据记录长度为 1024 字节。

“TABLE”参数的定义

下表列出了“TABLE”的参数信息：

表格 9-24　TABLE

字节偏移	参数和类型		注释
0	Device Number	Input **注：** 值的范围为 1 至 8。	设备编号、API 编号、插槽编号和子插槽编号用于寻址子模块。 可在 PROFINET 向导中找到设备编号、API 编号、插槽编号和子插槽编号。
1			
2	API Number	Input	
3			
4			
5			
6	Slot Number	Input	
7			
8	SubSlot Number	Input	
9			

字节偏移	参数和类型		注释
10 11	Record Index	Input	记录索引包括协议中或用户定义的记录索引。 有关协议中索引的详细信息，请参见 *PROFINET IO（版本 2.3）的技术规范*。
12 13	Buffer Length	Input	此参数是指缓冲区的字节数。缓冲区存储从设备读取或写入设备的数据记录。 取值范围：1 至 1024。
14 15 16 17	Data Address	Input	缓冲区从设备读取或写入设备的地址。 **注：** 如果缓冲区长度大于实际记录数据长度，则缓冲区包含所有记录数据。 如果缓冲区长度小于实际记录数据长度，则缓冲区包含部分记录数据，并会发生错误。
18 19	Actual record data length	Output	此参数适用于指令 RDREC，并返回设备指定的实际数据长度。
20 21 22 23	PROFINET Error Code	Output	PROFINET 协议定义的错误代码。 0 = 无错误 如果该值不为 0，请检查 *PROFINET IO（版本 2.3）的技术规范*中的特定错误代码。

“STATUS”参数的定义

下表列出了“STATUS”的参数信息：

表格 9-25 STATUS

字节偏移	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
0	A [1]	E [2]	错误代码 [3]					

[1] A：1 = 正在进行请求

[2] E：1 = 出错

[3]错误代码：系统错误代码。有关详细信息，请参见 指令 RDREC 和 WRREC 的系统定义错误代码 (页 507)。

9.19.2.2 指令 RDREC 和 WRREC 的系统定义错误代码

错误代码如下：

错误代码	说明
0	无错误。
1	数据长度参数为 0 或大于支持的最大长度（1024 字节）。
2	数据缓冲区不在 I、Q、M 或 V 存储区。
3	数据缓冲区不适合存储区。
4	表格与存储器不匹配。
5	设备编号无效，且不在范围内，范围 1 至 8。
6	实例不匹配：连接正忙于另一个实例，其设备编号、API 编号、插槽编号和子插槽编号与请求的实例相同，但缓冲区大小和数据地址不同。
7	PROFINET 设备未连接。
8	接收缓冲区的大小超过 1024 个字节。
9	调用序列无效。
10	参数无效（例如，超出范围）。
11	同时重新创建 AR。
12	RPC 报告超时错误。
13	RPC 报告通信错误。
14	IOD 的 RPC 服务器显示“正忙”（例如，稍后可重新调用）。
15	CLRPC 报告错误或 PDU 无法解析。
16	CM 响应正常，但出现 PROFINET 协议定义的 错误。
17	指令参数无效。
24	未启用 REQ。
25	缓冲区长度小于实际数据记录长度。
63	未知错误。

9.19.3 在物理 PROFINET 和存储器地址之间读取和写入多个字节

9.19.3.1 BLKMOV_BIR 和 BLKMOV_BIW 的输入和输出接口

BLKMOV_BIR 和 BLKMOV_BIW 指令如下：

表格 9-26 BLKMOV_BIR 和 BLKMOV_BIW

LAD/FBD	STL	描述
BLKMOV_BIR EN ENO IN OUT N	BMIR In、Out、N	“BLKMOV_BIR 立即读取”(BLKMOV_BIR Immediate Read) 指令读取物理 PROFINET 输入 IN 的多个字节，并将结果写入存储器地址 OUT，但不更新过程映像寄存器。 输入信号 N 定义字节数。 注： • N ≤ 128 • N 字节数不能超过子模块边界。
BLKMOV_BIW EN ENO IN OUT N	BMIW In、Out、N	“BLKMOV_BIW 立即写入”(BLKMOV_BIM Immediate Write) 指令从存储器地址 IN 读取多个字节，并写入物理 PROFINET 输出 OUT，且相应过程映像位置更新。 输入信号 N 定义字节数。 注： • N ≤ 128 • N 字节数不能超过子模块边界。

BLKMOV_BIR 和 BLKMOV_BIW 参数如下：

参数和类型		数据类型	操作数
IN (BLKMOV_BIR)	IN	BYTE	IB, *VD, *LD, *AC
IN (BLKMOV_BIW)	IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
OUT (BLKMOV_BIR)	OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC

参数和类型		数据类型	操作数
OUT (BLKMOV_BIW)	OUT	BYTE	QB, *VD, *LD, *AC
N	IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, Constant, *VD, *LD, *AC

9.19.3.2 指令 BLKMOV_BIR 和 BLKMOV_BIW 的错误代码

ENO = 0 非致命错误。	受影响的 SM 位
● 0006H 间接寻址 ● 无法访问扩展模块	无

通信 10

10.1 概述

S7-200 SMART 可实现 CPU、编程设备和 HMI 之间的多种通信：

- 以太网：
  - 编程设备到 CPU 的数据交换
  - HMI 与 CPU 间的数据交换
  - S7 与其他 S7-200 SMART CPU 的对等通信
  - 与其他具有以太网功能的设备间的开放式用户通信 (OUC)
  - 使用 PROFINET 设备的 PROFINET 通信
- RS485：
  - 使用 USB-PPI 电缆时，提供一个适用于编程的 STEP 7-Micro/WIN SMART V3 连接

    说明

    S7-200 SMART G2 CPU 不支持直接连接 USB-PPI 线缆。若需使用此线缆连接 S7-200 SMART G2 CPU，则需要自行制作 RS485 连接线。
  - 总共支持 126 个可寻址设备（每个程序段 32 个设备）
  - 支持 PPI（点对点接口）协议
  - HMI 与 CPU 间的数据交换
  - 使用自由端口在设备与 CPU 之间交换数据（XMT/RCV 指令）
- RS232：
  - 支持与一台设备的点对点连接
  - 支持 PPI 协议
  - HMI 与 CPU 间的数据交换
  - 使用自由端口在设备与 CPU 之间交换数据（XMT/RCV 指令）

10.2 CPU 通信连接

CPU 最多可支持下列数量的并发异步通信连接：

- 以太网编程端口：
  - 开放式用户通信 (OUC) 连接：八个主动（客户端）连接和八个被动（服务器）连接，支持 S7-200 SMART V3.0/S7-200 SMART G2 CPU 或其他以太网设备。
  - HMI/OPC 连接：八个专用 HMI/OPC 服务器连接。
  - PG 连接：一个编程设备 (PG) 连接。
  - 对等 (GET/PUT) 连接：八个主动（客户端）连接和八个被动（服务器）连接，支持 S7-200 SMART V3.0/S7-200 SMART G2 CPU 或网络设备。
  - PROFINET 连接：每个 PROFINET 控制器可支持十个连接（IO 设备或驱动器）。

  说明

  可通过以太网端口对 S7-200 SMART V3.0/S7-200 SMART G2 CPU 进行编程。配备 SETP-7 Micro/WIN SMART V3 的单个设备一次只能编程或监视一个 CPU。

  说明

  S7-200 SMART CPU 使用 GET 和 PUT 指令进行 CPU 到 CPU 的通信。

- 集成的 RS485 端口（端口 0）：四个支持 HMI 设备的连接和一个预留用于通过 STEP 7-Micro/WIN SMART V3 进行编程的连接。

  说明

  可执行以下 RS485 通信连接：
  - 使用 USB-PPI 电缆通过任意串行端口（包括 RS485 端口和信号板端口）对所有型号的 CPU 进行编程。
    Note: S7-200 SMART G2 CPU 不支持直接连接 USB-PPI 线缆。若需使用此线缆连接 S7-200 SMART G2 CPU，则需要自行制作 RS485 连接线。
  - 使用 RS485 和 RS232 端口进行 HMI 访问（数据读/写）和自由端口通信。

- CM01 信号板 (SB) RS232/RS485 端口（端口 1、端口 2）：四个支持 HMI 设备的连接。

10.3 CPU 通信端口

S7-200 SMART V3 CPU 提供具有以下类型通信的接口：

- 以太网端口：
  - STEP 7-Micro/WIN SMART 编程
  - GET/PUT 通信
  - HMI：以太网类型
  - 基于 UDP、TCP 或 ISO-on-TCP 的开放式用户通信 (OUC)
  - PROFINET 通信
- RS485 端口（端口 0）：
  - 使用 USB-PPI 电缆进行 STEP 7-Micro/WIN SMART V3 编程

    说明

    S7-200 SMART G2 CPU 不支持直接连接 USB-PPI 线缆。若需使用此线缆连接 S7-200 SMART G2 CPU，则需要自行制作 RS485 连接线。

  - TD/HMI：RS485 类型
  - 自由端口 (XMT/RCV) 包括 Siemens 提供的 USS 和 Modbus RTU 库
- RS485/RS232 信号板 (SB)：
  - TD/HMI：RS485 或 RS232 类型
  - 自由端口 (XMT/RCV)，包括 Siemens 提供的 USS（仅 RS485）和 Modbus RTU（RS485 或 RS232）库

10.4 以太网

10.4.1 本地/伙伴连接

本地/伙伴（远程）连接定义两个通信伙伴的逻辑分配以建立通信连接。通过以下内容定义连接：

- 涉及的通信伙伴（一个主动，一个被动）
- 连接类型（编程设备、HMI、CPU 或其他设备）。可在“组态 PLC 安全 (页 57)”中启用/禁用与 STEP 7-Micro/WIN SMART V3 或 HMI 的安全通信。
- 连接路径（网络、IP 地址、子网掩码、网关）

通信伙伴设置和建立通信连接。主动设备建立连接，被动设备则接受或拒绝来自主动设备的连接请求。建立连接后，可通过主动设备对该连接进行自动维护，并通过主动和被动设备对其进行监视。

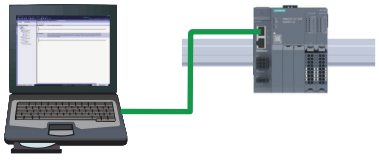
如果连接终止（例如，因断线或其中一个伙伴断开连接），主动伙伴将尝试重新建立连接。被动设备也将注意到连接出现终止并采取行动（例如，撤销新断开连接的主动伙伴的密码权限）。

S7-200 SMART CPU 既是主动设备，又是被动设备。主动设备（例如，运行 STEP 7-MicroWIN SMART 的计算机或 HMI）建立连接时，CPU 将根据连接类型以及给定连接类型所允许的连接数量来决定是接受还是拒绝连接请求。

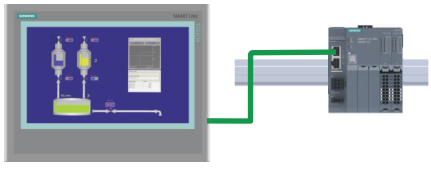
10.4.2 以太网网络组态示例

使用 S7-200 SMART CPU 以太网网络时，有三种不同类型的通信选项：

- 将 CPU 连接到编程设备
- 将 CPU 连接到 HMI
- 将 CPU 连接到另一个 S7-200 SMART CPU



CPU 连接到编程设备

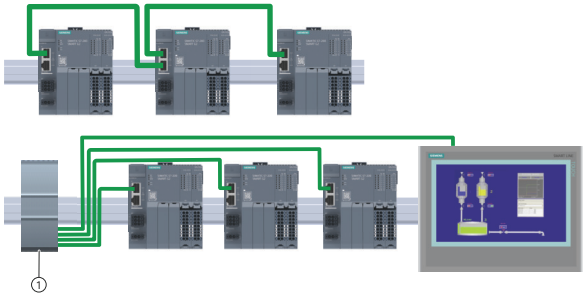


CPU 连接到 HMI

CPU 连接到另一个 CPU

以太网交换

S7-200 SMART CPU 有两个网络端口和一个内置以太网交换机。建立网络时，可选择以线性拓扑的方式连接设备，也可使用额外的交换机来实现星形拓扑。



① CSM1277 以太网交换机

可以使用安装在机架上的 CSM1277 4 端口以太网交换机来连接多个 CPU 和 HMI 设备。

10.4.3 分配 Internet 协议 (IP) 地址

10.4.3.1 为编程设备和网络设备分配 IP 地址

如果编程设备正在使用连接到工厂 LAN 的网络适配器卡（可能是万维网），则编程设备和 CPU 必须处于同一子网中。IP 地址与子网掩码相结合即可指定设备的子网。请联系本地网络管理员获取相关帮助。

网络 ID 是 IP 地址的第一部分（前三个八位位组）（例如，**211.154.184.16**），它决定用户所在的 IP 网络。子网掩码的值通常为 **255.255.255.0**；然而由于您的计算机处于工厂 LAN 中，子网掩码可能有不同的值（例如，**255.255.254.0**）以设置唯一的子网。子网掩码通过与设备 IP 地址进行逻辑 AND 运算来定义 IP 子网的边界。

说明

在万维网环境下，编程设备、网络设备和 IP 路由器可与全世界通信，但必须分配唯一的 IP 地址以避免与其他网络用户冲突。请联系公司 IT 部门熟悉工厂网络的人员分配 IP 地址。

说明

当不想将 CPU 连入公司 LAN 时，非常适合使用次级网络适配器卡。在首次测试或调试测试期间，这种安排尤其实用。

10.4.3.2 为项目中的 CPU 或设备组态或更改 IP 地址

必须为每个连接至以太网网络的 S7-200 SMART CPU 输入以下 IP 信息：

- “IP 地址”(IP Address)：每个 CPU 或设备必须具有一个 Internet 协议 (IP) 地址。CPU 或设备使用此地址在更加复杂的路由网络中传送数据。每个 IP 地址分为四段，每段占 8 位，并以点分十进制格式表示（例如，211.154.184.16）。IP 地址的第一部分用于表示网络 ID（您正位于什么网络中？），地址的第二部分表示主机 ID（对于网络中的每个设备都是唯一的）。IP 地址 192.168.x.y 是一个标准名称，视为未在 Internet 上路由的专用网的一部分。

  说明

  所有 S7-200 SMART CPU 都有下列默认 IP 地址：192.168.2.1

  说明

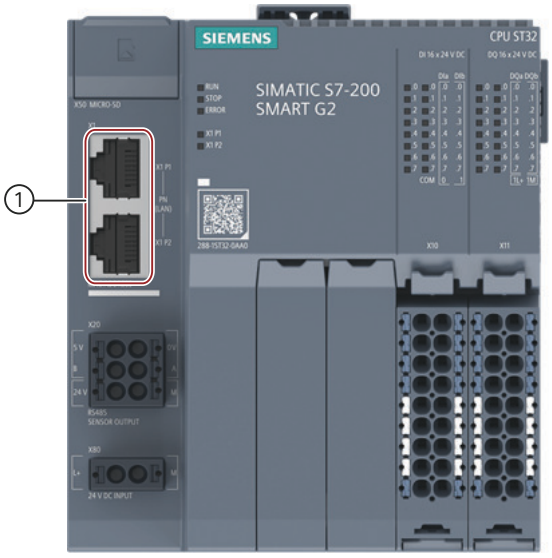
  必须为网络上的每台设备设定一个唯一的 IP 地址。

- “子网掩码”(Subnet Mask)：子网是已连接的网络设备的逻辑分组。在局域网 (LAN) 中，子网中的节点彼此之间的物理位置通常相对接近。子网掩码定义 IP 子网的边界。

  说明

  子网掩码 255.255.255.0 通常适用于本地网络。

- 默认网关 IP 地址：网关（或 IP 路由器）是 LAN 之间的链路。LAN 中的计算机可使用网关向其他网络发送消息，这些网络可能还隐含着其他 LAN。如果数据的目的地不在 LAN 内，网关会将数据转发给可将数据传送到其目的地的另一个网络或网络组。网关依靠 IP 地址来传送和接收数据包。



① PROFINET (LAN) 端口

有三种方法可组态或更改 CPU 或设备板载以太网端口的 IP 信息：

- 在“通信”(Communications) 对话框中组态 IP 信息（动态 IP 信息）
- 在“系统块”(System Block) 对话框中组态 IP 信息（静态 IP 信息）
- 在用户程序中组态 IP 信息（动态 IP 信息）

说明

CPU 中可以有静态或动态 IP 信息：

- 静态 IP 信息：如果已选中“系统块”(System Block) 中的“IP 地址数据固定为下面的值，不能通过其他方式更改”(IP address data is fixed to the values below and cannot be changed by other means) 复选框，则您所输入的以太网网络信息为静态信息。必须将静态 IP 信息下载至 CPU，然后才能在 CPU 中激活。而且，如果您想更改 IP 信息，则只能在“系统块”(System Block) 对话框中更改 IP 信息并将其再次下载至 CPU。
- 动态 IP 信息：如果未选中“系统块”(System Block) 中的“IP 地址数据固定为下面的值，不能通过其他方式更改”(IP address data is fixed to the values below and cannot be changed by other means) 复选框，则可通过其他方式更改 CPU 的 IP 地址，而且此 IP 地址信息被视为动态信息。可以在“通信”(Communications) 对话框中或使用用户程序中的 SIP_ADDR 指令更改 IP 地址信息。
- 对于静态和动态 IP，信息均存储在永久性存储器中。

在“通信”(Communications) 对话框中组态 IP 信息（动态 IP 信息）

通过“通信”(Communications) 对话框进行的 IP 信息更改立即生效，无需下载项目。

要访问此对话框，请执行以下操作之一：

- 在导航栏中单击“通信”(Communications) 按钮。
- 在项目树中，选择“通信”(Communications) 节点，然后按下 Enter，或双击“通信”(Communications) 节点。

可选择以下两种方式之一来访问 CPU：

- “已发现 CPU”(Found CPUs)：CPU 位于本地网络
- “已添加 CPU”(Added CPUs)：CPU 位于本地网络或远程网络（例如通过路由器访问另一网络中的 CPU）

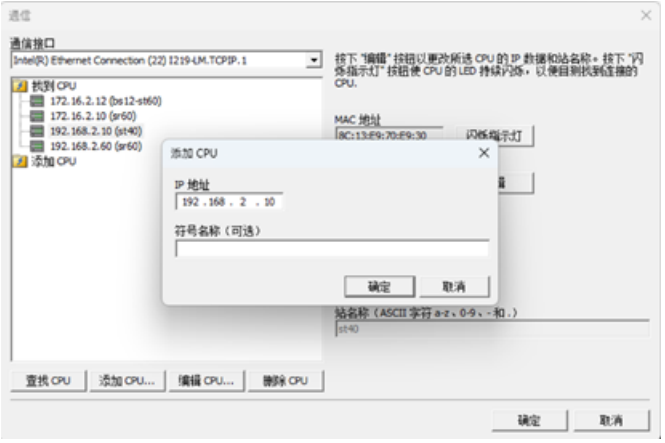
对与“已发现 CPU”（CPU 位于本地网络），可通过“通信对话框”(Communications dialog) 与您的 CPU 建立连接：

- 单击“通信接口”(Communication Interface) 下拉列表，并为您的编程设备选择“TCP/IP”网络接口卡 (NIC)。
- 单击“查找 CPU”(Find CPU) 按钮，将显示本地以太网网络中所有可操作 CPU（“找到 CPU”）。所有 CPU 都有默认 IP 地址。
- 高亮显示 CPU，然后单击“确定”(OK)。

对于“已添加 CPU”（CPU 位于本地网络或远程网络），可通过“通信对话框”(Communications dialog) 与您的 CPU 建立连接：

- 单击“通信接口”(Communication Interface) 下拉列表，并为您的编程设备选择“TCP/IP”网络接口卡 (NIC)。
- 单击“添加 CPU”(Add CPU) 按钮，执行以下任意一项操作：
  - 输入编程设备可访问但不属于本地网络的 CPU 的 IP 地址。
    您可以添加这些 CPU，在 STEP 7-Micro/WIN SMART 中选择其作为通信伙伴，然后像本地网络中的 CPU 一样对其进行编程和操作。只要存在有效的经由路由器的网络路径，STEP 7-Micro/WIN SMART 就可以与任意 S7-200 SMART CPU 进行通信。
  - 直接输入位于本地网络中的 CPU 的 IP 地址。

  可以在本地网络和/或远程网络中添加多个 CPU。通常情况下，STEP 7-Micro/WIN SMART 每次只能与一个 CPU 进行通信。所有 CPU 都有默认 IP 地址。
- 高亮显示 CPU，然后单击“确定”(OK)。



要输入或更改 IP 信息，执行以下操作：

- 单击所需的 CPU。
- 如果需要标别要组态或更改的 CPU，单击“闪烁指示灯”(Flash Lights) 按钮。此按钮会针对列表中高亮显示的 CPU 闪烁“STOP”、“RUN”和“FAULT”灯。
- 单击“编辑”(Edit) 按钮可对 IP 信息进行更改。
- 更改以下 IP 信息：
  - IP 地址
  - 子网掩码
  - 默认网关
  - 站名
- 按下“设置”(Set) 按钮。按下“设置”(Set) 按钮后，将在 CPU 中更新这些值。
- 完成后，单击“确认”(OK)。

当您在“通信”(Communications) 对话框中组态板载以太网端口的 IP 信息时，此信息为“动态”。如果未选中“系统块”(System Block) 对话框中的“IP 地址数据固定为下面的值，不能通过其他方式更改”(IP address data is fixed to the values below and cannot be changed by other means) 复选框，则必须在“通信”(Communications) 对话框中输入 IP 信息。可通过单击“设置”(Set) 按钮，输入新 IP 地址信息并更新 CPU 中的此信息。

在“系统块”(System Block) 对话框中组态 IP 信息（静态 IP 信息）

在“系统块”(System Block) 中进行的 IP 信息组态或更改为项目的一部分，在您将项目下载至 CPU 前不会生效。

要访问此对话框，请执行以下操作之一：

- 在导航栏中单击“系统块”(System Block) 按钮。
- 在项目树中，选择“系统块”(System Block) 节点，然后按下 Enter，或双击“系统块”(System Block) 节点。

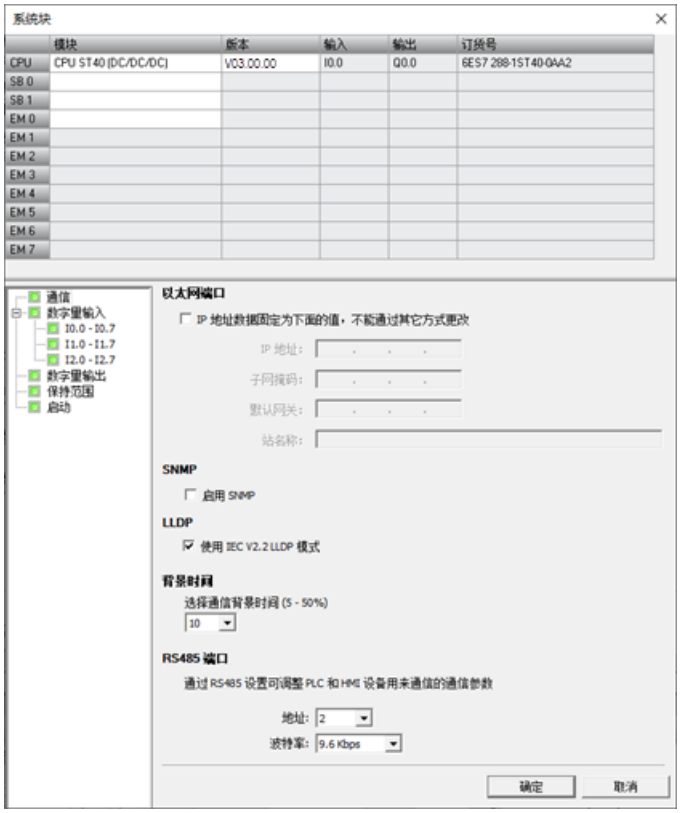
要输入或更改 IP 信息，执行以下操作：

- 如果尚未选中，单击“IP 地址数据固定为下面的值，不能通过其他方式更改”(IP address data is fixed to the values below and cannot be changed by other means) 复选框。以太网端口 IP 信息字段启用。
- 输入或更改以下 IP 信息：
  - IP 地址
  - 子网掩码
  - 默认网关
  - 站名

说明

站名称遵守标准 DNS（域名系统）命名规范。S7-200 SMART CPU 将站名限制为最多 63 个字符。站名可以包括小写字母 a 至 z、数字 0 至 9、连字符（减号）和句点。

不允许使用某些名称，这取决于用来设置站名的工具。站名称不能采用“n.n.n.n”格式，其中 n 取 0 到 999 之间的值。站名称不能以字符串“port-nnn”或“port-nnn-nnnnn”开头，其中“n”取数字 0 到 9 之间的值（例如，“port-123”和“port-123-45678”是非法站名称）。站名不能以连字符或句点开始或结束。



如果在“系统块”(System Block) 对话框中选中“IP 地址数据固定为下面的值，不能通过其他方式更改”(IP address data is fixed to the values below and cannot be changed by other

means) 复选框，则您为板载以太网端口输入的 IP 信息为静态信息。必须将静态 IP 信息下载至 CPU，然后才能在 CPU 中激活。如果您想更改 IP 信息，则只能在“系统块”(System Block) 对话框中更改此 IP 信息并将其再次下载至 CPU。

说明

如果已选中“IP 地址数据固定为下面的值，不能通过其他方式更改”(IP address data is fixed to the values below and cannot be changed by other means) 复选框，则无法在“通信”(Communications) 对话框中设置 IP 信息。

若要使用 SIP_ADDR 指令，必须取消选中“IP 地址数据固定为下面的值，不能通过其他方式更改”(IP address data is fixed to the values below and cannot be changed by other means) 复选框。

完成 IP 信息组态后，将项目下载到 CPU。所有具有有效 IP 地址的 CPU 和设备都显示在“通信”(Communications) 对话框中。

在用户程序中组态 IP 信息（动态 IP 信息）

SIP_ADDR 指令将 CPU 的 IP 地址设置为在其“ADDR”输入中找到的值，将 CPU 的子网掩码设置为在其“MASK”输入中找到的值，并将 CPU 网关设置为在其“GATE”输入中找到的值。

通过 SIP_ADDR 指令进行的 IP 信息组态或更改立即生效，无需下载项目。使用 SIP_ADDR 指令设置的 IP 地址信息存储在 CPU 中的永久存储器中。

有关详细信息，请参见“获取 IP 地址和设置 IP 地址（以太网）”。

10.4.3.3 搜索以太网网络上的 CPU 和设备

可在“通信”(Communication) 对话框中搜索和标识连接到以太网网络的 S7-200 SMART CPU。 要访问此对话框，请单击以下某项：

导航栏中的通信按钮

项目树中的通信

“视图”(View) 菜单功能区的“窗口”(Windows) 区域内“组件”(Component) 下拉列表中的“通信”(Communications)

“通信”(Communications) 对话框通过创建设备状态自动检测给定以太网网络上所有已连接且可用的 S7-200 SMART CPU。（请参见下图。） 选择 CPU 后，列出以下有关该 CPU 的详细信息：

- MAC 地址
- IP 信息
- 站名

CPU 的 IP 地址与 STEP 7-Micro/WIN SMART 项目不相关联。 打开 STEP 7-Micro/WIN SMART 项目不会自动选择 IP 地址或建立到 CPU 的连接。 每次创建新项目或打开现有 STEP 7-Micro/WIN SMART 项目，您都必须转至“通信”(Communications) 对话框建立到 CPU 的连接。 “通信”(Communications) 对话框将显示上次选择的 CPU。

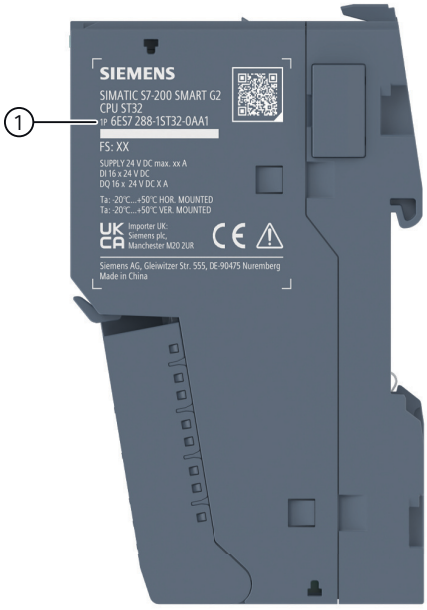
10.4.4 查找 CPU 上的以太网 (MAC) 地址

在以太网网络中，“介质访问控制”地址（MAC 地址）是制造商为了标识网络接口而分配的标识符。MAC 地址通常用制造商的注册标识号进行编码。

外观良好、按标准 (IEEE 802.3) 格式印制的 MAC 地址由六组数字组成，每组两个十六进制数，这些数字组用连字符 (-) 或冒号 (:) 分隔（例如 01-23-45-67-89-ab 或 01:23:45:67:89:ab）。

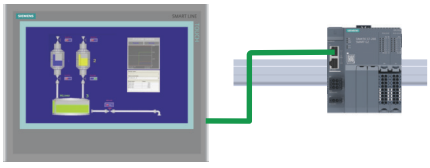
说明

每个 CPU 在出厂时都已装载了两个永久、唯一的 MAC 地址。您无法更改 CPU 的 MAC 地址。



① MAC 地址

10.4.5 HMI 与 CPU 通信



CPU 支持通过以太网端口与 HMI 通信。

设置 CPU 和 HMI 之间的通信时必须考虑以下要求：

- 组态/设置：
  - 必须为 CPU 组态一个 IP 地址。
  - 必须设置并组态 HMI，以便连接 CPU 的 IP 地址。

说明

机架安装的 CSM1277 4 端口以太网交换机可用于连接 CPU 和 HMI 设备。

- 支持的功能：
  - HMI 可以对 CPU 读/写数据。
  - 可基于从 CPU 重新获取的信息触发消息。
  - 系统诊断

要确保您的 CPU 和 HMI 正确通信，请依照下表中的步骤执行：

表格 10-1　组态 HMI 与 CPU 之间的通信时所需的步骤

步骤	任务
1	建立硬件通信连接 可通过以太网接口在 HMI 和 CPU 之间建立物理连接。由于 CPU 内置了自动跨接功能，所以对该接口既可以使用标准以太网电缆，又可以使用跨接以太网电缆。 有关详细信息，请参见“建立硬件通信连接”。
2	如果已创建包含 CPU 的项目，可在 STEP 7-Micro/WIN SMART 中打开该项目。如果没有创建，请创建项目并在项目中插入 CPU。
3	在项目中组态 IP 地址 使用相同的组态过程；但必须为 HMI 和 CPU 组态 IP 地址。必须为每个 CPU 和 HMI 设备都下载相应的组态。 有关详细信息，请参见“为项目中的 CPU 或设备组态或更改 IP 地址”(页 516)。

说明

可对 V 存储器特定范围的通信写入进行限制。这可能会影响到 HMI 通信。有关详细信息，请参见“组态系统安全”。

10.4.6 开放式用户通信

10.4.6.1 协议

概述

开放式用户通信 (OUC) 提供了一种机制，可使您的程序通过以太网发送和接收消息。您可以选择以太网协议作为传输机制：UDP、TCP 或 ISO-on-TCP

UDP（用户数据报协议）

用户数据报协议 (UDP) 使用一种协议开销最小的简单无连接传输模型。UDP 协议中没有握手机制，因此协议的可靠性仅等同于底层网络。无法确保对发送、定序或重复消息提供保护。对于数据的完整性，UDP 还提供了校验和，并且通常用不同的端口号来寻址不同函数。

TCP（传输控制协议）

传输控制协议 (TCP) 是一个因特网核心协议。在通过以太网通信的主机上运行的应用程序之间，TCP 提供了可靠、有序并能够进行错误校验的消息发送功能。TCP 能保证接收和发送的所有字节内容和顺序完全相同。TCP 协议在主动设备（发起连接的设备）和被动设备（接受连接的设备）之间创建连接。一旦连接建立，任一方均可发起数据传送。

TCP 协议是一种“流”协议。这意味着消息中不存在结束标志。所有接收到的消息均被认为是数据流的一部分。例如，客客户端设备向服务器发送三条消息，每条均为 20 个字节。服务器只看到接收到一条 60 字节的“流”（假设服务器在收到三条消息后执行一次接收操作）。

ISO-on-TCP

ISO-on-TCP 是一种使用 RFC 1006 的协议扩展。ISO-on-TCP 的主要优点是数据有一个明确的结束标志，这样您就可以知道何时接收到了整条消息。SPS7 协议 (Put/Get) 使用了 ISO-on-TCP 协议。ISO-on-TCP 仅使用 102 端口，并利用 TSAP（传输服务访问点）将消息路由至适当接收方（而非 TCP 中的某个端口）。

ISO-on-TCP 协议对接收到的每条消息进行划分。例如：客户端使用 ISO-on-TCP 协议向服务器发送三条消息。即使服务器在对收到的消息进行校验前会等待集齐所有消息，每条消息一经发出，服务器仍会接收每条消息且明确看到的是三条不同消息。这是 TCP 协议与 ISO-on-TCP 协议的不同之处。

10.4.6.2 连接

S7-200 SMART CPU 有两条用来执行连接管理的 OUC 指令：

- TCON 指令，用来建立一个主动连接（客户端）或打开一个被动连接（服务器）
- TDCON 指令，用来强制断开连接（例如，关闭连接）。RUN-to-STOP 转换强制关闭所有 CPU 创建的开放连接。

CPU 支持两种 OUC 连接类型：

- 主动：连接由本地 CPU 建立并维护。本地 CPU 负责向另一个设备发起连接请求并维护连接，这样，连接不会由于停滞状态而超时。
- 被动：在被动连接中，本地 CPU 打开一个端口和/或 TSAP，从而接收来自另一个设备的连接请求。

CPU 支持八个主动连接和八个被动连接。

CPU 根据传送给 TCON 指令的连接表创建被动或主动连接。UDP 连接始终为被动连接。TCP 和 ISO-on-TCP 连接使用一个组态参数来确定连接类型。

10.4.6.3 端口和 TSAP

概述

端口与传输服务访问点 (TSAP) 提供路由功能，能够将消息路由至 CPU 或其它设备内相应的接收器。

端口

您可以借助 UDP 和 TCP 协议选择本地端口号或远程端口号。当您选择 ISO-on-TCP 协议时，端口号固定为 102。

端口号必须在 1 到 49151 的范围内。建议端口号在 2000 到 5000 的范围内。S7-200 SMART CPU 端口号的范围和约束规则如下表所示：

端口号	描述
1 到 1999	• 您可以使用这些序号，但其不在推荐范围内。 • 有些端口不包括在内（见下述内容）。
2000 到 5000	推荐范围

端口号	描述
5001 到 49151	• 您可以使用这些序号，但其不在推荐范围内。 • 有些端口不包括在内（见下述内容）。
49152 到 65535	• 这些是动态端口或私有端口。 • 这些端口号的使用受到限制。

您不能将下表所示的端口号用于 S7-200 SMART CPU 中的本地端口号。远程端口号的使用不受限制：

端口号	描述
20	FTP 数据传输
21	FTP 控制
25	SMTP
80	网络服务器
102	ISO-on-TCP
135	用于 PROFINET 的 DCE
161	SNMP
162	SNMP 陷阱
443	HTTPS
34962 到 34964	PROFINET

无论是本地端口号还是远程端口号，您可以使多个主动连接使用同一个端口号。例如，一个 TCP 客户端可以在端口 2500 与多个服务器相连。通常，对于主动连接，本地端口和远程端口均为 2500 端口。

多个被动连接不能使用同一端口号作为本地端口号。例如，CPU 不允许在本地端口 2500 上存在多个 TCP 服务器（多个被动连接）。CPU 不知道向多个 2500 端口中的哪一个路由消息。

TSAP

传输服务访问点 (TSAP)，ISO-on-TCP 协议允许至单个 IP 地址的多个连接。TSAP 可唯一标识连接到同一个 IP 地址的这些通信端点连接。

端口 102 为 ISO-on-TCP 协议所专用。您不能为此协议设置端口号，不过，您可以为本地或远程伙伴设置 TSAP。

TSAP 规则如下：

- TSAP 须为 S7-200 SMART 字符串数据类型（长度字节，后接字符串）。
- TSAP 长度必须至少为 2 个字符，但不得超过 16 个 ASCII 字符。

- 本地 TSAP 不能以字符串“SIMATIC-”开头
- 如果本地 TSAP 恰好为 2 个字符，则必须以十六进制字符“0xE0”开头。例如：TSAP “$E0$01”是合法的，而 TSAP“$01$01”则是不合法的。（“$”字符表示后续值为十六进制字符。）

10.4.7 PROFINET 通信

10.4.7.1 简介

什么是 PROFINET IO？

PROFINET IO 是 PROFIBUS International 基于以太网的自动化标准。它定义了跨供应商通信、自动化和工程组态模型。

借助 PROFINET IO，可采用一种交换技术使所有站随时访问网络。因此，多个节点可同时传输数据，进而可更高效地使用网络。数据的同时发送和接收功能可通过交换式以太网的全双工操作来实现（带宽为 100 Mbps）。

PROFINET IO 系统由下列设备组成：

- PROFINET 控制器：对所连接 IO 控制器进行寻址以及与现场设备交换输入和输出信号的设备。
- PROFINET IO 设备：由 PROFINET 控制器进行监控的现场设备。PROFINET IO 设备由多个模块和子模块组成。

说明

S7-200 SMART PROFINET 网络中最多可有 10 个 PROFINET IO 设备和 64 个模块。

PROFINET 的目标

PROFINET 的目标如下：

- 实现工业联网，基于工业以太网（开放式以太网标准）
- 实现工业以太网与标准以太网组件的兼容性
- 凭借工业以太网设备实现高稳健性。工业以太网设备适用于工业环境（例如温度和抗干扰性）。
- 实现实时功能
- 其他现场总线系统的无缝集成

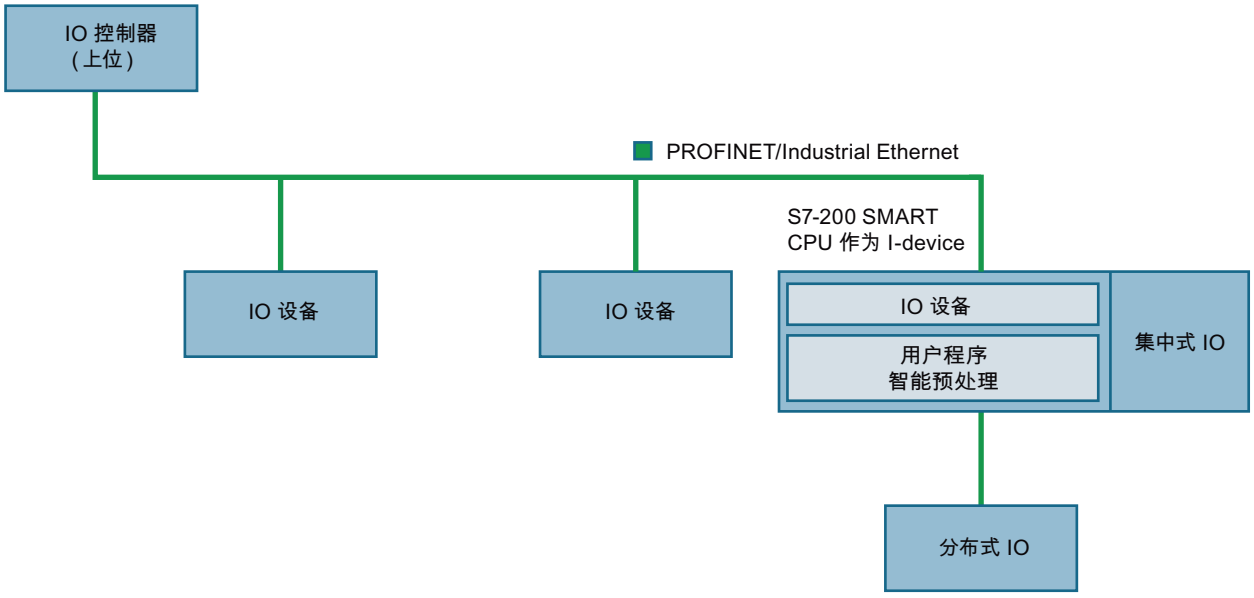
10.4.7.2 智能设备

智能设备功能

智能设备是采用“智能 IO 设备”组态的 CPU。

CPU 的智能设备功能允许在 CPU 与 IO 控制器之间进行数据交换，并且 CPU 可用作子过程的智能预处理单元。

预处理由 CPU 上的用户程序来完成。集中式或分布式 (PROFINET IO) I/O 中采集的过程值由用户程序进行预处理，并通过 PROFINET IO 接口提供给上位 IO 控制器。

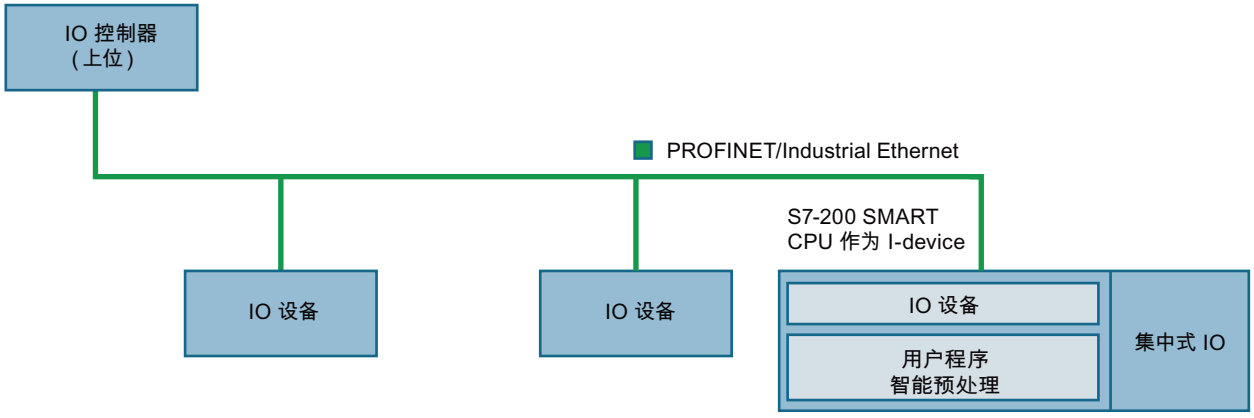


S7-200 SMART V3.0 或更高版本支持两种类型的智能设备：

- 不带下级 PROFINET IO 系统的智能设备
- 带下级 PROFINET IO 系统的智能设备

不带下级 PROFINET IO 系统的智能设备

智能设备自身没有分布式 IO。具有 IO 设备角色的智能设备的组态和参数分配与分布式 I/O 系统相同。



带下级 PROFINET IO 系统的智能设备

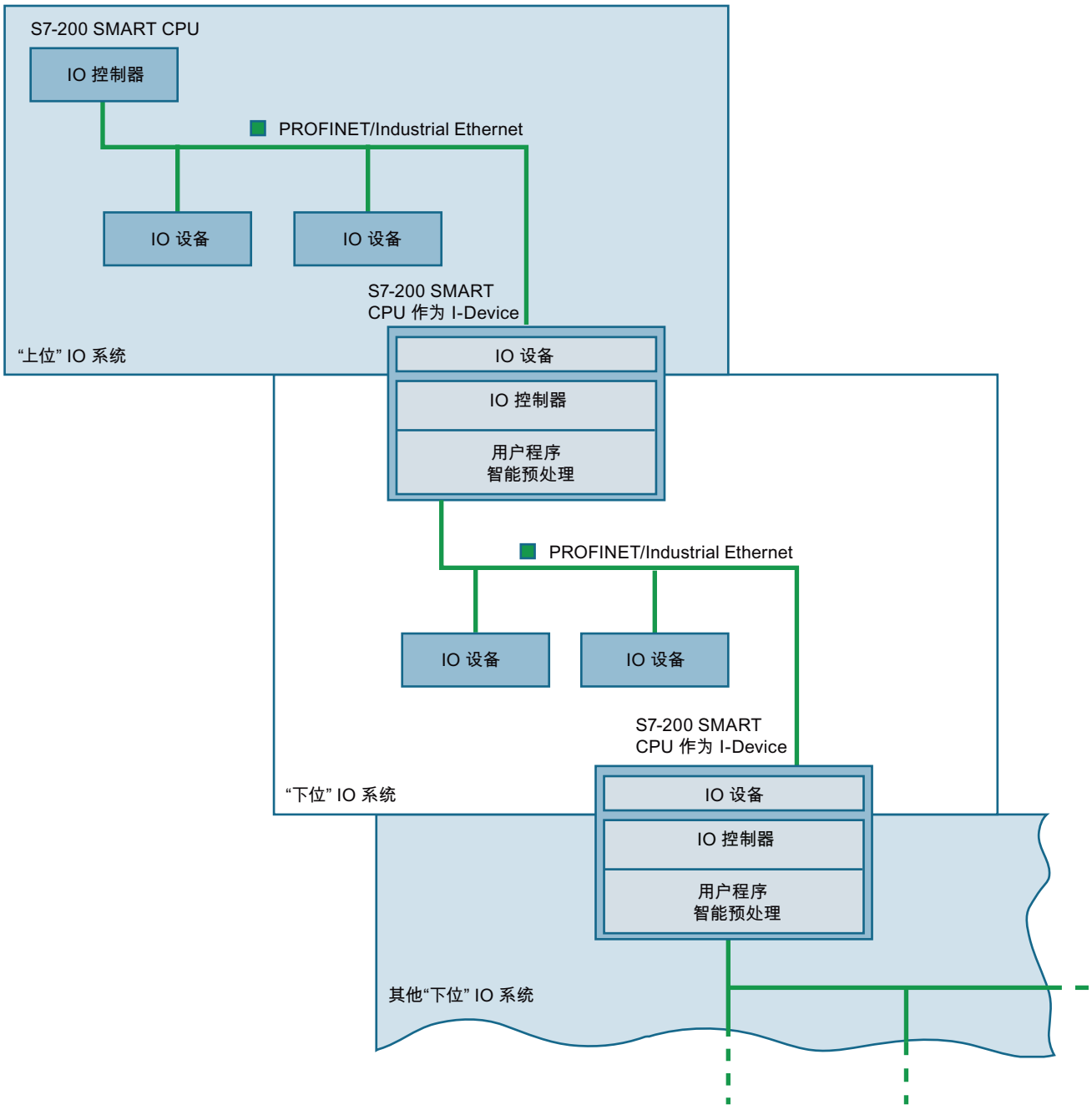
智能设备除了具有 IO 设备角色之外，也可以用作 PROFINET 接口上的 IO 控制器。

这意味着，智能设备可通过其 PROFINET 接口而成为上位 IO 系统的一部分，并可作为 IO 控制器来支持自身的下级 IO 系统。

下级 IO 系统可以包含智能设备。这样就可实现分层的 IO 系统结构。

说明

如果 S7-200 SMART CPU 作为带下级 PROFINET 系统的智能设备，支持的最大下级 IO 设备数为 10，并且只支持一个上位控制器。



10.4.7.3 XML 中的设备数据库文件：GSDML

PROFINET 设备的属性存储在 XML 文件中，其结构和规则由 GSDML 模式确定。

用于描述 GSD 文件的语言为 GSDML（通用站描述标记语言）。可从 PROFINET 设备制造商处获取 GSDML 模式（作为模式文件）。

STEP 7-Micro/WIN SMART V3 支持 GSDML 规范 2.45 或更早版本。导入 GSDML 文件时，STEP 7-Micro/WIN SMART V3 会验证 GSDML 文件。如果对象通过验证，STEP 7-Micro/WIN SMART V3 会将其置于 PROFINET 设备目录树 (页 539)中。

将 IO 设备添加到 PROFINET 网络前，可将其 GSDML 文件导入 STEP 7-Micro/WIN SMART V3 中。

10.4.7.4 PROFINET 设备命名

指定设备名称

在连接至 CPU 之前，PROFINET 网络上的设备必须具有指定的名称。如果尚未为设备指定名称，或需要更改设备名称，请使用“查找 PROFINET 设备”为 PROFINET 设备指定名称。

必须在 PROFINET 向导和“查找 PROFINET 设备”对话框中为每台 PROFINET 设备指定相同的名称。如果在这两处缺少名称或名称不匹配，则 PROFINET IO 数据交换模式将不会运行。

10.4.7.5 PROFINET 网络的 LED 状态指示灯

前面板上的 CPU 状态 LED “Error”指示 PROFINET 的工作状态：

- 对于 IO 设备，发生以下情况之一时，LED 以 1 Hz 的频率呈红色闪烁：
  - 任一 IO 设备与 IO 控制器失去连接
  - 任一 IO 设备发出报警
- 对于智能设备，发生以下情况之一时，LED 以 1 Hz 的频率呈红色闪烁：
  - 任一智能设备与上位控制器断开连接
  - 任一智能设备已连接上位控制器，但该智能设备的组态与上位控制器的组态不匹配。

10.4.7.6 组态

打开硬件网络拓扑

可使用硬件网络拓扑组态、分配参数并链接各个 PROFINET 硬件组件。

创建 PROFINET 网络

PROFINET 系统由 PROFINET 控制器及其分配的 PROFINET 设备组成。可为 CPU 选择不同角色，以组态 PROFINET 网络。

PROFINET 向导将引导您完成以下任务，以创建 PROFINET 网络：

- 将 CPU 组态为控制器 (页 539)
- 将 CPU 组态为不带下级 PROFINET IO 系统的智能设备 (页 547)
- 将 CPU 组态为带下级 PROFINET IO 系统的智能设备 (页 551)

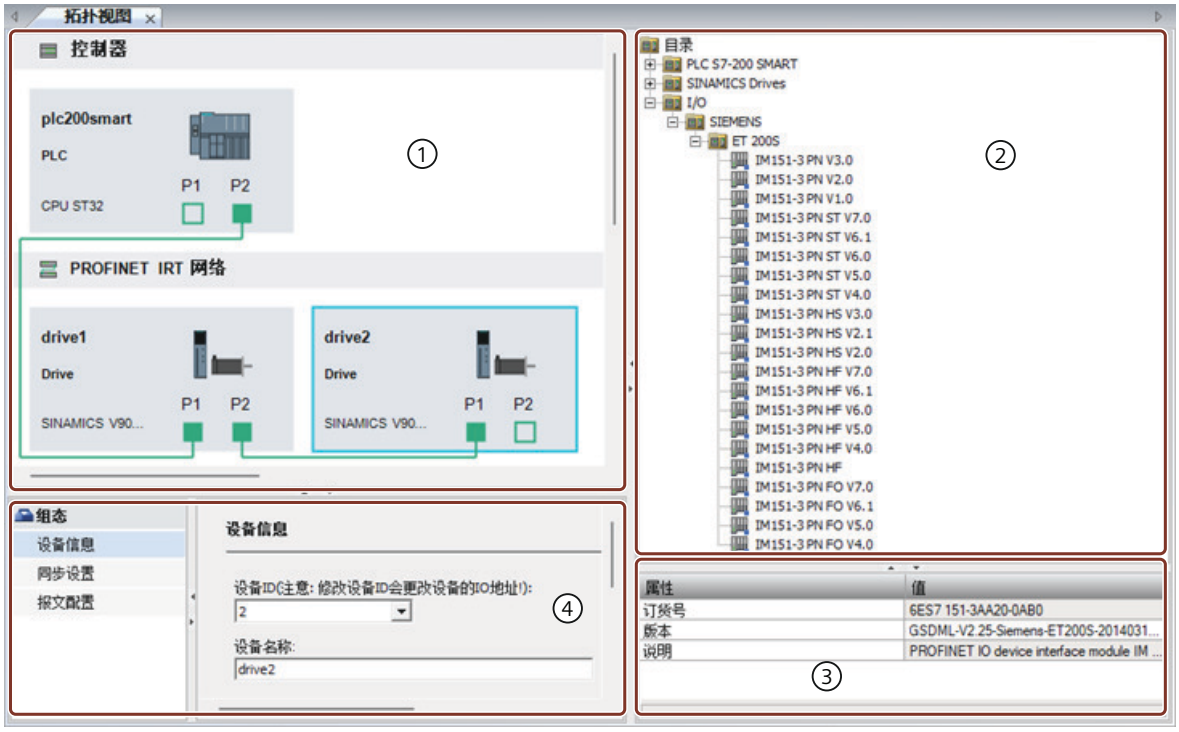
硬件网络拓扑

硬件网络拓扑的功能

硬件网络拓扑用于 PROFINET RT+IRT 网络的组态。双击项目树中的“硬件网络拓扑”，打开硬件网络拓扑。 硬件网络拓扑用于对设备和模块进行组态和参数分配。

硬件网络拓扑的结构

硬件网络拓扑由以下部分组成：



① 拓扑视图 (页 833)

② 设备目录

③ 设备属性

④ 设备组态

可以使用鼠标更改网络视图中区域之间的分隔。要变更分割线的位置，在区域间隔间单击，并在按住鼠标按钮的同时左右移动分隔线更改间距大小。通过快速拆分器（两个小箭头键），可以使用单击来最小化视图、最大化视图或恢复上一次选择的拆分。

为了高效地管理和配置网络资源，硬件拓扑网络提供了两种视图：拓扑视图和设备视图。在拓扑视图中，您可以对整个网络拓扑结构进行组态。当您需要对单个设备及其内部模块进行编辑和配置时，则须进入设备视图。

工具栏

该工具栏提供以下功能：

图标	含义
	将 PLC 设置为 RUN 模式
	将 PLC 设置为 STOP 模式
	编译
上传	将项目从 PLC 上传到 STEP 7-Micro/WIN SMART V3
下载	从 STEP 7-Micro/WIN SMART V3 下载到 PLC
	拓扑比较
	扫描拓扑 启动对当前在线拓扑的检测。

设备目录

当您打开硬件网络拓扑时，设备目录树将自动加载以下设备：

- CPU 内嵌的 PROFINET 设备
- 导入 GSD 文件支持的 PROFINET 设备

设备属性

当您在设备目录中选择一个设备时，设备属性窗口将显示该设备的基本属性。

属性	说明
定货号	只读
版本	您可以通过下拉菜单选择设备的 GSD 文件版本。如果系统中安装了多个包含相同设备的 GSD 文件，您可以在此处进行选择。完成设置后，设备在树状视图中的版本将更新为您所选的版本。默认情况下，系统会自动选择最高版本的 GSD 文件。
说明	只读

将 CPU 组态为控制器

将设备添加到 PROFINET 网络

当 IRT 和 RT 设备被添加到拓扑视图时，系统会自动为这些设备分配唯一的 ID。ID 分配过程将从最小的可用 ID 开始，并按顺序递增地分配给每个 IRT/RT 设备。

对于添加在 PROFINET 网络中的设备，其命名将按照添加顺序自动生成。

- 驱动器命名格式为 drive1, drive2, drive3...
- I/O 模块命名格式为 dev1, dev2, dev3...
- PLC 模块命名格式为 PLC1, PLC2, PLC3...

拖放设备

要通过项目树将设备添加到硬件配置中，请按以下步骤操作：

1. 打开硬件网络拓扑。
2. 在设备目录中选择所需的设备。
3. 单击所选的设备，将其选中。

4. 必要时，通过设备属性窗口中的下拉列表设置固件版本。
5. 将设备拖放到拓扑视图中。

设备的可拖放网络取决于其支持的 PROFINET 类型：

- RT 设备： 只能拖动到 RT 网络，不能拖动到 IRT 网络。
- IRT 设备： 只能拖动到 IRT 网络。
- 同时支持 RT 和 IRT 的设备： 可以拖动到 RT 网络 或 IRT 网络。

批量添加

1. 在设备树视图中，您可以通过右键菜单中的“批量添加”选项。
2. 选择批量添加设备的位置以及要添加的数量。

3. 点击“确认”。

PROFINET device 变量

将 PROFINET 设备添加到硬件网络拓扑的组态中时，STEP 7-Micro/WIN SMART V3 会将 PROFINET 设备变量添加到 PROFINET device 变量表中。

组态控制器

要在拓扑视图中更改控制器的属性和参数，请按以下步骤操作：

1. 在拓扑视图中，在控制器层中选中 PLC。
2. 在组态窗口点击“设备角色”。

说明

将 CPU 组态为控制器时，请确保已为 PROFINET 设备导入 GSDML 文件。

3. 在组态窗口点击“以太网”。

以太网
- 作为控制器时：必须配置固定 IP 地址
- 作为智能设备时：支持固定 IP 或动态分配

站名称　如您更改站名，设备名称将同步更新到设备视图中。

4. 在组态窗口点击“通信”，为控制器分配“发送时钟”和“启动时间”。

“启动时间”设置影响 CPU 通电后其切换至 RUN 模式所需的时间。“启动时间”的默认值为 10 s，最大值为一分钟：

– 如果“启动时间”设置为零，CPU 将立即切换到 RUN 模式。
– 如果在“启动时间”内建立完整连接，则 CPU 会在建立连接后切换至 RUN 模式。
– 如果“启动时间”结束后未建立连接，则 CPU 在“启动时间”结束后切换至 RUN 模式。

说明

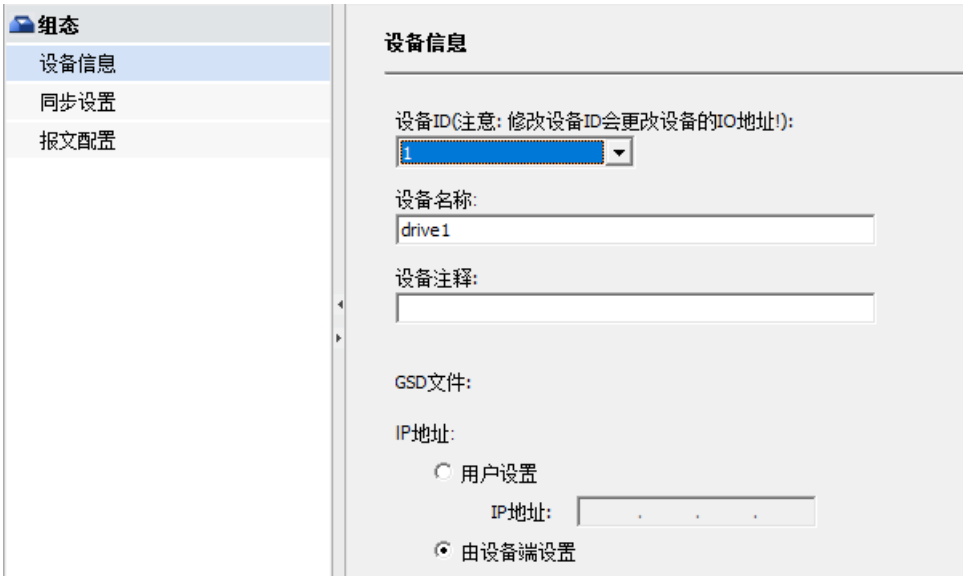
若当前项目的“发送时钟”设置为 1ms，当您添加最快发送时钟为 2ms 的 IRT 设备时，系统将自动把“发送时钟”调整为 2ms，并给出相应提示。

组态 IRT 设备

在硬件网络拓扑中插入设备后，可在拓扑视图中对设备的默认属性进行编辑。

要在拓扑视图中更改设备的属性和参数，请按以下步骤操作：

1. 在拓扑视图中，在 IRT 层中选中设备。
2. 在组态窗口点击“设备信息”。



表格标题	含义
设备 ID	模块通过 PROFINET 连接 PLC 时，PROFINET 的设备编号。
设备名称	设备名称必须和所插入模块的名称一致。 示例： • 如果模块插入一个名为“IODevice1”的前端模块中，则必须在此处设置为“IODevice1 ”。
设备备注	为设备添加注释。
IP 地址	组态 IP 地址的方法有两种： • 用户设置：由控制器给设备分配 IP 地址 • 固定 IP

3. 在组态窗口点击“同步设置”。

等时同步

更新时间	同控制器“通信”中设置的“发送时钟”。
数据保持	可将数据保持时间设定为“更新时间的 n 倍。取值范围为 3-255，默认值为 3。
Ti/To 值	Ti/To 值有以下两种设置： • 自动最小化： 自动计算并采用 Ti/To 最小值。 • 手动： 只有在选择了“手动”设置时才可以编辑 Ti 和 To。

4. 在组态窗口点击“报文设置”，选择或修改配置报文。

– 主报文：可选 105/5 号报文， 默认报文为 105 号。
– **附加报文：**可选 750 号报文或不启用， 默认不启用。
– 激活诊断：若设备的 GSD 文件包含“激活诊断”参数，则会显示该诊断功能；否则不显示。

通信时间 T_i、T_o、T_{DC}

计算跟随误差时，不会将设定值到驱动器的传输时间和实际位置值到控制器的传输时间计算在内。传输时间由以下通信时间得出：

- T_i：导入过程值的时间
- T_o：导出过程值的时间
- T_{DC}： PROFINET 接口的发送时钟

跟随误差是通过从通信时间 $T_i + T_o + T_{DC}$ 和 运动控制周期 T_{servo} 之前的延迟位置设定值中减去控制器中的实际位置计算得出的。

报文类型

报文	简要说明
标准报文	
5	• 控制字 STW1 和 STW2，状态字 ZSW1 和 ZSW2 • 速度设定值 32 位 (NSET)，实际速度值 32 位 (NACT) • 实际编码器值 1（G1_XIST1、G1_XIST2）（电机编码器） • 动态伺服控制 (DSC)[1)] – 速度预控制值 – 位置差值 (XERR) – Kv 因子增益位置控制 (KPC)
Siemens 报文（含扭矩限值）	
105	• 控制字 STW1 和 STW2，状态字 ZSW1 和 ZSW2 • 速度设定值 32 位 (NSET)，实际速度值 32 位 (NACT) • 实际编码器值 1（G1_XIST1、G1_XIST2）（电机编码器） • 动态伺服控制 (DSC)[1)] – 速度预控制值 – 位置差值 (XERR) – Kv 因子增益位置控制 (KPC) • 扭矩限值

1) 要使用动态伺服控制 (DSC)，必须将驱动器的电机编码器（报文中的第一个编码器）用作轴的第一个编码器。

组态 RT 设备

在硬件网络拓扑中插入设备后，可在拓扑视图中对设备的默认属性进行编辑。

要在拓扑视图中更改设备的属性和参数，请按以下步骤操作：

1. 在拓扑视图中，在 RT 层中选中设备。
2. 在组态窗口点击“设备信息”。

表格标题	含义
设备 ID	模块通过 PROFINET 连接 PLC 时，PROFINET 的设备编号。
设备名称	设备名称必须和所插入模块的名称一致。 示例： • 如果模块插入一个名为“IODevice1”的前端模块中，则必须在此处设置为“IODevice1 ”。
设备备注	为设备添加注释。
IP 地址	组态 IP 地址的方法有两种： • 用户设置：由控制器给设备分配 IP 地址 • 固定 IP

3. 在组态窗口点击“同步设置”。

更新时间： 数据交换的周期。
数据保持：默认值为 3。

添加/删除/移动 RT 模块

简介

在拓扑视图中将设备从硬件目录添加到组态后，即可向设备中添加模块。设备视图中图形区域与表格区域实现联动。当您在一个区域进行修改时，另一区域将自动同步刷新，确保数据和配置始终保持一致。

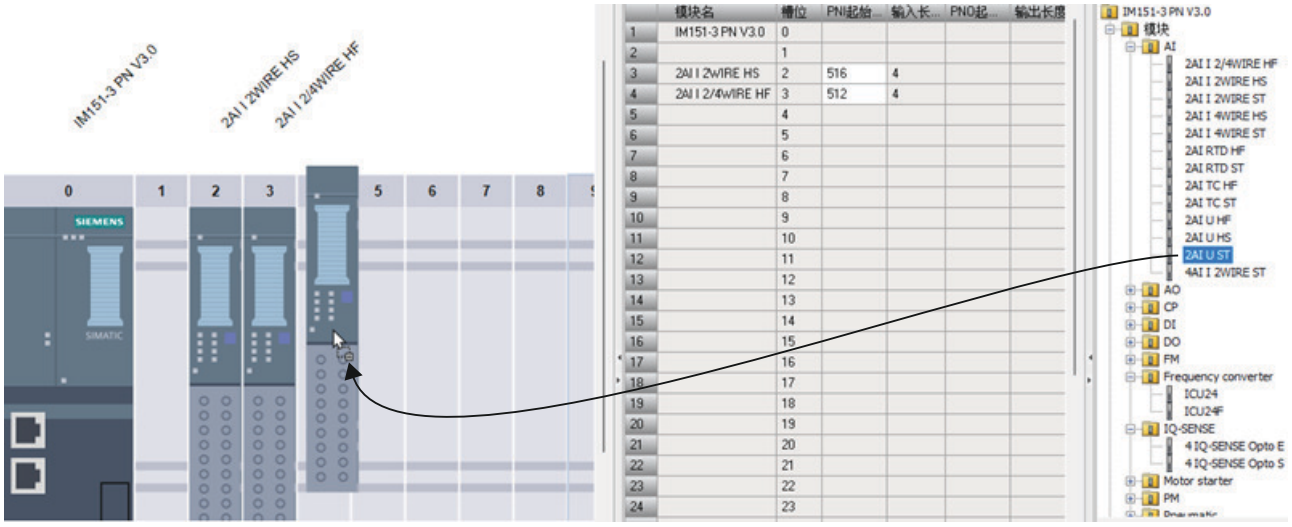
说明

不允许将任何其它模块拖入或移动到 SLOT-0。

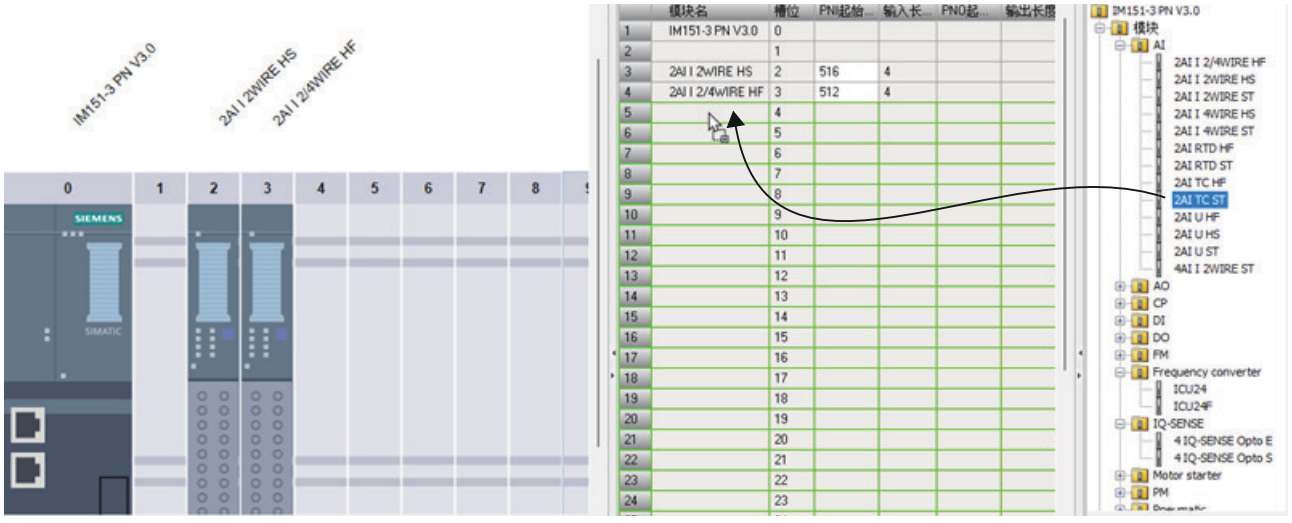
添加模块

从硬件目录中添加模块，请执行以下步骤：

1. 在硬件目录中选择所需的模块。
2. 必要时，通过硬件目录中的下拉列表设置固件版本。
3. 通过拖放操作将该模块从硬件目录移动到图形或表格区域内可用的有效插槽中：
   – 将模块拖到图形区域空闲的插槽上。



   – 将模块拖到表格区域空闲的插槽上。



添加模块后，STEP 7-Micro/WIN SMART V3 会将 PROFINET 设备变量添加到 PROFINET device 变量表中。

删除模块

通过以下任一方式删除模块：

- 用鼠标在图形或表格区域右键点击模块，在弹出的上下文菜单中选择”删除“。
- 在图形或表格区域选中模块，按下 Delete 键。

移动模块

要移动模块，请按以下步骤操作：

1. 选择要移动的模块。
2. 将模块拖放到将要移动到的位置。

所选的模块将移动到目标位置。

将 CPU 组态为不带下级的智能设备

要求

确保已在“系统块”对话框中选择了需要的 CPU 模块。

操作步骤

请按以下步骤组态智能设备：

1. 打开硬件网络拓扑。
2. 在拓扑视图中，在控制器层中选中 PLC。
3. 在组态窗口组态，勾选 “智能设备” 前的复选框。
   若勾选“PROFINET 接口参数由上位控制器分配”，设备端口组态功能将可用。否则，设备端口组态功能将不可用。

4. 在组态窗口组态以太网和站名。

以太网

- 分配固定 IP 地址：设备组态时， 配置的 IP 地址和站名必须与智能设备的实际 IP 地址和站名保持一致。
- 通过其他途径获取 IP 地址：设备组态时，将 IP 地址设定为“用户设置”，并填写具体的 IP 地址。

5. 在组态窗口点击”通信“，为控制器分配“发送时钟”(send clock) 和“启动时间”(start up time)。

“启动时间”(start up time) 设置影响 CPU 通电后其切换至 RUN 模式所需的时间。“启动时间”(start up time) 的默认值为 10 s，最大值为一分钟：

– 如果“启动时间”(start up time) 设置为零，CPU 将立即切换到 RUN 模式。
– 如果在“启动时间”(start up time) 内建立完整连接，则 CPU 会在建立连接后切换至 RUN 模式。
– 如果“启动时间”(start up time) 结束后未建立连接，则 CPU 在“启动时间”(start up time) 结束后切换至 RUN 模式。

6. 在组态窗口点击”智能设备“，组态传送区。
传送区用作与智能设备 CPU 的用户程序之间的接口，在上位控制器与智能设备之间交换通信数据。用户程序对输入进行处理并输出处理结果。

说明

- 支持的最大传送区数目为 8。
- 必须至少组态一个传送区。

“传送区”(Transfer Area) 表列出了以下项目：

项目	描述
传送区名 (Transfer area name)	一个子插槽的子模块名。 可以定义传送区名或使用默认传送区名。 传送区名不会下载到 PLC。 **注：** 支持的最大字符数为 64 个。
子插槽 (Subslot)	子插槽号。 子插槽号从 1000 开始，系统会自动为子插槽号生成连续数字。
类型 (Type)	传送区类型。 可选择一个传送区作为输入或输出。
地址 (Address)	传送区 IO 地址。 可输入输入或输出地址。 • 输入地址范围为 IB1152 到 IB1279。 • 输出地址范围从 QB1152 到 QB1279。
长度 (Length)	传送区 IO 数据长度。 • 传送区最大总输入大小为 128 字节。 • 传送区最大总输出大小为 128 字节。
注释 (Comment)	在传送区中添加注释。

“导出 GSDML 文件”(Export GSDML) 字段会列出以下项目：

项目	描述
标识 (Designation)	默认显示设备系列名。例如 CPU ST32。 也可根据需要输入任意信息： • 支持中文、英文和数字字符。 • 不支持字符“|”、“\\”、“/”、“:”、“\”、"、“<”、“>”、“*”和“?”。 • 支持的最大字符数为 20 个。
说明 (Description)	默认显示 CPU 信息，也可根据需要输入任意信息。
文件名 (File name)	会自动生成 GSDML 文件名，且文件名采用指定的格式。
输出文件夹 (Output folder)	用于保存 GSDML 文件的文件夹。可单击“浏览”(Browse) 按钮选择一个文件夹，也可以使用默认文件夹。

通过 GSD 文件组态智能设备

如果在其它项目或其它工程组态系统中使用智能设备，请按上述步骤组态上一级 IO 控制器和智能设备。

在组态了传送区后应单击“导出”(Export) 按钮，以便从该智能设备创建一个新的 GSD 文件。此 GSD 文件代表在其它项目组态的智能设备。

将 GSD 文件导入其它项目。

组态带下级 PROFINET IO 系统的智能设备项目

要求

确保已在“系统块”对话框中选择需要的 CPU 模块。

操作步骤

要组态带下级 PROFINET 网络的智能设备项目，请按下列步骤操作：

1. 打开硬件网络拓扑。
2. 在硬件网络拓扑中添加你需要的设备 (页 537)。
3. 选中“智能设备”(I-Device) 复选框。“控制器”(Controller) 为默认选中。
   如果组态带下级 PROFINET IO 系统的智能设备，则无法由上位控制器设置智能设备 PROFINET 接口参数。
4. 在智能设备组态页面将 CPU 组态为智能设备 (页 547)。
   在此步导出一个智能设备 GSDML 文件。
5. 在控制器组态页面中将 CPU 组态为控制器 (页 539)。
6. 将该智能设备项目下载到 CPU。

说明

对于带下级 PROFINET IO 系统的智能设备，如果智能设备的下级 IO 设备出错，则该智能设备会向上位控制器发送以下错误：设备运行时出错。

10.5 RS485

10.5.1 通信选择

10.5.1.1 概述

RS485 网络是一种差分（多点）网络，每个网络最多可有 126 个可寻址节点，每个网段最多可有 32 个设备。 中继器用来分割网络。 中继器不是可寻址节点；因此，中继器并不包括在可寻址节点计数中，但会包括在每个网段的装置计数中。

RS485 支持高速数据传输（12 Mbit/s 时传输距离为 100 m，187.5 Kbit/s 时传输距离为 1 km）。

RS485 可使用 PPI 协议和自由端口：

- PPI 协议：可在 RS485 或 RS232（半双工）上运行。 可能的连接包括：
  - PPI 协议设备
  - RS485 HMI 显示器
- 自由端口： 可在 RS485 或 RS232（半双工）上运行。 可能的连接包括：
  - RS485 兼容的设备（例如条形码扫描器）
  - 具有 RS485 接口的设备（例如控制系统）
  - 使用自由端口的第三方设备
  - 调制解调器

10.5.1.2 PPI 协议

定义

PPI 是一种主站-从站协议：主站设备向从站设备发送请求，从站设备进行响应。请参见下图。从站设备并不发出消息，而是等待主站向其发送请求或对其轮询，要求其进行响应。

HMI：主站　S7-200 SMART：从站

主站通过由 PPI 协议管理的共享连接与从站进行通信。PPI 不会限制可与任何一个从站通信的主站数目；但您无法在网络中安装 32 个以上主站。

PPI 协议和 S7-200 SMART CPU

PPI 高级协议允许网络设备在设备之间建立逻辑连接。对于 PPI 高级协议，每台设备可提供的连接数是有限的。请参见下表中 S7-200 SMART V3 CPU S7-200 SMART G2 CPU 支持的连接数。

所有 S7-200 SMART V3 CPU S7-200 SMART G2 CPU 都支持 PPI 和 PPI 高级协议。

表格 10-2 S7-200 SMART CPU 的连接数

模块	波特率	连接
RS485 端口	9.6 Kbps、19.2 Kbps 或 187.5 Kbps	5
RS485/RS232 信号板	9.6 Kbps、19.2 Kbps 或 187.5 Kbps	4

10.5.2 通信基础

10.5.2.1 S7-200 SMART 网络通信基础

S7-200 SMART 旨在解决通信和网络需求，不仅支持最简单的网络，而且支持更复杂的网络。S7-200 SMART 还提供可与其他设备通信的工具，例如使用自身通信协议的打印机和计量器。

STEP 7-Micro/WIN SMART V3 让网络设置和组态变得简单容易。

波特率和网络地址定义 (页 553)

为 S7-200 SMART CPU 设置波特率和网络地址 (页 554)

10.5.2.2 波特率和网络地址定义

波特率

数据在网络中的传输速度成为波特率，通常以千位 (Kbps) 或兆位 (Mbps) 为单位。波特率衡量给定时间内可传输的数据量。例如，波特率为 19.2 Kbps 说明传输率为每秒 19,200 位。

每一台通过给定网络进行通信的设备都必须组态为以相同波特率传输数据。因此，网络的最快波特率由连接到该网络的速度最慢的设备决定。

下表列出了 S7-200 SMART CPU 支持的波特率。

表格 10-3　S7-200 SMART CPU 支持的波特率

网络	波特率
PPI 协议	仅 9.6 Kbps、19.2 Kbps 和 187.5 Kbps
自由端口模式	1.2 Kbps 到 115.2 Kbps

网络地址

网络地址是分配给网络中每台设备的唯一编号。网络地址唯一可确保数据传输到正确设备或从正确设备中检索数据。S7-200 SMART CPU 支持的网络地址为 0 到 126。下表列出了 S7-200 SMART 设备的默认（出厂）设置。

表格 10-4　S7-200 SMART 设备的默认地址

S7-200 SMART 设备	默认地址
STEP 7-Micro/WIN SMART	0
HMI	1
S7-200 SMART CPU	2

10.5.2.3　为 S7-200 SMART CPU 设置波特率和网络地址

简介

要通过 RS485 网络与 STEP 7-Micro/WIN SMART 或 SIMATIC HMIs (页 525) 通信，必须组态 S7-200 SMART CPU 的 RS485 网络地址和波特率。

RS485 端口网络地址必须不同于 RS485 网络中所有设备的网络地址，RS485 端口波特率必须与 RS485 网络中其它设备的波特率相同。默认 RS485 端口网络地址为 2，每个 CPU 端口的默认 RS485 端口波特率为 9.6 Kbps。

CPU 的系统块存储 RS485 端口网络地址和波特率。为 CPU 选择参数后，必须将项目下载到 S7-200 SMART CPU。

步骤

要访问“系统块”(System Block) 对话框，请单击以下某项：

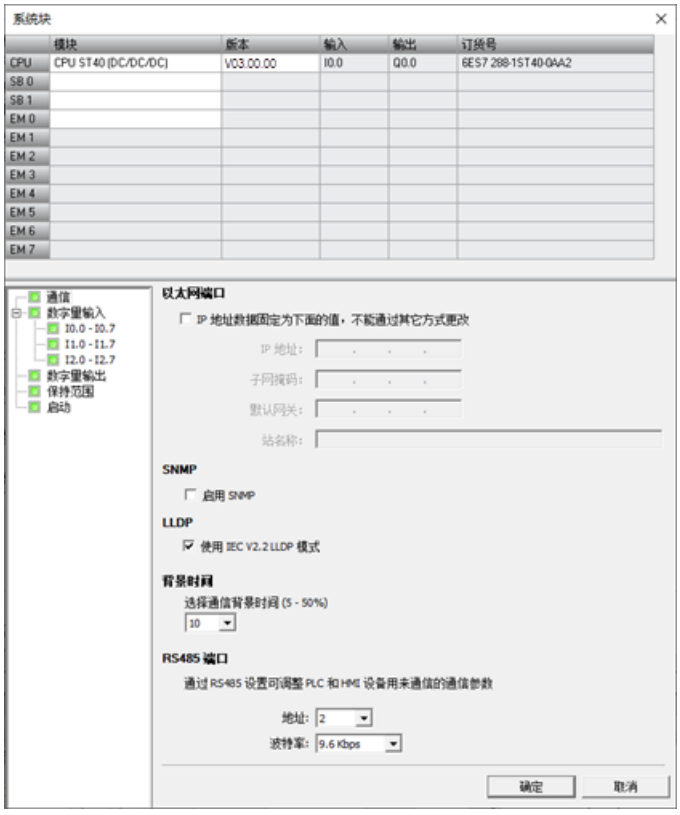
导航栏中的“系统块”(System Block) 按钮

项目树中的系统块

“视图”(View) 菜单功能区的“窗口”(Windows) 区域内“组件”(Component) 下拉列表中的“系统块”(System block)

选择“系统块”(System Block) 对话框后，执行以下步骤：

1. 为 RS485 端口选择网络地址和波特率。
2. 将系统块下载到 CPU。



说明

使用 SM 存储器设置自由端口协议波特率。

10.5.3 RS485 网络组态示例

10.5.3.1 单主站 PPI 网络

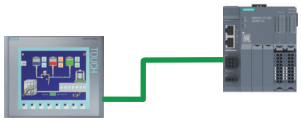
简介

仅使用 S7-200 SMART 设备的情况下可进行以下网络组态：

- 单主站 PPI 网络
- 多主站和多从站 PPI 网络
- 复杂 PPI 网络

单主站 PPI 网络

在下图的网络示例中，人机界面 (HMI) 设备（例如 TP 或 KP）为网络主站：

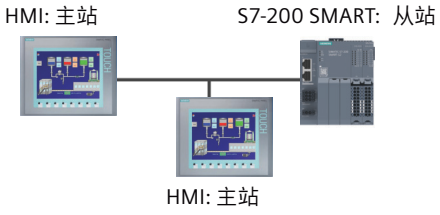


HMI：主站　S7-200 SMART：从站

在该网络示例中，CPU 是响应主站请求的从站。

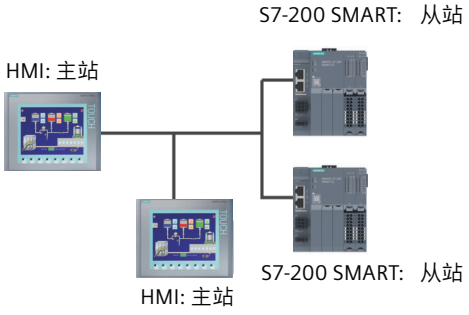
10.5.3.2 多主站和多从站 PPI 网络

下图显示了带一个从站的多个主站网络示例。HMI 设备共享网络。



HMI 设备是主站，必须具有单独的网络地址。S7-200 SMART CPU 是从站。

下图所示为多台主站与多台从站进行通信的 PPI 网络。在本示例中，HMI 可以向任意 CPU 从站请求数据。



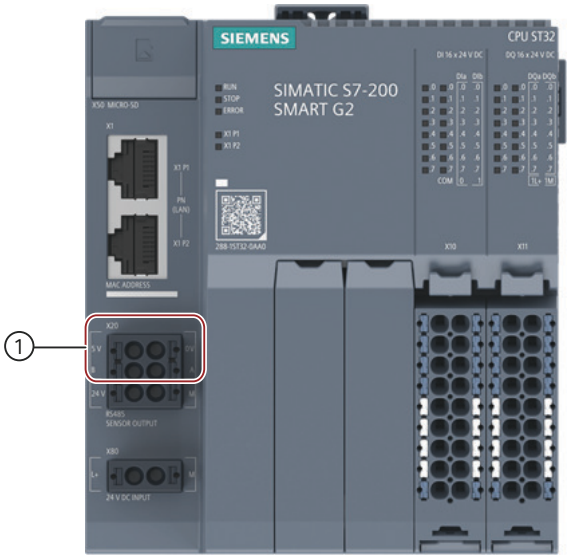
所有设备（主站和从站）的网络地址都不相同。S7-200 SMART CPU 为从站。

10.5.4 分配 RS485 地址

10.5.4.1 为项目中的 CPU 或设备组态或更改 RS485 地址

必须为每个连接至 RS485 网络的 S7-200 SMART CPU 输入以下信息：

- RS485 地址：每个 CPU 或设备必须具有一个 RS485 地址。CPU 或设备使用此地址在网络中传送数据。
- 波特率：数据在网络中的传输速度称为波特率，通常以 Kbps 或 Mbps 为单位。波特率衡量给定时间内可传输的数据量。（例如，传输速率为 19.2 Kbps）。

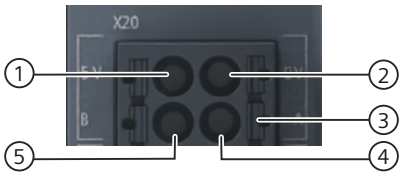


① RS485 端口

必须在“系统块”(System Block) 对话框中组态或更改 CPU 或设备的板载 RS485 端口的 RS485 网络信息，并将组态下载至 CPU。

RS485(X20)

下图列出了 RS485 引脚分配说明和弹簧式脱扣器的位置。



① +5 V 输出，内部有 100 Ω 串联电阻

② 5 V 回流， 逻辑公共端

③ 弹簧式脱扣器

④ RS485 信号 A

⑤ RS485 信号 B

在“系统块”对话框中组态 RS485 网络信息

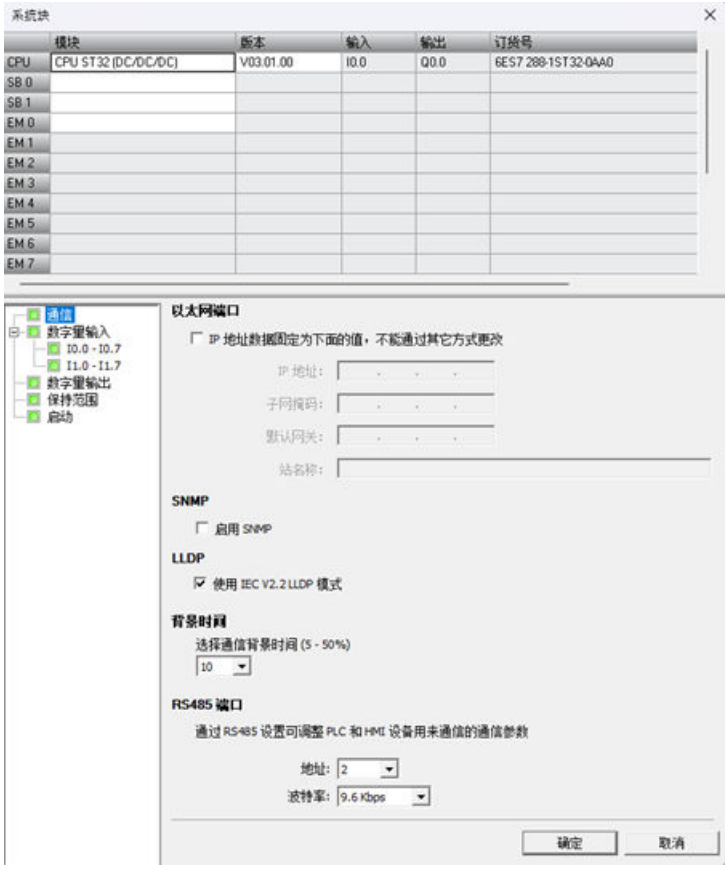
在系统块中进行的 RS485 网络信息组态或更改为项目的一部分，在您将项目下载至 CPU 前不会生效。

要访问此对话框，请执行以下操作之一：

- 在导航栏中单击“系统块”(System Block) 按钮。
- 在项目树中，选择“系统块”(System Block) 节点，然后按下 Enter，或双击“系统块”(System Block) 节点。

输入或更改以下访问信息：

- RS485 端口地址
- RS485 端口波特率



完成 RS485 网络组态后，将项目下载到 CPU。

所有具有有效 RS485 端口地址的 CPU 和设备均在“通信”(Communications) 对话框中显示。

10.5.5 构建网络

10.5.5.1 通用准则

请务必为所有可能遭雷电冲击的线路安装合适的浪涌抑制设备。

避免将低压信号线和通信电缆与交流电线路和高能、快速切换直流电线路敷设在同一接线槽内。 始终成对布线，中性线或公共线与火线或信号线成对。

S7-200 SMART CPU 的通信端口未绝缘。 可考虑使用 RS485 中继器为网络提供绝缘保护。

注意
防止意外电流 互连参考电位不同的设备可能导致意外电流从互连电缆中流过。 这些意外电流可能导致通信错误或设备损坏。 确保要使用通信电缆连接的所有设备都共用一个公共电路参考点或者进行隔离，以防止出现意外电流。

10.5.5.2 构建网络

步骤

要构建网络，进行如下操作：

1. 确定网络的距离、传输率和电缆 (页 561)
2. 在网络中使用中继器 (页 562)
3. 选择网络电缆 (页 563)
4. 连接器引脚分配
5. 偏置和端接网络电缆
6. 使用自由端口模式创建用户定义的协议 (页 566)
7. 对 RS232 设备使用 RS232/PPI 多主站电缆和自由端口模式 (页 568)

10.5.5.3 确定网络的距离、传输率和电缆长度

如下表所示，网段的最大长度由两个因素决定：绝缘（使用 RS485 中继器）和波特率。

如果要连接接地电位不同的设备，则必须进行隔离。当由于距离远而造成接地点被分开时，接地电位可能会不同。即使相隔不远，重型机械的负载电流也可能会导致接地电位不同。

表格 10-5　网络电缆的最大长度

波特率	非隔离 CPU 端口 [1]	带中继器的 CPU 端口
9.6 Kbps 到 187.5 Kbps	50 m	1,000 m
500 Kbps	不受支持	400 m
1 Mbps 到 1.5 Mbps	不受支持	200 m
3 Mbps 到 12 Mbps	不受支持	100 m
[1] 不使用隔离器或中继器时，允许的最大距离为 50 米。该距离为网段中第一个节点到最后一个节点之间的距离。		

10.5.5.4 网络中的中继器

RS485 中继器为网段提供偏置和端接。中继器的用途如下：

- **增加网络的长度**
  向网络添加一个中继器可以使网络延长 50 米。如果将两台中继器连接在一起，中间无其他节点（如下图所示），则可将网络扩展为波特率允许的最大电缆长度。一个网络最多可以串联 9 个中继器，但是网络的总长度不能超过 9600 米。
- **向网络添加设备**
  在 9.6 Kbps 时，每个网段最长为 50 米，最多可以连接 32 台设备。使用中继器可以向网络再添加一个网段（最多可以连接 32 台设备）。
- **电气隔离不同的网段**
  隔离网络可以使接地电位可能不相同的网段相互隔离，从而提高传输质量。

即使没有为网络中的中继器分配网络地址，也会将每个中继器计为网段上的一个节点。以下是配有中继器的网络示例。

10.5.5.5 RS485 电缆规范

S7-200 SMART CPU 网络对双绞线电缆采用 RS485 标准。下表列出了网络电缆的规格。一个网段中最多可以连接 32 台设备。

规范	说明
电缆类型	屏蔽双绞线
回路电阻	≤115Ω/km
有效电容	30 pF/m
额定阻抗	约为 135Ω 至 160Ω （频率 = 3MHz 至 20 MHz）
衰减	0.9 dB/100 m（频率为 200 kHz）
横截面积	0.3 mm^2 到 0.5 mm^2
电缆直径	8 mm +0.5 mm

10.5.5.6 连接器引脚分配

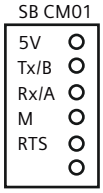
下表列出了为 S7-200 SMART G2 CPU 上的 RS485 通信端口提供物理连接的连接器，并描述了通信端口的引脚分配。

表格 10-6 S7-200 SMART G2 CPU 集成 RS485 端口（端口 0）的引脚分配

连接器	信号	集成 RS485 端口（端口 0）
X20 5V 0V B A	5 V	+5 V 输出，100 Ω 串联电阻
	0 V	逻辑公共端
	B	RS485 信号 B
	A	RS485 信号 A

CM01 信号板与 RS485 兼容。下表列出了为信号板提供物理连接的连接器，并描述引脚分配。

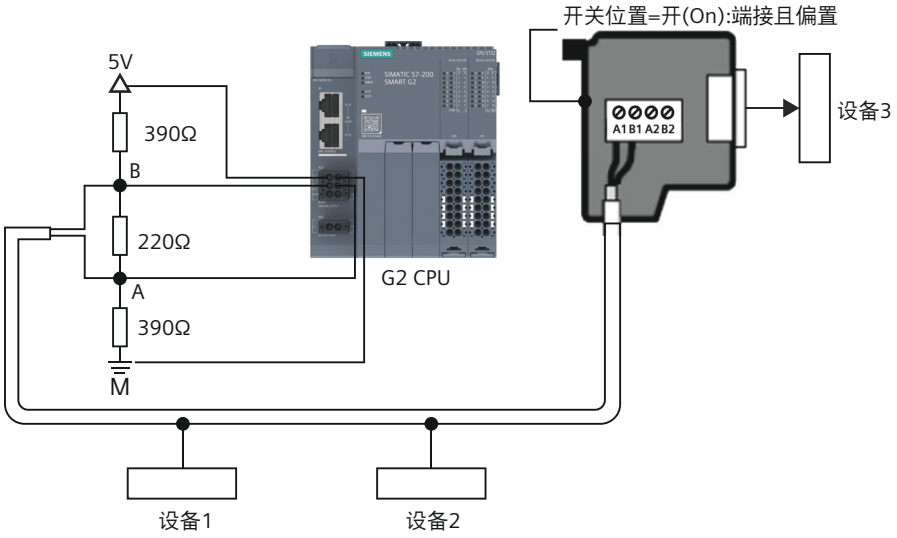
表格 10-7　S7-200 SMART G2 CM01 信号板 (SB) 端口（端口 1）的引脚分配

连接器	信号	**CM01 信号板 (SB) 端口（端口 1）：**
	+5 V DC	+5 V，内部有 100 Ω 串联电阻
	Tx/B	RS232-Tx/RS485-B
	Rx/A	RS232-Rx/RS485-A
	M 接地	逻辑公共端
	RTS (TTL)	请求发送

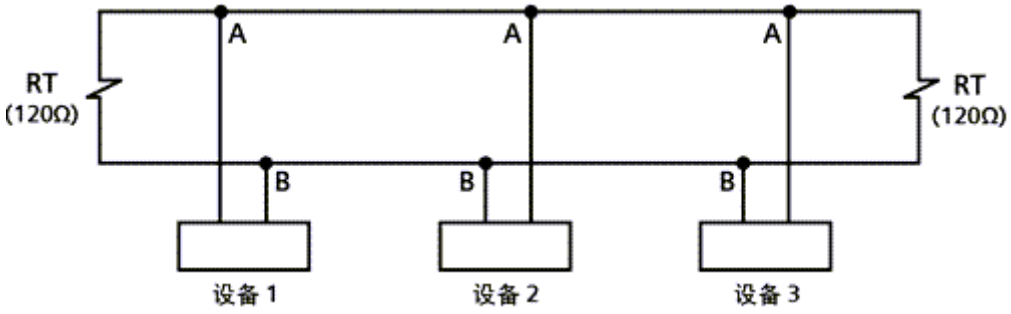
10.5.5.7 偏置和端接网络电缆

Siemens 提供了两种类型的网络连接器，可使用它们轻松地将多台设备连接到一个网络：

- 标准网络连接器



- 为了确保网络信号的完整性和稳定性，当您未使用西门子标准网络连接器时，强烈建议在电缆的两端均安装 120Ω 的终端电阻。



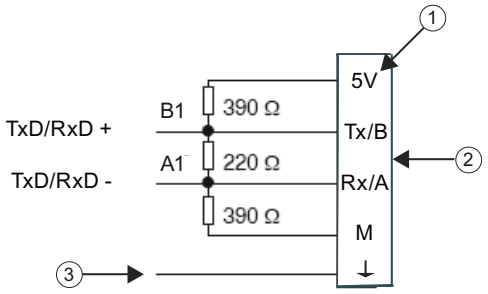
10.5.5.8 偏置和端接 CM01 信号板

可以使用 CM01 信号板轻松地将多台设备连接到网络。

信号板将所有信号从 S7-200 SMART CPU 传递至编程端口。

说明

S7-200 SMART G2 CM01 不需要连接电缆屏蔽。



① 端子名称
② 端子排
③ 电缆屏蔽

10.5.6 自由端口模式

10.5.6.1 使用自由端口模式创建用户定义的协议

简介

自由接口模式允许程序控制 S7-200 SMART CPU 的通信端口。可以在自由端口模式下使用用户定义的通信协议与多种类型的智能设备进行通信。自由端口模式支持 ASCII 和二进制协议。

使用自由端口模式

要启用 Freeport 模式，可使用特殊存储器字节 SMB30（用于集成的 RS485 端口（端口 0）），如果 CPU 型号支持，还可使用 SMB130（用于 CM01 信号板 (SB) 端口（端口 1））、SMB31（用于 CM01 信号板 (SB) 端口（端口 2））。程序通过以下方式控制通信端口的操作：

- **发送指令 (XMT) 和发送中断：**
  借助发送指令，S7-200 SMART CPU 可从 COM 端口发送最多 255 个字符。发送中断会在发送完成时通知 CPU 中的程序。
- **接收字符中断：**
  接收字符中断会通知用户程序已在 COM 端口上接收到字符。随后，程序将根据所执行的协议对该字符进行处理。
- **接收指令 (RCV)：**
  接收指令从 COM 端口接收整条信息，完全接收到该消息后，将为程序生成中断。使用 CPU 的 SM 存储器组态接收指令，根据定义的条件开始和停止接收消息。接收指令允许程序根据特定字符或时间间隔开始或停止接收消息。无需使用繁琐的接收字符中断方法，接收指令便可实现多数协议。

仅当 CPU 处于 RUN 模式时，才会激活自由端口模式。如果将 CPU 设为 STOP 模式，则所有自由端口通信都会中断，而且通信端口会按照 CPU 系统块中组态的设置恢复为 PPI 协议。

说明

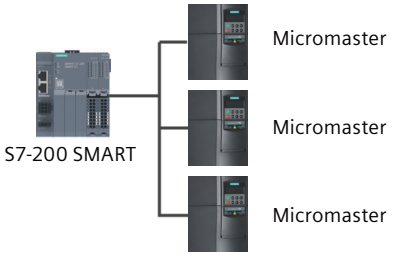
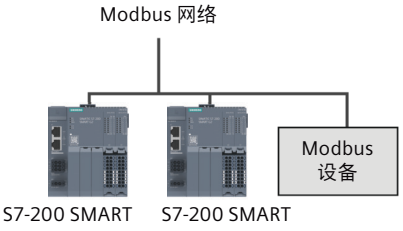
如果将 USB-PPI 电缆连接到 CPU 的 RS485 端口，则会强制 CPU 退出自由端口模式并启用 PPI 模式。这会使 STEP 7-Micro/WIN SMART 恢复 CPU 控制。

如果已将 USB-PPI 电缆连接到 CPU 的 RS485 端口，则 CPU 无法启用自由端口。移除 USB-PPI 电缆后，CPU 将不会自动重启自由端口。

说明

S7-200 SMART G2 CPU 不支持直接连接 USB-PPI 线缆。若需使用此线缆连接 S7-200 SMART G2 CPU，则需要自行制作 RS485 连接线。

表格 10-8　使用自由端口模式

网络组态		说明
通过 RS232 连接使用自由端口		示例：使用一个 S7-200 SMART CPU 与一个带 RS232 端口的电子秤。 • 使用以下方法之一连接两台设备： – RS232/PPI 多主站电缆将电子秤的 RS232 端口连接到 CPU 的 RS485 端口。（将电缆设为 PPI/自由端口模式，开关 5 = 0。） – 使用支持 RS232 和 RS485 的 CM01 信号板 (SB)，您可将 RS232 设备直接连接到 CPU SB RS232，无需 RS232/PPI 电缆。 • CPU 使用自由端口与电子秤进行通信。 • 波特率的范围为 1.200 Kbps 到 115.2 Kbps。 • 用户程序定义协议。
使用 USS 协议		示例：使用一个 S7-200 SMART CPU 与一个 SIMODRIVE MicroMaster 变频器。 • STEP 7-Micro/WIN SMART 提供 USS 库。 • CPU 为主站，变频器为从站。
创建模拟另一网络中的从站设备的用户程序		示例：将 S7-200 SMART CPU 连接到 Modbus 网络。 • CPU 中的用户程序模拟 Modbus 从站。 • STEP 7-Micro/WIN SMART 提供 Modbus 库。

10.5.6.2　对 RS232 设备使用 RS232/PPI 多主站电缆和自由端口模式

用途

可以使用 RS232/PPI 多主站电缆和自由端口通信功能将 S7-200 SMART CPU 连接到很多兼容 RS232 标准的设备。电缆必须设置为 PPI/自由端口模式（开关 5= 0）才能进行自由端口操作。开关 6 用于选择本地模式 (DCE)（开关 6 = 0）或远程模式 (DTE)（开关 6 = 1）。

当数据从 RS232 端口传输到 RS485 端口时，RS232/PPI 多主站电缆处于“发送”模式。当电缆空闲或从 RS485 端口向 RS232 端口传输数据时，电缆处于“接收”模式。一旦电缆检测到 RS232 传输线路上的字符，电缆便会立即从“接收”模式切换为“发送”模式。

CM01 信号板 (SB) 支持 RS232 半双工和 RS485。使用 CM01 信号板，您可将 RS232 设备直接连接到 CPU SB RS232 端口，无需 RS232/PPI 电缆。

波特率和转变时间

RS232/PPI 多主站电缆支持的波特率为 1.2 Kbps 到 115.2 Kbps。可使用 RS232/PPI 多主站电缆外壳上的 DIP 开关将电缆组态为正确的波特率。下表列出了波特率（每秒传输的位数）和对应的开关位置。

表格 10-9　转变时间和设置

波特率	转变时间	设置（1 = 上）
115200	0.15 ms	110
57600	0.3 ms	111
38400	0.5 ms	000
19200	1.0 ms	001
9600	2.0 ms	010
4800	4.0 ms	011
2400	7.0 ms	100
1200	14.0 ms	101

当 RS232 传输线路处于空闲状态的时间达到定义的电缆转变时间时，电缆开关会切换回“接收”模式。选择的电缆波特率决定转变时间，如上表所示。

如果在使用自由端口通信的系统中使用 RS232/PPI 多主站电缆，那么在下列情形下，S7-200 SMART CPU 中的程序必须考虑转变时间：

- CPU 响应 RS232 设备传送的消息。
  CPU 从 RS232 设备接收到请求消息后，CPU 必须延迟一段时间再发送响应消息，延时时间应该大于或者等于电缆的转变时间。
- RS232 设备响应 CPU 传送的消息。
  CPU 从 RS232 设备接收到响应消息后，CPU 必须延迟一段时间再发送下一条请求消息，延时时间应该大于或者等于电缆的转变时间。

在以上两种情况中，延时会使 RS232/PPI 多主站电缆有足够的时间从“发送”模式切换为“接收”模式，从而使数据能够从 RS485 端口传送到 RS232 端口。

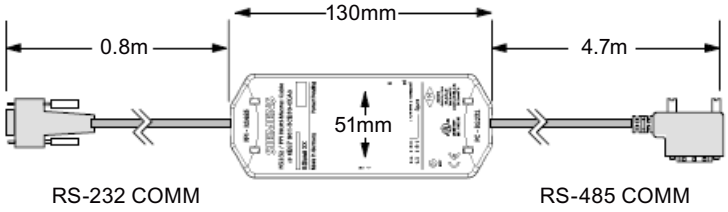
10.5.6.3 RS232/PPI 多主站电缆开关

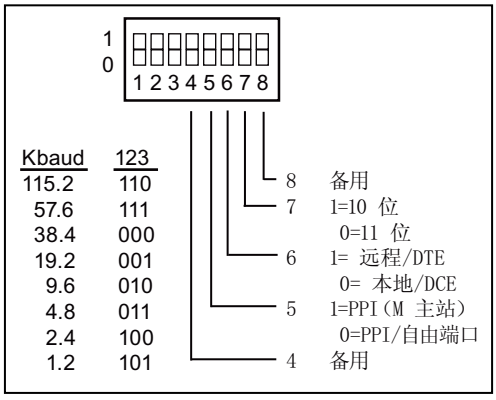
简介

可使用 PC/PPI 电缆将调制解调器的 RS232 通信端口连接到 S7-200 SMART CPU 的 RS485 端口。

CM01 信号板支持 RS232 半双工和 RS485。使用 CM01 信号板，您可将调制解调器直接连接到 RS232 端口，无需 PC/PPI 电缆。

PC/PPI 电缆上的 1、2、3 号开关设置波特率。6 号开关选择数据通信装置 (DCE) 或数据终端装置 (DTE) 模式。5 号开关选择 PC/PPI 电缆 RS232 端口上的 RTS 信号操作。





LED	颜色	说明
Tx	绿色	RS-232 发送指示灯
Rx	绿色	RS-232 接收指示灯
PPI	绿色	RS-485 发送指示灯

调制解调器通常使用 RS232 控制信号（如 RTS、CTS 和 DTR）以允许计算机控制调制解调器。将调制解调器与 PC/PPI 电缆配合使用时，您必须将调制解调器组态为在无这些信号的情况下操作。要确定组态调制解调器所需的命令，请查阅您的特定调制解调器的文档。

开关 7（仅限 V2.x CRs）

使用 S7-200 SMART Classic CPU 上的开关 7 选择 10 位或 11 位调制解调器协议。RS232-PPI 电缆标记可反映出此用途。

在 S7-200 SMART CPU 上将开关 7 用于不同用途。开关 7 选择 RS485 D 连接器上引脚 9 的极性。此开关对 SR 或 ST 型号 CPU 无影响。开关 7 与任何 S7-200 SMART CRs 型号 CPU 合用时，应设置为“1”（标记为“10 位”），以允许自由端口模式。

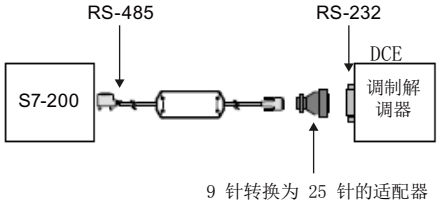
开关 6

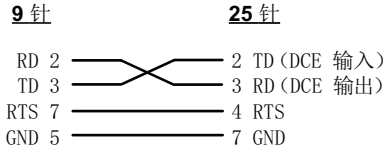
PC/PPI 电缆的 6 号开关可将电缆的 RS232 端口设置为 DCE 或 DTE 模式。如果要将 PC/PPI 电缆与 STEP 7-Micro/WIN SMART 配合使用或 PC/PPI 电缆已连接到计算机，将 PC/PPI 电缆设置为 DCE 模式。如果您将 PC/PPI 电缆与调制解调器（其为 DCE 设备）配合使用，则将 PC/PPI 电缆设为 DTE 模式。这样就无须在 PC/PPI 电缆和调制解调器之间安装一台空调制解调器适配器。根据调制解调器上的连接器，您可能仍需要使用一台 9 针至 25 针适配器。

开关 5

PC/PPI 电缆的 5 号开关选择 RS232 连接器上的 RTS 信号操作。选择“RTS for XMT”会产生以下影响：CPU 在 RS485 端口上发送数据时激活 RTS 信号，CPU 在接收数据时禁用 RTS 信号。选择“RTS 常开”(RTS Always) 会产生以下影响：无论 CPU 是在发送还是接收数据，RTS 信号始终在 PC/PPI 电缆的 RS232 端口上处于活动状态。仅当 PC/PPI 电缆设为 DTE 模式时，5 号开关才影响 RTS 信号。

下图显示公用调制解调器适配器的引脚分配。





下表显示 DTE 模式下 PC/PPI 电缆的 RS485 和 RS232 端口的引脚编号和功能。表 7-10 显示 DCE 模式下 PC/PPI 电缆的 RS485 和 RS232 端口的引脚编号和功能。仅当 PC/PPI 电缆处于 DTE 模式时，才提供 RTS 信号。

RS-485 连接器引脚		**RS-232 DTE 连接器引脚 ****	
引脚	信号说明	引脚	信号说明
1	接地（RS485 逻辑接地）	1	未使用：数据载波检测 (DCD)

RS-485 连接器引脚		RS-232 DTE 连接器引脚 **	
2	24 V 回流（RS485 逻辑接地）	2	接收数据 （RD）（向 PC/PPI 电缆输入）
3	信号 B (RxD/TxD+)	3	传送数据 (TD)（从 PC/PPI 电缆输出）
4	RTS（TTL 级别）	4	未使用：数据终端准备就绪 (DTR)
5	接地（RS485 逻辑接地）	5	接地（RS232 逻辑接地）
6	NC（无连接）	6	未使用：数据集准备就绪 (DSR)
7	24 V 电源	7	请求发送（RTS）（输出 — 可选择开关）
8	信号 A (RxD/TxD-)	8	未使用：允许发送 (CTS)
9	协议选择	9	未使用：振铃指示器 (RI)

**调制解调器可能要求将内孔接头转换为凸形接头，并将 9 针转换为 25 针。

RS-485 连接器引脚		RS-232 DCE 连接器引脚	
引脚	信号说明	引脚	信号说明
1	接地（RS485 逻辑接地）	1	未使用：数据载波检测 (DCD)
2	24 V 回流（RS485 逻辑接地）	2	接收数据 (RD)（向 PC/PPI 电缆输出）
3	信号 B (RxD/TxD+)	3	传送数据 (TD)（从 PC/PPI 电缆输入）
4	RTS（TTL 级别）	4	未使用：数据终端准备就绪 (DTR)
5	接地（RS485 逻辑接地）	5	接地（RS232 逻辑接地）
6	NC（无连接）	6	未使用：数据集准备就绪 (DSR)
7	24 V 电源	7	未使用：请求发送 (RTS)
8	信号 A (RxD/TxD-)	8	未使用：允许发送 (CTS)
9	协议选择	9	未使用：振铃指示器 (RI)

10.6 RS232

RS232 网络是两台设备之间的点对点连接。RS232 允许在较短的距离（最远 50 英尺）内以相对较慢的速度（最大 115.2 Kbps）进行数据传输。

可能的 RS232 连接包括：

- 自由端口
- 调制解调器
- RS232 兼容的设备（如条形码扫描器）

- 具有 RS232 接口的设备（如控制系统）
- RS232 显示器

库 11

11.1 库类型（Siemens 及用户定义）

库类型

Siemens 在 STEP 7-Micro/WIN SMART 的安装程序中提供了两种类型的库：

- Siemens 提供（Modbus RTU (页 580)、Modbus TCP (页 602)、开放式用户通信 (页 622)、PN Read Write Record 库 (页 666)、SINAMICS 库 (页 687)和 USS 协议 (页 670)）
- 用户自定义 (页 747)（从项目 POU 创建或从其他来源获取的库）

说明

必须使用“以管理员身份运行”(Run as administrator) 命令启动 STEP 7-Micro/WIN SMART V3，以创建一个用户自定义的库。

说明

为用户定义库提供的名称不可与 Siemens 提供的库名称相同。

说明

只可从主程序或中断例程中调用库函数，但不可同时从这两个程序中调用。

说明

程序空间因 V 存储器区域的起始地址而异。每个库的程序空间基于起始地址 VB0 计算得出。

Modbus RTU

STEP 7-Micro/WIN SMART 通过包含预组态子程序和中断程序（用于通过 CPU 串行端口进行 Modbus 通信），简化了与 Modbus 设备的通信。您可以利用 Modbus RTU 指令组态 S7-200 SMART，将其用作 Modbus RTU 主站或从站设备。

这些指令位于项目树中“指令”(Instructions) 文件夹的“库”(Libraries) 文件夹中。在将 Modbus RTU 库指令置入程序时，STEP 7-Micro/WIN SMART 会自动将一个或多个关联的子程序和中断程序置入项目。

Modbus TCP

STEP 7-Micro/WIN SMART 通过包含预组态子程序（专用于通过工业以太网进行 Modbus 通信），简化了与 Modbus 设备的通信。通过 Modbus TCP 协议指令，可将 S7-200 SMART 组态为 Modbus TCP 客户端或服务器设备。

这些指令位于项目树中“指令”(Instructions) 文件夹的“库”(Libraries) 文件夹中。在将 Modbus TCP 库指令置入程序时，STEP 7-Micro/WIN SMART 会自动将一个或多个关联子程序置入项目。

Open user communication

开放式用户通信 (OUC) 提供了一种机制，可使您的程序通过以太网发送和接收消息。您可以选择以太网协议作为传输机制：UDP、TCP 或 ISO-on-TCP

这些指令位于项目树中“指令”(Instructions) 文件夹的“库”(Libraries) 文件夹中。在将 OUC 库指令置入程序时，STEP 7-Micro/WIN SMART 会自动将一个或多个关联子程序置入项目。

PN Read Write Record library

STEP 7-Micro/WIN SMART 包含针对 PN Read Write Record 库特别设计的子例程，从而简化从任意 PROFINET 设备读取数据记录或向任意 PROFINET 设备写入数据记录的过程。

SINAMICS library

STEP 7-Micro/WIN SMART 通过包含专用于 SINAMICS 库的预组态子程序，简化了对驱动器的控制。通过 SINAMICS 指令，可控制物理驱动器的位置和速度，并读取或修改驱动器参数。

这些指令位于项目树中“指令”(Instructions) 文件夹的“库”(Libraries) 文件夹中。在将 SINAMICS 库指令置入程序时，STEP 7-Micro/WIN SMART 会自动将一个或多个关联子程序置入项目。

USS protocol

USS 协议库支持 Siemens 驱动器。STEP 7-Micro/WIN SMART USS 指令库包括专用于通过 USS 协议与变频器进行通信的预组态子例程和中断例程，从而使变频器控制更简便。可使用 USS 指令控制物理驱动器和读/写驱动器参数。

这些指令位于项目树中“指令”(Instructions) 文件夹的“库”(Libraries) 文件夹中。向程序中加入 USS 库指令时，STEP 7-Micro/WIN SMART 会自动向项目中添加一个或多个相关联的子例程

警告

安全：保护网络，防止物理访问和对 PLC 数据可能进行的读写操作。

通过 Modbus RTU、开放式用户通信和 USS 协议库指令进行的通信没有安全功能。如果攻击者能利用这些形式的通信以物理方式访问网络，则攻击者有可能读写 PLC 数据。对 PLC 数据未授权的访问可能造成严重人身伤害，甚至死亡。

必须通过限制物理访问来保护这些形式的通信。有关安全信息及建议，请参见以下文档：工业安全操作指南 (https://www.siemens.com/global/en/products/automation/topic-areas/industrial-cybersecurity.html)

11.2 Modbus 通信概述

STEP 7-Micro/WIN SMART V3 和 S7-200 SMART CPU 通过包含用于下列通信类型的预组态子程序和中断例程，简化了与 Modbus 设备的通信：

- 通过 CPU 串行端口进行的 Modbus RTU 通信
- 通过工业以太网进行的 Modbus TCP 通信

Modbus 通信的某些特性是 Modbus RTU 和 Modbus TCP 共有的。

11.2.1 Modbus 寻址

Modbus 地址为五到六位数，包含了数据类型和地址值。

Modbus RTU 主站/Modbus TCP 客户端寻址

Modbus RTU 主站和 Modbus TCP 客户端指令将地址映射到正确的函数，以发送到从设备或客户端设备。Modbus 地址定义如下：

- 00001 至 09999 是离散量输出（线圈）
- 10001 至 19999 是离散量输入（触点）
- 30001 至 39999 是输入寄存器（通常是模拟量输入）
- 40001 至 49999 和 400001 至 465535 是保持寄存器

所有 Modbus 地址均从 1 开始，也就是说第一个数据值从地址 1 开始。有效地址的实际范围取决于从站设备。不同的设备支持不同的数据类型和地址范围。

Modbus RTU 从站/Modbus TCP 服务器寻址

Modbus RTU 从站指令和 Modbus TCP 服务器指令支持以下地址：

- 00001 至 09216 是映射到 Q0.0 至 Q1151.7 的离散量输出。
- 10001 至 19216 是映射到 I0.0 至 I1151.7 的离散量输入。
- 30001 至 30266 是映射到 AIW0 - AIW530 的模拟量输入寄存器。
- 40001 至 49999 和 400001 至 465535 是映射到 V 存储器的保持寄存器。

将 Modbus 地址映射到 CPU 地址

所有 Modbus 地址均从 1 开始。

表格 11-1　将 Modbus 地址映射到 CPU 地址

Modbus 地址	CPU 地址
00001	Q0.0
00002	Q0.1
00003	Q0.2
...	...
01025	Q128.0 [1]
01026	Q128.1[1]
01027	Q128.2[1]
...	...
09215	Q1151.6[1]
09216	Q1151.7[1]
10001	I0.0
10002	I0.1
10003	I0.2
...	...
11025	I128.0[1]
11026	I128.1[1]
11027	I128.2[1]

Modbus 地址		CPU 地址
...		...
19215		I1151.6[1]
19216		I1151.7[1]
30001		AIW0
30002		AIW2
30003		AIW4
...		...
30266		AIW530
40001	400001	Vx（保持寄存器起始地址）
40002	400002	Vx+2 =（保持寄存器起始地址+2）
40003	400003	Vx+4 =（保持寄存器起始地址+4）
...		...
4yyyy	4zzzzz	Vx+2(yyyy-1) 或 Vx+2(zzzzz-1)

说明

[1] CPU V2.4 及更高版本支持更新的存储器地址：Q128.0 - Q1151.7 和 I128.0 - I1151.7。有关详细信息，请参见“CPU 存储器范围和特性 (页 1334)”。

限制从站可访问性的 MBUS 参数

Modbus 从站/协议允许您限制 Modbus 主站可访问的输入、输出、模拟量输入和保持寄存器（V 存储器）的数量。

- **MaxIQ** 分配允许 Modbus 主站访问的离散输入或输出（I 或 Q 地址）的最大数。
- **MaxAI** 指定允许 Modbus 主站访问的输入寄存器（AIW 地址）的最大数目。
- **MaxHold** 指定允许 Modbus 主站访问的保持寄存器（V 存储器字）的最大数目。

有关设置 Modbus RTU 从站存储器限制的更多信息，请参见 MBUS_INIT (页 594) 指令的描述。

有关设置 Modbus TCP 服务器存储器限制的更多信息，请参见 MBUS SERVER (页 609) 指令的描述。

11.2.2 Modbus 读取和写入功能

Modbus RTU 主站指令使用如下所示的 Modbus 功能读取或写入特定的 Modbus 地址。Modbus RTU 从站设备必须支持相应的 Modbus 功能，从而读取或写入特定 Modbus 地址。

表格 11-2 所需的 Modbus 从站功能支持

Modbus 地址	读取或写入	所需的 Modbus 从站功能
00001 - 09999 离散输出	读取	功能 1
	写入	功能 5 适用于单个输出点 功能 15 适用于多个输出点
10001 - 19999 离散输入	读取	功能 2
	写入	不可以
30001 - 39999 输入寄存器	读取	功能 4
	写入	不可以
40001 - 49999 保持寄存器 400001 - 465535	读取	功能 3
	写入	功能 6 适用于单个寄存器 功能 16 适用于多个寄存器

Modbus 消息长度

S7-200 SMART CPU 支持的 Modbus 消息为每条最多 240 个字节（1920 位或 120 个寄存器）的数据。某些从站设备支持的数据数可能少于 240 个字节。

11.3 Modbus RTU 库指令

11.3.1 Modbus RTU 通信概述

11.3.1.1 Modbus RTU 库功能

STEP 7-Micro/WIN SMART V3 包括 Siemens Modbus RTU 库。Modbus RTU 库包括预组态子例程和中断例程，这些例程能够使与 Modbus RTU 主站和从站设备的通信更为简单。

STEP 7-Micro/WIN SMART V3 支持主站和从站设备均通过 RS-485（集成端口 0 和信号板选件端口 1 和端口 2）和 RS-232（仅限信号板选件端口 1 和端口 2）进行 Modbus 通信。

Modbus RTU 主站指令可组态 S7-200 SMART，使其作为 Modbus RTU 主站设备运行并与一个或多个 Modbus RTU 从站设备通信。最多可以组态三个 Modbus RTU 主站。

Modbus RTU 从站指令可用于组态 S7-200 SMART，使其作为 Modbus RTU 从站设备运行并与 Modbus RTU 主站设备进行通信。

在项目树的“指令”(Instruction) 文件夹中打开“库”(Libraries) 文件夹，访问 Modbus 指令。向程序中加入 Modbus 指令时，STEP 7-Micro/WIN SMART V3 会向项目中添加一个或多个相关联的 POU。

说明

只可从主程序或中断例程中调用库函数，但不可同时从这两个程序中调用。

11.3.1.2 使用 Modbus 指令的要求

Modbus RTU 主站协议

Modbus 主站指令使用以下 CPU 资源：

- 执行 MBUS_CTRL/MB_CTRL2/MB_CTRL3 (页 585) 会初始化 Modbus 主站协议，并使分配的 CPU 端口（0、1 或 2）专用于 Modbus 主站通信。
  当您将 CPU 端口用于 Modbus 通信时，无法再将其用于任何其他用途，包括与 HMI 的通信。
- 对于由 MBUS_CTRL/MB_CTRL2/MB_CTRL3 指令分配的端口，其上所有与自由端口通信相关联的 SM 位置都会受到 Modbus 主站指令的影响。
- Modbus 主站指令使用中断执行某些功能。用户程序不得禁用这些中断。
- Modbus 主站指令程序大小
  - 3 个子例程和 1 个中断例程
  - Modbus 主站指令的变量需要 286 个字节的 V 存储器块。必须使用 STEP 7-Micro/WIN SMART V3 中的库存储器命令为该块分配起始地址。该命令位于项目树中“程序块”(Program Block) 节点下的“库”(Library) 节点的快捷存储器中，或在“文件”(File) 菜单功能区的“库”(Libraries) 部分。

  说明

  要将 CPU 通信端口从 Modbus 改回 PPI，以便可与 HMI 设备通信，应将 MBUS_CTRL/MB_CTRL2/MB_CTRL3 指令的模式参数设置为零 (0)。

Modbus RTU 从站协议

Modbus 从站协议指令使用以下 CPU 资源：

- MBUS_INIT 指令 (页 594)会初始化 Modbus 从站协议，并使分配的 CPU 端口（0、1 或 2）专用于 Modbus 从站通信。
  当您将 CPU 端口用于 Modbus 通信时，无法再将其用于任何其他用途，包括与 HMI 的通信。
- Modbus 从站指令会影响所有与由 MBUS_INIT 指令分配的端口上的自由端口通信相关联的 SM 位置。
- Modbus 从站指令程序大小：
  – 3 个子例程和 2 个中断例程。
  – 2113 个字节的程序空间，用于从站指令和支持程序。
  – Modbus 从站指令的变量需要 781 个字节的 V 存储器块。必须使用 STEP 7-Micro/WIN SMART V3 中的库存储器命令为该块分配起始地址。该命令位于项目树中“程序块”(Program Block) 节点下的“库”(Library) 节点的快捷存储器中，或在“文件”(File) 菜单功能区的“库”(Libraries) 部分。

说明

要将 CPU 通信端口从 Modbus 改回 PPI，以便可与 HMI 设备通信，应将 MBUS_INIT 指令的模式参数设置为零 (0)。

11.3.1.3 Modbus 协议的初始化和执行时间

- **Modbus RTU 主站协议：**主站协议在每次扫描时都需要少量时间来执行 MBUS_CTRL、MB_CTRL2 和 MB_CTRL3 指令（如果有）MBUS_CTRL/MB_CTRL2/MB_CTRL3 初始化 Modbus 主站（首次扫描）时该时间约为 0.2 ms，在后续扫描时约为 0.1 ms。
  MBUS_MSG/MB_MSG2/MB_MSG3 指令的执行延长了扫描时间，主要用于计算请求和响应的 Modbus CRC。CRC（循环冗余校验）确保通信消息的完整性。对于请求和响应中的每个字，PLC 扫描时间会延长约 86 微秒。最大请求/响应（读取或写入 120 个字）使扫描时间延长约 10.3 毫秒。读请求主要是在程序从从站接收响应时延长扫描时间，在发送请求时扫描时间延长得较少。写请求主要是在将数据发送到从站时延长扫描时间，在接收响应时扫描时间延长得较少。

- **Modbus RTU 从站协议：**Modbus 通信使用 CRC（循环冗余验校）确保通信消息的完整性。Modbus 从站协议使用预先计算的数值表来减少处理消息所需的时间。初始化该 CRC 表大约需要 11.3 毫秒。MBUS_INIT 指令执行该初始化，通常发生在进入运行模式后的首次扫描期间。如果 MBUS_INIT 指令和任何其他用户初始化操作所需时间超过了 500 毫秒的扫描看门狗时间，则需要复位看门狗定时器。输出模块看门狗定时器通过向模块的输出中执行写入操作来复位。
  MBUS_SLAVE 在对一个请求提供服务时会延长扫描时间。对于请求和响应中的每个字节，计算其 Modbus CRC 会使扫描时间延长约 40 微秒。最大请求/响应（读取或写入 120 个字）使扫描时间延长约 4.8 毫秒。

11.3.2 Modbus RTU 主站

11.3.2.1 使用 Modbus RTU 主站指令

STEP 7-Micro/WIN SMART 和 S7-200 SMART CPU 支持三种 Modbus RTU 主站。对于单个 Modbus RTU 主站，使用指令 MBUS_CTRL (页 585) 和 MBUS_MSG (页 587)。对于第二个 Modbus RTU 主站，使用指令 MB_CTRL2 (页 585) 和 MB_MSG2 (页 587)。对于第三个 Modbus RTU 主站，使用指令 MB_CTRL3 (页 585) 和 MB_MSG3 (页 587)。

如果在项目中使用三个 Modbus 主站，则要确保 MBUS_CTRL、MB_CTRL2 和 MB_CTRL3 使用不同的端口号。

操作步骤

要在 S7-200 SMART 程序中使用 Modbus RTU 主站指令，请执行以下步骤：

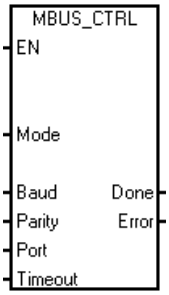
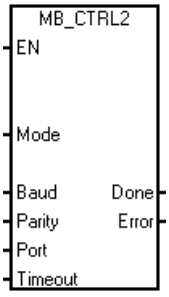
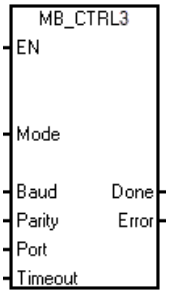
1. 在程序中插入 MBUS_CTRL/MB_CTRL2/MB_CTRL3 指令并在每次扫描时执行。可以使用 MBUS_CTRL/MB_CTRL2 / MB_CTRL3 指令启动或更改 Modbus 通信参数。插入 MBUS_CTRL/MB_CTRL2/MB_CTRL3 指令时，STEP 7-Micro/WIN SMART 会在程序中添加几个受保护的子程序和中断程序。
2. 在“文件”(File) 菜单功能区的“库”(Libraries) 区域中，单击“管理器”(Manager) 按钮 库管理器，指定 Modbus 库所需的 V 存储器的起始地址。或者，也可在项目树中右键单击“程序块”(Program Block) 节点，并从上下文菜单中选择“库存储器”(Library Memory)。
3. 在程序中加入一条或多条 MBUS_MSG/MB_MSG2/MB_MSG3 指令。可以根据需要在程序中添加任意数量的 MBUS_MSG/MB_MSG2/MB_MSG3 指令，但一次只能有一条指令处于激活状态。
4. 用通信电缆连接通过 MBUS_CTRL/MB_CTRL2 / MB_CTRL3 端口参数分配的 S7-200 SMART CPU 端口和 Modbus 从站设备。

注意
防止意外电流 互连参考电位不同的设备可能导致意外电流从互连电缆中流过。这些意外电流可能导致通信错误或设备损坏。 确保所有用通信电缆连接的设备均具有共同的电路参考点或已隔离，从而避免产生意外电流。

11.3.2.2 MBUS_CTRL/MB_CTRL2/MB_CTRL3 指令（初始化主站）

MBUS_CTRL、MB_CTRL2 和 MB_CTRL3 具有相同的作用和参数。MBUS_CTRL 用于单个 Modbus RTU 主站。MB_CTRL2 用于第二个 Modbus RTU 主站。MB_CTRL3 用于第三个 Modbus RTU 主站。相应地，MBUS_MSG 和 MBUS_CTRL 一同用于单个 Modbus RTU 主站。MB_MSG2 和 MB_CTRL2 一同用于第二个 Modbus RTU 主站。MB_MSG3 和 MB_CTRL3 一同用于第三个 Modbus RTU 主站。

表格 11-3　MBUS_CTRL、MB_CTRL2 和 MB_CTRL3 指令

LAD/FBD	STL	说明
  	**CALL MBUS_CTRL, Mode, Baud, Parity, Port, Timeout, Done, Error** **CALL MB_CTRL2, Mode, Baud, Parity, Port, Timeout, Done, Error** **CALL MB_CTRL3, Mode, Baud, Parity, Port, Timeout, Done, Error**	程序调用 MBUS_CTRL/MB_CTRL2/MB_CTRL3 指令来初始化、监视或禁用 Modbus 通信。 在执行 MBUS_MSG/MB_MSG2/MB_MSG3 指令前，程序必须先执行 MBUS_CTRL/MB_CTRL2/MB_CTRL3 且不出现错误。该指令完成后，将“完成”(Done) 位置为 ON，然后再继续执行下一条指令。 EN 输入接通时，在每次扫描时均执行该指令。 **注意：** 当初始化指令成功执行后，相关参数（包括：Baud、Parity 及 Port）的配置状态将被锁存。即，在 PLC 进入运行模式后，上述参数的修改将无法生效。

程序必须在每次扫描时（包括首次扫描）调用 MBUS_CTRL/MB_CTRL2/MB_CTRL3 指令，以便其监视通过 MBUS_MSG/MB_MSG2/MB_MSG3 指令启动的任何待处理消息的进度。除非

程序在每次扫描时都执行 MBUS_CTRL/MB_CTRL2/MB_CTRL3，否则 Modbus 主站协议将不会正确工作。

表格 11-4 MBUS_CTRL /MB_CTRL2 指令的参数

参数	数据类型	操作数
Mode	BOOL	I、Q、M、S、SM、T、C、V、L
Baud	DWORD	VD、ID、QD、MD、SD、SMD、LD、AC、常数、*VD、*AC、*LD
Parity、Port	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*AC、*LD
Timeout	WORD	VW、IW、QW、MW、SW、SMW、LW、AC、常数、*VD、*AC、*LD
Done	BOOL	I、Q、M、S、SM、T、C、V、L
Error	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD

“模式”(Mode) 输入的值用于选择通信协议。输入值为 1 时，将 CPU 端口分配给 Modbus 协议并启用该协议。输入值为 0 时，将 CPU 端口分配给 PPI 系统协议并禁用 Modbus 协议。

参数 **“波特”(Baud)** 将波特率设置为 1200、2400、4800、9600、19200、38400、57600 或 115200。

参数 **“奇偶校验”(Parity)** 应设置为与 Modbus 从站设备的奇偶校验相匹配。所有设置使用一个起始位和一个停止位。允许的值如下：0（无奇偶校验）、1（奇校验）和 2（偶校验）。

参数 **Port** 设置物理通信端口（0 = CPU 中集成的 RS-485，1 = 第一个 CM01 信号板选件上的 RS-485 或 RS-232，2 = 第二个 CM01 信号板选件上的 RS-485 或 RS-232）。

参数 **“超时”(Timeout)** 设为等待从站做出响应的毫秒数。“超时”(Timeout) 值可以设置为 1 ms 到 32767 ms 之间的任何值。典型值是 1000 ms (1 s)。“超时”(Timeout) 参数应设置得足够大，以便从站设备有时间在所选的波特率下做出响应。

“超时”(Timeout) 参数用于确定 Modbus 从站设备是否对请求做出响应。“超时”(Timeout) 值决定着 Modbus 主站设备在发送请求的最后一个字符后等待出现响应的第一个字符的时长。如果在超时时间内至少收到一个响应字符，则 Modbus 主站将接收 Modbus 从站设备的整个响应。

当 MBUS_CTRL/MB_CTRL2/MB_CTRL3 指令完成时，指令将在 Done 输出端返回 TRUE。

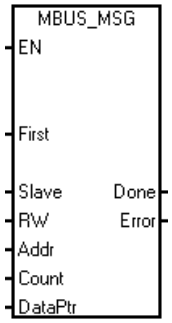
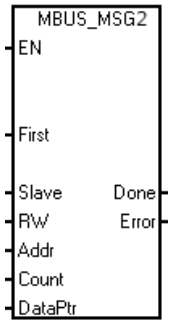
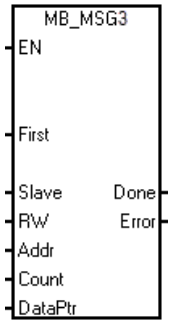
“错误”(Error) 输出包含指令执行的结果。

另请参见 Modbus RTU 主站执行错误代码 (页 590)

11.3.2.3 MBUS_MSG/MB_MSG2/MB_MSG3 指令

MBUS_MSG、MB_MSG2 和 MB_MSG3 具有相同的作用和参数。MBUS_MSG 用于单个 Modbus RTU 主站。MB_MSG2 用于第二个 Modbus RTU 主站。MB_MSG3 用于第三个 Modbus RTU 主站。

表格 11-5 MBUS_MSG/MB_MSG2/MB_MSG3 指令

LAD/FBD	STL	描述
  	**CALL MBUS_MSG, First, Slave, RW, Addr, Count, DataPtr, Done, Error** **CALL MB_MSG2, First, Slave, RW, Addr, Count, DataPtr, Done, Error** **CALL MB_MSG3, First, Slave, RW, Addr, Count, DataPtr, Done, Error**	程序调用 MBUS_MSG/MB_MSG2/MB_MSG3 指令，启动对 Modbus 从站的请求并处理响应。

EN 输入和 First 输入同时接通时，MBUS_MSG/MB_MSG2/MB_MSG3 指令会向 Modbus 从站发起主站请求。发送请求、等待响应和处理响应通常需要多个 PLC 扫描时间。EN 输入必须接通才能启用发送请求，并且必须保持接通状态，直到指令为 Done 位返回接通。

对于 MBUS_MSG、MB_MSG2 或 MB_MSG3 指令，每次都只能有一个指令处于激活状态。如果程序启用多个 MBUS_MSG 指令、多个 MB_MSG2 指令或多个 MB_MSG3 指令，则 CPU 将

处理第一个 MBUS_MSG 指令、第一个 MB_MSG2 指令或第一个 MB_MSG3 指令，所有后续 MBUS_MSG、MB_MSG2 或 MB_MSG3 指令将中止并生成错误代码 6。

表格 11-6 MBUS_MSG/MB_MSG2/MB_MSG3 指令的参数

参数	数据类型	操作数
First	BOOL	I、Q、M、S、SM、T、C、V、L（受上升沿检测元素控制的能流）
Slave	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*AC、*LD
RW	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*AC、*LD
Addr	DWORD	VD、ID、QD、MD、SD、SMD、LD、AC、常数、*VD、*AC、*LD
Count	INT	VW、IW、QW、MW、SW、SMW、LW、AC、常数、*VD、*AC、*LD
DataPtr	DWORD	&VB
Done	BOOL	I、Q、M、S、SM、T、C、V、L
Error	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD

有新请求要发送时，将参数 **First** 设置为接通，并仅保持一个扫描周期。First 输入以脉冲方式通过边沿检测元素（例如，上升沿），这将导致程序发送请求一次。有关详细信息，请参见示例程序 (页 598)。

参数 **“从站”(Slave)** 是 Modbus 从站设备的地址。允许范围为 0 至 247。地址 0 是广播地址。仅将地址 0 用于写入请求。系统不会响应对地址 0 的广播请求。并非所有从站设备都支持广播地址。S7-200 SMART Modbus 从站库不支持广播地址。

使用参数 **RW** 指示是读取还是写入该消息。0（读取）和 1（写入）。

离散量输出（线圈）和保持寄存器支持读请求和写请求。离散量输入（触点）和输入寄存器仅支持读请求。

参数**地址 (Addr)** 是起始 Modbus 地址。S7-200 SMART 支持以下地址范围：

- 对于离散量输出（线圈），为 00001 至 09999
- 对于离散量输入（触点），为 10001 至 19999
- 对于输入寄存器，为 30001 至 39999
- 对于保持寄存器，为 40001 至 49999 和 400001 至 465535

Modbus 从站设备支持的地址决定了 Addr 的实际取值范围。

参数 **“计数”(Count)** 用于分配要在该请求中读取或写入的数据元素数。对于位数据类型，“Count”是位数，对于字数据类型，则表示字数。

- 对于地址 0xxxx，“计数”(Count) 是要读取或写入的位数
- 对于地址 1xxxx，“计数”(Count) 是要读取的位数

- 对于地址 3xxxx，“计数”(Count) 是要读取的输入寄存器字数
- 对于地址 4xxxx 或 4yyyyy，“计数”(Count) 是要读取或写入的保持寄存器字数

MBUS_MSG/MB_MSG2/MB_MSG3 指令最多读取或写入 120 个字或 1920 个位（240 个字节的数据）。Count 的实际限值取决于 Modbus 从站设备的限制。

参数 **DataPtr** 是间接地址指针，指向 CPU 中与读/写请求相关的数据的 V 存储器。对于读请求，将 DataPtr 设置为用于存储从 Modbus 从站读取的数据的第一个 CPU 存储单元。对于写请求，将 DataPtr 设置为要发送到 Modbus 从站的数据的第一个 CPU 存储单元。

程序将 DataPtr 值以间接地址指针的形式传递到 MBUS_MSG/MB_MSG2/MB_MSG3。例如，如果要写入到 Modbus 从站设备的数据始于 CPU 的地址 VW200，则 DataPtr 的值将为 &VB200（地址 VB200）。指针必须始终是 VB 类型，即使它们指向字数据。

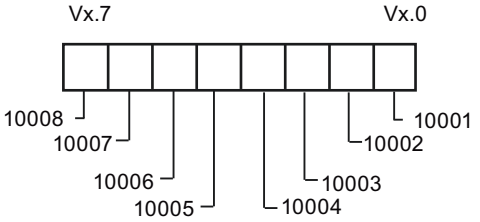
存储器布局

保持寄存器（地址 4xxxx 或 4yyyyy）和输入寄存器（地址 3xxxx）是字值（2 个字节或 16 个位）。CPU 字的格式与 Modbus 寄存器相同。编号较小的 V 存储器地址是寄存器的最高有效字节。编号较大的 V 存储器地址是寄存器的最低有效字节。下表显示了 CPU 字节和字寻址如何与 Modbus 寄存器格式相对应。

表格 11-7　Modbus 保持寄存器

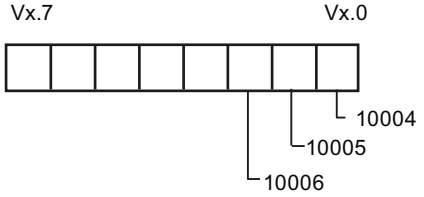
CPU 存储器字节地址		**CPU 存储器字地址**		**Modbus 保持寄存器地址**	
地址	**十六进制数据**	**地址**	**十六进制数据**	**地址**	**十六进制数据**
VB200	12	VW200	12 34	40001	12 34
VB201	34				
VB202	56	VW202	56 78	40002	56 78
VB203	78				
VB204	9A	VW204	9A BC	40003	9A BC
VB205	BC				

CPU 以压缩字节形式读写位数据（地址 0xxxx 和 1xxxx）区域；也就是说，每个字节由 8 位数据构成。第一个数据字节的最低有效位是寻址的位号（参数“地址”(Addr)）。如果打算仅写入单个位，则必须将该位设置为 DataPtr 指向的字节的最低有效位 (Vx.0)。



压缩字节的格式（离散量输入地址）

对于不是从字节边界开始的位数据地址，必须将与起始地址对应的位设置为字节的最低有效位。请参见从 Modbus 地址 10004 开始的 3 个位的压缩字节格式示例。



压缩字节的格式（从地址 10004 开始的离散量输入）

向离散量输出数据类型（线圈）执行写操作时，必须将这些位置于压缩字节内的正确位位置，然后通过 DataPtr 将数据传递到 MBUS_MSG/MB_MSG2/MB_MSG3 指令。

输出

在程序已发送请求之后、程序接收响应之前，Done 输出为 FALSE。响应完成或 MBUS_MSG / MB_MSG2 / MB_MSG 指令因错误中止时，Done 输出为 TRUE。

仅当 Done 输出为 TRUE 时，Error 输出 (页 590)才有效。

11.3.2.4 Modbus RTU 主站执行错误代码

编号高的错误代码（从 101 开始）是 Modbus 从站设备返回的错误。这些错误表明从站不支持所请求的功能，或者 Modbus 从站设备支持不所请求的地址（即，数据类型或地址范围）。

编号小的错误代码（1 到 12）是由 MBUS_MSG 指令检测到的错误。这些错误代码通常表明 MBUS_MSG 指令的输入参数有问题，或接收从站响应时出现问题。奇偶校验和 CRC 错误表明有响应但未正确接收数据。这通常是电气故障（例如连接有问题或电气噪声）引起的。

MBUS_CTRL 错误代码	描述
0	无错误
1	奇偶校验类型无效

MBUS_CTRL 错误代码	描述
2	波特率无效
3	超时无效
4	模式无效
9	端口号无效
10	信号板端口 1 缺失或未组态
11	信号板端口 2 缺失或未组态

MBUS_MSG 错误代码	描述
0	无错误
1	响应存在奇偶校验错误：仅当使用偶校验或奇校验时，才会出现该错误。传输受到干扰，并且可能收到不正确的数据。该错误通常是电气故障（例如，接线错误或影响通信的电气噪声）引起的。
2	未使用
3	接收超时：在超时时间内从站没有做出响应。可能原因为：与从站设备的电气连接存在问题、主站和从站的波特率/奇偶校验的设置不同、从站地址错误。
4	请求参数出错：一个或多个输入参数（“从站”(Slave)、“读写”(RW)、“地址”(Addr) 或 “计数”(Count)）被设置为非法值。有关输入参数的允许值的信息，请参见本文档。
5	未启用 Modbus 主站：每次扫描时，在调用 MBUS_MSG 之前调用 MBUS_CTRL。
6	Modbus 正忙于处理另一请求：某一时间只能有一条 MBUS_MSG 指令处于激活状态。
7	响应出错：收到的响应与请求不符。这意味着从站设备有问题或错误的从站设备对请求做出了应答。
8	响应存在 CRC 错误：传输受到干扰，并且可能收到不正确的数据。该错误通常是电气故障（例如，接线错误或影响通信的电气噪声）引起的。
11	端口号无效
12	信号板端口 1 缺失或未组态
13	信号板端口 2 缺失或未组态

MBUS_MSG 错误代码	描述
101	从站不支持该地址的请求功能：请参见“使用 Modbus 主站指令”帮助主题中的所需 Modbus 从站功能支持表。
102	从站不支持数据地址：“地址”(Addr) 加上“计数”(Count) 的请求地址范围超出从站允许的地址范围。
103	从站不支持数据类型：从站设备不支持“地址”(Addr) 类型。
104	从站设备故障
105	从站接受消息，但未按时做出响应：MBUS_MSG 发生错误，用户程序应在稍后重新发送请求。
106	从站繁忙，拒绝了消息：可以再次尝试相同的请求以获得响应。
107	从站因未知原因拒绝了消息。
108	从站存储器奇偶校验错误：从站设备有故障。

11.3.3 Modbus RTU 从站

11.3.3.1 使用 Modbus RTU 从站指令

步骤

要在 S7-200 SMART 程序中使用 Modbus 从站指令，请执行以下步骤：

1. 在程序中插入 MBUS_INIT 指令，并仅执行 MBUS_INIT 指令一个扫描周期。可以使用 MBUS_INIT 指令初始化或更改通信参数。插入 MBUS_INIT 指令时，会在程序中自动添加若干隐藏的子例程和中断例程。
2. 在“文件”(File) 菜单功能区的“库”(Libraries) 区域中，单击“管理器”(Manager) 按钮 库管理器，指定 Modbus 库所需的 V 存储器的起始地址。或者，也可在项目树中右键单击“程序块”(Program Block) 节点，并从上下文菜单中选择“库存储器”(Library Memory)。除了这个 V 存储器块之外，还可以使用 MBUS_INIT 的 HoldStart 和 MaxHold 参数定义另一个存储器块。注意，V 存储器中的程序分配不要重叠。如果存储区重叠，则 MBUS_INIT 指令将返回错误。

3. 在程序中仅添加一条 MBUS_SLAVE 指令。每次扫描时均应调用该指令，以处理收到的所有请求。
4. 用通信电缆连接通过 MBUS_INIT 端口参数分配的 S7-200 SMART CPU 端口和 Modbus 主站设备。

注意
防止意外电流 互连参考电位不同的设备可能导致意外电流从互连电缆中流过。这些意外电流可能导致通信错误或设备损坏。 确保所有用通信电缆连接的设备均具有共同的电路参考点或已隔离，以避免产生意外电流。

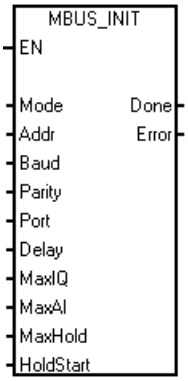
累加器（AC0、AC1、AC2、AC3）由 Modbus 从站指令使用，并显示在“交叉引用”列表中。在执行 Modbus 从站指令前，系统会先保存 Modbus 从站指令累加器中的值，在 Modbus 从站指令完成前恢复到累加器中，从而确保在执行 Modbus 从站指令时保留累加器中的所有用户数据。

Modbus 从站指令支持 Modbus RTU 协议。这些指令利用 S7-200 SMART CPU 的自由端口功能支持最常用的 Modbus 功能。支持以下 Modbus 功能：

功能	说明
1	读取单个/多个线圈（离散量输出）状态。功能 1 返回任何数量输出点 (Q) 的开/关状态。
2	读取单个/多个触点（离散量输入）状态。功能 2 返回任何数量输入点 (I) 的开/关状态。
3	读取单个/多个保持寄存器。功能 3 返回 V 存储器的内容。保持寄存器在 Modbus 中是字值，允许您在一次请求中读取多达 120 个字。
4	读取单个/多个输入寄存器。功能 4 返回模拟量输入值。
5	写入单个线圈（离散量输出）。功能 5 将离散量输出点设置为指定值。系统不强制该输出点，程序可以覆盖 Modbus 请求写入的值。
6	写入单个保持寄存器。功能 6 将单个保持寄存器值写入 S7-200 SMART 的 V 存储器中。
15	写入多个线圈（离散量输出）。功能 15 将离散量输出值写入 S7-200 SMART 的 Q 映象寄存器。起始输出点必须始于字节边界（例如，Q0.0 或 Q2.0），写入的输出数必须是八的倍数。这是对 Modbus 从站协议指令的一个限制。系统不强制这些输出点，程序可以覆盖 Modbus 请求写入的值。
16	写入多个保持寄存器。功能 16 将多个保持寄存器写入 S7-200 SMART 的 V 存储器。在一个请求中最多可写入 120 个字。

11.3.3.2 MBUS_INIT 指令（初始化从站）

表格 11-8 MBUS_INIT 指令

LAD/FBD	STL	说明
	**CALL MBUS_INIT, Mode, Addr, Baud, Parity, Port, Delay, MaxIQ, MaxAI, MaxHold, HoldStart, Done, Error**	MBUS_INIT 指令用于启用，初始化或禁用 Modbus 通信。在使用 MBUS_SLAVE 指令之前，必须先无错误地执行 MBUS_INIT。该指令完成后，立即置位“完成”(Done) 位，然后继续执行下一条指令。 EN 输入接通时，会在每次扫描时执行该指令。

每次通信状态改变时程序必须执行 MBUS_INIT 指令一次。因此，EN 输入以脉冲方式通过边沿检测元素，或者仅在首次扫描时执行 MBUS_INIT。

表格 11-9 MBUS_INIT 参数

输入/输出	数据类型	操作数
Mode、Addr、Parity、Port	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*AC、*LD
Baud、HoldStart	DWORD	VD、ID、QD、MD、SD、SMD、LD、AC、常数、*VD、*AC、*LD
Delay、MaxIQ、MaxAI、MaxHold	WORD	VW、IW、QW、MW、SW、SMW、LW、AC、常数、*VD、*AC、*LD
Done	BOOL	I、Q、M、S、SM、T、C、V、L
Error	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD

“模式”(Mode) 输入的值用于选择通信协议：输入值为 1 时，分配 Modbus 协议并启用该协议；输入值为 0 时，分配 PPI 协议并禁用 Modbus 协议。

参数 **“地址”(Addr)** 将地址设置为 1 至 247 之间（包括边界）的值。

参数 **“波特”(Baud)** 将波特率设置为 1200、2400、4800、9600、19200、38400、57600 或 115200。

参数 **“奇偶校验”(Parity)** 应设置为与 Modbus 主站的奇偶验校相匹配。所有设置使用一个停止位。接受的值如下：0（无奇偶校验）、1（奇校验）和 2（偶校验）。

参数 **Port** 设置物理通信端口（0 = CPU 中集成的 RS-485，1 = 第一个信号板选件上的 RS-485 或 RS-232，2 = 第二个信号板选件上的 RS-485 或 RS-232）。

参数 **“延时”(Delay)** 通过使标准 Modbus 信息超时时间增加分配的毫秒数来延迟标准 Modbus 信息结束超时条件。在有线网络上运行时，该参数的典型值应为 0。如果使用具有纠错功能的调制解调器，则将延时设置为 50 至 100 ms 之间的值。如果使用扩频无线通信，则将延时设置为 10 至 100 ms 之间的值。“延时”(Delay) 值可以是 0 至 32767 ms。

参数 **MaxIQ** 用于设置 Modbus 地址 0xxxx 和 1xxxx 可用的 I 和 Q 点数，取值范围是 0 至 9216。值为 0 时，将禁用所有对输入和输出的读写操作。建议将 MaxIQ 值设置为 9216。

说明

如果想要读取 I 1151.7 之后的地址，或者读写 Q 1151.7 之后的地址，您需要先把这部分地址映射到 V 区， 通过保持寄存器进行读写。

参数 **MaxAI** 用于设置 Modbus 地址 3xxxx 可用的字输入 (AI) 寄存器数，取值范围是 0 至 266。值为 0 时，将禁止读取模拟量输入。建议将 MaxAI 值设置为 266。

参数 **MaxHold** 用于设置 Modbus 地址 4xxxx 或 4yyyyy 可访问的 V 存储器中的字保持寄存器数。建议的 MaxHold 值为 32767。例如，如果要允许 Modbus 主站访问 2000 个字节的 V 存储器，请将 MaxHold 的值设置为 1000 个字（保持寄存器）。

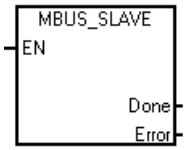
参数 **HoldStart** 是 V 存储器中保持寄存器的起始地址。该值通常设置为 VB0，因此参数 HoldStart 设置为 &VB0（地址 VB0）。也可将其他 V 存储器地址指定为保持寄存器的起始地址，以便在项目中的其他位置使用 VB0。Modbus 主站可访问起始地址为 HoldStart，字数为 MaxHold 的 V 存储器。

MBUS_INIT 指令完成时，“完成”(Done) 输出接通。

Error 输出 (页 597)字节包含指令的执行结果。仅当“完成”(Done) 接通时，该输出才有效。如果“完成”(Done) 关闭，则错误参数不会改变。

11.3.3.3 MBUS_SLAVE 指令

表格 11-10 MBUS_SLAVE 指令

LAD/FBD	STL	说明
	**CALL MBUS_SLAVE, Done, Error**	MBUS_SLAVE 指令用于处理来自 Modbus 主站的请求，并且必须在每次扫描时执行，以便检查和响应 Modbus 请求。 EN 输入接通时，会在每次扫描时执行该指令。 MBUS_SLAVE 指令没有输入参数。

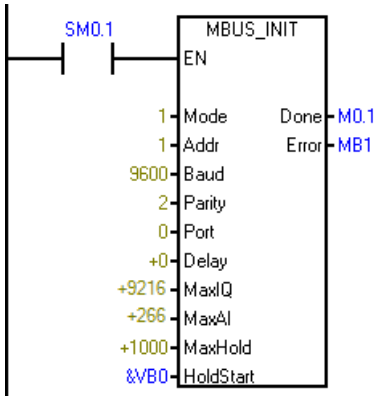
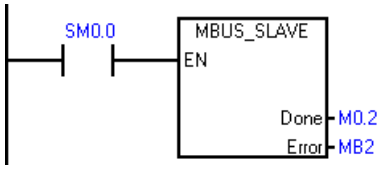
表格 11-11 MBUS_SLAVE 指令的参数

参数	数据类型	操作数
Done	BOOL	I、Q、M、S、SM、T、C、V、L
Error	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD

当 MBUS_SLAVE 指令响应 Modbus 请求时，“完成”(Done) 输出接通。如果未处理任何请求，“完成”(Done) 输出关闭。

Error 输出 (页 597)包含指令的执行结果。仅当 “完成”(Done) 接通时，该输出才有效。如果 “完成”(Done) 关闭，则错误参数不会改变。

表格 11-12　作为 Modbus 从站运行的 S7-200 SMART CPU 的示例程序

LAD		STL
	首次扫描时初始化 Modbus 从站协议。将从站地址设置为 1，将端口 0 设置 9600 波特且进行偶校验，允许访问所有 I、Q 和 AI 值，允许访问从 VB0 起的 1000 个保存寄存器（2000 个字节）。	**Network 1** **LD SM0.1** **CALL MBUS_INIT, 1, 1, 9600, 2, 0, 128, 32, 1000, &VB0, M0.1, MB1**
	每次扫描时执行 Modbus 从站协议。	**Network 2** **LD SM0.0** **CALL MBUS_SLAVE, M0.2, MB2**

11.3.3.4　Modbus RTU 从站执行错误代码

错误代码	描述
0	无错误
1	存储器范围错误
2	波特率或奇偶校验非法
3	从站地址非法
4	Modbus 参数值非法
5	保持寄存器与 Modbus 从站符号重叠
6	收到奇偶校验错误
7	收到 CRC 错误
8	功能请求非法/功能不受支持
9	请求中的存储器地址非法
10	从站功能未启用

错误代码	描述
11	端口号无效
12	信号板端口 1 缺失或未组态
13	信号板端口 2 缺失或未组态

11.3.4 Modbus RTU 主站示例程序

该示例程序显示了每当输入 I0.0 接通时，如何使用 Modbus 主站指令对 Modbus 从站的四个保持寄存器执行读写操作。

CPU 会将从 VW100 开始的四个字写入 Modbus 从站从地址 40001 开始的保持寄存器。

CPU 随后会读取 Modbus 从站从 40010 到 40013 的四个保持寄存器，并将数据存入 CPU 中从 VW200 开始的 V 存储器中。

本示例使用单个主站及 MBUS_CTRL 和 MBUS_MSG 指令。同一理念对使用第二个主站及 MB_CTRL2 和 MB_MSG2 指令或者使用第三个主站及 MB_CTRL3 和 MB_MSG3 指令的示例同样适用。

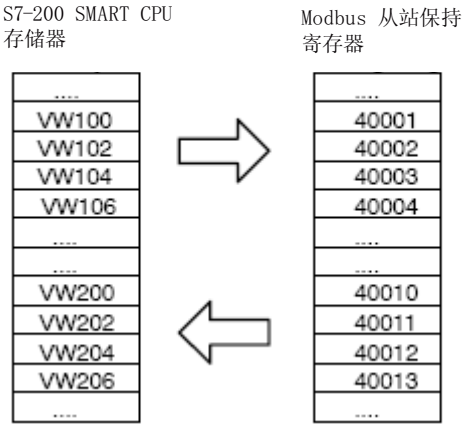
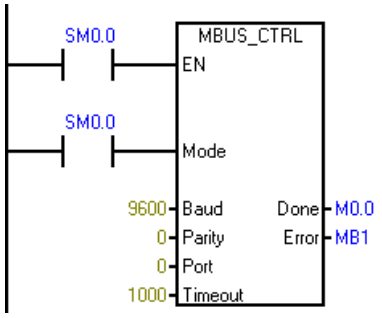
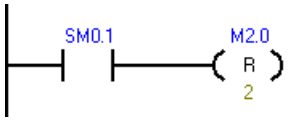
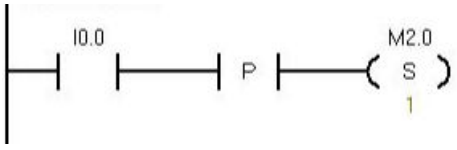
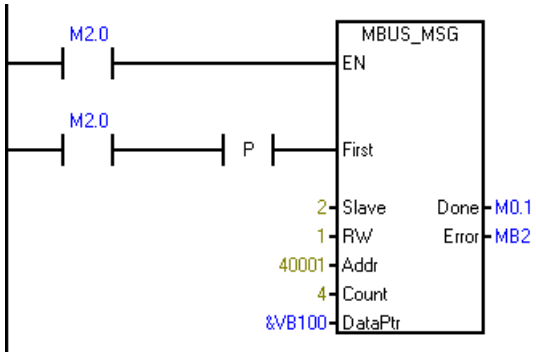
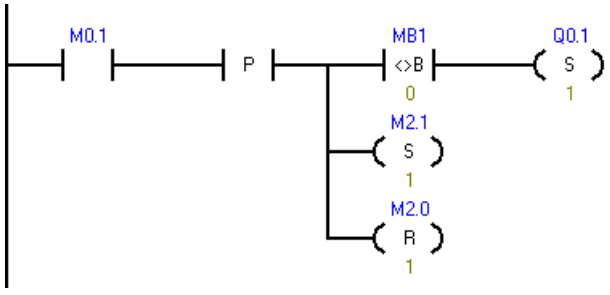
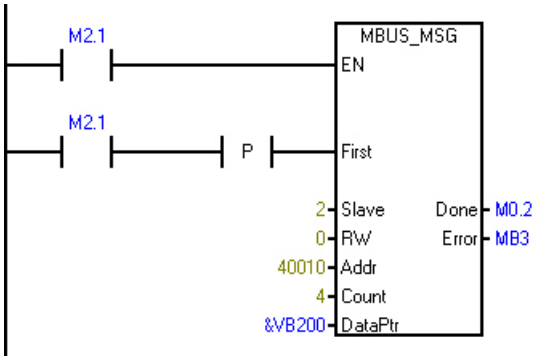
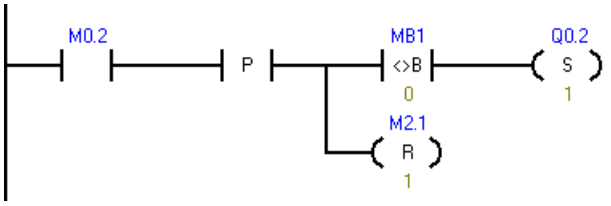


图 11-1　程序数据传送示例

如果 MBUS_MSG 指令返回错误，则以下程序会接通输出 Q0.1 和 Q0.2。

表格 11-13　Modbus 主站示例程序

LAD	描述
	程序段 1 通过在每次扫描时调用 MBUS_CTRL 来初始化和监视 Modbus 主站。Modbus 主站设为 9.6 Kbps，无奇偶校验。从站设备允许在 1000 毫秒（1 秒）内进行响应。
	程序段 2 第一次扫描时，复位用于两条 MBUS_MSG 指令的启用标记（M2.0 和 M2.1）。
	程序段 3 当 I0.0 从关闭变为接通时，设置第一条 MBUS_MSG 指令的启用标志 (M2.0)。
	程序段 4 当第一个启用标志 (M2.0) 接通时，调用 MBUS_MSG 指令。只需为启用该指令的第一次扫描设置 First 参数。 该指令会对从站 2 的 4 个保持寄存器执行写入 (RW = 1) 操作。从 CPU 中的 VB100-VB107（4 个字）获取写数据，然后写入到 Modbus 从站中的地址 40001 - 40004。
	程序段 5 第一条 MBUS_MSG 指令完成后（“完成”(Done) 位从 0 变为 1），会清除第一条 MBUS_MSG 指令的启用标志，然后设置第二条 MBUS_MSG 指令的启用标志。 如果错误 (MB1) 不为零，则置位 Q0.1 显示错误。

LAD	描述
	程序段 6 第二个启用标志 (M2.1) 接通时，调用第二条 MBUS_MSG 指令。只需为启用该指令的第一次扫描设置 First 参数。 该指令会对从站 2 的 4 个保持寄存器执行读取 (RW = 0) 操作。数据从 Modbus 从站中的地址 40010 - 40013 读取，并复制到 CPU 中的 VB200 - VB207（4 个字）。
	程序段 7 第二条 MBUS_MSG 指令完成（“完成”(Done) 位从 0 变为 1）后，清除第二条 MBUS_MSG 指令的启用标志。 如果错误 (MB1) 不为零，则置位 Q0.2 显示错误。

11.3.5 Modbus RTU 高级用户信息

概述

本主题包含可供 Modbus RTU 主站库的高级用户使用的信息。大多数用户不需要此信息，且不需要修改 Modbus RTU 主站库的默认操作。

重试

如果检测到下列任一错误，Modbus 主站指令会自动向从站设备重新发送请求：

- 在响应超时时间（MBUS_CTRL/MB_CTRL2/MB_CTRL3 指令中的 Timeout 参数）内没有响应（错误代码 3）。
- 响应字符之间的时间超出允许值（错误代码 3）。
- 来自从站的响应中存在奇偶校验错误（错误代码 1）。
- 来自从站的响应中存在 CRC 错误（错误代码 8）。
- 返回的功能与请求不符（错误代码 7）。

Modbus 主站在设置 Done 和 Error 输出参数之前重新发送额外两次请求。

程序执行 MBUS_CTRL/MB_CTRL2/MB_CTRL3 之后，可在 Modbus 主站变量表中找到变量 mModbusRetries 并更改值，以此更改重试次数。mModbusRetries 值为 BYTE 类型，范围为 0 到 255 次。

字符间超时

如果响应中各字符之间的时间超出了分配的时间限制，则 Modbus 主站执行中止来自从站设备的响应。默认时间设为 100 毫秒，允许 Modbus 主站指令通过有线或电话调制解调器用于大部分从站设备。如果 CPU 检测到此错误，MBUS CTRL/MB_CTRL2/MB_CTRL3 指令会在 Error 参数中返回错误代码 3。

通信时，字符之间可能需要较长的时间，这可能是传输介质（例如电话调制解调器）的原因，也可能是因为从站设备本身需要较长的时间。执行 MBUS_CTRL/MB_CTRL2/MB_CTRL3 之后，可在 Modbus 主站变量表中找到变量 *mModbusCharTimeout*，然后更改该值，以延长该超时时间。mModbusCharTimeout 值是 INT 类型，范围为 1 到 30000 毫秒。

单个位与多个位/字写入功能

一些 Modbus 从站设备不支持试用 Modbus 功能写入单个离散输出位（Modbus 功能 5）或写入单个保持寄存器（Modbus 功能 6）。相反，这些设备只支持多位写入（Modbus 功能 15）或多寄存器写入（Modbus 功能 16）。如果从站设备不支持单个位/字 Modbus 功能，则 MBUS_MSG/MB_MSG2/MB_MSG3 指令返回错误代码 101。

Modbus 主站协议允许用户强制 MBUS_MSG/MB_MSG2/MB_MSG3 指令使用多个位/字 Modbus 功能，而不使用单个位/字 Modbus 功能。程序执行 MBUS_CTRL/MB_CTRL2/MB_CTRL3 之后，可在 Modbus 主站变量表中找到变量 *mModbusForceMulti*，然后更改该值，以此强制执行多个位/字指令。将 mModbusForceMulti 设为 TRUE，以便在写入单个位或寄存器时强制使用多个位/字功能。

累加器用法

Modbus 主站指令使用累加器（AC0、AC1、AC2、AC3），它们显示在“交叉引用”列表中。Modbus 主站指令用于保存和恢复累加器中的值。执行指令时，所有 CPU 都会留存累加器中的所有用户数据。

保持寄存器地址大于 49999

Modbus 保持寄存器地址在 40001 到 49999 这一范围内。该范围足以满足大多数应用的要求，但有些 Modbus 从站设备将数据映射到地址范围更大的保持寄存器中。

MBUS_MSG/MB_MSG2/MB_MSG3 指令允许参数 **Addr** 采用其他范围，用于支持保持寄存器的扩展地址范围（地址 400001 至 465536）。

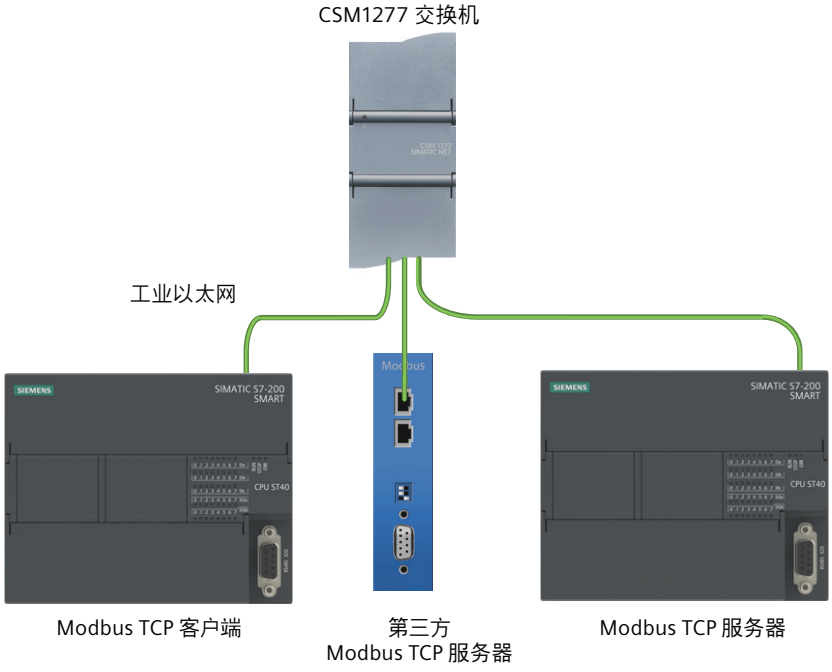
例如：要访问保持寄存器 16768，MBUS_MSG/MB_MSG2/MB_MSG3 的 **Addr** 参数应设为 416768。

扩展寻址允许访问 Modbus 协议支持的全部 65536 个可能地址。该扩展寻址仅适用于保持寄存器。

11.4 Modbus TCP 库

11.4.1 Modbus TCP 库功能

Modbus TCP 是通过工业以太网 TCP/IP 网络传输的 Modbus 通信。S7-200 SMART 采用客户端-服务器方法，Modbus 客户端设备通过该方法发起与 Modbus 服务器设备的 TCP/IP 连接。建立连接后，客户端向服务器发出请求，服务器将响应客户端的请求。客户端可请求从服务器设备读取部分存储器，或将一定数量的数据写入服务器设备的存储器。如果请求有效，则服务器将响应该请求；如果请求无效，则会回复错误消息。



STEP 7-Micro/WIN SMART 提供了两种 Modbus TCP 库指令，这些指令位于 STEP 7-Micro/WIN SMART V3 项目树中“指令”(Instructions) 文件夹的“库”(Libraries) 文件夹中。

- MBUS_CLIENT (页 604)
- MBUS_SERVER (页 609)

Modbus TCP 客户端协议

Modbus 客户端指令 (MBUS_CLIENT) 使用 CPU 的以下资源：

- 每次连接至 Modbus 服务器，均有一个主动连接资源。MBUS_CLIENT 自动生成连接 ID。
- Modbus 客户端使用以下程序实体：
  - 1 个子程序
  - 2849 个字节的程序空间
  - V 存储器的 662 字节模块（用于指令变量）
    必须通过 STEP 7-Micro/WIN SMART V3 的库存储器命令为该模块分配起始地址。将 MBUS_CLIENT 指令置于程序中后，可从项目树中的“程序块”(Program Block) 或“程序块 > 库”(Program Block > Library) 文件夹访问库存储器命令。

Modbus TCP 服务器协议

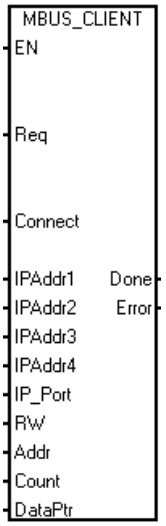
Modbus 服务器指令 (MBUS_SERVER) 使用 CPU 的以下资源：

- 每次连接至 Modbus 服务器，均有一个被动连接资源。MBUS_SERVER 自动生成连接 ID。
- Modbus 服务器使用下列程序实体：
  - 1 个子程序
  - 2969 个字节的程序空间
  - V 存储器的 445 字节模块（用于指令变量）
    必须通过 STEP 7-Micro/WIN SMART V3 的库存储器命令为该模块分配起始地址。将 MBUS_SERVER 指令置于程序中后，可从项目树中的“程序块”(Program Block) 或“程序块 > 库”(Program Block > Library) 文件夹访问库存储器命令。

11.4.2 Modbus TCP 客户端

11.4.2.1 MBUS_CLIENT 指令

表格 11-14 MBUS_CLIENT 指令

LAD/FBD	STL	描述
	**Call MBUS_CLIENT Req, Connect, IPAddr1, IPAddr2, IPAddr3, IPAddr4, IP_Port, RW, Addr, Count, DataPtr, Done, Error**	MBUS_CLIENT 作为 Modbus TCP 客户端通过 S7-200 SMART CPU 上的以太网端口进行通信。 MBUS_CLIENT 可建立客户端-服务器连接、发送 Modbus 功能请求、接收客户端响应，以及连接至 Modbus TCP 服务器和断开与此服务器的连接。

程序执行周期每次扫描都必须调用 MBUS_CLIENT，直到 Done 输出为 TRUE。在每个周期中，MBUS_CLIENT 均会退出，以便程序可以继续运行。客户端完成请求时，MBUS_CLIENT 将 Done 设置为 TRUE。

表格 11-15 参数的数据类型

参数和类型		数据类型	描述
Req	IN	BOOL	Req 参数允许程序向服务器发送 Modbus 请求。 FALSE：无 Modbus 通信请求 TRUE：请求与 Modbus TCP 服务器进行通信
Connect	IN	BOOL	Connect 参数允许程序连接到 Modbus 服务器设备或断开与此设备的连接。 如果 Connect = TRUE 且不存在连接，则 MBUS_CLIENT 尝试与分配的 IP 地址和端口号建立连接。 如果 Connect = FALSE 且存在连接，则 MBUS_CLIENT 尝试断开连接操作。当 Connect = FALSE 时，CPU 会忽略任何进一步请求。这意味着，如果程序调用 MBUS_CLIENT（相应的 Req = TRUE，但 Connect = FALSE），则 CPU 会忽略该请求。

参数和类型		数据类型	描述
IPAddr1	IN	BYTE	客户端尝试连接、且随后使用 Modbus 应用协议进行通信的服务器的 IP 地址的第一个八位字节。
IPAddr2	IN	BYTE	客户端尝试连接、且随后使用 Modbus 应用协议进行通信的服务器的 IP 地址的第二个八位字节。
IPAddr3	IN	BYTE	客户端尝试连接、且随后使用 Modbus 应用协议进行通信的服务器的 IP 地址的第三个八位字节。
IPAddr4	IN	BYTE	客户端尝试连接、且随后使用 Modbus 应用协议进行通信的服务器的 IP 地址的第四个八位字节。
IP_Port	IN	WORD	客户端尝试连接、且随后使用 Modbus TCP 进行通信的服务器的端口号。 默认值：502 将端口设置为设备的实际端口号。
RW	IN	BYTE	分配请求类型（读取或写入），其中 0 = 读取，1 = 写入。有关详细信息，请参见下面的 Modbus 功能表。
Addr	IN	DWORD	Modbus 起始地址：分配要通过 MBUS_CLIENT 进行访问的数据的起始地址。有关详细信息，请参见下面的 Modbus 功能表。
Count	IN	INT	Modbus 数据长度：此请求中要访问的位或保持寄存器的数量 范围 10001 至 19999 和 30001 至 39999 是只读地址。对于输入和输出位，最大 Count 值为 1920 位。对于输入和保持寄存器，最大 Count 值为 120 字。 有关详细信息，请参见下面的 Modbus 功能表。
DataPtr	IN_OUT	DWORD	指向 Modbus 数据寄存器的指针：DataPtr 指向与读取或写入请求关联的数据的 V 存储位置。对于读取请求，此位置是存储从 Modbus 服务器读取的数据的第一个存储位置。对于写入请求，此位置是要写入 Modbus 服务器的数据的第一个存储位置。
Done	OUT	BOOL	TRUE：以下任一条件时为真： • 客户端已与服务器建立连接 • 客户端已与服务器断开连接 • 客户端已接收 Modbus 响应 • 发生错误 FALSE：客户端正忙于建立连接或等待来自服务器的 Modbus 响应。
Error	OUT	BYTE	指令执行结果 仅在发生错误后的一个周期内有效

RW 和 Addr 参数选择 Modbus 通信功能

MBUS_CLIENT 指令使用 RW 输入指示读取或写入功能，使用 Addr 输入定义要读取或写入的数据类型。

下表显示了 MBUS_CLIENT 指令根据 RW 和 Addr 输入参数提供的 Modbus 功能：

FC	功能	RW	Addr	Count	CPU 地址
01	读取位	0	00001 至 09999	1 至 1920 位	Q0.0 至 Q1151.7
02	读取位	0	10001 至 19999	1 至 1920 位	I0.0 至 I1151.7
03	读取字	0	40001 至 49999 400001 至 465535	1 至 120 字	V 存储器
04	读取字	0	30001 至 39999	1 至 120 字	AIW0 - AIW530
05	写入单个位	1	00001 至 09999	1 位	Q0.0 至 Q1151.7
06	写入单个字	1	40001 至 49999 400001 至 465535	1 个字	V 存储器
15	写入多个位	1	00001 至 09999	1 至 1920 位	Q0.0 至 Q1151.7
16	写入多个字	1	40001 至 49999 400001 至 465535	1 至 120 字	V 存储器

多个客户端连接

Modbus TCP 客户端可支持多个连接，最多可达 PLC 允许的开放式用户通信连接的最大数。PLC 的连接总数（包括 Modbus TCP 客户端和服务器）不得超过支持的开放式用户通信连接的最大数。多个客户端连接必须具有不同的 IPAddr 或 IP_Port 输入参数。

建立连接

当 Connect 输入为 TRUE 时，客户端尝试与采用所提供 IP 地址和 IP 端口的服务器设备建立连接。如果服务器设备无法访问，连接请求最终会超时，这可能需要几秒钟。当连接请求正在进行时，任何其他操作均无法将其中断或中止。如果服务器不可用，则会立即拒绝客户端的连接请求。如果服务器可用，则客户端会建立连接，并可向服务器发送请求。如果没有可用于 Modbus 客户端的连接资源，则 MBUS_CLIENT 指令将返回错误。

处理请求

客户端仅在 Connect = TRUE 时处理请求。客户端与服务器建立连接后，在没有 Modbus 请求处于激活状态时，程序会通过调用 MBUS_CLIENT (Req = TRUE) 发送新请求。Modbus 客户端执行请求时，会捕获所有输入值。通过边沿检测元素（例如，上升沿）使 Req 输入呈脉冲形式，这会使指令发送一次请求。在请求处于激活状态时对输入值所做的任何后续更改都会导致 MBUS_CLIENT 返回错误代码。

客户端向服务器发送请求后，客户端会等待 mReceiveTimeout 时间段的响应。客户端在等待响应时，不可用于其他 Modbus 操作。如果客户端在 mReceiveTimeout 时间段内未收到响应，MBUS_CLIENT 将返回错误。

如果客户端收到来自服务器的有效响应，其将根据响应处理后续操作。随后，客户端返回到就绪状态，并可用于来自程序的其他请求。

断开已建立的连接

如果 Connect 输入为 FALSE，且客户端和服务器之间存在主动连接时，客户端会尝试断开与服务器的连接。如果正在进行连接或发送操作，则断开连接操作将返回错误。断开连接请求不能中断任何操作。如果没有正在进行的操作，则 CPU 将终止主动连接，且客户端返回空闲状态。随后，连接资源将可用于 CPU 中的其他操作。

11.4.2.2 Modbus TCP 客户端执行错误代码

MBUS_CLIENT 指令 (页 604)可返回以下错误代码：

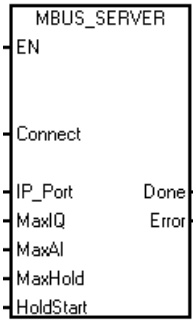
错误（十进制）	说明
0	无错误
32	未知状态 检查网络连接，查看程序是否未修改任何会干扰客户端/服务器通信的库符号。
33	连接正忙于另一个请求。一个连接一次只能用于一个 Modbus 请求。
34	Addr 输入是非法值。
35	Count 输入是非法值。
36	RW 输入是非法值。
37	请求的事务 ID 与服务器的响应不匹配。此错误表示服务器设备存在问题，或者响应请求的服务器设备错误。 从服务器收到无效的协议 ID。
38	从服务器收到无效的协议 ID。
39	服务器发送的字节数与“Count”输入值不匹配
40	请求的单元标识符与服务器的响应不匹配
41	请求的功能代码与服务器的响应不匹配
42	服务器发送的数据与 Modbus TCP 写入功能请求的数据不匹配
43	接收超时：服务器在 mReceiveTimeout 时间段内没有响应。检查与 Modbus 服务器设备的连接。
44	输入值与激活请求的值不匹配。
45	Modbus 数据寄存器范围超出 V 存储器范围。

除了上面列出的 MBUS_CLIENT 错误之外，另请参见“Modbus TCP 常规异常代码 (页 620)”和“开放式用户通信错误代码 (页 645)”

11.4.3 Modbus TCP 服务器

11.4.3.1 MBUS_SERVER 指令

表格 11-16 MBUS_SERVER 指令

LAD/FBD	STL	描述
	**Call MBUS_SERVER Connect, IP_Port, MaxIQ, MaxAI, MaxHold, HoldStart, Done, Error**	MBUS_SERVER 作为 Modbus TCP 服务器通过以太网端口进行通信。 MBUS_SERVER 可接受与 Modbus TCP 客户端连接的请求、接收 Modbus 功能请求，以及发送响应消息。

在每次扫描中执行 MBUS_SERVER 指令，以便 Modbus 服务器可在合理时间内响应 Modbus 客户端的请求。MBUS_SERVER 指令用于建立连接、接收请求和发送响应。每次扫描时，程序均应调用 MBUS_SERVER，否则 Modbus 服务器无法正常运行。

表格 11-17 参数的数据类型

参数和类型		数据类型	描述
Connect	IN	BOOL	可使用 Connect 参数连接至客户端设备，或从客户端设备断开连接。Modbus 服务器尝试创建“被动”连接，这表示服务器将接受来自任一请求 IP 地址的连接请求。 如果 Connect = TRUE，且客户端尚未与服务器建立连接，则服务器将被动监听 TCP 连接请求。 如果 Connect = FALSE 且存在连接，则服务器将发起断开连接操作。因此，程序可使用 Connect 参数来控制服务器何时可接受连接。当 Connect = FALSE 时，MBUS_SERVER 不会执行其他操作。 请注意，MBUS_SERVER 可在发生特定 TCP 错误时自动发起断开连接操作。
IP_Port	IN	WORD	客户端将尝试连接、且使用 Modbus 应用协议进行通信的服务器的端口号。 默认值：502 将端口设置为设备的实际端口号

参数和类型		数据类型	描述
MaxIQ	IN	WORD	参数 MaxIQ 将可用于 Modbus 地址 0xxxx 到 1xxxx 的 I 和 Q 点数设置为 0 至 9216。值为 0 时，将禁用所有对输入和输出的读写操作。建议将 MaxIQ 值设置为 9216。 **Note**: 如果想要读取 I 1151.7 之后的地址，或者读写 Q 1151.7 之后的地址，您需要先把这部分地址映射到 V 区， 通过保持寄存器进行读写。
MaxAI	IN	WORD	参数 MaxAI 将可用于 Modbus 地址 3xxxx 的字输入 (AI) 寄存器数设置为 0 至 266。值为 0 时，将禁止读取模拟量输入。要允许访问所有 CPU 模拟量输入，MaxAI 的建议值为 266。
MaxHold	IN	WORD	参数 MaxHold 用于设置 Modbus 地址 4xxxx 或 4yyyyy 可访问的 V 存储器中的字保持寄存器数。例如，如果要允许 Modbus 客户端访问 2000 字节的 V 存储器，则将 MaxHold 的值设置为 1000 字（保持寄存器）。
HoldStart	IN	DWORD	参数 HoldStart 为指向 V 存储器中保持寄存器起始位置的指针。通常将该值设置为 &VB0（VB0 地址）。可将其他 V 存储器地址设置为保持寄存器的起始地址，以允许在项目的其他位置使用 VB0。Modbus 客户端可访问 V 存储器的 HoldStart 字数（从 MaxHold 开始）。 如果 HoldStart 指向超出允许范围的存储位置，则 Modbus TCP 库指令将返回错误。CPU 还会生成非致命错误：间接寻址错误 (0x06)。
Done	OUT	BOOL	TRUE：MBUS_SERVER 执行下列操作之一： • 连接至客户端设备 • 与客户端断开连接 • 响应 Modbus 请求 • 返回错误 FALSE：没有请求用于此程序周期
Error	OUT	BYTE	指令执行结果 仅在发生错误后的一个周期内有效

断开连接

当 Connect = TRUE 时，CPU 使用来自开放式用户通信可用连接的一个被动连接资源。在程序请求 Modbus 操作时，保持 Connect 输入为 TRUE。可将 Connect 设为 FALSE，以释放连接资源。Modbus 服务器请求连接时，CPU 会捕获输入参数的值。如果在 Connect = TRUE 时更改输入值，则 MBUS_SERVER 会返回错误。

11.4.3.2 Modbus TCP 服务器执行错误代码

MBUS_SERVER 指令 (页 609)可返回以下错误代码：

错误（十进制）	说明
0	无错误
32	未知状态 检查网络连接，查看程序是否未修改任何会干扰客户端/服务器通信的库符号。
33	输入 MaxIQ 的值无效
34	输入 MaxAI 的值无效
35	输入 MaxHold 的值无效
36	HoldStart 输入不在 V 存储器中，或者保持寄存器的范围超出 V 存储器范围
37	保持寄存器与 Modbus 服务器符号重叠
38	输入值与当前连接的值不匹配。复位连接以更新输入值。

除了上面列出的 MBUS_SERVER 错误之外，另请参见“Modbus TCP 常规异常代码 (页 620)”和“开放式用户通信错误代码 (页 645)”

11.4.4 示例：Modbus TCP 应用程序

以下示例为两个 Modbus TCP 客户端与两个 Modbus TCP 服务器进行通信的项目。唯一的 IP 地址可标识每个服务器。程序逻辑可监视 MBUS_CLIENT 指令的“完成”输出，以确保程序不会中断正在进行的通信请求。该示例程序可执行下列功能：

- 写入输出位
- 读取输出位
- 写入保持寄存器
- 读取保持寄存器

程序、程序段和符号注释描述了下表中 Modbus TCP 示例程序的功能。

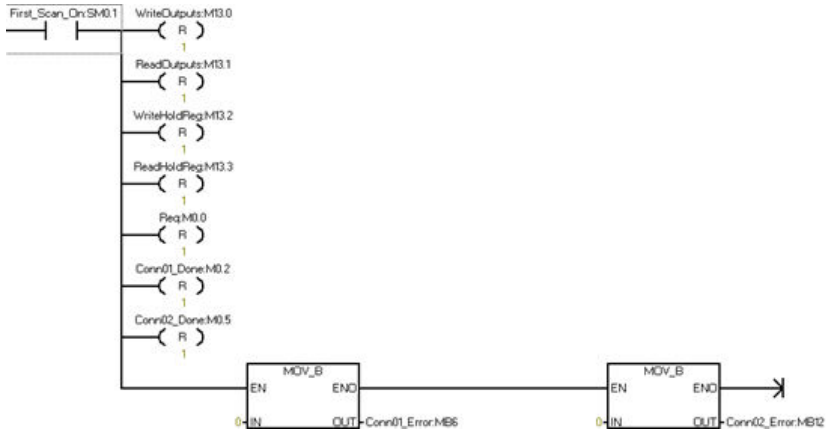
此示例的基本描述：

两个 Modbus 客户端与两个 Modbus 服务器设备建立连接。

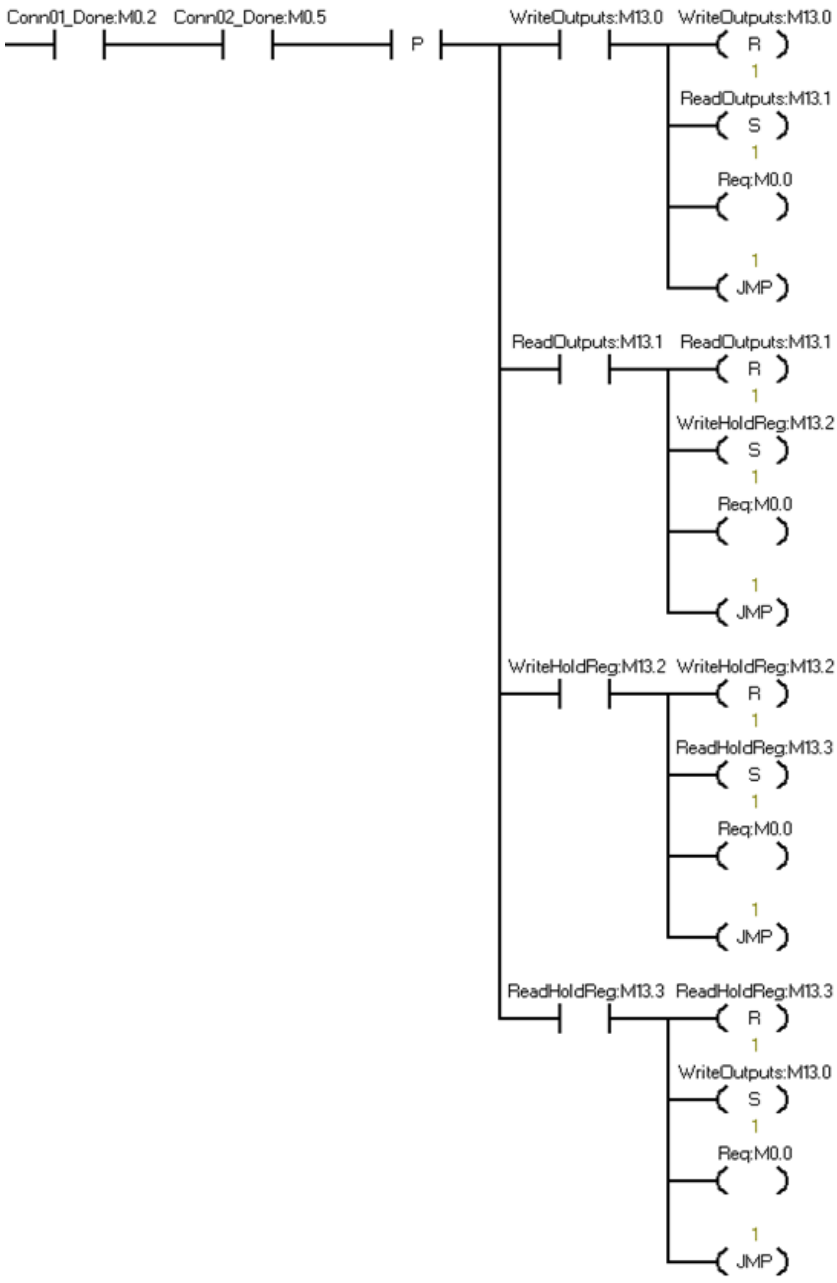
Modbus 服务器 01：IP 地址 192.168.2.10，端口 502

Modbus 服务器 02：IP 地址 192.168.2.66，端口 502

当 CPU 输入 0 转换为 TRUE 时，将启动一个序列，在该序列中 Modbus 客户端向两个 Modbus 服务器发送写入和读取请求。CPU 输入 0 设置为 False 时将关闭该序列。

LAD	说明
	程序段 1： 启动时，清除所有标志和错误。

符号	地址	注释
Conn01_Done	M0.2	Connection #1 has completed the current operation
Conn01_Error	MB6	Connection #1 Status
Conn02_Done	M0.5	Connection #2 has completed the current operation
Conn02_Error	MB12	Connection #2 Status
First_Scan_On	SM0.1	ON for the first scan cycle only
ReadHoldReg	M13.3	Initiate a Read Holding Register request
ReadOutputs	M13.1	Initiate a Read Output request
Req	M0.0	Initiate a Modbus request
WriteHoldReg	M13.2	Initiate a Write Holding Register request
WriteOutputs	M13.0	Initiate a Write Output request

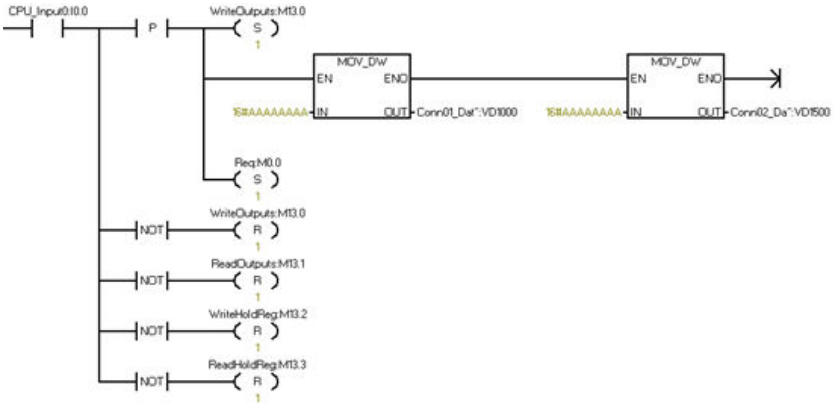
LAD	说明
	程序段 2： 两个客户端均完成 Modbus 请求后，将启动下一个 Modbus 请求。

LAD

符号	地址	注释
Conn01_Done	M0.2	Connection #1 has completed the current operation
Conn02_Done	M0.5	Connection #2 has completed the current operation
ReadHoldReg	M13.3	Initiate a Read Holding Register request
ReadOutputs	M13.1	Initiate a Read Output request
Req	M0.0	Initiate a Modbus request
WriteHoldReg	M13.2	Initiate a Write Holding Register request
WriteOutputs	M13.0	Initiate a Write Output request

说明

程序段 3：



程序段 4：

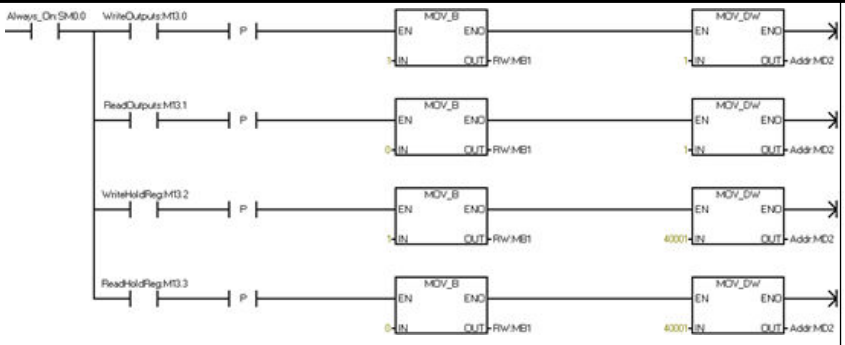
CPU_Input 0 上升沿触发 Modbus 请求序列开始在 V 存储器中写入一些数据，以发送到 Modbus 服务器。

将 Req 输入位设为 TRUE。

当 CPU_Input 0 为 False 时，停止发送 Modbus 请求。

符号	地址	注释
Conn01_Data	VD1000	Connection #1 Data Location
Conn02_Data	VD1500	Connection #2 Data Location
CPU_Input0	I0.0	
ReadHoldReg	M13.3	Initiate a Read Holding Register request
ReadOutputs	M13.1	Initiate a Read Output request
Req	M0.0	Initiate a Modbus request
WriteHoldReg	M13.2	Initiate a Write Holding Register request
WriteOutputs	M13.0	Initiate a Write Output request

LAD



符号	地址	注释
Addr	MD2	Modbus Address
Always_On	SM0.0	Always ON
ReadHoldReg	M13.3	Initiate a Read Holding Register request
ReadOutputs	M13.1	Initiate a Read Output request
RW	MB1	0 = Read Data; 1 = Write Data
WriteHoldReg	M13.2	Initiate a Write Holding Register request
WriteOutputs	M13.0	Initiate a Write Output request

说明

程序段 5：

设置所选 Modbus 功能的读/写模式和地址。

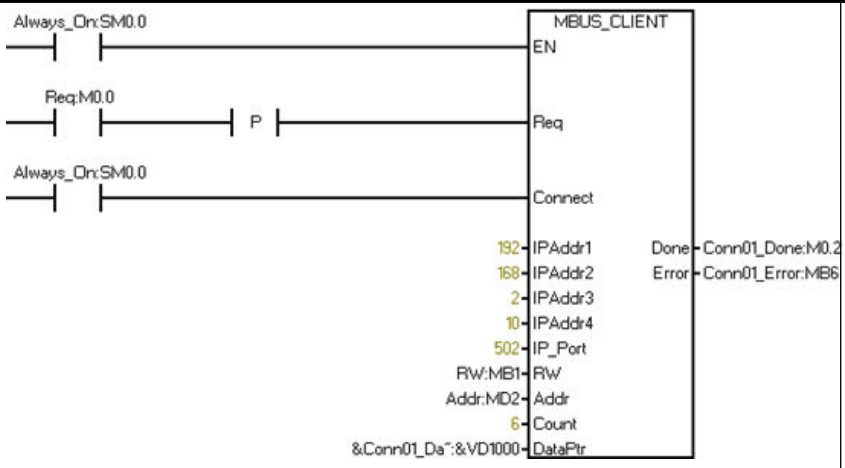
写入输出 = 功能代码 5（单个）/15（多个）。

读取输出 = 功能代码 1。

写入保持寄存器 = 功能代码 6（单个）/16（多个）。

读取保持寄存器 = 功能代码 3。

LAD



符号	地址	注释
Addr	MD2	Modbus Address
Always_On	SM0.0	Always ON
Conn01_Data	VD1000	Connection #1 Data Location
Conn01_Done	M0.2	Connection #1 has completed the current operation
Conn01_Error	MB6	Connection #1 Status
Req	M0.0	Initiate a Modbus request
RW	MB1	0 = Read Data; 1 = Write Data

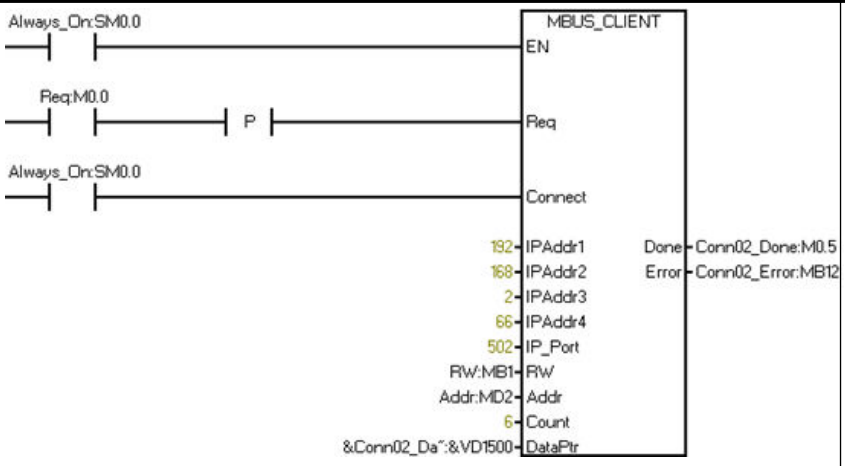
说明

程序段 6：

Modbus 客户端 01 与 Modbus 服务器 01 建立连接。

Req 为 TRUE 时，向服务器发送 Modbus 请求。

Modbus 客户端接收并处理来自服务器的响应后，MBUS_CLIENT 指令会将 Done 输出设置为 TRUE。

LAD	说明
	程序段 7： Modbus 客户端 02 与 Modbus 服务器 02 建立连接。 Req 为 TRUE 时，向服务器发送 Modbus 请求。 Modbus 客户端接收并处理来自服务器的响应后，MBUS_CLIENT 指令会将 Done 输出设置为 TRUE。

符号	地址	注释
Addr	MD2	Modbus Address
Always_On	SM0.0	Always ON
Conn02_Data	VD1500	Connection #2 Data Location
Conn02_Done	M0.5	Connection #2 has completed the current operation
Conn02_Error	MB12	Connection #2 Status
Req	M0.0	Initiate a Modbus request
RW	MB1	0 = Read Data; 1 = Write Data

11.4.5 Modbus TCP 高级用户信息

概述

本主题包含 Modbus TCP 库的高级用户信息。大多数用户不需要此信息，也不需要修改 Modbus TCP 库的默认操作。

MBUS_CLIENT 变量

下表显示了一些 Modbus 客户端变量，如果默认值不适用于您的应用，则可在程序中修改这些变量以调整 Modbus 客户端的操作：

变量	数据类型	默认值	说明
mBlocked_Proc_Timeout	REAL	3000	阻止过程超时：在将受阻的 Modbus 客户端实例移除（作为 ACTIVE）之前等待此实例的时间（以毫秒为单位）。例如，当程序发出客户端请求，且应用程序在完成请求之前停止执行客户端功能时，将发生这种情况。
mModbus_Unit_ID	WORD	255	Modbus 单元标识符：mModbus_Unit_ID 参数对应于 Modbus RTU 协议中的从站地址。如果 Modbus TCP 服务器用于 Modbus RTU 协议的网关，则 MB_UNIT_ID 可用于识别串行网络上连接的从站设备。MB_UNIT_ID 用于将请求转发到正确的 Modbus RTU 从站地址。 一些 Modbus TCP 设备可能要求 MB_UNIT_ID 参数在限制范围内。
mReceiveTimeout	REAL	2000	接收消息超时：MBUS_CLIENT 等待服务器响应请求的时间（以毫秒为单位）。范围：500 - 65,535 毫秒。
mConnected	BOOL	FALSE	连接状态：指示与指定服务器的连接状态是已连接还是已断开： TRUE：已连接 FALSE：已断开连接 程序可在处理 MBUS_CLIENT 请求后检查 mConnected。
mRetries	BYTE	3	重试次数：客户端在初始请求返回连接错误后，尝试断开连接并重新发送请求的次数 范围：0 至 255

重试

如果存在与连接相关的错误，Modbus 客户端指令会自动重新启动连接，并将请求重新发送至服务器设备：

设置 Done 和 Error 输出参数之前，Modbus 客户端会重新发送请求两次。

可通过在 Modbus 客户端符号表中查找符号 mModbusRetries，并在程序执行 MBUS_CLIENT 之前更改值，来更改重试次数。mRetries 值为 BYTE，重试次数范围为 0 到 255。

单个位与多个位/字写入功能

一些 Modbus 服务器设备不支持 Modbus 功能写入单个离散输出位（Modbus 功能 5）或写入单个保持寄存器（Modbus 功能 6）。相反，这些设备只支持多位写入（Modbus 功能 15）或多寄存器写入（Modbus 功能 16）。如果服务器设备不支持单个位/字 Modbus 功能，则 MBUS_CLIENT 指令将返回错误代码 1。

Modbus 客户端协议允许强制 MBUS_CLIENT 指令使用多个位/字 Modbus 功能，而非使用单个位/字 Modbus 功能。可通过在 Modbus 客户端符号表中查找符号 mModbusForceMulti，并在程序执行 MBUS_CLIENT 之前更改此值，来强制多个位/字指令。将 mModbusForceMulti 设置为 TRUE，可在写入单个位或寄存器时强制使用多个位/字功能。

保持寄存器地址大于 49999

Modbus 保持寄存器地址在 40001 到 49999 这一范围内。该范围足以满足大多数应用的要求，但有些 Modbus 从站设备将数据映射到地址范围更大的保持寄存器中。

MBUS_CLIENT 指令允许参数 **Addr** 的附加范围，以支持 400001 至 465536 的保持寄存器地址的扩展范围。

例如，要访问保持寄存器 16768，请将 MBUS_CLIENT 的 **Addr** 参数设置为 416768。

扩展寻址允许访问 Modbus 协议支持的全部 65536 个可能地址。此扩展寻址仅适用于保持寄存器。

MBUS_SERVER 变量

下表显示了一些 Modbus 服务器变量，如果默认值不适用于您的应用，则可在程序中修改这些变量以调整 Modbus 服务器的操作：

变量	数据类型	默认值	说明
mConnected	BOOL	0	连接状态：指示与指定客户端的连接状态是已连接还是已断开： TRUE：已连接 FALSE：已断开连接 每次执行 MBUS_SERVER 指令后，连接状态均为最新。

11.4.6 Modbus TCP 常规异常代码

错误编号：	说明
1	Modbus 异常（代码 0x01）：非法功能 - 服务器不支持请求的功能
2	Modbus 异常（代码 0x02）：非法数据地址 - Addr 加 Count 的请求地址范围超出了允许的服务器地址范围
3	Modbus 异常（代码 0x03）：非法数据值 - 服务器接收的 Modbus 协议出错
4	Modbus 异常（代码 0x04）：服务器设备故障 - 服务器尝试执行请求的操作时，发生不可恢复的错误
5	Modbus 异常（代码 0x05）：确认 - 服务器响应可能延迟；稍后重新发送请求
6	Modbus 异常（代码 0x06）：服务器设备忙 - 服务器拒绝了该消息；重新发送请求
7	Modbus 异常（代码 0x07）：否定确认 - 服务器因未知原因拒绝了该消息
10	Modbus 异常（代码 0x0A）：网关路径不可用 - 通常表示网关组态错误或过载。（仅限 Modbus TCP）
11	Modbus 异常（代码 0x0B）：网关目标设备无法响应 - 通常表示设备未存在于网络上。（仅限 Modbus TCP）

11.4.7 Modbus TCP 常规通信异常代码

Modbus TCP 通信异常代码如下：

错误代码	说明
161	数据长度参数大于允许的最大长度（1024 字节）。
162	数据缓冲区未处于 I、Q、M 或 V 存储区。
163	数据缓冲区不适合存储区。
164	表格参数不适合存储区。
165	连接在其它上下文中被锁定。您正在尝试同时在背景（Main 程序）和中断例程中访问同一连接。
166	UDP IP 地址或端口错误
167	实例不符：在另一实例中连接为忙，或是当发起请求时，为所请求的连接 ID 保存的数据与输入数据不符。

错误代码	说明
168	由于连接从未创建，所以连接 ID 不存在，或连接按您的要求终止（使用 TDCON 指令）。
169	使用此连接 ID 的 TCON 操作正在进行中。
170	使用此连接 ID 的 TDCON 操作正在进行中。
171	使用此连接 ID 的 TSEND 指令正在进行中。
172	发生了临时通信错误。此时无法启动连接。请稍后重试。
173	连接伙伴拒绝或主动断开连接（伙伴将断开与此 CPU 的连接）。
174	无法与连接伙伴建立连接（连接请求无应答）。
175	连接因不一致而断开。断开并重新连接以纠正这种情况。
176	连接 ID 已与不同的 IP 地址、端口或 TSAP 组合配合使用。
177	没有连接资源可用。所有请求类型（主动/被动）的连接都在使用中。
178	本地或远程端口号被保留，或端口号已用于另一服务器（被动）连接。
179	已发生以下 IP 地址错误之一： • IP 地址无效（例如，地址 0.0.0.0）。 • 该 IP 地址是此 CPU 的 IP 地址。 • 该 CPU 的 IP 地址为 0.0.0.0。 • IP 地址为广播地址或多播地址。
180	本地或远程 TSAP 错误（仅 ISO-on-TCP）
181	连接 ID 无效（65535 保留）
182	主动/被动错误（UDP 只允许被动）
183	连接类型不在所允许的类型中。
184	没有待决操作，因此没有要报告的状态。
185	接收缓冲区过小：CPU 接收的字节数超出缓冲区支持的长度。CPU 丢弃额外的字节。
191	未知错误

11.5 开放式用户通信库

11.5.1 概述

STEP 7-Micro/WIN SMART 开放式用户通信 (OUC) 库指令创建 OUC 指令（TCON、TSEND、TRECV 和 TDCON）所需的表。库指令根据需要构建表，调用 OUC 指令，然后在库指令的输出中呈现状态值。CPU 使用库存储器创建表以传递到 OUC 指令。开放式用户通信库需要使用 50 个字节的 V 存储器。

库指令如下所示：

- TCP_CONNECT：创建 TCP 连接。
- ISO_CONNECT：创建 ISO-on-TCP 连接。
- UDP_CONNECT：创建 UDP 连接。
- TCP_SEND：发送用于 TCP 和 ISO-on-TCP 连接的数据指令。
- TCP_RECV：接收用于 TCP 和 ISO-on-TCP 连接的数据指令。
- UDP_SEND：发送用于 UDP 连接的数据指令。
- UDP_RECV：接收用于 UDP 连接的数据指令。
- DISCONNECT：终止所有协议的连接。

说明

只可从主程序或中断例程中调用库函数，但不可同时从这两个程序中调用。

11.5.2 OUC 库指令共用的参数

以下参数是 OUC 库指令共用的参数：

- EN：将 EN 输入设置为 TRUE 以调用指令。必须将 EN 输入设置为 TRUE，直到指令完成（直到 Done 或 Error 置位）。仅当程序置位 EN 并且调用指令时，CPU 才会更新输出。
- Req：Req（请求）输入用于发起操作。Req 输入位由电平触发。应通过上升沿指令将 Req 输入连接到库指令，以便操作仅启动一次。指令为 Busy 时程序会忽略 Req 输入。

- Active: Active 输入用于指定连接指令是创建主动客户端连接 (Active = TRUE) 还是创建被动服务器连接 (Active = FALSE)。在主动连接中，本地 CPU 启动到远程设备的通信。在被动连接中，本地 CPU 等待远程设备启动通信。
  对于开放式用户通信，S7-200 SMART CPU 支持八个主动连接和八个被动连接。将 UDP 连接计作被动连接，因为没有建立主动通信。
- Done：当操作完成且没有错误时，OUC 指令置位 Done 输出。如果指令置位 Done 输出，Busy、Error 和 Status 输出为零。仅当 Done 输出置位时，其它输出（例如，接收到的字节数）才有效。
- Busy：Busy 输出指示正在进行操作。通过将 Req 设为 TRUE 启动操作时，OUC 指令置位 Busy 输出。对于对指令的所有后续调用，Busy 输出保持置位，直到操作完成。
- Error：Error 输出指示操作完成但有错误。如果 OUC 指令置位 Error 输出，则 Done 和 Busy 输出将设置为 FALSE。如果 OUC 指令置位 Error 输出，则 Status 输出会指明错误原因。如果 Error 输出置位，所有其它输出均无效。
- ConnID：ConnID 编号是连接的标识符。通过 TCP_CONNECT、ISO_CONNECT 或 UDP_CONNECT 创建连接时，会创建 ConnID。可以为 ConnID 选择 0 到 65534 范围内的任何值。每个连接必须具有唯一的 ConnID。程序使用 ConnID 指定后续发送、接收和断开操作所需的连接。
- IPaddr1, IPaddr2, IPaddr3 and IPaddr4：这些是远程设备 IP 地址的四个八位字节。
  IPaddr1 是 IP 地址的最高有效字节，IPaddr4 是 IP 地址的最低有效字节。例如：对于 IP 地址 192.168.2.15，设置以下值：
  - IPaddr1 = 192
  - IPaddr2 = 168
  - IPaddr3 = 2
  - IPaddr4 = 15

  IP 地址不能为以下值：
  - 0.0.0.0（针对主动连接）
  - 任何广播 IP 地址（例如，255.255.255.255）
  - 任何多播地址
  - 本地 CPU 的 IP 地址

  可以将 IP 地址 0.0.0.0 用于被动连接。通过选择 IP 地址 0.0.0.0，S7-200 SMART CPU 接受来自任何远程 IP 地址的连接。如果为被动连接选择一个非零的 IP 地址，CPU 将仅接受来自指定地址的连接。

- RemPort：RemPort 是远程设备上的端口号。端口号可用于 TCP 和 UDP 协议，从而路由设备内的消息。
  远程端口号的规则如下：
  – 有效端口号范围为 1 到 49151。
  – 建议采用的端口号范围为 2000 到 5000。
  – 对于被动连接，CPU 会忽略远程端口号（可以将其设置为零）。
- LocPort：LocPort 参数是本地 CPU 上的端口号。端口号可用于 TCP 和 UDP 协议，从而路由设备内的消息。对于所有被动连接，本地端口号必须唯一。
  本地端口号的规则如下：
  – 有效端口号范围为 1 到 49151。
  – 不能使用端口号 20、21、25、80、102、135、161、162、443 以及 34962 至 34964。这些端口具有特定用途。
  – 建议采用的端口号范围为 2000 到 5000。
  – 对于被动连接，本地端口号必须唯一（不重复）。
- RemTsap: RemTsap（远程传输服务访问点 (TSAP)）参数是指向 S7-200 SMART 字符串数据类型的指针。只能将 RemTsap 参数用于 ISO-on-TCP 协议。在将消息路由到适当的连接方面，远程 TSAP 字符串与端口号作用相同。
  RemTsap 的规则如下：
  – TSAP 为 S7-200 SMART 字符串数据类型（长度字节，后接字符）。
  – TSAP 字符串长度必须至少为 2 个字符，但不得超过 16 个字符。
- LocTsap：LocTsap（本地传输服务访问点 (TSAP)）参数是指向 S7-200 SMART 字符串数据类型的指针。只能将本地 TSAP 参数用于 ISO-on-TCP 协议。在将消息路由到适当的连接方面，本地 TSAP 字符串与端口号作用相同。
  LocTsap 的规则如下：
  – TSAP 为 S7-200 SMART 字符串数据类型（长度字节，后接字符）。
  – TSAP 字符串长度必须至少为 2 个字符，但不得超过 16 个字符。
  – 如果 TSAP 为 2 个字符，第一个字符必须是十六进制“E0”。
  – TSAP 不能以字符串“SIMATIC-”开头。

11.5.3 开放式用户通信库指令

11.5.3.1 TCP_CONNECT 指令

TCP_CONNECT 指令用于通过 TCP 协议创建到另一设备的连接。

LAD/FBD	STL	描述
	TCP_CONNECT Req, Active, ConnID, IPaddr1, IPaddr2, IPaddr3, IPaddr4, RemPort, LocPort, Done, Busy, Error, Status	TCP_CONNECT 用于创建从 CPU 到通信伙伴的 TCP 通信连接。

连接操作是异步的，可能需要几次扫描才能完成。连接操作待决时，Busy 输出具有值 TRUE。当 CPU 完成操作时，指令置位 Done 或 Error 输出。如果发生错误，则 Status 输出会显示错误代码。

指令处于繁忙状态时不得更改 TCP_CONNECT 的输入参数。CPU 需要凭借这一点了解这是启动连接过程的调用的延续。

您将连接 ID (ConnID) 输入分配给连接，然后当发送、接收或断开连接时使用此 ConnID 引用该连接。

Active 输入位确定这是主动连接（Active 设置为 TRUE）还是被动连接（Active 设置为 FALSE）。

如果这是主动连接（客户端），则 S7-200 SMART CPU 尝试联系并创建到指定 IP 地址和远程端口号 (RemPort) 的连接。CPU 打开本地端口 (LocPort) 以从远程设备接收消息。

当 Active 输入设置为 FALSE 时，S7-200 SMART CPU 会创建被动（服务器）连接。在这种情况下，CPU 打开请求的本地端口 (LocPort) 并接受来自远程设备的连接请求。如果要接受来自任何远程 IP 地址的连接请求，应将 IP 地址设为 0.0.0.0。如果 IP 地址不为零，则 CPU 只接受来自指定 IP 地址的连接请求。对于被动连接，CPU 会忽略远程端口号 (RemPort)，RemPort 可以设置为零。

您可以随时调用 TCP_CONNECT 指令以确定连接的当前状态。将 Req 输入设置为 FALSE 并提供有效的连接 ID (ConnID)，TCP_CONNECT 返回以下内容：

- Busy，如果连接过程仍在进行中。
- Done，如果连接处于激活状态并准备发送或接收。
- Error，如果连接不可用。Status 显示其中一种错误代码，用于指示存在的问题。

请注意，主动连接可能最多需要 30 秒的时间来确定远程设备是否允许连接。被动连接显示 Busy 状态，直到远程设备尝试连接到 CPU。

请注意，连接关闭后，S7-200 SMART 不会自动尝试重新连接到设备。如果远程设备断开设备连接，您的程序必须执行另一个 TCP_CONNECT 指令以重新连接设备。主动连接和被动连接皆如此。

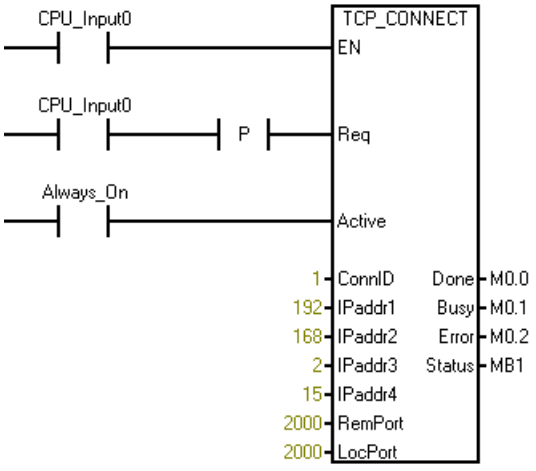
表格 11-18 TCP_CONNECT 指令的参数

参数	声明	数据类型	描述
EN	IN	BOOL	使能输入
Req	IN	BOOL	如果 Req = TRUE，CPU 启动连接操作。如果 Req = FALSE，则输出显示连接的当前状态。
Active	IN	BOOL	• TRUE = 主动连接 • FALSE = 被动连接
ConnID	IN	WORD	CPU 使用连接 ID (ConnID) 为其它指令标识该连接。可能的 ConnID 范围为 0 到 65534。
IPaddr1 ... IPaddr4	IN	BYTE	这些是 IP 地址的四个八位字节。IPaddr1 是 IP 地址的最高有效字节，IPaddr4 是 IP 地址的最低有效字节。
RemPort	IN	WORD	RemPort 是远程设备上的端口号。远程端口号范围为 1 到 49151。对于被动连接，使用零。
LocPort	IN	WORD	LocPort 是本地设备上的端口号。本地端口号范围为 1 到 49151，但存在一些限制。如需了解 LocPort 定义，请参见“OUC 库指令共用的参数” (页 622)。
Done	OUT	BOOL	当连接操作完成且没有错误时，指令置位 Done 输出。
Busy	OUT	BOOL	当连接操作正在进行时，指令置位 Busy 输出。

参数	声明	数据类型	描述
Error	OUT	BOOL	当连接操作完成但发生错误时，指令置位 Error 输出。有关详细信息，请参见“开放式用户通信库指令错误代码”（页 645）。
Status	OUT	BYTE	如果指令置位 Error 输出，Status 输出会显示错误代码。如果指令置位 Busy 或 Done 输出，Status 为零（无错误）。

示例

这是 TCP_CONNECT 指令的应用示例：



11.5.3.2 ISO_CONNECT 指令

ISO_CONNECT 指令使用 ISO-on-TCP 协议创建到另一设备的连接。除了 TCP 协议外，该协议还使用 RFC1006，以便更好地描绘消息。ISO-on-TCP 的优点是，对于发送的每条消息，接收到的消息都会不同。ISO-on-TCP 协议从不将接收到的多条消息组合成一条消息，而 TCP 协议会发生这种情况。ISO-on-TCP 协议使用 TSAP（传输服务访问点）路由设备中的消息，而不是端口的消息。

LAD/FBD	STL	描述
	ISO_CONNECT Req, Active, ConnID, IPaddr1, IPaddr2, IPaddr3, IPaddr4, RemTsap, LocTsap, Done, Busy, Error, Status	ISO_CONNECT 用于创建从 CPU 到通信伙伴的 ISO-on-TCP 通信连接。

连接操作是异步的，可能需要几次扫描才能完成。当连接操作待决时，指令置位 Busy 输出。当 CPU 完成操作时，指令置位 Done 或 Error 输出。如果发生错误，则 Status 输出会显示错误代码。

指令处于繁忙状态时不得更改 ISO_CONNECT 的输入参数。CPU 需要凭借这一点了解这是启动连接过程的调用的延续。

您将连接 ID (ConnID) 输入分配给连接，然后当发送、接收或断开连接时使用此 ConnID 引用该连接。

Active 输入位确定这是主动连接（Active 设置为 TRUE）还是被动连接（Active 设置为 FALSE）。

如果这是主动连接（客户端），则 S7-200 SMART CPU 尝试联系和创建到指定 IP 地址和远程 TSAP (RemTsap) 的连接。CPU 打开本地 TSAP (LocTsap) 以从远程设备接收消息。

当 Active 输入设置为 FALSE 时，S7-200 SMART CPU 会创建被动（服务器）连接。在这种情况下，CPU 打开请求的本地 TSAP (LocTsap) 并接受来自远程设备的连接请求。如果要接受来自任何远程 IP 地址的连接请求，应将 IP 地址设为 0.0.0.0。如果 IP 地址不为零，则 CPU 只接受来自指定 IP 地址的连接请求。对于被动连接，CPU 会忽略远程 TSAP 字符串 (RemTsap)，RemTsap 可以设置为空字符串（例如，“”）。

您可以随时调用 ISO_CONNECT 指令以确定连接的当前状态。将 Req 输入设置为 FALSE 并提供有效的连接 ID (ConnID)，ISO_CONNECT 返回以下内容：

- Busy，如果连接过程仍在进行中。
- Done，如果连接处于激活状态并准备发送或接收。
- Error，如果连接不可用。Status 显示其中一种错误代码，用于指示存在的问题。

请注意，主动连接可能最多需要 30 秒的时间来确定远程设备是否允许连接。被动连接显示 Busy 状态，直到远程设备尝试连接到 CPU。

请注意，连接关闭后 S7-200 SMART 不会自动尝试重新连接到设备。如果远程设备断开设备连接，您的程序必须执行另一个 ISO_CONNECT 指令以重新连接设备。主动连接和被动连接皆如此。

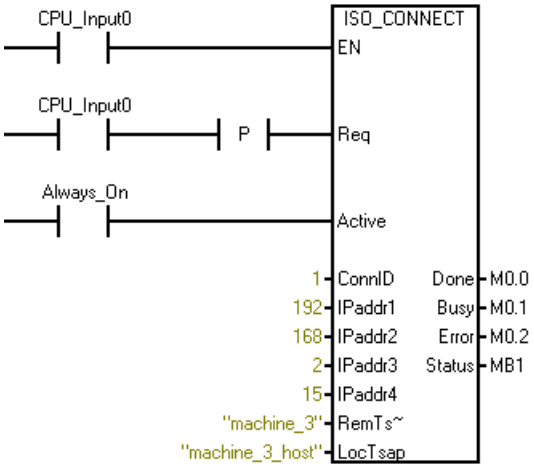
表格 11-19 ISO_CONNECT 指令的参数

参数	声明	数据类型	描述
EN	IN	BOOL	使能输入
Req	IN	BOOL	如果 Req = TRUE，CPU 启动连接操作。如果 Req = FALSE，则输出显示连接的当前状态。
Active	IN	BOOL	• TRUE = 主动连接 • FALSE = 被动连接
ConnID	IN	WORD	CPU 使用连接 ID (ConnID) 为其它指令标识该连接。可能的 ConnID 范围为 0 到 65534。
IPaddr1 ... IPaddr4	IN	BYTE	这些是 IP 地址的四个八位字节。IPaddr1 是 IP 地址的最高有效字节，IPaddr4 是 IP 地址的最低有效字节。
RemTsap	IN	String<254>	RemPort 是远程 TSAP 字符串。程序使用指针来传递字符串。（更多信息，请参见本表后面的示例。）
LocTsap	IN	String<254>	LocPort 是本地 TSAP 字符串。程序使用指针来传递字符串。（更多信息，请参见本表后面的示例。）
Done	OUT	BOOL	当连接操作完成且没有错误时，指令置位 Done 输出。
Busy	OUT	BOOL	当连接操作正在进行时，指令置位 Busy 输出。

参数	声明	数据类型	描述
Error	OUT	BOOL	当连接操作完成但发生错误时，指令置位 Error 输出。有关详细信息，请参见“开放式用户通信库指令错误代码”（页 645）。
Status	OUT	BYTE	如果指令置位 Error 输出，Status 输出会显示错误代码。如果指令置位 Busy 或 Done 输出，Status 为零（无错误）。

示例

这是 ISO_CONNECT 指令的应用示例：

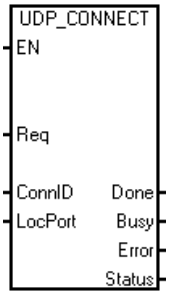


对于 RemTsap 和 LocTsap，STEP 7 Micro/WIN SMART 始终使用指针将字符串传递给 ISO_CONNECT 指令。如果您使用常量字符串（如以上示例所示），STEP7-Micro/WIN SMART 会自动创建字符串和指针。如果您希望在数据块中创建字符串，然后将指针传递给这些字符串之一，请执行以下步骤：

1. 在数据块中，创建字符串：
   - VB100 "machine_1"
   - VB120 "machine_2"
2. 将“＆VB100”或“＆VB120”（不带引号）用于 ISO_CONNECT 指令中的 TSAP 参数。

11.5.3.3 UDP_CONNECT 指令

UDP_CONNECT 指令使用 UDP 协议创建被动连接。UDP 是一种无连接协议，因此不会在此 CPU 和远程设备之间创建实际连接。UDP 连接打开所选本地端口以与 UDP 协议配合使用。

LAD/FBD	STL	描述
	**UDP_CONNECT Req, ConnID, LocPort, Done, Busy, Error, Status**	UDP_CONNECT 使用 UDP 协议创建被动连接以打开所选本地端口。

UDP_CONNECT 指令只需要连接 ID 和本地端口号即可创建连接。一个 UDP 连接可以将消息发送到任意数量的其它设备，因为 IP 地址和远程端口会随每个 UDP_SEND 指令一起提供。仅当需要多个本地端口时，才需要多个 UDP 连接。不能将同一本地端口号用于多个 UDP 连接。所有本地端口号必须唯一。

连接操作是异步的，可能需要几次扫描才能完成。没有与远程设备建立主动连接，也没有等待另一台设备连接到该 CPU。当连接操作待决时，Busy 输出置位。当连接操作完成时，程序置位 Done 输出。仅当输入参数发生问题或没有可用的被动连接时，程序才置位 Error 输出。如果程序将 Error 位置位，Status 输出字节将包含错误代码。

指令处于繁忙状态时，不得更改 UDP_CONNECT 的参数，CPU 借此可了解这是启动连接过程的调用的延续。

您可以调用 UDP_CONNECT 指令以确定连接的当前状态。将 Req 输入设置为 FALSE 并提供有效的连接 ID (ConnID)，UDP_CONNECT 指令返回以下内容：

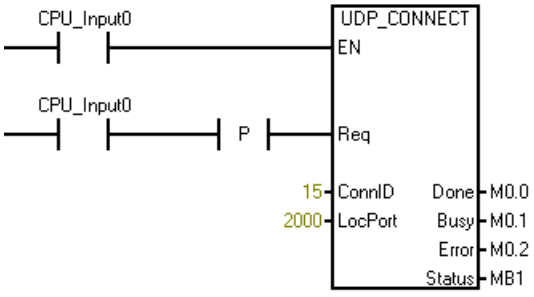
- 如果连接处于激活状态并准备发送或接收，指令置位 Done 输出。这仅表示您可以使用该连接，并不表示存在任何远程设备。
- 如果连接仍在进行，指令置位 Busy 输出。
- 如果连接不可用，指令置位 Error 输出。Status 输出字节显示其中一种错误代码，用于指示存在的问题。

表格 11-20 UDP_CONNECT 指令的参数

参数	声明	数据类型	描述
EN	IN	BOOL	使能输入
Req	IN	BOOL	如果 Req = TRUE，CPU 启动连接操作。如果 Req = FALSE，则输出显示连接的当前状态。
ConnID	IN	WORD	CPU 使用连接 ID (ConnID) 为其它指令标识该连接。可能的 ConnID 范围为 0 到 65534。
LocPort	IN	WORD	LocPort 是本地设备上的端口号。本地端口号范围为 1 到 49151，但存在一些限制。如需了解 LocPort 定义，请参见“OUC 库指令共用的参数” (页 622)。
Done	OUT	BOOL	当连接操作完成且没有错误时，指令置位 Done 输出。
Busy	OUT	BOOL	当连接操作正在进行时，指令置位 Busy 输出。
Error	OUT	BOOL	当连接操作完成但发生错误时，指令置位 Error 输出。有关详细信息，请参见“开放式用户通信库指令错误代码” (页 645)。
Status	OUT	BYTE	如果指令置位 Error 输出，Status 输出会显示错误代码。如果指令置位 Busy 或 Done 输出，Status 为零（无错误）。

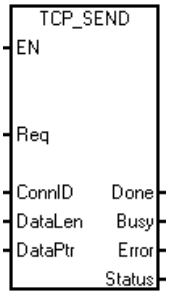
示例

这是 UDP_CONNECT 指令的应用示例：



11.5.3.4 TCP_SEND 指令

TCP_SEND 指令通过现有连接 (ConnID) 传输来自请求的缓冲区位置 (DataPtr) 的请求的字节数 (DataLen)。您可以将该指令用于 TCP 协议和 ISO-on-TCP 协议。

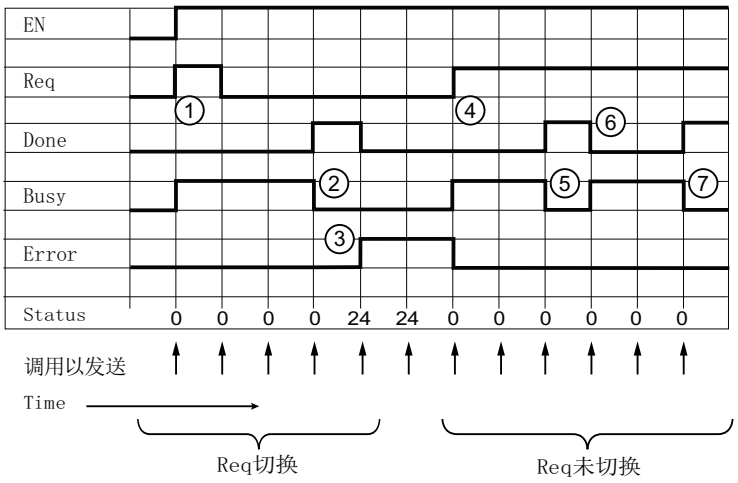
LAD/FBD	STL	描述
	**TCP_SEND Req, ConnID, DataLen, DataPtr, Done, Busy, Error, Status**	TCP_SEND 通过现有连接传输来自请求的缓冲区位置的请求的字节数。

当发生以下情况时，TCP_SEND 指令启动发送指定数量的字节的操作：

- 程序通过将 Req 输入设置为 TRUE 来调用指令。
- 连接当前未用于执行其它发送操作。

Req 输入由电平触发。建议对 Req 输入使用上升沿触发器，以便指令不启动意外的发送操作。TCP_SEND 处于繁忙状态时，程序会忽略 Req 输入。Done、Busy 和 Error 输出及 Status 输出字节显示各调用的 TCP_SEND 状态。

发送操作完成后，指令显示调用一次 TCP_SEND 的 Done 或 Error 状态。此后，TCP_SEND 通过错误代码 24 作出响应，这意味着操作待决（如果通过将 Req 输入设置为 FALSE 进行调用）。如果 Req 输入设置为 TRUE，则程序会启动另一个发送操作。下图显示了输入和输出参数之间的关系。



① Req 设置为 TRUE 以便开始执行消息发送操作。Busy 设置为 TRUE。

② 消息发送完成。Done 置位，Busy 清零。

③ EN 为 TRUE 且 Req 为 FALSE，但无任何消息发送操作正在执行。因此，Error 置位且显示错误代码 24。

④ Req 再次设置为 TRUE，因此开始执行另一消息发送操作。Busy 设置为 TRUE。

⑤ 消息发送完成。Done 置位，Busy 在一个扫描周期内清零。

⑥ Req 保持为 TRUE，因此开始执行另一消息发送操作。

⑦ 消息发送完成。

在一个发送操作中最多可以发送 1024 字节的数据。若在 Req 输入设置为 TRUE 时执行 TCP_SEND，程序会将用户存储器中发送缓冲区的数据复制到内部缓冲区。TCP_SEND 执行且指令置位 Busy 输出后，您可以更改程序发送缓冲区。

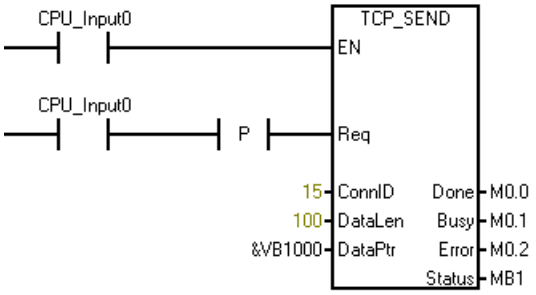
表格 11-21 TCP_SEND 指令的参数

参数	声明	数据类型	描述
EN	IN	BOOL	使能输入
Req	IN	BOOL	如果 Req = TRUE，CPU 启动发送操作。如果 Req = FALSE，则输出显示发送操作的当前状态。
ConnID	IN	WORD	连接 ID (ConnID) 是此发送操作所用连接的编号。使用您为 TCP_CONNECT 操作选择的 ConnID。
DataLen	IN	WORD	DataLen 是要发送的字节数（1 到 1024）。

参数	声明	数据类型	描述
DataPtr	IN	DWORD	DataPtr 是指向待发送数据的指针。这是指向 I、Q、M 或 V 存储器的 S7-200 SMART 指针（例如，＆VB100）。
Done	OUT	BOOL	当发送操作完成且没有错误时，指令置位 Done 输出。
Busy	OUT	BOOL	当发送操作正在进行时，指令置位 Busy 输出。
Error	OUT	BOOL	当发送操作完成但发生错误时，指令置位 Error 输出。有关详细信息，请参见“开放式用户通信库指令错误代码”（页 645）。
Status	OUT	BYTE	如果指令置位 Error 输出，Status 输出会显示错误代码。如果指令置位 Busy 或 Done 输出，Status 为零（无错误）。

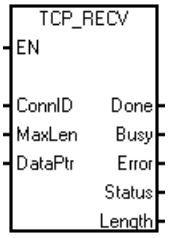
示例

这是 TCP_SEND 指令的应用示例：



11.5.3.5 TCP_RECV 指令

TCP_RECV 指令通过现有连接检索数据。您可以将该指令用于 TCP 协议和 ISO-on-TCP 协议。

LAD/FBD	STL	描述
	**TCP_RECV ConnID, MaxLen, DataPtr, Done, Busy, Error, Status, Length**	TCP_RECV 通过现有连接检索数据。

TCP_RECV 指令仅具有 EN（使能）输入。TCP_RECV 指令没有 Req（请求）输入。第一次执行 TCP_RECV 指令后，状态位显示指令处于繁忙状态。对 TCP_RECV 的后续调用会显示繁忙状态，直至 CPU 通过指定连接接收数据。

CPU 通过指定连接接收消息后，下一次执行 TCP_RECV 指令时，会执行以下任务：

- 将消息数据复制到程序的数据区 (DataPtr)
- 将 Length 输出设置为接收的字节数
- 置位 Done 输出，清除 Busy 和 Error 输出，且将 Status 输出字节值设置为零（无错误）

您应该分配接收区/缓冲区 (DataPtr) 和接收缓冲区最大长度 (MaxLen)，从而避免缓冲区溢出。如果 CPU 接收到的字节数超出程序缓冲区的容量（由 MaxLen 指定），TCP_RECV 指令会将 MaxLen 字节复制到程序的数据区，并丢弃所接收字节的其余部分。在这种情况下，指令置位 Error 输出且 Status 输出字节显示错误代码 25，这表示接收缓冲区过小。

在一条消息中最多可以接收 1024 字节的数据。TCP_RECV 指令始终在允许接收不同长度消息的模式下工作。

根据使用的协议，TCP_RECV 指令的操作有所不同。当您调用 TCP_CONNECT （TCP 协议）或 ISO_CONNECT （ISO-on-TCP 协议）创建连接时，要为连接选择相应的协议。

使用 TCP 协议时，TCP_RECV 指令返回自程序上次调用 TCP_RECV 指令后 S7-200 SMART CPU 通过指定连接接收到的所有字节。程序必须足够频繁地调用 TCP_RECV 指令以正确地描绘消息，因为 TCP 充当“流”协议。在 TCP 协议中没有消息描述（没有开始或结束标记）。因此，CPU 并不知晓消息何时开始或结束。

例如，让我们假设存在一个 TCP 客户端接连不断地将四条 20 字节的消息发送给 CPU，而且在此期间程序不调用 TCP_RECV 指令。程序在 CPU 接受所有四条消息后调用 TCP_RECV 指令时，TCP_RECV 指令以一条 80 个字节的接收消息返回此数据。程序负责足够频繁地调用 TCP_RECV 指令，以便按发送消息的原样接收每条消息。

当您使用 ISO-on-TCP 协议创建连接时，协议本身会描绘消息。TCP_RECV 指令在 CPU 中以单独消息的形式接收远程设备发送的所有消息并进行保存，无论程序何时或以何种频率调用 TCP_RECV 指令均如此。

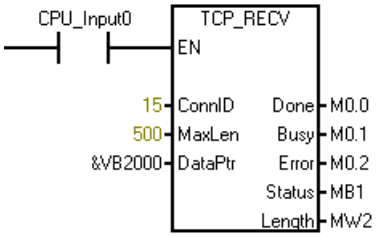
例如，让我们假设此时有一个 ISO-on-TCP 客户端接连不断地将四条 20 字节的消息发送给 CPU。还假设在此期间程序不会调用 TCP_RECV 指令。在对 TCP_RECV 指令的四次后续调用期间，ISO-on-TCP 协议提供了四条信息（每次调用一条消息）。ISO-on-TCP 在协议中具有起始和结束标记以描绘消息，并在接收设备中将消息分隔开，因此才会出现以上情况。

表格 11-22　TCP_RECV 指令的参数

参数	声明	数据类型	描述
EN	IN	BOOL	使能输入
ConnID	IN	WORD	连接 ID (ConnID) 是此接收操作所用连接的编号（连接过程中定义）。
MaxLen	IN	WORD	MaxLen 是要接收的最大字节数（例如，DataPtr 中缓冲区的大小（1 到 1024））。
DataPtr	IN	WORD	DataPtr 是指向接收数据存储位置的指针。这是指向 I、Q、M 或 V 存储器的 S7-200 SMART 指针（例如，＆VB100）。
Done	OUT	BOOL	当接收操作完成且没有错误时，指令置位 Done 输出。当指令置位 Done 输出时，Length 输出有效。
Busy	OUT	BOOL	当接收操作正在进行时，指令置位 Busy 输出。
Error	OUT	BOOL	当接收操作完成但发生错误时，指令置位 Error 输出。有关详细信息，请参见“开放式用户通信库指令错误代码”（页 645）。
Status	OUT	BYTE	如果指令置位 Error 输出，Status 输出会显示错误代码。如果指令置位 Busy 或 Done 输出，Status 为零（无错误）。
Length	OUT	WORD	Length 是实际接收的字节数。仅当指令置位 Done 或 Error 输出时，Length 才有效。如果指令置位 Done 输出，则指令接收整条消息。如果指令置位 Error 输出，则消息超出缓冲区大小 (MaxLen) 并被截短。

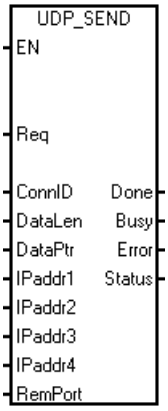
示例

这是 TCP_RECV 指令的应用示例：



11.5.3.6 UDP_SEND 指令

UDP_SEND 指令将来自请求的缓冲区位置 (DataPtr) 的请求的字节数 (DataLen) 传输到通过 IP 地址 (IPaddr1 – IPaddr4) 和端口 (RemPort) 指定的设备。该指令仅用于 UDP 协议和通过 UDP_CONNECT 创建的连接。

LAD/FBD	STL	描述
	**UDP_SEND Req, ConnID, DataLen, DataPtr, IPaddr1, IPaddr2, IPaddr3, IPaddr4, RemPort, Done, Busy, Error, Status**	UDP_SEND 指令将来自请求的缓冲区位置的请求的字节数传输到通过 IP 地址和端口指定的设备。

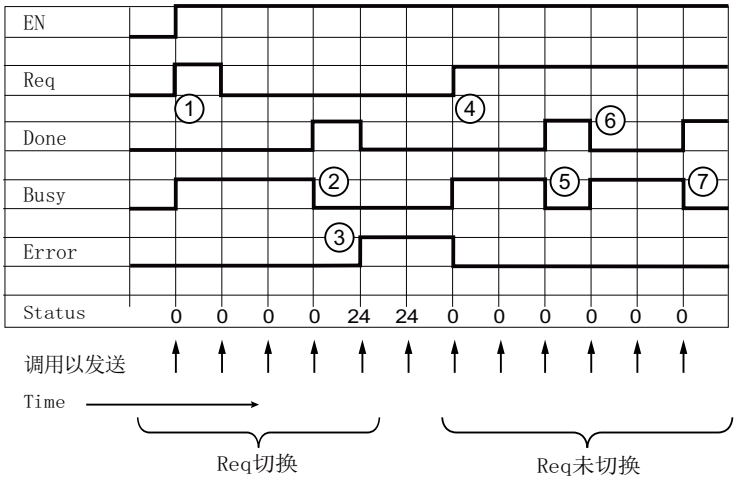
当发生以下情况时，UDP_SEND 指令启动发送指定数量的字节的操作：

- 程序通过将 Req 输入设置为 TRUE 来调用指令。
- 连接当前未用于执行其它发送操作。

Req 输入由电平触发。建议对 Req 输入使用上升沿触发器，以便指令不启动意外的发送操作。UDP_SEND 处于繁忙状态时，程序会忽略 Req 输入。Done、Busy 和 Error 输出及 Status 输出字节显示各调用的 UDP_SEND 状态。

发送操作完成后，指令显示调用一次 UDP_SEND 的 Done 或 Error 状态。此后，UDP_SEND 通过错误代码 24 作出响应，这意味着没有待决操作（如果通过将 Req 输入设置为 FALSE 进

行调用）。如果 Req 输入设置为 TRUE，则程序会启动另一个发送操作。下图显示了输入和输出参数之间的关系。



① Req 设置为 TRUE 以便开始执行消息发送操作。Busy 设置为 TRUE。

② 消息发送完成。Done 置位，Busy 清零。

③ EN 为 TRUE 且 Req 为 FALSE，但无任何消息发送操作正在执行。因此，Error 置位且显示错误代码 24。

④ Req 再次设置为 TRUE，因此开始执行另一消息发送操作。Busy 设置为 TRUE。

⑤ 消息发送完成。Done 置位，Busy 在一个扫描周期内清零。

⑥ Req 保持为 TRUE，因此开始执行另一消息发送操作。

⑦ 消息发送完成。

在一个发送操作中最多可以发送 1024 字节的数据。若在 Req 输入设置为 TRUE 时执行 UDP_SEND，程序会将用户存储器中发送缓冲区的数据复制到内部缓冲区。UDP_SEND 执行且指令置位 Busy 输出后，您可以更改程序发送缓冲区。

UDP_SEND 指令需要使用远程设备上的 IP 地址和端口号。当 UDP_CONNECT 创建连接后，会设置本地端口。IP 地址 (IPaddrx) 和远程端口号 (RemPort) 适用前面所述的相同的规则和限制。（有关这些规则，请参见“OUC 库指令共用的参数” (页 622)。）

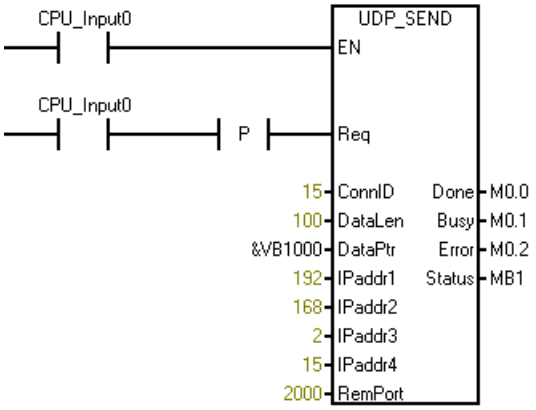
请注意，不能保证会传递 UDP 消息。如果远程设备不存在，不会返回错误。

表格 11-23 UDP_SEND 指令的参数

参数	声明	数据类型	描述
EN	IN	BOOL	使能输入
Req	IN	BOOL	如果 Req = TRUE，CPU 启动发送操作。如果 Req = FALSE，则输出显示发送操作的当前状态。
ConnID	IN	WORD	连接 ID (ConnID) 是此发送操作所用连接的编号（连接过程中通过 UDP_CONNECT 定义）。
DataLen	IN	WORD	DataLen 是要发送的字节数（1 到 1024）。
DataPtr	IN	DWORD	DataPtr 是指向待发送数据的指针。这是指向 I、Q、M 或 V 存储器的 S7-200 SMART 指针（例如，＆VB100）。
IPaddr1 … IPaddr4	IN	BYTE	这些是 IP 地址的四个八位字节。IPaddr1 是 IP 地址的最高有效字节，IPaddr4 是 IP 地址的最低有效字节。
RemPort	IN	WORD	RemPort 是远程设备上的端口号。远程端口号范围为 1 到 49151。
Done	OUT	BOOL	当连接操作完成且没有错误时，指令置位 Done 输出。
Busy	OUT	BOOL	当连接操作正在进行时，指令置位 Busy 输出。
Error	OUT	BOOL	当连接操作完成但发生错误时，指令置位 Error 输出。有关详细信息，请参见“开放式用户通信库指令错误代码”（页 645）。
Status	OUT	BYTE	如果指令置位 Error 输出，Status 输出会显示错误代码。如果指令置位 Busy 或 Done 输出，Status 为零（无错误）。

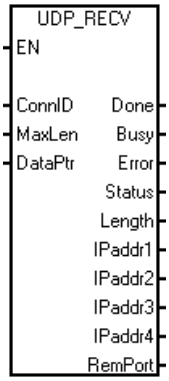
示例

这是 UDP_SEND 指令的应用示例：



11.5.3.7 UDP_RECV 指令

UDP_RECV 指令通过现有连接检索数据。该指令仅用于 UDP 协议以及通过 UDP_CONNECT 创建的连接。

LAD/FBD	STL	描述
	**UDP_RECV ConnID, MaxLen, DataPtr, Done, Busy, Error, Status, Length, IPaddr1, IPaddr2, IPaddr3, IPaddr4, RemPort**	UDP_RECV 通过现有连接检索数据。

UDP_RECV 指令仅具有 EN（使能）输入。UDP_RECV 指令没有 Req（请求）输入。第一次执行 UDP_RECV 指令后，状态输出显示指令处于繁忙状态。对 UDP_RECV 的后续调用显示繁忙状态，直至 CPU 通过指定连接接收数据。

CPU 通过指定连接接收消息后，下一次执行 UDP_RECV 指令时，会执行以下任务：

- 将消息数据复制到程序的数据区 (DataPtr)
- 将返回的 Length 设置为接收的字节数
- 将 IP 地址设置为发送消息的远程设备的地址

- 将远程端口号 (RemPort) 设置为远程设备的端口
- 置位 Done 输出，清除 Busy 和 Error 输出，且将 Status 输出字节值设置为零（无错误）

您应该分配接收区/缓冲区 (DataPtr) 和接收缓冲区最大长度 (MaxLen)，从而避免缓冲区溢出。如果 CPU 接收到的字节数超出程序缓冲区的容量（由 MaxLen 指定），UDP_RECV 指令会将 MaxLen 字节复制到程序的数据区，并丢弃接收的字节的其余部分。在这种情况下，指令置位 Error 输出且 Status 输出字节显示错误代码 25，这表示接收缓冲区过小。

在一条消息中最多可以接收 1024 字节的数据。UDP_RECV 指令始终在允许接收不同长度消息的模式下工作。来自远程设备的每条消息在 S7-200 SMART CPU 中均描绘为一条单独的消息。每次调用 UDP_RECV 指令时，该指令仅返回一条接收消息。

UDP_RECV 指令还会返回远程设备的 IP 地址和端口号。这允许程序通过 UDP_SEND （该指令需要远程 IP 地址和端口参数）响应远程设备。

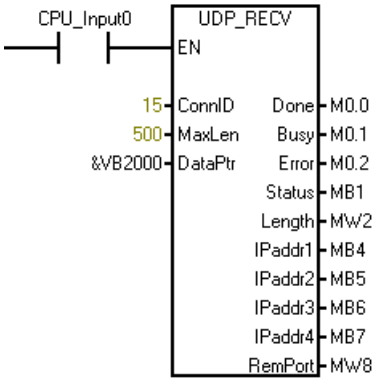
表格 11-24 UDP_RECV 指令的参数

参数	声明	数据类型	描述
EN	IN	BOOL	使能输入
ConnID	IN	WORD	CPU 将连接 ID (ConnID) 用于此接收操作（连接过程中定义）。
MaxLen	IN	WORD	MaxLen 是要接收的最大字节数（例如，DataPt 中缓冲区的大小（1 到 1024））。
DataPtr	IN	DWORD	DataPtr 是指向接收数据存储位置的指针。这是指向 I、Q、M 或 V 存储器的 S7-200 SMART 指针（例如，＆VB100）。
Done	OUT	BOOL	当接收操作完成且没有错误时，指令置位 Done 输出。当指令置位 Done 输出时，Length 输出有效。
Busy	OUT	BOOL	当接收操作正在进行时，指令置位 Busy 输出。
Error	OUT	BOOL	当接收操作完成但发生错误时，指令置位 Error 输出。有关详细信息，请参见“开放式用户通信库指令错误代码” (页 645)。
Status	OUT	BYTE	如果指令置位 Error 输出，Status 输出会显示错误代码。如果指令置位 Busy 或 Done 输出，Status 为零（无错误）。

参数	声明	数据类型	描述
Length	OUT	WORD	Length 是实际接收的字节数。仅当指令置位 Done 或 Error 输出时，Length 才有效。如果指令置位 Done 输出，则指令接收整条消息。如果指令置位 Error 输出，则消息超出缓冲区大小 (MaxLen) 并被截短。
IPaddr1 … IPaddr4	OUT	BYTE	这些是发送消息的远程设备 IP 地址的四个八位字节。IPaddr1 是 IP 地址的最高有效字节，IPaddr4 是 IP 地址的最低有效字节。
RemPort	OUT	WORD	RemPort 是发送消息的远程设备的端口号。

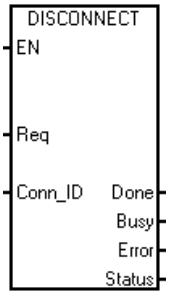
示例

这是 UDP_RECV 指令的应用示例：



11.5.3.8 DISCONNECT 指令

DISCONNECT 指令用于终止现有通信连接。DISCONNECT 指令用于所有协议。

LAD/FBD	STL	描述
	**DISCONNECT ConnID, LocPort, Done, Busy, Error, Status**	DISCONNECT 用于终止所有协议对应的现有通信连接。

程序通过将 Req 输入设置为 TRUE 来调用 DISCONNECT 指令时，DISCONNECT 指令启动连接终止操作。Req 输入由电平触发。建议对 Req 输入使用上升沿触发器。

如果请求的连接 (ConnID) 当前正忙于连接、断开连接，或者因为连接已被重用而无法找到，DISCONNECT 指令会返回错误。

断开操作完成后，DISCONNECT 指令为指令的至少一次调用显示 Done 或 Error 输出状态。指令可能返回针对后续调用的指定连接的 DISCONNECT 指令的状态，直到 CPU 重新将该连接用于另一连接操作。

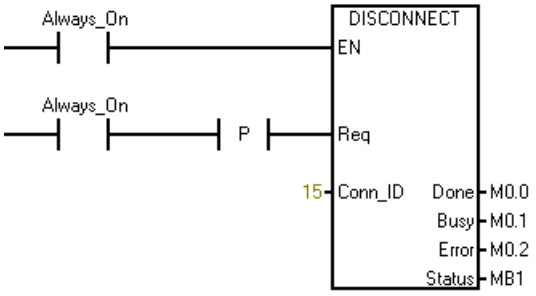
DISCONNECT 指令完成断开后，如果程序通过将 Req 设置为 FALSE 调用 DISCONNECT 指令，则指令会返回错误代码 24，这意味着没有待决操作。

表格 11-25 DISCONNECT 指令的参数

参数	声明	数据类型	描述
EN	IN	BOOL	使能输入
Req	IN	BOOL	如果 Req = TRUE，CPU 启动断开连接操作。
ConnID	IN	WORD	CPU 使用连接 ID (ConnID) 标识要终止的连接（连接过程中定义）。
Done	OUT	BOOL	当断开连接操作完成且没有错误时，指令置位 Done 输出。
Busy	OUT	BOOL	当断开连接操作正在进行时，指令置位 Busy 输出。
Error	OUT	BOOL	当断开连接操作完成但发生错误时，指令置位 Error 输出。有关详细信息，请参见“开放式用户通信库指令错误代码”（页 645）。
Status	OUT	BYTE	如果指令置位 Error 输出，Status 输出会显示错误代码。如果指令置位 Busy 或 Done 输出，Status 为零（无错误）。

示例

这是 DISCONNECT 指令的应用示例：



11.5.4 开放式用户通信库指令错误代码

下表列出开放式用户通信 (OUC) 库 (页 622)指令错误代码：

错误代码	描述	CONNECT	SEND	RECV	DISCONNECT
0	无错误	X	X	X	X
1	数据长度输入参数大于允许的最大值（1024 字节）。		X	X	
2	数据缓冲区未处于 I、Q、M 或 V 存储区。		X	X	
3	数据缓冲区不适合存储区。		X	X	
5	连接在另一上下文中被锁定。您正在尝试同时在背景（Main 程序）和中断程序组织单元 (POU) 中访问同一连接。	X	X	X	X
6	UDP IP 地址或端口错误		X		
7	实例不符：在另一实例中连接为忙，或是当发起请求时，为所请求的连接 ID 保存的数据与输入数据不符。	X	X	X	X

错误代码	描述	CONNECT	SEND	RECV	DISCONNECT
8	连接 ID 不存在，因为从未创建连接，或连接已由程序通过 DISCONNECT 指令终止。	X	X	X	X
9	TCP_CONNECT、ISO_CONNECT 或 UDP_CONNECT 指令正使用此连接 ID 执行。		X	X	X
10	DISCONNECT 指令正使用此连接 ID 执行。	X	X	X	
11	TCP_SEND 或 UDP_SEND 指令正使用此连接 ID 执行。		X		X
12	发生了临时通信错误。此时无法启动连接。再试一次。	X	X	X	
13	连接伙伴拒绝或主动终止了连接。伙伴发出与此 CPU 断开连接的命令。	X	X	X	
14	CPU 无法访问连接伙伴。连接请求无应答。	X	X	X	
15	由于存在不一致问题 CPU 中止了连接。断开并重新连接以纠正这种情况。	X	X	X	X
16	连接 ID 已与不同的 IP 地址、端口或 TSAP 组合配合使用。	X			
17	没有连接资源可用。所有请求类型（主动/被动）的连接都在使用中。	X			
18	本地或远程端口号被保留，或端口号已用于另一服务器（被动）连接。	X			
19	已发生以下 IP 地址错误之一： • IP 地址无效（例如，地址 0.0.0.0）。 • 该 IP 地址是此 CPU 的 IP 地址。 • 该 CPU 地址为 0.0.0.0。 • IP 地址为广播地址或多播地址。	X			
20	本地或远程 TSAP 错误（仅 ISO-on-TCP）	X			
21	连接 ID 无效（65535 保留）	X			

错误代码	描述	CONNECT	SEND	RECV	DISCONNECT
24	没有待决操作，因此没有要报告的状态。		X		X
25	接收缓冲区过小：CPU 接收的字节数超出缓冲区支持的长度。CPU 丢弃额外的字节。			X	
31	未知错误。断开并重新连接以解决问题。	X	X	X	X

11.5.5 开放式用户通信库示例

11.5.5.1 主动伙伴（客户端）

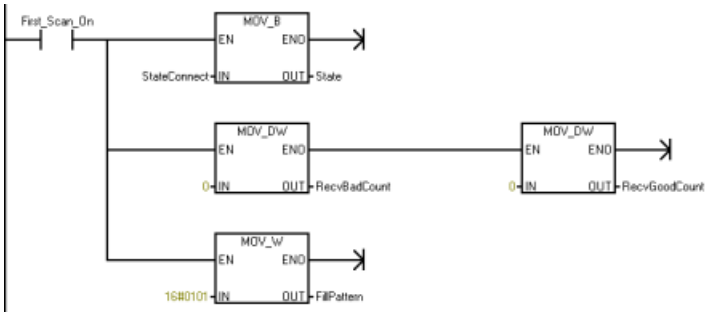
该程序用于实现简单状态机管理连接创建，周期性发送/接收消息，以及处理错误。

状态机的流程是建立连接，之后可重复发送和接收消息。如果连接断开，状态机尝试进行重新连接。

有关该程序的符号表，请参见“主动伙伴符号表”(页 657)。

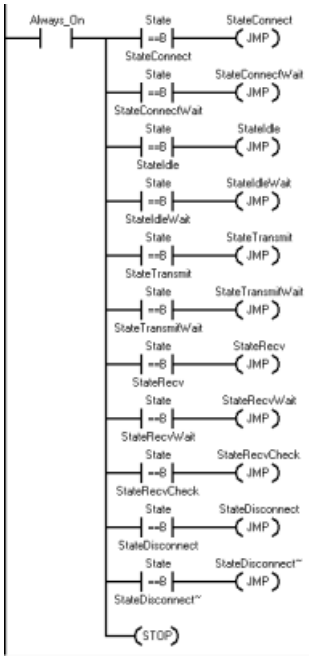
程序段 1：第一次扫描时....

初始化状态变量以初始化连接。清除接收到的优劣计数，并初始化某些待发送的数据。



程序段 2：处理状态机...

确定状态机的当前状态并跳转到状态处理程序标记处。如果该状态非法，则 CPU 进入 STOP 模式。

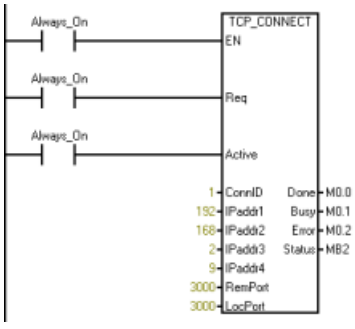


程序段 3：状态“连接”...

启动连接过程。

程序段 4：与被动设备建立主动连接。

该状态下此程序只能调用 TCP_CONNECT 指令一次。设置 Req 输入为 TRUE 以启动连接过程。由于状态机为连接的主动方，所以该指令设置 Active 输入为 TRUE。

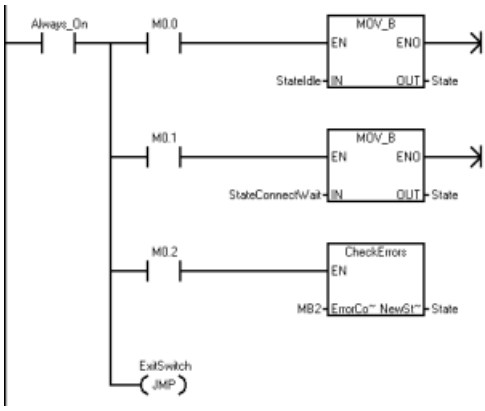


程序段 5：如果 Done 为 TRUE，则 CPU 建立该连接以进入“空闲”状态。

如果 Busy 为 TRUE，则 CPU 进入“连接等待”状态以等待连接的建立。

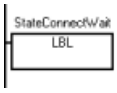
如果 Error 为 TRUE，则可能输入参数有误，检查输入参数以决定 CPU 接下来要进入哪个状态。

在所有情况下，都要先退出状态机才能进行该项扫描。该程序会继续进入下一项扫描的下一个状态。

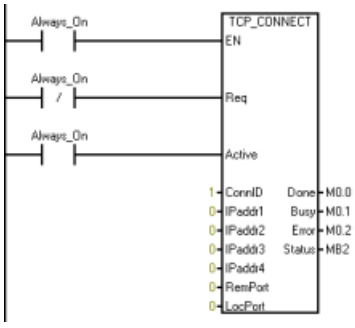


程序段 6：状态“连接等待”...

在该状态等待，直到与被动设备建立连接。



程序段 7：通过设置 Req = FALSE 并使用与上文中相同的连接 ID (ConnID) 调用 TCP_CONNECT 指令。执行该项操作以检查连接状态并确保 CPU 已建立该连接。

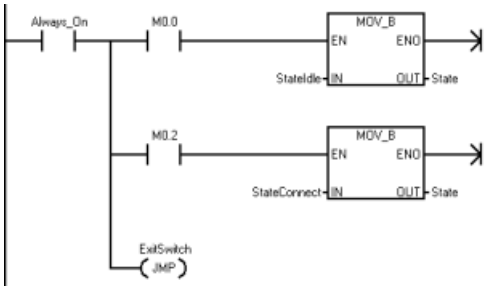


程序段 8：如果 Done 为 TRUE，则表示 CPU 已建立该连接，所以继续进入“空闲”状态。

如果 Busy 为 TRUE，则 CPU 停留在“连接等待”状态。如果没有搜索到其它设备，则 TCP_CONNECT 指令最终会超时并会返回错误消息，所以对于连接过程无需建立超时机制。

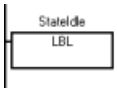
如果 Error 为 TRUE，则与被动设备的连接有误。在这种情况下，返回到“连接”状态并尝试再次建立连接。注意如果已搜索到被动设备，但其拒绝连接请求，则由于 CPU 不断尝试与被动设备建立连接，因此会快速出现连接错误并占用大量带宽。

在所有情况下，都要先退出状态机才能进行该项扫描。该程序会继续进入下一项扫描的下一个状态。

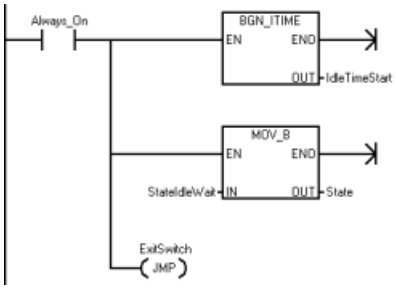


程序段 9：状态“空闲”...

该状态会在各消息之间建立时间延迟，所以不能快速推动程序段向下进行。符号“IdleTimeDelay”指定延迟时间。



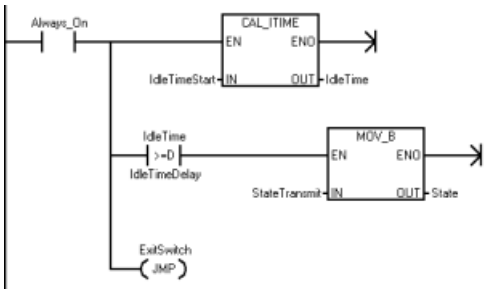
程序段 10：捕获间隔计时器并在下次传送前都处于“空闲等待”状态以延迟进程。



程序段 11：状态“空闲等待”...

在指定的毫秒数时间内处于该状态 (IdleTimeDelay)。

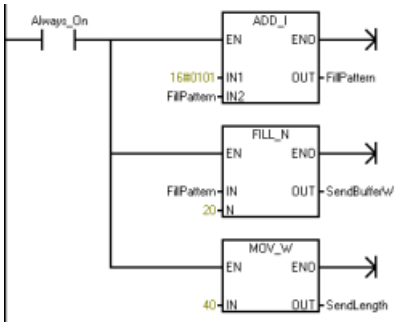
程序段 12：计算从进入“空闲”状态到现在的时间，如果其超过“IdleTimeDelay”值，则将状态更改为“传送”状态。



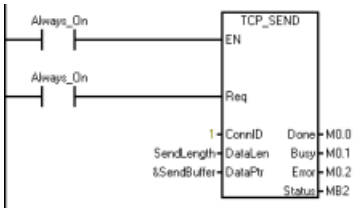
程序段 13：状态“传送”...

程序段 14：创建待发送到被动设备的消息。

对于测试程序，在缓冲区内填充 40 个字节（20 个字）。写下待发送到变量中的字节数，由此即可在“传送等待”状态使用与该值相同的值。



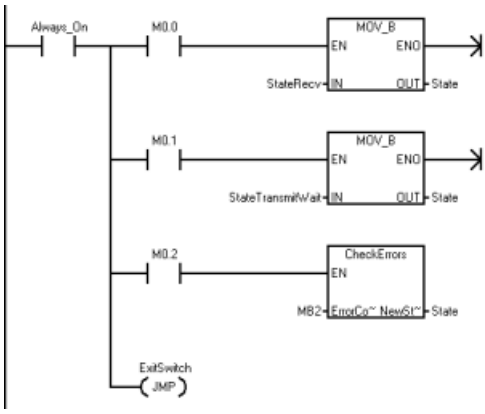
程序段 15：向被动设备发送新消息。设置 Req 为 TRUE 以初始化新的发送操作。



程序段 16：如果 Done 为 TRUE，则发送操作已完成（几乎不会很快完成），之后进入下一次扫描的“接收”状态。

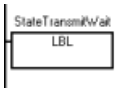
如果 Busy 为 TRUE（通常是这种情况），则进入“传送等待”状态以等待传送完成。

如果 Error 为 TRUE，则检查其原因，如果出现连接问题，可能需要更改状态。

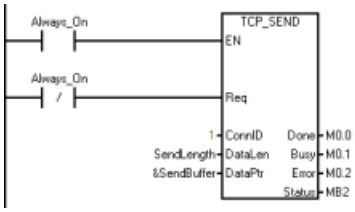


程序段 17：状态“传送等待”...

在该状态等待直到传送完成。



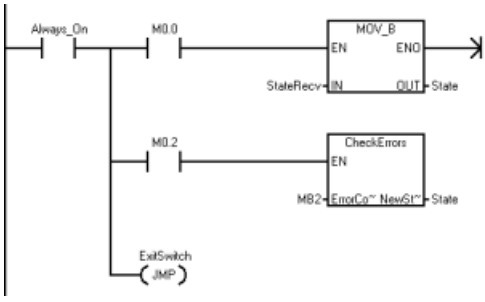
程序段 18：设置 Req = FALSE，调用 TCP_SEND 指令以确定发送是否已完成。确保使用初始化发送请求所用的发送长度和缓冲区指针。



程序段 19：如果 Done 为 TRUE，则发送已完成，之后进入“接收”状态。

如果 Busy 为 TRUE，则 CPU 停留在“传送等待”状态。

如果 Error 为 TRUE，则检查出现错误的原因，如果是连接出现问题，需要更改状态。

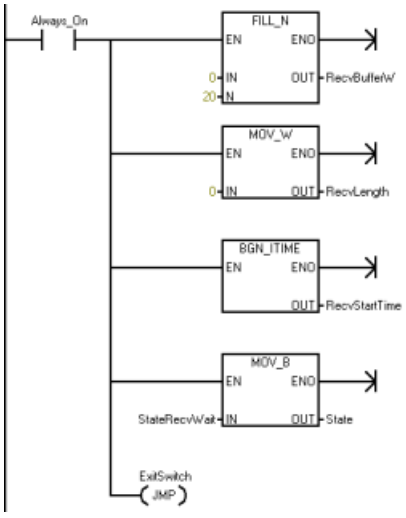


程序段 20：状态“接收”...

清空接收数据区，准备接收响应。

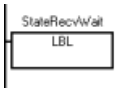
程序段 21：清理接收缓冲区和“RecvLength”，由此该区域内不会残留有上一个接收到的消息的数据。

捕获当前间隔时间值（在“RecvStartTime”中）以支持接收超时，之后进入“接收等待”状态。

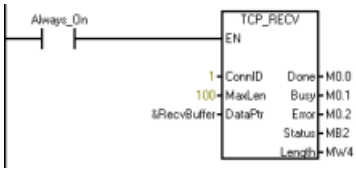


程序段 22：状态“接收等待”...

停留在该状态直到接收到某些数据或超时。



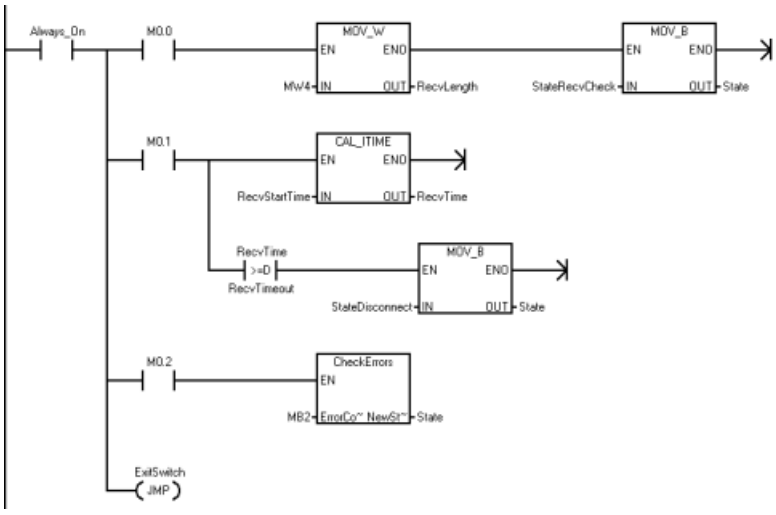
程序段 23：调用 TCP_RECV 指令以获取 CPU 接收到的任意消息。



程序段 24：如果 Done 为 TRUE，则该指令已接收到新数据，之后进入“接收检查”状态。

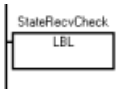
如果 Busy 为 TRUE，则停留在“接收”状态，直到停留时间达到接收超时值。如果停留在该状态的时间已超时，则断开与设备的连接，之后再重新与其建立连接。

如果 Error 为 TRUE，则检查错误代码以确定接下来的操作。

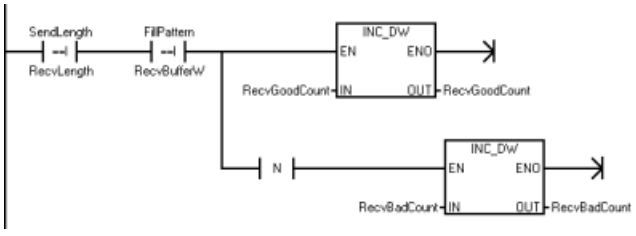


程序段 25：状态“RecvCheck”...

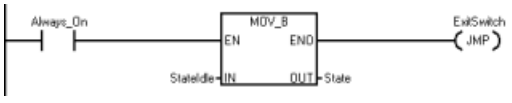
处理被动设备返回的数据。



程序段 26：该程序只能检查返回的数据以确保接收到的数据与发送出的数据等量，以及第一个词符合“填充模式”。无论响应好坏都记录下来，之后进入“空闲”状态，等待发送下一条消息。



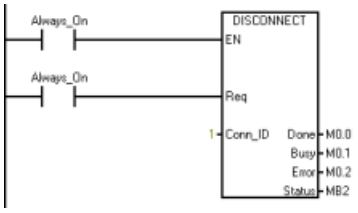
程序段 27：在所有情况下都进入“空闲”状态。



程序段 28：状态“断开连接”...

启动断开连接。

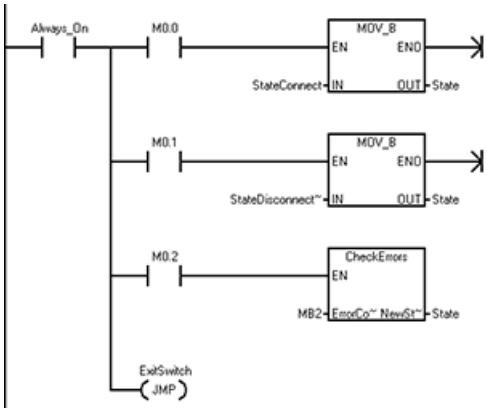
程序段 29：通过设置 Req = TRUE，调用 DISCONNECT 指令来启动与 ConnID 断开连接。



程序段 30：如果 Done 为 TRUE，则表示已断开连接（几乎不会发生），之后尝试再次建立连接。

如果 Busy 为 TRUE（通常是这种情况），则进入“断开连接等待”状态。

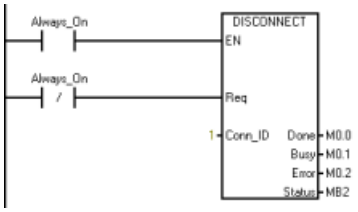
如果 Error 为 TRUE，则检查其原因，如果出现连接问题，可能需要更改状态。



程序段 31：状态“断开连接等待”...

在该状态等待直到断开连接操作完成。

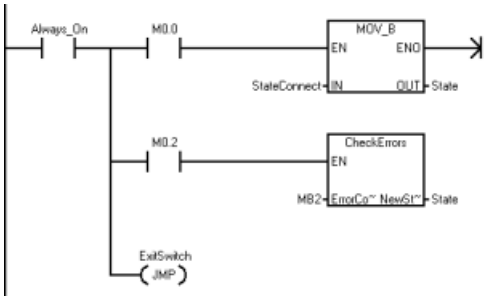
程序段 32：设置 Req = FALSE，调用 DISCONNECT 指令，以检查断开连接操作的状态。



程序段 33：如果 Done 为 TRUE，则表示已断开连接，之后在下次扫描时尝试再次建立连接。

如果 Busy 为 TRUE（通常是这种情况），则停留在“断开连接等待”状态。

如果出现错误，则检查其原因，可能需要根据错误代码更改状态。



程序段 34：退出开关。

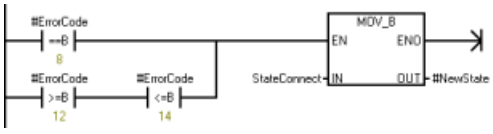
11.5.5.2 CheckErrors 子例程

CheckErrors 子例程用于检查开放式用户通信错误代码并确定程序是否需要更改状态。在主动伙伴（客户端）和被动伙伴（服务器）中使用相同的 CheckErrors 子例程。

程序段 1：如果未出现错误代码，程序会发生故障。断开并重新连接以纠正这个问题。

程序段 2：如果其中一个伙伴断开连接，则程序会显示错误代码 8、12、13 和 14。这些伙伴当前处于断开连接状态。

在所有这些情况下，都要与伙伴重新建立连接。设置状态为“连接”。

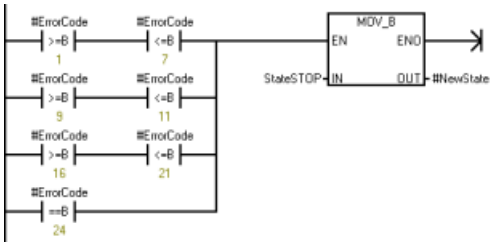


程序段 3：如果出现参数错误（错误代码 1 - 7），则停止该程序，因为该功能的某项输入存在组态错误。更改程序中的错误。

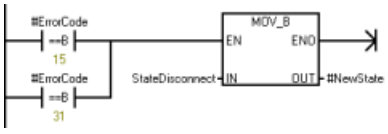
如果错误代码为 9（正在连接）、10（正在断开连接）、或 11（正在发送），则停止该程序因为状态机已损坏。设置该状态机停留在相关等待状态直到完成修复操作，且这些错误不会再次发生。

如果错误代码为 16 到 21 之间的数字，则这些错误代码会构成连接参数错误，且不会在此发生。如果出现这些错误，则说明存在故障，应停止程序向下执行。

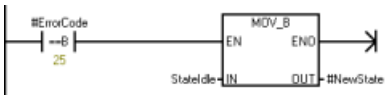
如果 SEND 和 DISCONNECT 指令返回错误 24，则表示当前没有待决操作。这可能意味着操作已完成，但是，不可能由于状态机的原因导致出现错误 24。考虑到该错误的存在，停止程序。



程序段 4：如果连接出现问题，则程序会返回错误 15 和 31。断开并重新连接以纠正这个问题。



程序段 5：如果 RECV 功能接收的数据量超出了缓冲区所支持容纳的数据量，则程序会返回错误 25。在我们的案例中，未出现错误继续执行程序。



11.5.5.3 主动伙伴变量表

下表列出了主动伙伴（客户端）程序的变量名、地址和注释。

变量	地址	注释
Always_On	SM0.0	始终接通
First_Scan_On	SM0.1	仅在首次扫描周期接通
State	VB0	
IdleTime	VD4	
IdleTimeStart	VD8	
RecvGoodCount	VD20	
RecvBadCount	VD24	
RecvStartTime	VD28	
RecvTime	VD32	

变量	地址	注释
FillPattern	VW12	
RecvLength	VW16	
SendLength	VW18	
SendBufferW	VW1000	
RecvBufferW	VW2000	
StateConnect	1	
StateConnectWait	2	
StateIdle	3	
StateIdleWait	4	
StateTransmit	5	
StateTransmitWait	6	
StateRecv	7	
StateRecvWait	8	
StateRecvCheck	9	
StateDisconnect	10	
StateDisconnectWait	11	
ExitSwitch	19	
RecvTimeout	100	接收超时的毫秒数
IdleTimeDelay	250	发送命令间隔的毫秒数

11.5.5.4 被动伙伴（服务器）

该程序用于实现简单状态机管理连接打开，接收消息，发送响应，以及处理错误。

状态机的流程是建立连接，之后可重复接收消息及发送响应。如果连接断开，状态机会响应被动连接请求。

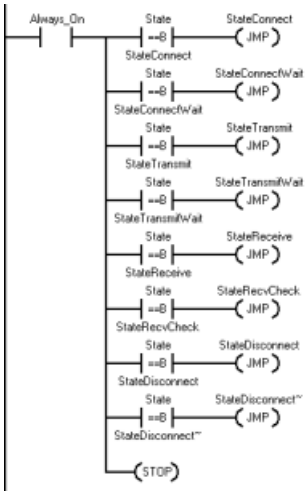
有关该程序的符号表，请参见“被动伙伴符号表”(页 665)。

程序段 1：第一次扫描时....

初始化状态变量以初始化连接。

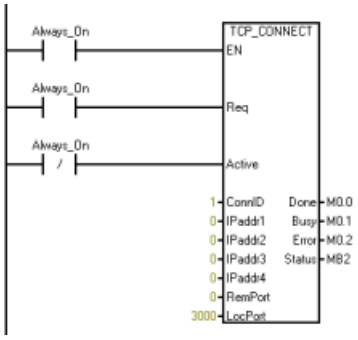
程序段 2：处理状态机...

确定状态机的当前状态并跳转到状态处理程序标记处。如果该状态非法，则 CPU 进入 STOP 模式。



程序段 3：状态连接...

程序段 4：启动连接过程。由于该连接端为服务器，所以设置 Active 输入为 FALSE。设置 IPaddr 输入为 0，由此服务器可接收从任意地址传输过来的连接。设置 RemPort 为 0，因为服务器连接操作无需用到该参数。设置 Req 输入为 TRUE，调用 TCP_CONNECT 指令以启动连接过程。

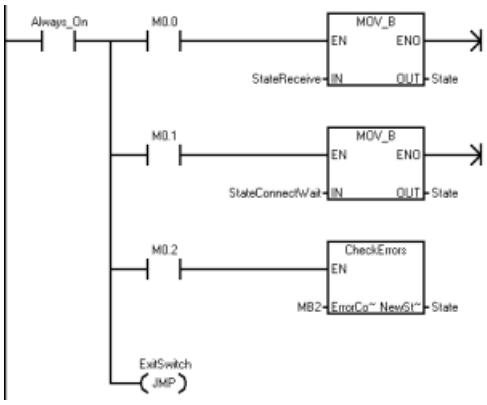


程序段 5：如果 Done 为 TRUE，则 CPU 已建立该连接，之后进入“空闲”状态。

如果 Busy 为 TRUE，则 CPU 进入“连接等待”状态以等待连接的建立。

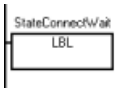
如果 Error 为 TRUE，则可能输入参数有误，检查输入参数以决定 CPU 接下来要进入哪个状态。

在所有情况下，都要先退出状态机才能进行该项扫描。该程序会继续进入下一项扫描的下一个状态。

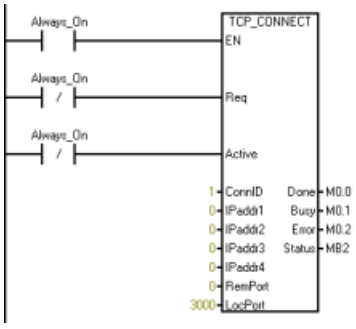


程序段 6：状态“连接等待”...

在该状态等待直到主动伙伴与该 CPU 创建连接。



程序段 7：通过设置 Req = FALSE 并使用与上文中相同的连接 ID (ConnID) 调用 TCP_CONNECT 指令。执行该项操作以检查连接状态。

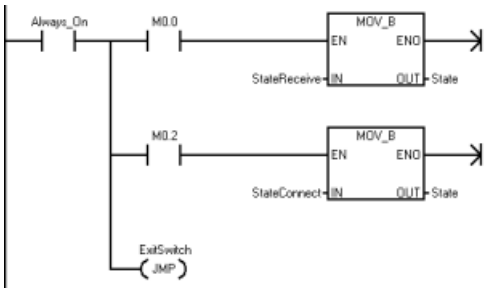


程序段 8：如果 Done 为 TRUE，则表示 CPU 已建立该连接，所以继续进入“空闲”状态。

如果 Busy 为 TRUE，则 CPU 停留在“连接等待”状态。由于该连接为被动连接，所以停留在“忙碌”状态直到主动伙伴连接到该 CPU。被动连接不会超时。

如果 Error 为 TRUE，则表示出现问题，返回到上一个操作，并在下次扫描时再次尝试建立连接。

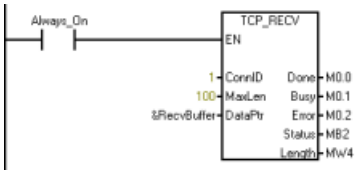
在所有情况下，都要先退出状态机才能进行该项扫描。该程序会继续进入下一项扫描的下一个状态。



程序段 9：状态接收...

停留在该状态直到服务器接收到某些数据。

程序段 10：调用 TCP_RECV 指令以获取 CPU 接收到的任意消息。

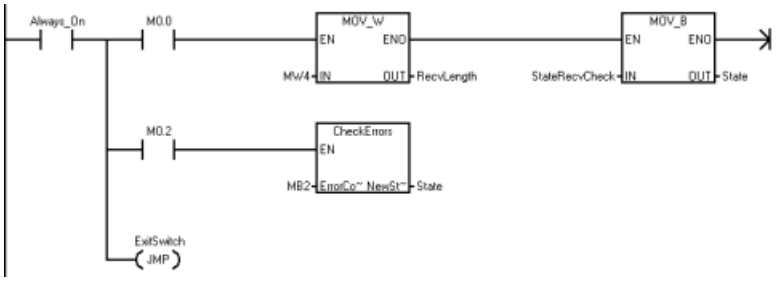


程序段 11：如果 Done 为 TRUE，则表示 CPU 已接收到新数据，之后进入“接收检查”状态。

如果 Busy 为 TRUE，则停留在“接收”状态直到接收到某些数据。

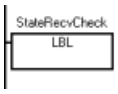
如果 Error 为 TRUE，则检查错误代码以确定接下来的操作。

在所有情况下，都要先退出状态机才能进行该项扫描。该程序会继续进入下一项扫描的下一个状态。

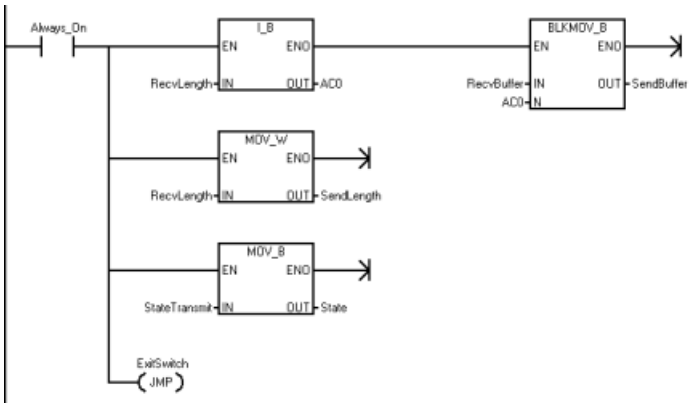


程序段 12：状态“接收检查”...

检查并处理接收到的数据。

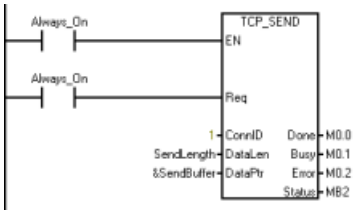


程序段 13：在该程序中，将数据传回给伙伴。将所有接收到的字节复制到发送缓冲区。更改为“传送”状态，之后退出程序，直到下一次扫描时再开启。



程序段 14：状态“传送”...

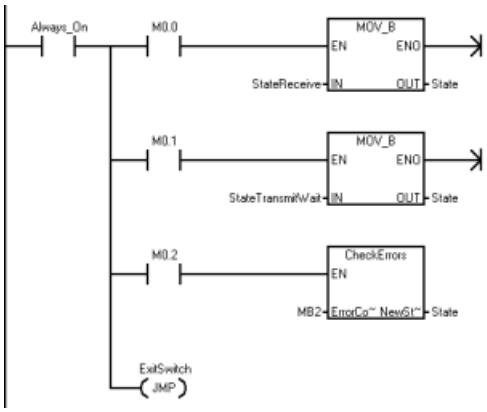
程序段 15：将数据发送回给伙伴。



程序段 16：如果 Done 为 TRUE，则发送操作已完成（几乎不会很快完成），之后进入下一次扫描的“接收”状态。

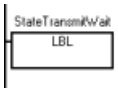
如果 Busy 为 TRUE（通常是这种情况），则进入“传送等待”状态以等待传送完成。

如果 Error 为 TRUE，则检查其原因，如果出现连接问题，可能需要更改状态。

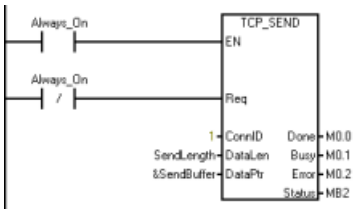


程序段 17：状态“传送等待”...

在该状态等待直到传送完成。



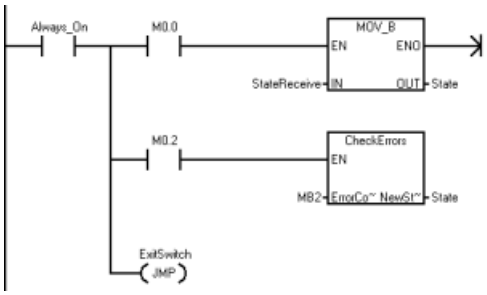
程序段 18：设置 Req = FALSE，调用 TCP_SEND 指令以确定发送完成的时间。确保使用初始化发送请求所用的发送长度和缓冲区指针。



程序段 19：如果 Done 为 TRUE，则发送已完成，之后进入“接收”状态。

如果 Busy 为 TRUE，则 CPU 停留在“传送等待”状态。

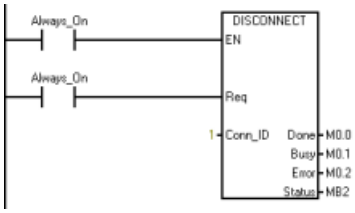
如果 Error 为 TRUE，则检查出现错误的原因，如果是连接出现问题，需要更改状态。



程序段 20：状态“断开连接”...

启动断开连接。

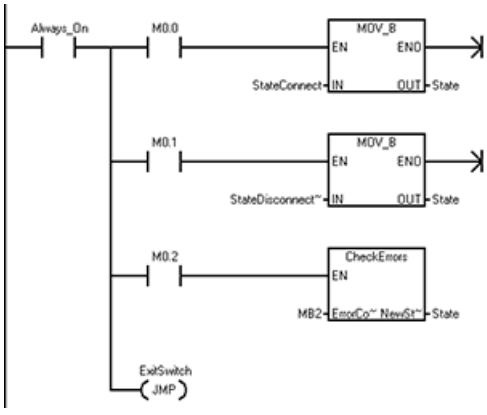
程序段 21：通过设置 Req = TRUE，调用 DISCONNECT 指令来启动与 ConnID 断开连接。



程序段 22：如果 Done 为 TRUE，则表示已断开连接（几乎不会发生），之后尝试再次建立连接。

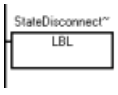
如果 Busy 为 TRUE（通常是这种情况），则进入“断开连接等待”状态。

如果 Error 为 TRUE，则检查其原因，如果出现连接问题，可能需要更改状态。

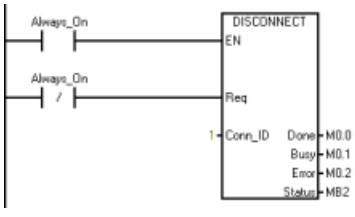


程序段 23：状态“断开连接等待”...

在该状态等待直到断开连接操作完成。

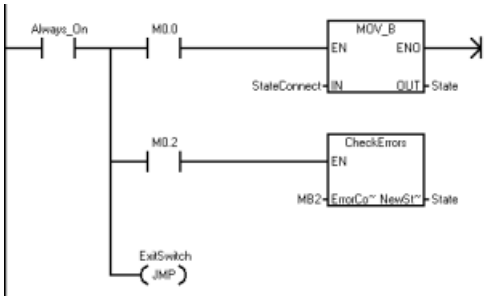


程序段 24：设置 Req = FALSE，调用 DISCONNECT 指令，以检查断开连接操作的状态。



程序段 25：如果 Done 为 TRUE，则表示已断开连接，之后在下次扫描时尝试再次建立连接。如果 Busy 为 TRUE（通常是这种情况），则停留在“断开连接等待”状态。

如果出现错误，则检查其原因，可能需要根据错误代码更改状态。



程序段 26：退出开关。

11.5.5.5 CheckErrors 子例程

CheckErrors 子例程 (页 656)用于检查开放式用户通信错误代码并确定程序是否需要更改状态。在主动伙伴（客户端）和被动伙伴（服务器）中使用相同的 CheckErrors 子例程。

11.5.5.6 被动伙伴变量表

下表列出了被动伙伴（服务器）程序的变量名、地址和注释。

变量	地址	注释
Always_On	SM0.0	始终接通
First_Scan_On	SM0.1	仅在首次扫描周期接通
State	VB0	
SendBuffer	VB1000	
RecvBuffer	VB2000	
RecvLength	VW16	
SendLength,	VW18	
SendBufferW	VW1000	
RecvBufferW	VW2000	

变量	地址	注释
StateConnect	1	
StateConnectWait	2	
StateTransmit	3	
StateTransmitWait	4	
StateReceive	5	
StateRecvCheck	6	
StateDisconnect	7	
StateDisconnectWait	8	
ExitSwitch	10	

11.6 PN Read Write Record 库

11.6.1 PN Read Write Record 特性

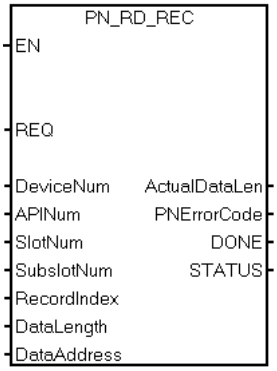
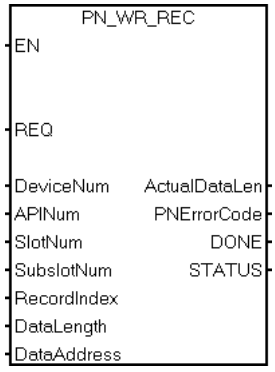
PN Read Write Record 库包括以下两个指令：

- PN_RD_REC：从任一连接的 PROFINET 设备读取数据记录
- PN_WR_REC：将数据记录写入任一连接的 PROFINET 设备。

11.6.2 PN Read Write Record 库的输入和输出接口

PN_RD_REC 和 PN_WR_REC 指令如下：

表格 11-26 PN_RD_REC 和 PN_WR_REC

LAD/FBD	STL	说明
	CALL PN_RD_REC、REQ、DeviceNum、APINum、SlotNum、SubslotNum、RecordIndex、DataLength、DataAddress、ActualDataLen、PnErrorCode、Done、STATUS	使用 PN_RD_REC 指令从 PROFINET 设备读取数据记录。
	CALL PN_WR_REC、REQ、DeviceNum、APINum、SlotNum、SubslotNum、RecordIndex BufferLength、DataAddress、ActualDataLen、PnErrorCode、Done、STATUS	使用 PN_WR_REC 指令将数据记录写入 PROFINET 设备。

PN_RD_REC 和 PN_WR_REC 指令的参数如下：

表格 11-27 PN_RD_REC 和 PN_WR_REC 指令的参数

参数和类型		数据类型	说明
REQ	IN	BOOL	REQ=1：传送数据记录
Device Number	IN	WORD **注：** 值的范围为 1 至 8。	设备编号、API 编号、插槽编号和子插槽编号用于寻址子模块。 可在 PROFINET 向导中找到设备编号、API 编号、插槽编号和子插槽编号。
API Number	IN	DWORD	
Slot Number	IN	WORD	
SubSlot Number	IN	WORD	
Record Index	Input	WORD	记录索引包括协议中或用户定义的记录索引。 有关协议中索引的详细信息，请参见 PROFINET IO（版本 2.3）的技术规范。
Data Length	Input	WORD	此参数是指缓冲区的字节数。缓冲区存储从设备读取或写入设备的数据记录。 取值范围：1 至 1024。
Data Address	Input	DWORD	缓冲区从设备读取或写入设备的地址。 **注：**如果缓冲区长度大于实际记录数据长度，则缓冲区包含所有记录数据。如果缓冲区长度小于实际记录数据长度，则缓冲区包含部分记录数据，并会发生错误。
Actual record data length	Output	WORD	此参数适用于指令 RDREC，并返回设备指定的实际数据长度。
PROFINET Error Code	Output	DWORD	PROFINET 协议定义的错误代码。0 = 无错误。 如果该值不为 0，请检查 PROFINET IO（版本 2.3）的技术规范中的特定错误代码。
DONE	OUT	BOOL	完成指令。
STATUS	OUT	BYTE	当前操作的状态。有关详细信息，请参见 为输入信号“STATUS”定义参数 (页 669)。

11.6.3 为输入信号“STATUS”定义参数

下表列出了“STATUS”的参数信息：

表格 11-28 STATUS

字节偏移	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
0	A [1]	E [2]	错误代码 [3]					

[1] A：1 = 正在进行请求

[2] E：1 = 出错

[3] 错误代码：系统错误代码。有关详细信息，请参见 PN Read Write Record 库的系统定义错误代码 (页 669)。

11.6.4 PN Read Write Record 库的系统定义错误代码

错误代码如下：

错误代码	说明
0	无错误。
1	数据长度参数为 0 或大于支持的最大长度（1024 字节）。
2	数据缓冲区不在 I、Q、M 或 V 存储区。
3	数据缓冲区不适合存储区。
4	表格与存储器不匹配。
5	设备编号无效，且不在范围内，范围 1 至 8。
6	实例不匹配：连接正忙于另一个实例，其设备编号、API 编号、插槽编号和子插槽编号与请求的实例相同，但缓冲区大小和数据地址不同。
7	PROFINET 设备未连接。
8	接收缓冲区的大小超过 1024 个字节。
9	调用序列无效。
10	参数无效（例如，超出范围）。
11	同时重新创建 AR。
12	RPC 报告超时错误。
13	RPC 报告通信错误。

错误代码	说明
14	IOD 的 RPC 服务器显示“正忙”（例如，稍后可重新调用）。
15	CLRPC 报告错误或 PDU 无法解析。
16	CM 响应正常，但出现 PROFINET 协议定义的 错误。
17	指令参数无效。
24	未启用 REQ。
25	缓冲区长度小于实际数据记录长度。
63	未知错误。

11.7 USS 协议库指令

11.7.1 USS 通信概述

11.7.1.1 USS 协议概述

STEP 7-Micro/WIN SMART V3 指令库包括专门设计用于通过 USS 协议与电机变频器进行通信的预组态子程序和中断例程，从而使控制 Siemens 变频器更加简便。可使用 USS 指令控制物理变频器和读/写变频器参数。

Siemens 设计了 USS 通信库，目的是为了支持 Siemens 的通用驱动，如 Siemens Micromaster 系列。Siemens 不希望使用 USS 通信库支持特殊用途的驱动器，如 V90 伺服驱动。V90 伺服驱动的控制接口不同于通用驱动的接口。为此，请勿将 USS 通信库用于 V90 伺服驱动。

您可在 STEP 7-Micro/WIN SMART V3 指令树的“库”(Libraries) 文件夹中找到这些指令。选择一条 USS 指令后，会自动添加一个或多个相关子例程和中断。

USS 协议库概述涉及以下主题：

- 使用 USS 协议的要求 (页 671)
- 计算与驱动器通信所需的时间 (页 672)

有关 USS 协议指令列表、错误代码及示例程序的信息，请参见“使用 USS 协议指令 (页 673)”。

说明

只可从主程序或中断例程中调用库函数，但不可同时从这两个程序中调用。

11.7.1.2 使用 USS 协议的要求

STEP 7-Micro/WIN SMART 指令库提供子例程、中断例程和指令来支持 USS 协议。USS 指令使用 S7-200 SMART CPU 中的下列资源：

- USS 协议是一种受中断驱动的应用程序。最差情况下，接收消息中断例程的执行最多需要 2.5 ms。在此期间，所有其他中断事件都需要排队，等待接收消息中断例程执行完毕后再进行处理。如果您的应用无法容许此类最糟情况下的延迟，则可能需要考虑采用其他解决方案来控制变频器。
- 初始化 USS 协议，使 S7-200 SMART CPU 端口专门用于 USS 通信。
  可使用 USS_INIT 指令为端口 0 、端口 1 或端口 2 选择 USS 或 PPI。（USS 是指用于 Siemens 变频器的 USS 协议。）当某个端口设置为使用 USS 协议与变频器进行通信后，就不能再将该端口用于任何其他用途，包括与 HMI 进行通信。STEP 7-Micro/WIN SMART V3 可以使用第二个通信端口在 USS 协议运行时监视控制程序。
- USS 指令会影响与所分配端口上自由端口通信相关的所有 SM 位置。
- USS 子例程和中断例程已存储在程序中。USS 指令最多将您的程序所需的存储器数量增加至 3050 个字节。根据所使用的特定 USS 指令，这些指令的支持例程可使控制程序的存储空间开销至少增加 2150 字节，最多增加 3050 字节。
- USS 指令的变量需要 402 字节的 V 存储器。该存储区的起始地址由用户指定，保留用于 USS 变量。
- 某些 USS 指令还需要 16 字节的通信缓冲区。作为指令的参数，需要为该缓存区提供一个 V 区的起始地址。建议您为 USS 指令的每个实例都指定一个唯一的缓冲区。
- 执行计算时，USS 指令使用累加器 AC0 至 AC3。还可以在程序中使用累加器，但累加器中的数值将由 USS 指令改动。
- USS 指令不能用在中断例程中。

11.7.1.3 计算与驱动器通信所需的时间

与驱动器之间的通信与 S7-200 SMART CPU 扫描不同步。在完成一个驱动器通信事务之前，CPU 通常已完成了多次扫描。以下因素有助于确定所需时间：

- 现有驱动器数量
- 波特率
- CPU 的扫描时间

当使用参数访问指令时，有些驱动器需要的延迟时间比较长。参数访问所需的时间取决于驱动器类型以及正在访问的参数。

USS_INIT 指令分配端口 0 使用 USS 协议之后，CPU 会以下表所示时间间隔定期轮询所有处于激活状态的驱动器。为此，必须设置各驱动器的超时参数：

表格 11-29 通信时间

波特率	激活驱动器的轮询时间间隔 （未激活任何参数访问指令）
1200	240 ms（最大）乘以驱动器数目
2400	130 ms（最大）乘以驱动器数目
4800	75 ms（最大）乘以驱动器数目
9600	50 ms（最大）乘以驱动器数目
19200	35 ms（最大）乘以驱动器数目
38400	30 ms（最大）乘以驱动器数目
57600	25 ms（最大）乘以驱动器数目
115200	25 ms（最大）乘以驱动器数目

11.7.2 USS 程序指令

11.7.2.1 使用 USS 协议指令

步骤

要在 S7-200 SMART 程序中使用 USS 协议指令，请按以下步骤操作：

1. 在程序中插入 USS_INIT 指令，并仅执行 USS_INIT 指令一个扫描周期。 可以使用 USS_INIT 指令初始化或更改 USS 协议通信参数。
   插入 USS_INIT 指令时，会在程序中自动添加若干隐藏的子例程和中断例程。
2. 只能在程序中为每台激活变频器放置一条 USS_CTRL 指令。
   可以根据需要增加任意数量的 USS_RPM_x 和 USS_WPM_x 指令，但某一时间只能有一条指令处于激活状态。
3. 在“文件”(File) 菜单功能区的“库”(Libraries) 区域中，单击“存储器”(Memory) 按钮 库管理器，指定 USS 库所需的 V 存储器的起始地址。 或者，也可在项目树中右键单击“程序块”(Program Block) 节点，并从上下文菜单中选择“库存储器”(Library Memory)。
4. 组态变频器参数，使之与程序中使用的波特率和地址相匹配。
5. 用通信电缆连接 S7-200 SMART CPU 与变频器。
   确保与变频器连接的所有控制设备（例如 S7-200 SMART CPU）均用短粗电缆连接到变频器使用的接地点或星点。

> ⚠ **小心**
>
> **防止意外电流**
>
> 互连参考电位不同的设备可能导致意外电流从互连电缆中流过。 这些意外电流可能导致通信错误或设备损坏。
>
> 确保所有用通信电缆连接的设备均具有共同的电路参考点或已隔离，以避免产生意外电流。
>
> 必须将屏蔽层连接到外壳地或 9 针连接器的引脚 1。 建议将变频器上的 2-0V 端子连接到外壳地。

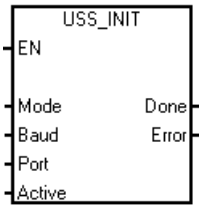
USS 协议指令由以下部分组成：

- USS_INIT (页 674)
- USS_CTRL (页 676)
- USS_RPM_X (页 679)
- USS_WPM_x (页 682)

本部分还将讨论 USS 协议程序示例 (页 684)和 USS 协议错误代码 (页 686)列表。

11.7.2.2 USS_INIT 指令

表格 11-30 USS_INIT 指令

LAD/FBD	STL	描述
	**CALL USS_INIT, Mode, Baud, Port, Active, Done, Error**	USS_INIT 指令用于启用和初始化或禁用 Siemens 变频器通信。在使用任何其他 USS 指令之前，必须执行 USS_INIT 指令且无错。该指令完成后，立即置位“完成”(Done) 位，然后继续执行下一条指令。

“EN”输入接通时，在每次扫描时均执行该指令。

每次通信状态变化时执行 USS_INIT 指令一次。使用边缘检测指令使“EN”输入以脉冲方式接通。要更改初始化参数，请执行新的 USS_INIT 指令。

表格 11-31 USS_INIT 指令的参数

输入/输出	数据类型	操作数
Mode、Port	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*AC、*LD
Baud、Active	DWORD	VD、ID、QD、MD、SD、SMD、LD、常数、AC、*VD、*AC、*LD
Done	BOOL	I、Q、M、S、SM、T、C、V、L
Error	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD

表格 11-32 USS_INIT 参数说明

参数	描述
Mode	此值用于选择通信协议： • 输入值为 1 时，将端口分配给 USS 协议并启用该协议。 • 输入值为 0 时，将端口分配给 PPI 协议并禁用 USS 协议。
Baud	将波特率设置为 1200、2400、4800、9600、19200、38400、57600 或 115200
Port	设置物理通信端口（0 = CPU 中集成的 RS485，1 = 首个 CM01 信号板选件上的 RS485 或 RS232，2 = 第二个 CM01 信号板选件上的 RS485 或 RS232）
Active	指示激活的变频器。有些变频器仅支持地址 0 至 30。当此值设置为 0 时，所有变频器都会激活。

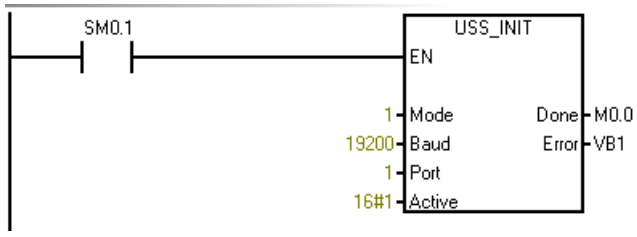
参数	描述
Done	当 USS_INIT 指令完成后接通
Error	该输出字节包含指令执行的结果。USS 协议执行错误代码 (页 686)定义了执行该指令产生的错误状况。

表格 11-33　激活变频器参数的格式

MSB LSB 31 30 29 28 3 2 1 0 D31 D30 D29 D2 D1 D0	该图显示了激活变频器输入的说明和格式。系统在后台自动轮询标记为“激活”(Active)的变频器，以控制变频器、收集状态并预防变频器发生串行链路超时。
• D0（变频器 0 激活位）： – 0 - 变频器未激活 – 1 - 变频器已激活 • D1（变频器 1 激活位）： – 0 - 变频器未激活 – 1 - 变频器已激活 • …	

要计算状态轮询和可能因执行指令而导致的错误条件之间的时间，请参见 USS 协议执行错误代码 (页 686)。

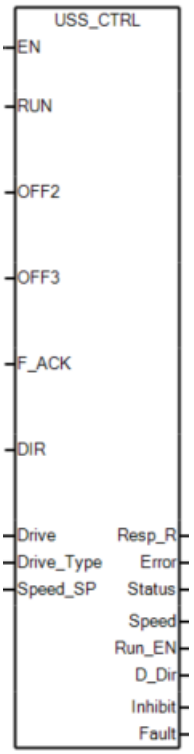
表格 11-34　USS_INIT 示例程序

Network 1	Network 1
	**LD SM0.1** **CALL USS_INIT, 1, 19200, 1, 16#1, M0.0, VB1**

如需 USS 协议指令列表以及错误代码和示例程序，请参见“使用 USS 协议指令”(页 673)。

11.7.2.3 USS_CTRL 指令

表格 11-35 USS_CTRL 指令

LAD/FBD	STL	说明
	**CALL USS_CTRL, RUN, OFF2, OFF3, F_ACK, DIR, Drive, Type, Speed_SP, Resp_R, Error, Status, Speed, Run_EN, D_Dir, Inhibit, Fault**	USS_CTRL 指令用于控制激活的 Siemens 变频器。 USS_CTRL 指令将所选命令放置到通信缓冲区中，如果已在 USS_INIT 指令的“激活”(Active) 参数中选择变频器，该命令随后将发送到这一被寻址的变频器（“变频器”参数）。

每台变频器只能分配一条 USS_CTRL 指令。

有些变频器仅以正值形式报告速度。 如果速度为负值，变频器将速度报告为正值，但取反“D_Dir”（方向）位。

“EN”位必须接通才能启用 USS_CTRL 指令。 该指令应始终启用。

表格 11-36 USS_CTRL 指令的参数

输入/输出	数据类型	操作数
RUN、OFF 2、OFF 3、F_ACK、DIR	BOOL	I、Q、M、S、SM、T、C、V、L、能流
Resp_R、Run_EN、D_Dir、Inhibit、Fault	BOOL	I、Q、M、S、SM、T、C、V、L
Drive、Type	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD、常数
Error	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD

输入/输出	数据类型	操作数
状态	WORD	VW、T、C、IW、QW、SW、MW、SMW、LW、AC、AQW、*VD、*AC、*LD
Speed_SP	REAL	VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*AC、*LD、常数
Speed	REAL	VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*AC、*LD

RUN 参数

RUN (RUN/STOP) 指示变频器是接通 (1) 还是关闭 (0)。 当“运行”(RUN) 位接通时，变频器收到一条命令，以指定速度和方向开始运行。 为使变频器运行，必须符合以下条件：

- 变频器在 USS_INIT 中必须选为“激活”(Active)。
- “OFF2”和“OFF3”必须设置为 0。
- “故障”(Fault) 和“禁止”(Inhibit) 必须为 0。

当“RUN”关闭时，会向变频器发送一条命令，将速度降低，直至电机停止：

- “OFF2”位用于允许变频器自然停止。
- “OFF3”位用于命令变频器快速停止。

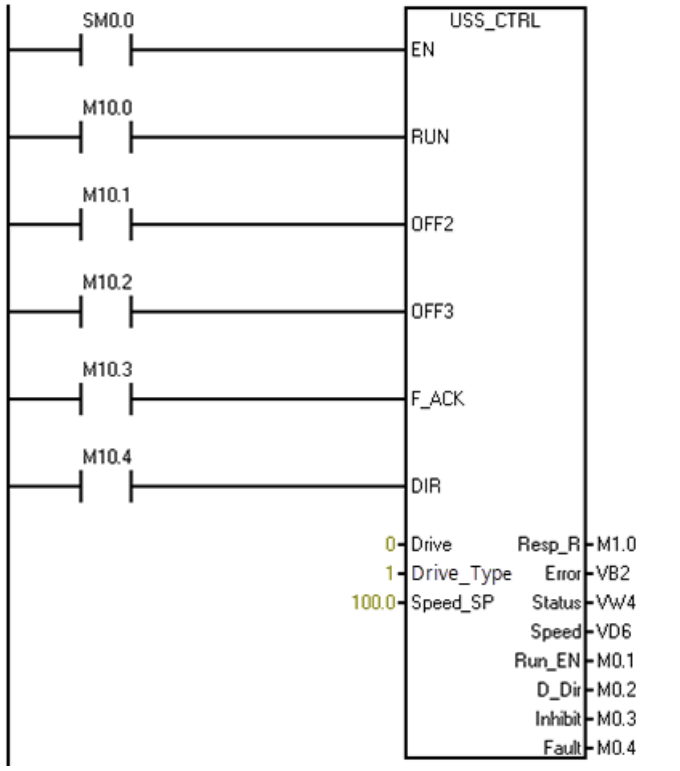
Resp_R 参数

“Resp_R”（收到响应）位确认来自变频器的响应。 系统轮询所有激活的变频器以获取最新的变频器状态信息。 每次 CPU 收到来自变频器的响应时，“Resp_R”位将接通一个扫描周期，并且以下所有值将更新。

参数	说明
F_ACK （故障确认）	确认变频器发生故障的位。 当“F_ACK”从 0 变为 1 时，变频器将清除故障（“故障”(Fault) 位）。
DIR （方向）	指示变频器移动方向的位。
Drive （驱动器地址）	表示接收 USS_CTRL 命令的变频器地址的输入。 有效地址： 0 到 31
Type （驱动器类型）	选择变频器类型的输入

参数	说明
Speed_SP （速度设定值）	变频器速度，该速度是全速的一个百分数： • “Speed_SP”为负值将导致变频器调转其旋转方向。 • 范围： -200.0% 到 200.0%
Error	字节，其中包含对变频器的最新通信请求的结果。 USS 协议执行错误代码(页 686)定义了执行该指令产生的错误状况。
状态	变频器返回的状态字的原始值。 下图显示了标准状态字和主反馈的状态位。
速度	变频器速度，该速度是全速的一个百分数。 范围： -200.0% 到 200.0%
Run_EN （RUN 使能）	指示变频器运行状况： • 运行中 (1) • 已停止 (0)
D_Dir	指示变频器的旋转方向
禁止	指示变频器上“禁止”(Inhibit) 位的状态： • 0: 未禁止 • 1: 已禁止 要清除“禁止”(Inhibit) 位，下列位必须断开： • “Fault” • “RUN” • “OFF2” • “OFF3”
故障	指示“故障”(Fault) 位的状态： • 0: 无故障 • 1: 故障 变频器显示故障代码。（有关所用变频器的信息，请参见用户手册）。 要清零“故障”(Fault) 位，应消除故障原因并接通“F_ACK”位。

表格 11-37 USS_CTRL 示例程序



要在 LAD 或 FBD 中显示：

```
程序段 1 // 驱动器 0 的控制功能框
LD SM0.0
= L60.0
LD M10.0
= L63.7
LD M10.1
= L63.6
LD M10.2
= L63.5
LD M10.3
= L63.4
LD M10.4
= L63.3
LD L60.0
CALL USS_CTRL, L63.7, L63.6, L63.5, L63.4,
L63.3, 0, 1, 100.0, M1.0, VB2, VW4, VD6,
M0.1, M0.2, M0.3, M0.4
```

仅在 STL 中显示：

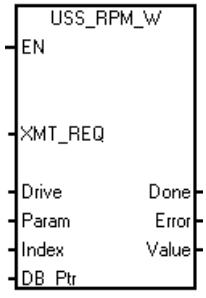
```
程序段 1 // 驱动器 0 的控制功能框
LD SM0.0
CALL USS_CTRL, M10.0, M10.1, M10.2, M10.3,
M10.4, 0, 1, 100.0, M1.0, VB2, VW4, VD6,
M0.1, M0.2, M0.3, M0.4
```

有关 USS 协议指令列表以及错误代码和示例程序，请参见“使用 USS 协议指令”(页 673)。

11.7.2.4 USS_RPM_x 指令

表格 11-38 USS_RPM_x 指令

LAD/FBD	STL	说明

	`CALL USS_RPM_W, XMIT_REQ, Drive, Param, Index, DB_Ptr, Done, Error, Value`
	`CALL USS_RPM_D, XMIT_REQ, Drive, Param, Index, DB_Ptr, Done, Error, Value`
	`CALL USS_RPM_R, XMIT_REQ, Drive, Param, Index, DB_Ptr, Done, Error, Value`
		USS 协议共有三条读取指令：
		• USS_RPM_W 指令用于读取无符号字参数。
		• USS_RPM_D 指令用于读取无符号双字参数。
		• USS_RPM_R 指令用于读取浮点参数。
		• 某一时间只能有一条读取 (USS_RPM_x) 或写入 (USS_WPM_x) 指令处于激活状态。

变频器确认接收到命令或出现错误条件时，USS_RPM_x 事务完成。 该进程等待响应时，逻辑扫描继续执行。

表格 11-39 USS_WPM_x 指令的有效操作数

输入/输出	数据类型	操作数
XMT_REQ	BOOL	I、Q、M、S、SM、T、C、V、L、受上升沿检测元素控制的能流
Drive	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD、常数
Param、Index	WORD	VW、IW、QW、MW、SW、SMW、LW、T、C、AC、AIW、*VD、*AC、*LD、常数
DB_Ptr	DWORD	&VB
Value	WORD DWORD、REAL	VW、IW、QW、MW、SW、SMW、LW、T、C、AC、AQW、*VD、*AC、*LD VD、ID、QD、MD、SD、SMD、LD、*VD、*AC、*LD
Done	BOOL	I、Q、M、S、SM、T、C、V、L
Error	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、 *VD、*AC、*LD

“EN”位必须接通才能启用对请求的发送，并在“Done”位置位之前保持接通，“Done”位置位表示过程完成。 例如，如果“XMT_REQ”输入接通，每次扫描时都会向变频器发送 USS_RPM_x 请求。 因此，“XMT_REQ”输入应通过沿检测元素以脉冲方式接通，该检测元素使得在“EN”输入的每次正跳变时发送一个请求。

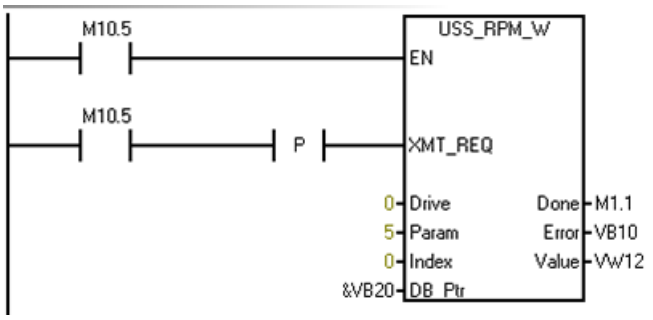
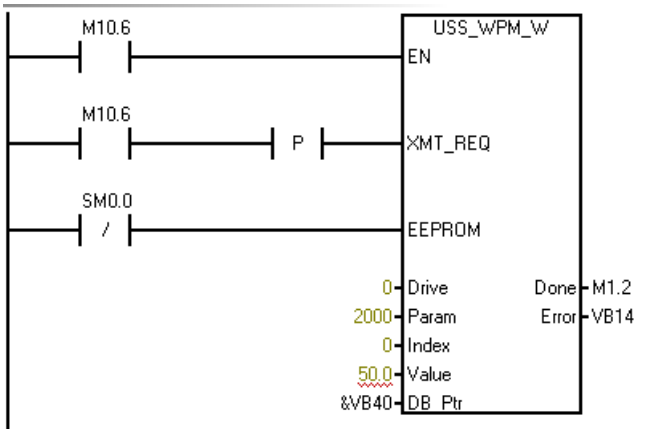
表格 11-40 USS_RPM_x 参数说明

参数	说明
XMT_REQ （传送请求）	如果接通，在每次扫描时会向变频器发送 USS_RPM_x 请求。
变频器	要接收 USS_RPM_x 命令的变频器地址。 各变频器的有效地址是 0 至 31。
Param	参数编号
索引	要读取的参数的索引值
DB_Ptr	必须为“DB_Ptr”输入提供 16 字节缓冲区的地址。 USS_RPM_x 指令使用该缓冲区存储发送到变频器的命令的结果。
Done	当 USS_RPM_x 指令完成后接通
Error	该输出字节包含指令执行的结果。 USS 协议执行错误代码 (页 686)定义了执行该指令产生的错误状况。
Value	参数值已恢复

USS_RPM_x 指令完成时，“完成”(Done) 输出接通，“错误”(Error) 输出字节和“值”(Value) 输出包含指令执行结果。“完成”(Done) 输出接通之前，“错误”(Error) 和“值”(Value) 输出无效。

USS_RPM_x 和 USS_WPM_x 示例程序

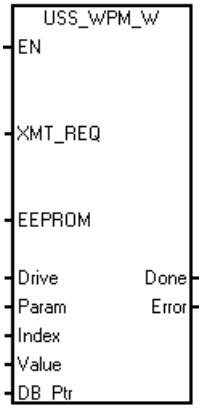
表格 11-41　USS_RPM_x 和 USS_WPM_x 示例程序

Network 1	Network 1
	LD M10.5 = L60.0 LD M10.5 EU = L63.7 LD L60.0 CALL USS_RPM_W, L63.7, 0, 5, 0, &VB20, M1.1, VB10, VW12
Network 2	**Network 2**
	LD M10.6 = L60.0 LD M10.6 EU = L63.7 LDN SM0.0 = L63.6 LD L60.0 CALL USS_WPM_W, L63.7, L63.6, 0, 2000, 0, 50.0, &VB40, M1.2, VB14

如需 USS 协议指令列表以及错误代码和示例程序，请参见“使用 USS 协议指令”（页 673）。

11.7.2.5 USS_WPM_x 指令

表格 11-42 USS_WPM_x 指令

LAD/FBD	STL	说明
	**CALL USS_WPM_W, XMT_REQ, EEPROM, Drive, Param, Index, Value, DB_Ptr, Done, Error** **CALL USS_WPM_D, XMT_REQ, EEPROM, Drive, Param, Index, Value, DB_Ptr, Done, Error** **CALL USS_WPM_R, XMT_REQ, EEPROM, Drive, Param, Index, Value, DB_Ptr, Done, Error**	USS 协议共有三种写入指令： • USS_WPM_W 指令用于写入无符号字参数。 • USS_WPM_D 指令用于写入无符号双字参数。 • USS_WPM_R 指令用于写入浮点参数。 某一时间只能有一条读取 (USS_RPM_x) 或写入 (USS_WPM_x) 指令处于激活状态。

变频器确认接收命令或出现错误条件时，USS_WPM_x 事务完成。 该进程等待响应时，逻辑扫描继续执行。

表格 11-43 USS_WPM_x 指令的有效操作数

输入/输出	数据类型	操作数
XMT_REQ	BOOL	I、Q、M、S、SM、T、C、V、L、受上升沿检测元素控制的能流
EEPROM	BOOL	I、Q、M、S、SM、T、C、V、L、能流
Drive	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD、常数
Param、Index	WORD	VW、IW、QW、MW、SW、SMW、LW、T、C、AC、AIW、*VD、*AC、*LD、常数
DB_Ptr	DWORD	&VB
Value	WORD DWORD、REAL	VW、IW、QW、MW、SW、SMW、LW、T、C、AC、AQW、*VD、*AC、*LD VD、ID、QD、MD、SD、SMD、LD、*VD、*AC、*LD
Done	BOOL	I、Q、M、S、SM、T、C、V、L
Error	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、 *VD、*AC、*LD

“EN”位必须接通才能启用对请求的发送，并在“Done”位置位之前保持接通，“Done”位置位表示过程完成。 例如，如果“XMT_REQ”输入接通，在每次扫描时向变频器发送 USS_WPM_x 请

求。 因此，“XMT_REQ”输入应通过沿检测元素以脉冲方式接通，该检测元素使得在“EN”输入的每次正跳变时发送一个请求。

表格 11-44 USS_WPM_x 参数说明

参数	说明
XMT_REQ（传送请求）	如果接通，在每次扫描时向变频器发送 USS_WPM_x 请求。
EEPROM	输入接通时可写入到变频器的 RAM 和 EEPROM，关闭时只能写入到 RAM。
变频器	USS_WPM_x 命令要发送的变频器地址。 各变频器的有效地址是 0 至 31。
Param	参数编号
索引	要写入的参数索引值
Value	要写入到变频器 RAM 的参数值。
DB_Ptr	必须为“DB_Ptr”输入提供 16 字节缓冲区的地址。 USS_RPM_x 指令使用该缓冲区存储发送到变频器的命令的结果。
Done	当 USS_RPM_x 指令完成后接通
Error	该输出字节包含指令执行的结果。 USS 协议执行错误代码 (页 686)定义了执行该指令产生的错误状况。

USS_WPM_x 指令完成时，“完成”(Done) 输出接通，“错误”(Error) 输出字节包含指令执行结果。 直到“完成”(Done) 输出接通，“错误”(Error) 输出才有效。

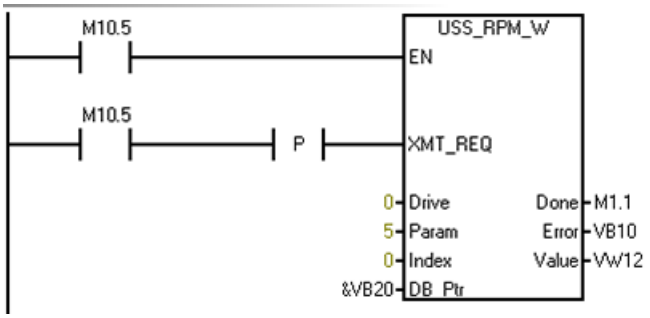
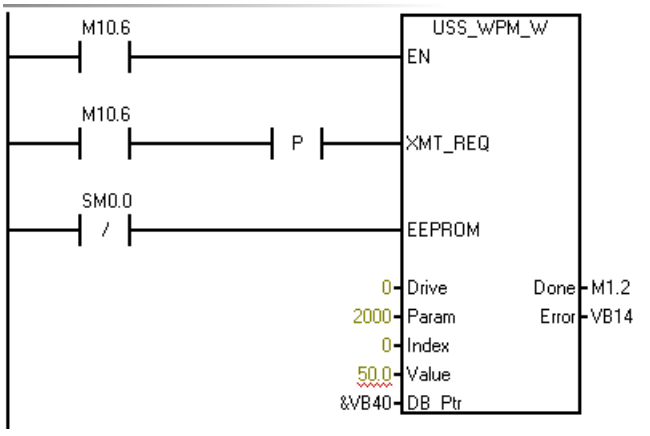
EEPROM

“EEPROM”输入接通时，该指令将数据写入到变频器的 RAM 和 EEPROM。 该输入关闭时，该指令仅将数据写入到变频器的 RAM。

注意
不要超出 EEPROM 的最大写入周期数 使用 USS_WPM_x 指令更新存储在变频器 EEPROM 中的参数集时，必须确保未超出可对 EEPROM 执行写入操作的最大次数（约为 50,000）。 超出最大写入周期数将导致存储数据损坏以及后续数据丢失，进而可能会造成财产损失。 读取周期数没有限制。 不要超出 EEPROM 的最大写入周期数。

USS_RPM_x 和 USS_WPM_x 示例程序

表格 11-45　USS_RPM_x 和 USS_WPM_x 示例程序

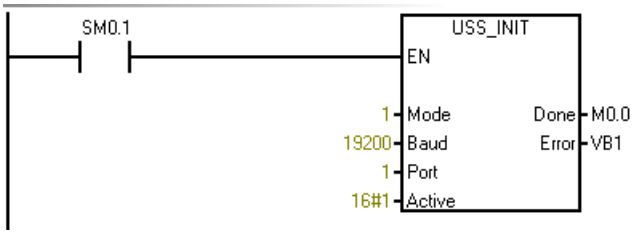
Network 1	Network 1
	LD M10.5 = L60.0 LD M10.5 EU = L63.7 LD L60.0 CALL USS_RPM_W, L63.7, 0, 5, 0, &VB20, M1.1, VB10, VW12
Network 2	**Network 2**
	LD M10.6 = L60.0 LD M10.6 EU = L63.7 LDN SM0.0 = L63.6 LD L60.0 CALL USS_WPM_W, L63.7, L63.6, 0, 2000, 0, 50.0, &VB40, M1.2, VB14

如需 USS 协议指令列表以及错误代码和示例程序，请参见“使用 USS 协议指令”(页 673)。

11.7.2.6　USS 协议示例程序

表格 11-46　USS 示例程序

Network 1	Network 1： 初始化 USS 协议：在第一次扫描中，为端口 1 启用 USS 协议，使变频器地址“0”激活，将波特率设置为 19200。

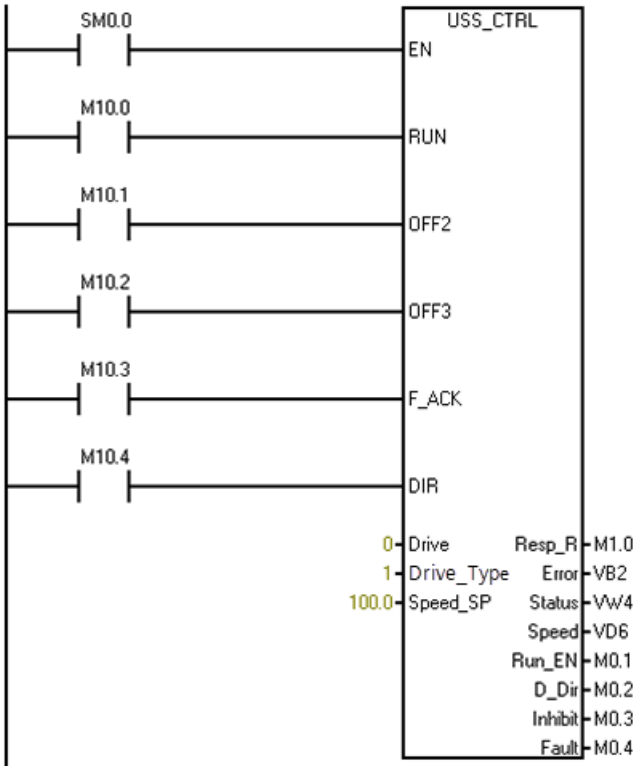



```
LD SM0.1
CALL USS_INIT, 1, 19200, 16#00000001,
Q0.0, VB1
```

Network 2

程序段 2：

控制变频器 0 的参数




```
LD SM0.0
CALL USS_CTRL, M10.0, M10.1, M10.2,
M10.3, M10.4, 0, 1, 100.0, M1.0, VB2,
VW4, VD6, M0.1, M0.2, M0.3, M0.4
```

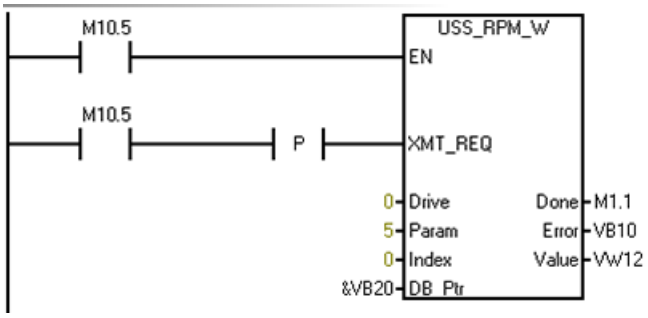
Network 3

程序段 3：

从变频器 0 读取字参数。

读取参数 5、索引 0：

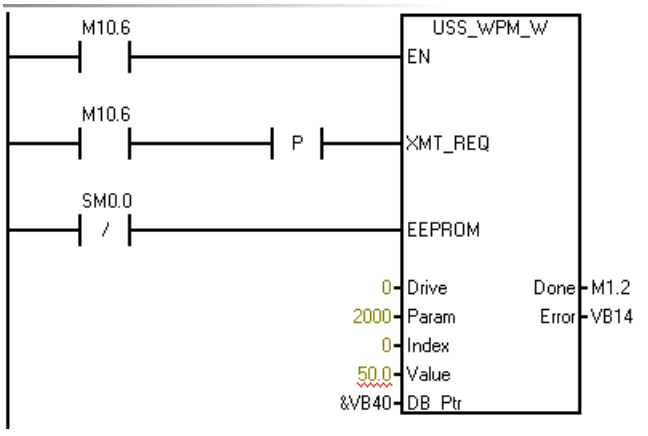
1. 将 M10.5 的状态保存到临时位置，以便在 LAD 中显示该程序段。
2. 将 I0.5 的上升边缘脉冲保存到一个临时 L 位置，以便将其传送至子例程。




```
LD M10.5
= L60.0
LD M10.5
EU
= L63.7
LD L60.0
CALL USS_RPM_W, L63.7, 0, 5, 0, &VB20,
M1.1, VB10, VW12
```

Network 4

程序段 4：
将字参数写入到变频器 0。写入参数 2000，索引 0。
注意： 此 STL 代码无法编译为 LAD 或 FBD。




```
LD M10.6
= L60.0
LD M10.6
EU
= L63.7
LDN SM0.0
= L63.6
LD L60.0
CALL USS_WPM_R, L63.7, L63.6, 0, 2000, 0,
50.0, &VB40, M1.2, VB14
```

如需 USS 协议指令列表以及错误代码和示例程序，请参见“使用 USS 协议指令”(页 673)。

11.7.3 USS 协议执行错误代码

表格 11-47 USS 协议执行错误代码

错误代码	描述
0	无错误
1	变频器无响应。
2	检测到来自变频器的响应存在检验和错误。
3	检测到来自变频器的响应存在奇偶校验错误。
4	用户程序的干扰导致错误。

错误代码	描述
5	尝试非法命令。
6	提供的变频器地址非法。
7	通信端口没有设置为用于 USS 协议通信。
8	通信端口正在忙于处理指令。
9	变频器速度输入超出范围。
10	变频器响应长度不正确。
11	变频器响应的第一个字符不正确。
12	变频器响应中的长度字符不受 USS 指令支持。
13	响应了错误的变频器。
14	提供的 DB_Ptr 地址不正确。
15	提供的参数编号不正确。
16	选择的协议无效。
17	USS 激活；不允许更改。
18	指定的波特率非法。
19	无通信：变频器未激活。
20	变频器响应中的参数或值不正确或包含错误代码。
21	返回一个双字值，而不是请求的字值。
22	返回一个字值，而不是请求的双字值。
23	端口号无效
24	信号板 (SB) 端口 1 缺失或未组态。
25	信号板 (SB) 端口 2 缺失或未组态。

如需 USS 协议指令列表以及错误代码和示例程序，请参见“使用 USS 协议指令”(页 673)。

11.8 SINAMICS 库

11.8.1 SINAMICS 库特性

SINAMICS 库包含预组态的子程序，更易控制驱动器。可使用 SINAMICS 库控制物理驱动器的位置和速度、读取或修改驱动参数。

STEP 7-Micro/WIN SMART 提供以下两组 SINAMICS 库指令：

- SINAMICS_Control：
  - SINA_POS (页 689)：通过 8 种不同的操作模式控制驱动器位置
  - SINA_SPEED (页 730)：控制驱动器速度
- SINAMICS_Parameter：
  SINA_PARA_S (页 737)：读取驱动参数或修改驱动参数

打开项目树下“指令”(Instruction) 文件夹的 Libraries 文件夹访问 SINAMICS 库指令。当将 SINAMICS 库指令置入程序时，STEP 7-Micro/WIN SMART V3 会将一个或多个关联子程序置入项目。

SINAMICS_Control

SINAMICS_Control 使用下列程序实体：

- 2 个子程序：
  - SINA_POS
  - SINA_SPEED
- 3867 字节程序空间
- V 存储器的 188 字节模块用于指令符号

SINAMICS_Parameter

SINAMICS_Parameter 使用下列程序实体：

- 4 个子程序：
  - PN_RD_REC_PARA_S
  - PN_WR_REC_PARA_S
  - SINA_PARA_S
  - ERROR_HANDLER
- 5050 字节程序空间
- V 存储器的 1314 字节模块用于指令符号

11.8.2 SINA_POS 指令

第三代 EPOS 不仅支持 LU（长度单位），还支持物理单位。EPOS 位置、速度、加速度和加加速度指的是负载侧（而非电机侧）。在第二代中，所有位置都表示为 32 位整数值，而在第三代中，所有位置都表示为 32 位浮点值。因此，在第三代中，LU 也有小数位，但在不损失精度的情况下最大位置范围会更小。32 位浮点值的有效分辨率为 24 位。

“SIEMENS 111 报文”(SIEMENS telegram 111) 是 32 位整数值的解决方案，而“SIEMENS 112 报文”(SIEMENS telegram 112) 是 32 位浮点值的解决方案。二代产品中，只可使用 SIEMENS 111 报文，而 SINA_POS 功能块用于控制驱动器。三代产品同样支持 SIEMENS 111 报文，也支持使用 SINA_POS 控制驱动器。但三代产品中，还可使用 SIEMENS 112 报文，并开发了一个用于控制的新功能块 SINA_POS_112。

11.8.2.1 使用 SINA_POS 指令的前提

使用 SINA_POS 指令的前提如下：

- SINAMICS V90 PN 驱动器和伺服电机已准备就绪。
- 驱动器和 S7-200 SMART CPU 已连接 PROFINET 网络。
- V-assistant 软件已与 SINAMICS V90 PN 连接。有关详细信息，请参阅 SINAMICS V-ASSISTANT 在线帮助 (https://support.industry.siemens.com/cs/cn/zh/view/109738316)。可在 SIEMENS 工业在线支持网站中下载 SINAMICS V-ASSISTANT 软件 (https://support.industry.siemens.com/cs/cn/en/view/109738387)和 SINAMICS V90：PROFINET GSD 文件 (https://support.industry.siemens.com/cs/cn/en/view/109737269)。

通过 V-assistant 组态 SINAMICS V90 PN 参数的步骤

通过 V-assistant 组态 SINAMICS V90 PN 参数的步骤如下：

1. 启动 V-assistant 软件。
2. 单击“在线”(Online) 选择工作模式。
3. 单击已连接的驱动器，单击“确认”(OK) 按钮。

4. 单击导航树的“选择驱动”(Select drive)，然后选择“控制模式”(Control Mode) 字段的“基本定位器控制 (EPOS)”(Basic positioner control (EPOS))。

5. 单击导航树的“设置 PROFINET” (Set PROFINET)，然后单击“选择报文”(Select telegram)。
6. 在“选择报文”(Selection of telegram) 字段，选择“西门子 111 报文”(SIEMENS telegram 111) 作为当前报文。SINA_POS 指令仅支持 SIEMENS 报文 111。

说明

在硬件网络拓扑中组态 PROFINET 网络时，确保报文与此步骤中的报文一致。

7. 单击导航树的“配置网络”(Configure network)。
8. 在“PN 站名”(Name of PN station) 字段定义 PN 站名称。

说明

在硬件网络拓扑中组态 PROFINET 网络时，确保设备名称与此步骤中的 PN 站名称一致。

9. 单击“保存并激活”(Save and active) 按钮。

结果：驱动器自动重启且组态的参数生效。

图 11-2　硬件网络拓扑中 PROFINET 网络组态示例

11.8.2.2　SINA_POS 指令的输入和输出接口

SINA_POS 指令可用于控制和设置驱动器的位置。

表格 11-48　SINA_POS 指令

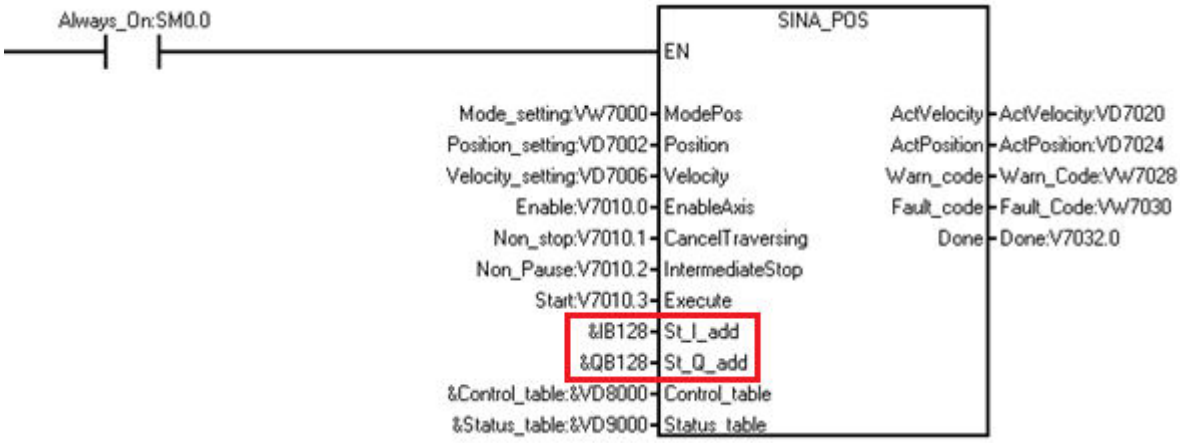
LAD/FBD	STL	描述
SINA_POS EN ModePos　ActVelocity Position　ActPosition Velocity　Warn_code EnableAxis　Fault_code CancelTraversing　Done IntermediateStop Execute St_I_add St_Q_add Control_table Status_table	CALL SINA_POS、ModePos、Position、Velocity、EnableAxis、CancelTraversing、IntermediateStop、Execute、St_I_add、St_Q_add、Control_table、Status_table、ActVelocity、ActPosition、Warn_code、Fault_code、Done	SINA_POS 指令可用于控制驱动器移动的位置。

说明

对于四个输入参数“St_I_add”、“St_Q_add”、“Control_table”和“Status_table”，寻址指令操作数模式为间接寻址。

必须在输入操作数的开头输入 & 符号并确保偏移量与 PROFINET 向导中的一致。

可以参照相对运动的示例：



表格 11-49 SINA_POS 指令的参数

参数和类型		数据类型	描述
ModePos	IN	INT	操作模式： 1 = 相对运动 2 = 绝对运动 3 = 恒速运动 4 = 主动回零 5 = 设置参考点 6 = 运行程序段 0 – 15 7 = 点动模式 8 = 增量点动 9 = 编码器调整（仅限 112）
Position	IN	111： DINT	直接设定值输入/MDI 模式的位置设定值 [LU] 或运行程序段模式的运行程序段编号。（默认值 = 0）
		112： REAL	用于 MDI 的物理单位位置设定值或用于操作模式运行程序段的运行时钟编号
Velocity	IN	111： DINT	MDI 模式的速度 [LU/分] （默认值 = 0 [1000 LU/分]）
		112： REAL	MDI 操作模式的物理单位速度设定值
EnableAxis	IN	BOOL	开关命令：0 = 关，1 = 开
CancelTraversing	IN	BOOL	0 = 拒绝激活运行任务 1 = 不拒绝（默认值）
IntermediateStop	IN	BOOL	0 = 激活运行命令中断 1 = 无中断（默认值）
Execute	IN	BOOL	激活运行任务/设定值接受/激活参考函数。
St_I_add	IN	DWORD	PROFINET IO I 存储区起始地址的指针。例如 &IB128。
St_Q_add	IN	DWORD	PROFINET IO Q 存储区起始地址的指针。例如 &QB128。
Control_table	IN	DWORD	control_table 起始地址的指针。例如 &VD8000。
Status_table	IN	DWORD	Status_table 起始地址的指针。例如 &VD9000。

参数和类型		数据类型	描述
ActVelocity	OUT	111：DWORD	实际速度
		112：REAL	实际速度 (MU)
ActPosition	OUT	111：DWORD	实际位置 (LU)
		112：REAL	实际位置 (MU)
Warn_code	OUT	WORD	来自 V90 的警告代码信息。有关详细信息，请参见 *SINAMICS V90、SIMOTICS S-1FL6 操作说明*。
Fault_code	OUT	WORD	来自 V90 的故障代码信息。有关详细信息，请参见 *SINAMICS V90、SIMOTICS S-1FL6 操作说明*。
Done	OUT	BOOL	当操作模式为相对运动或绝对运动时达到目标位置。

“Control_table”参数的定义

“Control_table”的参数如下：

表格 11-50　111 报文的 Control_table 参数

字节偏移	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
0	预留	预留	AckError [1]	FlyRef [2]	Jog2 [3]	Jog1 [4]	Negative [5]	Positive [6]
1	预留							
2	OverV [7]							
3								
4	OverAcc [8]							
5								
6	OverDec [9]							
7								
8	ConfigEpos [10]							
9								
10								
11								

表格 11-51　112 报文的 Control_table 参数

字节偏移	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
0	预留	预留	AckError [1]	FlyRef [2]	Jog2 [3]	Jog1 [4]	Negative [5]	Positive [6]
1	预留							
2	OverV [7]							
3								
4	Acc							
5								
6								
7								
8	Dec							
9								
10								
11								
12	fixStopTorque							
13								
14								
15								
16	refCoordinate							
17								
18								
19								
20	ConfigEpos [10]							
21								
22								
23								

[1] AckError：确认错误。（1 = 确认错误有效，0 = 确认错误无效）

[2] FlyRef：被动回零选择（1 = 激活参考点设置，0 = 未激活参考点设置）

[3] Jog2：点动信号源 2（1 = 激活正向点动，0 = 未激活正向点动）

[4] Jog1：点动信号源 1（1 = 激活负向点动，0 = 未激活负向点动）

[5] Negative：负向（1 = 激活负向旋转，0 = 未激活负向旋转）

[6] Positive：正向（1 = 激活正向旋转，0 = 未激活正向旋转）

[7] OverV：所有模式都已激活速度超驰。取值范围为 0% - 199%，默认值为 100%。例如，可将 OverV 设置为 60%。

[8] OverAcc：所有模式都已激活加速超驰。取值范围为 0%-100%，默认值为 100%。例如，可将 OverAcc 设置为 70%。

[9] OverDec：减速超驰已激活。取值范围为 0%-100%，默认值为 100%。例如，可将 OverAcc 设置为 50%。

[10] ConfigEpos：控制未在块中直接指定的 EPos 函数的输入。有关详细信息，请参见 组态输入“ConfigEPos”的说明 (页 696)。

组态输入“ConfigEPos”的说明

下表列出了“ConfigEpos”和“Telegram 111”之间的位映射关系：

ConfigEpos	Telegram 111
ConfigEPos.%X0	STW1.%X1
ConfigEPos.%X1	STW1.%X2
ConfigEPos.%X2	EPosSTW2.%X14
ConfigEPos.%X3	EPosSTW2.%X15
ConfigEPos.%X4	EPosSTW2.%X11
ConfigEPos.%X5	EPosSTW2.%X10
ConfigEPos.%X6	EPosSTW2.%X2
ConfigEPos.%X7	STW1.%X13
ConfigEPos.%X8	EPosSTW1.%X12
ConfigEPos.%X9	STW2.%X0
ConfigEPos.%X10	STW2.%X1
ConfigEPos.%X11	STW2.%X2
ConfigEPos.%X12	STW2.%X3
ConfigEPos.%X13	STW2.%X4
ConfigEPos.%X14	STW2.%X7
ConfigEPos.%X15	STW1.%X14
ConfigEPos.%X16	STW1.%X15
ConfigEPos.%X17	EPosSTW1.%X6
ConfigEPos.%X18	EPosSTW1.%X7

ConfigEpos	Telegram 111
ConfigEPos.%X19	EPosSTW1.%X11
ConfigEPos.%X20	EPosSTW1.%X13
ConfigEPos.%X21	EPosSTW2.%X3
ConfigEPos.%X22	EPosSTW2.%X4
ConfigEPos.%X23	EPosSTW2.%X6
ConfigEPos.%X24	EPosSTW2.%X7
ConfigEPos.%X25	EPosSTW2.%X12
ConfigEPos.%X26	EPosSTW2.%X13
ConfigEPos.%X27	STW2.%X5
ConfigEPos.%X28	STW2.%X6
ConfigEPos.%X29	STW2.%X8
ConfigEPos.%X30	STW2.%X9

“Status_table”参数的定义

“Status_table”位的定义如下所示：

表格 11-52 Status_table

字节偏移	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
0	预留	Overrange_Error [1]	AxisError [2]	AxisWarn [3]	Lockout [4]	AxisRef [5]	AxisPosOk [6]	Axisenabled [7]
1	对于 111 报文： Error ID [8]							
	对于 112 报文：Error ID [8]，AxisSpFixed							
2	Actmode [9]							
3								
4	Epos_zsw1 [10]							
5								
6	Epos_zsw2 [11]							
7								

[1] Overrange_Error：输入的数据超出范围。有关详细信息，请参见错误代码 3、4、5 (页 698)。

[2] AxisError：驱动器发生错误。（默认值 = 0）

[3] AxisWarn：已激活驱动器警告。（默认值 = 0）

[4] Lockout：取消激活接通。（默认值 = 0）

[5] AxisRef：设置参考点。（默认值 = 0）

[6] AxisPosOk：已达到轴的目标位置。（默认值 = 0）

[7] Axisenabled：驱动器已准备就绪且已接通。（默认值 = 0）

[8] Error ID：识别错误类型。有关详细信息，请参见 “Status_table”参数的错误代码 (页 698)。

[9] Actmode：当前激活的模式。（默认值 = 0）

[10] Epos_zsw1：EPos_zsw1 的状态（以位为单位）。有关详细信息，请参见 “Epos_zsw1”的分配 (页 699)。（默认值 = 0）

[11] Epos_zsw2：EPos_zsw2 的状态（以位为单位）。 有关详细信息，请参见 “Epos_zsw2” 的分配 (页 700)。（默认值 = 0）

“Status_table”参数的错误代码

下表列出了“Status_table”参数的错误代码：

表格 11-53 “Status_table”参数的错误代码

错误代码	说明
0	无错误。
1	检测到驱动器错误。
2	驱动器已禁用。
3	不支持所选模式。
4	参数 OverV、OverAcc 和 OverDec 的速率超出支持的取值范围。
5	运动模式“traversing block”的下所选块超出范围。

“Epos_zsw1”的分配

下表列出了“Epos_zsw1”的分配信息：

表格 11-54 Epos_zsw1

位	缩写	指示	驱动参数	函数图
0	ActTrvBit0	激活运行程序段，位 0	r2670.0	3650
1	ActTrvBit1	激活运行程序段，位 1	r2670.1	3650
2	ActTrvBit2	激活运行程序段，位 2	r2670.2	3650
3	ActTrvBit3	激活运行程序段，位 3	r2670.3	3650
4	ActTrvBit4	激活运行程序段，位 4	r2670.4	3650
5	ActTrvBit5	激活运行程序段，位 5	r2670.5	3650
6	Bit6	保留		
7	Bit7	保留		
8	StpCamMinAct	STOP 凸轮反向激活	r2684.13	3630
9	StpCamPlsAct	STOP 凸轮正向激活	r2684.14	3630
10	JogAct	已激活点动模式	r2094.0 [1]	2460
11	RefAct	参考点回零模式已激活	r2094.1 [1]	2460
12	FlyRefAct	被动回零已激活	r2684.1 [1]	3630
13	TrvBlAct	运行程序段模式已激活	r2094.2 [1]	2460
14	MdiStupAct	在直接设定值输入/MDI 模式中已激活设置	r2094.4 [1]	2460
15	MdiPosAct	在直接设定值输入/MDI 模式中已激活定位	r2094.3 [1]	2460

[1] r2669（函数图 3630）以位为单位显示。为此，P2099[0] = r2699 在连接器-双向转换器的输入端互联。

“Epos_zsw2”的分配

下表列出了“Epos_zsw2”的分配信息：

表格 11-55 Epos_zsw2

位	缩写	指示	驱动参数	函数图
0	TrkModeAct	跟进/跟踪模式已激活	r2683.0	3645
1	VeloLimAct	速度限制已激活	r2683.1	3645
2	SetPStat	设定值为静态	r2683.2	3645
3	111: PrntMrkOut	打印外部窗口标志	r2683.3	3614
	112	已到达目标位置		
4	FWD	轴向前移动	r2683.4	3635
5	BWD	轴向后移动	r2683.5	3635
6	SftSwMinAct	反向驱动软限位开关	r2683.6	3635
7	SftSwPlsAct	正向驱动软限位开关	r2683.7	3635
8	PosSmCam1	位置实际值 <= 凸轮开关位置 1	r2683.8	4025
9	PosSmCam2	位置实际值 <= 凸轮开关位置 2	r2683.9	4025
10	TrvOut1	包含运行程序段的直接输出 1	r2683.10	3616
11	TrvOut2	包含运行程序段的直接输出 2	r2683.11	3616
12	FxStpRd	已到达固定挡块	<未使用> (r2683.12)	3645
13	FxStpTrRd	已达到固定挡块夹持力矩	<未使用> (r2683.13)	3645
14	TrvFxStpAct	运行到固定挡块已激活	<未使用> (r2683.14)	3645
15	CmdAct	运行已激活	r2683.15	3645

11.8.2.3 带 SINA_POS 指令的 SINAMICS 模式选择

“ModePos”输入用于进行操作模式选择，其中有 8 种操作模式：

- 相对运动
- 绝对运动
- 恒速模式

- 主动回零
- 设置参考点
- 运行程序段
- 点动
- 增量点动
- 编码器调整（仅限 112 报文）

基本要求

除点动模式外，输入操作数“CancelTraversing”与“IntermediateStop”在所有模式下均具相关性，且在使用 SINAMICS 时必须设置为“1”。

在各个操作模式下，按照下列步骤启动驱动器：

- 将输入操作数“CancelTraversing”设置为 1。
- 将输入操作数“Intermediatestop”设置为 1。
- 在 Control_table 中，根据十进制系统将“ConfigEpos”设置为 3。

若要启动轴，将输入操作数“EnableAxis”设置为 1。

可以使用输入操作数“ModePos”设置或更改操作模式。

11.8.2.4 相对运动

相对运动模式支持电机轴启动相对于起始位置的定位运动。每次移动的距离都是递增的。

相对运动模式是借助 SINAMICS V90 PN 的“MDI relative positioning”功能实现的。使用 SINAMICS V90 PN 的集成位置控制器对运行轨迹进行位置控制。

要求

- 通过 ModePos = 1 选择本模式。
- 通过“EnableAxis”启动设备。
- 轴无需回零或通过编码器调整。
- 如果在大于 3 的操作模式下选择轴，则轴为静止。MDI 操作模式（1、2、3）可随时进行更改。

顺序

通过下列输入组态运行路径和动态响应：

- “位置”
- “速度”
- “OverV”（速度超驰）
- – 对于 111 报文：“OverAcc”（加速度超驰）
  – 对于 112 报文：“Acc”（加速度）
- – 对于 111 报文：“OverDec”（减速度超驰）
  – 对于 112 报文：“Dec”（减速度）

速度超驰请参见“速度”部分。例如，生效速度为 500 LU/分且 OverV 为 120%，则 SINAMICS V90 中的速度为 600LU/分。

必须将信号输入“CancelTraversing”和“IntermediateStop”设置为 1。“Jog1”和“Jog2”无效，必须将其设置为 0（假）。

相对运动的运行方向始终由运行路径的符号决定。例如，如果位置为 -1000，则方向为负。如果位置为 1000，则方向为正。

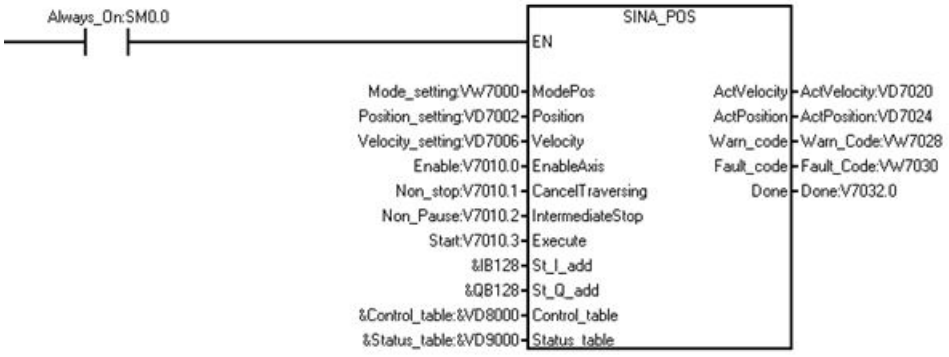
当上升沿为“Execute”时开始运行。可以使用“EPos_zsw1/EPos_zsw2”跟踪已激活命令的当前状态。

如果到达目标位置，则输出信号“Status_table”中的“AxisPosOK”位为 1。如果在运行过程中出现错误，则发出输出信号“Status_table”中的“AxisError”位。

说明

可以通过“ExecuteMode”使用新命令动态替换当前命令。仅适用于“ModePos”为 1/2/3 的模式。

相对运动模式示例



按以下方式设置相对运动模式：

1. 创建下列变量并将其写入相应输入操作数。

符号	地址	注释	相应输入操作数	数据类型	值		
Mode_setting	VW7000	模式选择	ModePos	WORD	1		
Position_setting	VD7002	位置长度	Position	DWORD	2500		
Velocity_setting	VD7006	速度	Velocity	DWORD	500		
Enable	V7010.0	启用驱动器。	EnableAxis	BOOL	1		
Non_stop	V7010.1	状态为不中止。	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	不暂停。	IntermediateStop	BOOL	1		
Start	V7010.3	启动驱动器。	Execute	BOOL	1		
Control_table	VD8000	控制参数。	Control_table	DWORD	VW8002	OverV	100
					VW8004	OverAcc	100
					VW8006	OverDec	100
					VD8008	ConfigEpos	3

111 报文
请参见“Control_table”参数的定义 (页 694)了解 112 报文的控制表。

说明

示例中的变量值仅供参考，请根据实际需要创建变量。

2. 在输入“St_I_add”和“St_Q_add”中输入数据。

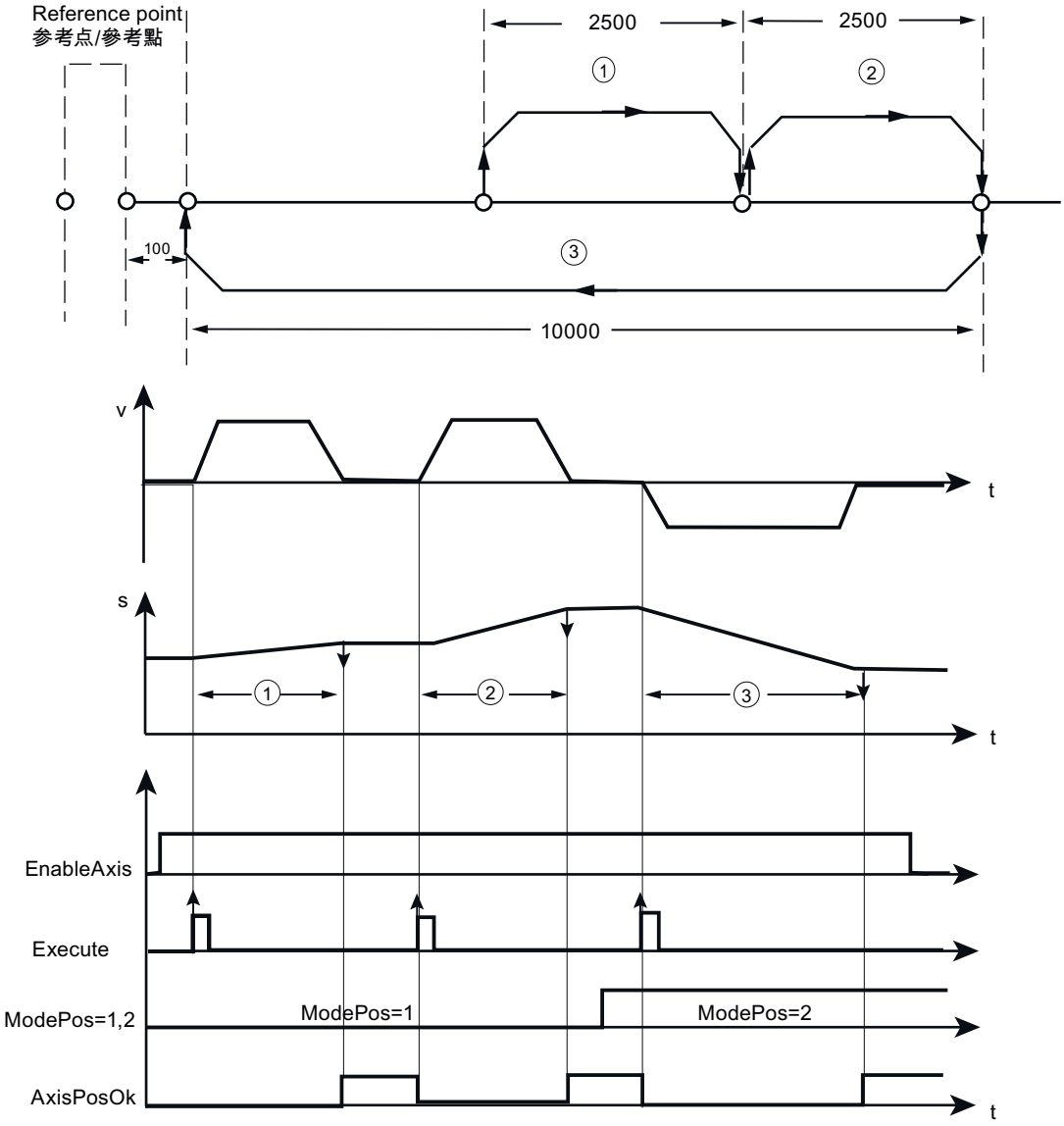
说明

对于四个输入参数“St_I_add”、“St_Q_add”、“Control_table”和“Status_table”，寻址指令操作数模式为间接寻址。必须在输入操作数的开头输入 & 符号。

对于输入操作数“St_I_add”和“St_Q_add”，确保偏移与 PROFINET 向导中的偏移一致。

结果：驱动器相对于初始位置距离移动了 2500 LU。例如，如果初始距离是 5000 LU，则驱动器位于 7500 LU 的位置。

下图显示了移动轨迹和动态参数：



① 相对运动

② 相对运动

③ 绝对运动

本例中，“v”表示速度，“s”表示位置，“t”表示时间。

11.8.2.5 绝对运动

绝对运动模式支持电机轴相对于绝对位置的定位运动。

绝对运动模式是借助 SINAMICS V90 PN 的“MDI absolute positioning”功能实现的。使用 SINAMICS V90 PN 的集成位置控制器实现绝对运动。

要求

- 通过“ModePos”= 2 选择本模式。
- 通过“EnableAxis”启动设备。
- 轴必须回零或通过编码器调整。
- 如果在大于 3 的操作模式下选择轴，则轴为静止。MDI 操作模式（1、2 或 3）可随时进行更改。

顺序

通过下列输入指定运行路径和动态响应：

- “位置”
- “速度”
- “OverV”（速度超驰）
- – 对于 111 报文：“OverAcc”（加速度超驰）
  – 对于 112 报文：“Acc”（加速度）
- – 对于 111 报文：“OverDec”（减速度超驰）
  – 对于 112 报文：“Dec”（减速度）

速度超驰请参见“速度”部分。例如，生效速度为 500 LU/分且 OverV 为 120%，则 SINAMICS V90 中的速度为 600LU/分。

必须将输入信号“CancelTraversing”和“IntermediateStop”设置为“1”。“Jog1”和“Jog2”无效，必须将其设置为“0”。

绝对运动中的运行方向始终是由到目标位置的最短距离确定的。输入“Positive”和“Negative”为“0"。

说明

可以使用“Positive”或“Negative”参数指定轴到目标位置的所需方向。

说明

可以使用以下 positioning control words 选择绝对运动方向：

- POS_STW1.9：
- POS_STW1.10：
  - 1 = 绝对运动/MDI 方向选取正向
  - 2 = 绝对运动/MDI 方向选取负向
  - 3 = 借助最短距离进行绝对运动

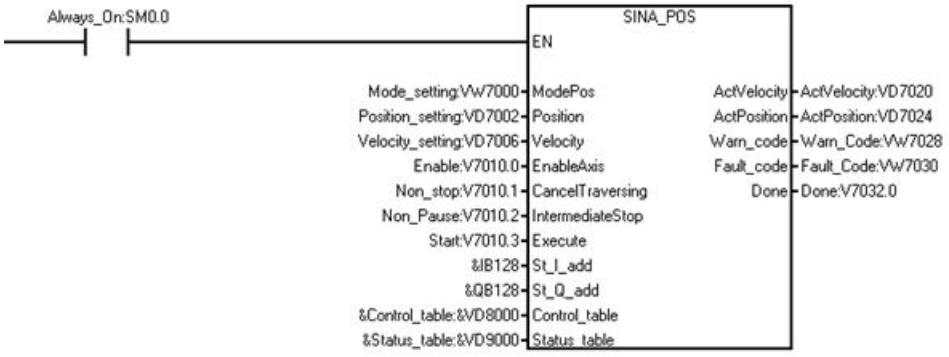
当上升沿为“Execute”时开始运行。可以使用“EPoszsw1/EPoszsw2”跟踪已激活命令的当前状态。

如果到达目标位置，则输出信号“Status_table”中的“AxisPosOK”位为 1。如果在运行过程中出现错误，则发出输出信号“Status_table”中的“AxisError”位。

说明

可以通过“Execute”使用新命令动态替换当前命令。仅适用于“ModePos”为 1/2/3 的模式。

绝对运动模式示例



按以下内容设置绝对运动模式：

1. 创建下列变量并将其写入相应输入操作数。
   报文 111：

符号	地址	注释	相应输入操作数	数据类型	值		
Mode_setting	VW7000	模式选择	ModePos	WORD	2		
Position_setting	VD7002	位置长度	Position	DINT	100		
Velocity_setting	VD7006	速度	Velocity	DINT	500		
Enable	V7010.0	启用驱动器。	EnableAxis	BOOL	1		
Non_stop	V7010.1	状态为不中止。	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	不暂停。	IntermidateStop	BOOL	1		
Start	V7010.3	启动驱动器。	Execute	BOOL	1		
Control_table	VD8000	控制参数。	Control_table	DWORD	VW8002	OverV	100
					VW8004	OverAcc	100
					VW8006	OverDec	100
					VD8008	ConfigEpos	3

报文 112：

符号	地址	注释	相应输入操作数	数据类型	值
Mode_setting	VW7000	模式选择	ModePos	WORD	2
Position_setting	VD7002	位置长度	Position	REAL	100.0
Velocity_setting	VD7006	速度	Velocity	REAL	500.0
Enable	V7010.0	启用驱动器。	EnableAxis	BOOL	1

符号	地址	注释	相应输入操作数	数据类型	值		
Non_stop	V7010.1	状态为不中止。	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	不暂停。	Intermidate Stop	BOOL	1		
Start	V7010.3	启动驱动器。	Execute	BOOL	1		
Control_table	VD8000	控制参数。	Control_table	DWORD	VW8002	OverV	100
					VD8004	Acc	100.0
					VD8008	Dec	100.0
					VD8012	fixStop Torque	100.0
					VD8016	refCoordinate	100.0
					VD8020	Config Epos	3

说明

示例中的变量值仅供参考，请根据实际需要创建变量。

2. 在输入“St_I_add”和“St_Q_add”中输入数据。

说明

对于四个输入参数“St_I_add”、“St_Q_add”、“Control_table”和“Status_table”，寻址指令操作数模式为间接寻址。必须在输入操作数的开头输入 & 符号。

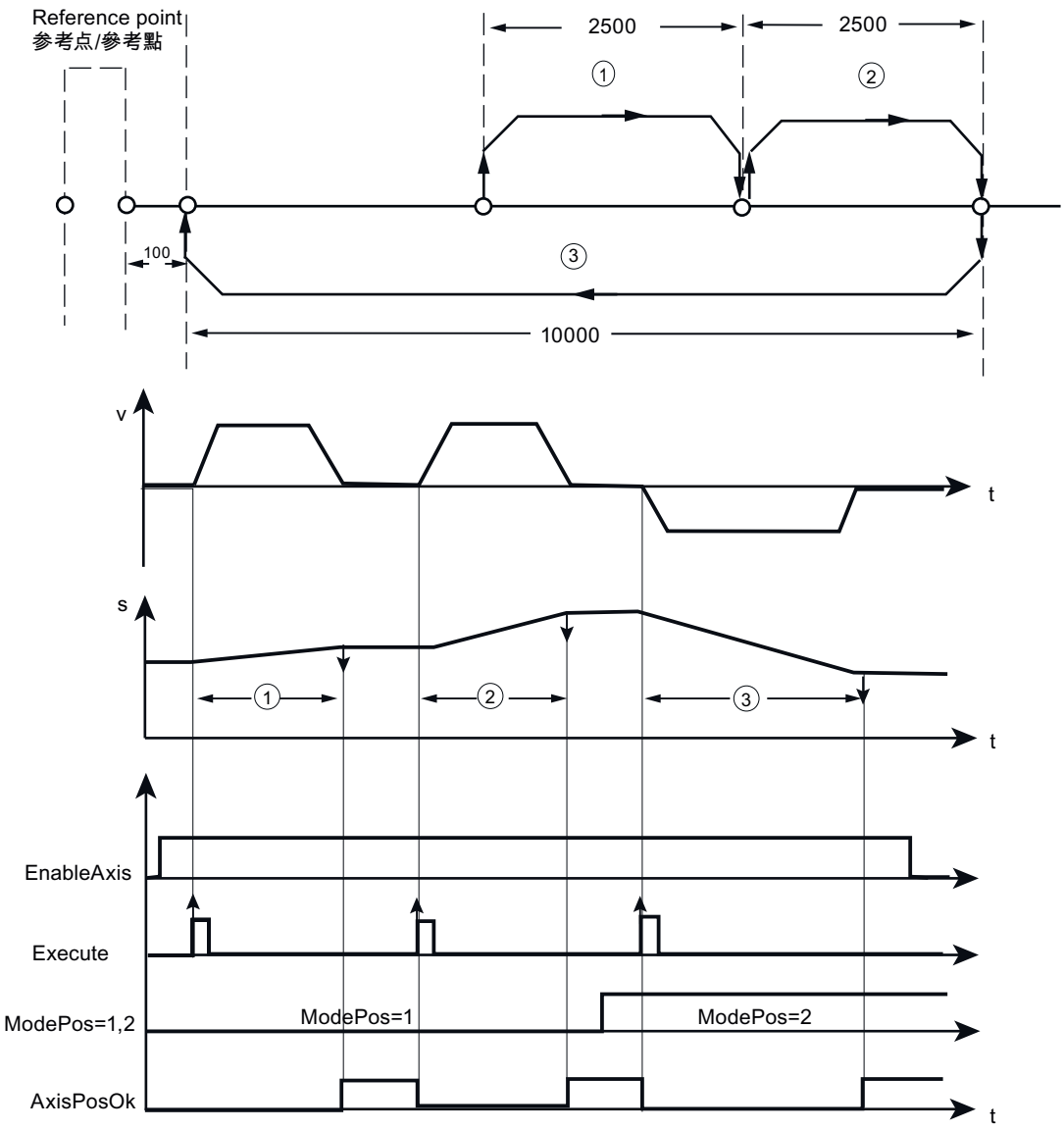
对于输入操作数“St_I_add”和“St_Q_add”，确保偏移与 PROFINET 向导中的偏移一致。

结果：驱动器位于位置 100。

说明

对于“绝对运动”操作模式，必须在“主动回零”(Referencing (active referencing)) 或“设置参考点”(Referencing (set reference point)) 模式中设置一个有效的参考位置。

下图显示了移动轨迹和动态参数：



① 相对运动

② 相对运动

③ 绝对运动

本例中，“v”表示速度，“s”表示位置，“t”表示时间。

11.8.2.6 恒速模式

恒速模式可使轴以恒定速度向正方向或负方向进行受位置控制的行进，无需借助 SINAMICS V90 PN 的“MDI set up”功能指定目标位置。

要求

- 通过“ModePos”= 3 选择本模式。
- 通过“EnableAxis”启动设备。
- 轴无需回零或通过编码器调整。
- 如果操作模式大于 3，则轴为静止。MDI 操作模式（1、2 或 3）可随时进行更改。

顺序

通过下列输入指定运行路径和动态响应：

- “正逻辑”或“负逻辑”
- “速度”
- “OverV”（速度超驰）
- “OverAcc”（加速超驰）
- “OverDec”（减速超驰）

必须将输入信号“CancelTraversing”和“IntermediateStop”设置为“1”。“Jog1”和“Jog2”无效，必须将其设置为“0”。

通过“Positive”和“Negative”确定运行方向。同时选择将停止轴，不产生其他警告和故障。

当上升沿为“Execute”时开始运行。可以使用“EPos_zsw1/EPos_zsw2”跟踪已激活命令的当前状态。

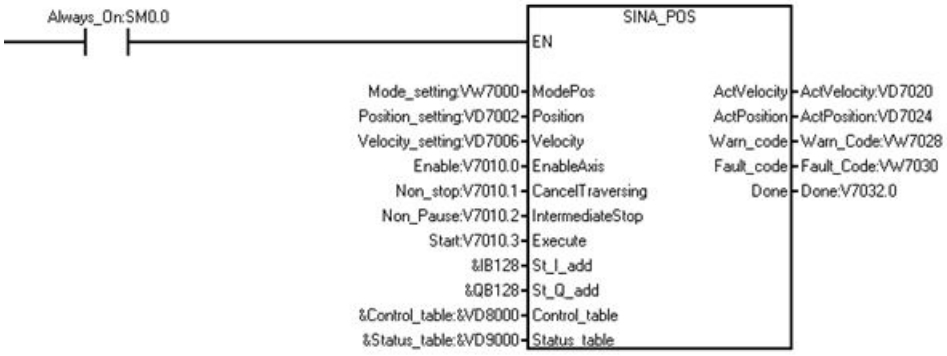
通过拒绝运行任务终止恒速模式且轴已停止时设置输出信号“AxisPosOk"。

如果在运行过程中出现错误，则发出输出信号“Status_table”中的“AxisError”位。

说明

可以通过“Execute”使用新命令动态替换当前命令。仅适用于“ModePos”为 1/2/3 的模式。

恒速模式示例


Always_On:SM0.0
SINA_POS
EN
Mode_setting:VW7000
Position_setting:VD7002
Velocity_setting:VD7006
Enable:V7010.0
Non_stop:V7010.1
Non_Pause:V7010.2
Start:V7010.3
&IB128
&QB128
&Control_table:&VD8000
&Status_table:&VD9000
ModePos
Position
Velocity
EnableAxis
CancelTraversing
IntermediateStop
Execute
St_I_add
St_Q_add
Control_table
Status_table
ActVelocity
ActPosition
Warn_code
Fault_code
Done
ActVelocity:VD7020
ActPosition:VD7024
Warn_Code:VW7028
Fault_Code:VW7030
Done:V7032.0

按以下内容设置恒速模式：

1. 创建下列变量并将其写入相应输入操作数。

符号	地址	注释	相应输入操作数	数据类型	值		
Mode_setting	VW7000	模式选择	ModePos	WORD	3		
Velocity_setting	VD7006	速度	Velocity	DWORD	500		
Enable	V7010.0	启用驱动器。	EnableAxis	BOOL	1		
Non_stop	V7010.1	状态为不中止。	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	不暂停。	IntermediateStop	BOOL	1		
Start	V7010.3	启动驱动器。	Execute	BOOL	1		
Control_table	VD8000	控制参数。	Control_table	DWORD	VW8002	OverV	100
					VW8004	OverAcc	100
					VW8006	OverDec	100
					VD8008	ConfigEpos	3
					V8000.0 [1]	Positive	1
					V8000.1 [1]	Negative	0

[1] V8000.0 和 V8000.1 的值不能同时为 1 或 0。

说明

示例中的变量值仅供参考，请根据实际需要创建变量。

2. 在输入“St_I_add”和“St_Q_add”中输入数据。

说明

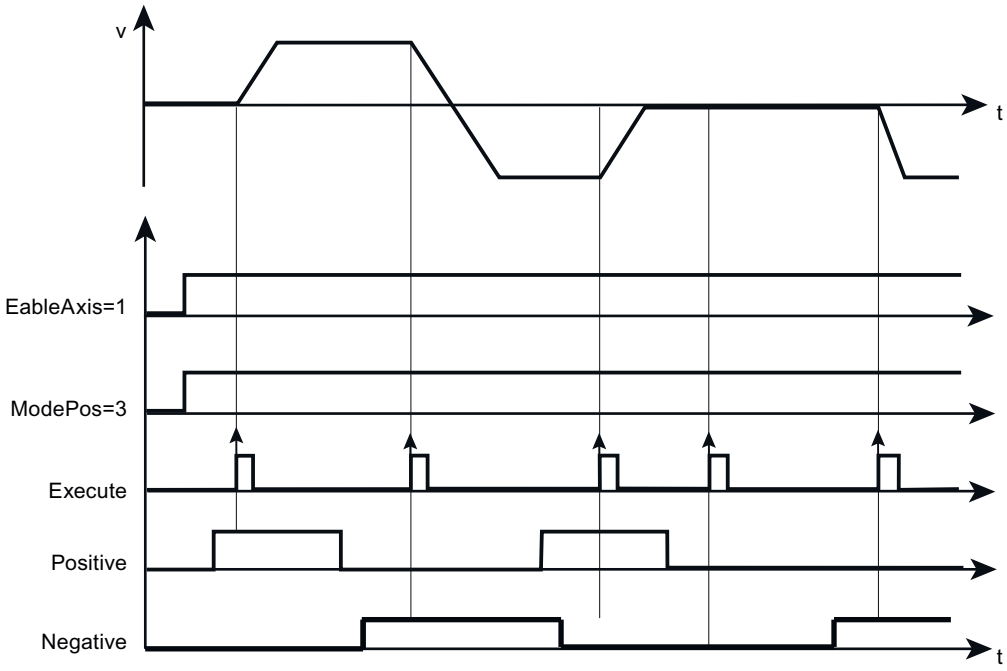
对于四个输入参数“St_I_add”、“St_Q_add”、“Control_table”和“Status_table”，寻址指令操作数模式为间接寻址。必须在输入操作数的开头输入 & 符号。

对于输入操作数“St_I_add”和“St_Q_add”，确保偏移与 PROFINET 向导中的偏移一致。

您可以按以下选项设置变量：

- 在输入信号“Control_table”中：
  - 若要使驱动器进行正向移动，将 V8000.0 设置为 1，V8000.1 设置为 0。
  - 若要使驱动器进行负向移动，将 V8000.1 设置为 1，V8000.0 设置为 0。
- 若要停止驱动器，将变量“Non_stop”设置为 0。
- 若要暂停驱动器，将变量“Non_Pause”设置为 0。

下图显示了移动轨迹和动态参数：



本例中，“v”表示速度，“t”表示时间。

11.8.2.7 主动回零

在“主动回零”(Referencing (active referencing)) 模式中能够通过 SINAMICS V90 PN 的“Active referencing”功能使轴以预定义速度和参考模式沿正向或负向回参考点。

要求

- 通过“ModePos”= 4 选择本模式。
- 通过“EnableAxis”启动设备。
- 轴为静止。

顺序

要求的速度作为速度配置文件保存在 SINAMICS V90 中。另外，预设的加速值和减速值已在轴的运行配置文件中激活。“OverV”速度超驰影响预组态的运行速度。例如，生效速度为 500 LU/分且 OverV 为 120%，则 SINAMICS V90 中的速度为 600LU/分。

必须将输入信号“CancelTraversing”和“IntermediateStop”设置为“1”。“Jog1”和“Jog2”无效，必须将其设置为“0”。

通过“Positive”和“Negative”确定运行方向。不允许同时选择，否则将产生故障。

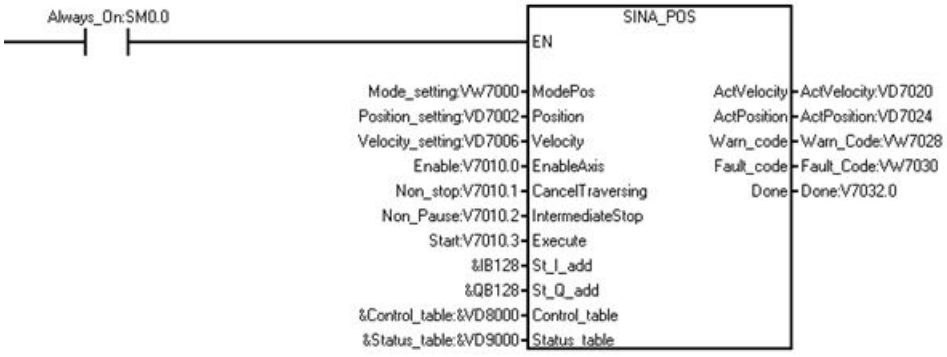
当上升沿为“Execute”时开始回参考点。

当上升沿为“Execute”时开始运行。可以使用“EPos_zsw1/EPos_zsw2”跟踪已激活命令的当前状态。

如果正确找到并评估了参考凸轮，则设置输出信号“Status_table”中的“AxisRef”位。

如果在运行过程中出现错误，则发出输出信号“Status_table”中的“AxisError”位。

主动回零模式示例



按以下内容将操作模式设为“Referencing (active referencing)”：

1. 创建下列变量并将其写入相应输入操作数。

符号	地址	注释	相应输入操作数	数据类型	值		
Mode_setting	VW7000	模式选择	ModePos	WORD	4		
Position_setting	VD7002	位置长度	Position	DWORD	2500		
Velocity_setting	VD7006	速度	Velocity	DWORD	500		
Enable	V7010.0	启用驱动器。	EnableAxis	BOOL	1		
Non_stop	V7010.1	状态为不中止。	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	不暂停。	IntermediateStop	BOOL	1		
Start	V7010.3	启动驱动器。	Execute	BOOL	1		
Control_table	VD8000	控制参数。	Control_table	DWORD	V8000.0 [1]	Positive	1
					V8000.1 [1]	Negative	0
					VD8008	ConfigEpos	3
						V8011.6（第 7 位）	1

[1] V8000.0 和 V8000.1 的值不能同时为 1 或 0。

说明

示例中的变量值仅供参考，请根据实际需要创建变量。

2. 在输入“St_I_add”和“St_Q_add”中输入数据。

说明

对于四个输入参数“St_I_add”、“St_Q_add”、“Control_table”和“Status_table”，寻址指令操作数模式为间接寻址。必须在输入操作数的开头输入 & 符号。

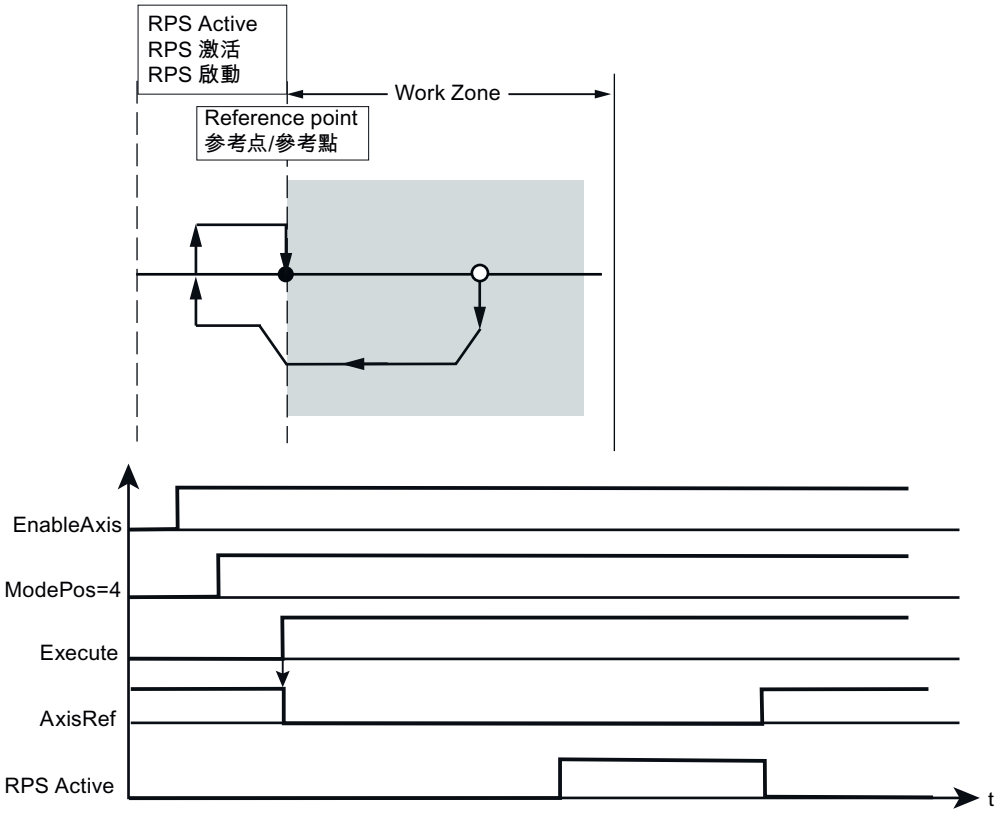
对于输入操作数“St_I_add”和“St_Q_add”，确保偏移与 PROFINET 向导中的偏移一致。

3. 在变量“Control_table”中，如果将 V8000.0 设置为 1 并将 V8000.1 设置为 0，则驱动器将沿正向移动寻找参考点。如果将 V8000.1 设置为 1 并将 V8000.0 设置为 0，则驱动器将沿负向移动寻找参考点。

说明

当外部参考信号直接与 PLC 相连接时，可通过数字量输入将 V8011.6 设为 1。驱动器停止且成功设置参考点。

下图显示了移动轨迹和动态参数：



本例中，“t”表示时间。

11.8.2.8 设置参考点

“设置参考点”(Referencing (set reference point)) 模式能够使轴在任意位置回参考点，通过 SINAMICS V90 PN 的“Set reference point”功能实现。

要求

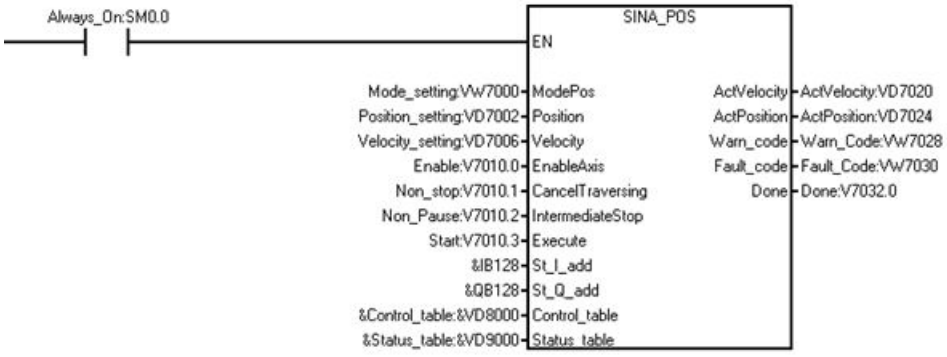
- 通过“ModePos”= 5 选择本模式。
- 轴可处于闭环控制，但必须静止。

顺序

当上升沿为“Execute”时，轴为静止且已设置参考点。

如果设置参考点时出现错误，则发出输出信号“Status_table”中的“AxisError”位。

设置参考点模式示例



按以下内容将操作模式设置为“设置参考点”(Referencing (set reference point))：

1. 创建下列变量并将其写入相应输入操作数。

符号	地址	注释	相应输入操作数	数据类型	值		
Mode_setting	VW7000	模式选择	ModePos	WORD	5		
Position_setting	VD7002	位置长度	Position	DWORD	2500		
Velocity_setting	VD7006	速度	Velocity	DWORD	500		
Enable	V7010.0	启用驱动器。	EnableAxis	BOOL	1		
Non_stop	V7010.1	状态为不中止。	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	不暂停。	IntermediateStop	BOOL	1		
Start	V7010.3	启动驱动器。	Execute	BOOL	1		
Control_table	VD8000	控制参数。	Control_table	DWORD	VD8008	ConfigEpos	3
Status_table	VD9000	状态参数。	Status_table	DWORD	V9000.2	AxisRef	1

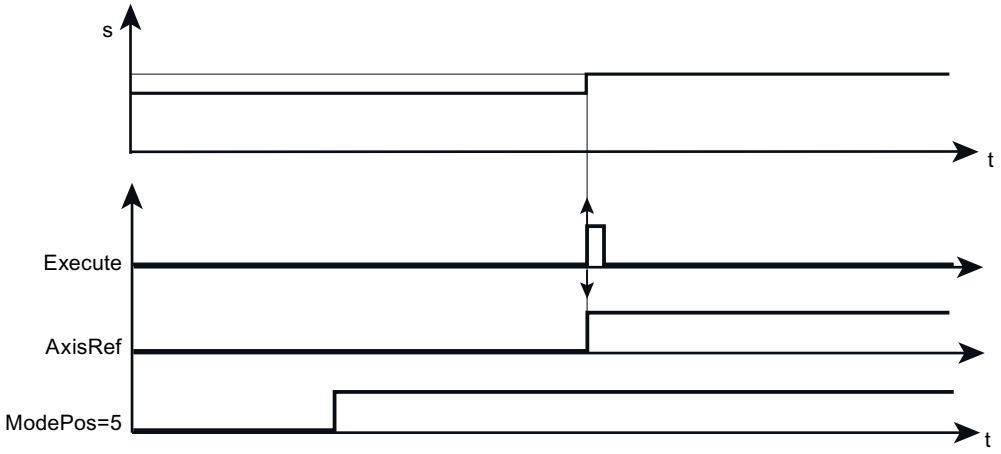
2. 在输入“St_I_add”和“St_Q_add”中输入数据。

说明

对于四个输入参数“St_I_add”、“St_Q_add”、“Control_table”和“Status_table”，寻址指令操作数模式为间接寻址。必须在输入操作数的开头输入 & 符号。

对于输入操作数“St_I_add”和“St_Q_add”，确保偏移与 PROFINET 向导中的偏移一致。

下图显示了移动轨迹和动态参数：



本例中，“s”表示位置，“t”表示时间。

11.8.2.9 运行程序段

“运行程序段”(Traversing blocks) 模式通过 SINAMICS V90 PN 的 "Traversing blocks" 功能实现。

要求

- 通过“ModePos”= 6 选择本模式。
- 通过“EnableAxis”启动设备。
- 轴为静止状态。
- 轴必须回零或通过编码器调整。

顺序

说明

可以使用“Position”输入选择要开始的运行任务。111 报文值为从 0 到 15，112 报文值为从 0 到 31。

如果该值超出范围，则在 "Status_table" 的 "Overrange_Error" 位显示错误代码 5 (页 698)。

SINAMICS V90 中的运行程序段参数指定任务模式、目标位置和动态响应。速度超驰“OverV”指的是存储在运行程序段中的速度设定值。必须将操作条件“CancelTraversing”和“IntermediateStop”设置为 1。“Jog1”和“Jog2”无效，必须将其设置为“0”。

运行方向由任务模式和位置设定值决定。在此，运行方向与“Positive”和“Negative”无关，必须将其设置为“0”。

说明

可以使用“Positive”或“Negative”参数指定轴到目标位置的所需方向。

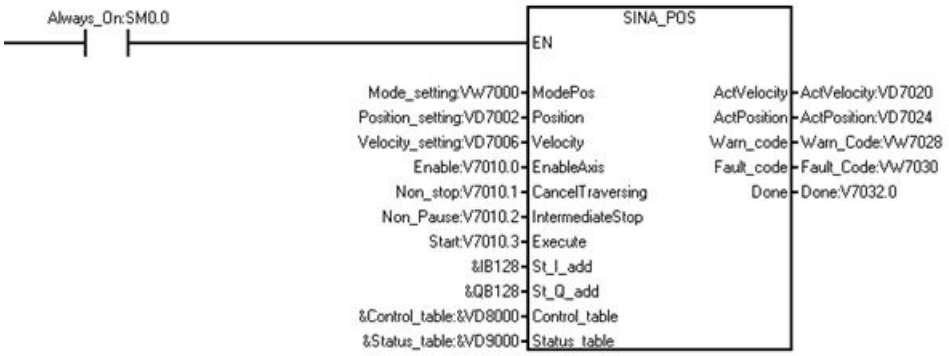
当上升沿为“ExecuteMode”时开始运行。可以使用“EPos_zsw1/EPos_zsw2”跟踪已激活命令的当前状态。

SINA_POS 指令在通过“AxisPosOk”确认何时成功到达运行路径终点。如果在运行过程中出现错误，则发出输出信号“Status_table”中的“AxisError”位。

说明

可以通过“Execute”使用新命令动态替换当前命令。仅适用于同一操作模式。

运行程序段模式示例



按以下内容将操作模式设置为“运行程序段”：

1. 创建下列变量并将其写入相应输入操作数。

符号	地址	注释	相应输入操作数	数据类型	值		
Mode_setting	VW7000	模式选择	ModePos	WORD	6		
Position_setting	VD7002	位置长度	Position	DWORD	输入所需块的索引。最多支持 16 块，因此这一变量值的范围为 0 至 15。		
Enable	V7010.0	启用驱动器。	EnableAxis	BOOL	1		
Non_stop	V7010.1	状态为不中止。	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	不暂停。	IntermediateStop	BOOL	1		
Start	V7010.3	启动驱动器。	Execute	BOOL	1		
Control_table	VD8000	控制参数。	Control_table	DWORD	VD8008	Config Epos	3

说明

示例中的变量值仅供参考，请根据实际需要创建变量。

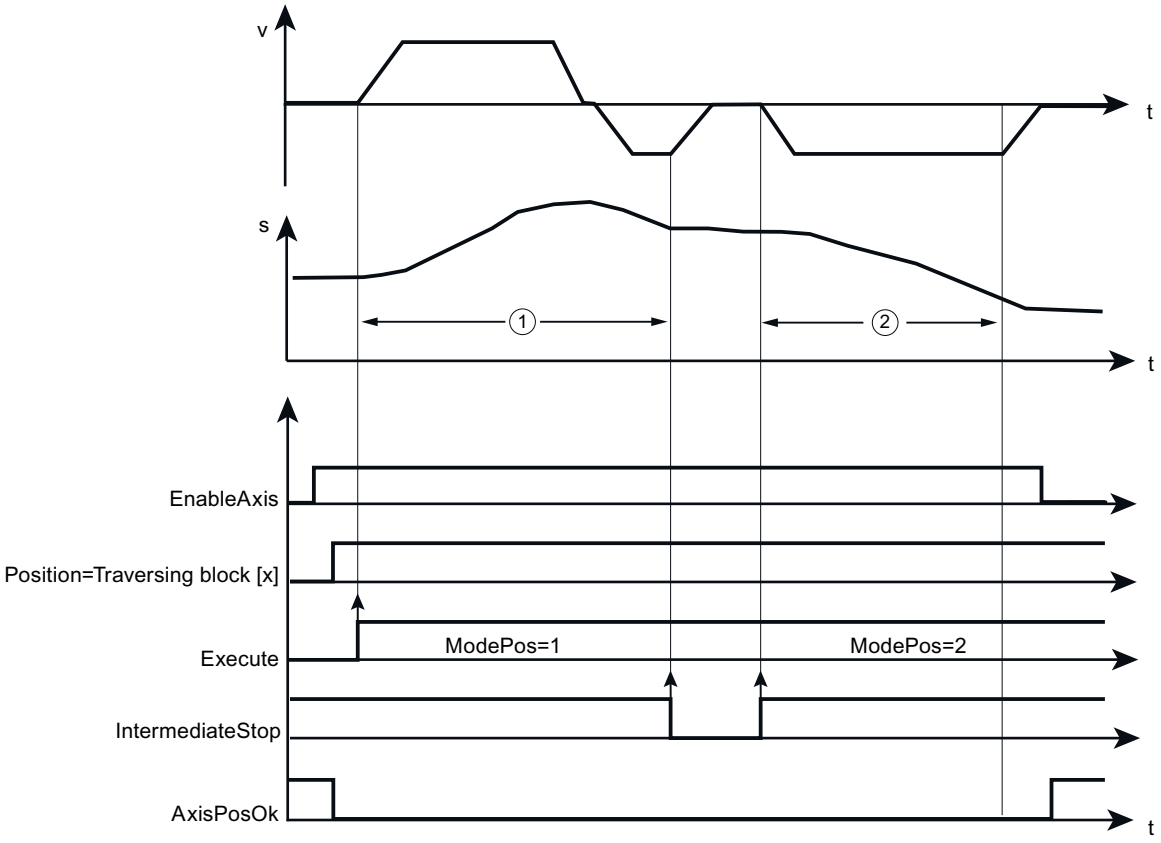
2. 在输入“St_I_add”和“St_Q_add”中输入数据。

说明

对于四个输入参数“St_I_add”、“St_Q_add”、“Control_table”和“Status_table”，寻址指令操作数模式为间接寻址。必须在输入操作数的开头输入 & 符号。

对于输入操作数“St_I_add”和“St_Q_add”，确保偏移与 PROFINET 向导中的偏移一致。

下图显示了移动轨迹和动态参数：



本例中，“v”表示速度，“s”表示位置，“t”表示时间。

11.8.2.10 点动

“点动”(Jog) 模式通过 SINAMICS V90 PN 的“Jog”功能实现。可通过 SINAMICS V90 PN 的集成位置控制器使轴以一定速度按指定位置运行。

要求

- 通过“ModePos”= 7 选择本模式。
- 通过“EnableAxis”启动设备。
- 轴为静止。
- 轴无需回零或调整。

顺序

通过 V-assistant 形式或 SINAMICS V90 中用于组态操作模式的 SINA_PARA_S 指令指定点动速度。SINAMICS V90 在轴的动态响应中使用加速和减速。

速度超驰也适用于此操作模式，通过“OverV”进行设置。例如，生效速度为 500 LU/分且 OverV 为 120%，则 SINAMICS V90 中的速度为 600LU/分。

这一操作模式与输入信号“CancelTraversing”和“IntermediateStop”无关，必须将其设置为“1"。

说明

在 SINA_POS 中，“Jog1”和“Jog2”为点动模式的信号源。“Jog1”为负向信号源，“Jog2”为正向信号源。

可以在 SINAMICS V90 PN 中设置运行距离。

速度设定值设置了点动的运行方向。

这一操作模式与输入参数“Positive”和“Negative”无关，可将其设置为“0"。

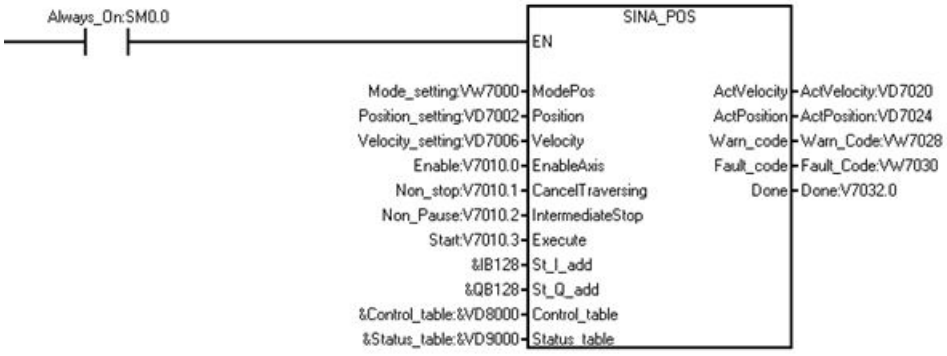
可以使用“EPos_zsw1/EPos_zsw2”跟踪已激活命令的当前状态。

SINA_POS 指令通过“AxisEnabled”显示当前命令进程，通过“AxisPosOk”在轴静止时知晓点动函数已终止（“Jog1”或“Jog2”= 0）。如果在运行过程中出现错误，则发出输出信号“Status_table”中的“AxisError”位。

说明

可以通过“Jog1”或“Jog2”使用新命令动态替换当前命令。仅适用于“ModePos”为 1/2/3 的模式。

点动模式示例



按以下内容设置点动模式：

1. 创建下列变量并将其写入相应输入操作数。

符号	地址	注释	相应输入操作数	数据类型	值		
Mode_setting	VW7000	模式选择	ModePos	WORD	7		
Enable	V7010.0	启用驱动器。	EnableAxis	BOOL	1		
Non_stop	V7010.1	状态为不中止。	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	不暂停。	IntermediateStop	BOOL	1		
Control_table	VD8000	控制参数。	Control_table	DWORD	V8000.2 [1]	Jog1	0
					V8000.3 [1]	Jog2	1
					VD8008	Config Epos	3

[1] V8000.2 和 V8000.3 的值不能同时为 1 或 0。

说明

示例中的变量值仅供参考，请根据实际需要创建变量。

2. 在输入“St_I_add”和“St_Q_add”中输入数据。

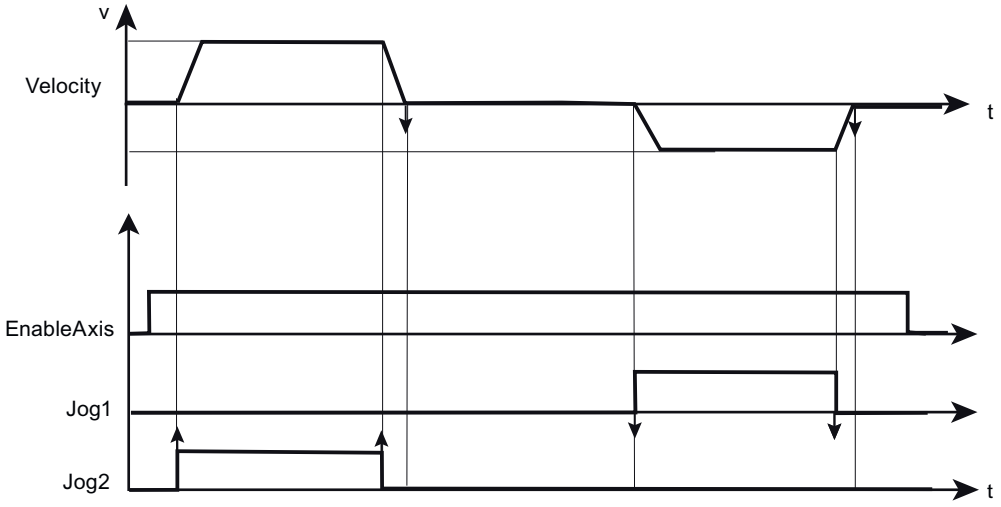
说明

对于四个输入参数“St_I_add”、“St_Q_add”、“Control_table”和“Status_table”，寻址指令操作数模式为间接寻址。必须在输入操作数的开头输入 & 符号。

对于输入操作数“St_I_add”和“St_Q_add”，确保偏移与 PROFINET 向导中的偏移一致。

3. 在变量“Control_table”中，如果将 V8000.2 设置为 1 并将 V8000.3 设置为 0，则驱动器将沿负向移动。如果将 V8000.3 设置为 1 并将 V8000.2 设置为 0，则驱动器将沿正向移动。
4. 若要暂停驱动器，将变量 V8000.2 或 V8000.3 设置为 0。

下图显示了移动轨迹和动态参数：



本例中，“v”表示速度，“t”表示时间。

11.8.2.11 增量点动

增量点动模式通过 SINAMICS V90 PN 的“Jog”功能实现。可通过 SINAMICS V90 PN 的集成位置控制器使轴采用位置受控且受距离制约的形式进行移动。

要求

- 通过“ModePos”= 8 选择本模式。
- 通过“EnableAxis”启动设备。
- 轴为静止。
- 轴**无需**回零或调整。

顺序

通过 V-assistant 形式或 SINAMICS V90 中用于组态操作模式的 SINA_PARA_S 指令指定点动速度。SINAMICS V90 在轴的动态响应中使用加速和减速。

速度超驰也适用于此操作模式，通过“OverV”进行设置。例如，生效速度为 500 LU/分且 OverV 为 120%，则 SINAMICS V90 中的速度为 600LU/分。

操作条件“CancelTraversing”和“IntermediateStop”与此操作模式无关，可设置为“1”。

说明

在 SINA_POS 中，“Jog1”和“Jog2”为点动模式的信号源。Jog 1 为负向信号源，Jog 2 为正向信号源。

可以在 SINAMICS V90 PN 中设置增量运行距离。

点动时的行进方向由设置的速度设定值决定。输入“正向”和“反向”与此操作模式无关，可设置为“0”（标准情况）。

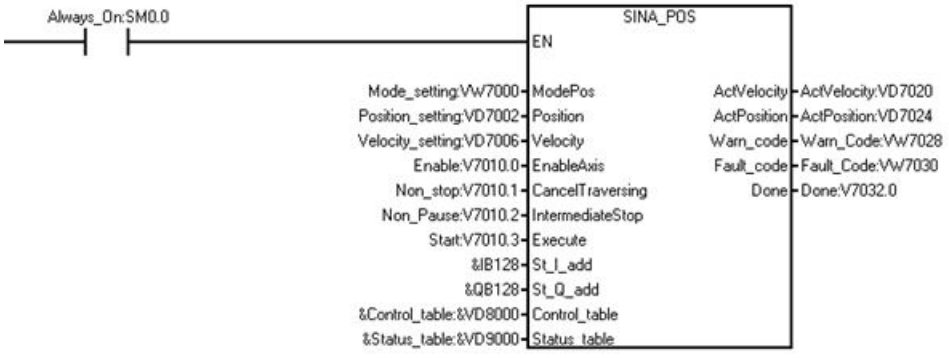
可以使用“EPosZSW1/EPosZSW2”跟踪已激活命令的当前状态。

该块通过“AxisEnabled”显示当前命令进程，通过位“AxisPosOk”在轴静止时确认点动函数已终止（“Jog1”或“Jog2”= 0）。如果在运行过程中出现错误，则发出输出信号“Status_table”中的“AxisError”位。

说明

可以通过“Jog1”或“Jog2”使用新命令动态替换当前命令。这仅适用于保持在一种点动模式下的情况。

增量点动模式示例



按以下内容将操作模式设为“增量点动”：

1. 创建下列变量并将其写入相应输入操作数。

符号	地址	注释	相应输入操作数	数据类型	值		
Mode_setting	VW7000	模式选择	ModePos	WORD	8		
Enable	V7010.0	启用驱动器。	EnableAxis	BOOL	1		
Non_stop	V7010.1	状态为不中止。	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	不暂停。	IntermediateStop	BOOL	1		
Control_table	VD8000	控制参数。	Control_table	DWORD	V8000.2 [1]	Jog1	0
					V8000.3 [1]	Jog2	1
					VD8008	Config Epos	3

[1] V8000.2 和 V8000.3 的值不能同时为 1 或 0。

说明

示例中的变量值仅供参考，请根据实际需要创建变量。

2. 在输入“St_I_add”和“St_Q_add”中输入数据。

说明

对于四个输入参数“St_I_add”、“St_Q_add”、“Control_table”和“Status_table”，寻址指令操作数模式为间接寻址。必须在输入操作数的开头输入 & 符号。

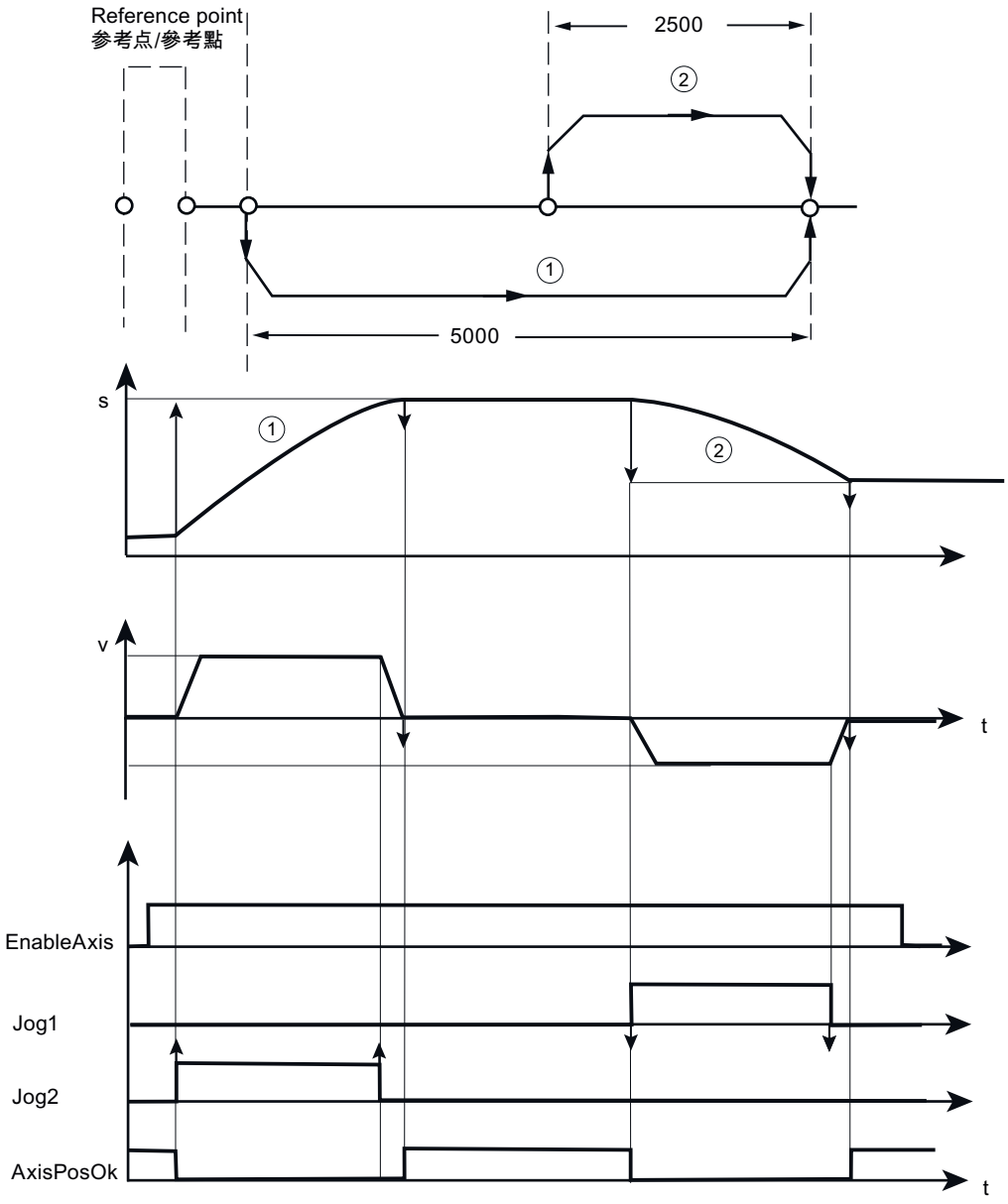
对于输入操作数“St_I_add”和“St_Q_add”，确保偏移与 PROFINET 向导中的偏移一致。

您可以对变量进行以下设置：

- 要使驱动器沿负向旋转，将 V8000.2 设置为 1，将 V8000.3 设置为 0。
- 要使驱动器沿正向旋转，将 V8000.3 设置为 1，将 V8000.2 设置为 0。

结果：驱动器按照 V-assistant 中设定的增量点动速度运行。无论是将“Jog1”还是将“Jog2”设置为 1，达到增量距离后，电机都会停止。

下图显示了移动轨迹和动态参数：



本例中，“v”表示速度，“s”表示位置，“t”表示时间。

11.8.2.12 编码器调整（仅限 112 报文）

该功能仅在使用了带绝对值编码器的伺服电机时才可用。只需对机械装置进行一次调整，这些调整在机器关闭后也可保留。在进行绝对值编码器调整前，必须先通过点动模式或手动模式将机器轴移动到所需位置。可以使用位置标记、固定挡块或测量轴与固定原点位置之间的距离作为参考。编码器调整通过“编码器调整”驱动功能执行。

说明

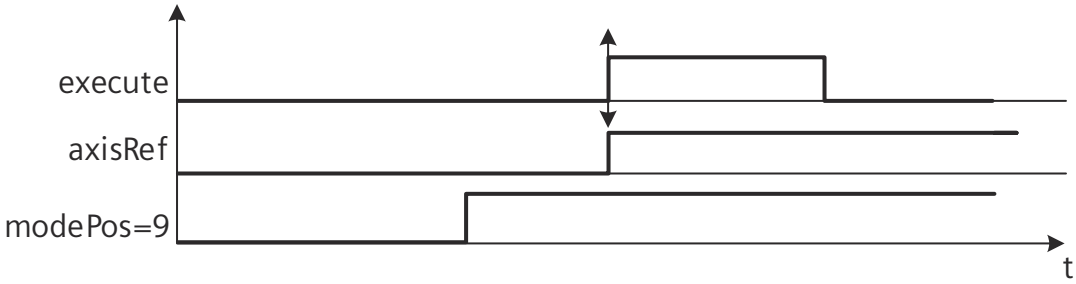
调整绝对值编码器后，需要将偏移值保存到驱动器中。

要求

- 通过“modePos”= 9 选择该操作模式。
- 轴可处于闭环控制，但必须在 standstill 状态下。

顺序

- 轴处于 standstill 位置移动到所需位置，然后通过“execute”的上升沿进行设置。
- 如果在设置原点位置时发生故障，则输出“error”输出信号。下图显示了编码器调整模式的工作原理。



11.8.3 SINA_SPEED 指令

11.8.3.1 使用 SINA_SPEED 指令的前提

使用 Speed_control 指令的前提如下：

- SINAMICS V90 PN 驱动器和伺服电机已准备就绪。
- 驱动器和 S7-200 Smart CPU 之间已连接 PROFINET 网络。
- V-assistant 软件已与 V90 PN 连接。有关详细信息，请参阅 SINAMICS V-ASSISTANT 在线帮助 (https://support.industry.siemens.com/cs/cn/zh/view/109738316)。可在 SIEMENS 工业在线支持网站中下载 SINAMICS V-ASSISTANT 软件 (https://support.industry.siemens.com/cs/cn/en/view/109738387)和 SINAMICS V90：PROFINET GSD 文件 (https://support.industry.siemens.com/cs/cn/en/view/109737269)。

通过 V-assistant 组态 SINAMICS V90 PN 参数的步骤

按以下步骤组态 SINAMICS V90 PN parameters：

1. 启动 V-assistant 软件。
2. 单击“在线”(Online) 选择工作模式。
3. 单击已连接的驱动器，单击“确认”(OK) 按钮。
4. 单击导航树的“选择驱动”(Select drive)，然后选择“控制模式”(Control Mode) 字段中的“速度控制 (s)”(Speed control (s))。

5. 单击导航树的“设置 PROFINET”(Set PROFINET)，然后单击“选择报文”(Select telegram)。

6. 在“选择报文”(Selection of telegram) 字段，选择“标准报文 1”(Standard telegram 1) 作为当前报文。SINA_SPEED 指令仅支持标准报文 1。

说明

在硬件网络拓扑中组态 PROFINET 网络时，确保报文与此步骤中的报文一致。

7. 单击导航树的“配置网络”(Configure network)。
8. 在“PN 站名”(Name of PN station) 中定义 PN 站名称。

说明

在硬件网络拓扑中组态 PROFINET 网络时，确保设备名称与此步骤中的 PN 站名称一致。

9. 单击“保存并激活”(Save and active) 按钮。

结果：驱动器重启且组态的参数生效。

图 11-3　硬件网络拓扑中 PROFINET 网络组态示例

11.8.3.2　SINA_SPEED 指令的输入和输出接口

SINA_SPEED 指令用于设置驱动器速度。

表格 11-56　SINA_SPEED 指令

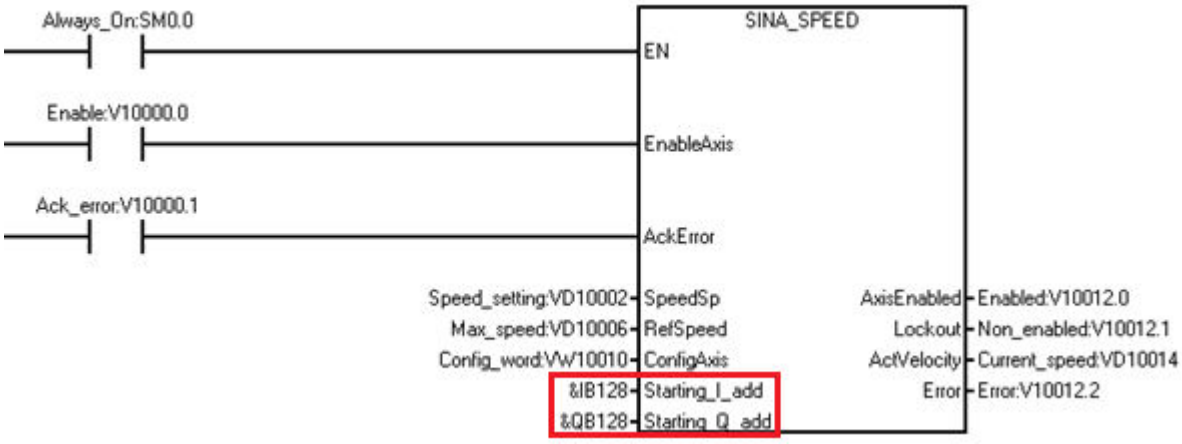
LAD/FBD	STL	说明
SINA_SPEED EN EnableAxis AckError SpeedSp　AxisEnabled RefSpeed　Lockout ConfigAxis　ActVelocity Starting_I_add　Error Starting_Q_add	CALL SINA_SPEED,EnableAxis, AckError, SpeedSp, RefSpeed, ConfigAxis Starting_I_add,Starting_Q_add, AxisEnabled, Lockout, ActVelocity, Error	SINA_SPEED 指令可用于控制驱动器移动的速度。

说明

对于两个输入“St_I_add”和“St_Q_add”，寻址指令操作数模式为间接寻址。

必须在输入操作数的开头输入 & 符号并确保偏移量与 PROFINET 向导中的一致。

请参考下图：



SINA_SPEED 指令的参数如下：

表格 11-57 SINA_SPEED 指令的参数

参数和类型		数据类型	说明
EnableAxis	IN	BOOL	“EnableAxis”= 1 -> 启动驱动器
AckError	IN	BOOL	确认轴故障 ->“AckFlt”= 1
SpeedSp	IN	REAL	速度设定值。SpeedSp 的值随 Refspeed 而变化。例如，如果将 Refspeed 设置为 1000 rpm，则 SpeedSp 值的范围为（0，1000 rpm）。
Refspeed	IN	REAL	驱动器的额定速度。取值范围（6，210000 rpm）。

参数和类型		数据类型	说明
ConfigAxis	IN	WORD	一个输入，用于控制未在块中直接指定的 SINA_SPEED 函数。有关详细信息，请参见 ““ConfigAxis”参数的定义 (页 734)”。（默认值 = 0）
Starting_I_add	IN	DWORD	PROFINET IO I 存储区起始地址的指针。
Starting_Q_add	IN	DWORD	PROFINET IO Q 存储区起始地址的指针。
AxisEnabled	OUT	BOOL	正在执行或启动驱动器。
Lockout	OUT	BOOL	1 = 接通禁止激活。
ActVelocity	OUT	REAL	实际速度 -> 取决于标定因子 RefSpeed。
Error	OUT	BOOL	1 = 组故障激活。

11.8.3.3 “ConfigAxis”参数的定义

下表列出了“ConfigAxis”参数的定义：

表格 11-58 ConfigAxis

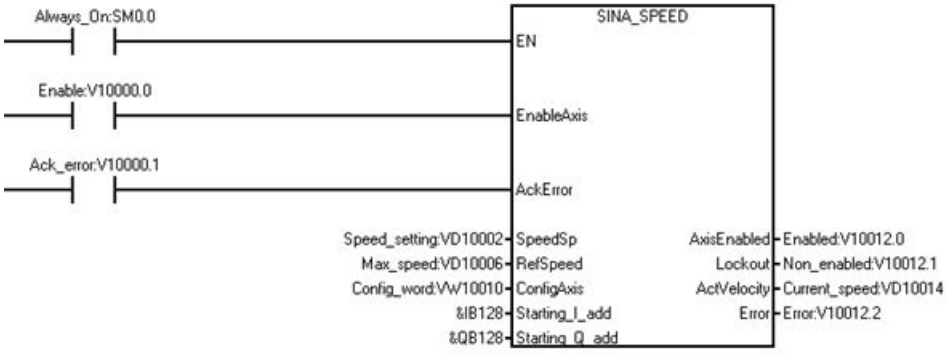
ConfigAxis	含义
Bit0	OFF2
Bit1	OFF3
Bit2	Inverter enable
Bit3	Enable ramp-function generator
Bit4	Continue ramp-function generator
Bit5	Enable speed setpoint
Bit6	Direction of rotation
Bit7	Unconditionally open holding brake
Bit8	Motorized potentiometer increase setpoint
Bit9	Motorized potentiometer, decrease setpoint
Bit10	Reserved
Bit11	Reserved
Bit12	Reserved
Bit13	Reserved

ConfigAxis	含义
Bit14	Reserved
Bit15	Reserved

11.8.3.4 SINA_SPEED 指令示例

SINA_SPEED 指令用于通过标准报文 1 对驱动器进行周期性控制。

示例



按如下所述插入并组态 SINA_SPEED 指令：

1. 创建下列变量并将其写入相应输入操作数。

符号	地址	注释	相应输入操作数	数据类型	值
Speed_setting	VD10002	速度	SpeedSp	REAL	100
Max_speed	VD10006	最大速度	RefSpeed	REAL	3000
Config_word	VW10010	ConfigAxis 参数表	ConfigAxis	WORD	63 **注：** 要启用驱动器，必须将此变量设置为 63（十进制）。
Enable	V10000.0	启用驱动器。	EnableAxis	BOOL	1
Ack_error	V10000.1	确认故障。	AckError	BOOL	
Enabled	V10012.0	轴已启用。	AxisEnabled	BOOL	
Non_enabled	V10012.1	驱动器未启用	Lockout	BOOL	
Current_speed	VD10014	实际速度	ActVelocity	REAL	
Error	V10012.2	驱动器错误	Error	REAL	

2. 在输入“St_I_add”和“St_Q_add”中输入数据。

说明

对于输入“St_I_add”和“St_Q_add”，寻址指令操作数模式为间接寻址。必须在输入操作数的开头输入 & 符号。

对于输入操作数“St_I_add”和“St_Q_add”，确保偏移与 PROFINET 向导中的偏移一致。

11.8.4 SINA_PARA_S 指令

11.8.4.1 使用 SINA_PARA_S 指令的前提

使用 SINA_PARA_S 指令的前提如下：

- SINAMICS V90 PN 驱动器和伺服电机已准备就绪。
- 驱动器和 S7-200 Smart CPU 之间已连接 PROFINET 网络。
- V-assistant 软件已与 SINAMICS V90 PN 连接。有关详细信息，请参见 SINAMICS V-ASSISTANT 在线帮助 (https://support.industry.siemens.com/cs/cn/zh/view/109738316)。可在 SIEMENS 工业在线支持网站中下载 SINAMICS V-ASSISTANT 软件 (https://support.industry.siemens.com/cs/cn/en/view/109738387)和 SINAMICS V90：PROFINET GSD 文件 (https://support.industry.siemens.com/cs/cn/en/view/109737269)。

通过 V-assistant 组态 SINAMICS V90 PN 参数的步骤

通过 V-assistant 组态 V90 PN 参数的步骤如下：

1. 启动 V-assistant 软件。
2. 单击“在线”(Online) 选择工作模式。
3. 单击已连接的驱动器，单击“确认”(OK) 按钮。
4. 单击导航树的“选择驱动”(Select drive)，然后选择“控制模式”(Control Mode) 字段的“速度控制 (s)”(Speed control (s)) 或“基本定位器控制 (EPOS)”(Basic positioner control (EPOS))。

5. 单击导航树的“设置 PROFINET”(Set PROFINET)，然后单击“选择报文”(Select telegram)。

6. 在“选择报文”(Selection of telegram) 字段，选择所需报文作为当前报文。SINA_PARA_S 指令仅支持 SIEMENS 报文 111 或标准报文 1。

说明

在硬件网络拓扑组态 PROFINET 网络时，确保报文与此步骤中的报文一致。

7. 单击导航树的“配置网络”(Configure network)。
8. 在“PN 站名”(Name of PN station) 中定义 PN 站名称。

说明

在硬件网络拓扑中组态 PROFINET 网络时，确保设备名称与此步骤中的 PN 站名称一致。

9. 单击“保存并激活”(Save and active) 按钮。

结果：驱动器重启且组态的参数生效。

图 11-4　硬件网络拓扑中 PROFINET 网络组态示例

11.8.4.2　SINA_PARA_S 指令的输入和输出接口

SINA_PARA_S 指令用于管理非周期性参数。

SINA_PARA_S 指令如下：

表格 11-59　SINA_PARA_S 指令

LAD/FBD	STL	说明
SINA_PARA_S EN Start ReadWrite Parameter　ValueRead1 Index　ValueRead2 ValueWrite1　Format ValueWrite2　ErrorNo Device_Number　ErrorId Device_Parameter　PN_Error_Code Status Status_Bit	CALL SINA_PARA_S,Start, ReadWrite, Parameter, Index, ValueWrite1, ValueWrite2, Device_Number, Device_Parameter, ValueRead1, ValueRead2, Format, ErrorNo, ErrorId, PN_Error_Code, Status, Status_Bit	SINA_PARA_S 指令用于设置驱动器参数或从驱动器读取参数。

下表列出了 SINA_PARA_S 指令的输入和输出信号：

表格 11-60 SINA_PARA_S 指令的参数

参数和类型		数据类型	说明
Start	IN	BOOL	启动任务（0 = 无任务或取消任务；1 = 开始执行任务）
ReadWrite	IN	BOOL	任务类型选择 0 = 读取，1 = 写入
Parameter	IN	INT	参数编号
Index	IN	INT	参数索引
ValueWrite1	IN	REAL	REAL 格式的参数值
ValueWrite2	IN	DINT	DINT 格式的参数值
DeviceNo	IN	WORD	设备编号
Device_Parameter	IN	DWORD	“Device_Parameter”起始地址的指针。“Device_Parameter”指 PROFINET 从站的参数。
ValueRead1	OUT	REAL	从驱动器读取的参数值（REAL 格式）
ValueRead2	OUT	DINT	从驱动器读取的参数值（DINT 格式）
Format	OUT	BYTE	读取的参数的格式 (页 741)
ErrorNo	OUT	WORD	根据 PROFIdrive 配置文件的错误编号 (页 742)
ErrorID	OUT	DWORD	错误 ID (页 743)。 第一个字：二进制编码，指示出现故障的参数访问 第二个字：故障类型
PN_Error_Code	OUT	DINT	根据 PROFINET 协议的错误代码。有关详细信息，请参见 *PROFINET IO（版本 2.3）的技术规范*。
Status	OUT	BYTE	当前操作的状态。 (页 744)
Status_bit	OUT	BYTE	状态表 (页 744)

“Device_parameter”的定义

下表列出了“Device_parameters”的位定义：

表格 11-61 Device_parameters

字节偏移	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
0	驱动器类型：对于 V90 驱动器，选择 2。对于 S200 驱动器，选择 1。对于其它驱动器，请参见相关手册。							
1	保留							
2	API 编号 [1]							
3								
4								
5								
6	插槽编号 [1]							
7								
8	子插槽编号 [1]							
9								

说明

[1] 有关 API 编号、插槽编号和子插槽编号的信息，请转至硬件网络拓扑获取信息。

“Format”参数的定义

从驱动器读取的参数格式如下：

表格 11-62 Format

ID	说明
02	整型 8
03	整型 16
04	整型 32
05	无符号 8
06	无符号 16
07	无符号 32

ID	说明
08	浮点
10	八进制字符串 8（16 位）
13	时间差（32 位）
41	字节
42	字
43	双字
44	错误

“ErrorNo”参数的定义

ErrorNo 指 PROFIdrive 协议定义的错误代码。

表格 11-63 ErrorNo

错误 ID	说明
00 十六进制	参数编号非法（访问的参数不存在）
01 十六进制	无法更改参数值（对无法更改的参数值发出更改请求）
02 十六进制	超出上限值或低于下限值（更改请求对应的值超出限值范围）
03 十六进制	子索引错误（访问的参数索引不存在）
04 十六进制	无数组（通过子索引访问无索引的参数）
05 十六进制	数据类型错误（更改请求对应的值与参数的数据类型不匹配）
06 十六进制	不允许进行设置，仅限复位（更改请求对应的值不等于 0，无权限）
07 十六进制	无法更改描述性元素（对无法更改的描述性元素发出更改请求）
09 十六进制	描述数据不可用（访问不存在的描述数据，参数值可用）
0B 十六进制	无主控制权（更改请求，但没有主控制权）
0F 十六进制	文本数组不存在（尽管参数值可用，但请求了不存在的文本数组）。
11 十六进制	因操作状态而无法执行请求（由于未指定的临时原因而无法访问）
14 十六进制	不允许的值（更改请求对应的值在限定范围内，但由于其它永久性原因而非法，例如，一个具有已定义单独值的参数）
15 十六进制	响应时间过长（实际响应时间超过传输的最大长度）
16 十六进制	参数地址非法（属性值、元素数、参数编号、子索引或上述组合的值非法或不受支持）
17 十六进制	格式非法（针对非法或不受支持的格式的更改请求）
18 十六进制	值的数量不一致（参数数据的值数与参数地址中的元素数不匹配）

错误 ID	说明
19 十六进制	驱动器对象不存在（访问不存在的驱动器对象）
20 十六进制	无法更改参数文本
21 十六进制	服务不受支持（非法或不受支持的请求 ID）
6B 十六进制	针对已启动的控制器的更改请求不可行。（驱动器拒绝更改请求，因为电机已开启）请遵循 SINAMICS V90、SIMOTICS S-1FL6 操作指令中参数列表相关部分给出的“可更改”参数属性（U、T）
6C 十六进制	未知单元。
77 十六进制	下载过程中不支持更改请求
81 十六进制	下载过程中不支持更改请求
82 十六进制	禁止接受主控制权
83 十六进制	无法进行所需互连（虽然连接器输入需要一个浮点值，但连接器输出未提供浮点值）
84 十六进制	逆变器不接受更改请求（逆变器正忙于内部计算）
85 十六进制	未定义访问方法
87 十六进制	专有技术保护已激活，访问已锁定
C8 十六进制	更改请求的值低于当前有效限值（更改请求对应的值处于“绝对”限值范围内但低于当前有效的下限值）
C9 十六进制	更改请求的值高于当前有效限值（例如，对于驱动功率，参数值过大）
CC 十六进制	更改请求未经允许（不允许进行更改，因为未提供访问代码）

“ErrorID”参数的定义

有两种错误类型：

- ErrorID[1]
- ErrorID[2]

表格 11-64 ErrorID[1]

ID	说明
0×000	无故障激活
0×001	内部报文错误激活
0×002	参数化错误激活
0×003	RDREC 和 WRREC 调用错误激活

ID	说明
0×004	将启动输入设置为“0”，以在激活数据传输期间取消作业
0×005	检测到未知数据类型； ErrorID[2] 的评估显示在最高值位有未知数据类型的参数

表格 11-65 ErrorID[2]

ID	说明
0×01	未知错误

“STATUS”参数的定义

STATUS 的参数如下：

表格 11-66 STATUS

字节偏移	位 6	位 5	位 4	位 3	位 2	位 1	位 0
0	A [1]	E [2]	错误代码 [3]				

[1] A：1 = 正在进行请求

[2] E：1 = 出错

[3] 错误代码：系统错误代码。有关详细信息，请参见 指令 RDREC 和 WRREC 的系统定义错误代码 (页 507)。

“Status_bit”参数的定义

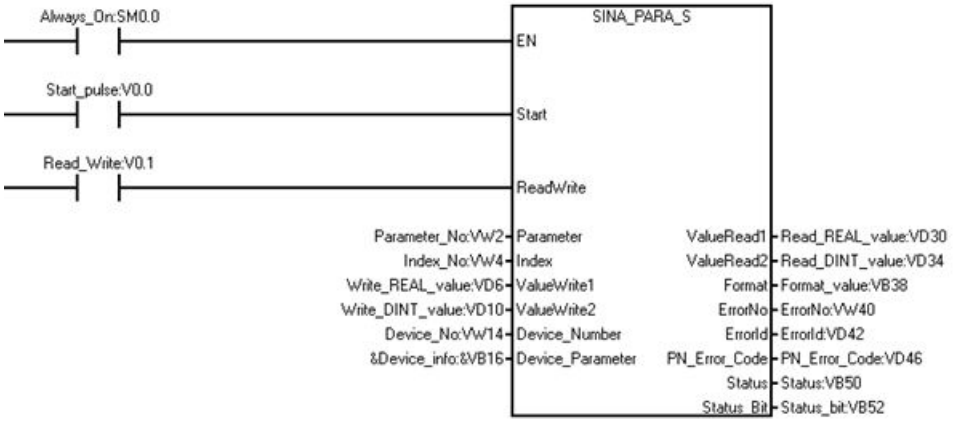
表格 11-67 Status_bit

字节偏移	位 3	位 2	位 1	位 0
0	错误	已完成	繁忙	就绪

11.8.4.3 SINA_PARA_S 指令示例

SINA_PARA_S 指令可以在非周期性通信中读写驱动器参数。

示例


Always_On:SM0.0
Start_pulse:V0.0
Read_Write:V0.1
SINA_PARA_S
EN
Start
ReadWrite
Parameter_No:VW2
Index_No:VW4
Write_REAL_value:VD6
Write_DINT_value:VD10
Device_No:VW14
&Device_info:&VB16
Parameter
Index
ValueWrite1
ValueWrite2
Device_Number
Device_Parameter
ValueRead1
ValueRead2
Format
ErrorNo
ErrorId
PN_Error_Code
Status
Status_Bit
Read_REAL_value:VD30
Read_DINT_value:VD34
Format_value:VB38
ErrorNo:VW40
ErrorId:VD42
PN_Error_Code:VD46
Status:VB50
Status_bit:VB52

按照以下步骤使用 SINA_PARA_S 指令修改驱动器参数：

1. 创建下列变量并将其写入相应输入操作数。

符号	地址	相应输入操作数	数据类型	值		
Start_pulse	V0.0	Start	BOOL	1		
Read_Write	V0.1	ReadWrite	BOOL	0 或 1		
Parameter_No	VW2	Parameter	INT	29070		
Index_No	VW4	Index	INT	1		
Write_REAL_value	VD6	ValueWrite1	REAL			
Write_DINT_value	VD10	ValueWrite2	DINT			
Device_No	VW14	Device_number	WORD	1		
Device_info	VB16	Device_Parameter	DWORD	VB16	Drive type	2
				VD18	API number	14848
				VW22	Slot number	1
				VW24	Subslot number	3
Read_REAL_value	VD30	ValueRead1	REAL			
Read_DINT_value	VD34	ValueRead2	DINT			
Format_value	VB38	Format	BYTE			
ErrorNo	VW40	ErrorNo	WORD			
ErrorId	VD42	ErrorID	DWORD			
PN_Error_Code	VD46	PN_Error_Code	DINT			
Status	VB50	Status	BYTE			

符号	地址	相应输入操作数	数据类型	值		
Status_bit	VB52	Status bit	BYTE	V52.0	Ready	
				V52.1	Busy	
				V52.2	Done	
				V52.3	Error	

2. 从驱动器读取参数，获取参数数据类型信息。
   – 将变量“Read_Write”设置为 0，开始读取驱动器参数的任务。
   – 在变量“Device_info”中输入设备参数信息。
   – 在 VB16“Axis number”中输入轴编号。
   – 在变量“Parameter_No”中输入参数编号。在变量“Index_No”中输入索引。
   – 将变量“Start_pulse”设置为 1 以开始任务。

   结果：
   如果参数数据类型为 REAL，则变量“Read_REAL_value”显示该值。
   如果参数数据类型为 DINT，则变量“Read_DINT_value”显示该值。
3. 将变量“Read_Write”设置为 1，开始将参数写入驱动器的任务。
4. 修改变量“Write_REAL_value”或“Write_DINT_value”中的参数：
   如果在步骤 2 中，变量“Format_value”显示下列数据：16#02、16#05、16#41、16#42、16#03、16#06、16#0A 或 16#08，则修改变量“Write_REAL_value”中的参数。
   如果在步骤 2 中，变量“Format_value”显示下列数据：16#43、16#04、16#07 或 16#0D，则修改变量“Write_DINT_value”中的参数。
5. 将变量“Start_pulse”设置为 1。

结果：

在 V-assistant 软件中，检查是否已成功修改参数。

11.9 用户定义的库编程提示

简介

指令库的创建始于一个 STEP 7-Micro/WIN SMART V3 项目。您可以选择使用任何包含子例程（SBR）或功能块（FB）的现有项目来生成指令库，但为了简化管理，我们更推荐创建一个专用于库源码的新项目。若需利用分散在不同项目中的现有子例程或功能块来构建指令库，应首先通过导入/导出功能或直接复制粘贴，将所有相关组件整合到一个统一的项目中。在所有所需的子例程和功能块整合完毕后，将调用这些组件的代码添加到主程序块中。此步骤旨在验证将在库中成为指令的子例程/功能块，并允许在正式创建指令库之前进行充分的测试和调试。请注意，此项目既可作为可下载的 S7-200 SMART 程序使用，也可以是指令库的源。

必须具有管理员权限才能创建用户定义的库。如果您使用“以管理员身份运行”(Run as administrator) 命令启动 STEP 7-Micro/WIN SMART，这将提供足够的权限。

数据类型与存储区使用规则

在创建指令库时，只有 V 区（变量区）和 DB2 区（用户变量区）的变量才允许使用复杂数据类型（如结构体、数组等）。其他用户地址区仅支持使用基本数据类型。

说明

在创建库时，请留意以下限制。如果您的 POU 包含以下任何情况，库创建过程将无法成功完成:

- POU 中包含了由运动控制自动生成的全局变量
- POU 中引用了绑定到 I、Q、M、S 区域的 UDT 或数组类型变量
- POU 中使用了 V 区的绝对地址编程

库中的功能块实例可以访问用户全局变量表中的所有基本类型变量。此外，它们还可以访问 V 区和用户变量区（DB2）中定义的复杂数据类型变量。

由于库 POU（程序组织单元）内部只能识别以库自身变量表为作用域的变量，而运动控制相关的变量（如轴、轴组、CamTable 等对象）是由运动控制组态自动生成，并无法直接加入到库的变量表中。因此，这些运动控制对象必须通过子例程或功能块的 IN_OUT 形参进行传递。这种方式确保了库能够安全、灵活地操作运动控制对象，避免了直接引用全局变量可能导致的冲突。

全局存储器使用

要消除库和项目之间出现存储器使用冲突的可能性，应避免在用户定义库中使用全局存储器。这里列出了最小化全局存储器使用的一些方法：

- 尽量用子例程/功能块参数代替全局存储器
  通过使用子例程/功能块参数，尽量减少库对全局存储器的依赖性。您可以指定库指令使用的存储器。
  例如，您可能有一个计算数值并将该输出存储在 V 存储单元的子例程。程序的其余部分则会读取该 V 存储器位置，以便确定计算的结果。如果您希望将该子例程放入库，考虑在子例程中增加一个输出参数，并将计算结果存储在该参数中。这样就无需使用 V 存储单元，并允许您决定结果的存储位置。

- 使用临时变量进行计算
  将临时变量用于计算和临时结果可尽量减少指令库对全局存储器的使用。您在子例程/功能块变量表中定义的临时变量位于本地存储器中，不会与项目冲突。
  本地存储器并非永久性存储器，不可在两次指令调用之间保存数据。本地存储器的持续时间仅为声明该本地存储器的子例程/功能块的执行阶段。

- 需要 V 存储器时
  当您必须在指令库中的子例程/功能块中使用全局 V 存储器时，应为所有 V 存储器声明变量，并在程序中使用这些变量。编译该库时，编译器将使用这些变量来决定库所需的存储器大小。
  通用规则是，将包括在库中的子例程/功能块所占用的 V 存储器分组到连续位置中。例如，假设子程序使用以下变量：

名称	地址
Output1	VB0
Output2	VB10
Output3	VB20

  库编译对此作出解释，指出库需要使用 21 个字节的 V 存储器。但是，如果我们按照以下方式重新分配变量，变量会使用 V 存储器的连续字节：

名称	地址
Output1	VB0
Output2	VB1
Output3	VB2

  编译器对此作出解释，指出库仅需要使用 3 个字节的 V 存储器。

说明

在重新分配变量地址以整合存储器使用时应当小心。有时，子例程/功能块地址之间具有内在的关系，这些关系必须加以保留。例如，一个变量可能代表表格开始，而另一个变量代表该表格的第三个元素。仅合并不相关变量的地址。

- 当需要 I/O 地址时
  有时您必须在计划包括在用户定义库中的子例程/功能块中使用实际 I/O 地址。必要时，可为所有使用子程序/功能块中的变量的 I/O 地址声明变量。
- 存储器初始化
  指令库不包括数据块。因此，如果子例程/功能块需要具有特定初始值的 V 存储器，则子例程/功能块应设置这些值。
- 当子例程/功能块使用累加器时
  用累加器进行计算通常会减少特定解决方案所需的代码数量，因为累加器可以作为字节、字或双字地址进行寻址。但是，累加器是全局资源，如果在计划包括在库中的子例程/功能块中使用累加器，则必须注意，子例程/功能块不会破坏累加器的内容。要在库子例程/功能块中安全地使用累加器，请执行以下步骤：
  – 在子例程/功能块开始位置，使用移动指令将累加器的当前内容复制到本地存储器中。这样将保留相应的内容，这样，子例程/功能块便可在之后将其恢复。
  – 您的子例程/功能块现在可自由使用累加器并可根据需要分配值。
  – 在子例程/功能块的结束位置，使用移动指令将本地存储器中保存的值恢复到累加器中。

指令和参数命名

当您设计成为库指令的子例程/功能块时，在为子例程/功能块和其参数命名时要格外小心。

在为成为库中的指令的子例程/功能块命名时，请考虑以下因素：

- 选择准确地描述指令目的的名称。
  请记住，使用库的人可能对每条指令的详细内容并不了解。请选择清楚地描述指令操作的名称。
- 避免使用通用名称。
  通过避免使用通用名称，即可尽量减少名称与用户已经使用的子例程/功能块名称重复的可能性。如果带有相同名称的 POU 已经在项目中存在，用户则无法调用指令。
- 注意 LAD 和 FBD 功能框标题的可视空间限制。
  在 LAD 和 FBD 编辑器中，STEP 7-Micro/WIN SMART V3 限制了指令名称（作为方框指令标题）可以显示的字符数。确切的可视字符数取决于用户选择的字体大小和网格宽度。
- 考虑本地化问题。
  如果您打算在全球各地分发指令库，请考虑使用用不同的语言容易识别的指令名称。避免使用方言、俚语或可作不同解释的词语。

以上命名指南大多也适用于指令参数。您为子例程/功能块在变量表中定义指令参数的名称。请记住，输入和输出参数名称可能是用户用于确定如何使用指令的唯一文档。定义的名称应有助于用户确定每个参数所代表的内容。

签名冲突及其处理

在将外部指令库导入到现有项目时，系统会执行完整性检查，以识别并处理签名冲突。当库中某个 SBR、FB 或 UDT 的名称及其内部结构与项目中已存在的同类型元素完全一致时，即会发生签名冲突，此时系统将提示您存在冲突，并建议为即将导入的库元素指定一个新的命名空间。一旦您成功指定命名空间并完成首次导入，后续从该库导入其他元素时，系统将自动沿用之前指定的命名空间进行导入。

使用子例程/功能块注释记录程序

如果您选择用密码保护库指令，则库用户无法看到指令中的源代码。但是，他们可以查看库中各子例程/功能块的注释。

子例程/功能块注释是记录每条指令功能的最佳位置。请考虑使用子例程/功能块注释描述下列信息：

- 该指令具体执行什么功能？
- 程序应如何调用指令？
  例如，如果指令仅限在首次扫描时调用，或者指令依赖于另一条指令是否成功完成，则应使用子例程/功能块注释记录指令的这些详细用法。
- 指令使用何种全局存储器？
  如果指令直接引用全局存储器，子例程/功能块注释可提醒用户这一用法，以便用户对程序进行设计时避免存储器访问冲突。
- 指令如何指示错误？
  很多指令可能会通过输出参数或在全局存储器位置中指示错误条件。使用子例程/功能块注释可通知用户可能的错误以及可采取的措施。

创建库

在“文件”(File) 菜单功能区的“库”(Libraries) 部分中，单击“创建库”(Create library) 按钮，通过项目中的子例程/功能块创建指令库。

调试库指令

创建指令库后，建立一些使用新库的测试项目，以便在将库分发给其他用户之前测试库指令的功能。您可能会发现原先没有意识到的指令和初始源项目之间的依赖性。当您测试指令库时，尽量避免对别人使用库的方法作出任何假设。请记住，使用库的人不具备您所具备的库操作的详细知识。

在项目中使用库

创建了用户定义库之后，其他项目便可使用它。必须关闭从中创建库的项目，然后打开一个新项目或其他项目以使用您的库。当打开新项目时，项目树中的库文件夹显示您在项目选项中为用户自定义库配置的文件夹中的库。

也可以打开在项目选项中为用户自定义库配置的文件夹，然后添加从其他来源获取的库。还可以从文件夹中删除库。文件夹中的所有库可用于新项目。您可以从项目树的库节点选择“刷新库”(Refresh Libraries) 命令以更新可用于项目的库集。请注意，“刷新库”(Refresh Libraries) 不包括您从当前打开的项目创建的库。

有关为需要 V 存储器的用户定义库分配存储器的信息，请参见“库管理器”主题。

调用库指令

已向项目添加库之后，按照与所有其他指令相同的方式添加指令和编辑参数。按照与子例程/功能块调用或功能框指令相同的方式提供指令参数以及逻辑控制指令的执行。

分发库指令

创建库时，输出是单个文件（.smartlibV3 文件），包含 STEP 7-Micro/WIN SMART V3 识别和使用指令库所需的所有内容。如要将指令库分发给其他 STEP 7-Micro/WIN SMART V3 用户，可将此文件的副本提供给他们。

反之，要使用其他人给您的指令库，请将 .smartlibV3 文件复制到计算机中的用户自定义库文件夹中，并将该库加入到项目中。

更改库中包括的子例程/功能块

如果您希望对库中的代码作出更改，打开作为库来源的项目，编辑包括在库中的 POU。创建库时，.smartlibV3 文件将包含新更改。

更改库时，通过“创建库”(Create Library) 对话框更新版本和名称（如果必要）。

说明

如果不更改 STEP 7-Micro/WIN SMART V3 存储库文件或库文件名称的位置，则 STEP 7-Micro/WIN SMART V3 会用新版本的库覆盖现有版本，基本上就清除了现有版本。更改库版本时，应更改库文件的名称，或更改存储库的文件夹。

使用版本检查功能

当您分发库的更新版本供您的用户使用时，现有项目不会“自动”更新为新版本。

为了帮助用户在不同版本的指令库之间进行转换，STEP 7-Micro/WIN SMART V3 提供了版本检查功能。右键单击库并从上下文菜单中选择“检查版本”(Check version) 可执行版本检查。

版本检查功能可查看当前项目中使用的每条库指令。如果包含的库具有一个不同的指令版本，则会采取以下两项措施之一：

- 如果指令的两个版本的签名相同，则 STEP 7-Micro/WIN V3 将使用新版本库的子程序对构成当前指令的子例程/功能块予以更新。如果出现这种情况，STEP 7-Micro/WIN SMART V3 将自动对项目进行更新，无需用户进行任何手动操作。
- 如果指令的两个版本的签名不匹配，则无法进行自动更新。在这种情况下，STEP 7-Micro/WIN SMART V3 会从项目中删除子例程/功能块。如果出现此种情形，用户必须用对新版本指令的调用替代对该指令的调用。这会将正确版本的子例程/功能块添加至项目。

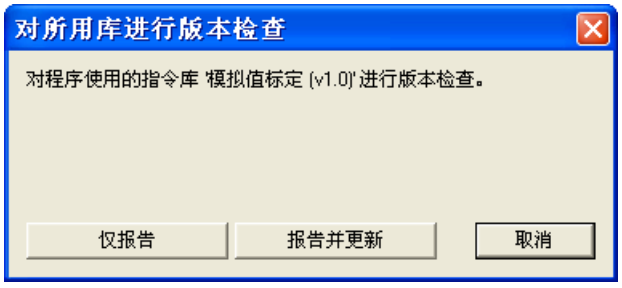
如果新版本需要的存储器分配情况不同，则 STEP 7-Micro/WIN SMART V3 将从现有的起始 V 存储器地址分配所需存储器。您可根据需要随时更改库存储器分配。

更新项目以使用最新版本的库

要更新现有项目以使用最新版本的库：

1. 添加新版本的库后，在项目树“指令”(Instruction) 文件夹的“库”(Libraries) 文件夹中右键单击库名称。
2. 在上下文菜单中，单击“检查版本”(Check Version)。

仅当当前项目使用该库时，“检查版本”(Check Version) 上下文菜单命令才可用。
选择该选项会载入以下对话框：



3. 在该对话框中，您可以选择对当前项目执行版本更新，或只扫描当前项目以便确定需要进行哪些操作。
   - **“只报告”(Report Only)**：“只报告”(Report Only) 选项对程序进行扫描，以便确定在决定将项目更新至最新库版本时应当作出何种更改。STEP 7-Micro/WIN SMART V3 将在输出窗口中显示报告，并且不会对项目进行任何更改。
   - **“报告并更新”(Report and Update)**：“报告并更新”(Report and Update) 选项对程序进行扫描，以便确定需要作出何种更改以及如何执行这些更改。在执行更新之前，首先选择“只报告”(Report Only)，并仔细检查 STEP 7-Micro/WIN SMART V3 将对项目进行的更改。另外，在执行更新之前，应保存项目备份副本。

11.10 如何重新加载库

要将配置的库文件夹中的库重新加载到项目，右键单击项目树的库节点，并从快捷菜单中选择“刷新库”(Refresh Libraries)。

STEP 7-Micro/WIN SMART V3 从库文件夹及其子文件夹中寻找任何库文件（.smartlibV3 文件）并进行加载。随后，您可以看到它们列在项目树的库节点下。

说明

如果从当前打开的项目中创建库，STEP 7-Micro/WIN SMART V3 不会将这些库加入到项目树的“库”(Libraries) 节点中。“刷新库”(Refresh Libraries) 命令加载库文件夹中的库，但对应于当前项目实例的库除外。

另请参见

应用程序用户参考手册

入门指南

用户定义库示例 (页 763)

11.11 用户定义库示例

11.11.1 用于缩放模拟值的库示例

您可创建库来保存程序逻辑，您可能要在多个项目中使用该程序逻辑来执行通用功能。 此示例显示了如何使用库来存储用于缩放模拟值的子例程。

用于缩放模拟值的库子例程

创建库之前，先创建一个包含相应子例程的项目，该子例程可执行您希望最终包含在库中的通用功能。 此示例使用三个执行模拟值缩放例程的子例程。 每个子例程又执行不同类型的缩放功能：

- Scale_I_to_R (S_ITR) (页 755)： 该子例程从整数范围向浮点范围缩放模拟值。 初始（整数）值是子例程的输入。 缩放的浮点值是子例程的输出。 输入的整型范围上下限以及输出的浮点型范围上下限均为该子例程的输入。
- Scale_R_to_I (S_RTI) (页 758)： 该子例程从浮点范围向整数范围缩放模拟值。 初始（浮点）值是子例程的输入。 缩放的整数值是子例程的输出。 输入的浮点型范围上下限以及输出整型范围上下限均为该子例程的输入。
- Scale_R_to_R (S_RTI) (页 761)： 该子例程从浮点范围向另一浮点范围缩放模拟值。 初始值（符合输入浮点范围）是该子例程的输入。 子例程的输出是缩放至输出范围的模拟值。 输入的浮点型范围上下限以及输出浮点型范围上下限均为该子例程的输入。

11.11.2 子例程 S_ITR：整数转换为实数

将整数缩放为实数

此子例程的用途是将输入范围中的整数缩放为输出范围中的实数。

此子例程首先会读取由调用 POU 传递的值： 要缩放的输入、输入范围上限 (ISH)、输入范围下限 (ISL)、输出范围上限 (OSH) 和输入范围下限 (OSL)。

下一步是用 OSH 中减去 OSL，以确定存回 OSH 中的输出缩放范围。 接下来，该子例程会从输入中减去 ISL，并将所得的差先转换为双字，然后再转换为实数。 之后，下一条指令将该实数乘以输出范围。

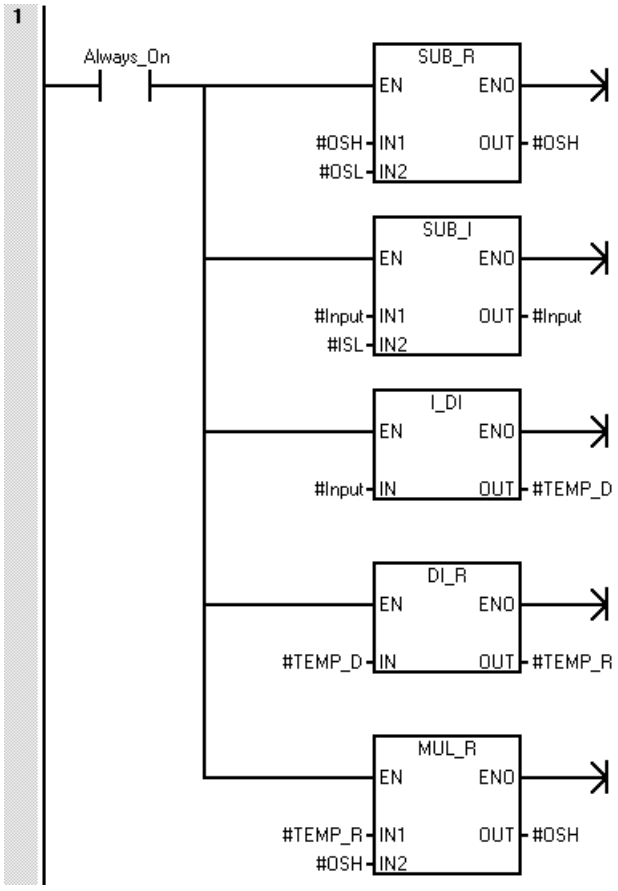
再下一步是用 ISH 减去 ISL，以计算输入缩放范围，将计算所得的值转换为双字，并再转换为实数。

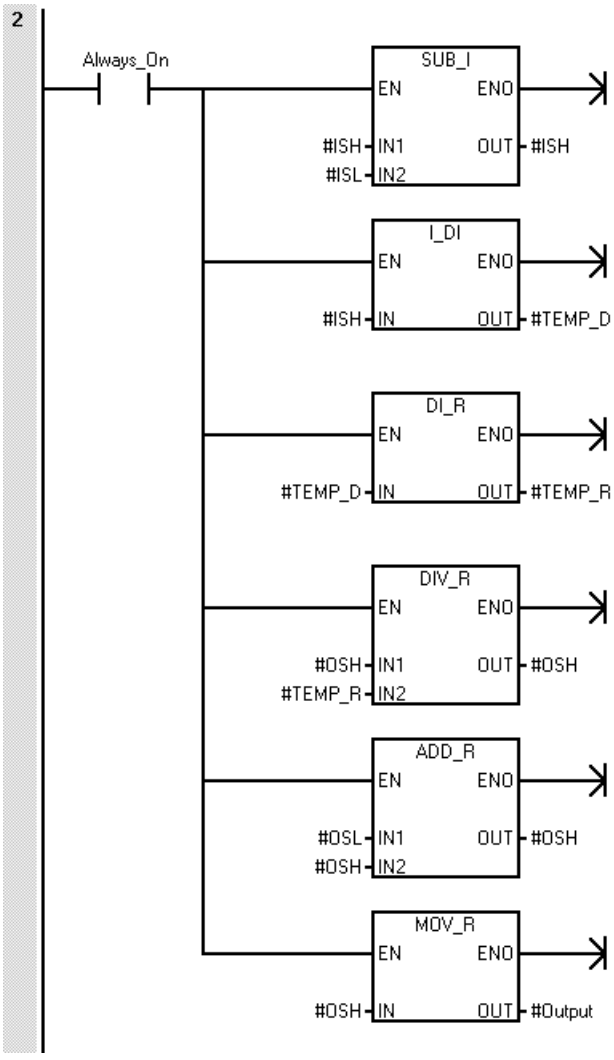
接下来，该子例程用输出缩放范围除以计算所得的实数，并将结果与输出范围下限值 (OSL) 相加，以确定最终输出值。 最后，该子例程会将此值移动到要由调用 POU 使用的子例程的输出。

公式如下：

输出 = [(OSH - OSL) * (输入 - ISL) / (ISH - ISL)] + OSL

变量	变量类型	数据类型	定义
Output	OUT	REAL	已缩放输出值
Input	IN	INT	模拟量输入值
OSH	IN	REAL	已缩放输出值的范围上限
OSL	IN	REAL	已缩放输出值的范围下限
ISH	IN	INT	模拟量输入值的范围上限
ISL	IN	INT	模拟量输入值的范围下限


1
Always_On
SUB_R
EN
ENO
#OSH
IN1
OUT
#OSH
#OSL
IN2
SUB_I
EN
ENO
#Input
IN1
OUT
#Input
#ISL
IN2
I_DI
EN
ENO
#Input
IN
OUT
#TEMP_D
DI_R
EN
ENO
#TEMP_D
IN
OUT
#TEMP_R
MUL_R
EN
ENO
#TEMP_R
IN1
OUT
#OSH
#OSH
IN2



11.11.3 子例程 S_RTI：实数转换为整数

将实数缩放为整数

此子例程的用途是将输入范围中的实数缩放为输出范围中的整数。

此子例程首先会读取由调用 POU 传递的值： 要缩放的输入、输入范围上限 (ISH)、输入范围下限 (ISL)、输出范围上限 (OSH) 和输入范围下限 (OSL)。

下一步是用 OSH 中减去 OSL，以确定存回 OSH 中的输出缩放范围。 接下来，该子例程会从输入中减去 ISL，并将所得的差先转换为双字，然后再转换为实数。

下一步是用输入减去 ISL，然后用所得的差乘以计算所得的实数输出缩放范围。

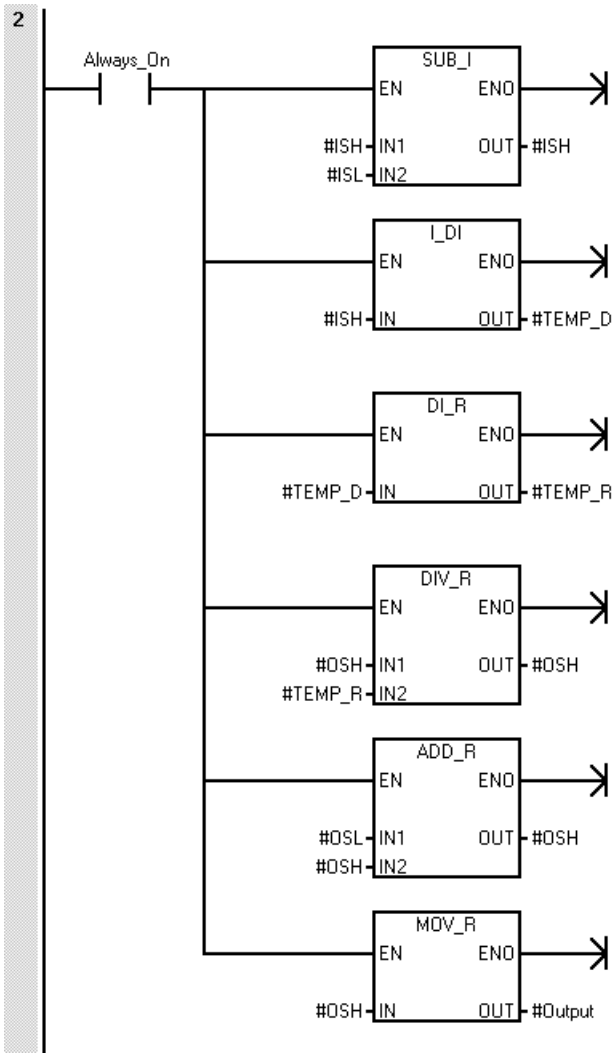
再用 ISH 减去 ISL，以计算输入缩放范围，用输出范围除以计算所得的输入锁定范围，然后将商舍入为双字。

之后，该子例程将根据该数据类型的界限限制检查已舍入的输出值，以确保最终值的准确性。在界限检查后，子例程会将双字值转换为整数，用该整数加上输出范围下限 (OSL)，并将所得值移动到子例程输出。

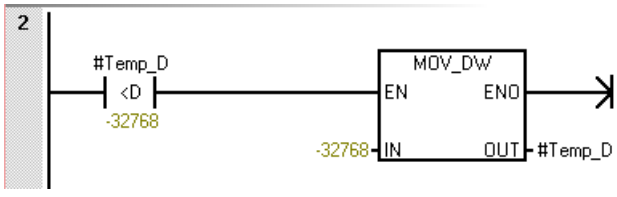
公式如下：

输出 = [(OSH - OSL) * (输入 - ISL) / (ISL - ISL)] + OSL

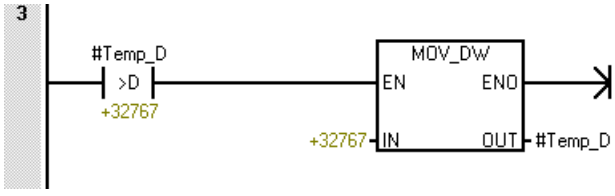
变量	变量类型	数据类型	定义
Output	OUT	INT	已缩放输出值
Input	IN	REAL	模拟量输入值
OSH	IN	INT	已缩放输出值的范围上限
OSL	IN	INT	已缩放输出值的范围下限
ISH	IN	REAL	模拟量输入值的范围上限
ISL	IN	REAL	模拟量输入值的范围下限



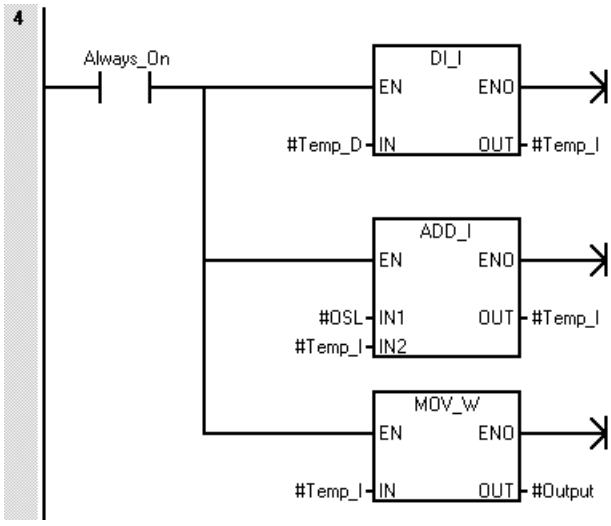
程序段 2 检查 Temp_D 是否小于 -32768



程序段 3 检查 Temp_D 是否大于 32767



程序段 4 对整数值进行转换，并加上下限偏移量



11.11.4 子例程 S_RTR： 实数转换为实数

将实数缩放为实数

此子例程用于将输入范围内的标准化模拟量输入缩放到输出范围内。

此子例程首先会读取由调用 POU 传递的值： 要缩放的输入、输入范围上限 (ISH)、输入范围下限 (ISL)、输出范围上限 (OSH) 和输入范围下限 (OSL)。 所有输入都为实数。

下一步是用 OSH 减去 OSL，以决定输出缩放范围。 之后，子例程会用输入减去 ISL，并用所得的差乘以输出范围。

下一步是用 ISH 减去 ISL，以确定输入缩放范围。

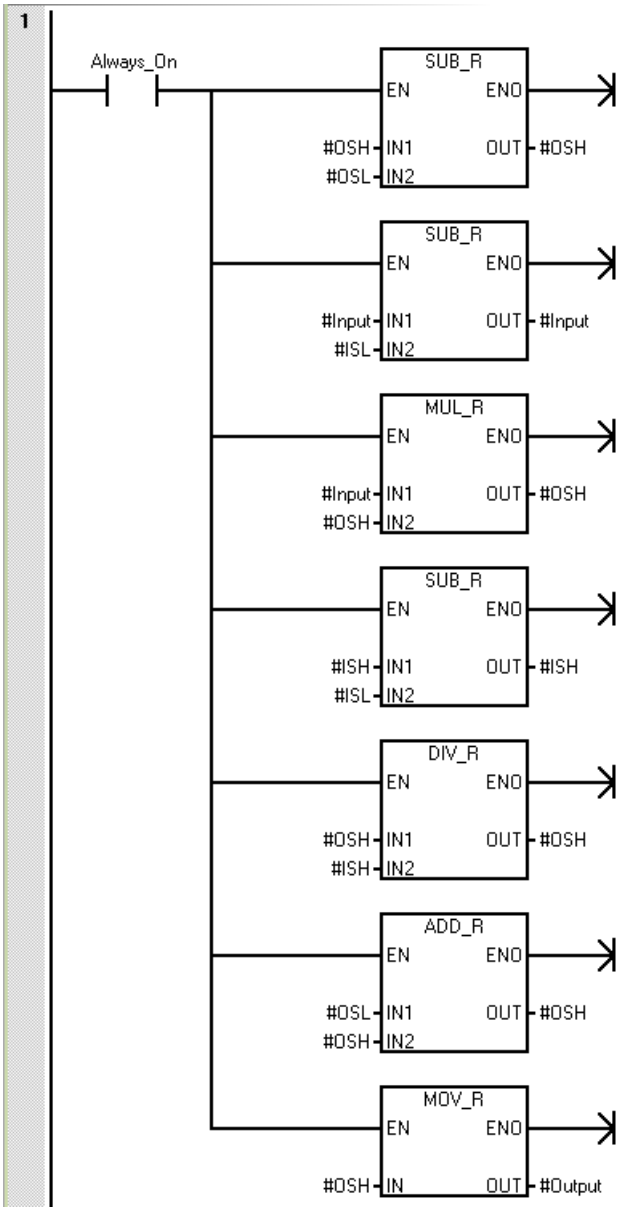
此时，子例程用该输出范围除以取消缩放的输入，以确定最终输出值，然后将该值移动至调用 POU 将使用的子例程输出。

公式如下：

输出 = [(OSH - OSH) * (输入 - ISH) / (ISH - ISL)] + OSL

变量	变量类型	数据类型	定义
Output	OUT	REAL	已缩放输出值
Input	IN	REAL	模拟量输入值
OSH	IN	REAL	已缩放输出值的范围上限

变量	变量类型	数据类型	定义
OSL	IN	REAL	已缩放输出值的范围下限
ISH	IN	REAL	模拟量输入值的范围上限
ISL	IN	REAL	模拟量输入值的范围下限



11.11.5 创建用户定义库

现在，您可以通过示例模拟量缩放程序创建包含三个缩放子例程 S_ITR (页 755)、S_RTI (页 758) 和 S_RTR (页 761) 的库。

在项目中创建指令库

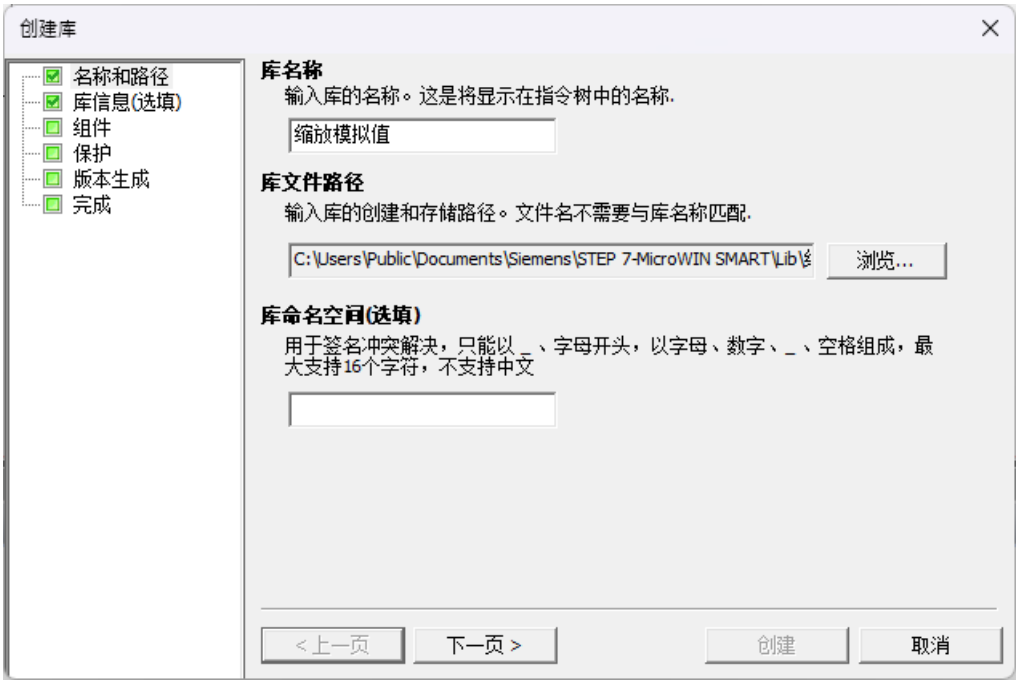
要通过模拟量缩放子例程创建库，请按以下步骤操作：

1. 在“文件”(File) 菜单功能区的“库”(Libraries) 区域中，单击“创建”(Create) 按钮 创建。
2. 在“名称和路径”(Name and Path) 节点中，输入下列信息。
   - 库名称：输入库名称。例如，为库提供名称“Analog_Scaling”。
     库名称可以包含空格和大小写混合字母。STEP 7-Micro/WIN SMART V3 将库添加到项目中后，在此处应用的名称会立即显示在项目树中。
   - 库文件路径：默认情况下，STEP 7-Micro/WIN SMART V3 使用您在项目选项中配置的默认库文件夹。但是，可以使用“浏览”(Browse) 按钮导航并选择文件夹。如果希望其它项目访问该库，使用您在项目选项中配置的文件夹。
   - 库命名空间：库命名空间用于解决在项目中导入多个库时可能出现的程序组织单元（POU）和用户自定义数据类型（UDT）的签名冲突问题。通过为库指定唯一的命名空间，可以确保其内部元素的名称在全局范围内是唯一的。命名空间功能仅适用于库中的程序组织单元（POU）和用户自定义数据类型（UDT）。命名空间仅支持单层结构，不支持嵌套子命名空间。
     如果为库设置了命名空间，则导入该库的 POU 和 UDT 将以“命名空间.POU/UDT”的形式显示。您在用户代码中访问这些元素时，必须使用完整的路径。

说明

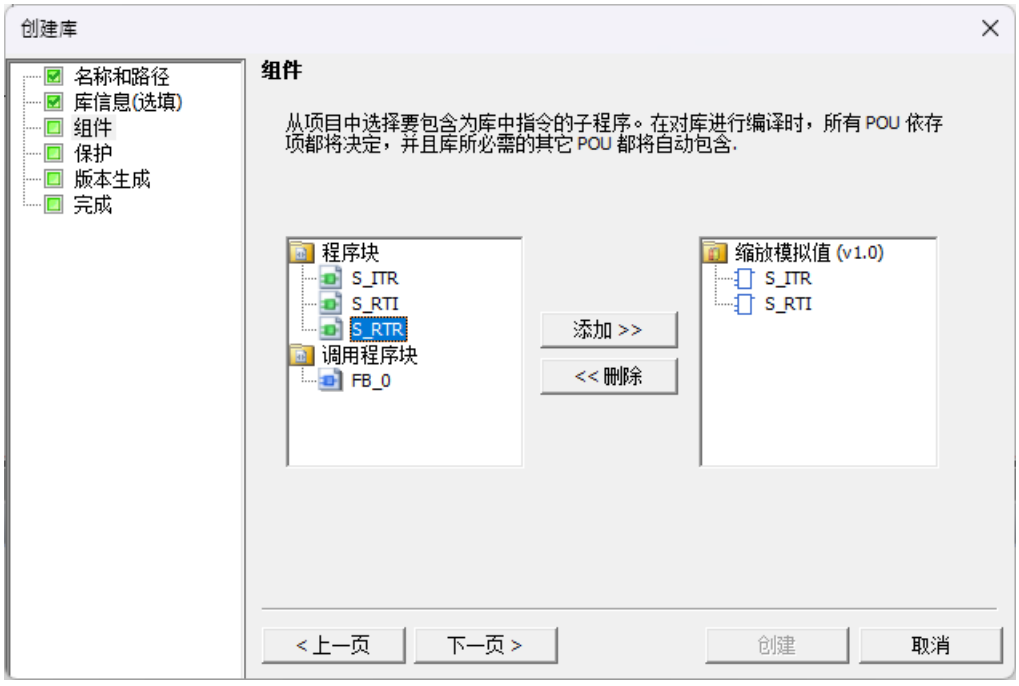
命名空间名称必须遵循以下规则：

- 只能由字母（a-z, A-Z）、数字（0-9）和下划线（_）组成
- 必须以字母或下划线开头
- 长度最多为 15 个字符
- 不支持中文

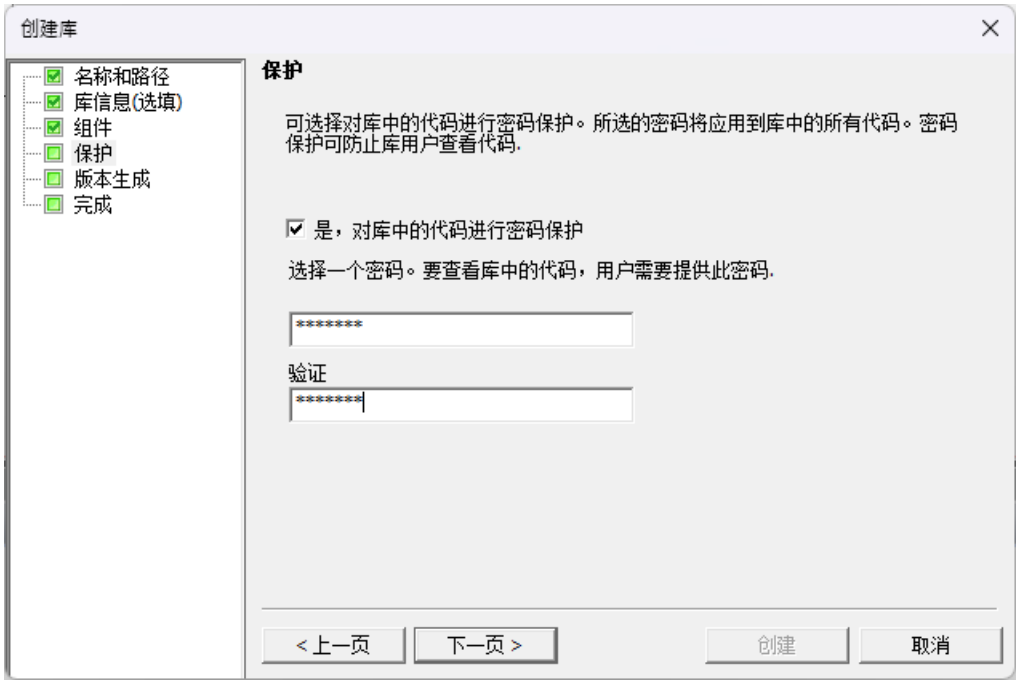


3. 在“库信息”(Libraries information) 节点中，输入库信息。

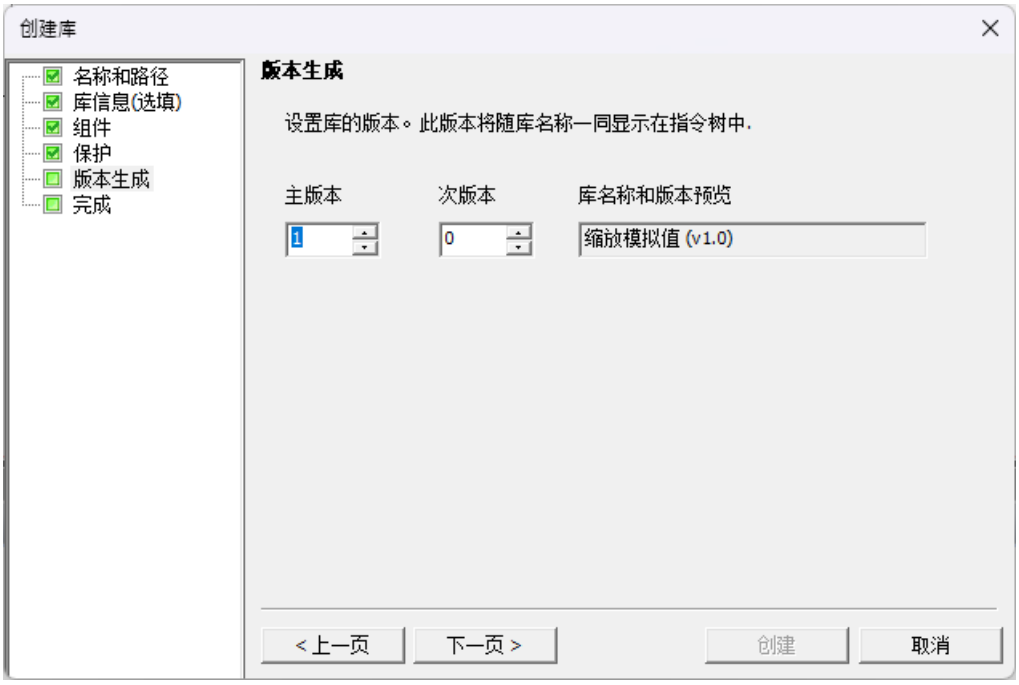
4. 在“组件”(Components) 节点中，分别选择三个缩放子例程，并逐一单击“添加”(Add) 按钮。



5. 如果要用密码来保护您的库，可使用“保护”(Protection) 选项卡。选中该复选框，输入并确认密码，要求用户输入密码后才能对库中的代码进行查看或修改。如果库用户未输入密码，则该用户只能看到其中存在哪些库子例程以及各子例程对应哪些变量表。



6. 在“版本生成”(Versioning) 节点中，可以提供版本号，这在同一个库具有不同版本的情况下非常有用。



7. 完成后，单击“完成”(Completion) 节点中的“创建”(Create)。STEP 7-Micro/WIN SMART V3 会对您的库子例程进行编译，并在输出窗口中显示任何错误。若存在错误，应将其更正并重复此过程来创建库。库成功编译后，STEP 7-Micro/WIN SMART 会创建该库并将其保存到您指定的位置。

说明

依赖管理

当从一个包含依赖库的项目创建新库时，所有被引用的依赖库内容按依赖关系顺序导出。当存在相互依赖关系时且无冲突时，则进行内容合并；如存在冲突，则进行命名空间整体替换。

用户库创建时，系统默认会将所有依赖库的内容（包括 UDT、SBR、FB 及库变量表）一并打包导出。当此库被引入新项目后，其自身内容及所有依赖库的 UDT、SBR、FB 和库变量表将同时存在于新项目中。如需分配 V 区地址，必须通过库管理器为所有相关库存储区进行 V 区地址分配。

11.11.6 在项目中使用库

创建库 (页 763)后，该库可用于其它项目。新项目显示项目树的“指令”(Instructions) 文件夹中库节点下的所有库。对于现有项目，可以刷新库文件夹 (页 754)以显示您在项目选项中配置的库文件夹中的现有库。

请注意，库名称和版本与您创建该库时所输入的内容相同。您选择的子例程在库中作为指令显示。

调用库子例程

加入了库之后，您就可以通过从指令树拖放的方式来放置库指令，或在 LAD 和 FBD 中通过“插入框 (F9)”(Insert Box (F9)) 列表来放置库指令。

要插入对 Analog_Scaling 库中的 S_ITR（将整数缩放为实数）的调用，请按以下步骤操作：

1. 在程序编辑器窗口中打开 POU，并滚动至您希望插入调用的程序段。
2. 展开项目树中“指令”(Instructions) 文件夹下的“库”(Libraries) 文件夹。您可以从指令树拖动适当的库指令，并将指令放置在程序编辑器中的适当程序段单元格中，或将光标放在程序编辑器的单元格上，然后双击指令树中的库指令。在 LAD 和 FBD 中，您还可按下 F9 或在程序编辑器中单击“插入框”(Insert Box) 按钮 。
   插入对库指令的调用之后，STEP 7-Micro/WIN SMART V3 会向项目中添加一个或多个 POU。请记住，库指令实际上是库包含的子例程。当您在项目中插入对某一库指令的首次调用时，STEP 7-Micro/WIN SMART V3 会添加相应的 POU；但当您插入后续调用时，便不会添加 POU，这是因为所需子例程已成为您项目的一部分。
   您会注意到，您的项目中已经自动添加了新的 POU (S_ITR)。该 POU 在项目树中出现，在程序编辑器中有一个选项卡。您放置的指令现在成为对该子例程的调用。
3. 编辑程序中的调用指令的参数并为每个参数指定有效操作数。您可查看 S_ITR 子例程的变量表，以查找各个参数的数据类型。

库缩放指令的使用示例

设想一个盛放液体的封闭容器。

压差传感器会在模拟量输入模块的输入端提供电流（0 mA 到 20 mA）。该电流与容器中的液位成正比。

该模拟模块必须校准为，在 10 m 液位处，模拟电流值 20 mA 转换为值 32000。在 0 m 液位处，模拟电流值 0 mA 必须

对应值 0。程序会将转换所得的数字值标定为以米为单位的高度值。

对于液位指示灯，相应的电压须由模拟量输出模块生成。此电压的生成方式为在模拟量输出字 (AQW) 处写入对应的整数值。

模拟量输出模块以 0V 到 10V 电压的形式向测量仪表提供液位情况（介于 0 m 和 10 m 之间）。测量仪表的模拟显示屏会对电压作出响应，其指针偏移量与液位成正比。

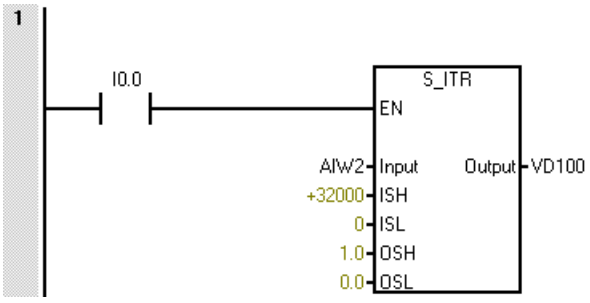
此公式可缩放介于任意给定最大和最小缩放点之间的任意给定值。该程序还会特别接收到模拟量输入字 (AIW)，并会在缩放后将其发送到模拟量输出模块。首先，程序会读取 AIW（该

值介于 0 mA 和 20 mA 之间，即 0 和 32000 之间），并会将其缩放为标准值（即介于 0.0 和 1.0 之间的值）。然后该值将被缩放为 0.0 到 100.0 和 0 到 32000。

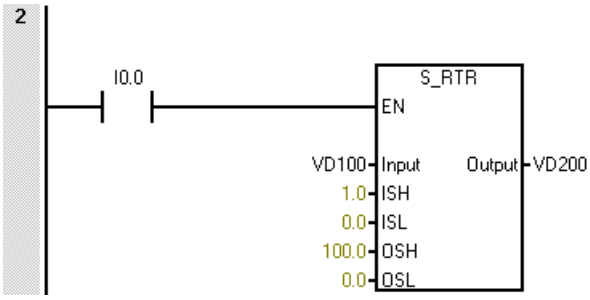
变量	定义
输出	已缩放输出值
输入	模拟量输入值
OSH	已缩放输出值的范围上限
OSL	已缩放输出值的范围下限
ISH	模拟量输入值的范围上限
ISL	模拟量输入值的范围下限

缩放值计算公式：输出 = [(OSH - OSL) * (输入 - ISL) / (ISH - ISL)] + OSL

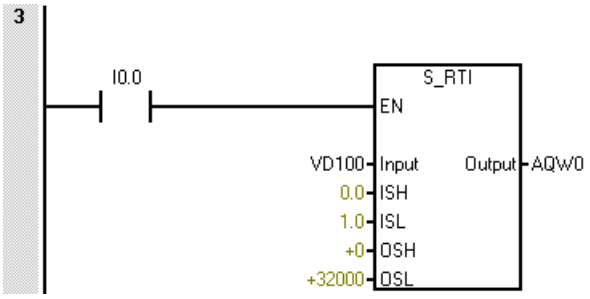
程序段 1：读取模拟量输入并将范围标准化为 0.0 至 1.0。



程序段 2：将标准化值缩放为范围 0.0 到 100.0（百分数形式）



程序段 3：将标准化值缩放至范围 0 到 32000，并将其传递到模拟量输出



11.11.7 更改库版本

经过一段时间后，您可能需要对原有库进行更改或需要创建新版本。

说明

更改库版本时，需要保持库名称和命名空间与原版本一致，可更改库文件的名称，或使用其它文件夹存储之前的库。如果您未更改库文件的名称或位置，则 STEP 7-Micro/WIN SMART V3 在保存新版本的库时会覆盖掉原有版本。之后，原始库将无法使用。

要创建现有库的新版本，请按以下步骤操作：

1. 如果选择，将您在项目选项中配置的库文件夹中的原始库复制到另一个位置作为备份。
2. 打开从中创建库的原始项目。这个项目是包含库子例程代码的项目，而不是调用库指令的项目。
3. 进行必要的更改。
4. 在“文件”(File) 菜单功能区的“库”(Libraries) 区域中，单击“创建库”(Create library) 按钮 创建。
5. 按照与创建库 (页 763)相同的步骤进行操作，但为该库提供一个新版本号，或者可能要提供一个新名称。

调试和故障排除 12

STEP 7-Micro/WIN SMART V3 提供了多个软件工具来帮助调试和测试程序。这些功能包括：查看 CPU 正在执行的程序的状态、为 CPU 指定运行程序的扫描次数以及强制值。

请使用硬件故障排除指南 (页 782)作为排除硬件故障时确定原因和可能解决方案的指导。

12.1 调试程序

12.1.1 书签功能

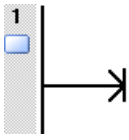
可在程序中设置书签 ，以便于在长程序中的指定程序段间来回移动：

切换书签： 单击该按钮可在当前光标位置指定的程序段处设置或删除书签。

下一个书签： 单击此按钮将移动到程序中下一个标有书签的程序段。

前一个书签： 单击此按钮将移动到程序中上一个标有书签的程序段。

删除所有书签： 单击此按钮可删除程序中的所有书签。



12.1.2 交叉引用表

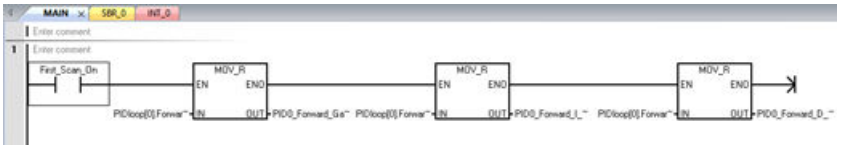
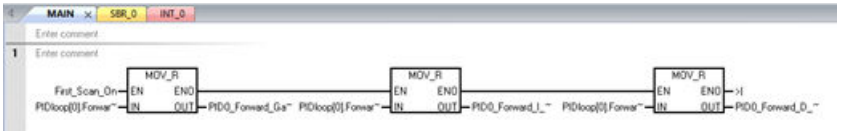
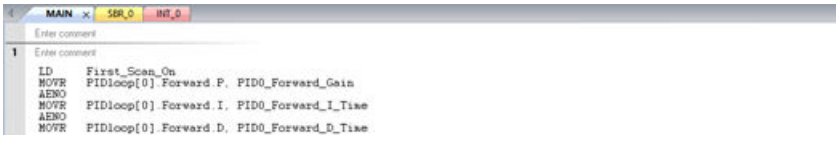
若要了解程序中是否已经使用以及在何处使用某一变量名或存储器分配，使用交叉引用表。交叉引用表标识在程序中使用的所有操作数，并标识 POU (页 151)、程序段或行位置以及每次使用操作数时的指令上下文。在交叉引用表中双击某一元素可显示 POU 的对应部分。

检查元素的交叉引用

1. 打开交叉引用表
2. 选择要在程序中查询的操作数。 可使用保持按钮 保持当前检查的元素。

示例

以下示例以所有三种语言显示一个简单程序的交叉引用表：LAD、FBD 和 STL。

语言	程序
交叉引用	 ① 刷新 ② 保持引用结果 ③ 使用操作数的 POU ④ 使用操作数的行或程序段
LAD	
FBD	
STL	

12.1.3 字节使用表

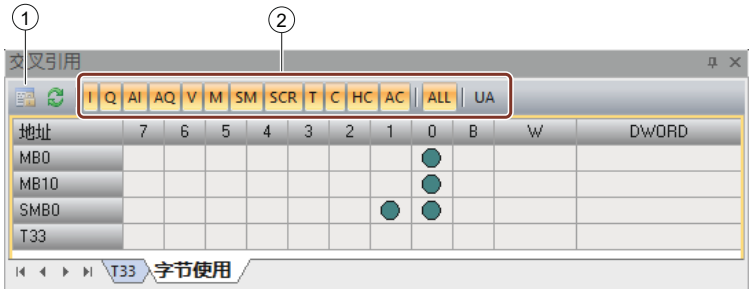
交叉引用表 (页 771)的字节使用表允许用户查看程序使用了哪些存储器地址中的哪些字节（精确到具体某一位）。它还有助于识别重复赋值错误。

- **地址**表示存储器中的地址。
- **0...7** 表示已分配存储器的位。

- **B** 表示已分配存储器的一个字节。
- **W** 表示已分配一个字（16 位）。
- **DWORD** 表示已分配一个双字（32 位）。

示例 1：字节使用表解读

这个字节使用表示例显示与其关联程序使用以下存储位置：VB0 中的一个字节；AIW 16；AQW 16；AQW 18；双字 VB0 至 VB3；双字 VB4 至 VB7。



① 刷新　　② 存储器类型

12.2 显示程序状态

12.2.1 程序编辑器如何显示状态

要在程序编辑器中显示当前数据值和 I/O 状态，需单击程序编辑器工具栏中的“程序状态”(Program Status) 开/关按钮，或单击“调试”(Debug) 菜单功能区中的 程序状态。

状态数据采集随后开始，并在执行程序过程中显示所有逻辑运算的结果。也可单击程序编辑器工具栏中的“暂停状态”(Pause Status) 开关按钮，或单击“调试”(Debug) 菜单功能区中的按钮 暂停状态，暂停和恢复程序状态采集。

状态图表 (页 777)将在扫描结束时显示值。

执行状态颜色指示

- 扫描程序时，电源线 (LAD) 会变色显示。
- 图示中的能流或逻辑流会变色显示。

- 通电或在逻辑上为真的触点和线圈 (LAD) 标为蓝色。
  从“工具”(Tool) 菜单功能区的“选项”(Options) 设置中选择“颜色”(Colors) 选项卡，您可自定义颜色选项。
- 方框指令 - 指令通电且无错成功执行时，方框指令标有颜色。
- 绿色定时器和计数器表示定时器和计数器包含有效数据。
- 红色表示指令执行时发生错误。
- 跳转和标签指令激活时，显示为能流的颜色。如果未激活，则显示为灰色。
- 灰色（默认分配）表示无能流 – 指令未扫描（跳过或未调用）或 PLC 处于 STOP 模式。
- 布尔能流位（仅限 FBD）为蓝色。
- LAD、FBD 和 STL 程序编辑器在扫描周期的执行程序阶段随着每条指令的执行，显示操作数的值并指示能流状态。执行状态能够显示中间数据值，它们可能因执行后续程序指令而被覆盖。所有显示的 PLC 数据值都是从一个程序扫描周期中采集的。

STL 程序中的程序状态示例

在 STL 中启动程序状态监视时，程序编辑器窗口划分成代码区和状态区。可根据要监视的值类型自定义状态区。

在 STL 状态中可监视三种值类别：

操作数	每条指令最多可监视 64 个操作数。
逻辑堆栈	最多可监视逻辑堆栈中的四个最近值。
指令状态位	最多可监视十一个状态位。

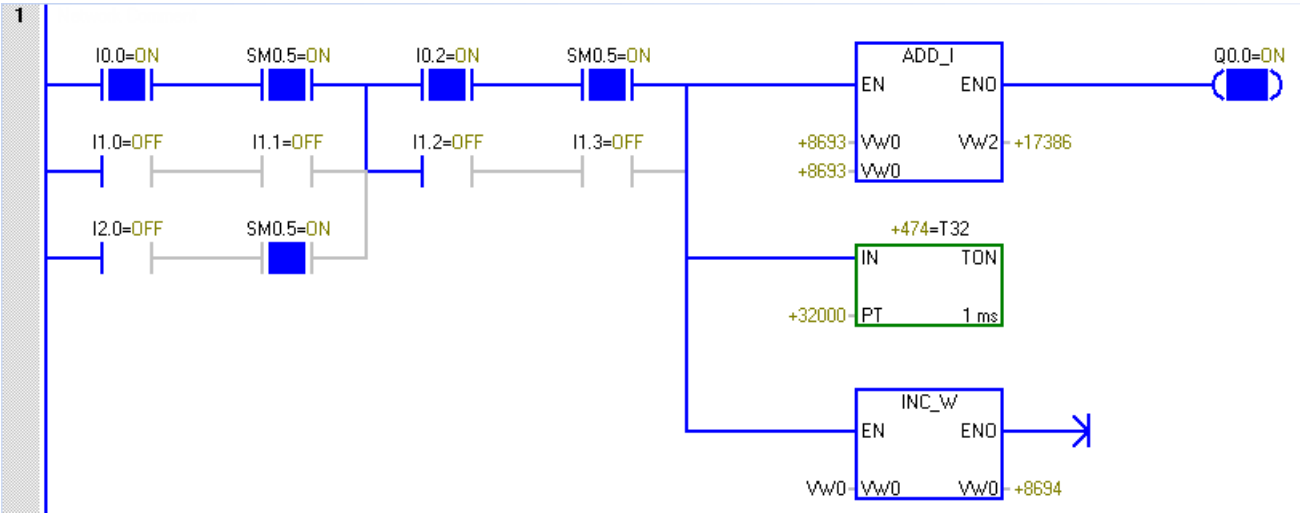
在“工具”(Tools) 菜单功能区的“选项”(Options) 设置中，通过“程序编辑器”(Program Editor) 选项的“STL 状态”(STL Status) 选项卡 (页 776)可以选择或取消选择这些值类别。如果取消选择某一项，它就不会出现在状态显示中。

1

		Op 1	Op 2	Op 3	0123	
LD	I0.0	ON			1000	1
A	SM0.5	ON			1000	1
LD	I1.0	OFF			0100	0
A	I1.1	OFF			0100	0
OLD					1000	1
LD	I2.0	OFF			0100	0
A	SM0.5	ON			0100	1
OLD					1000	1
LD	I0.2	ON			1100	1
A	SM0.5	ON			1100	1
LD	I1.2	OFF			0110	0
A	I1.3	OFF			0110	0
OLD					1100	1
ALD					1000	1
LPS					1100	1
MOVW	VW0, VW2	+8525	+8525		1100	1
AENO					1100	1
+I	VW0, VW2	+8525	+17050		1100	1
AENO					1100	1
=	Q0.0	ON			1100	1
LRD					1100	1
TON	T32, +32000	+294	+32000		1100	1
LRD					1100	1
INCW	VW0	+8526			1100	1
LRD					1100	1
INCW	VW1000	+8526			1100	1

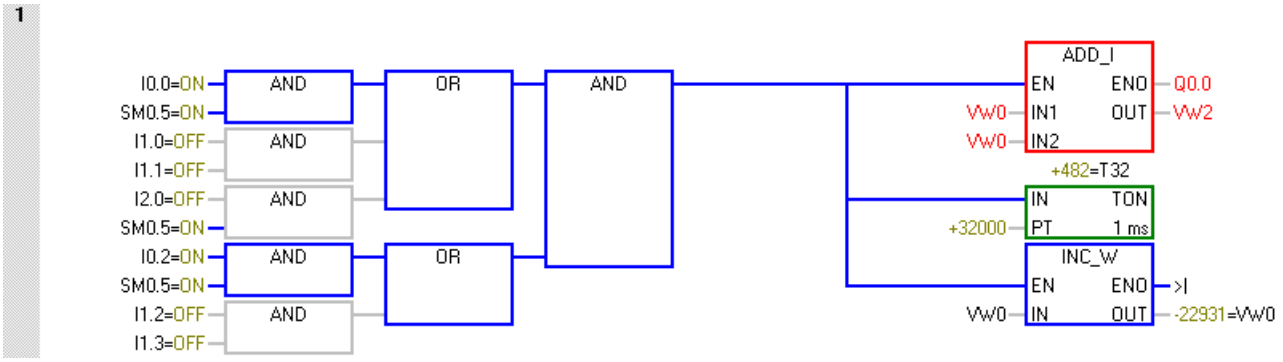
LAD 程序编辑器中的程序状态示例

下面所示为 LAD 程序编辑器中的状态。程序编辑器在扫描周期的执行程序阶段随着每条指令的执行，显示操作数的值并指示能流状态。



FBD 程序编辑器中的程序状态示例

下面所示为 FBD 程序编辑器中的状态。红色 ADD_I 指令框指示操作数中有错误。



12.2.2 组态 STL 状态选项

要组态 STL 程序状态显示选项，请按以下步骤操作：

1. 在“工具”(Tools) 菜单功能区的“设置”(Settings) 区域单击“选项”(Options) 按钮。

2. 单击“选项”(Options) 下的“程序编辑器 > STL > 状态”(Program Editor > STL > Status)。
3. 对下列 STL 程序状态选项进行组态：

类型 (Type)	选择 STL 程序状态文本的字体类型。
样式 (Style)	选择“常规”(Regular)、“斜体”(Italic)、“粗体”(Bold) 或“粗斜体”(Bold Italic) 的文本样式。
大小 (Size)	选择字体的点大小。
监视值	可借助这些复选框和选择框在“程序状态”(Program Status) 显示中包括或移除操作数、堆栈值和指令状态位（即标志）。
操作数数目 (Number of operands)	如果选择在“程序状态”显示中包括操作数，则可编辑“操作数”(Operands) 列表框，显示更多或更少的操作数。允许的最大数目为 64。

堆栈位数 (Number of Stack Bits)	如果选择在“程序状态”显示中包括逻辑堆栈，则可编辑“逻辑堆栈”(Logic Stack) 列表框，显示更多或更少堆栈值。允许使用的最大数目为四。
指令状态位	如果选择在程序状态显示中包括指令状态位，则选择要显示或省略（如果有）的“指令状态位”(Instruction Status Bits)。 复选标记指示您选择在程序状态显示中监视的特定状态位；如果取消选择该复选框，则 STEP 7-Micro/WIN SMART 的“程序状态”(Program Status) 中不会显示状态位。

另请参见

如何在程序编辑器中显示状态 (页 773)

12.3 使用状态图来监视程序

在状态图表中，可输入地址或已定义的变量，通过显示当前值来监视或修改程序输入、输出或变量的状态。通过状态图表还可强制或更改过程变量的值。可以创建多个状态图表，以查看程序不同部分中的元素。

可以将定时器和计数器值显示为位或字。如果将定时器或计数器值显示为位，则会显示指令的输出状态（0 或 1）。如果将定时器或计数器值显示为字，则会显示定时器或计数器的当前值。

创建新图表

要创建新的状态图表，请确保“状态图表”(Chart Status) (页 773) 和“程序状态”(Program Status) (页 773) 处于关闭状态，然后使用以下方法之一创建新图表：

- 在项目树 (页 160)中，右键单击“状态图表”(Status Chart) 文件夹，然后选择上下文菜单命令**“插入 > 图表”(Insert > Chart)**。
- 在“编辑”(Edit) 菜单功能区的“插入”(Insert) 区域，单击“对象”(Object) 下方的下拉箭头，然后从下拉菜单中选择“图表”(Chart)。
- 从状态图表编辑器的状态图表选项卡，或从现有状态图表中的任何单元，右键单击并选择上下文菜单命令**“插入 > 图表”(Insert > Chart)**。
- 在状态图表工具栏中单击“插入”(Insert) 按钮。

成功插入新的状态图表后，新图表将显示在项目树中的“状态图表”(Status Chart) 下，新选项卡显示在“状态图表”(Status Chart) 窗口底部。

打开现有图表

如果状态图表编辑器并未打开，您可从项目树、导航栏或从“视图”(View) 菜单的“窗口”(Windows) 区域中的“组件”(Component) 下拉列表打开现有状态图表。如果状态图表编辑器已打开，您可以单击编辑器中的状态图表选项卡切换到该状态图表。

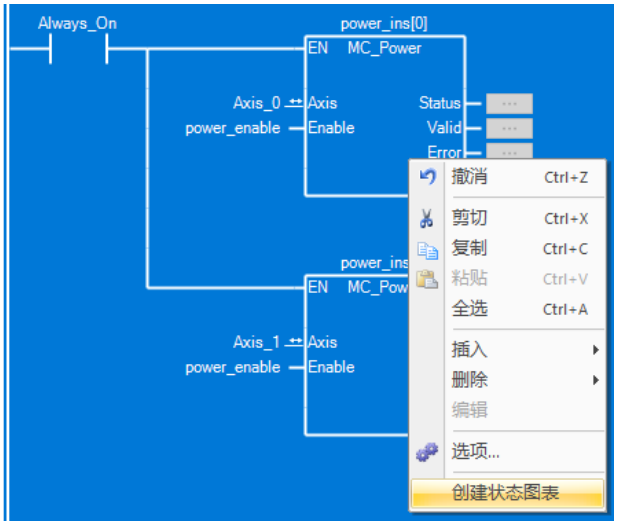
构建状态图表

要构建状态图表，请按以下步骤操作：

1. 在“地址”(Address) 字段中为每个所需值输入地址（或变量名）。变量名必须是已在全局变量表中定义的名称。
2. 如果元素是位（例如，I、Q 或 M），格式在“格式”(Format) 列中被设为位。如果元素是字节、字或双字，选择“格式”(Format) 列中的下拉列表，然后从可用选项中选择有效格式。
3. 要插入附加行，使用下列方法之一：
   – 在“编辑”(Edit) 菜单功能区的“插入”(Insert) 区域中，单击“行”(Row) 按钮。
   – 右键单击状态图表中的单元格以弹出上下文菜单，然后选择菜单命令**“插入 > 行”(Insert > Row)**。
   新行被插入在状态图表中光标当前位置的上方。也可将光标放在最后一行的最后一个单元格中，然后按向下箭头键，在状态图表的底部插入一行。

通过一段程序代码构建状态图表

在程序编辑器中高亮显示所选的程序段，单击右键，然后从上下文菜单中选择**“创建状态图表”(Create Status Chart)**。新图表针对可以采集状态的选择区域中的每个唯一操作数包含一个条目。STEP 7-Micro/WIN SMART 按条目在程序中出现的顺序放置条目，为图表指定默认名称，然后在状态图表编辑器中最后一个选项卡之后添加此图表。



通过程序编辑器创建图表时请注意，每次选择 **“创建状态图表”(Create Status Chart)** 都只添加前 4096 个地址。在 STEP 7-Micro/WIN SMART 创建状态图表之后，您就可以编辑图表条目。

还可以通过按住 Ctrl 键并将操作数从 LAD 或 FBD 程序编辑器拖动到状态图表的方式向状态图表添加条目。从 STL，您可以选择一个地址并将其拖动到状态图表。

说明

一个项目最多可存储 32 个状态图表。

将全局变量从全局变量表复制到状态图表

可从全局变量表复制地址或全局变量名，然后将其粘贴到状态图表，从而更快地构建图表。

12.4 强制特定值

强制特定值的规则如下：

- CPU 允许您强制任意或全部 I/O 点（I 和 Q 位）。
- 最多可强制 16 个存储器值（V 或 M）或模拟量 I/O 值（AI 或 AQ）
- 对于 PROFINET I/O 值，最多可强制 100 个字节。
- V 存储器、M 存储器或 PROFINET I/O 值可强制为字节、字或双字。
- 模拟量值只能按字形式进行强制，以偶数字节开始（例如 AIW6 或 AQW14）。所有强制值都存储在 CPU 的非易失性存储器中。

说明

不可通过 SD 卡传输 PROFINET 端口的强制信息。

因为扫描周期内强制数据可能会更改（通过程序、I/O 更新周期或通信处理周期），所以 CPU 会在扫描周期的不同时间重新应用这些强制值。

- *读取输入：*读取时，CPU 会将强制值应用到输入。
- *执行程序中的控制逻辑：*CPU 会将强制值应用到所有立即 I/O 访问。程序执行后，强制数据最多可用于 16 个存储器值。

- *处理任何通信请求*：CPU 将把强制值应用到所有读/写通信访问。
- *写入输出*：写入时，CPU 会将强制值应用到输出。

说明

强制功能会覆盖立即读取或立即写入指令。强制功能也会覆盖系统块 (页 211)中组态的 STOP 模式值。如果 CPU 进入 STOP 模式，则输出的是强制值，而不是为系统块中的输出组态的 STOP 模式值。

可以使用状态图表来强制值。

1. 要强制新值，在“状态图表”(Status Chart) 的“新值”(New Value) 列中输入值，然后单击“状态图表”(Status Chart) 工具栏中的“强制”(Force) 按钮，或右键单击“新值”(New Value) 列并从上下文菜单中选择“强制”(Force)。
2. 要强制现有值，在“当前值”(Current Value) 列中选择值，然后单击“状态图表”(Status Chart) 工具栏中的“强制”(Force) 按钮，或右键单击“当前值”(Current Value) 列中的值并从上下文菜单中选择“强制”(Force)。

状态 LED (页 145) 用于指示 CPU 是否有强制数据。

12.5 在 STOP 模式下写入和强制输出

要在 STOP 模式 (页 86)下启用“写入”(Write) 和“强制”(Force) 功能，在“调试”(Debug) 菜单功能区的“设置”(Settings) 区域单击“STOP 模式下强制”(Force in Stop) 按钮。

S7-200 SMART PLC 支持在 PLC 处于 STOP 模式时写入和强制输出（模拟量和数字量）。但作为一项安全防范措施，必须在 STEP 7-Micro/WIN SMART V3 中通过“STOP 模式下强制”(Force in Stop) 设置专门启用此功能。

警告

写入或强制输出对过程设备的影响

如果在写入或强制输出时已将 S7-200 SMART PLC 连接到过程设备，则 PLC 可将这些更改传送到该设备。这将导致设备内出现异常，进而导致人员死亡或重伤和/或财产损失。

仅当确保过程设备可以安全接受相关变更时，再执行写入和强制输出操作。

默认情况下，STEP 7-Micro/WIN SMART V3 不会启用“STOP 模式下强制”(Force in STOP)。PLC 处于 STOP 模式时，STEP 7-Micro/WIN SMART V3 会防止用户写入或强制输出。单击“调试”(Debug) 菜单中的“STOP 模式下强制”(Force in STOP) 按钮，会针对当前项目的当前编辑会话启用写入和强制功能。打开其他项目时，“STOP 模式下强制”(Force in STOP) 返回其默认状态，STEP 7-Micro/WIN SMART V3 会防止用户在 PLC 处于 STOP 模式时写入和强制输出地址。

状态 LED (页 145) 用于指示 CPU 在 STOP 模式下是否有强制数据。

12.6 如何执行有限次数的扫描

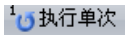
您可以指定 PLC 对程序执行有限次数扫描（从 1 次扫描到 65,535 次扫描）。 通过选择 PLC 运行的扫描次数，您可以在程序改变过程变量时对其进行监控。

第一次扫描时，SM0.1 的值为 1（打开）。

在执行单次扫描或多次扫描前，如果 PLC 尚未处于 STOP 模式，请将 PLC 更改为 STOP 模式 (页 145)。

执行单次扫描

要执行单次扫描，在“调试”(Debug) 菜单功能区的“扫描”(Scan) 区域单击“执行单次扫描”(Execute Single) 按钮。

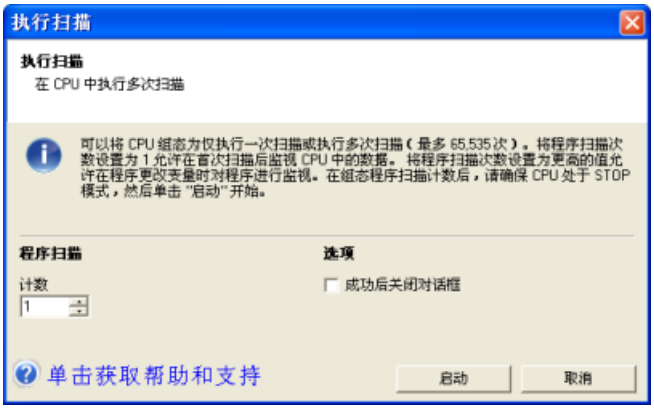


执行多次扫描

要执行多次扫描，请按以下步骤操作：

1. 在“调试”(Debug) 菜单功能区的“扫描”(Scan) 区域单击“执行多次扫描”(Execute Multiple) 按钮。

   出现“执行扫描”(Execute Scans) 对话栏。

2. 输入所需扫描次数值，然后单击“启动”(Start) 执行所输入的扫描次数。

说明

准备恢复正常的程序运行时，将 PLC 改回 RUN 模式 (页 145)。

另请参见

调试和监视功能概述 (页 771)

如何在编辑器窗口中显示状态 (页 773)

如何在状态图表中显示状态 (页 777)

如何下载程序 (页 133)

时间戳不匹配错误 (页 1257)（确保编程设备中的项目与 PLC 中的项目相匹配）

交叉引用和元素使用 (页 771)（确保程序编辑不引起重复赋值）

强制值 (页 779)

在 STOP 模式下强制输出 (页 780)

12.7 硬件故障排除指南

表格 12-1 S7-200 SMART G2 硬件的故障排除指南

问题	可能的原因	可能的解决方案
输出停止工作	受控设备产生的浪涌损坏了输出	连接到感性负载（例如电机或继电器）时，应使用相应的抑制电路。请参见接线指南 (页 102)。
	接线松动或不正确	检查接线并更正
	负载过大	检查负载是否超出触点额定值
	输出点受到强制	检查 CPU 是否有强制 I/O
CPU 上的 ERROR 灯亮起（红色）	电噪声	请参见第 3 章中的接线指南。控制面板必须与良好的接地点相连并且高压接线不能与低压接线并行走线。 将 24 V DC 传感器电源的 M 端子接地
	组件损坏	递送硬件以进行维修或更换

问题	可能的原因	可能的解决方案
CPU LED 全部不亮	保险丝熔断	使用线路分析器并监视输入电源，以检查过压尖峰的幅值和持续时间。根据此信息向电源接线添加类型正确的避雷器设备。
	24 V 电源线接反	有关安装现场接线的信息，请参见接线指南 (页 102)。
	电压不正确	
与高能量设备相关的间歇操作	接地不正确	请参见接线指南 (页 102)。
	在控制柜内布线	控制面板必须与良好的接地点相连并且高压接线不能与低压接线并行走线。 将 24 V DC 传感器电源的 M 端子接地。
	输入滤波器的延时太短	增加系统数据块中输入滤波器的延时。
连接到外部设备时，串行通信（RS-485 或 RS-232）会造成损坏。 外部设备上的端口或 CPU 上的端口会造成损坏。	如果所有非隔离设备（如PLC、计算机或其它设备）的电路公共参考电位不相同，通信电缆会提供意外电流通路。意外电流可导致通信错误或对电路造成损坏。	• 请参见接线指南 (页 102)和第 8 章中的网络指南。 • 连接没有公共电位参考点的设备时，请购买网络隔离器或隔离型 RS485 到 RS485 中继器。 请参见附录查看 S7-200 SMART 设备的产品编号。
其它通信问题 (STEP 7-Micro/WIN SMART V3)		有关网络通信的信息，请参见编程手册中的附录“错误代码 (页 1257)”。

PID 回路和整定 13

13.1 PID 自整定功能介绍

通过 STEP 7-Micro/WIN SMART 的 PID 整定控制面板，您可便捷地对 S7-200 SMART PLC 的 PID 参数进行自整定。

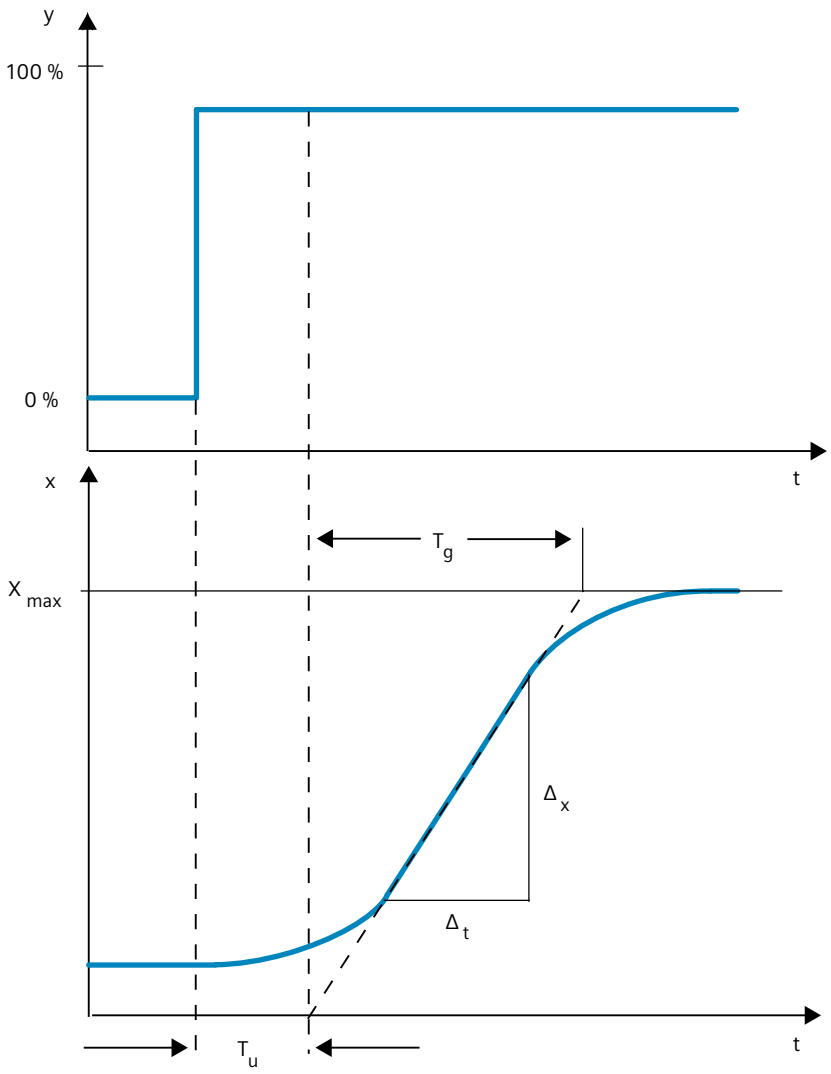
在 CPU 处于在线 RUN 状态时，您可通过控制面板在程序中触发 PID 自整定操作。该功能将自动计算并推荐合适的增益、积分时间及微分时间参数。针对不同的被控对象和预期控制效果，您可以选择合适的回路响应类型，包括快速响应、中速响应、慢速响应或极慢速响应等模式。

通过 PID 整定控制面板，您可以启动自整定过程、中止自整定过程以及在图表中监视结果。控制面板会显示所有可能出现的错误情况或警告。您可以应用通过自整定计算出的增益、复位和速率值。

本文以正向加热/负向制冷的双向温度 PID 调节自整定过程为示例，其他物理量的 PID 自整定方法亦可参照本文档进行操作。

13.2 预调节

预调节可确定对输出值跳变的过程响应，根据受控系统的最大斜率和死区时间计算 PID 参数。



时间响应可由使用变量延迟时间 Tu、恢复时间 Tg 和最大值 Xmax 来大致确定。这些变量可通过最大值的切点和阶跃响应的转折点来确定。在很多情况下，无法记录达到最大值的响应特性，因为过程值不能超过特定值。在这种情况下，使用上升速率 vmax 来识别受控系统 (vmax = Δx/Δt)。

过程值越稳定，PID 参数的计算就越容易，结果也就越精确。因此，建议在“未激活”模式下启用预调节。检查参数设置中的“不激活 PID”。只要过程值的上升速率明显高于噪声，则过程值的噪声可容忍。

STEP 7-Micro/WIN SMART 具有不同的预调节类型，具体取决于组态：

注：建议在 PIDx_CTRL 处于“未激活”模式下时进行预调节。

- 预调节正向/加热
- 预调节负向/制冷
- 预调节正向/加热和负向/制冷

如果要调节正向/加热和负向/制冷的 PID 参数，则可通过“预调节正向/加热”然后“预调节反向/制冷”（而非“预调节正向/加热和负向/制冷”）获得更好的控制响应。但是，分两步进行预调节需要更多的时间。

单向预调节的要求

正向/加热

- | 设定值 - 过程值 | > 0.3 × |过程值上限 - 过程值下限 |
- | 设定值 - 过程值 | > 0.5 × 设定值
- 设定值 > 过程值

负向/制冷

- | 设定值 - 过程值 | > 0.3 × |过程值上限 - 过程值下限 |
- | 设定值 - 过程值 | > 0.5 × 设定值
- 设定值 < 过程值

双向预调节的要求

正向/加热

- | 设定值 - 过程值 | > 0.3 × |过程值上限 - 过程值下限 |
- | 设定值 - 过程值 | > 0.5 × 设定值
- 设定值 > 过程值

负向/制冷

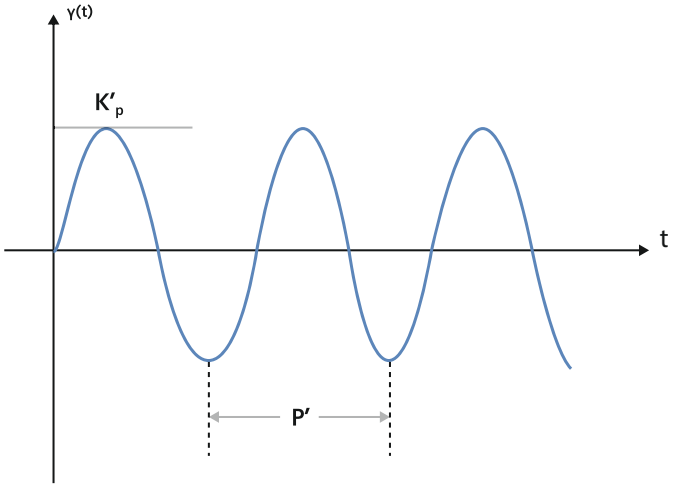
- 成功执行正向/加热预调节
- | 设定值 - 过程值 | < 0.05 × |过程值上限 - 过程值下限 |

正向/加热和负向/制冷

- | 设定值 - 过程值 | > 0.3 × |过程值上限 - 过程值下限 |
- | 设定值 - 过程值 | > 0.5 × 设定值
- 设定值 > 过程值

13.3 精确调节

精确调节会生成恒定的、有限过程值振荡。PID 参数根据该振荡的振幅和频率针对工作点进行调节。根据结果重新计算所有 PID 参数。基于精确调节的 PID 参数通常比基于预调节的 PID 参数具有更好的主控和扰动特性。



K'_P 振幅

P' 振荡周期

在 PID 控制面板中，精确调节类型因不同的组态而异：

- 如果未在 PID 回路向导中启用双向输出，则只有一种精确调节类型可用：
  - 精确调节
- 如果启用双向输出并在 PID 回路向导中选择“正常”(Normal) 作为“控制器类型”(Controller Type)，则以下调节类型可用：
  - 精确调节正向
  - 精确调节负向
- 如果启用双向输出并在 PID 回路向导中选择“温度”(Temperature) 作为“控制器类型”(Controller Type)，则以下调节类型可用：
  - 精确调节加热
  - 精确调节制冷

调节偏移

可为精确调节指定一个临时偏移量，这个临时偏移量用作反方向的输出：

- 对于精确调节类型“精确调节正向/加热”，可设置负向偏移参数。
  启动调节之前，输入一个负向调节偏移，该偏移可使过程值降至小于设定点值的值。负向调节偏移用作输出来生成过程值的反向振荡。
- 对于精确调节类型“精确调节负向/制冷”，可设置正向偏移参数。

说明

- 调节偏移值范围为 0% 至 100%。
- 如果将调节偏移设为 0%，则会禁用调节偏移。

说明

- 如果组态调节偏移，调节偏移将用作输出，来为过程值生成正向或负向振荡。
- 如果未组态调节偏移，整定开始时的 PID 输出值加上“初始输出步”值将用作输出，来为过程值生成正向振荡。整定开始时的 PID 输出值减去“初始输出步”值将用作输出，来为过程值生成负向振荡。
- 如果过程值无法越过设定值，或者使过程值越过设定值需要很长时间，则调整调节偏移或初始输出步。

两步整定法

如果过程值接近设定值，则可以直接执行精确调节。

如果过程值与设定点值相距较远，并且需要进行精确调节，PLC 将自动使用两步整定法，使过程值接近设定点值。

13.4 PID 回路定义表

通过在 PID 指令框中针对表 (TBL) 输入的起始地址，为回路表分配存储空间。S7-200 SMART CPU 的 PID 指令引用包含回路参数的此回路表。

- 单向：120 个字节
- 双向：240 个字节

如果使用 PID 整定控制面板，则可通过控制面板处理所有与 PID 回路表的交互。如果需要通过操作员面板提供自整定功能，您的程序必须提供操作员和 PID 回路表之间的交互，这样才能启动和监视自整定过程，然后应用推荐的整定值。

表格 13-1 单向 PID 的回路表

偏移	字段	格式	类型	说明
0	过程变量 (PV_n)	REAL	输入	包含过程变量，其值必须标定在 0.0 到 1.0 之间。
4	设定值 (SP_n)	REAL	输入	包含设定值，其值必须标定在 0.0 到 1.0 之间。
8	输出 (M_n)	REAL	输入/输出	包含计算出的输出，其值必须标定在 0.0 到 1.0 之间。
12	增益 (K_C)	REAL	输入	包含增益，为比例常数。可以是正数或负数。
16	采样时间 (T_S)	REAL	输入	包含采样时间，单位为秒。必须是正数。
20	积分时间或复位 (T_I)	REAL	输入	包含积分时间或复位，单位为分。
24	微分时间或速率 (T_D)	REAL	输入	包含微分时间或速率，单位为分。
28	偏置 (MX)	REAL	输入/输出	包含偏置或积分和值，介于 0.0 到 1.0 之间。
32	前一过程变量 (PV_{n-1})	REAL	输入/输出	包含上次执行 PID 指令时存储的过程变量值。
36	PID 扩展表 ID	ASCII	常数	“PIDA”（单向 PID 的 PID 扩展表，版本 A）：ASCII 常数 “PIDB”（双向 PID 的 PID 扩展表，版本 B）：ASCII 常数
40	AT 控制 (ACNTL)	BYTE	输入	请参见表“控制和状态字段的具体描述”。
41	AT 状态 (ASTAT)	BYTE	输出	请参见表“控制和状态字段的具体描述”。
42	AT 结果 (ARES)	BYTE	输入/输出	请参见表“控制和状态字段的具体描述”。
43	AT 组态 (ACNFG)	BYTE	输入	请参见表“控制和状态字段的具体描述”。
44	预留	REAL	输入	预留
48	预留	REAL	输入	预留
52	初始输出步 (STEP)	REAL	输入	输出值中步变化的标准化大小，用于使 PV 产生振荡（建议的范围：0.1 到 0.4）。

偏移	字段	格式	类型	说明
56	看门狗时间 (WDOG)	REAL	输入	两次过零之间允许的最大秒数值（范围：60 到 7200）。
60	建议增益 (AT_K_C)	REAL	输出	自整定过程确定的建议回路增益。
64	建议积分时间 (AT_T_I)	REAL	输出	自整定过程确定的建议积分时间。
68	建议微分时间 (AT_T_D)	REAL	输出	自整定过程确定的建议微分时间。
72	实际步大小 (ASTEP)	REAL	输出	自整定过程确定的标准化输出步大小值。
76	实际滞后 (AHYS)	REAL	输出	自整定过程确定的标准化 PV 滞后值。
80	预留	WORD	输入	预留
82	预留	BYTE	输入	预留
83	预留	DWORD	输入	预留
87	预留	DWORD	输入	预留
91	回路设定值范围	REAL	输入	默认值为 100.0。
95	微调偏移量	REAL	输入	微调偏移量
99	控制规范	BYTE	输入	控制规范： 位 7 / 位 6 / 位 5 / 位 4 / 位 3 / 位 2 / 位 1 / 位 0 快速响应系统 / 预留 / CNTL_TYPE / 预留 / INTEGRAL_LIM_EN / DZ_EN / RG_EN 快速响应系统：0 - 非快速响应系统；1 - 快速响应系统 DZ_EN：0 - 死区未使能，1 - 使能 RG_EN：0 - 控制区未使能，1 - 使能 CNTL_TYPE：0 - 正常，1 - 温度 INTEGRAL_LIM_EN：0 - 积分限幅未使能，1 - 使能
100	死区	REAL	输入	死区：范围 0.0-1.0
104	控制区	REAL	输入	控制区：范围 0.0-1.0
108	预留	BYTE	输入	预留

偏移	字段	格式	类型	说明
109	积分上限	BYTE	输入	积分上限：范围 0-100 • 0：积分已禁用。 • 100：积分限幅已禁用。
110	预留	WORD	输入	预留
112	预留	REAL	输入	预留
116	预留	REAL	输入	预留

说明

上表描述了单向 PID 回路的定义。

如果组态双向 PID，上表描述了正向/加热侧的 PID 回路定义。

表格 13-2　回路表（偏移 120-240 用于负向/制冷侧）

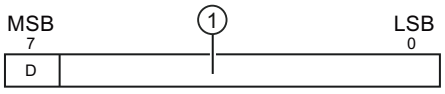
偏移	字段	格式	类型	说明
120	预留	REAL	输入	预留
124	预留	REAL	输入	预留
128	输出 (M_n)	REAL	输入/输出	包含计算出的输出，其值必须标定在 0.0 到 1.0 之间。
132	增益 (K_C)	REAL	输入	包含增益，为比例常数。可以是正数或负数。
136	预留	REAL	输入	预留
140	积分时间或复位 (T_I)	REAL	输入	包含积分时间或复位，单位为分。
144	微分时间或速率 (T_D)	REAL	输入	包含微分时间或速率，单位为分。
148	偏置 (MX)	REAL	输入/输出	包含偏置或积分和值，介于 0.0 到 1.0 之间。
152	预留	REAL	输入	预留
156	PID 扩展表 ID	ASCII	常数	“PIDB”（双向 PID 的 PID 扩展表，版本 B）：ASCII 常数
160	AT 控制 (ACNTL)	BYTE	输入	请参见表“控制和状态字段的具体描述”。
161	AT 状态 (ASTAT)	BYTE	输出	请参见表“控制和状态字段的具体描述”。
162	AT 结果 (ARES)	BYTE	输入/输出	请参见表“控制和状态字段的具体描述”。
163	AT 组态 (ACNFG)	BYTE	输入	请参见表“控制和状态字段的具体描述”。
164	预留	REAL	输入	预留
168	预留	REAL	输入	预留

偏移	字段	格式	类型	说明
172	初始输出步 (STEP)	REAL	输入	输出值中步变化的标准化大小，用于使 PV 产生振荡（建议的范围：0.0 到 0.4）。
176	看门狗时间 (WDOG)	REAL	输入	两次过零之间允许的最大秒数值（范围：60 到 7200）。
180	建议增益 (AT_K_C)	REAL	输出	自整定过程确定的建议回路增益。
184	建议积分时间 (AT_T_I)	REAL	输出	自整定过程确定的建议积分时间。
188	建议微分时间 (AT_T_D)	REAL	输出	自整定过程确定的建议微分时间。
192	实际步大小 (ASTEP)	REAL	输出	自整定过程确定的标准化输出步大小值。
196	实际滞后 (AHYS)	REAL	输出	自整定过程确定的标准化 PV 滞后值。
200	预留	WORD	输入	预留
202	预留	BYTE	输入	预留
203	预留	DWORD	输入	预留
207	预留	DWORD	输入	预留
211	回路设定值范围	REAL	输入	默认值为 100.0。
215	微调偏移量	REAL	输入	微调偏移量
219	控制规范	BYTE	输入	控制规范： 位 7：快速响应系统 位 6：预留 位 5：CNTL_TYPE 位 4：预留 位 3：预留 位 2：INTEGRAL_LIM_EN 位 1：DZ_EN 位 0：RG_EN 快速响应系统：0 - 非快速响应系统；1 - 快速响应系统 DZ_EN：0 - 死区未使能，1 - 使能 RG_EN：0 - 控制区未使能，1 - 使能 CNTL_TYPE：0 - 正常，1 - 温度 INTEGRAL_LIM_EN：0 - 积分限幅未使能，1 - 使能
220	死区	REAL	输入	死区：范围 0.0-1.0
224	控制区	REAL	输入	控制区：范围 0.0-1.0
228	预留	BYTE	输入	预留

偏移	字段	格式	类型	说明
229	积分上限	BYTE	输入	积分上限：范围 0-100 • 0：积分已禁用。 • 100：积分限幅已禁用。
230	预留	BYTE	输入	预留
231	预留	BYTE	输入	预留
232	预留	REAL	输入	预留
236	预留	REAL	输入	预留

表格 13-3　控制和状态字段的具体描述

字段	说明	
AT 控制 (ACNTL) 输入 - 字节	MSB 7 … LSB 0 0 0 0 0 0 0 1 ATEN	
	ATEN	设为 1 可启动精确调节；设为 2 可启动预调节；设为 0 可中止自调节。
AT 状态 (ASTAT) 输出 - 字节	MSB 7 … LSB 0 W0 W1 W2 W3 AH WRSP RUNIN IP	
	W0	警告：此偏差设置比滞后设置大了不到 4 倍。
	W1	警告：不一致的过程偏差可能导致输出步值的调整不正确。
	W2	警告：实际平均偏差比滞后设置大了不到 4 倍。
	W3	警告：未检测到拐点或采样数据拟合不良。预调节参数可能不准确。
	AH	正在进行自动滞后计算： 0 - 没有进行 1 - 正在进行
	WRSP	正向/加热侧预调节成功。等待过程值达到设定值，以进行负向/制冷侧预调节。
	RUNIN	两位控制调节正在进行中。
	IP	预调节或精确调节正在进行中： 0 - 没有进行 1 - 正在进行
	每次预调节或精确调节序列启动时，CPU 都会清除警告位并置位进行位。自整定完成后，CPU 会清除进行位。	

字段	说明	
AT 结果 (ARES) 输入/输出 - 字节	 ① 结果代码	
	D	Done 位： 0 – 自整定未完成 1 – 自整定完成 必须设置为 0，自整定才能启动
	结果代码	00 - 正常完成（推荐的整定值可用） 01 - 用户中止 02 - 已中止，过零时看门狗超时 03 - 已中止，过程 (PV) 超出范围 04 - 已中止，超出最大滞后值 05 - 已中止，检测到非法组态值 06 - 已中止，检测到数字错误 07 - 已中止，在没有能流时执行 PID 指令（回路处于手动模式） 08 - 已中止，自整定只适用于 P、PI、PD 或 PID 回路 09 - 正向/加热精确调节中止，过程值 (PV) 超出范围 10 - 负向/制冷精确调节中止，过程值 (PV) 超出范围 11 - 精确调节中止，输出方向不受支持 12 到 15 - 预留 16 - 负向/制冷预调节中止，因为正向/加热预调节未完成或未成功 17 - 预调节失败。由于缺乏数据，曲线不拟合。 18 - 预调节失败。由于计算的参数无效而中止。 19 - 无法启动预调节。SP 与 PV 的差值必须超过过程 (PV) 上限和下限差值的 30%。 20 - 无法启动预调节。SP 和 PV 的差值必须超过 SP 的 50%。 21 - 无法启动预调节。SP 与 PV 的差值必须小于过程 (PV) 上限和下限差值的 5%。 22 - 仅支持 ZN 和 CHR 方法进行预调节。 23 - CHR 方法不支持 PD 控制类型。

字段	说明	
AT 组态 (ACNFG) 输入 - 字节	MSB 7 / LSB 0 0 0 0 PTM R1 R0 DS HS	
	PTM	预调节方法： • 0 - ZN 方法； **注：**需要结合 R0 和 R1 的位值来确定使用哪种 ZN 方法。 • 1 - CHR 方法；
	R1 R0	动态响应
	0 0	快速响应
	0 1	中速响应
	1 0	慢速响应
	1 1	极慢速响应
	DS	预留
	HS	预留

说明

具有 PID 向导组态的项目不直接使用标准 PID 指令。如果您使用 PID 向导组态，则您的程序必须使用“PIDx_CTRL”来激活 PID 向导子例程。

为在应用程序中简化 PID 回路控制的使用，STEP 7-Micro/WIN SMART 提供一个用于组态 PID 回路的 PID 向导。

13.5 先决条件

要进行自整定的回路必须处于自动模式。回路输出必须由 PID 指令的执行来控制。如果回路处于手动模式，则自整定失败。

理想状态下，自整定启动时，回路输出值应该在控制范围中心附近。自整定过程在回路的输出中加入一些小的阶跃变化，使得控制过程产生振荡。如果回路输出接近其控制范围的任一限值，自整定过程引入的阶跃变化可能导致输出值超出最小或最大范围限值。

如果发生这种情况，可能会生成自整定错误条件，当然也会使推荐值并非最优化。

13.6 自整定序列

确定了滞后值和偏差值之后，就会开始执行自整定序列。初始输出阶跃应用到回路输出后，就会开始整定过程。

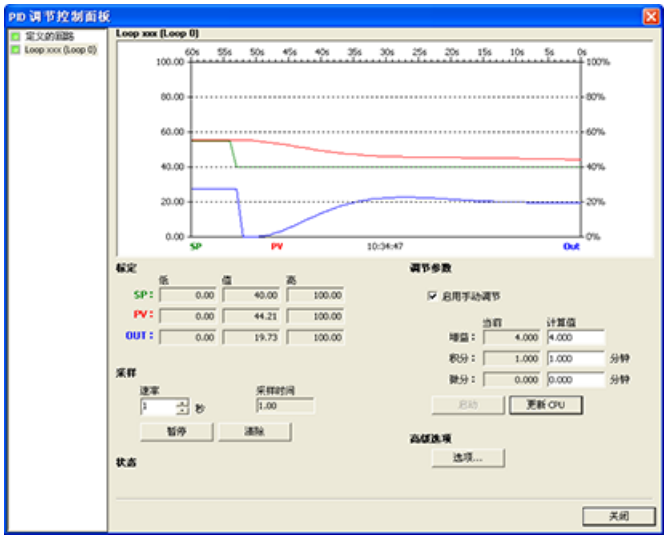
输出值的这一变化会导致过程变量值产生相应的变化。当输出的变化使 PV 远离设定值以至于超出滞后区范围时，自整定器就会检测到过零事件。每次发生过零事件时，自整定器将反方向改变输出。

整定器会继续对 PV 进行采样，并等待下一个过零事件。要完成序列，整定器总共需要 12 次或更多次过零事件。所观察到的峰峰 PV 值幅值（峰值误差）和过零发生时的速率与控制过程的动态特性直接相关。

在自整定过程一开始，会适当地调节输出阶跃值，促使 PV 的峰峰值更接近想要得到的偏差值。一旦做出调整后，新的输出阶跃值将写入回路表的“实际阶跃大小”字段 (ASTEP)。

如果两次过零事件之间的时间超出过零看门狗间隔时间，则自整定序列将以出错终止。过零看门狗间隔时间的默认值为两小时。

下图显示了一个直接作用回路的自整定序列过程中输出和过程变量的变化情况。使用 PID 整定控制面板启动和监视整定序列。



请注意自整定器是如何改变输出，以使过程（用 PV 值表示）经受小幅振荡的。PV 振荡的频率和振幅反映过程增益和固有频率。

根据自整定过程期间采集到的过程的频率和增益的相关信息，能够计算出最终增益和频率值。通过这些值可以计算出增益（回路增益）、复位（积分时间）和速率（微分时间）的建议值。

说明

回路类型决定了自整定器计算的整定值。例如，对于 PI 回路，自整定器将计算增益和积分时间值，但建议的微分时间将是 0.0（无微分作用）。

自整定序列完成后，回路输出会恢复到初始值。下一次执行回路时，将执行正常的 PID 计算。

13.7 例外情况

警告情况

整定执行过程中会产生三种警告情况。整定执行过程会在回路表 ASTAT 字段的三个位中报告以下三种警告，并且这三位一旦被置位，将会一直保持到下一次自整定序列启动：

- 警告 0：如果偏差值没有超过滞后值的 4 倍，就会产生该警告。该检测在已经计算出滞后值之后执行，而滞后值取决于自滞后设置。
- 警告 1：在自整定过程最开始的 2.5 个循环周期内，如果两次峰值误差超出 8 倍，将产生此警告。
- 警告 2：如果测得的平均峰值误差没有超过滞后值的 4 倍，就会产生此警告。
- 警告 3：未检测到拐点或采样数据拟合不良。预调节参数可能不准确。

错误条件

除了上述警告情况之外，还有几种错误情况。下表列出了可能导致每种错误的情况和描述。

表格 13-4 整定执行期间的错误情况

结果代码（在 ARES 中）		条件
01	用户中止	整定进行时，EN 位被清除
02	因过零看门狗超时而中止	超出过零看门狗时间间隔半个周期
03	因过程超出范围而中止	PV 超出范围： • 自滞后序列期间，或 • 第四次过零前两次超出范围，或 • 第四次过零之后
04	因滞后值超出最大值而中止	用户指定的滞后值，或 自动确定的滞后值大于最大值
05	因非法组态值而中止	在下列范围内检测错误： • 初始回路输出值小于 0.0 或大于 1.0 • 用户指定的偏差值小于等于滞后值或大于最大值 • 初始输出阶跃小于等于 0.0 或大于最大值 • 过零看门狗间隔时间小于最小值 • 回路表中的采样时间值为负数
06	因数字错误而中止	遇到非法浮点数或除数为零

结果代码（在 ARES 中）		条件
07	执行 PID 指令时无能流（手动模式）	正在执行自整定或请求执行自整定时，执行 PID 指令时无能流
08	自整定只适用于 P、PI、PD 或 PID 回路	回路类型不是 P、PI、PD 或 PID
09	正向/加热精确调节已中止	过程值 (PV) 超出范围
10	负向/制冷精确调节已中止	过程值 (PV) 超出范围
11	精确调节已中止	输出方向不受支持
12 到 15	预留	
16	负向/制冷预调节已中止	正向/加热预调节未完成或未成功
17	预调节失败	由于缺乏数据，曲线不拟合
18	预调节失败	计算的参数无效
19	无法启动预调节	SP 与 PV 的差值必须超过过程 (PV) 上限和下限差值的 30%。
20	无法启动预调节	SP 和 PV 的差值必须超过 SP 的 50%。
21	无法启动预调节	SP 与 PV 的差值必须小于过程 (PV) 上限和下限差值的 5%。
22	无效预调节组态	仅支持 ZN 和 CHR 方法进行预调节
23	无效预调节组态	CHR 方法不支持 PD 控制类型

13.8 关于过程变量超限的说明（结果代码 3）

如果过程变量的值大于 0.0 小于 1.0，自整定器就会认为它在范围内。

如果在自滞后序列期间检测到 PV 超限，则会立即中止整定过程，并会生成过程超限错误。

如果在整定序列起始点与第四次过零之间检测到 PV 超限，则输出阶跃值将减半，整定序列将从头开始。 重新启动后，如果在第一次过零之后再次检测到 PV 超限事件，整定会中止，并生成过程超限错误。

如果在第四次过零之后发生任何 PV 超限事件，整定都会立即中止，并会生成过程超限错误。

13.9 PID 整定控制面板指令

STEP 7-Micro/WIN SMART V3 中包含 PID 整定控制面板，允许以图形方式监视 PID 回路行为。此外，控制面板还可用于启动自整定序列、中止序列以及应用建议的整定值或您自己的整定值。

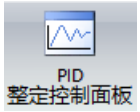
说明

如果 SP 在自动调节完成后的控制过程中被修改，西门子建议再次下载该项目。

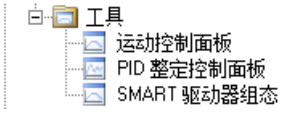
要使用控制面板，必须与 CPU 通信，并且该 CPU 中必须存在一个用于 PID 回路的向导生成的组态。要使控制面板显示对 PID 回路的操作，CPU 必须处于 RUN 模式。

采用以下任意一种方式打开 PID 控制面板：

- 单击“工具”(Tools) 菜单功能区的“工具”(Tools) 区域中的“PID 控制面板”(PID Control Panel) 按钮。



- 在项目导航 (页 157)中打开“工具”(Tools) 文件夹，选择“PID 整定控制面板”(PID Tune Control Panel) 节点，然后按 Enter 键；或双击“PID 整定控制面板”(PID Tune Control Panel) 节点。



如果所连接的 CPU 处于 RUN 模式，则 STEP 7-Micro/WIN SMART V3 将打开 PID 控制面板：

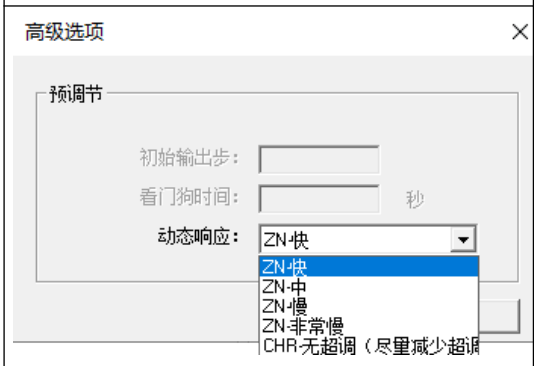
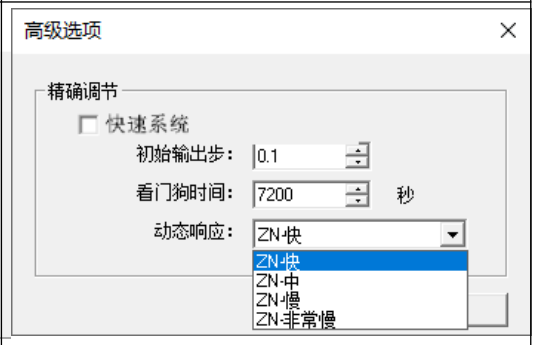
PID 控制面板包含以下字段：

- 当前值：显示 SP（设定值）、PV（过程变量）、OUT（输出）、“采样时间”、“增益”、积分时间和微分时间的值。SP、PV 和输出正向/加热和输出负向/制冷分别以绿色、红色、蓝色和橙色显示；使用相同颜色的图例来标明 PV、SP 和 OUT 的值。
- 图形显示区：图形显示区中用不同的颜色显示了 PV、SP 和输出值相对于时间的函数。PV 和 SP 共用图形左侧纵轴，输出使用图形右侧的纵轴。
- 整定参数：画面右下角是“调节参数（分钟）”(Tuning Parameters (Minutes)) 区域。在此处显示“调节类型”(Tuning type)、“增益”(Gain)、“积分时间”(Integral Time) 和“微分时间”(Derivative Time) 的值。在“计算值”(Calculated) 列中单击，可对这些值的三个源中任意一个源进行修改。

  调节类型

  – 预调节：预调节正向/加热，预调节负向/制冷，预调节正向/加热与负向/制冷
  – 精确调节：精确调节正向/加热、精确调节负向/制冷
  – 手动调节：手动调节正向/加热、手动调节负向/制冷
- “更新 CPU”(Update CPU) 按钮：可以使用“更新 CPU”(Update CPU) 按钮将所显示的“增益”、“积分时间”和“微分时间”值传送到被监视的 PID 回路的 CPU。可以使用“启动调节”(Start tuning) 按钮启动自调节序列。自调节序列启动后，“启动调节”(Start tuning) 按钮将变为“停止调节”(Stop tuning) 按钮。

- 采样：在“采样”(Sampling) 区域，您可以选择图形显示区的采样速率，范围为每 1 到 240 秒进行一次采样。
  可单击“暂停”(Pause) 按钮冻结图形。可单击“继续”(Resume) 按钮以选定的速率重新启动数据采样。在图形区域内单击鼠标右键并选择“清除”(Clear) 可清除图形。
- “高级选项”(Advanced Options)：可以使用“选项”(Options) 按钮对调节过程的参数进行进一步组态。（请参见下图。）

预调节	精确调节
 无需为预调节设置“初始步”(Initial step) 和“看门狗时间”(Watchdog time)。	

快速系统

对于某些变化很快的系统，建议选中该选项。

初始输出步

初始输出步表示 PID 整定的初始输出值。初始步值用实际输出的最大功率的百分比表示。PID 精确调节启动后，控制器会主动更改 PID 输出值，观察整个系统的响应。

如果未组态调节偏移，整定开始时的 PID 输出值加上“初始输出步”值将用作输出，来为过程值生成正向振荡。整定开始时的 PID 输出值减去“初始输出步”值将用作输出，来为过程值生成负向振荡。

说明

- 为生成一个恒定的、有限的过程值振荡，建议设置“初始输出步”(initial output step) 参数。
- 初始输出步值在 0 到 0.4 之间（不包括 0）。

看门狗时间

过程值必须在看门狗时间内达到或越过设定值一次。否则，将出现看门狗超时错误。

如果提前清楚系统响应缓慢，则可增加看门狗时间。

动态响应

– ZN-快：快速响应可能产生超调，并符合欠阻尼调节条件。
– ZN-中：中速响应可能频临超调，并符合临界阻尼调节条件。
– ZN-慢：慢速响应不会导致超调，符合强衰减调节条件。

– ZN-非常慢：极慢速响应不会导致超调，符合强过阻尼调节条件。
– CHR-无超调（尽量减少超调）：此响应可能具有最小的超调或初始输出步。

在“动态响应”(Dynamic Response) 字段中，可使用下拉按钮选择希望在控制过程中使用的回路响应类型。

说明

如果通过预调节得到的 PID 参数不令人满意，建议更改动态响应方法以获得更好的 PID 参数。

回路监视

完成预调节或精确调节序列并将建议的调节参数传送到 CPU 之后，可以使用控制面板监视回路对设定值步变化的响应。

如果您希望牺牲一部分响应速度来消除这些振荡，您可以选择中速响应或者慢速响应类型，然后重新运行自整定过程。

一旦您有了一个好的起点，您就可以使用控制面板来进一步优化参数。之后就可以监视回路对设定值变化的响应。通过这种方式，您可以微调您的控制过程，使您的应用达到最佳效果。

为简化应用程序中 PID 回路控制的使用，STEP 7-Micro/WIN SMART V3 提供一个用于组态 PID 回路的 PID 向导。

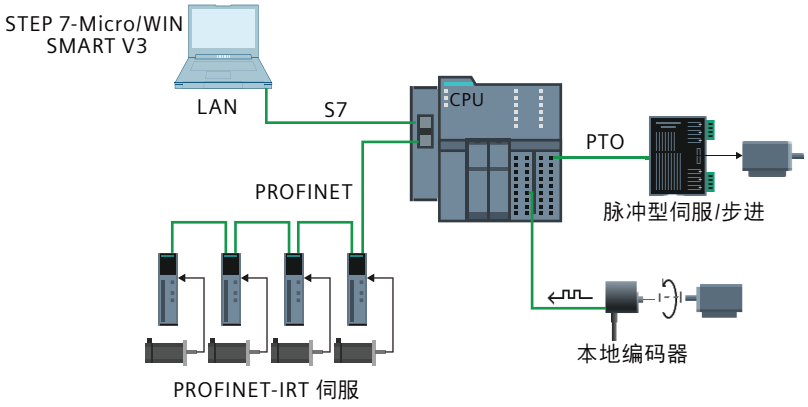
运动控制 (PLCopen)

14

S7-200 SMART 运动控制为运动轴的开环和闭环控制提供了一种易于实施的解决方案。

根据 PLCopen，具有 PROFIdrive 功能的驱动器和脉冲型驱动器将通过标准运动控制指令控制。轴控制面板以及诊断功能有助于轻松完成驱动器的调试和优化工作。

S7-200 SMART 运动控制概览



运动任务与功能

S7-200 SMART 运动控制支持以下功能：

运动任务/功能
连续运动 可在滚筒、皮带、链和其它传送带中实现连续运动。
速度控制的连续运动
扭矩限值
定位 可为需要动态响应和精度的运动任务实现精确的循环定位。
精确的绝对和相对定位（闭环位置控制）
高动态响应
通过硬件和软件限位开关进行行程范围限制
线性和旋转轴

运动任务/功能	
凸轮/齿轮传动 对于对精确速度、扭矩和位置的连续加工步骤，可实现生产机器的协调轴运动操作	
	在不指定同步位置的条件下进行齿轮传动
	凸轮传动
轴组 可通过多达三个机械耦合轴实现\搬运控制	
	坐标系与区域的校准
	采用线性运动的方式移动运动系统

14.1 基本概念

PROFINET IO 中的 RT 等级

PROFINET IO 是一个基于以太网技术的可扩展的实时通信系统。 扩展的方法可以通过不同的实时等级实现：

- **RT：** 以区分优先顺序的非同步以太网帧传送数据。 所需带宽在 TCP/IP 通信的空闲带宽区域以内。
- **IRT：** 同步传输数据，具有高稳定性，用于对时间要求高的应用（例如，运动控制）。 所需带宽来自于为周期数据保留的带宽区域。

根据设备的情况，并非支持所有的实时等级。

IPO (Interpolate)

在存在 IRT 轴的系统中，IPO 的核心作用是在每一个固定的 IRT 周期内，对预先规划的宏观运动曲线（如直线、圆弧或复杂轨迹）执行插补运算，从而精确计算出轴在当前周期内的移动距离和目标速度。

PROFIdrive

PROFIdrive 即为标准化驱动技术配置文件，可通过 PROFINET IO 连接驱动装置和编码器。支持 PROFIdrive 配置文件的驱动装置都可根据 PROFIdrive 标准进行连接。

PROFIdrive 报文

PROFIdrive 报文用于传送设定值和实际值、控制字和状态字以及控制器与驱动器/编码器之间的其它参数。

如果使用 PROFIdrive 报文进行连接，则会根据 PROFIdrive 配置文件处理和开启驱动器及编码器。

IRT 轴支持的报文类型 (页 844):

- 主报文：支持 5 号报文和 105 号报文
- 附加报文：支持 750 号报文

轴

在运动控制系统中，将运动控制的对象称为“轴”。

轴种类	说明
虚拟轴	通过 STEP 7-Micro/WIN SMART V3 运动控制功能可将轴组态为虚拟轴。虚拟轴的行为类似于实际轴，但没有驱动装置或编码器连接。设定值仅在控制器内进行处理。
PTO 轴	通过脉冲串输出控制电机运动的轴类型: • 开环控制 • 脉冲频率决定转速，脉冲数量决定位移
编码器轴	目前仅支持本地编码器轴。编码器轴可用于监控轴的运动状态，并可作为凸轮（Cam）和齿轮（Gear）等应用的主轴。 • 单轴指令：只支持 MC_Home（模式 0 和 1）、MC_TouchProbe、MC_AbortTrigger。 • 同步运动：仅能作为 MC_CamIn 和 MC_GearIn 指令的主轴使用。 • 轴组：不支持
IRT 轴	支持 IRT（等时实时）通信协议的同步运动轴: 每个 IRT 轴都分配有一个驱动器（通过 PROFIdrive 报文分配）和编码器（通过 PROFIdrive 报文分配）。 通过对硬件接口、机械限位和回零操作进行参数配置，可以建立编码器值与实际位置之间的映射关系。在尚未建立位置参考（如未回零）的情况下，轴仍可执行运动指令。即使在未回零状态下，轴也能执行相对位移指令。

轴组

两个及以上运动轴的协同控制单元。

凸轮传动

在凸轮传动期间，主轴和从轴将通过同步操作功能进行耦合（通过凸轮进行指定）。凸轮传动过程中的传输行为通过凸轮曲线表示。

齿轮传动

齿轮传动过程中，从轴的位置等于主轴位置乘以传动比。将传动比指定为两个整数之间的比例。其结果是一个线性传动函数。

运动控制指令

使用运动控制指令在用户程序的轴中启动运动控制工作，并由此在轴中执行所需功能。可以使用运动控制指令的输出参数跟踪运行中作业的状态。

作业（Job）

在接收到运动控制指令后被触发，校验运动控制指令的输入。

14.2 运动控制资源

对象		说明	
CPU 版本		V3.0/V3.1 CPU	V3.2 CPU
控制轴最大数量		总共最多支持 16 个轴。	• 总共最多支持 16 个轴。 • 总共最多支持 8 个物理可动轴 （ PTO 及 PN IRT 轴）。
	PTO 轴	最多 5 个	最多 5 个
	虚拟轴	最多 16 个	最多 16 个
	本地编码器轴	最多 2 个	最多 2 个
	PN IRT 轴	0	最多 8 个
	直线插补控制最大轴数	单个轴组最多控制 3 个轴	单个轴组最多控制 3 个轴
最大轴组数	最多 2 个		

对象			说明
凸轮	每个 PLC 允许添加最多 8 个凸轮。		
	凸轮数据点数	每个凸轮表支持的最大关键点数	361 个
		每个凸轮表支持的最大点数	65535 个
		所有凸轮表可添加的最大关键点数	1444 个

说明

HSC6 和 HSC7 的使用与编码器轴的使用不兼容。因此，如果启用了 HSC6 或 HSC7，只能使用一个编码器轴。如果同时启用了 HSC6 和 HSC7，则无法使用编码器轴，反之亦然。

说明

PN IRT 轴暂不支持配置为轴组。

PROFINET IRT 轴配置建议

在规划和配置基于 PROFINET IRT 的运动控制系统时，为了确保系统性能和稳定性，请参考以下关于 IRT 周期、运动控制资源占用率与轴数量的建议：

IRT 周期	运动控制资源占用率	建议 IRT 轴数量
1 ms	50%	不超过 2 根
2 ms	50%	不超过 4 根
4 ms	50%	不超过 8 根
4 ms	65%	8 根 IRT 轴 + 8 根虚轴

极端配置示例：

对于包含 3 根 IRT 轴、5 根 PTO 轴以及 8 根虚轴的复杂系统，建议将 IRT 周期设定为 4ms，并将运动控制资源占用率调整至 75%，以确保系统稳定运行。

说明

在运动控制应用中，当处理器资源占用率达到 50% 时，这意味着运动控制任务已经消耗了中央处理器（CPU）可用计算能力的一半。在此特定负载水平下，若不考虑用户中断及其他通信开销，系统的扫描周期将比没有运动控制任务运行时最多延长一倍。

PTO 轴资源占用

PTO（脉冲串输出）轴采用 4ms 同步插补时钟，5 根 PTO 轴将占用约 35% 的运动控制资源。

14.3 运动控制概述

14.3.1 控制模式

14.3.1.1 位置控制模式（闭环）

轴连同驱动装置中的控制器一起，构成级联控制系统。最内侧的控制级联是电流控制，下一级联是速度控制。二者均位于驱动装置中。位置控制器是最外侧的级联，在 PLC 的轴对象中。

轴的位置控制器是一个使用或不使用预控制速度的闭环 P 控制器。使用伺服增益系数设置比例作用控制器的增益。为实现位置控制回路的高动态响应，Kv 因子应设置为较大的值。在位置控制器中使用 Kv 因子时，可通过控制偏差形成调节变量的组成部分，从而校正位置偏差和干扰变量。

启用位置控制时，编码器系统、实际值计算、控制器和监视器均处于激活状态。

如果闭环位置控制处于未激活状态，则编码器系统、实际值计算和监视在实际值侧激活。

在跟随模式下，设定值会跟踪实际值。运动作业不执行。实际位置和实际速度将更新。这意味着，当轴因外部影响而移动时，可以进行跟踪。

动态伺服控制 (DSC)

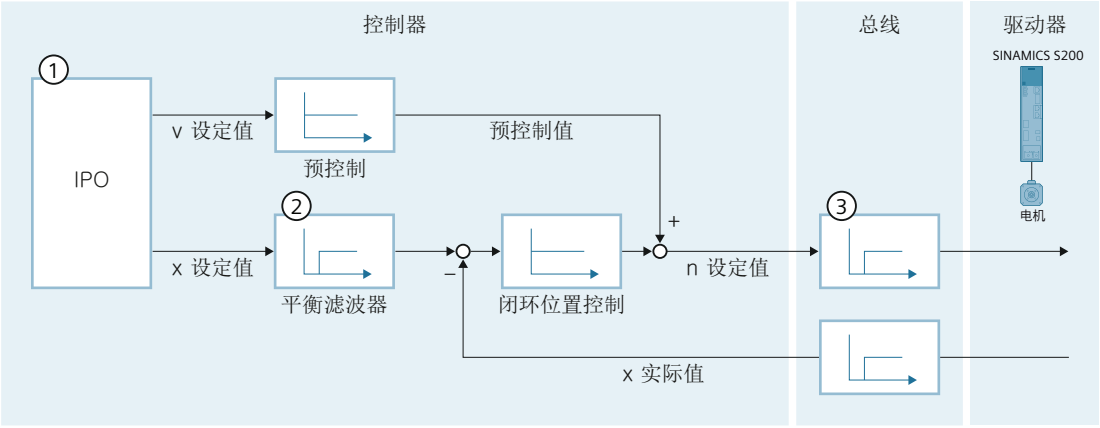
如果驱动器支持动态伺服控制 (DSC)，则可选用驱动器中的闭环位置控制器。如果您使用支持 DSC 的报文，DSC 会自动激活。驱动器中的位置控制器通常与快速速度控制循环一起使用。这样，可以提高数字耦合驱动器的控制性能。

使用 DSC 时必须满足以下要求：

- 将驱动器的电机编码器（报文中的第一个编码器）用作轴的第一个编码器。
- 将以下 PROFIdrive 报文之一用于该驱动器：
  - 标准报文 5
  - SIEMENS 报文 105

控制结构

- 在驱动装置中进行位置控制；为此，驱动装置必须支持动态伺服控制 (DSC)。
  使用 SINAMICS 驱动装置时，这属于一种标准情况，因为驱动装置中更快的控制周期（例如 125 µs）会带来更高的控制质量。
  下图所示为带 DSC 时的有效闭环控制结构：



① 运动控制插补器

② 速度环响应时间的内部考量

③ 控制器与驱动装置之间的通信

速度前馈

在位置控制过程中可使用速度前馈最大程度消除跟随误差。如此可以在需要时实现更快速的定位，因为位置控制器只需要补偿扰动。

使用速度前馈时，速度设定值将加到位置控制器的输出中。您可以设定添加该设定值时的权重。

速度环响应时间

在此字段中，配置速度环响应时间 (T_{vtc})。

速度环响应时间包含在平衡滤波器中。

平衡滤波器是闭环转数控制回路的简化模型。平衡滤波器用于防止位置控制器在加速和减速阶段覆盖转数调节变量。为此，位置控制器的位置设定值延迟了速度环响应时间，该时间与速度预控制有关。

有关速度环响应时间的配置，请注意以下几点：

- 如果不使用任何速度预控制 (0%)，则速度环响应时间的配置不相关。
- 如果使用速度预控制 (>0%) 并将速度环响应时间设置为 0.0 s（默认值），则轴将超调。要找到正确的设置，请优化位置控制器。

位置环增益（Kv 因子）

在该字段中，配置位置控制回路的增益 Kv。

Kv 因子影响以下参数：

- 定位精度和停止控制
- 运动的一致性
- 定位时间

实际轴的机械状态（刚性）越好，可以配置的 Kv 因子越大。这样可以减小跟随误差，实现更快的动态响应。

14.3.1.2 速度控制模式

当运动控制轴采用 Profidrive DSC 控制方式时，PLC 会周期性地向驱动器发送速度给定。

MC_MoveJog 和 MC_MoveVelocity 支持速度控制模式。

速度控制模式下的给定值与参数

位置给定不再生效，只保留速度给定。

- 预控制： 100%。
- Kv： 0，位置控制环关闭。

输入管脚

MC_MoveVelocity 和 MC_MoveJog 指令新增了一个 管脚 SpeedControlled，用于选择是进行位置控制还是速度控制。

说明

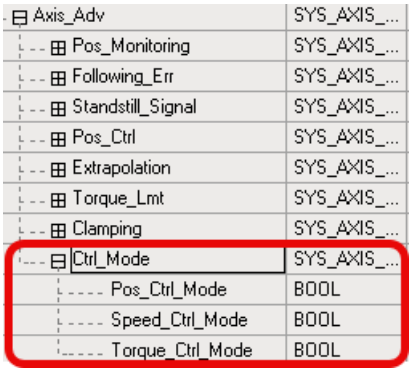
仅 IRT 轴和虚轴支持速度控制模式。PTO 轴和编码器轴暂不支持该模式。对于不支持速度控制模式的轴，将返回错误码 0x1038。

操作指令	修改 SpeedControlled 管脚
MC_MoveJog	支持在 MC_MoveJog 处于 Busy 状态时修改 SpeedControlled 参数。修改后的参数会立即生效。
MC_MoveVelocity	在修改 SpeedControlled 管脚后，需要通过一个上升沿信号才能生效。因此，对于 MC_MoveVelocity 功能块，这种行为也被视为一次指令重启。

轴的当前控制模式

变量表中新增“轴当前控制模式”字段，用于指示当前轴处于位置控制模式、速度控制模式或扭矩控制模式。

对于 PTO 轴和编码器轴，该字段始终为 0。



速度控制模式

跟随误差

跟随误差监控只在位置控制模式下有效，速度控制模式下暂不生效。

软硬限位

速度控制模式下，软硬限位继续生效。

实例

前置条件

- 轴已组态并正确配置跟随误差监控、软硬限位。
- 程序中使用了 MC_MoveJog 或 MC_MoveVelocity 指令。

执行流程与结果

执行条件	结果
设置 SpeedControlled 为 TRUE，触发指令。	跟随误差监测不会生效：设置信息里显示位置给定是 0。轴控制模式显示为速度控制。当轴运动的实际位置到达硬/软限位时，轴会按照配置的急停方式急停并进入 ErrorStop。
设置 SpeedControlled 为 False，触发指令。	跟随误差监测重新生效：设置信息里显示位置在切换后等于实际信息，随后根据运动情况增加或减少。轴控制模式显示为位置控制模式。

Cam 和 Gear

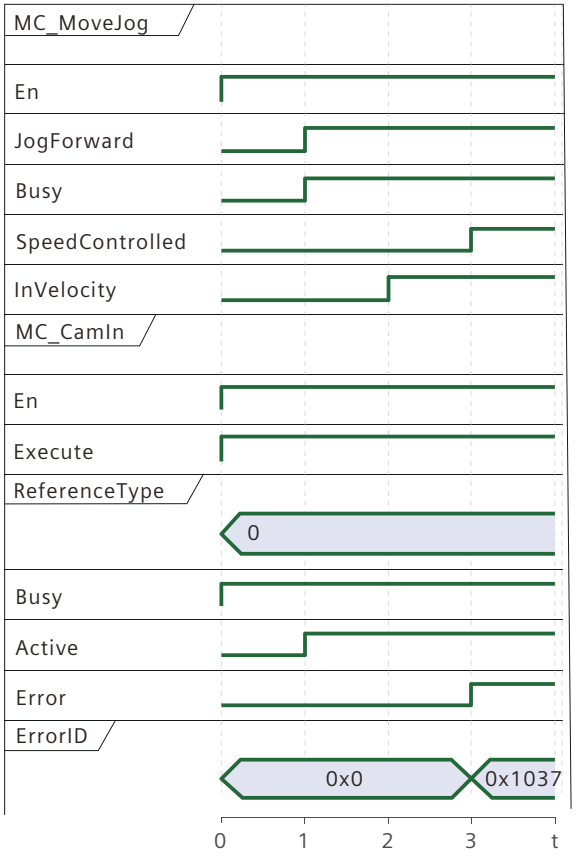
速度控制模式下，仅当从轴的同步参考位置 被配置为主轴的实际位置时，从轴能在速度模式下执行凸轮同步或齿轮同步运动。

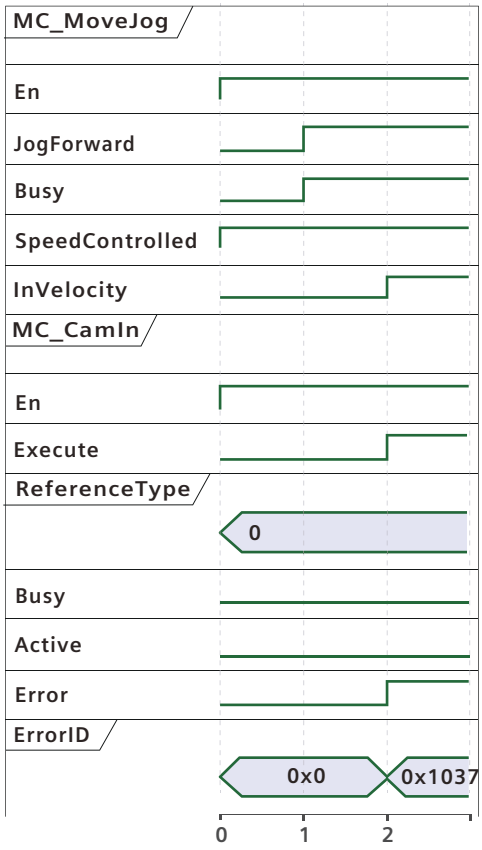
实例

前置条件

- 轴已组态。
- 程序中主轴调用 MC_MoveJog 或 MC_MoveVelocity 指令。从轴调用 MC_CamIn 或 MC_GearIn 指令，从轴能够实现与主轴的同步运动。

时序图


MC_MoveJog
En
JogForward
Busy
SpeedControlled
InVelocity
MC_CamIn
En
Execute
ReferenceType
0
Busy
Active
Error
ErrorID
0x0
0x1037
0
1
2
3
t



切换控制模式

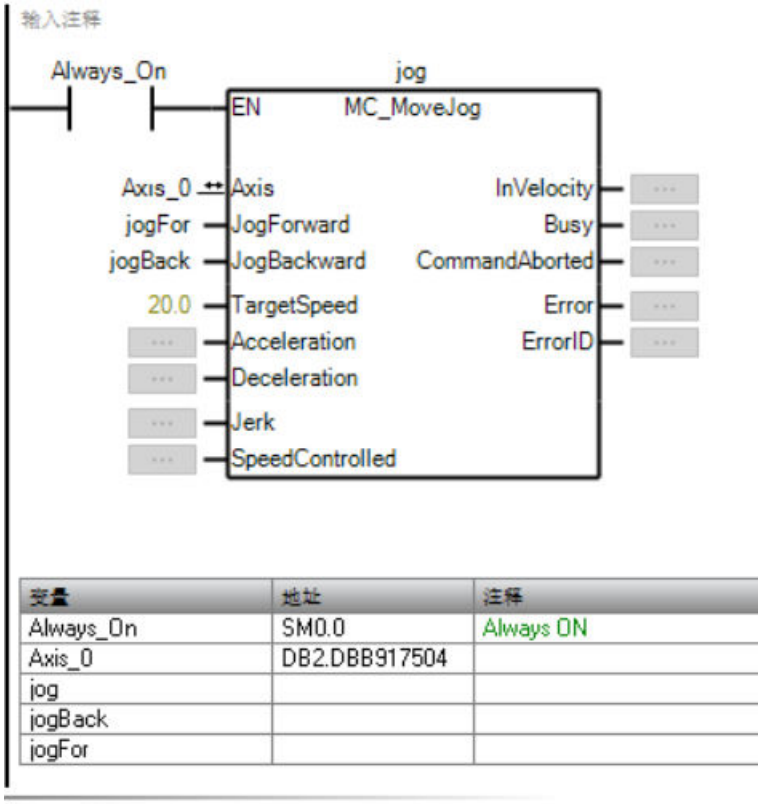
速度控制模式可以切换至位置控制模式。切换完成后，Setting_Info.Pos 将自动更新为当前的实际位置。

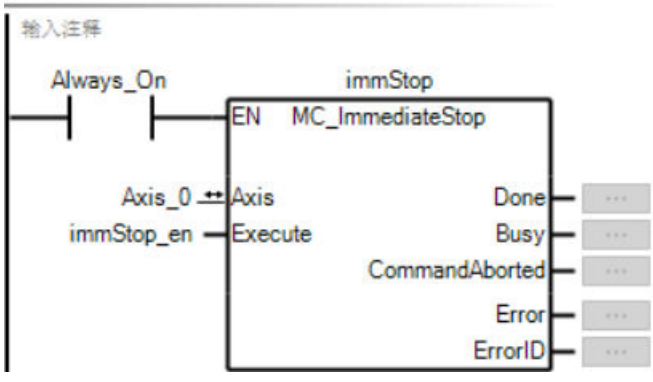
可切换到位置控制模式	可切换到扭矩控制模式	不会影响控制模式的指令	在速度模式下不能运行的指令
MC_MoveRelative MC_MoveAbsolute MC_Power MC_CamIn MC_GearIn MC_CamOut MC_GearOut MC_Home	MC_TorqueControl (mode 1)	MC_ImmediateStop MC_Stop MC_Halt MC_Reset MC_TorqueRange MC_TorqueControl (mode 0) MC_TorqueLimit (mode 0 限位模式)	MC_CamIn (以速度模式运行的轴为主轴，且 reference type=1) MC_GearIn (以速度模式运行的轴为主轴，且 reference type=1) MC_TorqueLimit (mode 1 固定停止点模式)

实例一

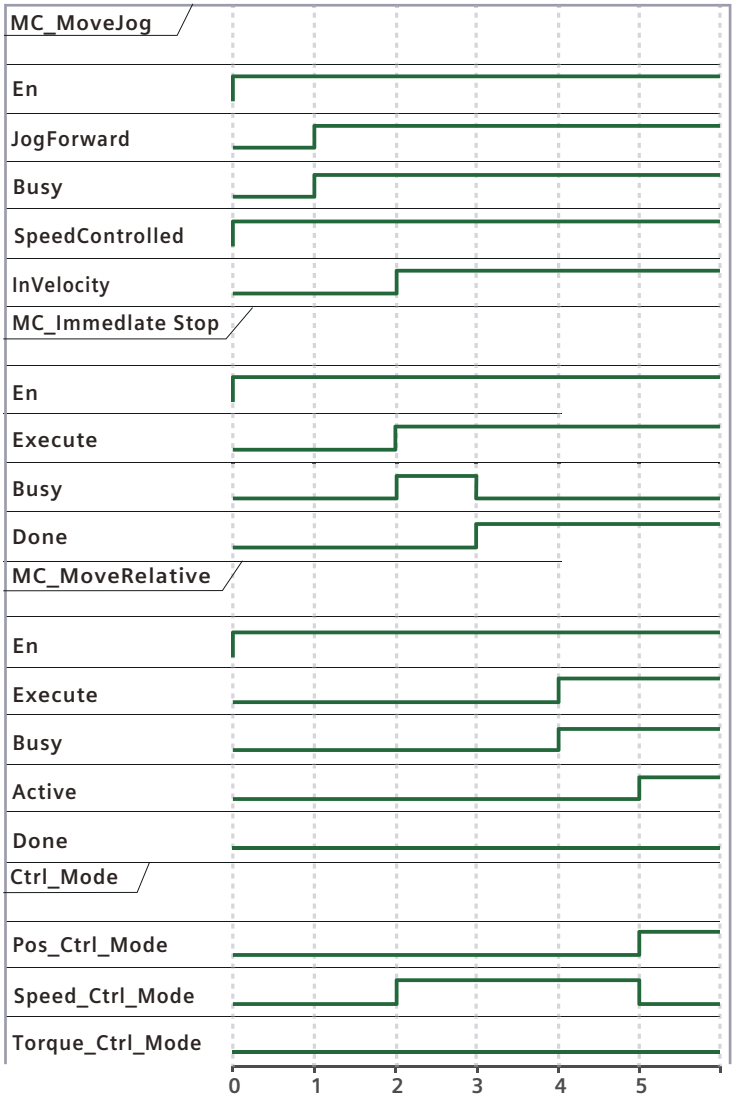
前置条件

- 轴已组态。
- 程序中使用了 MC_MoveJog 或 MC_MoveVelocity 指令。





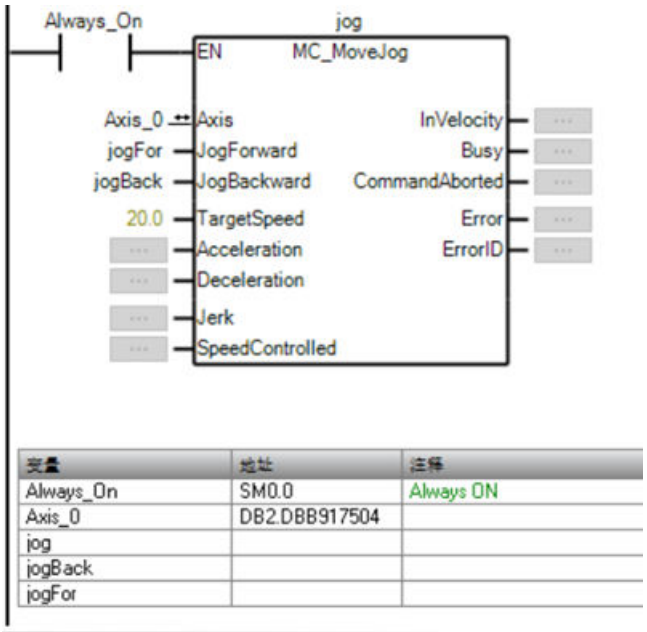
时序图



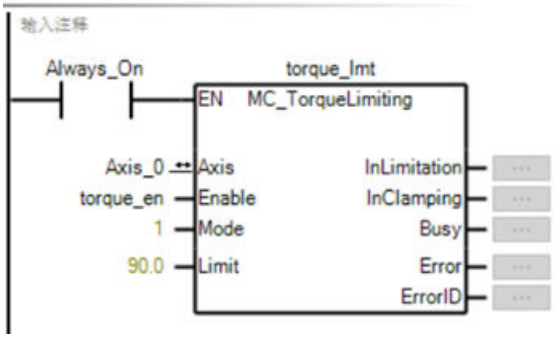
实例二

前置条件

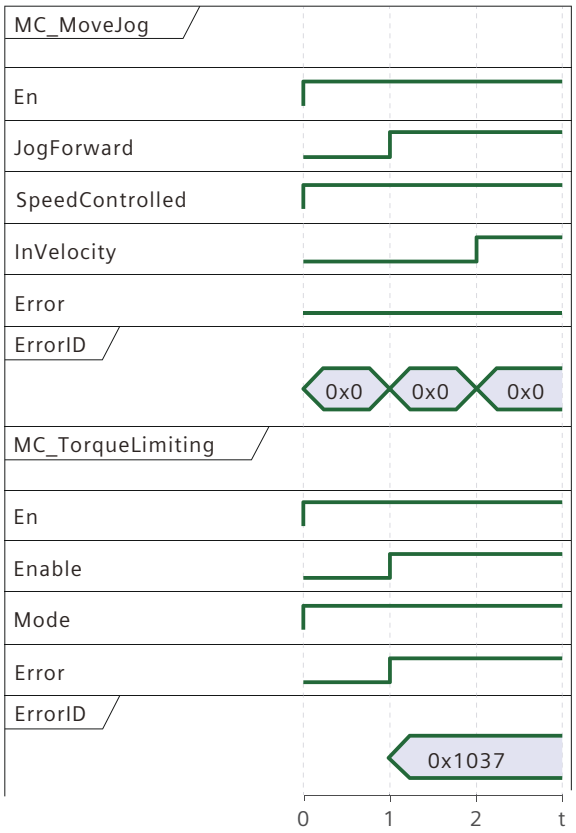
- 轴已组态。

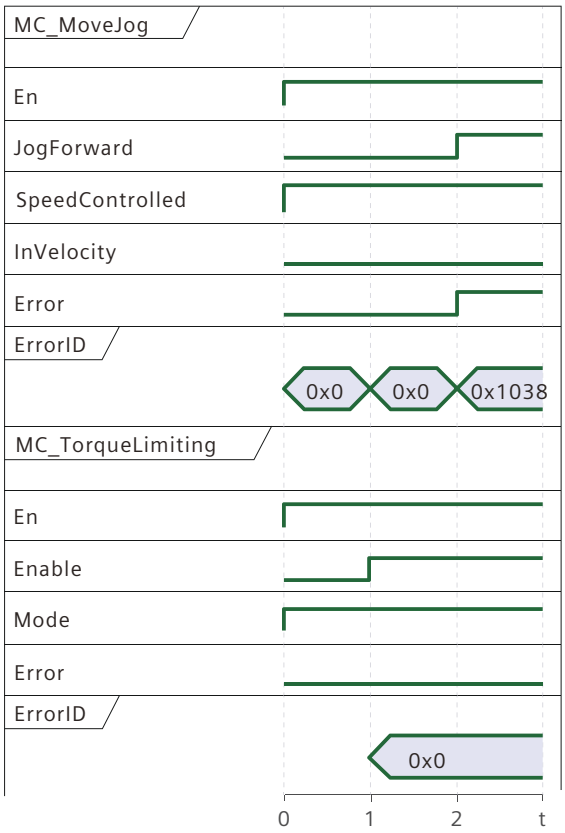


变量	地址	注释
Always_On	SM0.0	Always ON
Axis_0	DB2.DBB917504	
jog		
jogBack		
jogFor		



时序图


MC_MoveJog
En
JogForward
SpeedControlled
InVelocity
Error
ErrorID
0x0
0x0
0x0
MC_TorqueLimiting
En
Enable
Mode
Error
ErrorID
0x1037
0
1
2
t



14.3.1.3 单轴转矩控制

使用单轴转矩控制的条件

- 仅适用于 IRT 轴。
- 报文配置
  - 如需使用 torque control 指令，设置模式 mode=0 或 mode=1 时，均须确保已配置 750 号报文。
  - 若需使用 torque control 的 mode=1，除了 750 号报文外，还必须配置 105 号报文。

说明

进行转矩控制时，为安全起见请务必设定速度限制值。

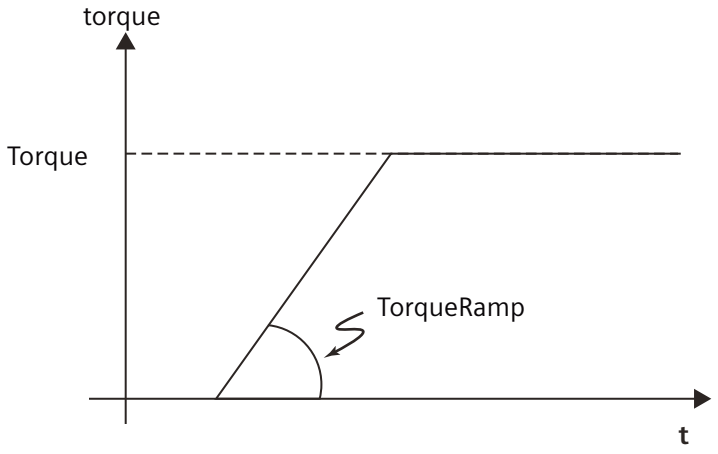
TorqueRamp 参数说明

转矩控制是指持续产生指定转矩的功能。

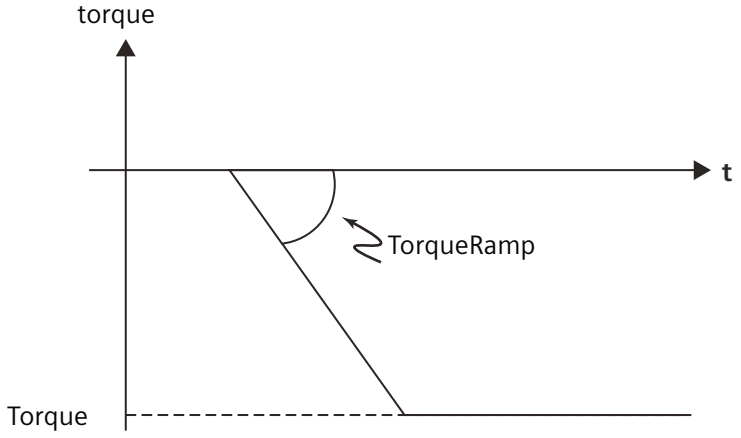
可通过 TorqueRamp(转矩斜度) 指定达到 Torque (目标转矩) 前的转矩变化率。

Torque(目标转矩) 指定为“0” 时，轴不会移动，但轴处于正在动作的状态。

- 方向指定 = 正方向时



- 方向指定 = 负方向时



MC 功能模块应用说明

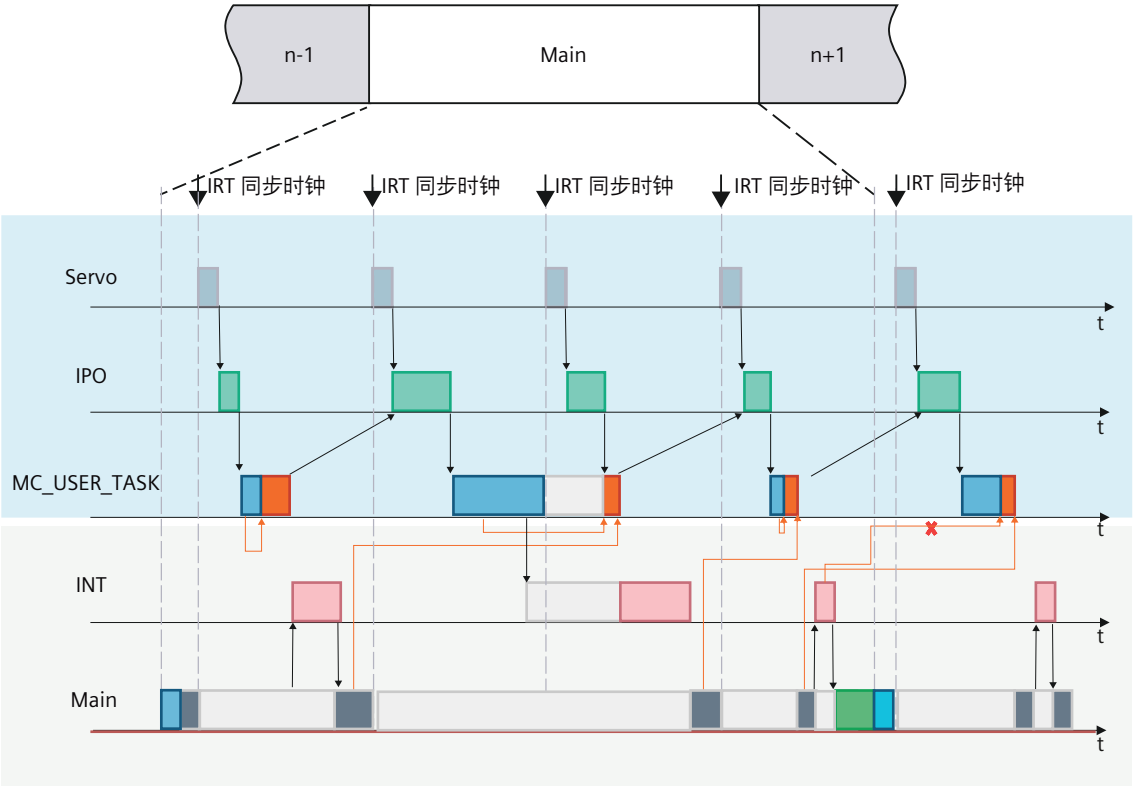
MC 功能模块中使用伺服驱动器的转矩控制模式。

伺服驱动器中通过转矩控制循环接收 MC 功能模块发送的转矩指令值进行控制。

参见

速度控制模式 (页 812)

14.3.2 任务机制



Servo

IRT 伺服控制任务

执行特征	时序关系
• 最高优先级，表示实时伺服控制任务 • 周期性执行 • 不可中断，确保控制的实时性和精确性 • 直接响应 IPO 层的插补输出	• 在每个主时钟周期内执行 • IPO 层保持严格的时序同步

IPO

插补任务

执行特征	时序关系
• 按照轴号由小到大的顺序进行插补计算 • 事件标记指示特定的插补事件触发点 • 负责生成精确的位置运动指令	• Post IPO：插补任务完成后的后处理阶段 • 添加 IPO 任务（Add IPO task：动态添加新的插补任务到执行队列 • 在运动控制用户任务（MC User Task）完成运动规划后接收新任务

运动控制用户任务（MC_USER_TASK）

该任务会在插补（IPO）任务完成后立即执行。所有用户定义的运动控制指令，无论是基于 IRT 还是 PTO（脉冲串输出）模式，都将在此任务中进行统一的解析和处理。

- 已配置 IRT 时： 任务与 IRT 的等时周期同步运行，保证高精度和确定性的运动控制。
- 未配置 IRT 时： 如果系统未配置 IRT 功能，运动用户任务将退而求其次，与 PTO 的基础 4 毫秒时钟周期同步运行。

阶段一	阶段二	阶段三
运动控制程序执行： • 处理用户定义的运动控制逻辑 • 优先级高于 Main Scan，确保运动控制的确定性	作业处理 1. 收集所有异步指令请求,包含： – Main Scan – 运动控制用户任务 – 控制面板 2. 调用运动控制作业生成 IPO 任务： – 按照优先级统一进行任务调度 – 运动规划和插补 – 向 IPO 层发送插补任务	系统监控任务

Main Scan

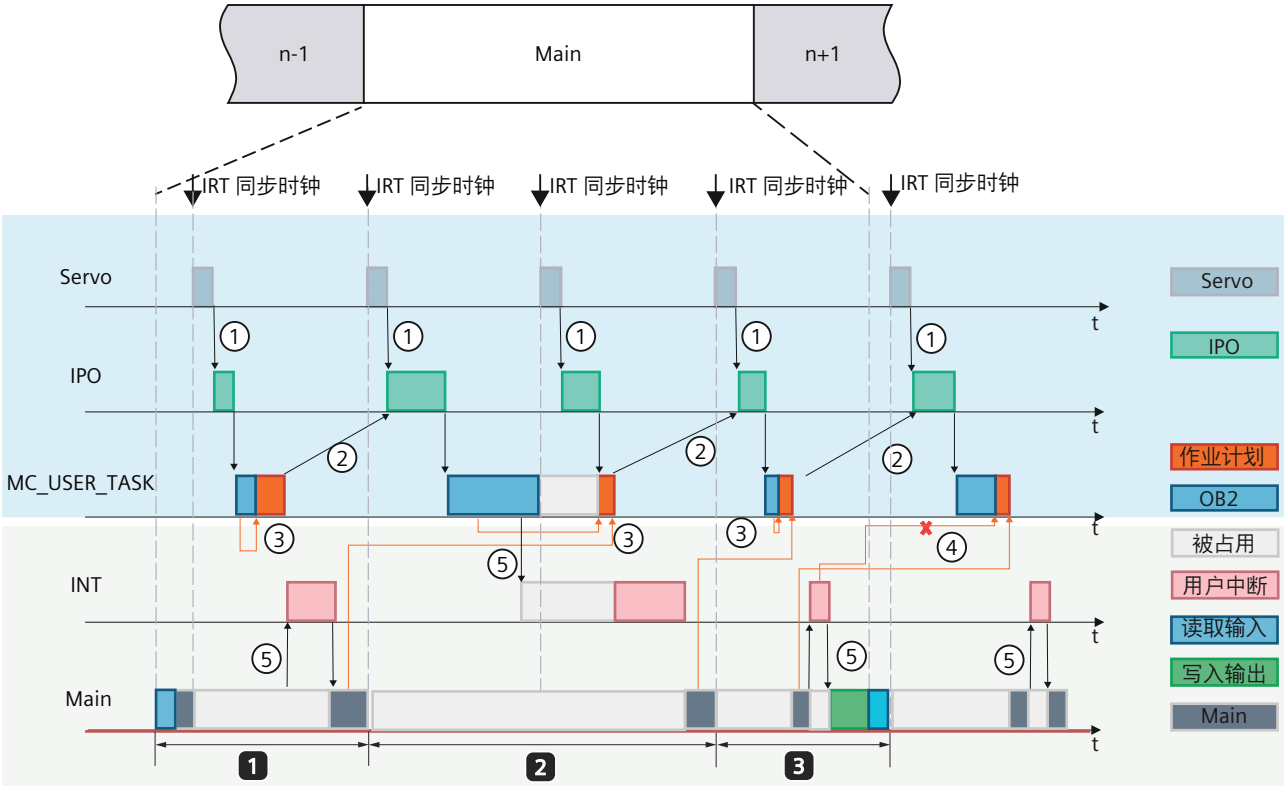
与运动控制用户任务异步执行，互不干扰。

优先级排序

Servo > IPO > MC_USER_TASK > 用户中断 > Main Scan

14.3.3 过程响应

如果配置了 IRT 轴，且运动控制用户任务（MC_USER_TASK）调用了运动指令并执行，可以确保运动控制的核心计算和插补任务得到及时处理。在这些关键的运动控制 OB 执行完毕后，剩余的时间会留给主程序来处理，直到下一个 IRT 周期开始。使用 V3.2 或更高版本的 CPU 并需配置运动控制功能时，可在程序块中插入 MC_USER_TASK。



① 事件（Event）

② 添加 IPO（插补）任务

③ 发送作业计划

④ 中断程序里面不能执行运动控制指令

⑤ 此时触发硬件中断

1 运动控制（与伺服通信 + IPO + MC_USER_TASK）在一个 IRT 周期执行完，主程序是持续扫描的，在一个 IRT 周期可能存在多次扫描。

2 运动控制（与伺服通信 + IPO + MC_USER_TASK）在 1 个 IRT 周期不能执行完。有用户中断触发但是需要等运动控制用户任务(OB2)执行完，才能执行用户中断。
即运动控制用户任务(OB2)可以打断用户中断的执行，但是用户中断不能打断运动控制用户任务(OB2)的执行。

3 中断程序里面不可以放运动控制指令

运动控制超时条件

运动控制

- 运动控制连续三次超过 IRT 周期
- 运动控制每 50 次的平均时间连续三次 ≥ IRT 周期 * 运动控制在 CPU 资源中的占比 (页 224)

MC_USER_TASK 超时条件

- MC_ USER_TASK 的扫描时间连续三次 ≥ IRT 周期 * 运动控制在 CPU 资源中的占比 (页 224)* MC_USER_TASK 在运动控制中的占比 (页 224)

14.3.4 兼容性

14.3.4.1 V3.1 和 V3.2 运动控制变更

类别	V3.1	V3.2
PN IRT 轴	不支持	支持
转矩控制模式	不支持	转矩控制： MC_TorqueControl 转矩限制： MC_TorqueRange MC_TorqueLimiting
速度控制模式	不支持	支持 PN IRT 轴和虚拟轴 MC_MoveVelocity [SpeedControlled=1] MC_MoveJog [SpeedControlled=1]
轨迹	不支持 IRT 周期	支持 IRT 周期
MC_USER_TASK(OB2)	不支持	支持
MC 指令 Jerk 引脚	未支持 Jerk 的指令： MC_Halt MC_Stop MC_CamOut MC_GearOut MC_MoveJog (减速段)	支持 Jerk 的指令： MC_Halt MC_Stop MC_CamOut MC_GearOut MC_MoveJog

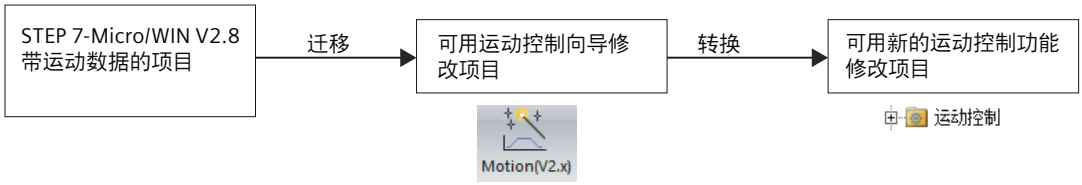
类别	V3.1	V3.2
缓冲模式	支持 Buffered 的指令： MC_Power MC_Halt MC_MoveAbsolute MC_MoveRelative	支持 Buffered 的指令： MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn
电子凸轮	1. MC_CamIn 不支持以下引脚： StartMode MasterStartDistance MasterOffset SlaveOffset 2. 从轴超速时报错	1. 从轴 PN IRT 轴和虚拟轴时， MC_CamIn 新增支持引脚： StartMode MasterStartDistance MasterOffset SlaveOffset 2. 从轴超速时采用最大速度追赶
电子齿轮	从轴超速时报错	从轴超速时采用最大速度追赶
MC_PhasingRelative	以下引脚超过限值时报错： Velocity Acceleration Deceleration	以下引脚超过限值时正常运行： Velocity Acceleration Deceleration
故障复位	只允许在 ErrorStop 下触发	允许在任何状态机下触发
探针	MC_TouchProbe: 与轴状态机无关 与回零标志无关	MC_TouchProbe: 轴状态机为 Disabled、Homing 和 ErrorStop 时，不能触发指令 回零标志未置位时，不能触发指令
虚拟轴	• 轴状态机为 Disabled 时，回零标志会丢失 • 当 CPU 切换为 RUN 模式时，轴位置会清零 • 轴变量中的速度为插补周期的平均速度	• 轴状态机为 Disabled 时，回零标志也不会丢失 • 当 CPU 切换为 RUN 模式时，轴位置也不会清零 • 轴变量中的速度为插补周期的末速度，InVelocity 管脚提前一个周期置位
软限位	提前减速，停止在软限位上 线性轴和模态轴都支持软限位	从软限位处开始减速 只有线性轴支持软限位
控制面板	必须在 PLC STOP 时才允许启动	• PTO 轴：在 PLC RUN 或 STOP 时，都允许启动 • PN IRT 轴：在 PLC RUN 时才允许启动

类别	V3.1	V3.2
MC 指令重启	自己打断自己	以下指令允许运动状态下重启： MC_MoveAbsolute MC_MoveRelative MC_Halt MC_MoveVelocity MC_Stop MC_GearIn MC_CamIn
在 INT 中调用 MC 指令	不报错	下载时报错

14.3.4.2 全新运动控制全面取代旧版运动向导

Siemens 推荐使用全新运动控制，同时保留运动向导，以便打开和修改使用 STEP 7-Micro/WIN SMART（2.8 版）创建的项目。

- 默认情况下，将禁用旧版运动向导并且灰显。
- STEP 7-Micro/WIN SMART V3 可将具有历史运动数据的 V2.8 项目迁移到 STEP 7-Micro/WIN SMART V3，并可使用运动向导或全新运动控制对项目进行修改。
  - 迁移后，将默认禁用全新运动控制和指令。可使用运动控制修改历史项目。
  - 如要使用全新运动控制来修改迁移后的项目，需要先转换 (页 828)项目。转换项目后，将禁用运动向导和历史指令。



14.3.4.3 转换为 STEP 7-Micro/WIN SMART V3 (PLCopen)运动控制

如果从较早版本的 STEP 7-Micro/WIN SMART 迁移了旧版运动控制，Siemens 建议将旧运动向导中的创建的轴转换至最新的 STEP 7-Micro/WIN SMART V3 运动控制。

说明

转换后无法恢复

该操作无法恢复，请备份原文件。

说明

项目必须使用新指令重新编程。

转换完成后，“运动向导”生成的数据块和运动指令 SBR 将不可用。项目使用的指令也会失效。

必须使用新的运动控制指令对项目重新编程。

说明

轴组组态已丢弃

当前版本中已丢弃旧轴组组态。

条件

- 项目已迁移到 STEP 7-Micro/WIN SMART V3 中。

转换项目

1. 在项目树中打开“向导”(Wizards) 文件夹。
2. 双击“运动控制 (V2.x)”(Motion(V2.x))，然后选择“转换”(Convert)。
3. 单击“开始”(Start) 按钮继续操作。
4. 双击转换后的轴，打开组态视图，查看每个面板并进行相应组态。

匹配规则

一些功能在转换前后的匹配规则。

运动向导	3.x 运动控制	备注
轴名称	轴名称	轴名称中的空格替换为下划线“_”
轴包络 **注：**包络与轴绑定。	全局变量表 1 中的 SYS_AXIS 类型变量 **注：**任何轴都可使用变量表 1	• 仅当选择相对或绝对模式时，包络中的 vel、endPosition 和注释才会转换。对于其他两种模式，必须在后续的包络指令中指定引脚模式和速度操作。

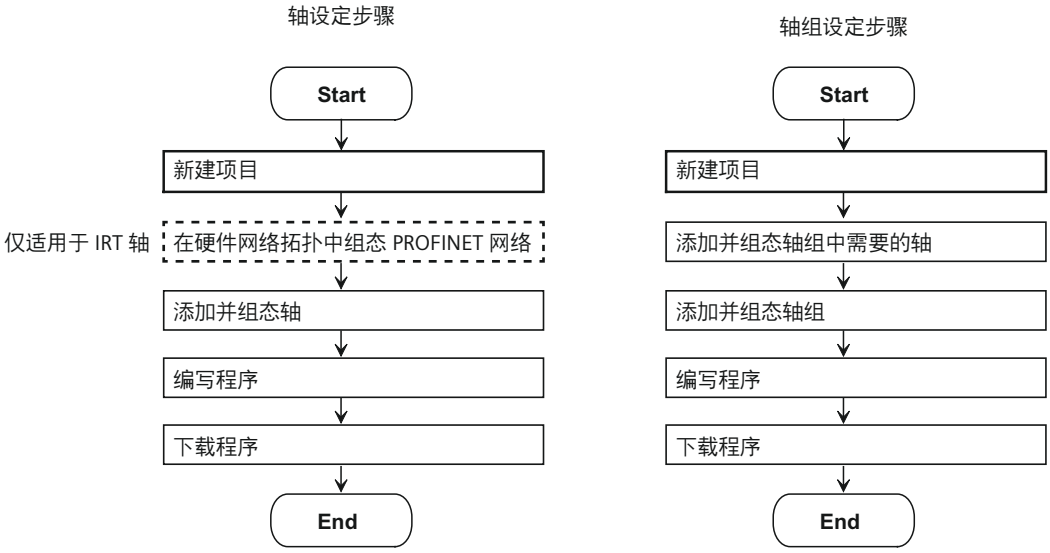
运动向导		3.x 运动控制	备注
加速时间/减速时间：ms		加速时间/减速时间：单位/s^2	• 加速度 = (maxVel - ss_speed) / accTime * 1000 • 减速度= (maxVel - ss_speed) / decelTime * 1000 • ss_speed：启停速度。PTO 速度必须大于 200 个脉冲。 – 脉冲：ss_speed=200 – 其它：ss_speed=200.0 * (工作台旋转一圈的移动量 / (电机旋转一圈脉冲数)) maxVel：最大速度
加加速度时间 (ms)		加加速度时间（单位/s^3）	加加速度 = acc / jerkTime * 1000
测量系统	英尺	英寸	
	弧度	度	
	米	厘米	
单相（1 个输出）		脉冲 + 方向	
RPS 搜索顺序 2		原点标记（正）+ 偏移	
参考点偏移量		已删除	
“搜索顺序”（脉冲）中的 ZP 输入计数		“回原点”(Homing) 页面中的偏移量 (mm)	偏移量 [1] = (脉冲 - 1) * 每转单位 每转单位：已组态的轴（工作台）旋转一圈所移动的距离。
轴组，2D/3D 直线插补		轴组类型，坐标系统 (2D/3D)	
轴组路径段		全局变量表中的变量表 1	转换后的变量名称以“Grpx_”开头
反冲补偿		已删除	

[1] 如果接近方向为负，则偏移值为负数；如果为脉冲输出，则换算后偏移量为“脉冲-1”。

14.4 组态

14.4.1 设定步骤

轴的设定流程



说明

在新建包含 IRT 轴的项目时，请确保选择 S7200 SMART G2 CPU 且固件版本不低于 3.2。

① 双击选择 CPU 及其版本。

② 双击进入硬件网络拓扑（仅适用于 IRT/RT 网络）。

③ 添加、组态运动控制轴、轴组、凸轮。

④ 使用运动控制指令编写运动控制程序。

14.4.2 硬件网络拓扑视图

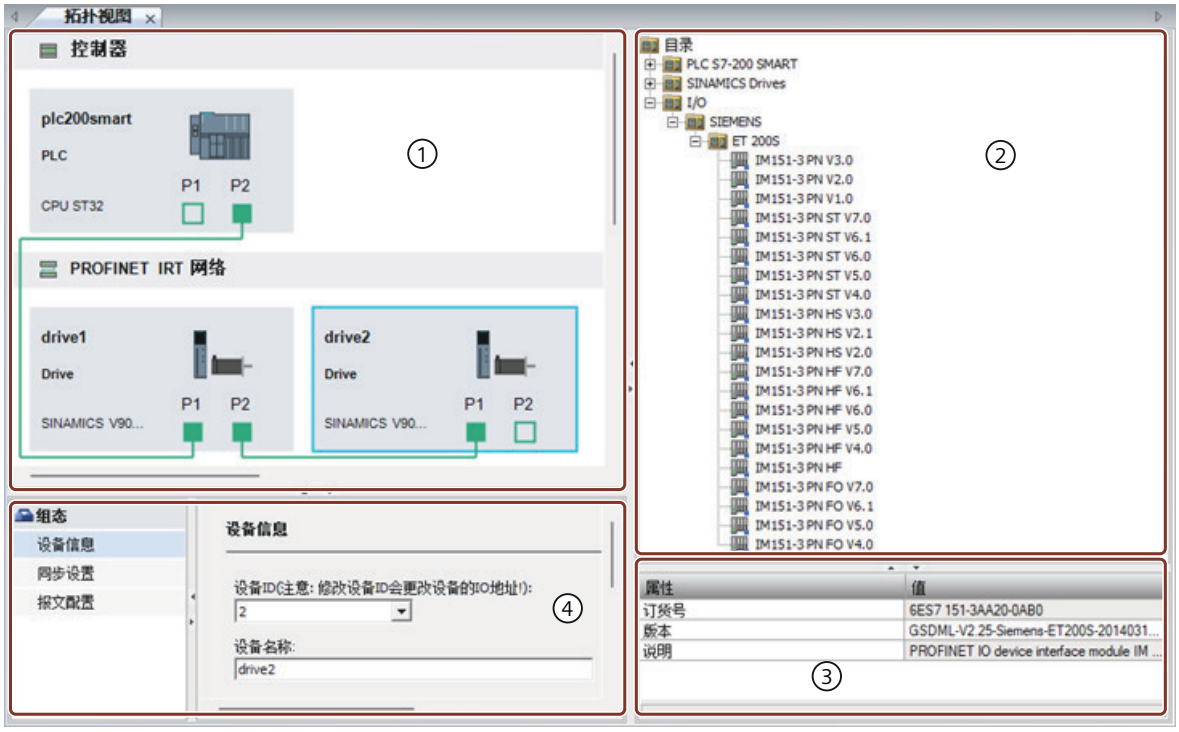
14.4.2.1 硬件网络拓扑

硬件网络拓扑的功能

硬件网络拓扑用于 PROFINET RT+IRT 网络的组态。双击项目树中的“硬件网络拓扑”，打开硬件网络拓扑。 硬件网络拓扑用于对设备和模块进行组态和参数分配。

硬件网络拓扑的结构

硬件网络拓扑由以下部分组成：



① 拓扑视图 (页 835)

② 设备目录

③ 设备属性

④ 设备组态

可以使用鼠标更改网络视图中区域之间的分隔。要变更分割线的位置，在区域间隔间单击，并在按住鼠标按钮的同时左右移动分隔线更改间距大小。通过快速拆分器（两个小箭头键），可以使用单击来最小化视图、最大化视图或恢复上一次选择的拆分。

为了高效地管理和配置网络资源，硬件拓扑网络提供了两种视图：拓扑视图和设备视图。在拓扑视图中，您可以对整个网络拓扑结构进行组态。当您需要对单个设备及其内部模块进行编辑和配置时，则须进入设备视图。

工具栏

该工具栏提供以下功能：

图标	含义
	将 PLC 设置为 RUN 模式
	将 PLC 设置为 STOP 模式
	编译
上传	将项目从 PLC 上传到 STEP 7-Micro/WIN SMART V3
下载	从 STEP 7-Micro/WIN SMART V3 下载到 PLC
	拓扑比较 (页 851)
	扫描拓扑 (页 851) 启动对当前在线拓扑的检测。

设备目录

当您打开硬件网络拓扑时，设备目录树将自动加载以下设备：

- CPU 内嵌的 PROFINET 设备
- 导入 GSD 文件支持的 PROFINET 设备

设备属性

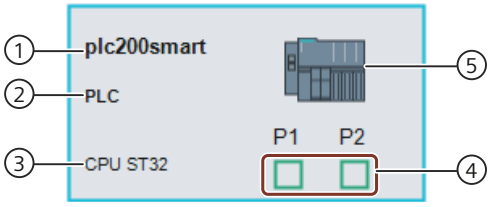
当您在设备目录中选择一个设备时，设备属性窗口将显示该设备的基本属性。

属性	说明
定货号	只读
版本	您可以通过下拉菜单选择设备的 GSD 文件版本。如果系统中安装了多个包含相同设备的 GSD 文件，您可以在此处进行选择。完成设置后，设备在树状视图中的版本将更新为您所选的版本。默认情况下，系统会自动选择最高版本的 GSD 文件。
说明	只读

14.4.2.2 拓扑视图

拓扑视图中，整个页面分为三层：控制器层、PROFINET IRT 网络层、 PROFINET RT 网络层。

设备



① 设备名称　　如设备名称修改，设备视图将同步更新。

② 设备类型　　PLC, Drive, I/O

③ 设备型号　　设备的型号与系统中选择的类型保持一致。

④ 端口信息
- 只有支持 IRT 功能的 PLC 和 IRT 设备会显示两个端口的详细信息，这两个端口分别标识为 P1 和 P2。
- RT 设备： 不会提供具体的端口信息。

⑤ 设备示意图

控制器层

系统将根据系统块中选定的 CPU 自动创建一个设备。

- 控制器层仅包含这一个自动创建的设备。
- 此设备不允许被删除。
- 不允许从硬件目录或其他层向此层拖动其他设备。
- 当 CPU 从支持 IRT 切换到不支持 IRT 时，现有的拓扑组态所有信息将保持不变。若存在不兼容情况，系统将在编译时报错。

PROFINET IRT 层

PROFINET IRT 网络层用来添加 IRT 设备。

- 在该层中，最多可配置 8 个 IRT 设备。同时，所有 IRT 和 RT 设备的数量总和不得超过 10 个。
- 在 IRT 层的设备通过“手拉手”的方式连接。

- 该层被划分为两行，每行最多支持放置 4 个设备。
- 如果当前 CPU 的类型和版本不支持 IRT 功能，则 IRT 网络层将禁用设备拖入操作。

PROFINET RT 网络层

PROFINET RT 网络层用来添加 RT 设备。

- 在该层中，最多可配置 10 个 RT 设备。同时，所有 IRT 和 RT 设备的数量总和不得超过 10 个。
- 该层被划分为两行，每行最多支持放置 5 个设备。

说明

仅 RT 设备支持访问设备视图。IRT 设备暂不提供此设备视图功能。

14.4.2.3 设备视图

设备视图用于进一步展示和管理 RT 类型设备的详细信息。您可以通过以下任一方式进入 RT 设备的设备视图：

- 在拓扑视图中双击 RT 设备。
- 在拓扑视图中右键单击 RT 设备，从弹出的上下文菜单中选择 “转到设备视图” 选项。

说明

仅 RT 设备可进入设备视图，IRT 没有设备视图。

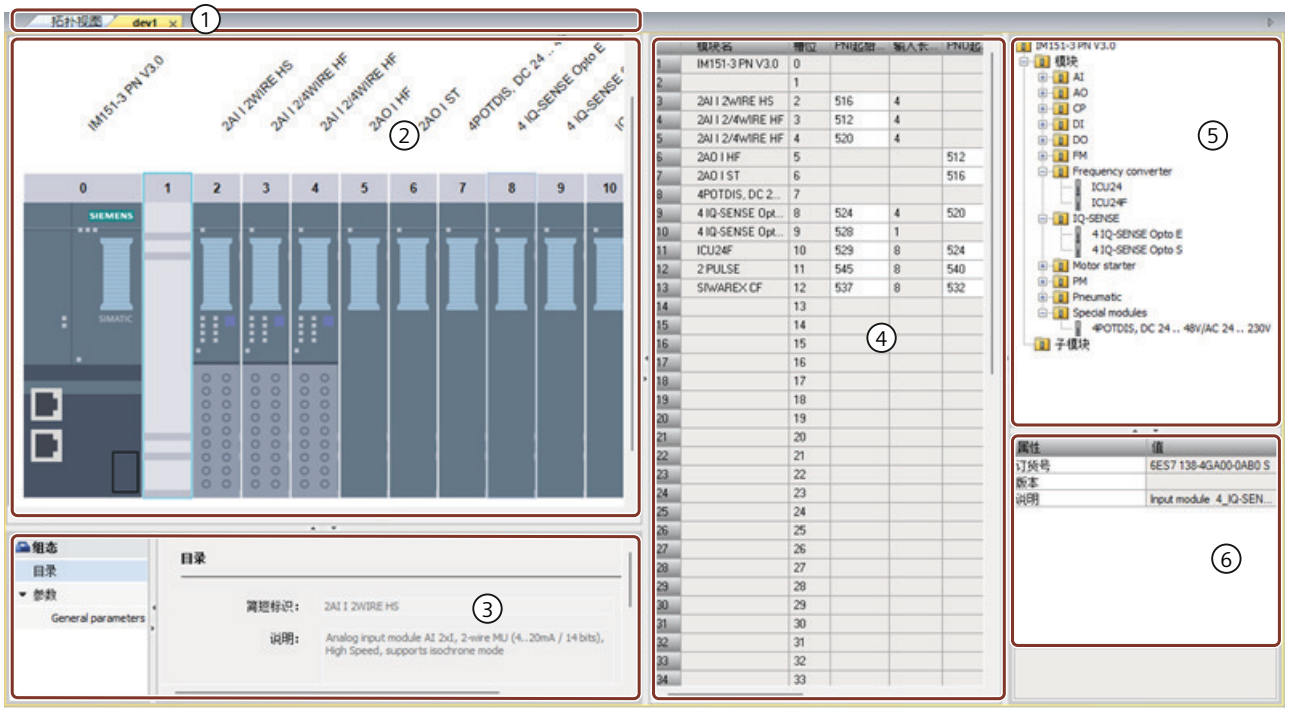
简介

在设备视图可执行以下任务：

- 分配设备参数
- 组态和分配模块参数

结构

下图显示了设备视图的组成部分：



① 切换开关：设备视图/拓扑视图

② 设备视图的图形区域

③ 模块组态：在图形区域或者表格区域选中模块或子模块时，其对应参数将在模块组态中显示，同时左侧提供导航栏。

④ 设备视图的表格区域

⑤ 模块树

⑥ 模块属性：在模块树选中模块时，其对应参数将在模块属性中显示。

可以使用鼠标更改设备视图中区域之间的分隔。要变更分割线的位置，在区域间隔间单击，并在按住鼠标按钮的同时左右移动分隔线更改间距大小。通过快速拆分器（两个小箭头键），可以使用单击来最小化视图、最大化视图或恢复上一次选择的拆分。

表格区域

通过设备视图的表格区域可总览所用的模块以及最重要的技术数据和组织数据。

14.4.3 编辑设备和网络

14.4.3.1 将设备添加到 PROFINET 网络

当 IRT 和 RT 设备被添加到拓扑视图时，系统会自动为这些设备分配唯一的 ID。ID 分配过程将从最小的可用 ID 开始，并按顺序递增地分配给每个 IRT/RT 设备。

对于添加在 PROFINET 网络中的设备，其命名将按照添加顺序自动生成。

- 驱动器命名格式为 drive1, drive2, drive3...
- I/O 模块命名格式为 dev1, dev2, dev3...
- PLC 模块命名格式为 PLC1, PLC2, PLC3...

拖放设备

要通过项目树将设备添加到硬件配置中，请按以下步骤操作：

1. 打开硬件网络拓扑。
2. 在设备目录中选择所需的设备。
3. 单击所选的设备，将其选中。
4. 必要时，通过设备属性窗口中的下拉列表设置固件版本。
5. 将设备拖放到拓扑视图中。

设备的可拖放网络取决于其支持的 PROFINET 类型：

- RT 设备： 只能拖动到 RT 网络，不能拖动到 IRT 网络。
- IRT 设备： 只能拖动到 IRT 网络。
- 同时支持 RT 和 IRT 的设备： 可以拖动到 RT 网络 或 IRT 网络。

批量添加

1. 在设备树视图中，您可以通过右键菜单中的“批量添加”选项。
2. 选择批量添加设备的位置以及要添加的数量。

3. 点击“确认”。

PROFINET device 变量

将 PROFINET 设备添加到硬件网络拓扑的组态中时，STEP 7-Micro/WIN SMART V3 会将 PROFINET 设备变量添加到 PROFINET device 变量表中。

14.4.3.2 转至在线

通过此功能您可以访问硬件网络拓扑中设备的 Web Server。

“转至在线”功能仅适用于在拓扑页面配置的以下设备类型：

- S200
  支持配置于 IRT 域或 RT 域。
- G200 Basic
  仅支持配置于 RT 域。

说明

系统支持同时打开最多十个在线页面，但每个设备一次仅能打开一个。

操作流程

1. 在 拓扑视图选中 S200 或 G200 Basic 设备后，在右键菜单中选择“转至在线”选项。
2. 若检测到设备，系统获取设备当前 IP 地址：
   – 如 IP 地址为 0.0.0.0， 系统弹出对话框，提示您需先对设备进行 IP 地址设置，方可继续后续操作。
   – 如 IP 地址有效， 系统打开该设备的 Web Server。
3. 若未检测到目标设备在线，系统将提示框“当前设备不在线”。

说明

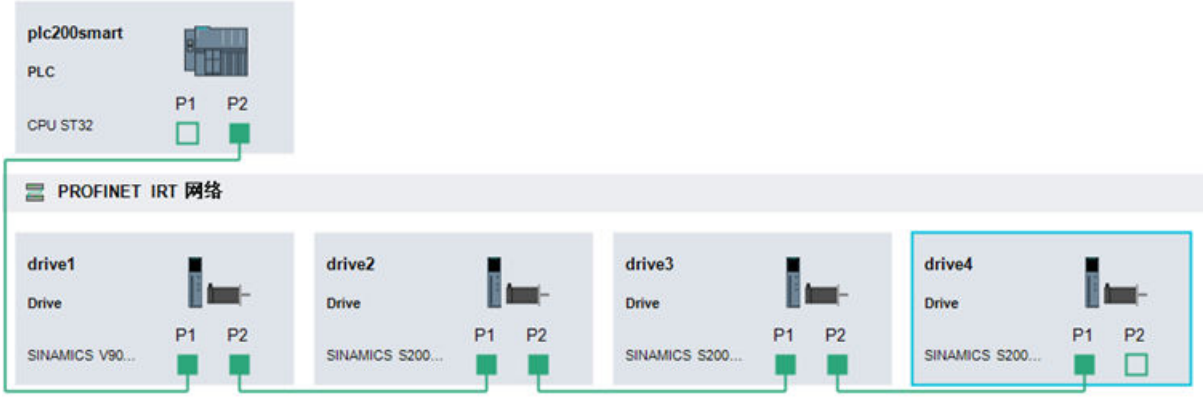
当首次访问设备的 Web Server 时，可能会弹出“不信任”安全警告页面。此时，您需手动确认并选择“继续访问”，方可正常进入。

建议您在首次访问 Web Server 后，根据弹窗引导，下载安装并信任设备证书

14.4.3.3 设备连接

IRT 设备在添加至拓扑视图后，IRT 设备之间以及 IRT 设备与控制器 (Controller) 之间会自动建立起连接。RT 设备不支持任何形式的连接。

系统不支持手动建立设备间的连接。



控制器连接

作为控制器，其连接端口具有灵活性，可以通过 P1 或 P2 端口与 IRT 设备的 P1 端口进行连接。

可通过右键“交换端口”功能调整 PLC 设备通信端口。

IRT 设备之间连接

IRT 设备之间采用自动的“手拉手”连接模式。即，一个 IRT 设备的 P2 端口会自动连接到下一个相邻 IRT 设备的 P1 端口。

以 SINAMICS S200 PROFINET 伺服驱动器为例，IRT 设备的 P1 端口对应伺服驱动器 X150-P1 口，IRT 设备的 P2 端口对应伺服驱动器 X150-P 2 口。

14.4.3.4 复制/粘贴/删除设备

删除设备

通过以下任一方式删除设备：

- 用鼠标右键点击选中的设备，在弹出的上下文菜单中选择“删除”。
- 选中设备，按下 Delete 键。

说明

删除规则

- 控制器 (Controller)：不允许删除。
- IRT 设备：确认删除后，系统将执行删除操作，并自动重新调整剩余 IRT 设备之间的“手拉手”连接，以保持网络拓扑的连贯。
- RT 设备：确认删除后，系统将执行删除操作，并且被删除 RT 设备之后的所有设备会自动向前移动一位。

复制/粘贴设备

可在拓扑视图中复制硬件设备。复制的设备和模块可以粘贴到拓扑视图的 IRT 网络或 RT 网络。

要复制硬件设备，请按以下步骤操作：

1. 在拓扑视图选择要复制的硬件设备。
2. 从右键菜单中选择“复制”。
   如果“复制”菜单项不可用，说明所选内容中至少包含一个不能复制的组件。

3. 选择要粘贴剪贴板内容的位置。
4. 从右键菜单中选择“粘贴”。
   如果“粘贴”菜单项不可用，说明剪贴板为空或至少包含一个无法在该位置粘贴的组件。

说明

命名规则

- 如果被复制设备的名称以数字结尾（例如 Device1），则粘贴的设备将自动命名为该数字递增后的名称（例如 Device2）。如果递增后的名称已存在，系统将继续递增（n+2, n+3 等），直至找到一个唯一的名称。
- 如果被复制设备的名称不以数字结尾（例如 Sensor），则粘贴的设备名称将为原名称后加数字“1”（例如 Sensor1）。

14.4.3.5 移动/交换/插入设备

您可以在拓扑视图中选中设备，然后拖动鼠标来执行设备的移动、交换或插入操作。

规则

- 控制器 (Controller)： 控制器设备不允许被移动或交换。
- RT 设备： RT 网络层中的设备没有严格的顺序依赖性。因此，RT 层内部设备之间的移动、交换或插入操作没有实际意义，系统不允许在 RT 设备之间执行这些操作。
- 在执行移动或插入操作时，目标位置必须支持源设备的类型。否则，操作将无法完成。

说明

当跨层移动或插入操作可能因缺少报文信息等原因导致失败时，请根据系统提示信息进行处理。

移动设备

说明

当设备在不同网络层之间进行移动时，系统将自动清除其配置信息。

要移动设备，请按以下步骤操作：

1. 选择要移动的硬件设备。
2. 将设备拖放到将要移动到的位置。

所选的设备将移动到目标位置。

交换设备

说明

此操作仅支持在 IRT 网络层内的硬件设备之间进行位置互换。

要交换设备，请按以下步骤操作：

1. 选择要移动的设备。
2. 将设备拖放到将要交换到的设备上。

设备位置互换。

插入设备

说明

此操作仅支持在 IRT 网络层内的设备之间进行插入。

当目标位置位于两个设备之间时，拖放设备，系统将执行插入操作：

1. 选择要插入的设备。
2. 将设备拖放到目标位置。

设备插入目标位置。

14.4.4 组态设备

14.4.4.1 组态控制器

要在拓扑视图中更改控制器的属性和参数，请按以下步骤操作：

1. 在拓扑视图中，在控制器层中选中 PLC。
2. 在组态窗口点击“设备角色”。

   说明

   将 CPU 组态为控制器时，请确保已为 PROFINET 设备导入 GSDML 文件。

3. 在组态窗口点击“以太网”。

以太网
- 作为控制器时：必须配置固定 IP 地址
- 作为智能设备时：支持固定 IP 或动态分配

站名称　如您更改站名，设备名称将同步更新到设备视图中。

4. 在组态窗口点击“通信”，为控制器分配“发送时钟”和“启动时间”。

“启动时间”设置影响 CPU 通电后其切换至 RUN 模式所需的时间。“启动时间”的默认值为 10 s，最大值为一分钟：

– 如果“启动时间”设置为零，CPU 将立即切换到 RUN 模式。
– 如果在“启动时间”内建立完整连接，则 CPU 会在建立连接后切换至 RUN 模式。
– 如果“启动时间”结束后未建立连接，则 CPU 在“启动时间”结束后切换至 RUN 模式。

说明

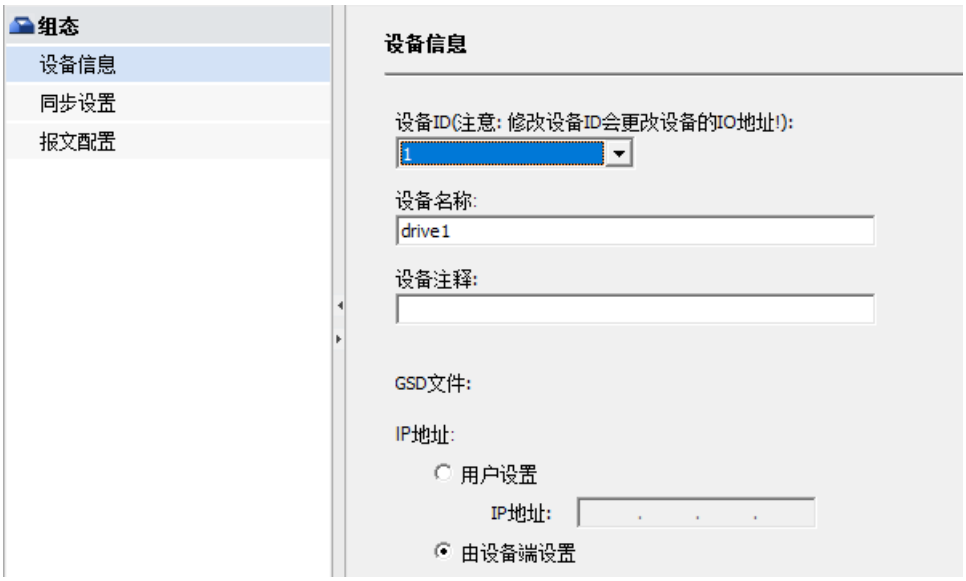
若当前项目的“发送时钟”设置为 1ms，当您添加最快发送时钟为 2ms 的 IRT 设备时，系统将自动把“发送时钟”调整为 2ms，并给出相应提示。

14.4.4.2 组态 IRT 设备

在硬件网络拓扑中插入设备后，可在拓扑视图中对设备的默认属性进行编辑。

要在拓扑视图中更改设备的属性和参数，请按以下步骤操作：

1. 在拓扑视图中，在 IRT 层中选中设备。
2. 在组态窗口点击“设备信息”。



表格标题	含义
设备 ID	模块通过 PROFINET 连接 PLC 时，PROFINET 的设备编号。
设备名称	设备名称必须和所插入模块的名称一致。 示例： • 如果模块插入一个名为“IODevice1”的前端模块中，则必须在此处设置为“IODevice1 ”。
设备备注	为设备添加注释。
IP 地址	组态 IP 地址的方法有两种： • 用户设置：由控制器给设备分配 IP 地址 • 固定 IP

3. 在组态窗口点击“同步设置”。

等时同步

更新时间	同控制器“通信”中设置的“发送时钟”。
数据保持	可将数据保持时间设定为“更新时间的 n 倍。取值范围为 3-255，默认值为 3。
Ti/To 值	Ti/To 值有以下两种设置： • 自动最小化： 自动计算并采用 Ti/To 最小值。 • 手动： 只有在选择了“手动”设置时才可以编辑 Ti 和 To。

4. 在组态窗口点击“报文设置”，选择或修改配置报文。

– 主报文：可选 105/5 号报文， 默认报文为 105 号。
– **附加报文：**可选 750 号报文或不启用， 默认不启用。
– 激活诊断：若设备的 GSD 文件包含“激活诊断”参数，则会显示该诊断功能；否则不显示。

通信时间 T_i、T_o、T_{DC}

计算跟随误差时，不会将设定值到驱动器的传输时间和实际位置值到控制器的传输时间计算在内。传输时间由以下通信时间得出：

- T_i：导入过程值的时间
- T_o：导出过程值的时间
- T_{DC}： PROFINET 接口的发送时钟

跟随误差是通过从通信时间 $T_i + T_o + T_{DC}$ 和 运动控制周期 T_{servo} 之前的延迟位置设定值中减去控制器中的实际位置计算得出的。

报文类型

报文	简要说明
标准报文	
5	• 控制字 STW1 和 STW2，状态字 ZSW1 和 ZSW2 • 速度设定值 32 位 (NSET)，实际速度值 32 位 (NACT) • 实际编码器值 1（G1_XIST1、G1_XIST2）（电机编码器） • 动态伺服控制 (DSC)[1)] – 速度预控制值 – 位置差值 (XERR) – Kv 因子增益位置控制 (KPC)
Siemens 报文（含扭矩限值）	
105	• 控制字 STW1 和 STW2，状态字 ZSW1 和 ZSW2 • 速度设定值 32 位 (NSET)，实际速度值 32 位 (NACT) • 实际编码器值 1（G1_XIST1、G1_XIST2）（电机编码器） • 动态伺服控制 (DSC)[1)] – 速度预控制值 – 位置差值 (XERR) – Kv 因子增益位置控制 (KPC) • 扭矩限值

1) 要使用动态伺服控制 (DSC)，必须将驱动器的电机编码器（报文中的第一个编码器）用作轴的第一个编码器。

14.4.4.3 组态 RT 设备

在硬件网络拓扑中插入设备后，可在拓扑视图中对设备的默认属性进行编辑。

要在拓扑视图中更改设备的属性和参数，请按以下步骤操作：

1. 在拓扑视图中，在 RT 层中选中设备。
2. 在组态窗口点击“设备信息”。

表格标题	含义
设备 ID	模块通过 PROFINET 连接 PLC 时，PROFINET 的设备编号。
设备名称	设备名称必须和所插入模块的名称一致。 示例： • 如果模块插入一个名为“IODevice1”的前端模块中，则必须在此处设置为“IODevice1 ”。
设备备注	为设备添加注释。
IP 地址	组态 IP 地址的方法有两种： • 用户设置：由控制器给设备分配 IP 地址 • 固定 IP

3. 在组态窗口点击“同步设置”。

更新时间： 数据交换的周期。
数据保持：默认值为 3。

14.4.4.4 添加/删除/移动 RT 模块

简介

在拓扑视图中将设备从硬件目录添加到组态后，即可向设备中添加模块。设备视图中图形区域与表格区域实现联动。当您在一个区域进行修改时，另一区域将自动同步刷新，确保数据和配置始终保持一致。

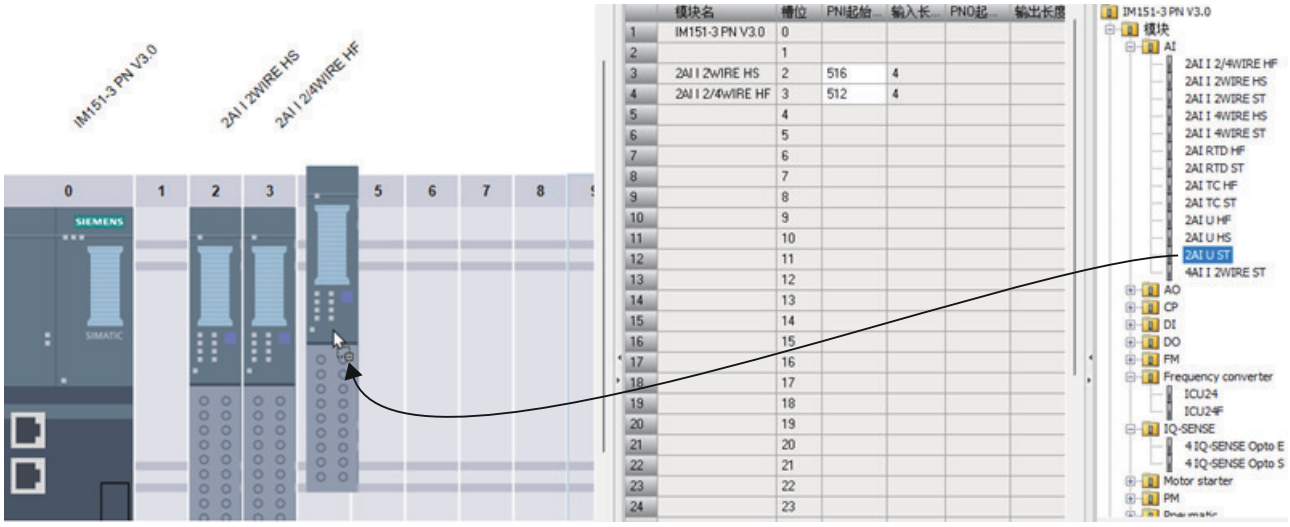
说明

不允许将任何其它模块拖入或移动到 SLOT-0。

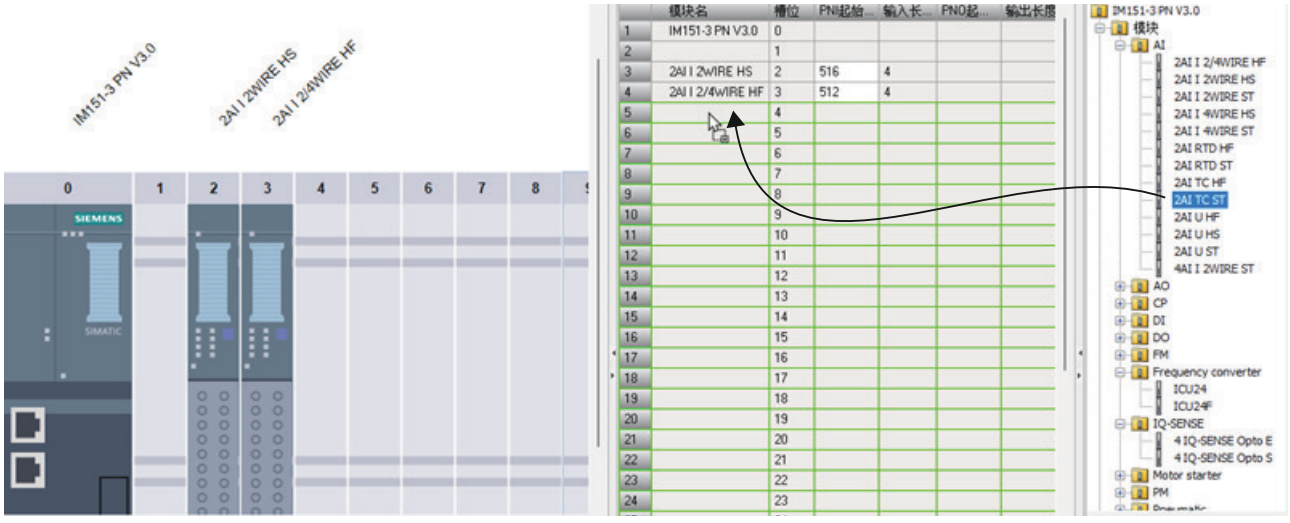
添加模块

从硬件目录中添加模块，请执行以下步骤：

1. 在硬件目录中选择所需的模块。
2. 必要时，通过硬件目录中的下拉列表设置固件版本。
3. 通过拖放操作将该模块从硬件目录移动到图形或表格区域内可用的有效插槽中：
   – 将模块拖到图形区域空闲的插槽上。



   – 将模块拖到表格区域空闲的插槽上。



添加模块后，STEP 7-Micro/WIN SMART V3 会将 PROFINET 设备变量添加到 PROFINET device 变量表中。

删除模块

通过以下任一方式删除模块：

- 用鼠标在图形或表格区域右键点击模块，在弹出的上下文菜单中选择”删除“。
- 在图形或表格区域选中模块，按下 Delete 键。

移动模块

要移动模块，请按以下步骤操作：

1. 选择要移动的模块。
2. 将模块拖放到将要移动到的位置。

所选的模块将移动到目标位置。

14.4.5 拓扑比较

拓扑比较用于对比当前工程中 IRT 拓扑视图的组态与 PLC 实际连接的端口网络拓扑信息。通过此功能，您可以及时发现并定位实际网络与配置之间连接关系的所有不一致之处。

说明

如果拓扑视图中未组态 IRT 设备，则比较功能将被禁用。

前提条件

为确保能被正确扫描，设备必须满足以下条件：

- 已连接到控制器。
- 已初始化 IP 地址。

操作

要找出已组态拓扑和实际拓扑之间的差异，执行以下步骤

1. 工具栏中，单击“比较”按钮。
2. 点击“扫描拓扑”按钮扫描实际连接的设备和配置。

说明

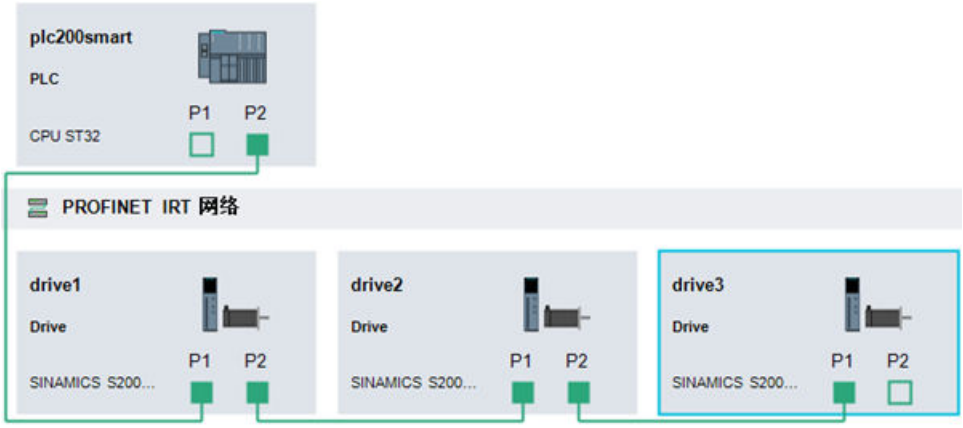
如果设备配置或连接发生更改，请手动点击“扫描拓扑”按钮，以确保系统准确反映最新变动。

结果

扫描完成后，系统将显示更新时间点以及拓扑对比的详细信息。

特殊情况

在进行拓扑比较时，由于无法直接从控制器（Controller）查询到设备的完整网络连接关系，当组态中存在 IP 地址配置为 0.0.0.0 的 IRT 设备时，系统会尝试通过其相邻设备的网络连接关系来推断这些 0.0.0.0 设备的实际连接。以下将根据设备与控制器的连接方式，阐述几种具体情况（假设物理连接如下图所示）：



- 当 IP 为 0.0.0.0 的设备直接连接到控制器时（例如 Drive1）：
  - PLC200Smart 对应 0.0.0.0 的设备的 P2 的伙伴关系为空。
  - Drive1 对应控制器的 P1 的伙伴关系也会显示为空。
  - P2 端口仍可被正确识别和显示。
- 当 IP 为 0.0.0.0 的设备非直接连接到控制器时（例如 Drive2 或 Drive3）：可以正确推断出 0.0.0.0 设备的网络连接关系，正确显示所有端口的伙伴关系和连接状态 。

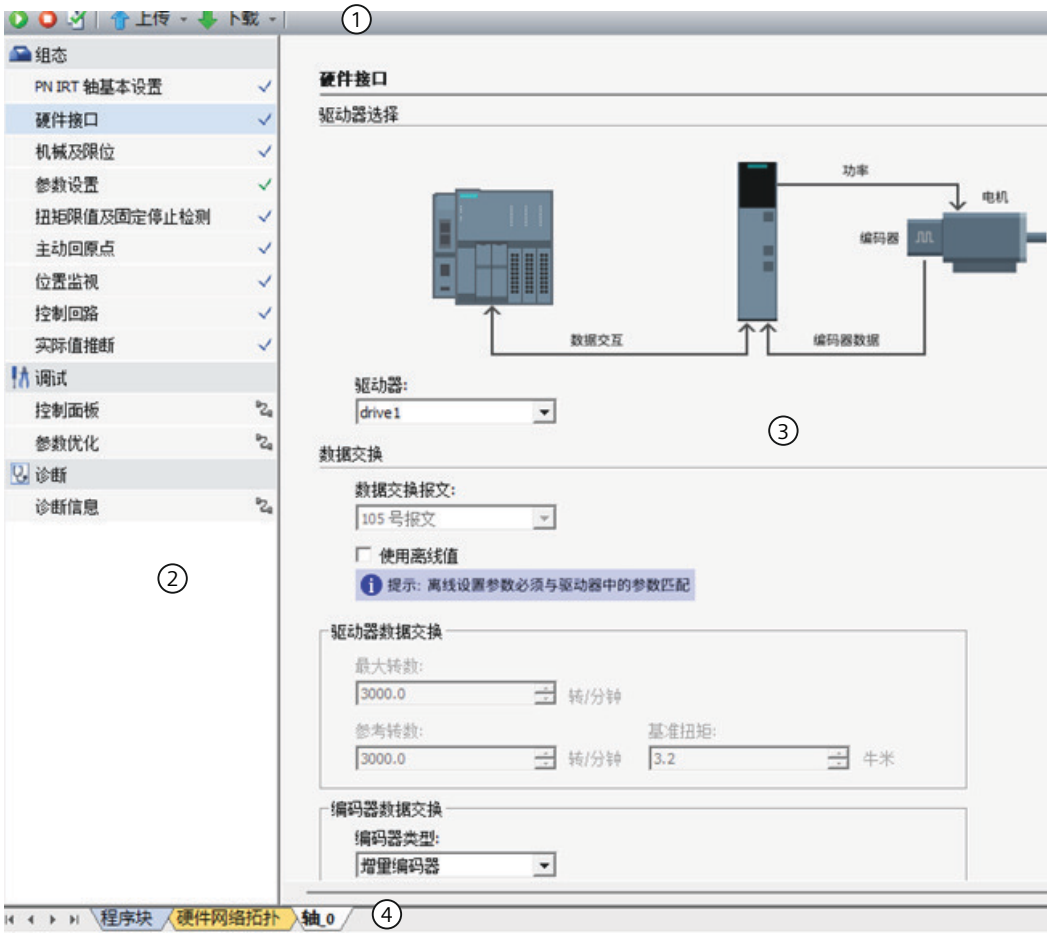
14.4.6 运动控制组件组态窗口

添加轴后，使用以下方法之一打开组态窗口：

- 在项目树中打开“运动控制”(Motion control) 文件夹 运动控制，方法是单击分支展开 图标或双击“运动控制”(motion control) 文件夹图标。双击要组态的组件，或者选择组件并按 Enter 键。
- 双击运动控制窗口底部的组件表。

轴

轴编辑器窗口包括以下组件：



① 工具栏：常用操作按钮

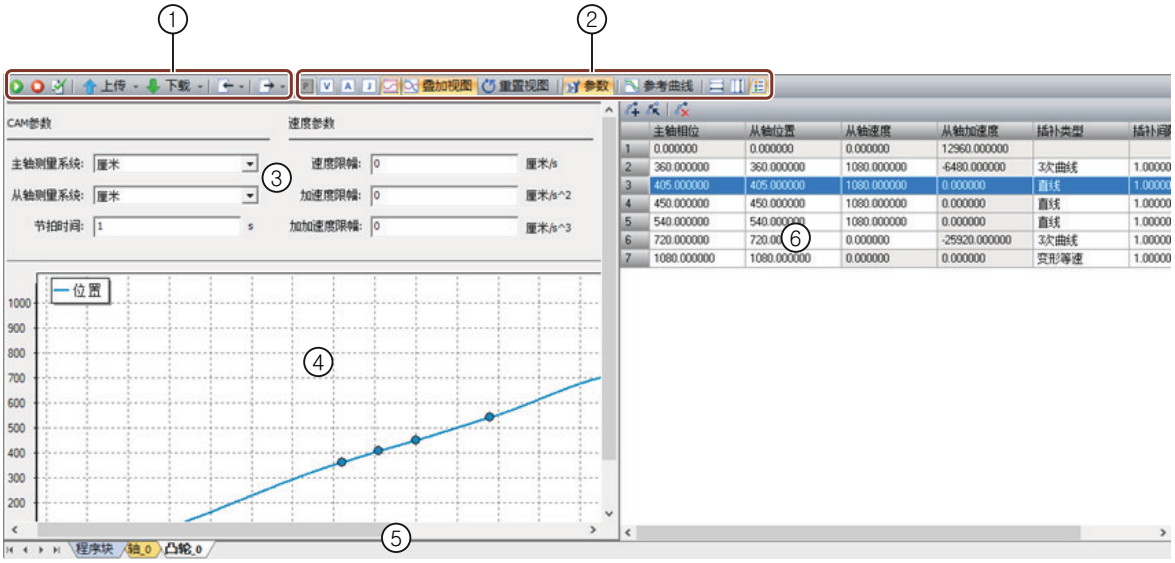
② 组态、调试和诊断的目录

③ 组态/调试/诊断页面

④ 组件选择器：可在不同运动控制组件的组态页面之间进行切换。

凸轮

凸轮编辑器窗口包括以下组件：



① 工具栏：常用操作按钮

② 曲线工具栏

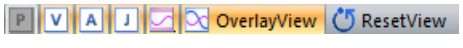
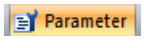
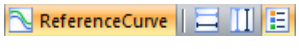
③ 参数组态面板

④ 位置/速度/加速度/加加速度曲线页面

⑤ 组件选择器：可在不同运动控制组件的组态页面之间进行切换。

⑥ 关键点组态表

凸轮工具栏

	将 CPU 工作模式更改为 RUN 或 STOP 编译程序
	上传和下载传送
	导入 (页 913)和导出 (页 913)凸轮数据/组态表
	图形控制工具 (页 912)
	显示或隐藏参数设置面板 (页 911)
	曲线控制工具 (页 912)

14.4.7 组态轴

在运动控制系统中，“轴”为受控对象。

要创建组件，按以下步骤操作：

- 双击项目树中的“运动控制”(Motion Control) 文件夹；右键单击组件文件夹并单击组件名称。

说明

目前仅固件版本为 3.2 的 ST32 CPU 才支持添加 PN IRT 轴。

当轴添加完成后，组态页面将显示其默认配置，并自动生成对应的系统运动控制变量表。

说明

HSC6 和 HSC7 的使用与编码器轴的使用不兼容。因此，如果启用了 HSC6 或 HSC7，只能使用一个编码器轴。如果同时启用了 HSC6 和 HSC7，则无法使用编码器轴，反之亦然。

说明

PN IRT 轴暂不支持配置为轴组。

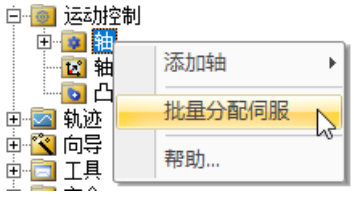
批量添加 IRT 轴

前提条件：

- 您已在项目中的硬件网络拓扑视图中成功添加了至少一个 IRT 设备。
- 当前项目配置中，可用的 IRT 轴数量未超出系统限制 (页 808)。

通过该功能，您可以一次性为已添加到硬件网络拓扑中的 IRT 设备自动创建 IRT 轴。

1. 双击打开项目树中的“运动控制”文件夹。
2. 右键单击“轴”文件夹，在弹出的右键菜单中，选择“批量添加伺服”。

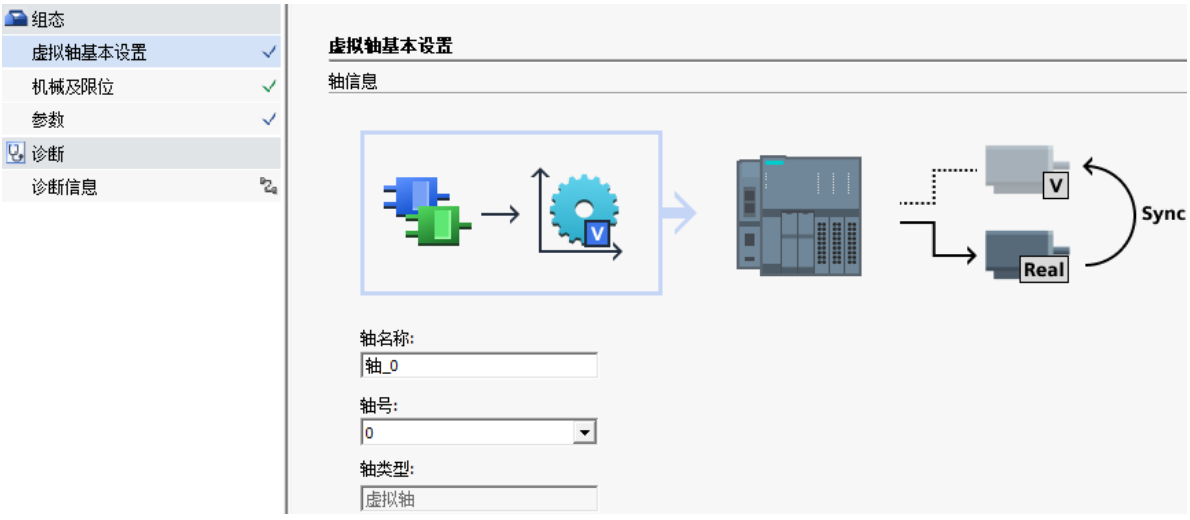


14.4.7.1 组态虚拟轴

虚拟轴基本设置

添加虚拟轴后，将显示如下图所示的“虚拟轴基本设置”(Virtual Axis Basic Setting) 对话框。

可为新添加的轴组态自定义名称。默认名称为“轴_x”，其中“x”代表按照数字顺序创建的轴。



说明

轴名称最多可包含 23 个字符，包括字母、数字、中文字符、下划线和扩展字符（ASCII 128 至 ASCII 255）。首字符必须为字母、汉字或下划线。不允许使用关键字作为变量名，也不允许使用以数字开头的变量名，包含非字母数字字符的变量名，或与现有指令名称或关键字相同的变量名。轴名称不可重复，也不能与现有变量名相同。

机械及限位

机械

在“组态”(CONFIGURATION) 面板中单击“机械及限位”(Mechanical and Limits) 时，将显示以下对话框：

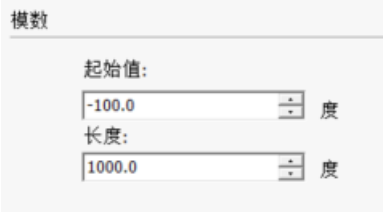
组态项	模式	说明
轴模式	线性	在此模式下，轴可沿直线移动，通常由线性导轨或轨道引导。
	旋转	在此模式下，轴可以围绕中心点或旋转轴旋转。
测量系统	脉冲 厘米、毫米、英寸或度	控制轴运动的所有速度均以“测量系统”每秒为单位表示。

模数

只有在 **“轴模式”(Axis mode)** 中选择了“旋转”(Rotary)，才能应用“模数”(Modulo) 组态。

“初始值”(Initial value) 定义了轴开始移动时的度数，而“长度”(Length) 定义了轴需要移动多少度。模块范围为左闭右开。例如，下例中模数的范围为 [-100, 900)。

- 如果通过 MC_Home 将位置设为 900.0，则执行该指令后，轴位置将显示为 -100.0。
  如果通过 MC_Home 将位置设为 1500.0，则执行该指令后，轴位置将显示为 500.0。



软件限位

当软件限位功能启用后，轴仅在超过设定的限位位置后才启动减速停，这意味着轴的最终停止位置将超出所设定的限位。

默认情况下禁用软件限位。禁用此输入时，也禁用所有其他相关参数。选中“启用软件限位”(Enabled Software limit) 复选框时，可以访问“软件负限位位置”(Software negative limit)、“软件正限位位置”(Software positive limit)、和“响应”(Response) 参数。

说明

软件限位功能仅支持线性轴。当轴类型被组态为旋转轴时，相关的软件限位组态选项将自动隐藏。

说明

如果 TO 中的 Homed=0, 软件限位将不会生效。

软件负/正限位位置

可定义轴停止移动的位置。

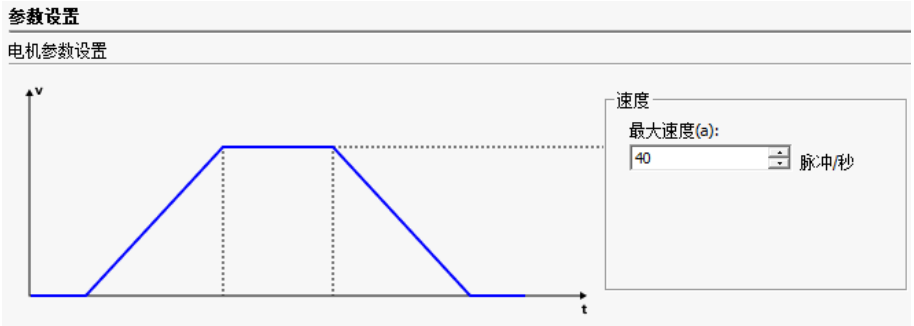
响应

“减速停止”(Decelerated Stop) 可通过在参数 (页 859)上介绍的“最大减速度”定义。

参数

电机参数设置

在“电机参数设置”(Motor Speeds) 对话框中，可以定义应用的最大速度。

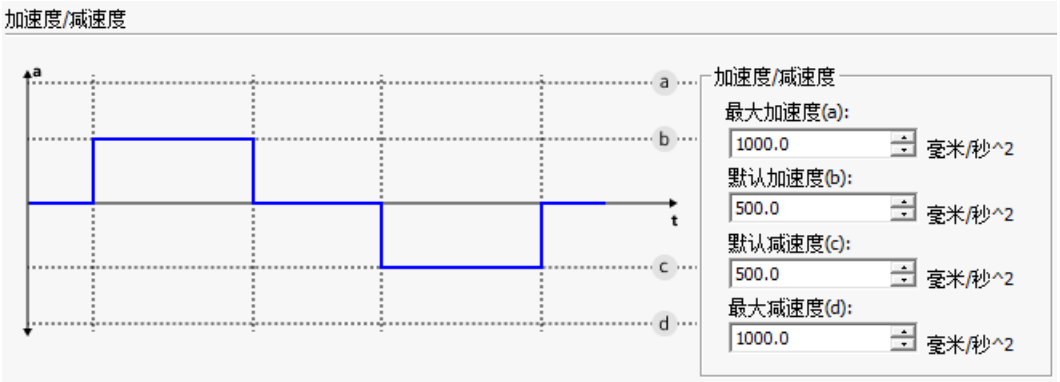


最大速度

- 在电机力扭矩能力范围内，输入应用中最佳操作速度的速度值。驱动负载所需的扭矩由摩擦力、惯性以及加速/减速时间决定。
- 速度的测量单位需在测量系统 (页 857)中组态；默认值为 1000.0 毫米/秒或 36000.0 度/秒。

加速度/减速度

在“加速度/减速度”(Acceleration/Deceleration) 对话框中，可为应用指定加速度和减速度。



- **最大加速度**：用于限制用户组态的运动指令的加速度最大值。
- **默认加速度**：运动控制指令 (页 952)中的加速度参数默认值。

- **默认减速度**：运动控制指令 (页 952)中的减速度参数默认值。
- **最大减速度**：用于限制用户组态的运动指令的减速度最大值。
  和
  – 对于虚拟轴，最大减速度为软件限位 (页 857)的“减速停止”(Decelerated Stop) 中使用的最大值。

加加速度

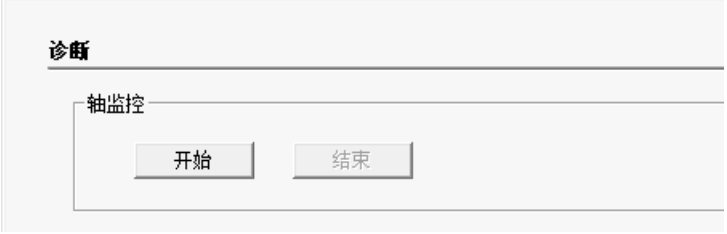
在“加加速度”(Jerk) 对话框中，可选择“启用加加速度”(Enable Jerk) 复选框以向加速度或减速度应用加加速度补偿。加加速度补偿提供较平稳的位置控制，方法是减少移动包络的加速和减速部分中的加加速度（速率变化）。减少加加速度可改善位置追踪性能。

诊断信息

命令

诊断功能可用于监视已组态轴的最重要状态和错误消息。

可通过单击“开始”(Start) 按钮和“结束”(Stop) 按钮来启用和禁用运动控制诊断过程。



轴状态

对话框中显示了轴状态。详细信息将显示在“错误代码 (页 1068)”(ErrorCode)、“位置”(Position)、“速度”(Velocity)。

轴状态	说明
错误	轴发生错误。有关详细信息，请参见运动控制指令中的 ErrorID 和 ErrorInfo 参数
禁用	轴已禁用且无法通过运动控制指令来控制
待机	轴为静止状态
离散运动	轴以离散运动状态运动
连续运动	轴以连续运动状态运动
同步运动	轴以同步运动状态运动
已归位	轴已返回至其原点位置
停止中	轴正在停止
错误停止	轴正经历错误状态，当前处于 ErrorStop 状态
软件正限位开关	已达到或超过软件正限位开关
软件负限位开关	已达到或超过软件负限位开关

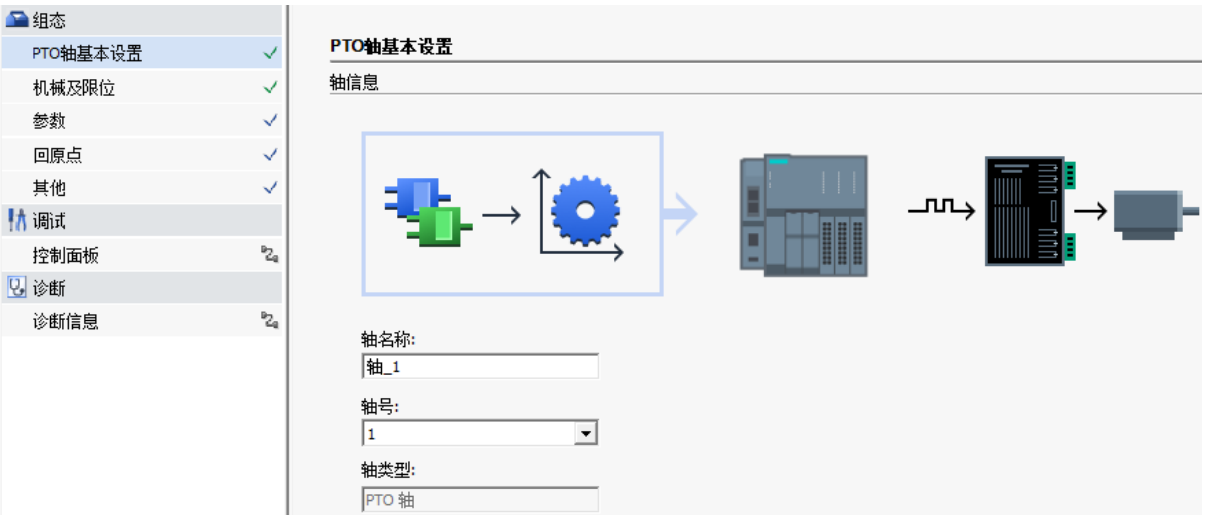
14.4.7.2 组态 PTO 轴

PTO 轴基本设置

轴信息

添加 PTO 轴后，将显示如下图所示的“PTO 轴基本设置”(PTO Axis Basic Settings) 对话框。

可为新添加的轴组态自定义名称。默认名称为“轴_x”，其中“x”代表按照数字顺序创建的轴。

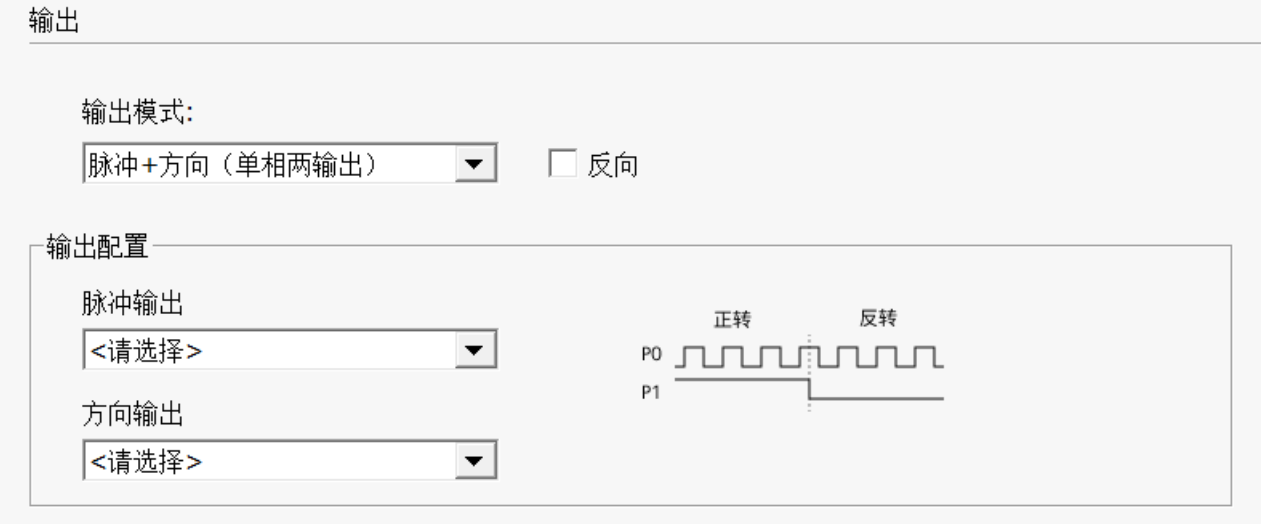


说明

- 轴名称最多可包含 23 个字符，包括字母、数字、中文字符、下划线和扩展字符（ASCII 128 至 ASCII 255）。首字符必须为字母、汉字或下划线。不允许使用关键字作为变量名，也不允许使用以数字开头的变量名，包含非字母数字字符的变量名，包含扩展字符集中字符的变量名，或与现有指令名称或关键字相同的变量名。轴名称不可重复，也不能与现有变量名相同。
- 可以在运动控制中创建最多 5 个 PTO 轴。

输出

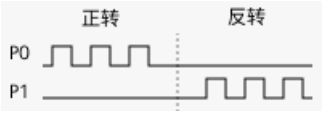
可组态步进电机/伺服驱动器的脉冲和方向输出接口。



步进电机/伺服驱动器的“输出模式”(Output mode) 接口有三个选项。

- 脉冲+方向（单相两输出）：如果选择“脉冲+方向（单相两输出）”(Pulse+Direction (Single Phase 2 output)) 选项，则一个输出（脉冲）控制脉冲，另一输出（方向）控制脉冲方向。如果脉冲处于正向，则方向为高（有效）。如果脉冲处于负向，则方向为低（无效）。脉冲+方向（单相两输出）如下图所示：

- 正向/反向（两相 2 输出）：如果选择两相（2 输出）选项，则一个输出 (CW) 脉冲用于正向，另一个输出脉冲用于负向。正向/反向（两相输出）如下图所示：



- A/B 相（双输出）：如果选择“A/B 相（双输出）”(A/B phase (2 output)) 选项，则两个输出均以指定速度产生脉冲，但相位相差 90 度。A/B 相（双输出）为 1X 组态，表示 1 个脉冲是每个输入的正跳变之间的时间量。这种情况下，方向由先变为高电平的输出跳变决定。针对正向，A 领先 B。针对负向，B 领先 A。下图显示 A/B 相（双输出）：

可通过选择或取消选择“反向”(Reverse) 复选框切换正向和负向。例如，如果电机接线方向错误，可选择“反向”(Reverse) 复选框，以避免对硬件进行重新接线。

说明

当一个 Q 点被配置并分配给脉冲序列输出（PTO）轴功能后，该 Q 点无法再被强制。

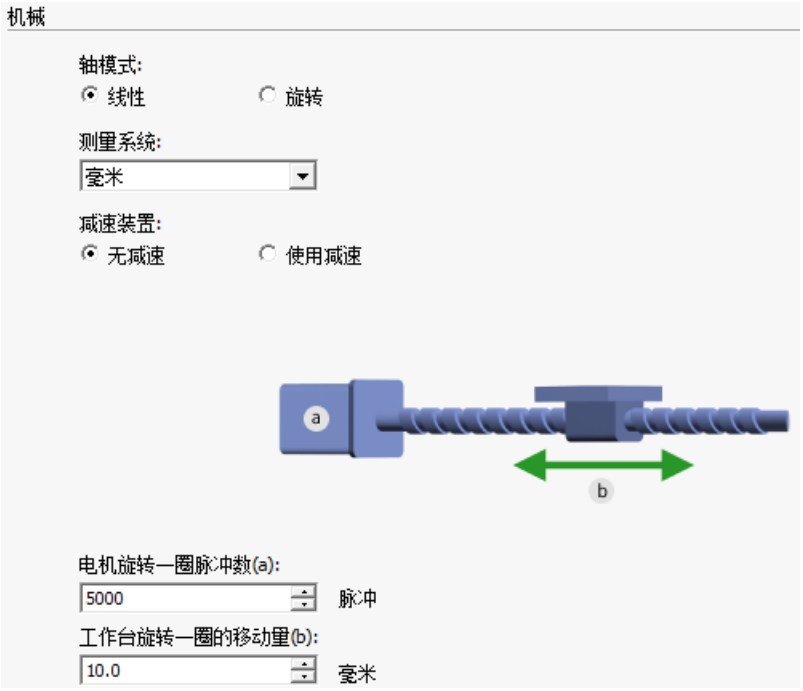
说明

“输出模式”(Output mode) 的默认设置为“脉冲+方向（单相双输出）”(Pulse+Direction (Single Phase 2 output))，且取消选择“反向”(Reverse) 复选框。

机械及限位

机械

在“组态”(CONFIGURATION) 面板中单击“机械及限位”(Mechanical and Limits) 时，将显示以下对话框：



组态项	模式	说明
轴模式	线性	在此模式下，轴可沿直线移动，通常由线性导轨或轨道引导。
	旋转	在此模式下，轴可以围绕中心点或旋转轴旋转。

组态项	模式	说明
测量系统	脉冲	• 控制轴运动的所有速度均以“每秒脉冲数”为单位表示。 • 控制轴运动的所有距离均以脉冲数为单位表示。 脉冲模式不支持使用“减速装置”(Reduction gear)。
	厘米、毫米、英寸或度	控制轴运动的所有速度均以“测量系统”每秒为单位表示。 必须组态的参数如下： • 电机/编码器旋转一圈脉冲数（请参见电机/编码器或驱动器的数据表）。 • 工作台旋转一圈的移动量 如果在“减速装置”(Reduction gear) 中选择“使用减速”(Use reduction)，则必须对“减速比分子”(Gear numerator) 和“减速比分母”(Gear denominator) 参数进行组态。
减速装置 [1]	无减速	电机/编码器或轴的输出速度与输入速度成正比，无需额外减速。本质上，这意味着运动以 1:1 的比例从输入传输到输出。
	使用减速	通过设置“减速比分子”(Gear numerator) 和“减速比分母”(Gear denominator)，可以确定减速装置的减速比，从而影响输出速度和扭矩的变化。

[1] **注：**“电机/编码器旋转一圈脉冲数”(Number of pulses per rotation of the motor/encoder) 参数的取值范围为 1 到 9,999,999 个脉冲。此参数的默认值为 5,000。

模数

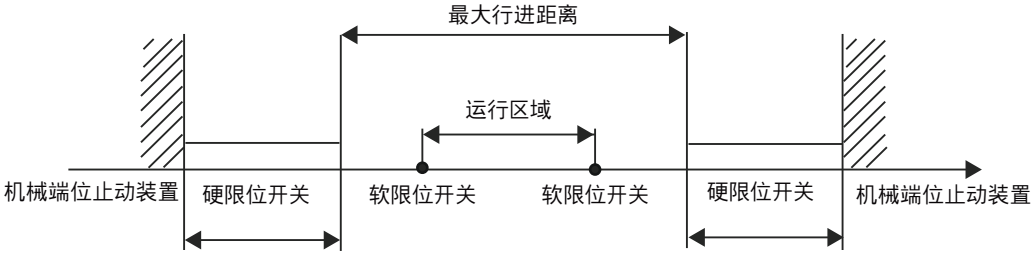
只有在 **“轴模式”(Axis mode)** 中选择了“旋转”(Rotary)，才能应用“模数”(Modulo) 组态。

“初始值”(Initial value) 定义了轴开始移动时的度数，而“长度”(Length) 定义了轴需要移动多少度。模块范围为左闭右开。例如，下例中模数的范围为 [-100, 900)。

- 如果通过 MC_Home 将位置设为 900.0，则执行该指令后，轴位置将显示为 -100.0。
  如果通过 MC_Home 将位置设为 1500.0，则执行该指令后，轴位置将显示为 500.0。

硬件限位

软件/硬件限位开关是用来保证轴能够在工作台的有效范围内运行，当轴由于故障超过的限位开关，不管轴碰到了是软限位还是硬限位，轴都是停止运行并报错。



可以定义将硬件负限位 (LMT-) 和硬件正限位 (LMT+) 输入分配给哪些引脚，以及限位输入的特性，包括“响应”(Response)和“触发电平”(Trigger Level)。

默认情况下禁用 LMT-/LMT+ 输入。禁用此输入时，也禁用所有其他相关参数。选中“启用硬件负限位开关”(Enabled Hardware negative limit switches) 或“启用硬件正限位开

关"(Enabled Hardware positive limit switches) 复选框时，可以访问"输入"(Input)、"响应"(Response) 和"触发电平"(Trigger Level) 参数。

组态项	说明
输入	可将 LMT-/LMT+ 分配给 CPU 上的一个输入。但此输入 (I) 不能再用于此轴或其它轴的运动控制功能。因此，如果运动控制组态中的另一个功能使用输入 (I)，则该输入 (I) 不会再此输入的选择下拉列表中显示。 **注**：如果已为一个运动控制功能选择一个输入 (I)，则该输入 (I) 不可用于 CPU 中的其他功能，包括 HSC 和沿中断。
触发电平	• 如果设为高电平，有电流流入输入时读取逻辑 1。 • 如果设为低电平，没有电流流入输入时读取逻辑 1。逻辑 1 电平解释为已达到限值。不管触发电平如何，有电流流入输入时，与这些输入对应的 LED 就会亮起。 默认设置是高电平。
响应	在此参数中，可以指定 LMT-/LMT+ 输入触发时应发生的情况。

警告

更改数字量输入通道的滤波时间存在的风险

如果数字量输入通道的滤波时间更改自以前的设置，则新的"0"电平输入值可能需要保持长达 12.8 ms 的累积时间，然后滤波器才会完全响应新输入。在此时间段内，持续时间小于 12.8 ms 的短"0"脉冲事件可能无法检测到或无法计数。

滤波时间的这种更改会引发意外的机械或过程操作，这可能会导致人员死亡、重伤和/或设备损坏。

对 CPU 循环上电以确保新的滤波时间立即生效。

说明

在系统块中，需编程应用中 LMT-/LMT+ 输入所需的滤波时间。只能在系统块中分配输入的滤波时间。

软件限位

默认情况下禁用软件限位。禁用此输入时，也禁用所有其他相关参数。选中“启用软件限位”(Enabled Software limit) 复选框时，可以访问“软件负限位位置”(Software negative limit)、“软件正限位位置”(Software positive limit)、和“响应”(Response) 参数。

说明

如果在上电后未执行 MC_Home，软件限位将不会生效。

软件负/正限位位置

可定义轴停止移动的位置。

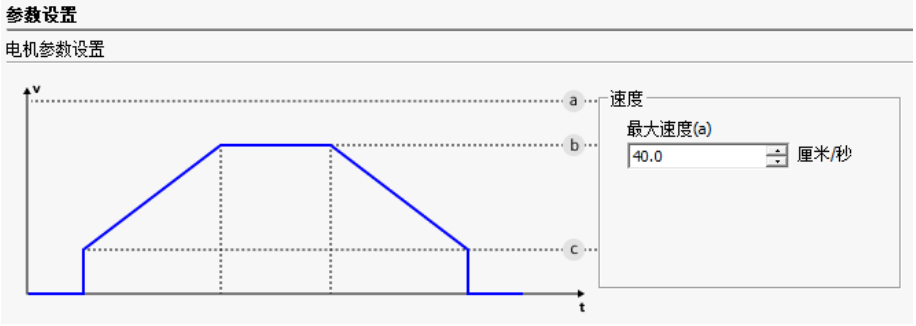
响应

“减速停止”(Decelerated Stop) 可通过在参数 (页 868)上介绍的“最大加速度/减速度”(Max acceleration/deceleration) 定义。

参数

电机参数设置

在“电机参数设置”(Motor Speeds) 对话框中，可以定义应用的最大速度。



最大速度

- 在电机力扭矩能力范围内，输入应用中最佳操作速度的速度值。驱动负载所需的扭矩由摩擦力、惯性以及加速/减速时间决定。
- 速度的测量单位需在测量系统中组态；速度不能低于 STEP 7 Micro/WIN SMART V3 运动控制的启停速度。

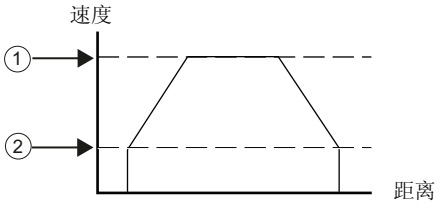
> ⚠ **警告**
>
> **设置合适的最大速度**
>
> 线性轴的默认最大速度为 400.0 毫米/秒，模态轴的默认最大速度为 400.0 度/秒。
> 请务必根据实际应用场景合理调整最大速度，避免设备损坏或人员伤害。

启停速度

- 无法组态“启停速度”(Start/Stop velocity)，根据工程单位生成固定速度。系统定义的最小启动速度为 200 Hz。

$$\text{CPU 脉冲} = \frac{\text{电机旋转一圈脉冲数(a)*减速比分子(c)}}{\text{工作台旋转一圈的移动量(b)*减速比分母(d)}} * \text{移动距离}$$

最大速度和启停速度定义运动轴的斜率特性，如下图所示：

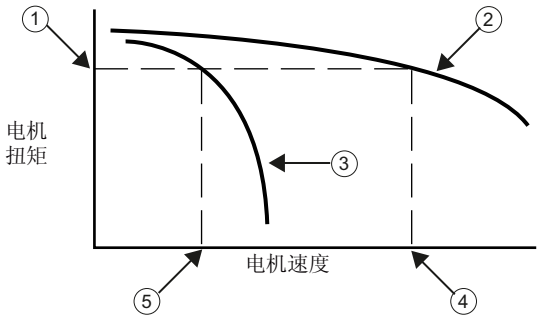


① 最大速度

② 启停速度

在电机数据单中，对于电机和给定负载，有不同的方式定义启停速度。

有关应用中可用的正确速度，请参见电机数据表。下图显示典型的电机扭矩/速度曲线：



① 驱动负载所需的扭矩

② 电机扭矩与速度特性

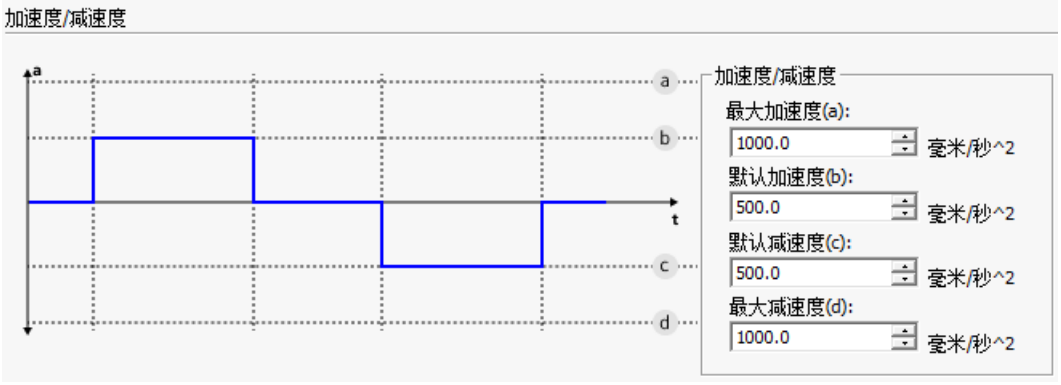
③ 启停速度/扭矩：此曲线随负载惯性增加向较低速度移动。

④ 电机可驱动负载的最大速度：已组态的最大速度不应超过此值。

⑤ 此负载的启停速度

加速度/减速度

在“加速度/减速度”(Acceleration/Deceleration) 对话框中，可为应用指定加速度和减速度。



- **最大加速度**：用于限制用户组态的运动指令的加速度最大值。
- **默认加速度**：运动控制指令 (页 952)中的加速度参数默认值。
- **默认减速度**：运动控制指令 (页 952)中的减速度参数默认值。
- **最大减速度**：用于限制用户组态的运动指令的减速度最大值。
  – 对于 PTO 轴，最大减速度为硬件限位/软件限位的“减速停止”(Decelerated Stop) 中使用的最大值以及“其他 (页 874)”(Others) 组态中 STP 功能的“减速停止”(Decelerated Stop) 中使用的最大值。
  电机加速度和减速度设置要经过测试来确定。开始时，应输入一个较大的值。然后逐渐调整输入值，直到电机开始失速，这样可以优化应用中的这些设置。

加加速度

在“加加速度”(Jerk) 对话框中，可选择“启用加加速度”(Enable Jerk) 复选框以向加速度或减速度应用加加速度补偿。加加速度补偿提供较平稳的位置控制，方法是减少移动包络的加速和减速部分中的加加速度（速率变化）。减少加加速度可改善位置追踪性能。

回原点

RPS 信号

启用 RPS

在“RPS 信号”(RPS Signal) 对话框中，可以定义参考点查找输入分配给哪个引脚以及 RPS 输入的特性，包括“触发电平”(Trigger Level)。

RPS 输入定义执行参考点查找命令时的原点位置或参考点。

默认情况下禁用 RPS 输入。禁用此输入时，也禁用该页面上的所有其他参数。选中“启用”(Enabled) 复选框时，可以访问“输入”(Input) 和“触发电平”(Trigger Level) 参数。

输入参数

可将 RPS 分配给 CPU 上的一个输入。可分配 CPU 上从 I0.0 到 I1.7 之间的任何一个输入，但此输入 (I) 不能再用于此轴或其他轴的运动控制功能。因此，如果运动控制组态中的另一个功能使用输入 (I)，则该输入 (I) 不会再此输入的选择下拉列表中显示。

因此，如果运动控制向导中的另一功能使用一个输入，则该输入不显示在此输入的选择下拉列表中。该输入启用后默认为列表中的下一可用输入。

电平参数

在此参数中，可指定 RPS 输入的触发电平。如果设为高电平，有电流流入输入时读取逻辑 1。如果设为低电平，没有电流流入输入时读取逻辑 1。逻辑 1 电平解释为已达到参考点或原点位置。不管触发电平如何，有电流流入输入时，与这些输入对应的 LED 就会亮起。默认设置是高电平。

启用 ZP

可通过选择“启用 ZP”(Enable ZP) 复选框定义 ZP 输入分配给哪个输入引脚。零脉冲 (ZP) 输入有助于建立参考点查找 (RPS) 命令中所用参考点或原点位置。通常，每转一圈，电机驱动器/放大器就会产生一个 ZP 脉冲。

可分配 CPU 上的 ZP 输入。可分配 CPU 上的 I0.0 到 I1.7，但此输入 (I) 不能再用于此轴或其他轴的运动控制功能。因此，如果运动控制组态中的另一个功能使用输入 (I)，则该输入 (I) 不会再此输入的选择下拉列表中显示。

回原点

图 14-1　PTO 轴回原点

回零标志

选择负向或正向作为原始方向。

逼近原点方向/回原点起始方向

通过选择方向，您可以确定在主动回原点过程中用于寻找回原点开关的逼近方向以及回原点的方向。回原点方向指定了在执行回原点操作时，轴向回原点开关配置端移动的方向。

触碰硬件限位时

如果选择此“触碰硬件限位后自动回转”(Auto-return when hardware limit is reached)，则轴能够反向旋转至归零点。

速度

- 逼近原点速度：定义逼近原点的速度
- 回原点起始速度：定义回原点的起始速度

说明

输入值的限值为：启停速度 ≤ 逼近原点速度 ≤ 回原点起始速度 ≤ 最大速度

原点偏移

如果期望的原点位置与回原点开关的位置有偏移，则可在该字段中指定原点位置偏移。如果值不等于 0，轴回到回原点开关位置后将执行以下动作：

1. 以回原点速度使轴移动原点位置偏移值。
2. 到达“原点偏移”(home position offset) 后，原点处于“MC_Home”运动控制指令“位置”(Position) 参数中指定的原点。

其他

紧急停止

启用急停 STP

在“急停”(Emergency Stop) 对话框中，可以通过选择“启用 STP”(Enable STP) 复选框来定义停止输入的引脚以及停止输入的特性。

STP 输入有以下功能：

- 使任何激活的运动控制命令减速至启动-停止速度并在减速后立即停止脉冲
- 当“触发”(Trigger) 设置为**“电平触发”(Level)** 时防止启动新的运动命令
- 可作为来自步进/伺服驱动器的就绪信号
- 如果激活，通过将“触发器”(Trigger) 设置为**“边沿”(Edge)**，停止前一个运动后，可以启动新的运动。

默认情况下不选择“启用 STP”(Enable STP) 复选框，且 STP 对话框上的所有其它参数都会禁用。选中“启用急停 STP”(Enable STP) 复选框时，可以访问“输入”(Input)、“响应”(Response)、“触发”(Trigger) 和“电平触发”(Level) 参数。

输入

可将 STP 分配给 CPU 上的一个输入。可分配 CPU 上从 I0.0 到 I1.7 之间的任何一个输入，但此输入 (I) 不能再用于此轴或其他轴的运动控制功能。因此，如果运动控制组态中的另一个功能使用输入 (I)，则该输入 (I) 不会再此输入的选择下拉列表中显示。

说明

如果为一个运动控制功能选择一个输入，则该输入不可用于 CPU 中的其他功能，包括 HSC 和沿中断。

“响应”(Response) 参数

在此参数中，可以指定如果 STP 输入激活则应发生什么。默认设置是“减速停止”(Decelerated Stop)。可从以下两个选项中选择：

- “减速停止”(Decelerated Stop)：使任何激活的运动控制命令减速至启停速度并停止脉冲
- “立即停止”(Immediate Stop)：使任何激活的运动控制命令立即停止脉冲

“触发器”(Trigger) 参数

在此参数中，可指定 STP 输入的触发器。默认设置为“电平”(Level)。可从以下两个选项中选择：

- 电平：如果选择电平触发器，每当输入激活时运动都会停止。电平触发器设置阻止启动任何新的运动，直到 STP 输入变为未激活状态。
- “边沿”(Edge)：如果选择边沿触发器，仅当输入从非激活变为激活时运动才停止。边沿触发设置允许前一个运动停止后启动新的运动，即使 STP 输入仍处于非激活状态。

当运动停止时，输入变为非激活状态，轴可以开始另一运动。

探针

可通过选择“启用探针 0”(Enable probe 0) 和“启用探针 1”(Enable probe 1) 来启用探针控制。

输入

选择接触式探针的输入。可分配 CPU 上从 I0.0 到 I1.7 之间的任何一个输入。
在使用 MC_TouchProbe 指令前，必须组态“探针”。

诊断信息

命令

诊断功能可用于监视已组态轴的最重要状态和错误消息。

可通过单击“开始”(Start) 按钮和“结束”(Stop) 按钮来启用和禁用运动控制诊断过程。

轴状态

对话框中显示了轴状态。详细信息将显示在“错误代码 (页 1068)”(ErrorCode)、“位置”(Position)、“速度”(Velocity)。

轴状态	说明
错误	轴发生错误。有关详细信息，请参见运动控制指令中的 ErrorID 和 ErrorInfo 参数
禁用	轴已禁用且无法通过运动控制进行控制
待机	轴为静止状态
离散运动	轴以离散状态运动
连续运动	轴以连续状态运动
同步运动	轴以同步状态运动
已归位	轴已返回至其原点位置
停止中	轴正在停止
错误停止	轴正经历错误状态，当前处于 ErrorStop 状态
件正限位开关	已达到或超过硬件正限位开关。
件负限位开关	已达到或超过硬件负限位开关。
软件正限位开关	已达到或超过软件正限位开关。
软件负限位开关	已达到或超过软件负限位开关。
参考开关	轴回原点时，RPS 信号已被触发。
急停开关	STP 信号已被触发

14.4.7.3 组态编码器轴

编码器轴基本设置

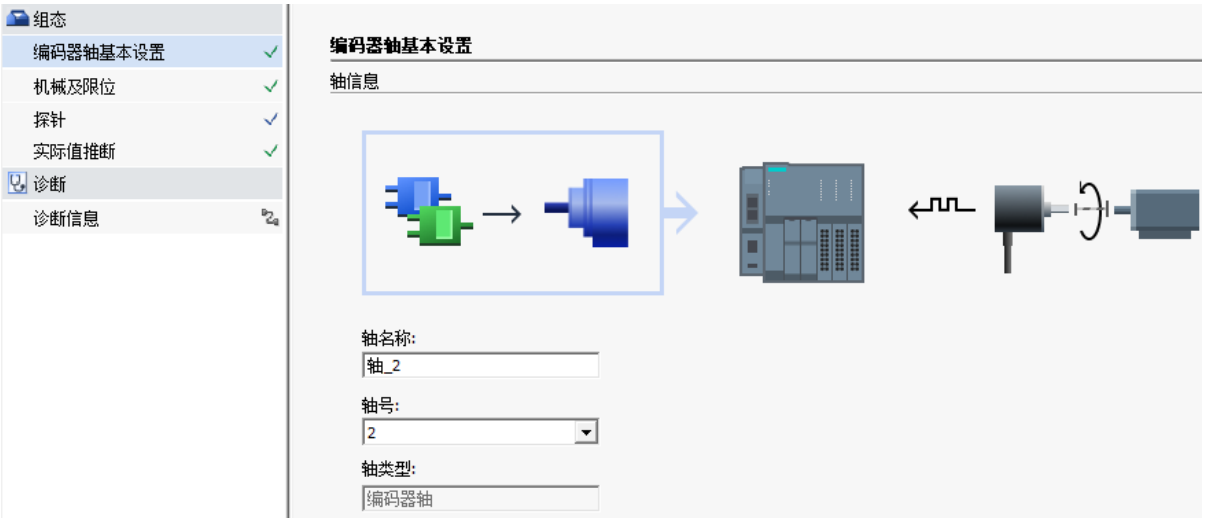
轴信息

说明

使用编码器轴时，确保系统块中组态的数字量输入滤波器对于所选数字量输入 ≤ 1.6μs。

添加编码器轴后，将显示如下图所示的“编码器轴基本设置”(Encoder Axis Basic Settings) 对话框。

可为新添加的轴组态自定义名称。默认名称为“轴_x”，其中“x”代表按照数字顺序创建的轴。



说明

- 轴名称最多可包含 23 个字符，包括字母、数字、中文字符、下划线和扩展字符（ASCII 128 至 ASCII 255）。首字符必须为字母、汉字或下划线。不允许使用关键字作为变量名，也不允许使用以数字开头的变量名，包含非字母数字字符的变量名，包含扩展字符集中字符的变量名，或与现有指令名称或关键字相同的变量名。轴名称不可重复，也不能与现有变量名相同。
- 可以在运动控制中创建最多 2 个编码器轴。
  注：HSC6 和 HSC7 的使用与编码器轴的使用不兼容。因此，如果启用了 HSC6 或 HSC7，只能使用一个编码器轴。如果同时启用了 HSC6 和 HSC7，则无法使用编码器轴，反之亦然。

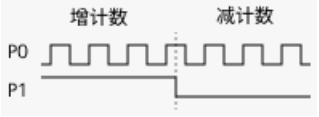
输入

可组态步进电机/伺服驱动器的脉冲和方向输入接口。

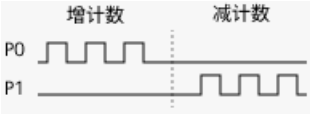
步进电机/伺服驱动器的“输入模式”(Input mode) 接口有四个选项。

- 脉冲 (A)：如果选择“脉冲 (A)”(Pulse(A)) 选项，则输入（脉冲）接受脉冲。在此模式下，CPU 仅接受正向运动命令。下图显示脉冲 (A) 单相（假设极性为正）：

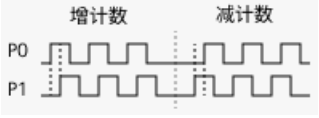
- 脉冲 (A) + 方向 (B)：如果选择“脉冲 (A) + 方向 (B)”(Pulse(A) + Direction(B)) 选项，则一个输入（脉冲）接受脉冲，另一输入（方向）控制脉冲方向。如果脉冲处于正向，则方向为高（有效）。如果脉冲处于负向，则方向为低（无效）。脉冲 (A) + 方向 (B) 如下图所示（假定已取消选择“反向”(Reverse) 复选框）：



- 递增计数 (A) + 递减计数 (B)：如果选择“递增计数 (A) + 递减计数 (B)”(Count up(A) + Count down(B)) 选项，则一个输入为正方向脉冲，另一个输出为负方向脉冲。递增计数 (A) + 递减计数 (B) 如下图所示（假定已取消选择“反向”(Reverse) 复选框）：



- 增量编码器（AB 相移）：如果选择“增量编码器（AB 相移）”(Incremental encoder (AB phase shifting)) 选项，则两个输入均以指定速度产生脉冲，但相位相差 90 度。增量编码器（AB 相移）为 1X 频率，表示 1 个脉冲是每个输入的正跳变之间的时间量。针对正向，A 领先 B。针对负向，B 领先 A。增量编码器（AB 相移）如下图所示（假定已取消选择“反向”(Reverse) 复选框）：



可通过选择或取消选择“反向”(Reverse) 复选框切换正向和负向。例如，如果电机接线方向错误，可选择“反向”(Reverse) 复选框，以避免对硬件进行重新接线。

可以为实际速度滤波器设置一个 0 至 2000 ms 的时间常量，以确保输出速度更平滑。

说明

“输入模式”(Input mode) 的默认设置为“脉冲 (A)”(Pulse(A))。

说明

将编码器轴用作 MC_CamIn 或 MC_GearIn 的主轴时，编码器轴的速度不受“编码器轴基本设置”(Encoder Axis Basic Settings) 页面中“实际速度滤波器的时间常数”(Time Constant of actual velocity filter) 参数的影响。

机械及限位

机械

在“组态”(CONFIGURATION) 面板中单击“机械及限位”(Mechanical and Limits) 时，将显示以下对话框：

组态项	模式	说明
轴模式	线性	在此模式下，轴可沿直线移动，通常由线性导轨或轨道引导。
	旋转	在此模式下，轴可以围绕中心点或旋转轴旋转。

组态项	模式	说明
测量系统	脉冲	• 控制轴运动的所有速度均以“每秒脉冲数”为单位表示。 • 控制轴运动的所有距离均以脉冲数为单位表示。 脉冲模式不支持使用“减速装置”(Reduction gear)。
	厘米、毫米、英寸或度	控制轴运动的所有速度均以“测量系统”每秒为单位表示。 必须组态的参数如下： • 电机/编码器旋转一圈脉冲数（请参见电机/编码器或驱动器的数据表）。 • 工作台旋转一圈的移动量 如果在“减速装置”(Reduction gear) 中选择“使用减速”(Use reduction)，则必须对“减速比分子”(Gear numerator) 和“减速比分母”(Gear denominator) 参数进行组态。
减速装置 [1]	无减速	电机/编码器或轴的输出速度与输入速度成正比，无需额外减速。本质上，这意味着运动以 1:1 的比例从输入传输到输出。
	使用减速	通过设置“减速比分子”(Gear numerator) 和“减速比分母”(Gear denominator)，可以确定减速装置的减速比，从而影响输出速度和扭矩的变化。

[1] **注：**“电机/编码器旋转一圈脉冲数”(Number of pulses per rotation of the motor/encoder) 参数的取值范围为 1 到 9,999,999 个脉冲。此参数的默认值为 5,000。

模数

只有在 **“轴模式”(Axis mode)** 中选择了“旋转”(Rotary)，才能应用“模数”(Modulo) 组态。

“初始值”(Initial value) 定义了轴开始移动时的度数，而“长度”(Length) 定义了轴需要移动多少度。模块范围为左闭右开。例如，下例中模数的范围为 [-100, 900)。

- 如果通过 MC_Home 将位置设为 900.0，则执行该指令后，轴位置将显示为 -100.0。如果通过 MC_Home 将位置设为 1500.0，则执行该指令后，轴位置将显示为 500.0。

探针

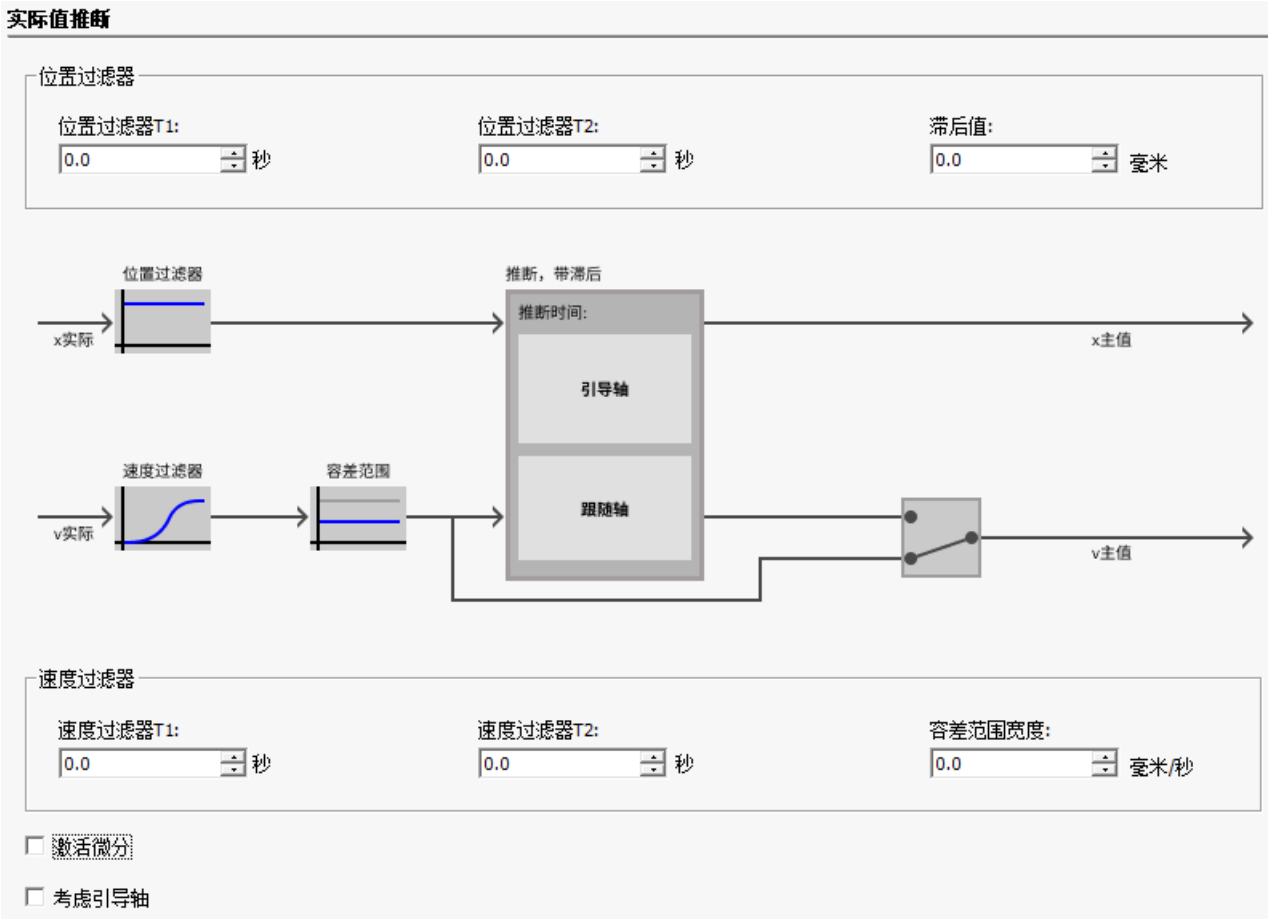
可通过选择“启用探针 0”(Enable probe 0) 和“启用探针 1”(Enable probe 1) 来启用探针控制。

输入

选择接触式探针的输入。可分配 CPU 上从 I0.0 到 I1.7 之间的任何一个输入。在使用 MC_TouchProbe 指令前，必须组态“探针”。

实际值推断

在“实际值推断”中为用于同步操作耦合的实际值组态外插补。此处设置的值仅在当前轴的实际值用作主值时生效。



位置过滤器 T1 和 T2

T1 和 T2	输入 PT2 滤波器的时间常数对位置进行滤波。
滞后值	输入应用到位置的外推实际值的迟滞功能值。指定值采用组态的长度单位。

速度过滤器

速度过滤器 T1 和 T2	输入用于平滑实际速度的 PT2 滤波器的时间常数，以及平滑后的实际速度的容差带宽。
容差带宽	为优化容差带宽的应用，请输入与容差区间相同的带宽作为噪声信号带宽。
激活微分	选中此复选框后，主值速度将通过对外推主值位置进行微分而获得。 取消选中此复选框后，将应用滤波后的实际速度。
考虑引导轴	选中此复选框后，引导轴相关的外推时间将包含在有效外推时间的计算中。 取消选中此复选框后，引导轴相关的外推时间将不包含在有效外推时间的计算中。

诊断信息

命令

诊断功能可用于监视已组态轴的最重要状态和错误消息。

可通过单击“开始”(Start) 按钮和“结束”(Stop) 按钮来启用和禁用运动控制诊断过程。

轴状态

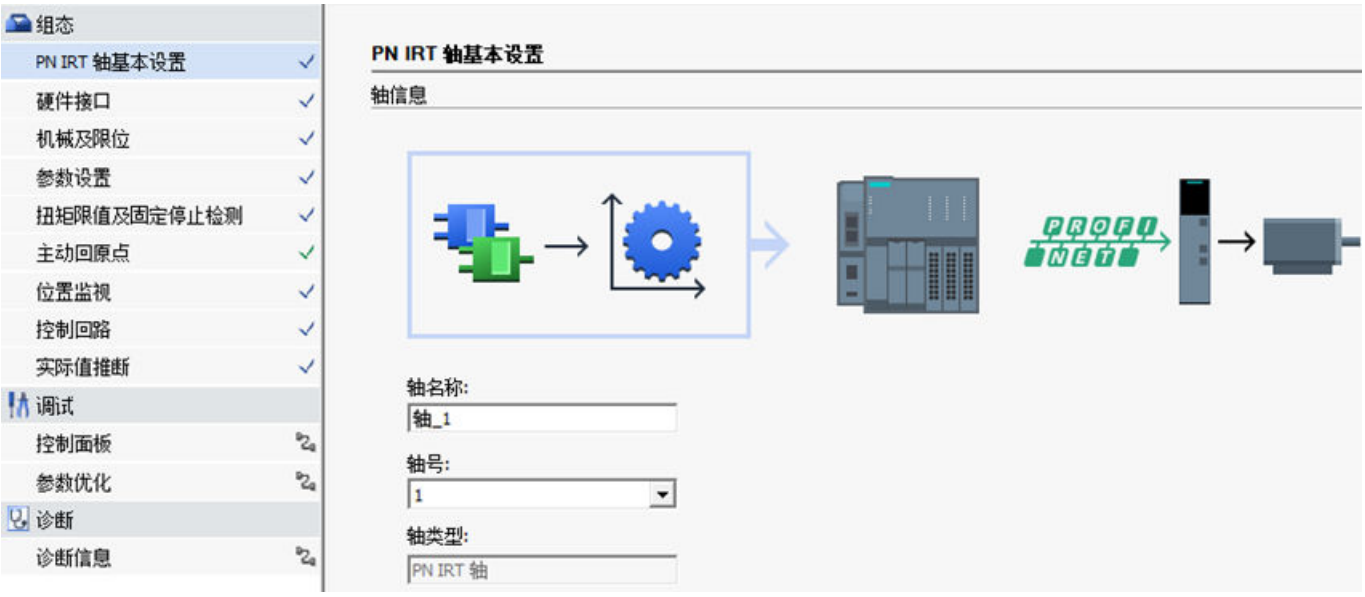
对话框中显示了轴状态。详细信息将显示在“错误代码 (页 1068)”(ErrorCode)、“位置”(Position)、“速度”(Velocity)。

轴状态	说明
启用编码器轴	轴已启用
错误	轴发生错误。有关详细信息，请参见运动控制指令中的 ErrorID 和 ErrorInfo 参数
已归位	如果系统 Homed，则表示其移到预先确定的已知位置，通常是运动范围的起始位置或结束位置。该过程允许系统在后续操作期间准确地跟踪其位置

14.4.7.4 组态 PN IRT 轴

PN IRT 轴基本设置

添加 PN IRT 轴后，将显示如下图所示的 " PN IRT 轴基本设置"(PN IRT Axis Basic Setting) 对话框。



在此页面，您可以切换轴的 ID 和修改轴的名称，但无法修改轴的类型。

对于新添加的轴，您可以在当前页面、页面底部的选项卡或项目树中自定义其名称。默认名称为 "轴_x"，其中"x"代表按照数字顺序创建的轴。

说明

轴名称最多可包含 23 个字符，包括字母、数字、中文字符、下划线和扩展字符（ASCII 128 至 ASCII 255）。首字符必须为字母、汉字或下划线。不允许使用关键字作为变量名，也不允许使用以数字开头的变量名，包含非字母数字字符的变量名，或与现有指令名称或关键字相同的变量名。轴名称不可重复，也不能与现有变量名相同。

硬件接口

在 "硬件接口"(Hardware Interface) 组态窗口中，选择和组态自己需要使用的驱动器。

前提

- 您已在硬件网络拓扑中完成了 IRT 网络的组态，并添加了所有需要的驱动器。

如果硬件目录中没有该驱动器，则必须在“选项”(Options) 菜单中将该驱动器安装成设备描述文件 (GSD)。

说明

每个驱动器设备只能分配给一个轴。例如，如果驱动器设备_1 已分配给轴 _1，则该设备将不会出现在其他轴的驱动器选择下拉列表中。

驱动器选择

在下拉列表中，选择您在硬件网络拓扑中在 IRT 网络层添加的驱动器。

数据交换

您在硬件网络拓扑中配置的数据交换报文和附加报文，可在“数据交换”部分中查看。

使用离线值

如勾选“使用离线值”复选框，驱动器离线值和编码器离线值将同时启用。

说明

- 在线值优先： 如果您输入的离线值与设备实际读取到的在线值存在差异，系统将优先采用当前设备的在线值。
- 未能读取到设备在线值且您已选择在线模式： 系统将直接报错。
- 未能读取到设备在线值且您已选择离线模式： 系统将采用您输入的离线值。

- 驱动器
  - **最大**转速
    在该字段中，组态驱动装置的最大速度。
  - **参考**转速
    在该字段中，根据制造商的技术数据，组态驱动装置的基准速度。
  - 基准扭矩
    在该字段中，对照驱动装置组态来组态其基准扭矩。
    力/扭矩减速需要参考基准扭矩，这可以通过报文 10x 提供支持。
- 编码器：对于闭环位置控制，定位轴需要以编码器位置形式表示的实际位置值。编码器的位置通过 PROFIdrive 报文发送至控制器。
  在下拉列表中，选择编码器的编码器类型。可以使用以下编码器类型：
  - 增量编码器
  - 单圈绝对值编码器
  - 多圈绝对值编码器

	输入范围	默认值	单位
最大转速	1.0～10000.0	3000.0	转/分钟
参考转速	1.0～10000.0	3000.0	转/分钟
基准扭矩	0.0 ~ 1000000000.0（1E09）	3.2	N·m
编码器每转增量	1 ~ 9999999	2048	
转速	1 ~ 9999999	1	
Gx_XIST1	0 ~ 31	11	
Gx_XIST2	0 ~ 31	9	

机械及限位

机械

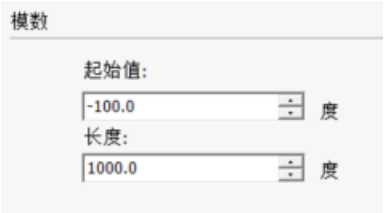
在“组态”(CONFIGURATION) 面板中单击“机械及限位”(Mechanical and Limits) 时，将显示以下对话框：

组态项	模式	说明
轴模式	线性	在此模式下，轴可沿直线移动，通常由线性导轨或轨道引导。
	旋转	在此模式下，轴可以围绕中心点或旋转轴旋转。
单位	度、厘米、毫米、英寸或度	控制轴运动的所有速度均以“测量系统”每秒为单位表示。默认值为毫米。
减速装置	无减速	电机/编码器或轴的输出速度与输入速度成正比，无需额外减速。本质上，这意味着运动以 1:1 的比例从输入传输到输出。
	使用减速	通过设置“驱动电机转速”和“负载转数”，从而影响输出速度和扭矩的变化。

模数

只有在 **“轴模式”(Axis mode)** 中选择了“旋转”(Rotary)，才能应用“模数”(Modulo) 组态。

“初始值”(Initial value) 定义了轴的初始位置，而“长度”(Length) 定义了轴需要移动多少度。模块范围为左闭右开。



电机和编码器反向

如果要将电机和编码器的旋转方向改为反向旋转，则需选中“电机和编码器反向”复选框。

位置参数

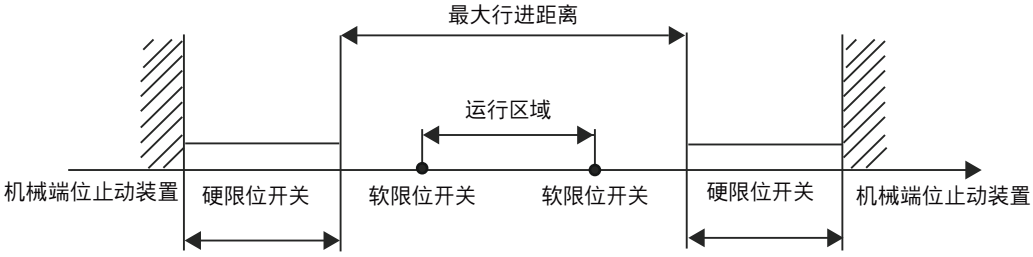
	线性轴	旋转轴
单位	毫米	度
位置参数	在“丝杠螺距”组态字段中，组态负载随丝杠旋转一圈而移动的距离。	在“旋转一圈的长度”组态字段中，组态负载旋转一圈的度数。

减速装置

勾选“使用减速”后，可以组态“驱动电机转速”和“负载转速”。

硬件限位

软件/硬件限位开关是用来保证轴能够在工作台的有效范围内运行，当轴由于故障超过的限位开关，不管轴碰到了是软限位还是硬限位，轴都是停止运行并报错。



可以定义将硬件负限位 (LMT-) 和硬件正限位 (LMT+) 输入分配给哪些引脚，以及限位输入的特性，包括“响应”(Response)和“触发电平”(Trigger Level)。

默认情况下禁用 LMT-/LMT+ 输入。禁用此输入时，也禁用所有其他相关参数。选中“启用硬件负限位开关”(Enabled Hardware negative limit switches) 或“启用硬件正限位开关”(Enabled Hardware positive limit switches) 复选框时，可以访问“输入”(Input)、“响应”(Response) 和“触发电平”(Trigger Level) 参数。

组态项	说明
输入	可将 LMT-/LMT+ 分配给 CPU 上的一个输入。但此输入 (I) 不能再用于此轴或其它轴的运动控制功能。因此，如果运动控制组态中的另一个功能使用输入 (I)，则该输入 (I) 不会再此输入的选择下拉列表中显示。 **注：**如果已为一个运动控制功能选择一个输入 (I)，则该输入 (I) 不可用于 CPU 中的其他功能，包括 HSC 和沿中断。
触发电平	• 如果设为高电平，有电流流入输入时读取逻辑 1。 • 如果设为低电平，没有电流流入输入时读取逻辑 1。逻辑 1 电平解释为已达到限值。不管触发电平如何，有电流流入输入时，与这些输入对应的 LED 就会亮起。 默认设置是高电平。
响应	在此参数中，可以指定 LMT-/LMT+ 输入触发时应发生的情况。

⚠ 警告

更改数字量输入通道的滤波时间存在的风险

如果数字量输入通道的滤波时间更改自以前的设置，则新的“0”电平输入值可能需要保持长达 12.8 ms 的累积时间，然后滤波器才会完全响应新输入。在此时间段内，持续时间小于 12.8 ms 的短“0”脉冲事件可能无法检测到或无法计数。

滤波时间的这种更改会引发意外的机械或过程操作，这可能会导致人员死亡、重伤和/或设备损坏。

对 CPU 循环上电以确保新的滤波时间立即生效。

说明

在系统块中，需编程应用中 LMT-/LMT+ 输入所需的滤波时间。只能在系统块中分配输入的滤波时间。

软件限位

默认情况下禁用软件限位。禁用此输入时，也禁用所有其他相关参数。选中“启用软件限位”(Enabled Software limit) 复选框时，可以访问“软件负限位位置”(Software negative limit)、“软件正限位位置”(Software positive limit)、和“响应”(Response) 参数。

说明

如果 TO 中的 Homed=0, 软件限位将不会生效。

说明

当轴组态的最大加速度远大于轴的最大减速度时，软限位减速停时的减速度可能突破最大减速度限制。

说明

对于 IRT 轴，当其工作在非位置控制环模式下时（例如，在使用 MC_MoveJog、MC_MoveVelocity 或 TorqueControl (Mode 1) 等指令时），软件限位的参考位置将从指令给定位置自动切换为轴的实际位置。

软件负/正限位位置

可定义轴停止移动的位置。

响应

“减速停止”(Decelerated Stop) 可通过在参数设置 (页 890)上介绍的“最大减速度”定义。

参数设置

电机参数设置

在“电机参数设置”(Motor Speeds) 对话框中，可以定义应用的最大速度。

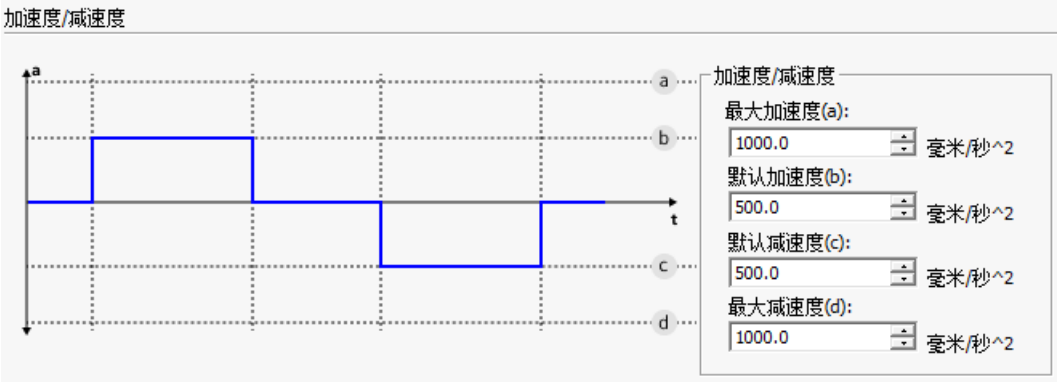
最大速度

- 在电机力扭矩能力范围内，输入应用中最佳操作速度的速度值。驱动负载所需的扭矩由摩擦力、惯性以及加速/减速时间决定。
- 速度的测量单位需在单位 (页 886)中组态；

> ⚠ **警告**
>
> **设置合适的最大速度**
>
> 线性轴的默认最大速度为 1000.0 毫米/秒，模态轴的默认最大速度为 36000.0 度/秒。请务必根据实际应用场景合理调整最大速度，避免设备损坏或人员伤害。

加速度/减速度

在“加速度/减速度”(Acceleration/Deceleration) 对话框中，可为应用指定加速度和减速度。



- **最大加速度**：用于限制用户组态的运动指令的加速度最大值。
- **默认加速度**：运动控制指令 (页 952)中的加速度参数默认值。
- **默认减速度**：运动控制指令 (页 952)中的减速度参数默认值。
- **最大减速度**：用于限制用户组态的运动指令的减速度最大值。
  对于 PN IRT 轴硬件限位/软件限位使用的减速度为轴配置的最大减速度。
  电机加速度和减速度设置要经过测试来确定。开始时，应输入一个较大的值。然后逐渐调整输入值，直到电机开始失速，这样可以优化应用中的这些设置。

加加速度

在“加加速度”(Jerk) 对话框中，可选择“启用加加速度”(Enable Jerk) 复选框以向加速度或减速度应用加加速度补偿。加加速度补偿提供较平稳的位置控制，方法是减少移动包络的加速和减速部分中的加加速度（速率变化）。减少加加速度可改善位置追踪性能。

扭矩限值及固定停止检测

前提

- 在硬件拓扑网络， IRT 网络添加了 PROFINET 设备并组态了 105 报文。
- 在硬件接口页面选择了该设备。

扭矩限值

在“扭矩限值”(Torque limiting) 中组态驱动器的扭矩限值百分比。

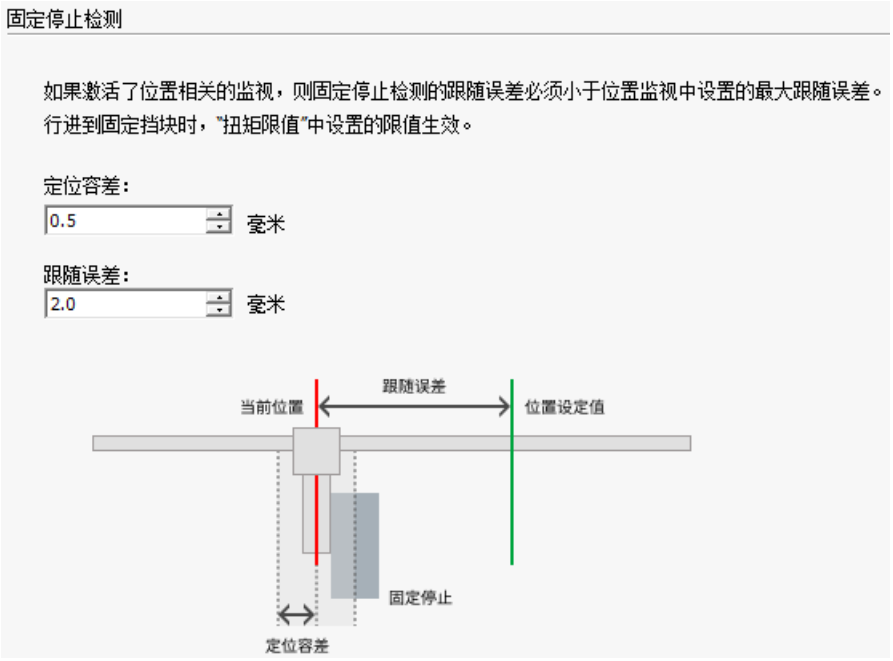
基于位置的监视

当启动扭矩/力限制时，如果扭矩已经到达阈值，电机位置闭环控制会受到影响，导致实际位置与设定位置持续偏离，跟随误差持续增加。

为了及时检测并响应上述情况，可以使能位置监视，包含跟随误差，standstill 等监测。

若勾选“禁用位置相关监视”，当 TorqueLimiting 进入使能状态后，位置监视失效，当 TorqueLimiting 退出使能状态后，位置监视恢复。

固定停止检查



定位容差	超出定位容差即视为固定挡块脱离或折返。为检测固定挡块是否脱离或折返，位置设定值必须处于定位容差范围外。组态的定位容差必须小于已组态的跟随误差。
跟随误差	固定挡块检测开始生效时的跟随误差值。如果驱动装置在运动作业期间因机械固定挡块停止运行，则会增大跟随误差。累积的跟随误差用作固定挡块的检测标准。

InClamping 工作机制

进入 InClamping 状态

通过持续监测跟随误差是否超过预设的设定阈值来判定轴是否已接触到固定挡块。

状态行为：

- 一旦轴进入 InClamping 状态，当前正在执行的运动指令将被中止。
- 系统将立即记录轴的当前实际位置。
- 轴将进入 Standstill 状态 。

在此状态下，由于设定位置与实际位置之间存在一定偏差，系统将产生一个维持扭矩，使轴牢固地夹紧在挡块位置。

如果在 InClamping 状态下，再次尝试执行与挡块方向相同的运动指令，该指令将报错，错误信息为 “InClamping 下执行 FB 错误” 。需要注意的是，尽管指令报错，轴不会进入 ErrorStop 状态，并且电机将继续对挡块施加压力，保持夹紧状态。

正常脱离 InClamping 状态

正常脱离 InClamping 状态需要通过执行与挡块方向相反的运动指令。

在此脱离过程中，轴的运动状态将由当前调用的指令进行管理和控制。

当轴的实际位置超出定位公差的设定范围时，系统即判定轴已成功脱离 InClamping 状态。

异常脱离 InClamping 状态

在未执行反方向运动指令的情况下，如果轴的实际位置意外地脱离了定位公差的设定范围，则视为异常脱离 InClamping 状态。

异常脱离可能由以下情况引起：

- 机械挡块被人为挪走。
- Limit 值过大，导致轴突破挡块限制。

一旦发生异常脱离，系统将立即报错，错误信息为 “异常脱离 InClamping”。此时，轴将进入 ErrorStop 状态，需要通过执行 MC_Reset 指令进行复位操作才能恢复。

异常脱离判定

当设定位置（或目标位置）位于定位公差外，而实际位置却从挡块方向进一步偏离，超出了定位公差外设定范围时，系统将判定为异常脱离。

说明

在异常脱离发生时，由于跟随误差的急剧变化，电机可能会产生快速移动，进而引发过载错误。

如果目标位置仍然在定位公差的设定范围内，系统将无法检测到异常脱离情况。

主动回原点

当轴触发了主动回原点操作，则轴就会按照组态的速度去寻找原点信号，并完成回原点命令。

模式选择

选择回原点模式，主动回原点支持以下回原点模式：

- 使用来自伺服驱动器的零位信号
- 使用参考挡块及来自伺服驱动器的零位信号
- 使用数字量作为原点信号
- 使用参考挡块及数字量输入作为原点信号

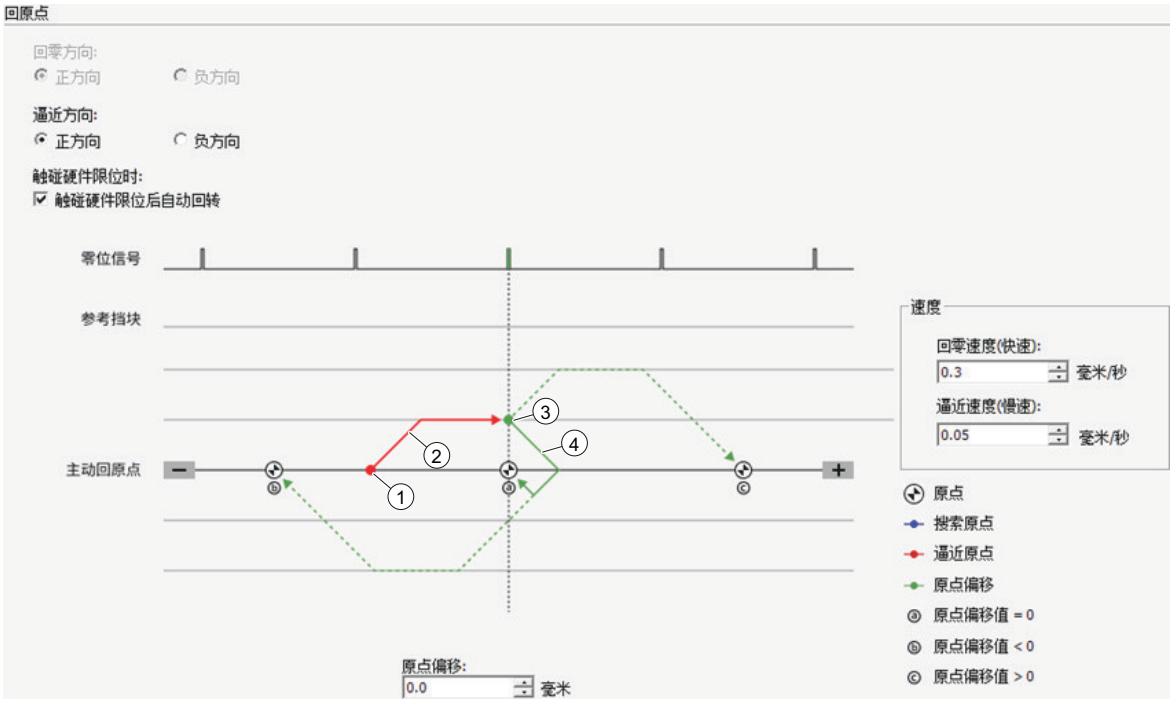
输入信号

- 输入：选择原点信号。
- 触发电平：原点信号的有效电平。
- 原点检测方向：有效的输入信号有两个边沿，指定其中一个边沿作为原点信号。

回原点

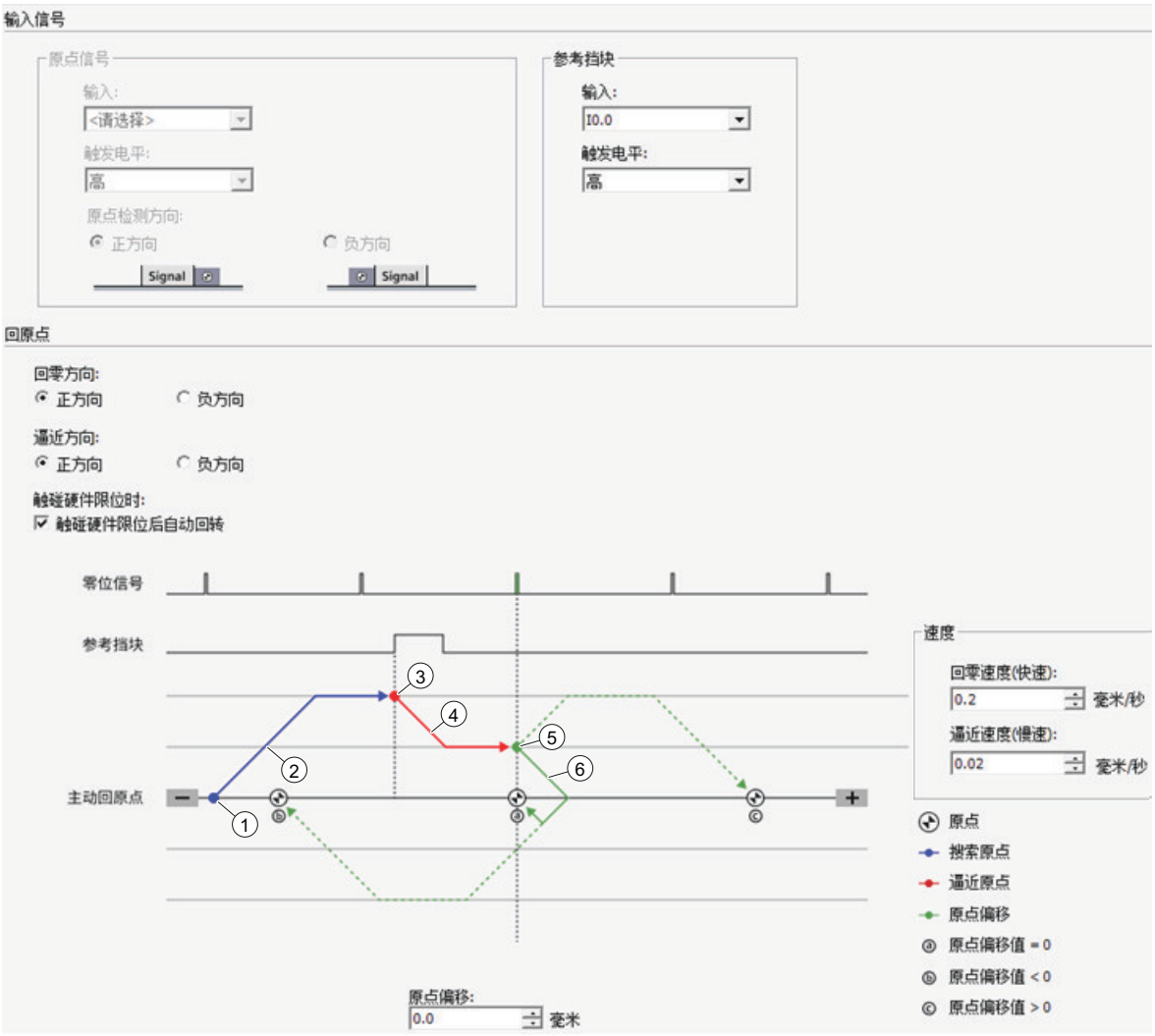
- 回零方向：指定搜索原点信号的初始运动方向。当程序中触发了主动回零后，轴立即以“回零方向”运动来搜索原点信号。
- 逼近方向：指定最后接近原点信号的方向。
- 触碰硬件限位后自动回转：勾选后，在搜索原点信号的过程中，如果触发了硬件限位，轴会自动反向继续搜索。否则，轴会停止，指令报错。
- 回零速度：指定搜索原点信号的运动速度。当程序中触发了主动回零后，轴立即以“回零速度”来搜索原点信号。
- 逼近速度：指定最后接近原点信号的速度。在轴搜索到原点信号后，立刻以逼近速度继续运行，并以逼近方向再次接近原点信号。“逼近速度”要小于“回零速度”，“回零速度”和“逼近速度”都不宜设置得过快。在可接受的范围内，建议设置较慢的速度值。
- 原点偏移：最终原点位置与原点信号的偏移。当设置为 0 时，原点位置和原点信号重合。
- 零位信号：零位信号通常由增量式编码器产生，增量式编码器每转一圈都会产生一个零位信号。在回零过程中，零位信号可以作为高精度的“原点信号”。
- 原点信号：原点信号是定位系统原点的基准位置。它可以是参考挡块信号的某个边沿，也可以是高精度的零位信号。它是系统确定“原点”的关键参照点。在回零路径图中，原点信号通常是红色线段的终点。

示例：使用来自伺服驱动器的零位信号



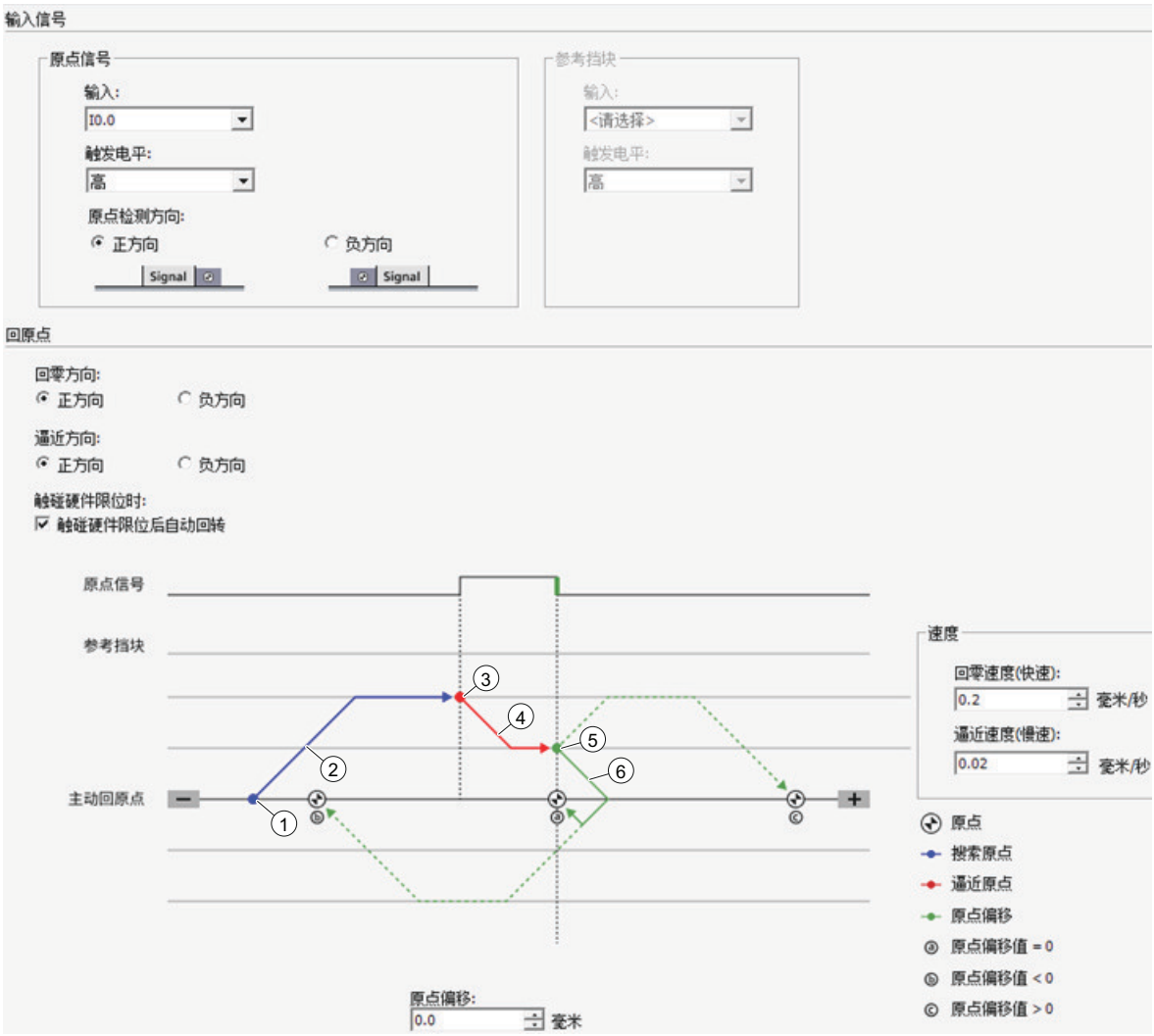
①	通过运动控制指令“MC_Home”开始主动回原点
②	以逼近速度在逼近方向上搜索零位信号
③	检测到零位信号
④	以回零速度运动到原点位置。如果设置了原点偏移，电机将移动到偏移后的位置。

示例：使用参考挡块及来自伺服驱动器的零位信号



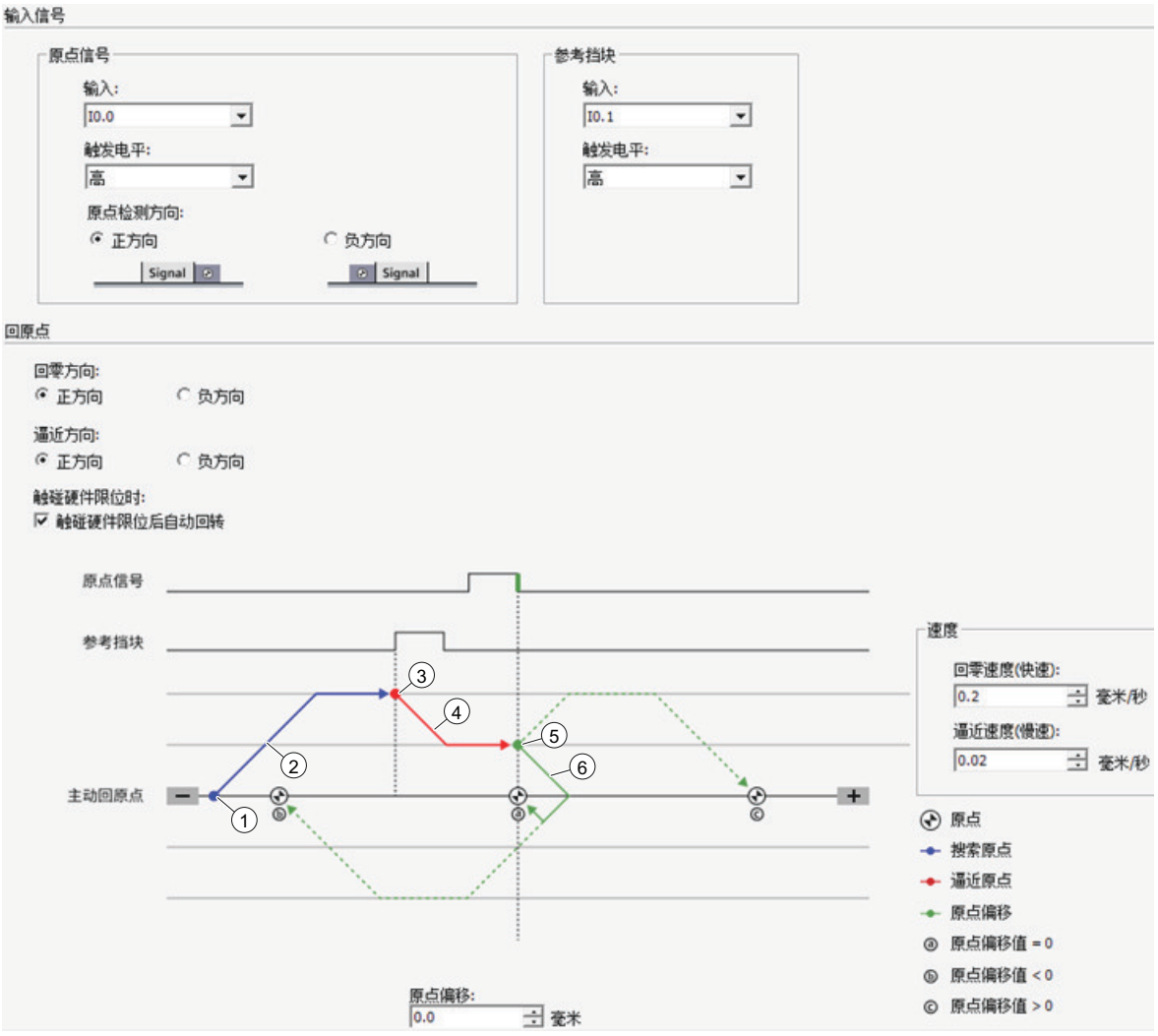
①	通过运动控制指令“MC_Home”开始主动回原点
②	以回零速度在回零方向上搜索参考挡块信号上升沿
③	检测到参考挡块的上升沿
④	以逼近速度在逼近方向上搜索零位信号
⑤	检测到零位信号
⑥	以回零速度运动到原点位置。如果设置了原点偏移，电机将移动到偏移后的位置。

示例：使用数字量作为原点信号



①	通过运动控制指令“MC_Home”开始主动回原点
②	以回零速度沿回零方向搜索原点信号上升沿
③	检测到原点信号的上升沿
④	以逼近速度在逼近方向上搜索原点信号的下降沿
⑤	检测到原点信号的下降沿
⑥	以回零速度运动到原点位置。如果设置了原点偏移，电机将移动到偏移后的位置。

示例：使用参考挡块及数字量输入作为原点信号



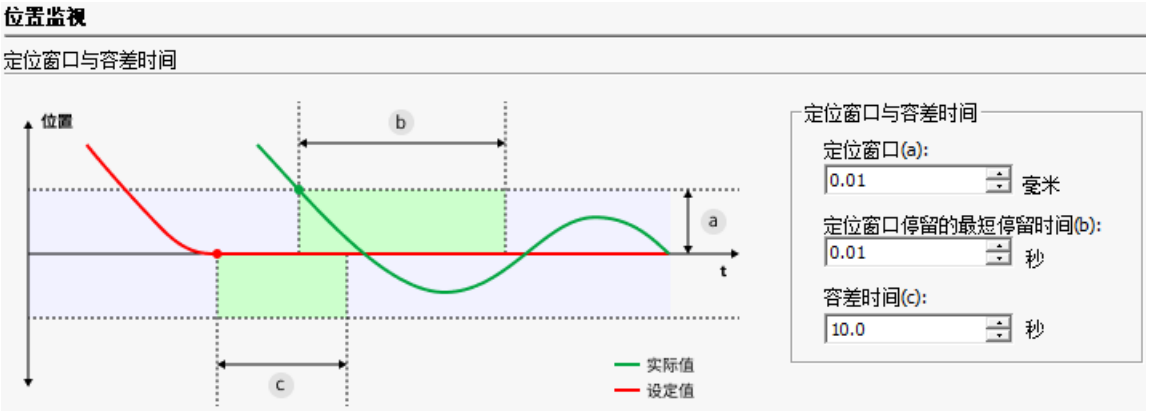
①	通过运动控制指令“MC_Home”开始主动回原点
②	以回零速度在回零方向上搜索参考挡块信号上升沿
③	检测到参考挡块的上升沿
④	以逼近速度在逼近方向上搜索原点信号
⑤	检测到原点信号
⑥	以回零速度运动到原点位置。如果设置了原点偏移，电机将移动到偏移后的位置。

位置监视

位置监视(Positioning monitoring)

组态用于监视目标位置的标准。

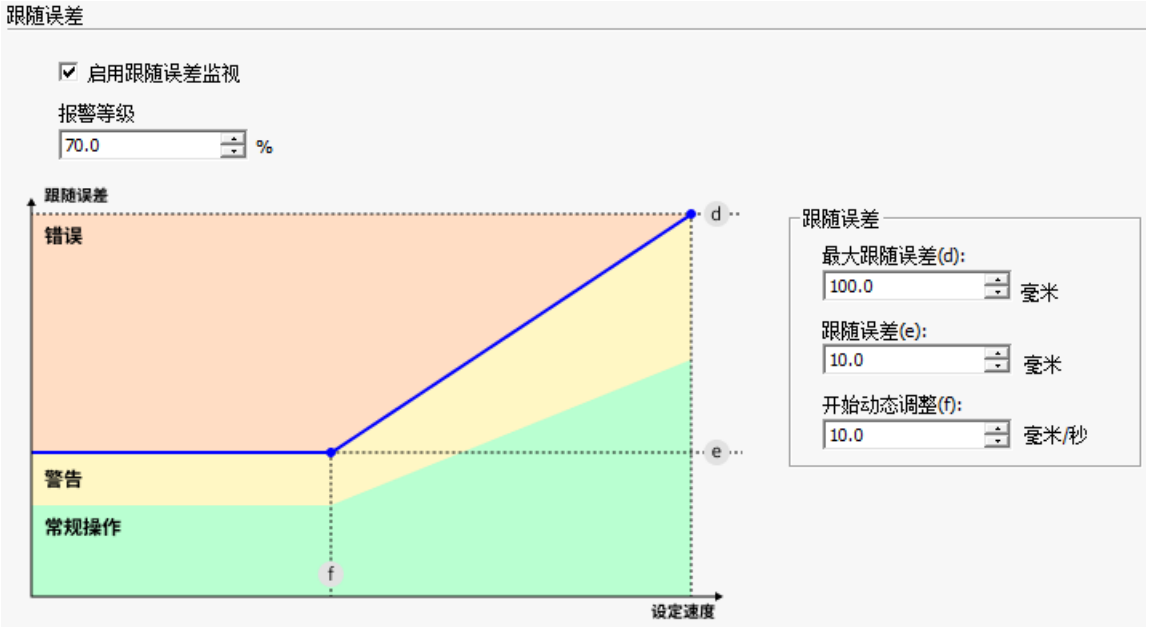
用于在设定值计算结束时对实际位置的状态进行监控。当实际位置在容差时间内到达定位窗口，且在最短停留时间内停留在该窗口，这样就完成了一个运动命令。



定位窗口(a)	组态定位窗口的大小。
在定位窗口停留的最短时间(b)	组态最短停留时间。实际位置值在“定位窗口”范围内的持续时间必须大于“在定位窗口中的最短停留时间”。
容差时间(c)	组态容差时间。一旦速度设定值达到零值，则实际位置值必须在“容差时间”范围内到达“定位窗口”。

跟随误差(Following Error)

组态轴的实际位置与设定位置之间的容许偏差。可根据当前轴速度动态调整跟随误差。

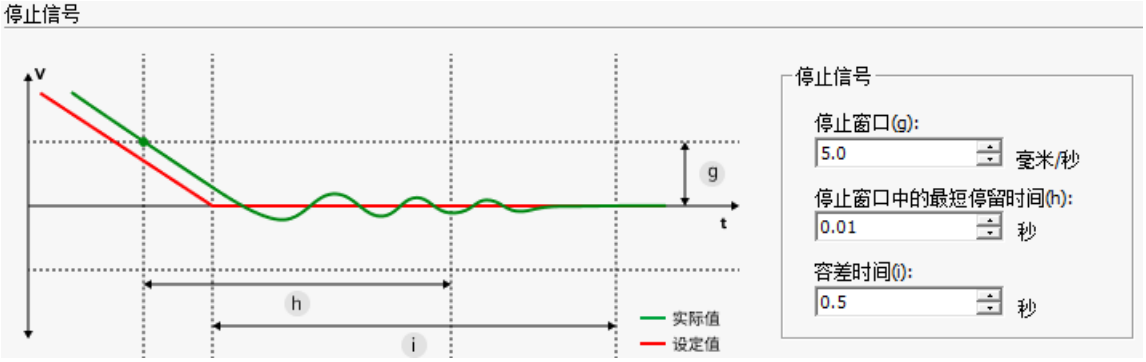


启用跟随误差监控	勾选此选项时，轴在错误范围（橙色）内停止。
最大跟随误差(d)	组态最大速度时容许的跟随误差。
跟随误差(e)	小于“启动动态调整”速度时的容许跟随误差（无动态调整）。
开始动态调整(f)	超过该组态速度时，将会动态调整跟随误差。
报警等级	当前跟随误差限值的一个百分比值；超过该百分比值时，将会输出一个警告。

跟随误差是轴的位置设定值与实际位置值之间的差值。计算跟随误差时，会将设定值到驱动器的传输时间、实际位置值到控制器的传输时间计算在内。允许跟随误差取决于速度设定值。当速度设定值小于“开始动态调整”时，跟随误差的容许范围为常数；而当速度设定值高于“开始动态调整”时，跟随误差则随速度设定值按比例增长。当设定位置值与实际位置值之间超出允许跟随误差容许偏差范围时，轴将停止运行。在运动轴调试初期，可以将“跟随误差”设置为较大值以避免运动轴频繁出现跟随误差错误。

停止信号(Standstill signal)

组态停止检测标准。

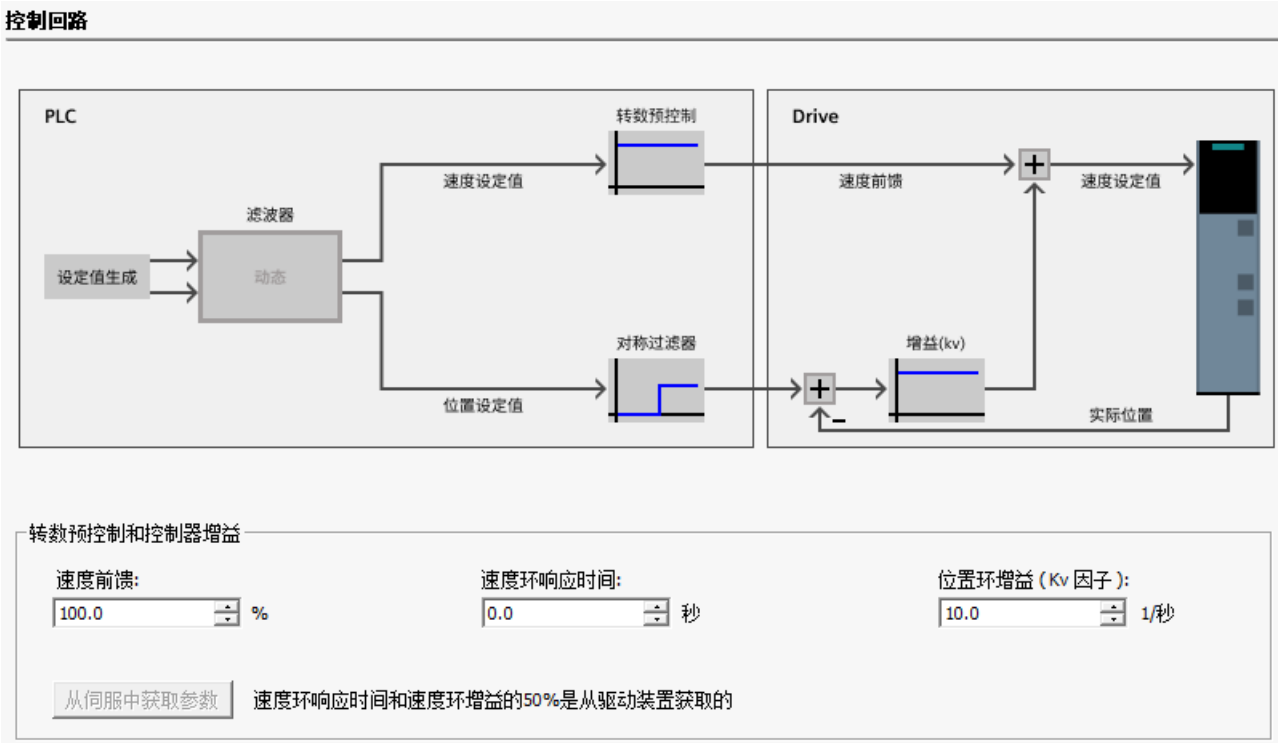


停止窗口(g)	组态停止窗口的大小。
在停止窗口停留的最短时间(h)	组态停止窗口中的最短停留时间。轴的实际速度必须在“停止窗口”内保持“停止窗口停留的最短时间”，轴显示停止。
容差时间(i)	组态容差时间在该容差时间内，位置值必须到达停止窗口。

控制回路

组态位置控制回路的速度前馈和位置环增益（kv 因子）。

激活速度速度前馈后，设定值会被延迟速度环响应时间，然后再设定控制偏差。这样可防止实际位置越过或超前位置设定值。速度环响应时间是速度控制回路动态特性的一种简化替代模型。速度环响应时间包含在平衡过滤器中。



速度前馈	修改控制回路的速度前馈百分比。 “速度前馈”可提高系统的动态响应特性，但过大的设置值会使位置控制系统超调。
速度环响应时间	组态速度环响应时间(Tvtc)。
位置环增益 （Kv 因子）	• 定位精度和停止控制 • 运动的一致性 • 定位时间 轴的机械状态（硬度越高）越好，可以组态的 Kv 因数越大。这可以减小跟随误差，获得更大的动态响应。

从伺服中获取参数

您可以通过此功能获取 PN IRT 轴控制回路的伺服参数

说明

目前此功能仅支持 S200

前提

- 电机已成功与 PLC 连接，已成功配置了轴。
- 从伺服中获取参数之前，需确保伺服已经优化。

步骤

要使用在驱动装置中为轴组态位置控制，请按以下步骤操作：

1. 选择“从伺服中获取参数”选项。
2. 输入伺服的用户名和密码进行登录（伺服端用户管理功能启用时）。

速度前馈	自定义，不从伺服中获取，默认值为 100%。
速度环响应时间	获取伺服参数 r5277：前馈对称时间估算，单位已转换。
位置环增益 （Kv 因子）	获取伺服参数 r5276：估算出的 kv 系数，值为 r5276 的 50%，单位已转换。

说明

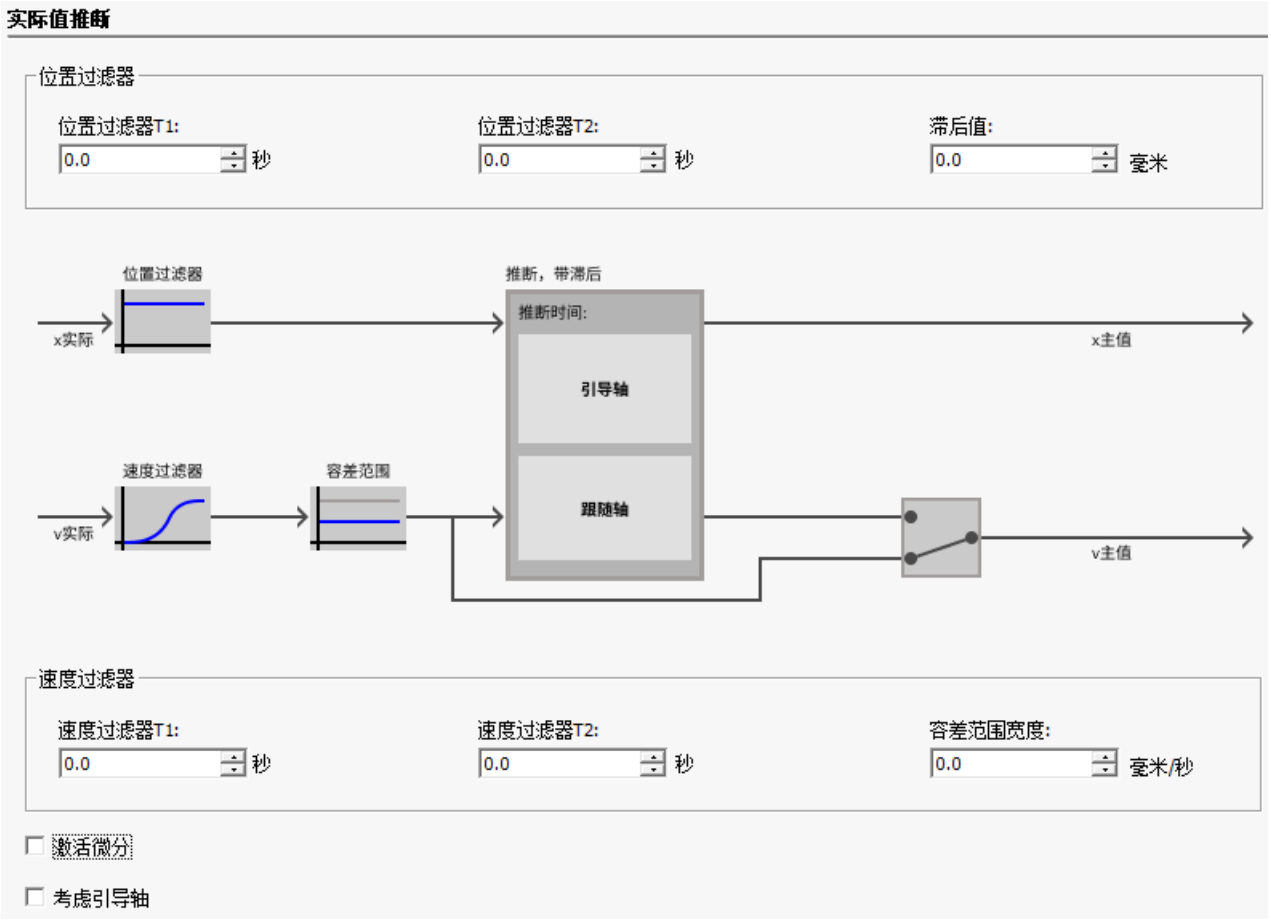
如果设备名称配置错误或网卡接口选择有误，系统会提示您设备未找到，需重新选择设备，点击“是”之后，您可在弹出的对话框中重新选择设备。

说明

如果设备尚未配置 IP 地址，系统将弹出一个对话框，引导您进行 IP 地址设置。成功完成设置后，该 IP 地址将同步更新至硬件拓扑网络和实际设备中。

实际值推断

在“实际值推断”中为用于同步操作耦合的实际值组态外插补。此处设置的值仅在当前轴的实际值用作主值时生效。



位置过滤器 T1 和 T2

T1 和 T2	输入 PT2 滤波器的时间常数对位置进行滤波。
滞后值	输入应用到位置的外推实际值的迟滞功能值。指定值采用组态的长度单位。

速度过滤器

速度过滤器 T1 和 T2	输入用于平滑实际速度的 PT2 滤波器的时间常数，以及平滑后的实际速度的容差带宽。
容差带宽	为优化容差带宽的应用，请输入与容差区间相同的带宽作为噪声信号带宽。
激活微分	选中此复选框后，主值速度将通过对外推主值位置进行微分而获得。 取消选中此复选框后，将应用滤波后的实际速度。
考虑引导轴	选中此复选框后，引导轴相关的外推时间将包含在有效外推时间的计算中。 取消选中此复选框后，引导轴相关的外推时间将不包含在有效外推时间的计算中。

诊断信息

命令

诊断功能可用于监视已组态轴的最重要状态和错误消息。

可通过单击“开始”(Start) 按钮和“结束”(Stop) 按钮来启用和禁用运动控制诊断过程。

轴状态

对话框中显示了轴状态。详细信息将显示在“错误码 (页 1068)”(ErrorCode)、“位置”(Position)、“速度”(Velocity)、“方向”(Direction) 对话框中。

轴状态	说明
错误	轴发生错误。有关详细信息，请参见运动控制指令中的 ErrorID 和 ErrorInfo 参数
禁用	轴已禁用且无法通过运动控制进行控制
待机	轴为静止状态
离散运动	轴以离散状态运动

轴状态	说明
连续运动	轴以连续状态运动
同步运动	轴以同步状态运动
回原点中	轴正在返回原点
停止中	轴正在停止
错误停止	轴正经历错误状态，当前处于 ErrorStop 状态
已归位	轴已返回至其原点位置
件正限位开关	已达到或超过硬件正限位开关。
件负限位开关	已达到或超过硬件负限位开关。
软件正限位开关	已达到或超过软件正限位开关。
软件负限位开关	已达到或超过软件负限位开关。
参考挡块	参考挡块
原点信号	数字输入
轴在固定挡块处锁紧	夹紧
配置错误	配置错误
跟随误差超限	跟随误差超限
跟随误差告警	跟随误差告警
配置告警	配置跟随误差告警

伺服状态

对话框中显示了伺服状态和名称。

14.4.8 组态轴组

14.4.8.1 轴组基本设置

轴组信息

添加轴组后，将显示如下图所示的“轴组基本设置”(Axes Group Basic Settings) 对话框。

可为新添加的轴组组态自定义名称。默认名称为“AxesGroup_x”，其中“x”代表按照数字顺序创建的轴组。

说明

- 轴组名称最多可包含 23 个字符，包括字母、数字、中文字符、下划线和扩展字符（ASCII 128 至 ASCII 255）。第一个字符仅限使用字母或下划线。不允许使用关键字作为变量名。轴组名称不可重复，也不能与现有变量名相同。
- 可以在运动控制中创建最多 2 个轴组。
- 可在不同轴组中重复使用同一个轴。

轴组类型

对于轴组类型，选择适当的维数：

- 2D 轴组：用于平面运动控制，如 X-Y 平面
- 3D 轴组：用于空间运动控制，如 X-Y-Z 坐标系

轴选择

选择要包含在 Axes Group 中的单轴。PTO 轴和虚轴都可以被选择到轴组中。

仅支持线性轴，且轴组内的轴单位必须一致。

参数

设置轴组的参数，如最大速度、最大加速度、最大减速度。

14.4.8.2 诊断信息

命令

诊断功能可用于监视已组态轴组的最重要状态和错误消息。

可通过单击“开始”(Start) 按钮和“结束”(Stop) 按钮来启用和禁用运动控制诊断过程。

轴组状态

对话框中显示了轴组的状态。详细信息将显示在“错误代码 (页 1068)”(ErrorCode) 对话框中。

状态消息 - 轴组	说明
Error	轴发生错误。有关详细信息，请参见运动控制指令中的 ErrorID 和 ErrorInfo 参数
Disabled	轴组已禁用且无法通过运动控制指令来控制。
Standby	轴组处于“备用”状态
Moving	轴组处于“正在移动”状态
Stopping	轴组处于“正在停止”状态
ErrorStop	轴正经历错误状态，当前处于 ErrorStop 状态

14.4.9 组态凸轮

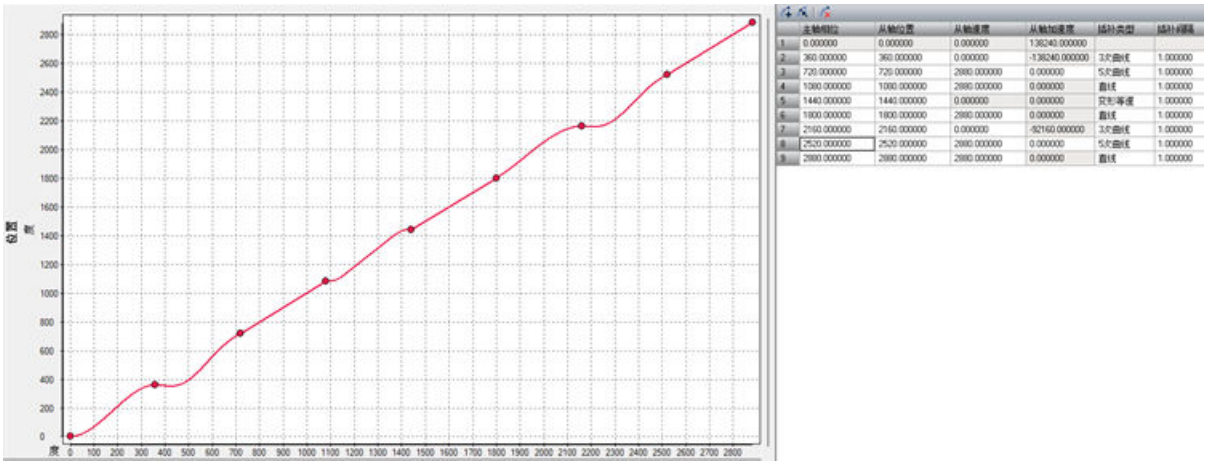
与传统凸轮机构相比，电子凸轮拥有更好的灵活性、准确性和效率。电子凸轮根据预定义的电子凸轮表，使用脉冲串控制从轴与主轴同步。

14.4.9.1 组态凸轮表

凸轮表是一份数据表，用于指定跟随主轴移动的从轴的位置。水平轴代表主轴相位，而垂直轴代表从轴位置。

凸轮表设置主轴相位和从轴位置。主轴始终处于绝对坐标系中。从轴位置始终处于相对坐标系中。

当在凸轮表中输入关键点，并在这些关键点之间设置曲线（例如直线或 3 次多项式曲线）时，一条曲线会将这些点全部连接起来，如左图所示。如此会产生一个从轴运行曲线。



添加或删除关键点

添加关键点时请注意以下事项：

- 不允许删除第一行或在其上方添加行。这意味着首个关键点的主轴相位和从轴偏移量默认设置为 0 且不能更改。
- 必须为凸轮表添加至少 2 行数据。
- 主轴相位必须按升序排序。
- 主轴上的最后一个点的位置和周期共同确定了主轴的运行速度。可参数面板中设置周期。
- 添加的点应为两行之间的中间点，例如，如果在 0 到 1800 的中间添加了一行，则添加的行的位置将为 900 点。
- 如果在最后一行下方添加一个点，则将最后一行的关键点视为中间点，并在最后一行基础上加上前两行之间的差值。例如，在 0 和 1800 下方添加了一行，则添加的行的位置为 3600。

在当前位置下方添加行

在当前位置上方添加行

删除当前行

可根据应用需求在关键点组态表中设置凸轮关键点。

组态凸轮时注意下列规则：

数量限制	说明
每个 PLC 允许添加的最大凸轮数量	8
每个凸轮表支持的最大关键点数	361
所有凸轮表可添加的最大关键点数	1444

为关键点设置值

更改主轴相位、从轴位置、从轴速度、从轴加速度、插补类型或插补间隔，双击要编辑的单元格并输入新值。

说明

凸轮表的第一行无法编辑。

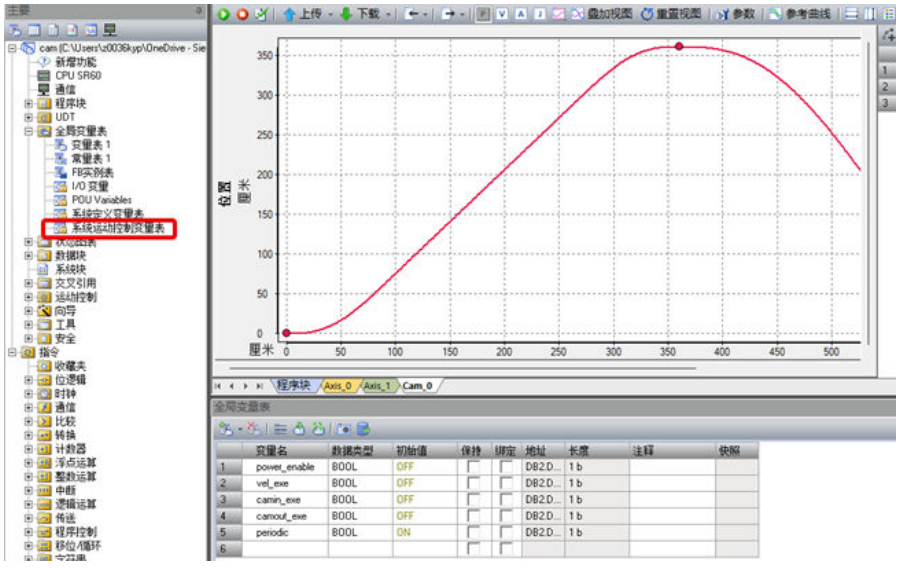
参数	说明
主轴相位	设置主轴相位
从轴位置	设置从轴位置
从轴速度	• 当插补类型为常数，或变形等速时，从轴速度设置为 0 且不可编辑。 • 插补类型为直线时，会在选择时自动创建从轴速度。 • 插补类型为三次多项式或五次多项式时，可以手动编辑。
从轴加速度	插补类型为五次多项式时，可以手动编辑。
插补类型	要设置关键点的插补类型，单击对应单元格中的向下箭头并在下拉列表中选择插补类型。 曲线类型：常量、直线、三次多项式、五次多项式、变形等速
插补间隔	每个凸轮的最大点数：65535 根据主轴位置在每个间隔处进行插补。

变量

组态凸轮时，会在全局变量表中创建以下三个变量：“Cam_x”、“Cam_x_Data”、“Cam_x_Node”。

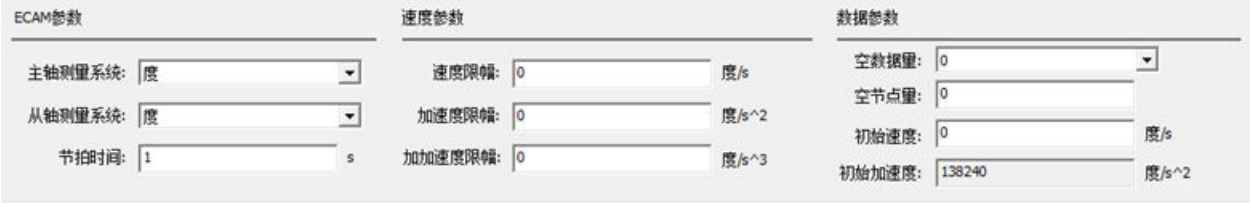
说明

“Cam_x_Data”和“Cam_x_Node”的值可通过程序段或状态图标更改。



14.4.9.2 设置参数

可以单击工具栏中的 以显示或隐藏参数设置区域。



ECAM 参数

主轴单位　　单击向下箭头并从下拉列表中选择单位。

只影响 P V A J 水平轴单位的显示，并不检查 CamIn 时绑定的主轴的单位是否和此处设置的单位相同。

从轴单位	单击向下箭头并从下拉列表中选择单位。 只影响 P V A J 垂直轴单位的显示，并不检查 CamIn 时绑定的从轴的单位是否和此处设置的单位相同。
节拍时间	节拍时间可用于仿真并确定当前周期下的从轴速度包络特征及其最大值。这有助于确定是否发生了任何超出速度限制的行为。 **注**：此处前提为主轴处于匀速运动状态。
速度参数	
速度限制	• 速度限制涵盖正向和负向两个范围。例如，如果设置了 10 mm/s 的速度限制，则正向或负向 10 mm/s 内的任何速度都在速度限制范围内。然而，如果速度超过 10 mm/s 或低于 -10 mm/s，则超出了速度限制。 • 此速度限制设置仅影响特性曲线上标记为速度限制的范围。相反，它使用从轴各个轴组态中指定的速度限制。
加速度限制	• 加速度限制涵盖正向和负向两个范围。 • 此加速度限制设置仅影响特性曲线上标记为加速度限制的范围。相反，它使用从轴各个轴组态中指定的加速度限制。
加加速度限制	• 加加速度限制涵盖正向和负向两个范围。 • 此加加速度限制设置仅影响特性曲线上标记为加加速度限制的范围。相反，它使用从轴各个轴组态中指定的加加速度限值限制。
数据参数	
空数据量	可以在此处预定义空数据量，此预定义数据将包含在 Cam_Data 表中。
空节点量	可以在此处预定义空节点的数量，此预定义数据包含在 Cam_Node 表中。
初始速度	• 首个点的速度。 • 更改关键点的设置（包括位置、插补类型、速度、加速度）可能会影响此处的初始速度设置。 • 当 PLC 实际运行该凸轮曲线时，此设置不会影响从轴的初始速度。
初始加速度	• 首个点的加速度。 • 更改关键点的设置（包括位置、插补类型、速度、加速度）可能会影响此处的初始加速度设置。 • 当 PLC 实际运行该凸轮曲线时，此设置不会影响从轴的初始加速度。

14.4.9.3 查看曲线

缩放/移动曲线

放大曲线：可以使用鼠标左键滚动并选择要放大的特定区域。在按住 CTRL 键的同时滚动鼠标滚轮进行放大或缩小。

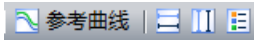
移动曲线：可以通过在曲线区域单击并按住鼠标右键来移动曲线。

曲线查看工具

可使用工具栏中的工具查看曲线。

	在此模式下，您可以通过拖动鼠标来移动时序图
	以画布中心位置放大视图，每次增加 10%。每次按下该按钮，时间轴和数值轴的范围都会缩小。 曲线显示变得更大
	以画布中心位置缩小视图，每次缩小 10%。每次按下该按钮，时间轴和数值轴的范围都会放大。 曲线显示变得更小
	在时序图区域内，按住鼠标左键并拖动，从左上角向右下角绘制一个矩形选择框。释放鼠标左键后，系统将自动放大所框选的区域

	显示主轴相位与从轴位置之间关系的曲线。该曲线无法隐藏。
	显示或隐藏主轴相位和从轴速度之间的关系图。
	显示或隐藏主轴相位和从轴加速度之间的关系图。
	显示或隐藏主轴相位和从轴加加速度之间的关系图。
	显示或隐藏限制曲线。
	叠加视图：显示同一图形中的所有曲线。
	重置视图：可以使用鼠标左键滚动并选择要放大的特定区域。按下“重置视图”(Reset View) 键返回到上一视图。



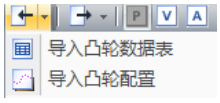
	显示或隐藏参考曲线。
	显示或隐藏水平参考线。
	显示或隐藏垂直参考线。
	显示或隐藏垂直图例。

14.4.9.4 导入/导出凸轮表

STEP7 Micro/WIN SMART V3 可导入或导出凸轮配置或凸轮数据。

导入凸轮数据

1. 要导入凸轮配置或凸轮数据，单击导入键旁边的向下键，然后选择要导入相应的表。



说明

导入凸轮配置表时应注意以下内容：

- **更新规则**：除了节点名称外，所有其他字段均将覆盖现有表中的相应数据。
- **值有效性检查**：系统自动验证导入数据的有效性。
- **主轴相位递增**：确保主轴相位值升序排列。
- **跳过不满足要求的行**：不满足上述要求的行将被忽略，且不会导入至系统。
- **点数计算和限值**：导入点数及插补后点数根据实际导入节点动态计算得来。每个凸轮配置表的最大点数限值均为 361，所有凸轮配置表的总点数不得超出 1444。
- **处理空数据**：对于空数据条目，系统将自动从下拉列表中选择最接近的向下舍入值。

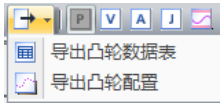
说明

导入凸轮数据表时应注意以下内容：

更新绘图区域：导入后，绘图区域将添加一条新的参考曲线。
总长度限制：导入数据表的总长不得超出 65535。

导出凸轮表

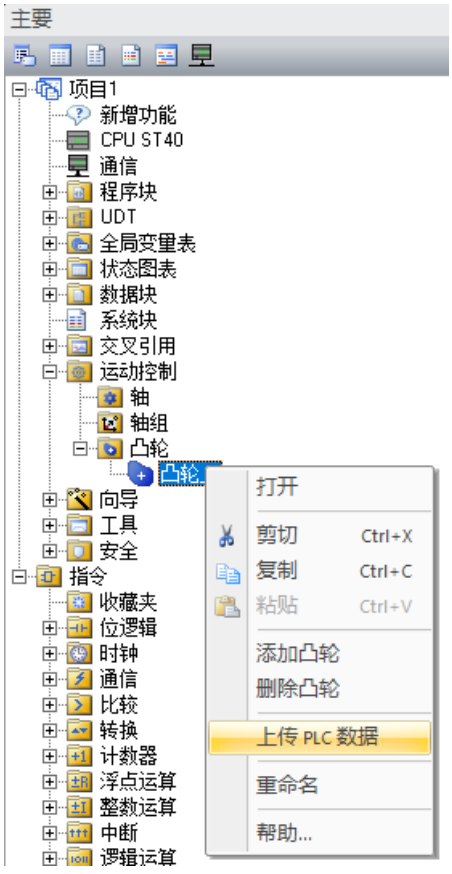
1. 要导出凸轮组态或凸轮数据，单击导出键旁边的向下键，然后选择导出相应的表。



从 PLC 上传凸轮表

STEP7 Micro/WIN SMART V3 允许以 CSV 文件形式从 PLC 中导出相应的凸轮，同时在凸轮编辑器的图形界面中将 CPU 中的凸轮曲线显示为参考曲线。

1. 连接到 PLC。
2. 单击 凸轮_x，然后在上下文菜单中选择 “上传 PLC 数据”(Upload PLC Data)。



14.5 调试（仅适用于 PTO 轴 和 PN IRT 轴）

14.5.1 控制面板

可使用 “控制面板”(Control Panel) 来帮助开发 STEP 7-Micro/WIN SMART V3 的运动控制功能。可通过该界面便捷地调试、操作和监视 STEP 7-Micro/WIN SMART V3 的工作状态。他们可以检查控制系统接线是否正确，调整运动控制参数组态并测试每个预定义的运动曲线。

前提条件

使用“控制面板”(Control Panel) 前，确保已完成以下操作：

- 您已组态轴并将项目下载到 CPU。否则，CPU 就没有运行所需的有效程序。
- CPU 工状态被设置为“RUN”模式。

打开控制面板

1. 在程序编辑器底部单击要调试的轴的选项卡。
2. 单击“控制面板”(Control Panel)。

①
- 激活：建立与 CPU 的在线连接并接管轴主控权。
- 禁用：禁用控制面板中的轴控制。

②
- 激活：启用轴，将轴状态从“Disabled”更改为“StandStill”。
- 禁用：禁用轴，将轴状态从“StandStill”更改为“Disabled”。

③ 控制面板

④ 实时轴移动状态

⑤ 显示当前轴状态

⑥ 显示轴运动时发生的错误代码 (页 1068)。

激活控制面板

单击 ① 中的 “激活”(Activate) 按钮来激活控制面板。如果不满足指定的激活条件，则会出现一个对话框，提供有关所需采取操作的分步说明。

- 如果未连接 CPU，则会弹出窗口提示连接 CPU。
- 如果 CPU 正处于“STOP”状态，且当前控制轴为 IRT 轴时，则会弹出窗口提示将 CPU 的工作状态设置为“RUN”。
- 如果当前打开的程序与 CPU 中的程序不一样，则会提示重新下载。

控制面板

单击 ② 中的 “激活”(Activate) 按钮激活电机驱动器，轴状态将从“Disabled”更改为“Standstill”。

可组态并测试“连续运行”(Continuous run)、“回原点”(Homing)、“相对运动”(Relative) 和“绝对运动”(Absolute)。轴的实时位置和速度可通过“轴位置”(Axis position) 对话框监视。

连续运行

输入速度和加速度/减速度

- 按住“正”(Positive) 可执行以正向连续速度移动的指令。
- 按住“负”(Negative) 可执行以负向连续速度移动的指令。

回原点

回原点模式/回零标志/偏移量在回原点 (页 871)中设置。可调整偏移量并单击“保存”(Save)。

- 单击“回原点”(Home)，已触发的 MC_Home 指令将 offset_adjust 的值传递给固件以进行调试。
- 单击“停止”(Stop) 以停止返回回原点位置的运动。

相对运动

绝对运动

输入速度和加速度/减速度

设置相对移动的距离和绝对移动的位置

单击“开始”(Start) 以开始移动。单击“停止”(Stop) 以停止移动。

说明

轴处于运动状态 (Continuous\Homing\Relative\Absolute) 时，如果 CPU 与 STEP 7-Micro/WIN SMART V3 断开连接的时间超过五秒，则轴将进入 ErrorStop 状态。当 CPU 重新连接到 STEP 7-Micro/WIN SMART V3 时，会在“控制面板”中看到错误 0x1090 (页 1068)。

14.5.2 参数优化

此面板用于优化 IRT 轴的位置环参数。调优时需逐轴进行，且每次仅限一个轴运行。

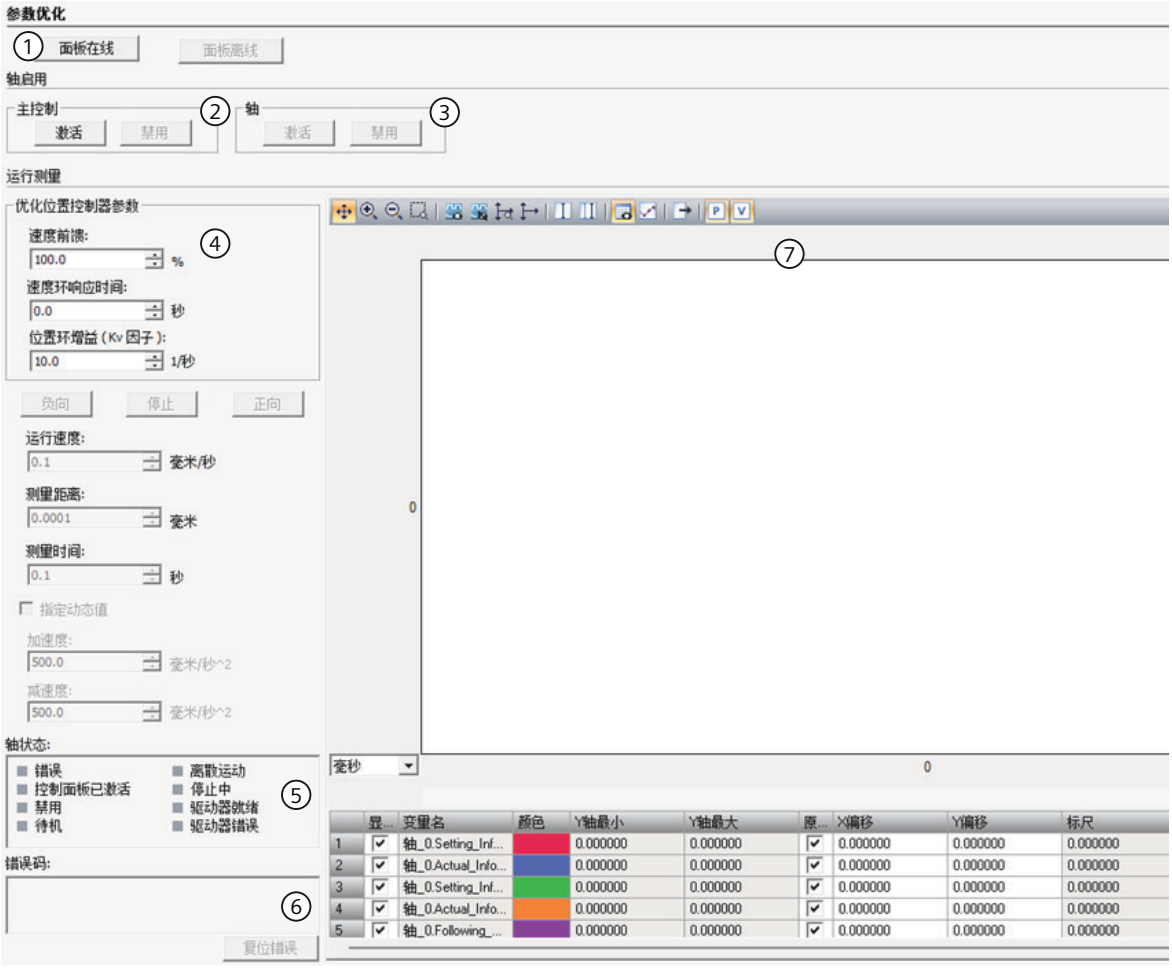
前提条件

使用“参数优化”前，确保已完成以下操作：

- 轴组态已下载至 CPU 且伺服轴成功连接。
- CPU 必须处于运行 (RUN) 模式。
- 轴对象在用户程序中处于禁用状态 (MC_Power.Enable = FALSE)。
- 轴对象无错误。

打开参数优化

1. 在程序编辑器底部单击要调试的 IRT 轴的选项卡。
2. 单击“参数优化”。



① 面板在线状态

② • 激活：建立与 CPU 的在线连接并接管轴主控权。
• 禁用：禁用控制面板中的轴控制。

③ • 激活：启用轴，将轴状态从“Disabled”更改为“StandStill”。
• 禁用：禁用轴，将轴状态从“StandStill”更改为“Disabled”。

④ 运行测量 [2]

⑤ 显示当前轴状态

⑥ 显示轴运动时发生的错误代码 (页 1068)[3]。

⑦ 轨迹 [4]

[1] 控制面板的在线/离线状态，以及主控制激活区域和轴命令区域中的按钮状态，在控制面板与参数优化页面之间保持同步。

[2] 在轴激活之前，‘运行测量’部分的输入字段和按钮将保持不可编辑状态。

[3] 若激活的参数优化面板报告轴存在错误，您可在线查看轴状态和错误代码。点击“复位错误”按钮可清除轴错误。错误清除后，轴状态将自动更新并在状态指示灯中显示。

[4] 调优过程中，可用的跟踪 ID 范围是 129 到 143。调优过程生成的跟踪数据将自动存储至 SLOT 4。

激活参数优化

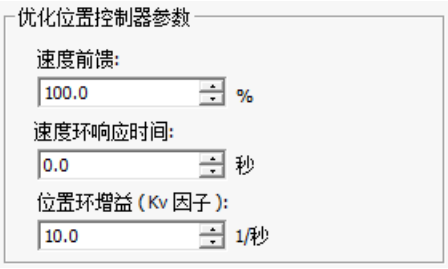
单击 ③ 中的 “激活”(Activate) 按钮来激活参数优化。如果不满足指定的激活条件，则会出现一个对话框，提供有关所需采取操作的分步说明。

- 如果未连接 CPU，则会弹出窗口提示连接 CPU。
- 如果轴处于 STOP 状态, 则会弹出窗口提示将 CPU 的工作状态设置为 RUN。
- 如果当前打开的程序与 CPU 中的程序不一样，则会提示重新下载。

优化位置控制环参数

单击 ② 中的 “激活”(Activate) 按钮激活电机驱动器，轴状态将从“Disabled”更改为 “Standstill”。

优化控制器



运行测量

轴激活并处于 Standstill 状态时，正向和反向运行按钮方可启用。当轴处于任意运行状态（如 Discrete、Continuous、Stopping 或 Homing）时，停止按钮将始终保持启用状态。此停止按钮的操作行为与 MC_Stop 功能一致。

- 运行速度：用于参数调优时的轴运行速度。其取值范围为 (0, 最大速度]。
- 测量距离：在调优过程中，轴按照设定的运行速度完成的相对移动距离。
- 测量时间：指数据轨迹（trace）的采集持续时间。
- 指定动态值：您可为调优过程指定动态值。启用此功能后，轴将按照用户设定的加速度和减速度进行运行。若未指定动态值，系统将默认采用轴配置中已设定的动态参数。

说明

Trace 最多支持记录 32768 个点。在 1 ms 的 IRT 刷新周期下，系统会根据您设定的测量时间，调整采样频率以在最大点数限制内捕获尽可能多的数据。(超过 32768 个点就减小采样频率，最终不超过 32768 个点）。

以下是采样频率和采样点的计算方式：

- 原采样点 = 测量时间 / IRT 周期（向上取整）
- 采样频率增大的倍数 = 原采样点 / 32768 + x（若原采样点是 32768 的倍数，则 x=0；否则，x=1）
- 最终的采样频率 = 采样频率增加的倍数 * IRT 周期
- 最终的采样点 = 测量时间 / 最终的采样频率（向上取整）

14.6 通用机制

PLCopen 对运动控制功能块进行了标准化，制定了 IEC 61131-3(JIS B 3503)规格语言程序接口的定义。 S7 200 SMART G2 运动控制功能遵从 PLCOpen 定义的运动控制功能，并进行了一些拓展。除了单轴定位、电子凸轮功能、多轴插补控制等功能，还对启动指令的基本步骤进行了定义。

利用 PLCopen 运动控制用功能块，可以减小对硬件的依赖，提高用户程序的再利用率。 还能降低培训、支持等的成本。

指令的执行流程

在基于功能块的运动控制编程中，每个运动指令的执行都依赖于其对应的功能块实例（Function Block Instance）。每个运动指令在程序中被使用时，都会对应一个唯一的功能块实例。每个功能块实例内部都存储着该运动指令的输入参数和输出结果数据。当一个运动指令在程序组织单元中被调用时，其执行流程如下：

1. 输入参数传递： 调用指令时提供的输入参数（例如目标位置、速度、使能信号等）会被传递并存储到该 FB Instance 的相应输入管脚数据中。
2. 指令触发与执行： 一旦输入条件满足 FB Instance 内部的逻辑就会被触发，并开始执行对应的运动控制任务。
3. 输出状态反馈： 指令的执行状态、当前位置、错误代码或完成标志等结果，会实时更新并存储在 FB Instance 的输出管脚数据中。这些输出数据可供程序的其它部分读取和使用

同一个运动指令可以创建多个实例。

输出管脚通用机制

说明

下表描述了 FB 实例输出引脚的通用机制。如果任何特定指令表现出特殊的行为，请始终以该指令的说明为准。

输出互斥性	• 在 Execute 模式下： 输出 Busy、Done、Error 和 CommandAborted 是互斥的：在一个指令上，它们之中只有一个可以为 TRUE。如果 Execute 为 TRUE，则这些输出中必须有一个为 TRUE。Active、Error、Done 和 CommandAborted 这些输出中，在同一时间只有一个会被置位。 • 在 Enable 模式下： 输出 Status 和 Error 是互斥的，在一个指令上，它们之中只有一个可以为 TRUE。
输出状态	• 在 Execute 模式下： Done、Error、ErrorID 和 CommandAborted 输出会在 Execute 的下降沿被复位。然而，Execute 的下降沿不会停止甚至不影响实际指令的执行。如果 Execute 在指令完成前被复位，响应的输出至少保留一个扫描周期。 • 在 Enable 模式下： Valid、Enabled、Busy、Error 和 ErrorID 输出会在 Enable 的下降沿同时被复位，所以不会延后一个扫描周期复位。
Done 输出的行为	当指令动作成功完成时，Done 输出被置位。当多个指令按顺序在同一轴上工作时，适用以下规则：当一个轴上的某个运动在未达到最终目标时被同一轴上的另一个运动中断，则第一个指令的 Done 将不会被置位。
Busy 输出的行为	• 在 Execute 模式下： 每个指令都可以有一个 Busy 输出，表示该功能块尚未完成，并且可以预期新的输出值。 Busy 在 Execute 的上升沿被置位，并在 Done、Aborted 或 Error 输出之一被置位时复位。 • 在 Enable 模式下： 每个指令都可以有一个 Busy 输出，表示该功能块正在工作，并且可以预期新的输出值。Busy 在 Enable 的上升沿被置位，并保持置位，只要指令正在执行任何动作。Busy 和 Error 可以同时为 TRUE。错误期间，指令继续处于工作状态，当错误被恢复后，Error 自动复位。
Active 输出	Active 在指令接管相应轴的运动控制时被置位。接管控制意味着插补（IPO）被执行。 Active 至少比 Busy 滞后一个 POU 周期，因为只有当指令被扫描时，Active 输出引脚才会被刷新。

14.6.1 指令重启

基本描述

前置条件

- 运动控制 FB 实例已由上升沿信号触发并启动，且当前处于 Busy 状态（即正在执行或处于 buffer 状态）。

执行流程与结果

再次触发执行指令，FB 实例支持重复触发功能，并且所有输入参数均合法。

当前正在运行的 FB 参数将被更新。同时，CommandAborted 输出引脚不会置位。

说明

当 MC_TouchProbe 指令的 Execute 引脚被置高时，如果在此期间更改了轴 ID，则重新触发该指令将针对新的轴 ID 执行探针功能。

实例

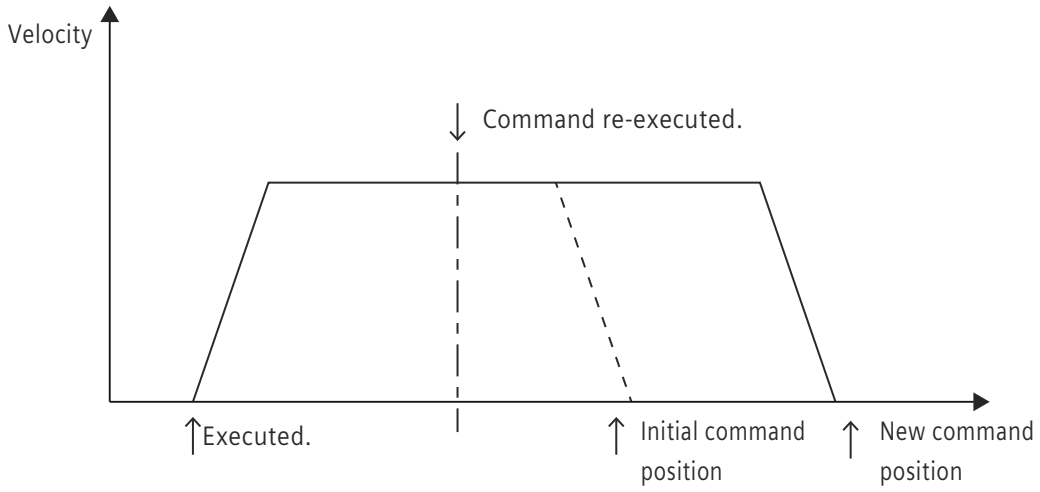
前置条件

- 轴的初始位置为 0。

执行流程与结果

执行条件	结果
1. 通过一个上升沿信号触发 MoveRelative。运动的参数设定为：Distance 100，Velocity 5，Acceleration 5，Deceleration 5，以及 Jerk 0。 2. 当轴在执行上述运动过程中，其位置达到 50 时，MoveRelative 再次接收到一个上升沿触发信号。此时，新的运动参数被设定为：Distance 150，Velocity 5，Acceleration 5，Deceleration 5，以及 Jerk 0。	• 轴首先从静止状态，以 5 的加速度加速至 5。 • 当轴抵达 Position 50 时，运动指令重启，轴随即以 5 的速度匀速运动，超过第一次触发时的 Target position 100，并最终在 150 的位置停下来。

匀速运动时重启指令



运动指令重启与中断指令（Abort）

运动指令重启与中断指令（Abort）在行为上相似。两者均会基于轴当前的实时运动状态（包括当前速度、加速度和减速度），并结合新指定的运动参数（即更新后的目标速度、加速度、减速度及目标位移），重新进行运动规划。因此，在运动曲线的变化是基本一致的。

然而，在目标位移的参考点上，两者存在区别。

- 运动指令重启：其目标位移始终相对于前一个运动的原始起始点进行计算。
- 中断：以指令触发时的当前轴位置为起点进行规划。

支持单实例重启的指令及参数

指令的重新执行，仅在可更新参数（即以下列出的参数）被修改后，并通过重新触发该功能块实例来实现。如果修改了非可更新参数，即使重新触发功能块实例，系统也将忽略该触发请求。

单轴

指令	Position/Distance	Velocity	Acceleration	Deceleration
MC_MoveAbsolute	Y	Y	Y	Y
MC_MoveRelative	Y	Y	Y	Y
MC_MoveVelocity		Y	Y	Y
MC_Stop				Y
MC_Halt				Y

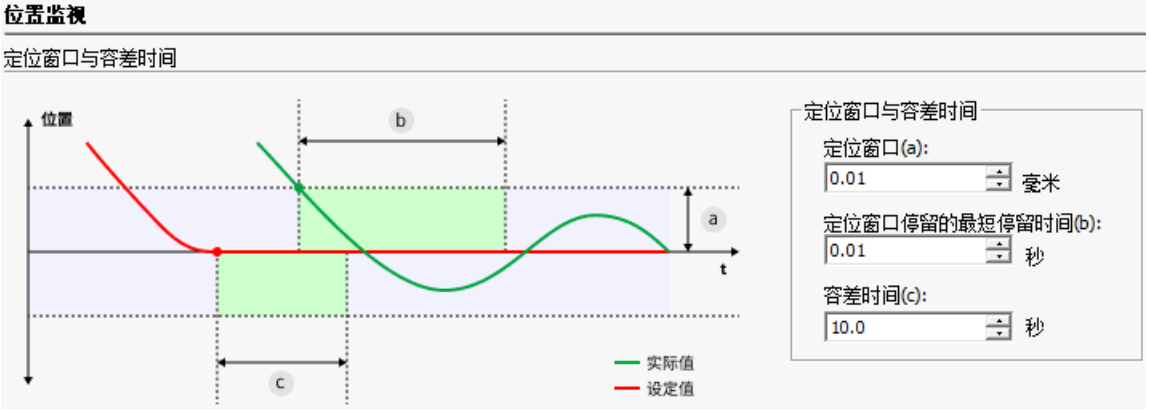
凸轮和齿轮

- MC_CamIn: 只能修改管脚上的 CamTable。
- MC_GearIn: 只能修改管脚 RatioNumerator, RatioDenominator, Acceleration, Deceleration。

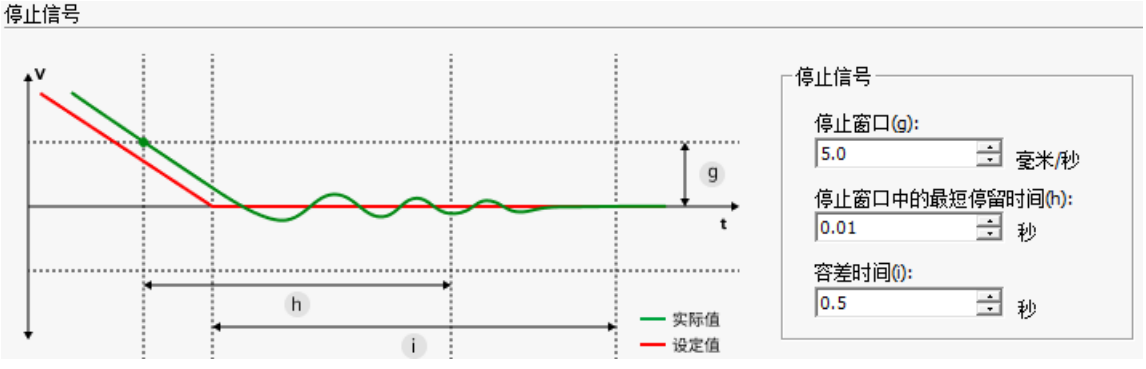
不生效情况

即使是支持重启的参数，其更新也可能在特定条件下不生效。

- 对于目标位置的更新，如果发生在给定值已经结束，则会被忽略。以 IRT 轴为例，如下图所示，如果轴已经进入 C 段进行位置监视，则目标位置更新的重启会被忽略。



- 对于速度控制指令，比如 MC_Stop，如果给定速度已经到 0，则不再响应重启。以 IRT 轴为例，进入速度监视后，就不再响应重启。

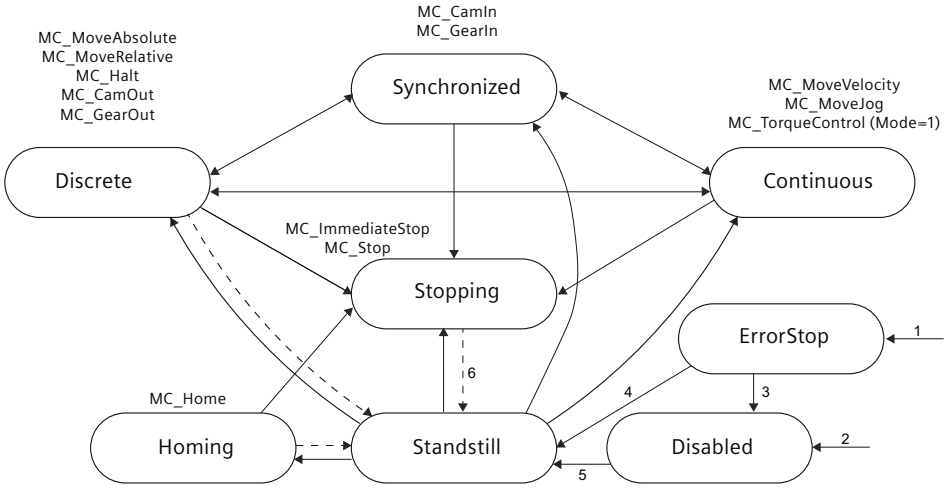


14.6.2 状态机

14.6.2.1 单轴状态机

单轴状态

启动轴对应的运动控制指令后，轴的运动情况如下图所示。
运动控制指令按顺序执行，且轴处于下表所述状态中的一种。



① 轴的故障检测逻辑检测到故障时会进入此状态。

② 如果轴正常且 MC_Power.Enable=FALSE

③ 调用 MC_Reset 以复位轴故障且 MC_Power.Status=FASLE 时

④ 调用 MC_Reset 以复位轴故障且 MC_Power.Status=TRUE 时

⑤ MC_Power.Enable=TRUE 且 MC_Power.Status=TRUE 时

⑥ MC_Stop(MC_ImmediateStop).Done=TRUE 且
MC_Stop(MC_ImmediateStop).Execute=FALSE 时

状态		说明
伺服关闭	Disabled	轴处于伺服关闭、停止或启动就绪状态。
	ErrorStop	轴的故障检测逻辑检测到故障时，轴处于伺服关闭、正在停止状态。

状态		说明
伺服开启	Standstill	轴处于伺服开启、停止状态。
	Discrete	执行具有指定目标位置的定位的状态。 包括等待定位完成的状态，以及定位操作过程中将值设置为“0”和将速度设置为“0”的状态。
	Continuous	执行没有指定目标位置的连续操作的状态。 速度控制期间变更至此状态。 此状态还包括目标速度设置为“0”和速度设置为“0”的状态，以及连续运行过程中过冲值设置为“0”和速度设置为“0”的状态。
	Synchronized	通过同步控制指令对轴状态进行同步控制。 此状态还包括同步控制指令切换后的同步等待状态。
	Stopping	通过 `MC_Stop` 指令或 `MC_ImmediateStop` 指令使轴停止的状态。 此状态还包括通过 MC_Stop 指令使轴停止后 Execute 为 TRUE 的状态。 此状态下，轴指令无法执行操作。
	ErrorStop	轴为伺服开启状态，且轴的故障检测逻辑检测到故障。 其中包括轴因异常情况而减速的状态。 ErrorStop 状态下无法执行轴运动指令。必须使用 `MC_Reset` 指令复位轴。
	Homing	使用 `MC_Home` 搜索原点位置时的状态。

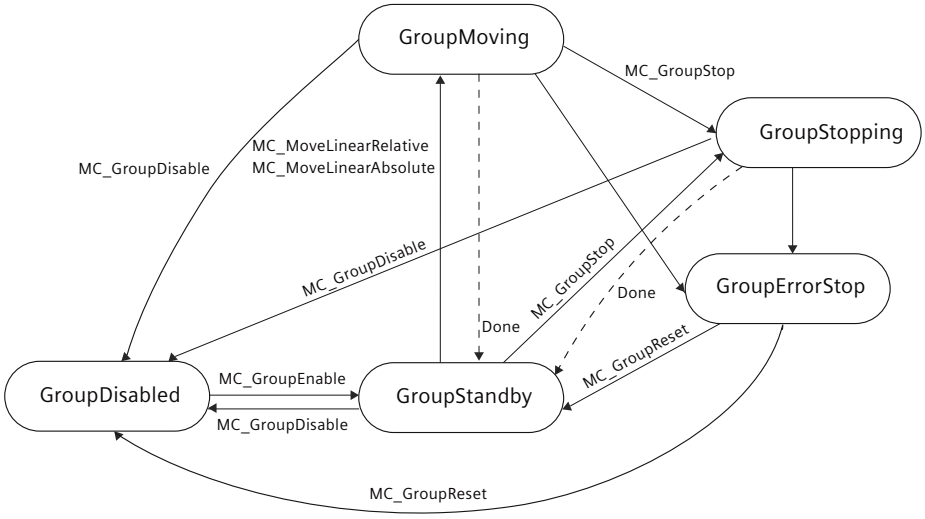
说明

所有运动控制指令都不能在中断程序中调用。

14.6.2.2 轴组状态机

轴组状态机

在为轴组启动相应的运动控制命令后，轴组的状态机如下图所示。



GroupErrorStop	轴组的故障检测逻辑检测到故障时会进入此状态。
GroupStandby	• 当轴组处于 GroupDisabled 状态时调用 MC_GroupEnable • 当轴组 GroupMoving 完成时 • 当轴组完成 GroupStopping，且 Execute=FALSE 时 • 当因错误导致轴组停止时，调用 GroupReset
GroupDisabled	• 在错误停止 ErrorStop 状态下，如果任何轴的 MC_Power.Status 为 FALSE，那么在调用 MC_GroupReset 后，轴组可以直接返回到 GroupDisabled 状态 • 在轴组处于运动、停止或待机状态时调用 MC_GroupDisable 指令，轴组将立即停止并进入 GroupDisabled 状态
GroupMoving	当轴组处于 GroupStandby 状态时，调用轴组运动指令
GroupStopping	当轴组处于 GroupStandby 或 GroupMoving 状态时调用 MC_GroupStop，在执行 MC_GroupStop 后，轴组将进入 GroupStopping 状态

状态	说明
轴组已禁用	未使用轴组时的状态。
轴组已启用	使用轴组时的状态。

状态		说明
	GroupStandby	尚未执行的轴组指令的动作命令状态。
	GroupMoving	执行具有指定目标位置的定位的状态。 其中包括等待定位完成的状态。
	GroupStopping	通过 `MC_GroupStop` 指令使轴停止的状态。 该状态还包括轴已停止，但 MC_GroupStop 的 Execute 仍然为 TRUE 时。 此状态下，轴组指令无法执行操作。
	GroupErrorStop	轴组的故障检测逻辑检测到故障。 其中包括轴因异常情况而减速的状态。 此状态下无法执行轴运动指令。

14.6.3 缓冲模式

14.6.3.1 单轴命令缓冲模式

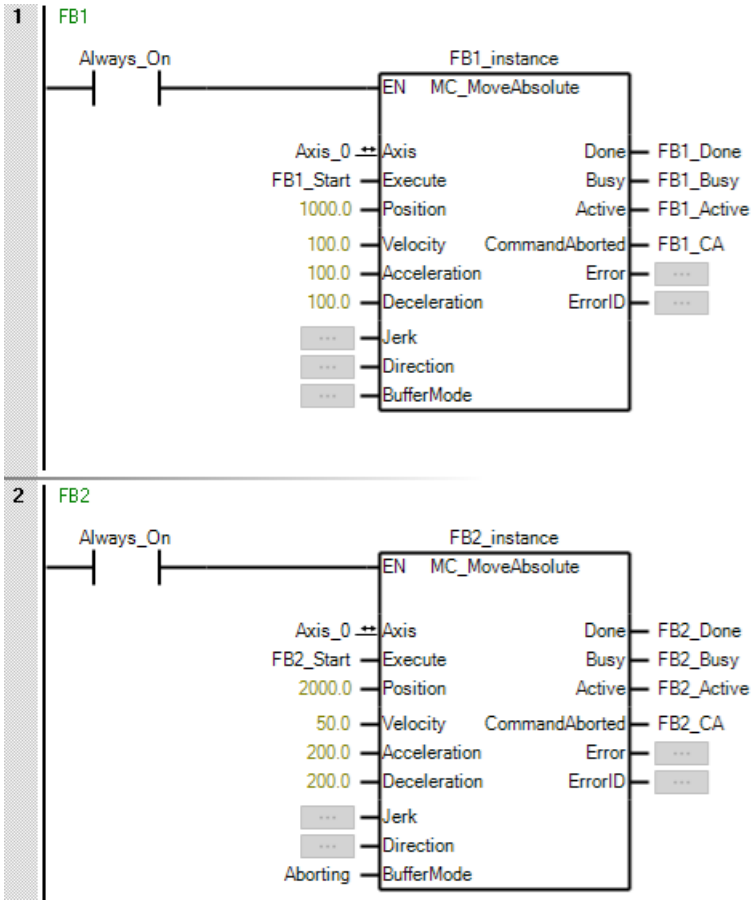
下表给出了单轴命令缓冲模式的概述。

S7-200 SMART 3.0 CPU 和 S7-200 SMART G2 CPU 最多可缓冲 8 条指令。

指令	是否可以指定为缓冲模式	是否可后接缓冲命令	激活下一个缓冲指令的相关信号
MC_Power	NO	NO	--
MC_Home	NO	NO	--
MC_Stop	NO	NO	--
MC_Halt	NO	YES	Done
MC_MoveAbsolute	YES	YES	Done
MC_MoveRelative	YES	YES	Done
MC_MoveVelocity	YES	YES	InVelocity
MC_CamIn	YES	YES	EndOfProfile
MC_CamOut	NO	NO	--
MC_GearIn	YES	Yes	InGear
MC_GearOut	NO	NO	--
MC_PhasingRelative	NO	NO	--

指令	是否可以指定为缓冲模式	是否可后接缓冲命令	激活下一个缓冲指令的相关信号
MC_ImmediateStop	NO	NO	--
MC_MoveJog	NO	NO	--

示例：Aborting 运动



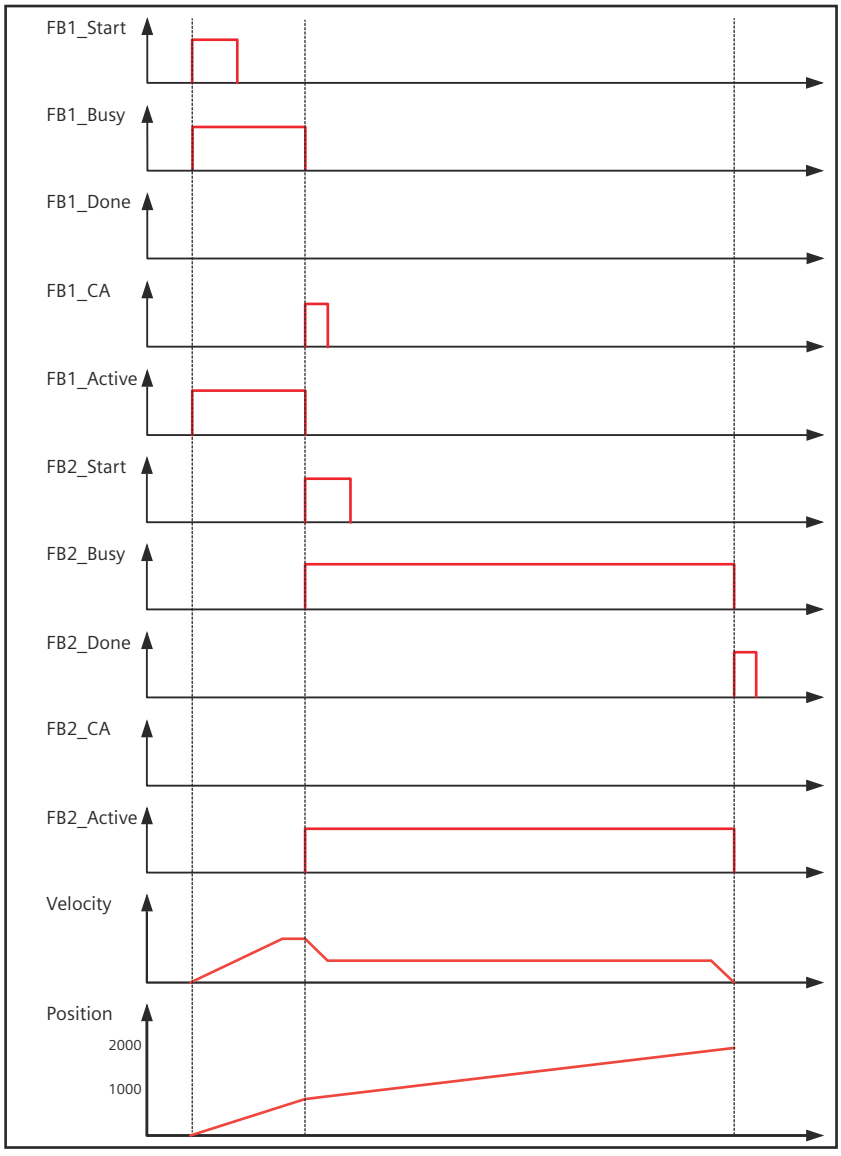
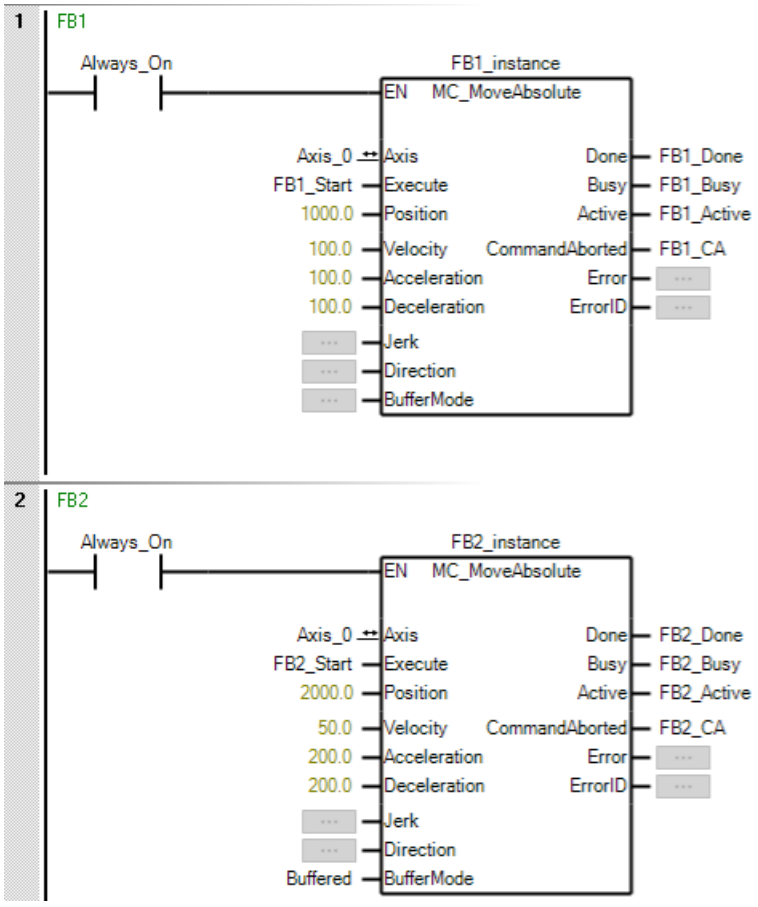


图 14-2　上例中 FB2 中断 FB1 的时序图（立即从断点执行 FB2）

示例：缓冲运动


1
FB1
Always_On
FB1_instance
EN
MC_MoveAbsolute
Axis_0
Axis
FB1_Start
Execute
1000.0
Position
100.0
Velocity
100.0
Acceleration
100.0
Deceleration
Jerk
Direction
BufferMode
Done
FB1_Done
Busy
FB1_Busy
Active
FB1_Active
CommandAborted
FB1_CA
Error
ErrorID
2
FB2
Always_On
FB2_instance
EN
MC_MoveAbsolute
Axis_0
Axis
FB2_Start
Execute
2000.0
Position
50.0
Velocity
200.0
Acceleration
200.0
Deceleration
Jerk
Direction
Buffered
BufferMode
Done
FB2_Done
Busy
FB2_Busy
Active
FB2_Active
CommandAborted
FB2_CA
Error
ErrorID

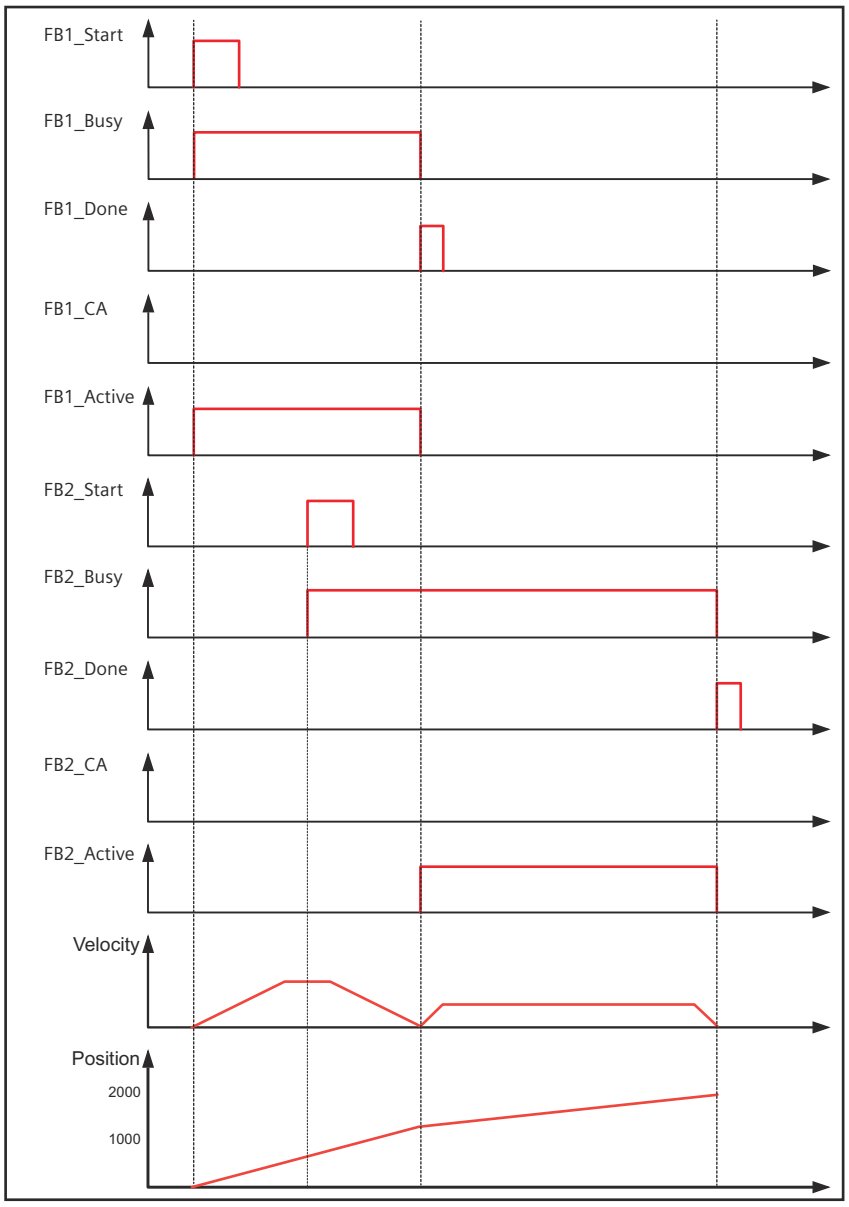


图 14-3　上例中 Buffered 模式下的时序图（无延迟地停止到速度 0 并启动 FB2）

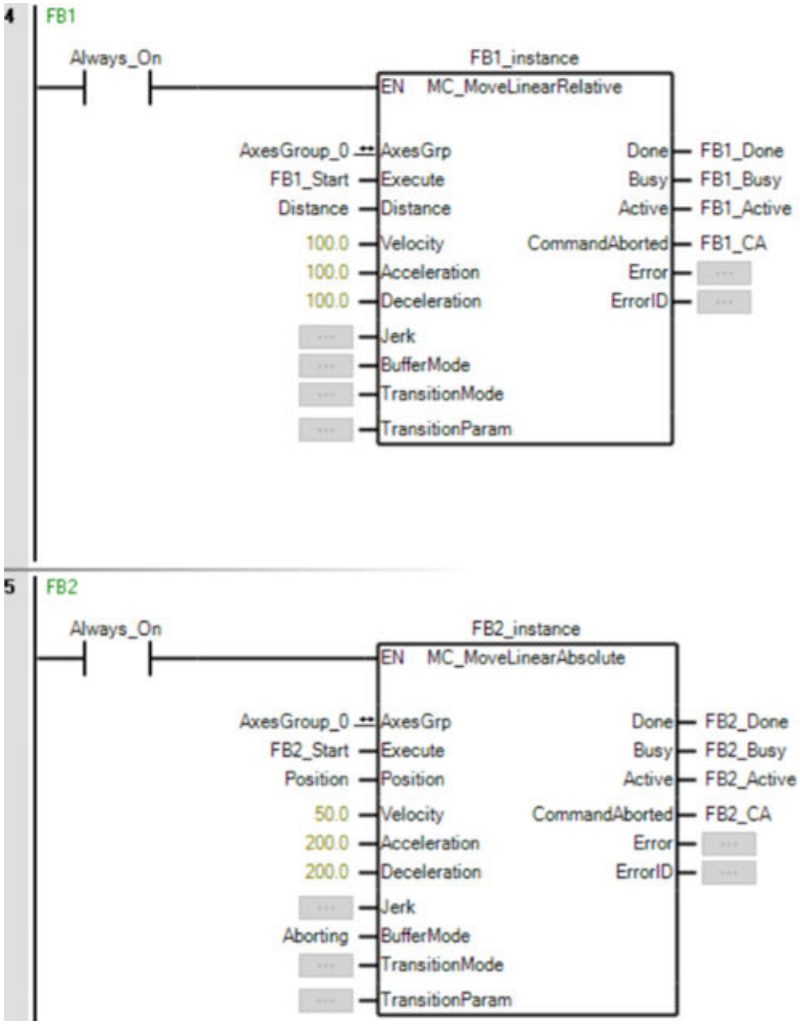
14.6.3.2　轴组命令缓冲模式

下表给出了轴组命令缓冲模式的概述。

S7-200 SMART 3.0.1 CPU 和 S7-200 SMART G2 CPU 最多可缓冲 8 条指令。

指令	是否可以指定为缓冲模式	是否可后接缓冲命令	激活下一个缓冲指令的相关信号
MC_GroupEnable	NO	NO	--
MC_GroupDisable	NO	NO	--
MC_GroupReset	NO	NO	--
MC_GroupStop	NO	NO	--
MC_MoveLinearAbsolute	YES	YES	Done
MC_MoveLinearRelative	YES	YES	Done

示例：Aborting 运动



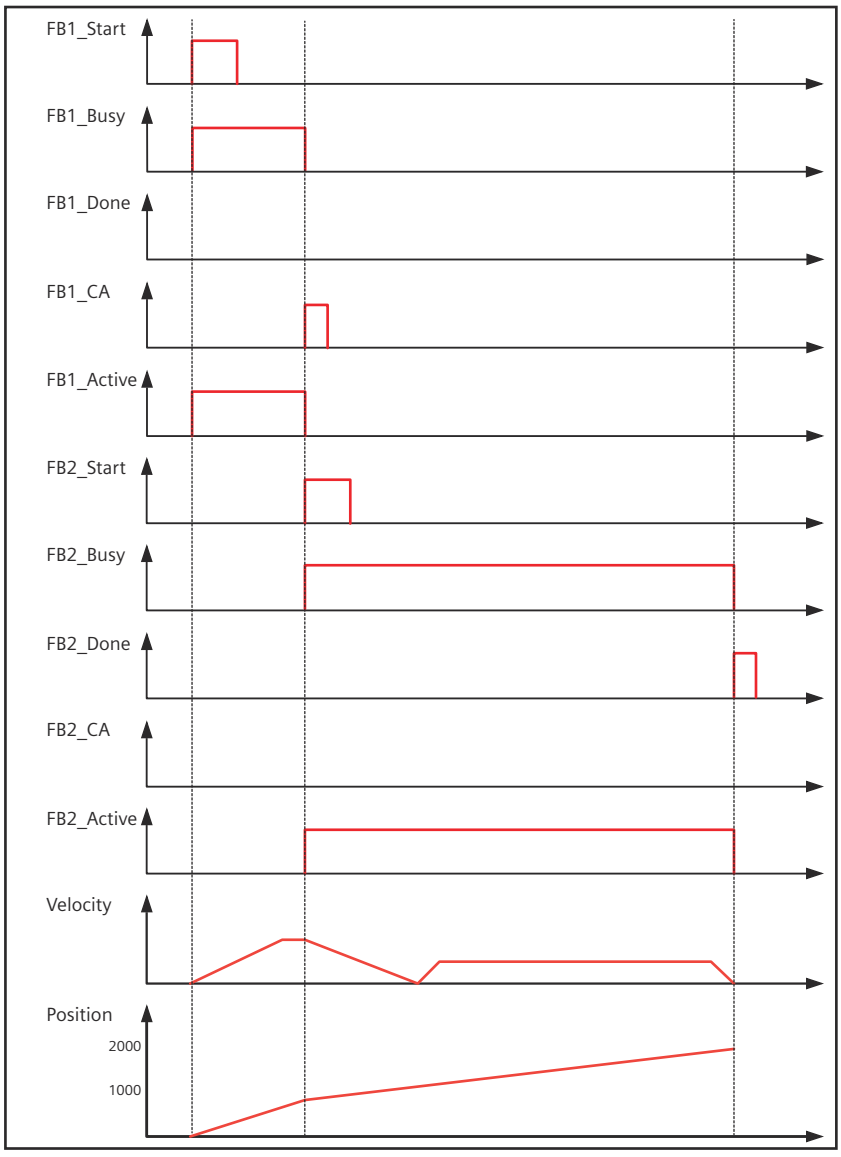
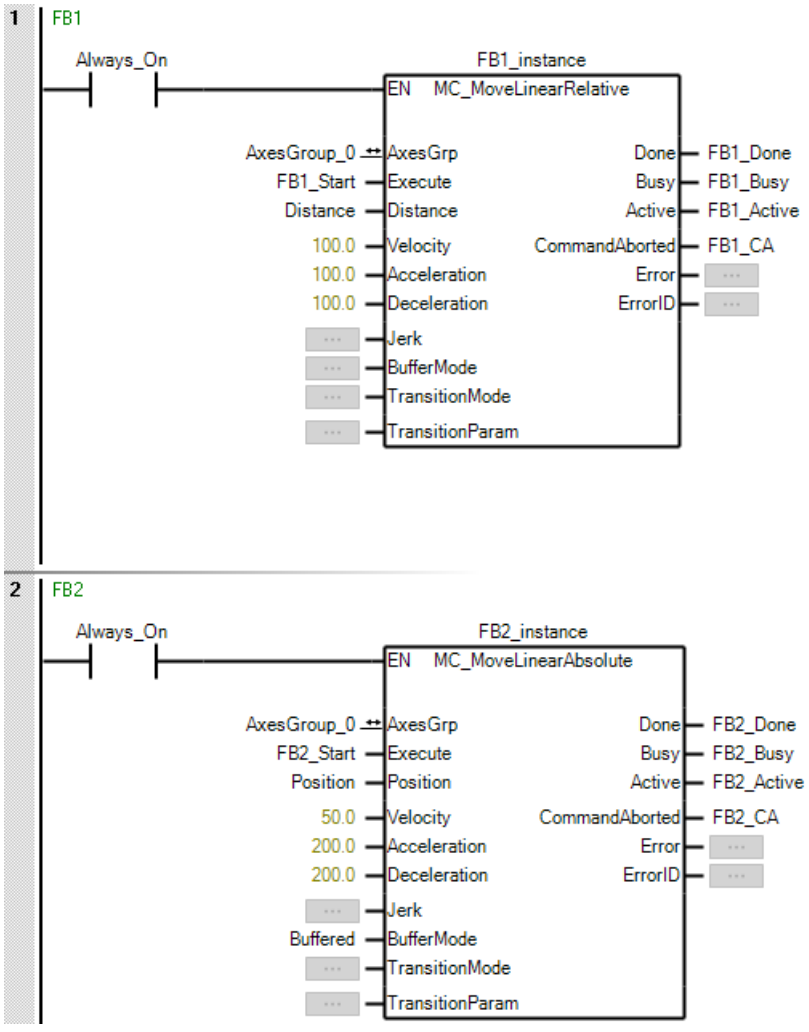


图 14-4　上例中 FB2 中断 FB1 的时序图（按 FB1 减速度停止后，再按 FB2 参数运行到目的坐标）

示例：缓冲运动


1
FB1
Always_On
FB1_instance
EN
MC_MoveLinearRelative
AxesGroup_0
AxesGrp
FB1_Start
Execute
Distance
Distance
100.0
Velocity
100.0
Acceleration
100.0
Deceleration
Jerk
BufferMode
TransitionMode
TransitionParam
Done
FB1_Done
Busy
FB1_Busy
Active
FB1_Active
CommandAborted
FB1_CA
Error
ErrorID
2
FB2
Always_On
FB2_instance
EN
MC_MoveLinearAbsolute
AxesGroup_0
AxesGrp
FB2_Start
Execute
Position
Position
50.0
Velocity
200.0
Acceleration
200.0
Deceleration
Jerk
Buffered
BufferMode
TransitionMode
TransitionParam
Done
FB2_Done
Busy
FB2_Busy
Active
FB2_Active
CommandAborted
FB2_CA
Error
ErrorID

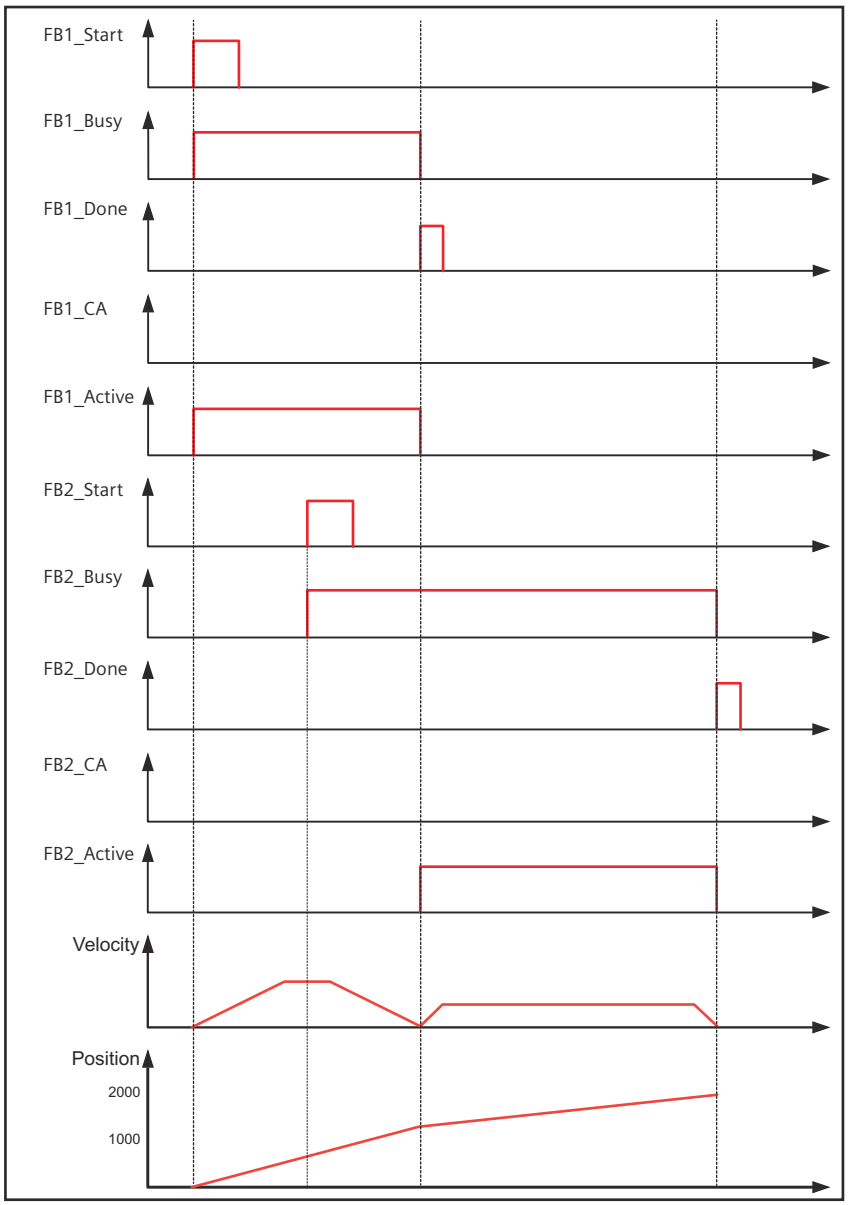


图 14-5　上例中 Buffered 模式下的时序图（无延迟地停止到速度 0 并启动 FB2）

14.6.4　变量概览

运动控制系统变量的类型

轴变量为系统定义的变量，由轴参数、当前位置和某些 MC 功能模块的错误信息等监视信息组成。通过 STEP 7-Micro/WIN SMART V3 创建轴时，轴变量按照创建的顺序注册在全局变量表中。

轴变量的数据类型为结构类型，而其数据名称为“轴_x”。

- 轴变量
  用于处理伺服驱动、虚拟伺服驱动、编码器的变量。
  – STEP 7-Micro/WIN SMART V3 创建的变量名称为：轴_0 ~ 轴_15

说明

使用 MC 类型数组设置 MC 指令

在添加轴时，系统会根据轴号将其分配到对应的轴数组中。例如，轴_1 对应的轴号为 1，该轴将被分配到 AXIS_ARRAY[1]。

使用“FOR”循环执行 MC 指令，确保使用 MC 类型数组设置这些指令。

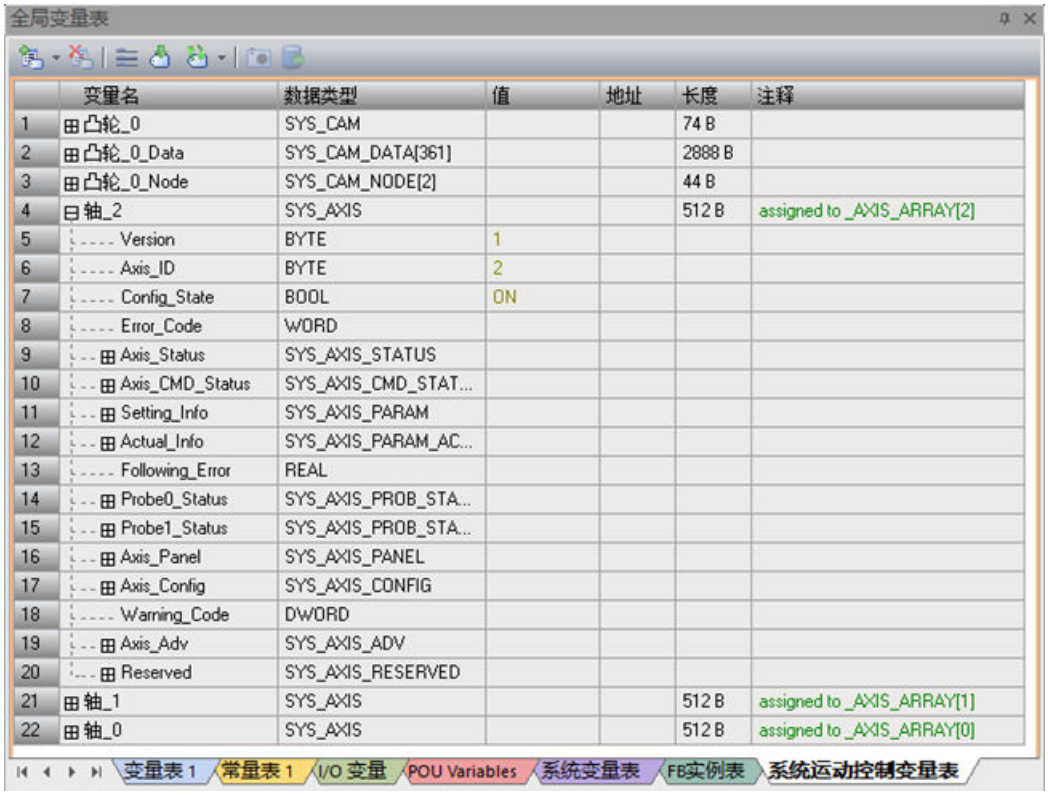
因为每个 MC 指令需与实例关联。如果不使用 MC 类型数组进行设置，程序则可能产生无法预测的结果。

要确保程序正确运行，确保使用 MC 类型数组设置 MC 指令。

- 凸轮变量
  用于处理主轴和从轴位置之间关系的变量。
  – STEP 7-Micro/WIN SMART V3 创建的变量名称为：凸轮_0 ~ 凸轮_7，凸轮_0_Data ~ 凸轮_7_Date，凸轮_0_Node ~ 凸轮_7_Node

在程序中，可以使用 STEP 7-Micro/WIN SMART V3 创建的系统定义的变量名。

运动控制变量表示例



全局变量表

	变量名	数据类型	值	地址	长度	注释
1	凸轮_0	SYS_CAM			74 B	
2	凸轮_0_Data	SYS_CAM_DATA[361]			2888 B	
3	凸轮_0_Node	SYS_CAM_NODE[2]			44 B	
4	轴_2	SYS_AXIS			512 B	assigned to _AXIS_ARRAY[2]
5	Version	BYTE	1			
6	Axis_ID	BYTE	2			
7	Config_State	BOOL	ON			
8	Error_Code	WORD				
9	Axis_Status	SYS_AXIS_STATUS				
10	Axis_CMD_Status	SYS_AXIS_CMD_STAT...				
11	Setting_Info	SYS_AXIS_PARAM				
12	Actual_Info	SYS_AXIS_PARAM_AC...				
13	Following_Error	REAL				
14	Probe0_Status	SYS_AXIS_PROB_STA...				
15	Probe1_Status	SYS_AXIS_PROB_STA...				
16	Axis_Panel	SYS_AXIS_PANEL				
17	Axis_Config	SYS_AXIS_CONFIG				
18	Warning_Code	DWORD				
19	Axis_Adv	SYS_AXIS_ADV				
20	Reserved	SYS_AXIS_RESERVED				
21	轴_1	SYS_AXIS			512 B	assigned to _AXIS_ARRAY[1]
22	轴_0	SYS_AXIS			512 B	assigned to _AXIS_ARRAY[0]

变量表 1 | 常量表 1 | I/O 变量 | POU Variables | 系统变量表 | FB实例表 | 系统运动控制变量表

运动控制系统变量属性

运动控制系统变量均具有以下特征：

属性	运动控制系统变量的属性
全局或本地	全局
读写权限	Cam_Node 和 Cam_Data 可读写； 其它参数为只读
保持	否

14.6.5 变量一览表

运动控制变量为系统定义的变量。在用户程序中使用这些变量时，将以系统定义的变量名（AX_x、凸轮_x、凸轮_x_Data、凸轮_x_Node）对其进行引用。

如果更改了轴名称，例如将“Axis_0”改为“My_Axis_0”，则此变量的名称在全局变量表中也会相应更改。

变量			数据类型	说明
Version			BYTE	TO 的版本号
Axis_ID			BYTE	轴号
Config_State			BOOL	轴配置状态
Error_Code			WORD	轴错误码
Axis_Status.			SYS_AXIS_STATUS	
	ENC_Enable		BOOL	仅供编码器轴使用 • 0：未使能 • 1：使能
	Error		BOOL	因 PN 通信错误/运动错误/配置错误时变为 TRUE。
	ControlPanel Active		BOOL	表示是否激活控制面板 • 0：未激活 • 1：已激活
	Disabled		BOOL	轴未使能且无故障变为 TRUE。
	Standstill		BOOL	变为 TRUE 时表示： • 轴已使能 • 轴无故障 • 轴上未激活任何运动指令
	Discrete		BOOL	正在执行离散运动时变为 TRUE。
	Continuous		BOOL	正在执行连续运动时变为 TRUE
	Synchronized		BOOL	执行同步运动时变为 TRUE。
	Homing		BOOL	执行原点复位时变为 TRUE
	Stopping		BOOL	正在减速停止时变为 TRUE。
	ErrorStop		BOOL	在系统检测到轴相关错误时变为 TRUE。在此状态下，轴可能处于启用或禁用状态，并可通过 MC_Power 指令进行控制。然而，只要错误未被清除，轴将持续保持在 ErrorStop 状态。 ErrorStop 状态的核心目的是使轴尽可能安全地停止运动。一旦进入此状态，系统将不接受任何后续的运动指令，直至成功执行复位操作。

变量			数据类型	说明
	Homed		BOOL	在原点确定, 且已执行过了 MC_home 指令时变为 TRUE。
	LMTpHW_State		BOOL	到达正向硬限位时变为 TRUE。
	LMTmHW_State		BOOL	到达负向硬限位时变为 TRUE。
	LMTpSW_State		BOOL	到达正向软限位时变为 TRUE。
	LMTmSW_State		BOOL	到达负向软限位时变为 TRUE。
	RPS_State		BOOL	到达原点搜索信号变为 TRUE。
	STP_State		BOOL	到达急停信号变为 TRUE（Only for PTO 轴）
	RefBlock_State		BOOL	是否已到达参考挡块 • 0：未到达 • 1：已到达
	DIInput_State		BOOL	是否已到达数字量输入点 • 0：未到达 • 1：已到达
	InClamping		BOOL	驱动器保持在固定挡块位置处（夹紧）。轴位置位于定位容差范围内。
	VelLimit		BOOL	同步过程中从轴超过其设置的最大速度
	Reserved4		BOOL	预留
	Reserved5		BOOL	预留
	Drive_Ready		BOOL	变为 TRUE 时表示： • 控制器与驱动器之间的周期性通讯已建立 • 驱动器没有错误 • 驱动器配置正确
	Drive_Err		BOOL	发生驱动器错误时变为 TRUE。
	Encoder_Err		BOOL	发生编码器错误时变为 TRUE。
	Data_Exchange_Err		BOOL	运行中 PN 断联或驱动器 SOL 连续超时变为 TRUE

变量			数据类型	说明
	Config_Err		BOOL	当满足以下复合条件时，轴将进入错误状态（变为 TRUE），并等待 PLC 重新下载配置： • 驱动器参数读取操作失败 • 系统在三次重试后仍无法成功读取参数 • MW 未配置离线参数
	Following_Err_Exceed		BOOL	跟随误差超出预设阈值时变为 TRUE。
	Following_Err_Warning		BOOL	当跟随误差超出预设阈值时，系统将触发超差预警。此预警的触发条件与系统配置相关。在此情况下，轴不会进入错误状态，也不会置 Error 位。
	Config_Warning		BOOL	当检测到驱动器的在线参数与离线参数不一致时变为 TRUE，系统不会置 Error 位。
Axis_CMD_Status.			SYS_AXIS_CMD_STATUS	
	Power_CMD		BOOL	正在执行 MC_Power 指令变为 TRUE。
	MoveVelocity_CMD		BOOL	正在执行 MC_MoveVelocity 指令时变为 TRUE。
	MoveRelative_CMD		BOOL	正在执行 MC_MoveRelative 指令时变为 TRUE。
	MoveAbsolute_CMD		BOOL	正在执行 MC_MoveAbsolute 指令时变为 TRUE。
	Reset_CMD		BOOL	正在执行 MC_Reset 指令时变为 TRUE。
	Home_CMD		BOOL	正在执行 MC_Home 指令时变为 TRUE。
	Stop_CMD		BOOL	正在执行 MC_Stop 指令时变为 TRUE
	Halt_CMD		BOOL	正在执行 MC_Halt 指令时变为 TRUE。
	ImmediateStop_CMD		BOOL	正在执行 MC_ImmediateStop 指令时变为 TRUE。
	Reserved_CMD		BOOL	预留
	TouchProbe_CMD		BOOL	正在执行 MC_TouchProbe 指令时变为 TRUE。
	AbortingTrigger_CMD		BOOL	正在执行 MC_AbortingTrigger 指令时变为 TRUE。
	GearIn_CMD		BOOL	正在执行 MC_GearIn 指令时变为 TRUE。

变量			数据类型	说明
	GearOut_CMD		BOOL	正在执行 MC_GearOut 指令时变为 TRUE。
	MoveJog_CMD		BOOL	正在执行 MC_MoveJog 指令时变为 TRUE
	CamIn_CMD		BOOL	正在执行 MC_CamIn 指令时变为 TRUE。
	CamOut_CMD		BOOL	正在执行 MC_CamOut 指令时变为 TRUE。
	PhasingRelative_CMD		BOOL	正在执行 MC_PhasingRelative 指令时变为 TRUE
	TorqueRange_CMD		BOOL	正在执行 MC_TorqueRange 指令时变为 TRUE。
	TorqueControl_CMD		BOOL	正在执行 MC_TorqueControl 指令时变为 TRUE
	TorqueLimiting_CMD		BOOL	正在执行 MC_TorqueLimiting 指令时变为 TRUE
	Reserved		BOOL	预留
Setting_Info.			SYS_AXIS_PARAM	
	Pos		REAL	位置设定值
	Vel		REAL	速度设定值
	Acc		REAL	加速度设定值
	Decel		REAL	减速度设定值
	Jerk		REAL	加加速度设定值
	Reserved		REAL	预留
	Torque		REAL	力矩设定值
Actual_Info.			SYS_AXIS_PARAM_ACTUAL	
	Pos		REAL	实际位置
	Vel		REAL	实际速度
	Acc		REAL	实际加速度
	Decel		REAL	实际减速度
	Reserved0		REAL	预留
	Torque		REAL	实际力矩

变量			数据类型	说明
	Reserved1		REAL	预留
Following_Error			REAL	当前轴的跟随误差
Probe0.Status.			SYS_AXIS_PROB_STATUS	
	Enable		BOOL	是否使用探针 0 • 0：未使用探针 • 1：使用探针
	RisingEdgeDone		BOOL	
	FallingEdgeDone		BOOL	
Probe1.Status.			SYS_AXIS_PROB_STATUS	
	Enable		BOOL	是否使用探针 1 • 0：未使用探针 • 1：使用探针
	RisingEdgeDone		BOOL	
	FallingEdgeDone		BOOL	
Axis_Panel.			SYS_AXIS_PANEL	
	Target_Pos		REAL	位置设定值
	Target_Vel		REAL	速度设定值
	Target_Acc		REAL	加速度设定值
	Target_Dir		BYTE	方向设定值
	Home_Rps_Offset		REAL	回零偏移
	Error_Code		WORD	控制面板运动错误
	Reserved0		REAL	预留
	Reserved1		DWORD	预留
Axis_Config.				
	Name		STRING	轴名称

变量			数据类型	说明
	Axis_Type		BYTE	轴类型 • 虚轴：0 • PTO 轴：1 • 编码器轴：2 • PN IRT 轴：3
	Engineer_Sys		BYTE	• 厘米： 0 • 毫米：1 • 英寸：2 • 度：3 • 脉冲：4
	Pulse_Of_OneMotor		DWORD	PLS/REV 电机/编码器旋转一圈的脉冲数
	Units_Revolution		REAL	工作台旋转一圈的移动量
	Units		STRING	单位
	Gear_Numerator		DWORD	减速比分子
	Gear_Denominator		DWORD	减速比分母
	Linear_Rotary		BYTE	• 旋转: 0 • 线性: 1
	Modulo_StartVal		REAL	模数起始值
	Modulo_Length		REAL	模数长度
	Reserved0		REAL	预留
	PositionLmt_SW.		SYS_AXIS_SOFTLMT	
		SwLimit_En	BOOL	软件限位启用 • 0：未启用软件限位 • 1：启用软件限位
		MaxLimit_Resp	BYTE	软件正限位响应（减速停/立即停模式）

变量			数据类型	说明
		MinLimit_Resp	BYTE	软件负限位响应（减速停/立即停模式）
		MaxLimit_Pos	REAL	软件正限位位置
		MinLimit_Pos	REAL	软件负限位位置
	Dynamic_Default.		SYS_AXIS_DYNAMIC_DEFAULT	
		Default_Acc	REAL	默认加速度
		Default_Decel	REAL	默认减速度
		Default_Jerk	REAL	默认加加速度
		Reserved0	REAL	预留
		Reserved1	REAL	预留
	Dynamic_Limit.		SYS_AXIS_DYNAMIC_LIMIT	
		Max_Vel	REAL	最大速度
		Max_Acc	REAL	最大加速度
		Max_Decel	REAL	最大减速度
		Reserved0	REAL	预留
		Reserved1	REAL	预留
	Reserved1		REAL	预留
	Reserved2		REAL	预留
	Reserved3		REAL	预留
	Reserved4		REAL	预留
Reserved			DWORD	
Axis_Config_Adv.			SYS_AXIS_CONFIG_ADV	扩展参数
	Pos_Monitoring.		SYS_AXIS_POS_MONITOR	

变量			数据类型	说明
		Tolerance_Time	REAL	**定位窗口容差时间** 容差时间为从轴的速度达到设定值 0 起，至其实际位置进入定位窗口内的最大允许持续时间。 从达到零设定速度到进入定位窗口的最大允许时间间隔。
		MinDwell_Time	REAL	在定位窗口停留的最短时间
		Window	REAL	定位窗口
		Reserved0	REAL	预留
	Following_Err.		SYS_AXIS_FOLLOWING_ERR	
		Enable	BOOL	• 跟随误差监控未启用: 0 • 跟随误差监控已启用: 1
		Reserved1	BYTE	
		Min_Value	REAL	在速度低于 Dynamic_Vel 值时允许的跟随误差。
		Max_Value	REAL	在达到最大速度时允许的最大跟随误差。
		Dynamic_Vel	REAL	Min_Value 在速度低于该值时允许使用，并保持恒定值。
		Warning_Level	REAL	百分比值形式的警告限值，与最大允许跟随误差相关（与特性曲线速度相关）。
		Reserved2	REAL	
	Standstill_Signal.		SYS_AXIS_STANDSTILL	
		Vel_Threshold	REAL	停留时间启动速度阈值 如果速度低于此阈值，则最短停留时间开始。
		MinDwell_Time	REAL	指定最短停留时间
		Tolerance_Time	REAL	预留
	Pos_Ctrl.		SYS_AXIS_POS_CTRL	
		Kv	REAL	闭环位置控制的比例增益("Kv"> 0.0)

变量			数据类型	说明
		Kpc	REAL	位置控制的速度预控制，推荐的设置： • 通过 PROFIdrive 的等时同步驱动器接口：100.0% • 通过 PROFIdrive 的非等时同步驱动器接口：0.0 到 100.0% • 模拟驱动器接口：0.0 到 100.0%
		Vtc	REAL	
	Extrapolation.		SYS_AXIS_EXTRAPOLATION	
		Differentiation	BOOL	激活微分 • 选中此复选框时：主值速度由外推主值位置差异得出。 • 清除此复选框时：将应用滤波后的实际速度。
		Leading_Axis	BOOL	考虑引导轴 • 选中此复选框时：与引导轴相关的外推时间将被计入有效外推时间的计算。 • 清除此复选框时：有效外推时间的计算将不包括与引导轴相关的外推时间。
		Reserved4	BYTE	预留
		Pos_FilterT1	REAL	位置过滤器时间常数 T1
		Pos_FilterT2	REAL	位置过滤器时间常数 T2
		Vel_FilterT1	REAL	速度过滤器时间常数 T1
		Vel_FilterT2	REAL	速度过滤器时间常数 T2
		Vel_Tolerance	REAL	速度的容差带宽
		Hysteresis	REAL	外推实际位置值滞后
		Reserved5	REAL	预留
	Torque_Lmt		SYS_AXIS_TORQUELMT	
		PosLmt_En	BOOL	• 0: 禁用位置相关监视 • 1: 保持位置相关的监视启用
		Reserved6	BYTE	预留
		Torque_Force	REAL	扭矩限制百分比

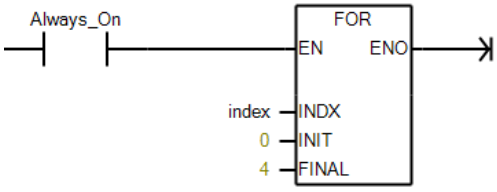
变量			数据类型	说明
		Reserved7	BYTE	预留
	Clamping		SYS_AXIS_CLAMPING	
		Pos_Tolerance	REAL	定位容差
		FollowingErr_Dev	REAL	跟随误差
	Ctrl_Mode		SYS_AXIS_CTRL_MODE	
		Pos_Ctrl_Mode	BOOL	
		Speed_Ctrl_Mode	BOOL	
		Torque_Ctrl_Mode	BOOL	

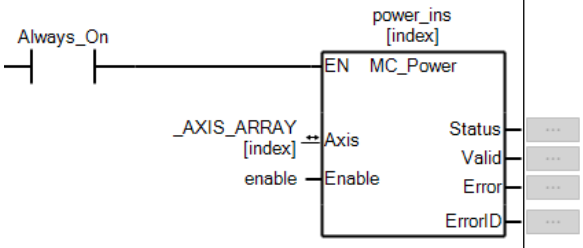
14.6.6 使用数组实现多轴控制

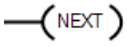
可通过调用数组实现多轴控制。

下方示例中定义了五个轴，将使用 FOR 循环和数组启用这五个轴。_AXIS_ARRAY[0...15] 对应从 0 到 15 的轴 ID。

表格 14-1　示例：启用轴 ID 为 0 到 4 的轴

LAD/FBD	说明	STL
程序段 1：  	使用 for 循环控制多个轴	`LD Always_On` `FOR index, 0, 4`

程序段 2：
 
 	
	启用多个轴	`LD Always_On`
		
		`FBCALL MC_Power,`
		`power_ins[index],`
		`_AXIS_ARRAY[index],`
		`enable,`
		`power_ins[index].Status,`
		`power_ins[index].Valid,`
		`power_ins[index].Error,`
		`power_ins[index].ErrorID`

程序段 3：  		

14.7 运动控制指令

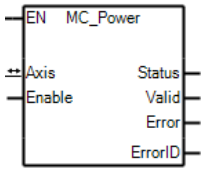
14.7.1 基本指令

14.7.1.1 MC_Power

MC_Power 指令主要用于控制轴的使能与禁用。它是所有轴运动操作的必要前提，只有在 MC_Power 成功使能轴后，其他运动控制指令（例如 MC_MoveAbsolute、MC_MoveRelative 等）才能正常工作。

应根据 MC_Power 的状态来触发后续运动指令。

参数

LAD/FBD	STL	说明
	**FBCALL MC_Power, INSTANCE, Axis, Enable, Status, Valid, Error, ErrorID**	将轴切换至工作状态

参数	声明	数据类型	初始值	说明
EN	IN	BOOL	-	能流
Axis	IN_OUT	SYS_AXIS	-	引用轴
Enable	IN	BOOL	-	• TRUE: 使能 • FALSE: 未使能
Status	OUT	BOOL	False	进入工作状态时变为 TRUE
Valid	OUT	BOOL	False	TRUE: MC_Power FB 实例已成功绑定到轴
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1068)

说明

轴处于错误停止状态时禁用此轴会使轴保持在错误停止状态。

超驰响应

- 运动控制指令无法中止“MC_Power”的执行（包括“启用”和“禁用”）
- 禁用轴（输入参数“Enable”= FALSE）会立即中止关联轴的所有运动控制指令

前置条件

说明

单实例控制

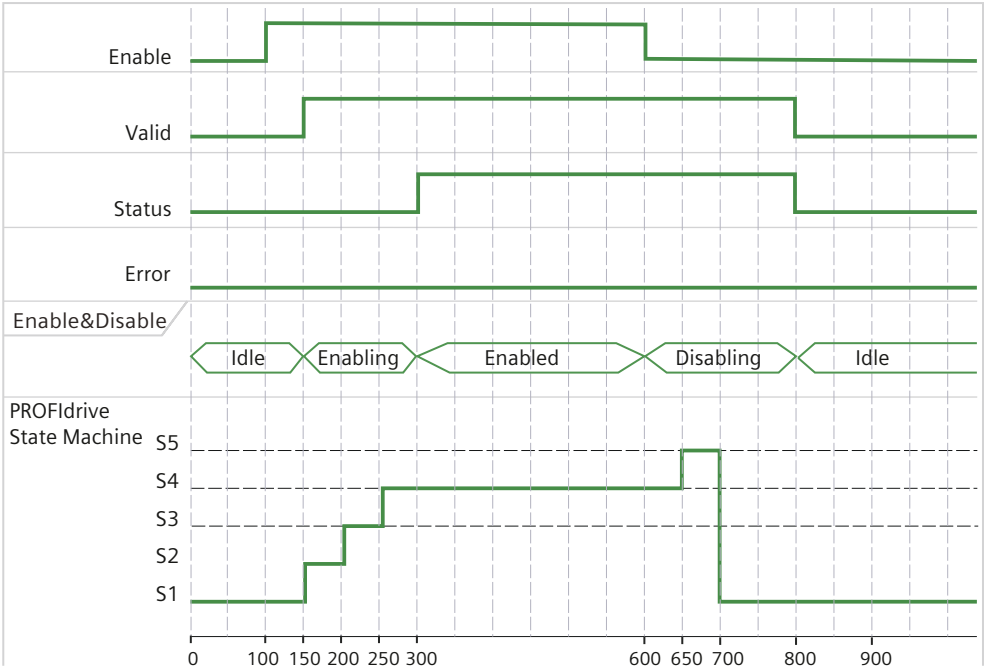
每个轴仅允许由一个 MC_Power FB 实例使能。如果一个轴已被某个 MC_Power FB 实例成功使能，则无法使用另一个 MC_Power FB 实例来使能该轴。

PN IRT 轴须满足以下条件

- 控制器与驱动器之间已建立通信。
- 轴必须已正确组态， 轴状态处于"Drive Ready"。

启用轴

当 Enable 参数从 FALSE 切换为 TRUE 时，MC_Power 会首先执行一系列内部检查，例如验证驱动器状态和通信状态，以确保操作环境已准备就绪。如果所有检查均通过，MC_Power 将向驱动器发送使能信号。此时，Valid 输出会变为 TRUE，在轴成功使能后，Status 输出将变为 TRUE。



S1 禁止上电： （驱动器关闭，必要时制动器闭合)

S2 准备上电： （驱动器已准备好接收电源）

S3 已上电： （驱动器已上电，必要时制动器释放）

S4 运行中： （驱动器已释放，制动器已释放）

S5 正在关断： （驱动器按照预定义斜坡进行制动）

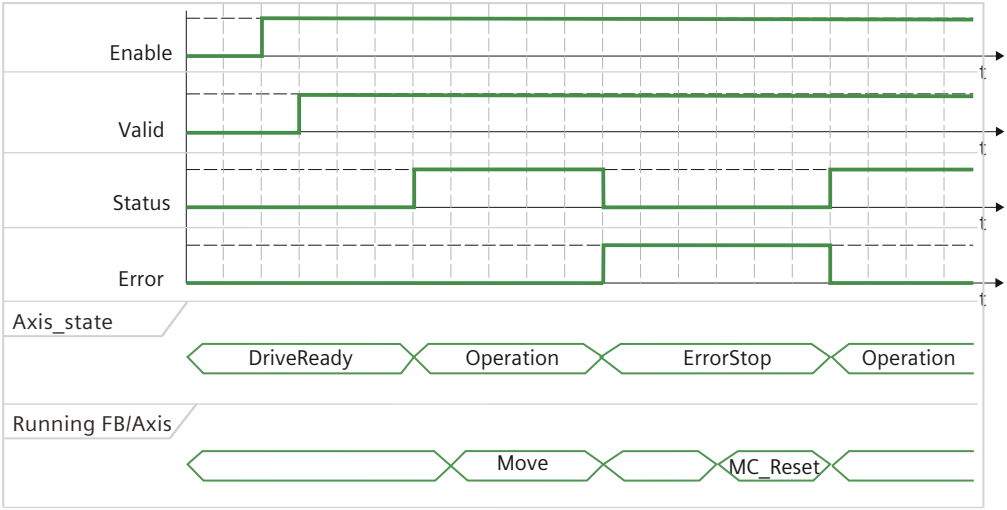
禁用轴

如果轴没有发生任何错误，MC_Power 的 Status 输出将保持为 TRUE，表示轴处于正常使能状态。

一旦检测到轴对象发生错误，MC_Power FB 实例会立即更新其输出：

- Status 输出将变为 FALSE。
- Error 输出将变为 TRUE。

要清除 MC_Power 中的错误状态并使轴能够重新进入使能状态，需要使用 MC_Reset (页 961)。



14.7.1.2 MC_TouchProbe

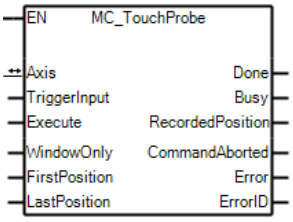
MC_TouchProbe 指令用于在探针触发时，捕获并记录轴的当前位置。

说明

对于 PN IRT 轴，TouchProbe 仅支持驱动器模式，由驱动器执行探针位置检测，须在驱动器端组态探针。而 PTO 和编码器仅支持控制器模式，直接读取 PLC 接收到的总线周期反馈位置。

在配置西门子伺服驱动器 S200 时，请查阅 S200 的操作说明，以获取关于探针功能的详细组态指导。

参数

LAD/FBD	STL	说明
	**FBCALL MC_TouchProbe, INSTANCE, Axis, TriggerInput, Execute, WindowOnly, FirstPosition,LastPositio n,Done, Busy, RecordedPosition, CommandAborted, Error, ErrorID**	该指令可组态接触式探针。

参数	声明	数据类型	初始值	说明
Axis	IN_OUT	SYS_AXIS	-	引用轴
TriggerInput	IN	MC_TRIGGER_REF	-	引用触发信号源
Execute	IN	BOOL	-	上升沿触发该指令
WindowOnly	IN	BOOL	False	如果设置为 True，则仅在窗口期内（如下所定义）的触发事件有效。
FirstPosition	IN	REAL	0.0	接受触发事件的起始位置（正方向）， 值包含在窗口中
LastPosition	IN	REAL	0.0	窗口的停止位置，值包含在窗口中
Done	OUT	BOOL	False	触发事件已记录
Busy	OUT	BOOL	False	指令正在运行中[1]
RecordedPosition	OUT	REAL	0.0	捕获事件触发的位置
CommandAborted	OUT	BOOL	False	当前指令被另一个指令打断
Error	OUT	BOOL	False	指令发生了错误
ErrorID	OUT	WORD	0	错误代码 (页 1068)

[1] 当 MC_TouchProbe 指令处于 Busy 状态时，即使轴已超出 [FirstPosition, LastPosition] 限定范围且探针未被触发，该指令仍将继续检测，而不会报告错误。

说明

使用 MC_TouchProbe 命令

- 如果两个 MC_TouchProbe 命令具有相同的轴编号和探针 ID，则后一个命令报错退出，前一个命令执行不受影响。
- PTO 和编码器需组态探针。每个单轴范围为 [I0.0~I1.7]。这些探针不得与 HSC、INT、Limit、STP、RPS、ZP 或任何其它正在使用的探针输入冲突。
- 多个轴同时执行探针命令时，它们彼此互不干扰。

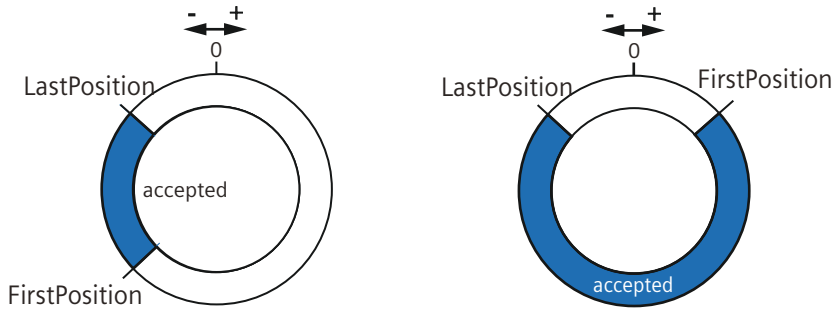
说明

当 MC_TouchProbe 指令的 Execute 引脚被置高时，如果在此期间更改了轴 ID，则重新触发该指令将针对新的轴 ID 执行探针功能。

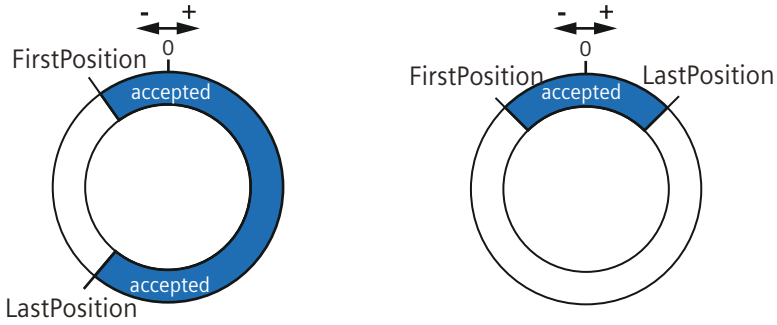
线性轴和模态轴

- 线性轴： 仅当轴的当前位置处于 FirstPosition 和 LastPosition 定义的区间内时，TouchProbe 功能才会检测探针信号并输出有效结果。
- 模态轴： FirstPosition 和 LastPosition 将对循环周期进行求模，以确定在一个循环周期内的有效检测区间。

A. FirstPosition < LastPosition



B. FirstPosition > LastPosition

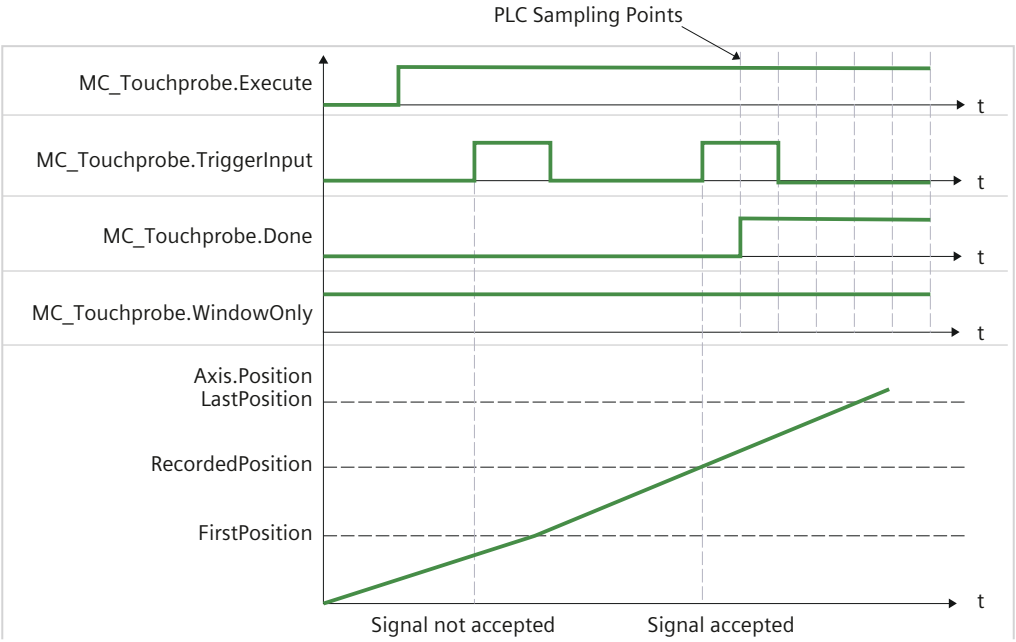


前置条件

- 轴已正确组态。
- 轴已使能并已回零。
- 对于 PN IRT 轴，驱动器的探针功能配置成功。
- 必须正确组态 TriggerInput。先在变量表中为 TriggerInput 添加一个变量，然后将其数据类型设置为 MC_TRIGGER_REF。

MC_TRIGGER_REF 结构要素变量	说明
ProbeID	• 关联 ProbeID = 0：已关联轴组态 probe0 • 关联 ProbeID = 1：已关联轴组态 probe1
TriggerEdge	• TriggerEdge = 0：探针的上升沿有效 • TriggerEdge = 1：探针的下降沿有效

执行流程



超驰响应

- MC_TouchProbe 指令可由以下运动控制指令中止：
  MC_ AbortTrigger
- MC_TouchProbe 指令无法由任何运动控制指令缓冲。
- 新的 MC_TouchProbe 指令不会中止或缓冲任何其他激活的运动控制指令。

14.7.1.3 MC_AbortTrigger

MC_AbortTrigger 指令取消或中止一个已激活的 MC_TouchProbe 操作。

- IRT 轴: 由于底层协议定义的 AbortTrigger 命令针对指定轴上的所有 TouchProbe 实例生效，而非单个，因此 MC_AbortTrigger 会终止该轴上所有当前激活的 TouchProbe 操作。
- PTO 轴: PTO 的 AbortTrigger 仅中断指定轴上具有特定 probeID 探针。

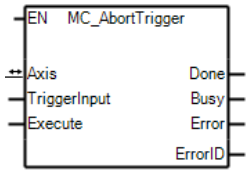
说明

如果 MC_TouchProbe 和 MC_AbortTrigger 同时在下一个 IPO 之前被使能，按照下面的执行流程: 如果 MC_AbortTrigger 先执行，就报错；如果 MC_TouchProbe 先执行，MC_AbortTrigger 就取消 MC_TouchProbe 操作。

说明

MC_TouchProbe 不支持连续触发模式，仅支持单次触发。一旦探针被成功检测到，MC_TouchProbe 功能块的 Done 位将置为 TRUE。

参数

LAD/FBD	STL	说明
	**FBCALL MC_AbortTrigger, INSTANCE, Axis, TriggerInput, Execute,Done, Busy, Error, ErrorID**	此指令用于中断连接到触发事件的指令（例如 MC_TouchProbe）。

参数	声明	数据类型	初始值	说明
EN	IN	BOOL	-	能流
Axis	IN_OUT	SYS_AXIS	-	引用轴

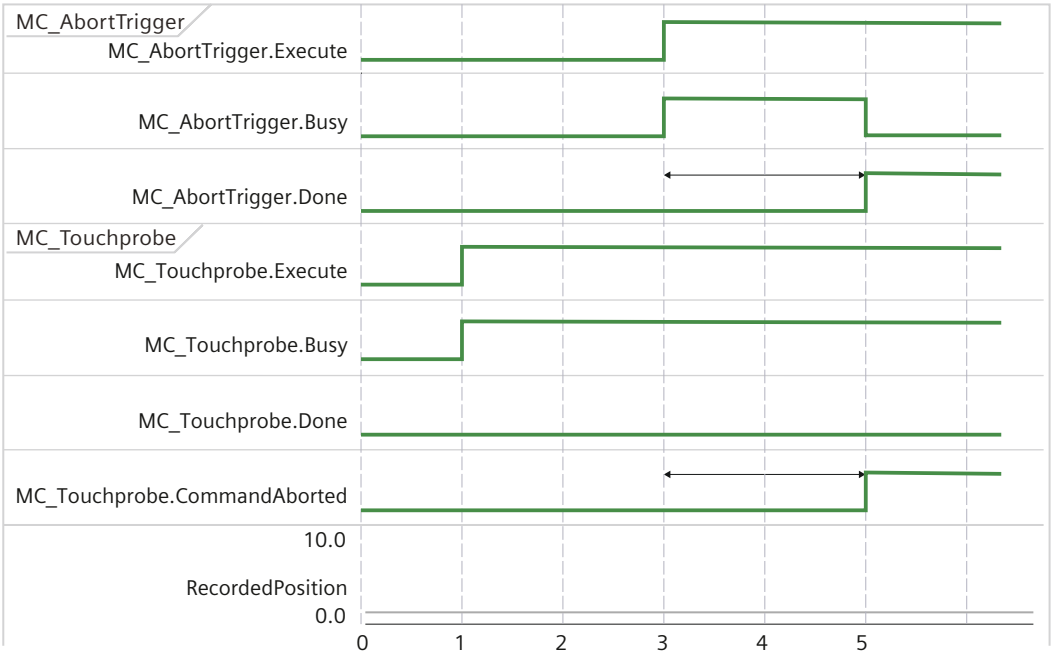
参数	声明	数据类型	初始值	说明
TriggerInput	IN_OUT	MC_TRIGGER_REF	-	引用触发信号源
Execute	IN	BOOL	-	上升沿触发该指令
Done	OUT	BOOL	False	已中止触发器功能
Busy	OUT	BOOL	False	指令正在运行中
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1068)

前置条件

- 轴已正确组态。
- 驱动器的探针功能配置成功。
- 轴上正在运行 MC_TouchProbe 指令。

执行流程

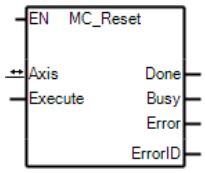
即使 MC_AbortTrigger 的 Execute 输入端被置位以中断 MC_TouchProbe，MC_TouchProbe 的 CommandAborted 输出端也不会立即变为高电平。



14.7.1.4 MC_Reset

MC_Reset 的主要功能是清除轴的错误状态。

参数

LAD/FBD	STL	说明
	`FBCALL MC_Reset, INSTANCE, Axis, Execute, Done, Busy, Error, ErrorID`	该指令通过复位所有内部轴相关错误，从 **ErrorStop** 状态转换为 **Standstill** 或 **Disabled** - 它不会影响功能块实例的输出。

参数	声明	数据类型	初始值	说明
EN	IN	BOOL	-	能流
Axis	IN_OUT	SYS_AXIS	-	引用轴
Execute	IN	BOOL	-	上升沿触发该指令
Done	OUT	BOOL	False	作业已经结束
Busy	OUT	BOOL	False	指令正在运行中
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1068)

前置条件

- 轴已正确组态。

重新启动轴

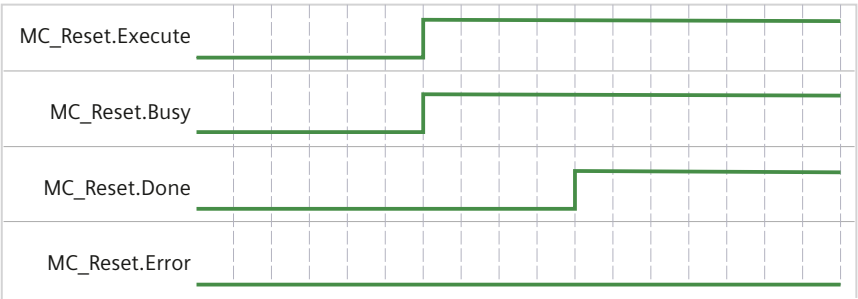
要重启某个轴，请按以下步骤操作：

1. 首先检查是否满足上述要求。
2. 在参数“Execute”的上升沿，进行重启。

如果参数“Done”的值显示为“TRUE”，则表示重启作业已经结束。

说明

即使完成重启，如果错误没能消除，会重新进入 error stop 状态。（如网络连接问题，servo/电机连接问题）



超驰响应

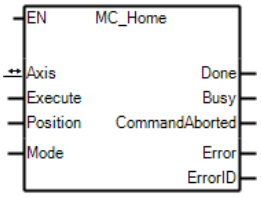
- MC_Reset 指令无法被其他任何运动控制指令中止或缓冲。
- 新的 MC_Reset 指令不会中止或缓冲任何其他激活的运动控制指令。

14.7.1.5 MC_Home

MC_Home 把轴实际的机械位置和 S7-200 SMART 程序中轴的位置统一。

参数

表格 14-2 MC_Home

LAD/FBD	STL	说明
	**FBCALL MC_Home, INSTANCE, Axis, Execute, Position, Mode, Done, Busy, CommandAborted, Error, ErrorID**	此指令设置或搜索轴的原点。

参数	声明	数据类型	初始值	说明
EN	IN	BOOL	-	能流
Axis	IN_OUT	SYS_AXIS	-	引用轴
Execute	IN	BOOL	-	上升沿触发该指令
Position	IN	REAL	0.0	指定回零后的位置 指令结束后，轴的位置还要结合 Mode 来确定。

参数	声明	数据类型	初始值	说明
Mode	IN	BYTE	-	回原点模式： • 0：直接回原点（绝对） • 1：直接回原点（相对） • 3：主动回原点[1]
Done	OUT	BOOL	False	成功设置或找到原点
Busy	OUT	BOOL	False	指令正在运行中
CommandAborted	OUT	BOOL	False	当前指令被另一个指令打断
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1068)

[1] 模式 3 只支持 PTO 轴和 IRT 轴， 执行主动回原点之前，必须在轴 “主动回原点” 页面中正确组态参数且相关的 PTO 轴和 IRT 轴都必须处于静止状态(Axis_Status.StandStill = TRUE)。

说明

对于模态轴，MC_Home 对 Position 引脚的值进行转换以处于模数范围内，然后将其设置为轴位置。

回原点模式

模式	说明	适用轴
模式 0	直接回原点（绝对） 轴的当前位置被设置为参数“Position”的值。	适用于所有轴
模式 1	直接回原点（相对） 轴的当前位置将根据参数“Position”的值进行偏移。	适用于所有轴
模式 3	主动回原点 运动完成后，轴将定位在“Position”参数的值处。	适用于 PTO 轴和 IRT 轴

回零状态的保持

下表说明了在不同场景中，各类轴回零状态（Homed）的保持与变化情况。

行为	PTO 轴		虚轴		编码器轴	
	回零状态	轴位置	回零状态	轴位置	回零状态	轴位置
MC_Power OFF	丢失	保持	保持	保持		
CPU STOP -> RUN	丢失	重置	保持	保持	保持	根据实际位置变化
重新执行 MC_Home 但未完成期间	丢失	根据实际位置变化	丢失	保持	丢失	根据实际位置变化
PLC 断电	丢失	重置	丢失	重置	丢失	重置
程序下载	丢失	重置	丢失	重置	丢失	重置

行为	PN IRT 轴 (增量式编码器和单圈绝对值编码器)		PN IRT 轴 (多圈绝对值编码器)	
	回零状态	轴位置	回零状态	轴位置
MC_Power OFF	保持	保持	保持	保持
CPU STOP -> RUN	保持	保持	保持	保持
重新执行 MC_Home 但未完成期间	丢失	根据实际位置变化	丢失	根据实际位置变化
PLC 断电/伺服断联	丢失	增量式编码器：不确定的值。 单圈绝对值编码器：根据实际位置变化。然而，当其内部计数发生溢出时，所报告的位置数据将变得不可靠。	保持	保持
程序下载	丢失	保持	保持	保持

前置条件

绝对直接回零（模式 0）及相对直接回零（模式 1）

- 轴已正确组态
- 轴已启用

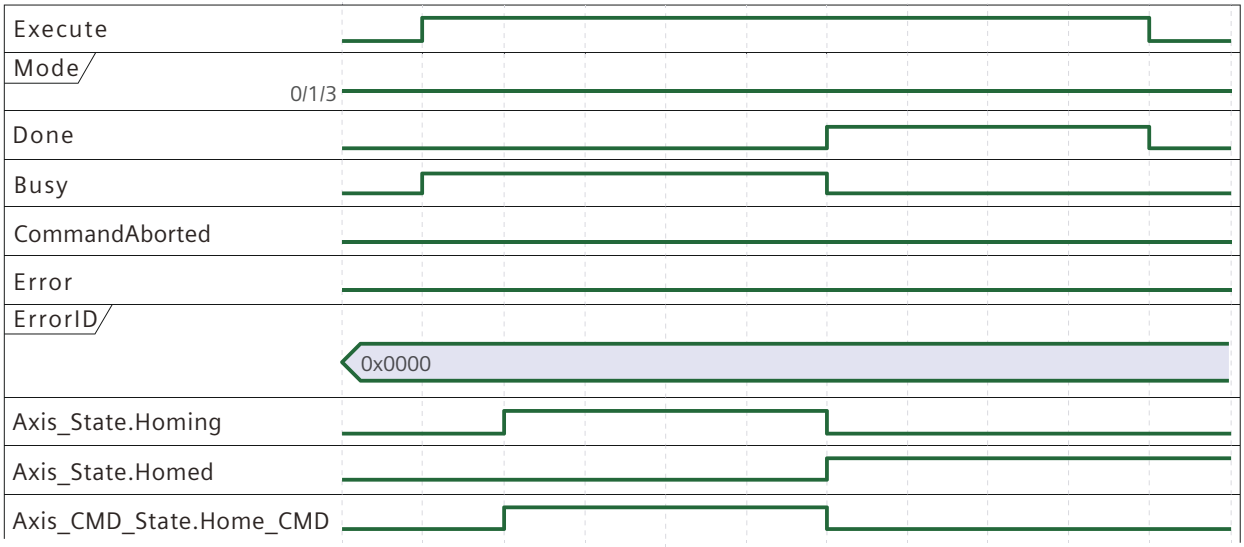
- 轴未被轴组使能
- 轴状态要求
  - 编码器轴：必须处于使能状态(Axis_Status.ENC_Enable = TRUE)
  - 其他轴：必须处于未使能状态或静止状态（Axis_Status.Disabled =TRUE or Axis_Status.StandStill = TRUE）

主动回零（模式 3）

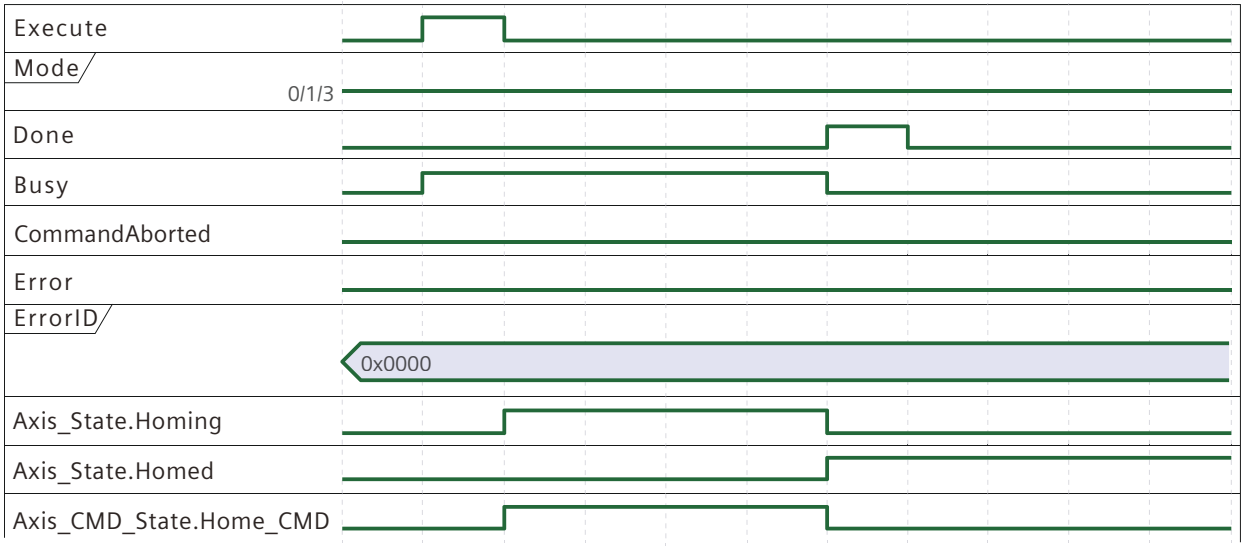
- 支持 PTO 轴和 IRT 轴
- 轴未被轴组使能
- 轴必须处于静止状态（Axis_Status=StandStill）

原点未确定时执行 MC_Home

- 触发信号在回零过程中持续高电平

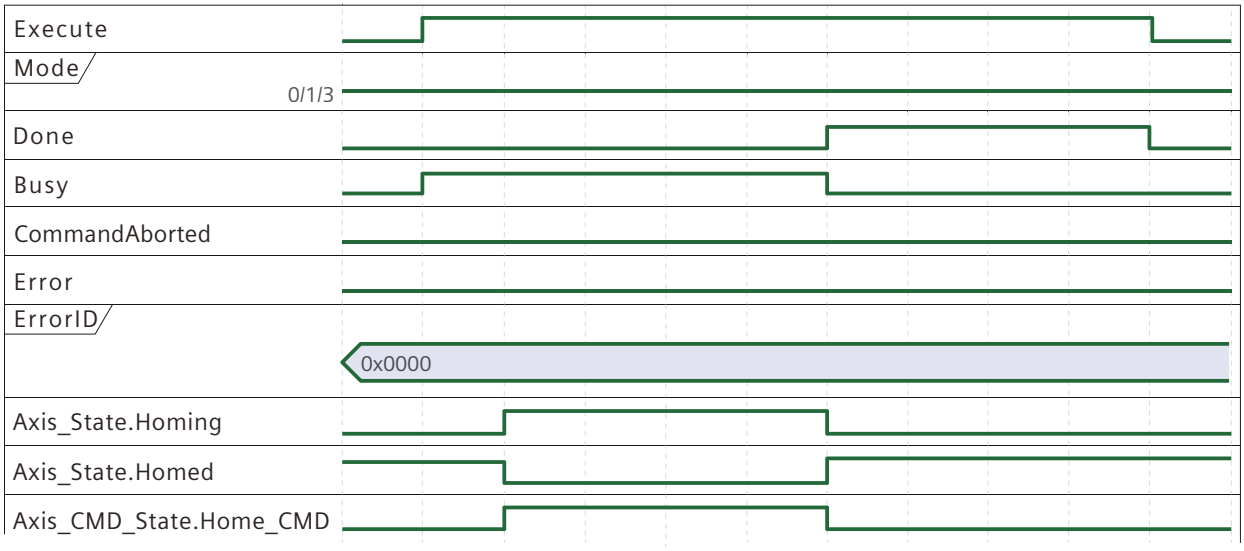


- 触发信号在回零结束前变为低电平

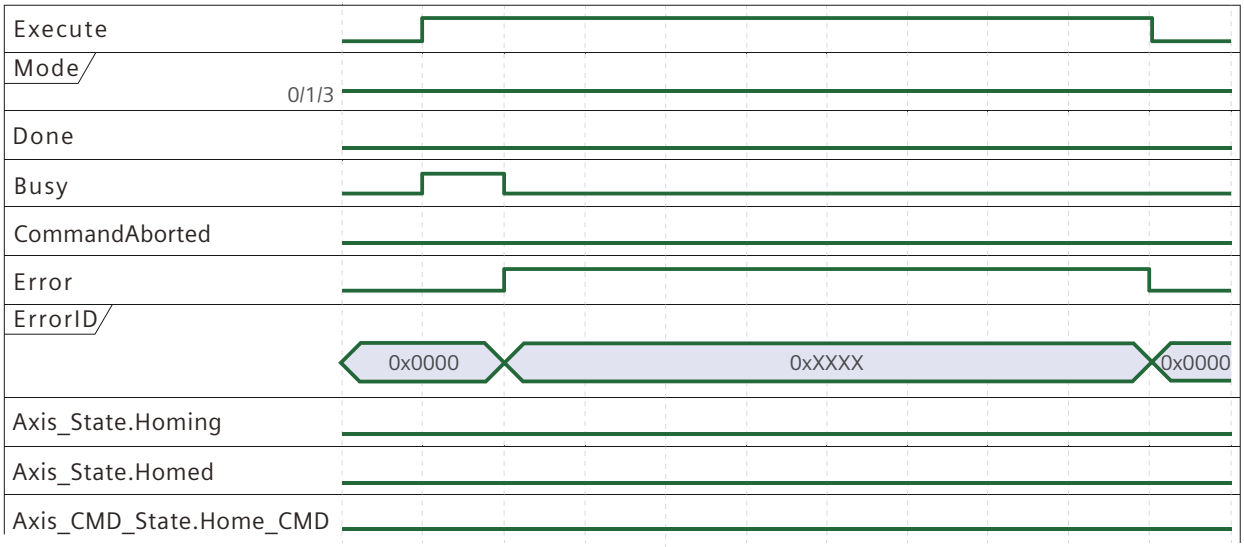


原点确定时执行 MC_Home

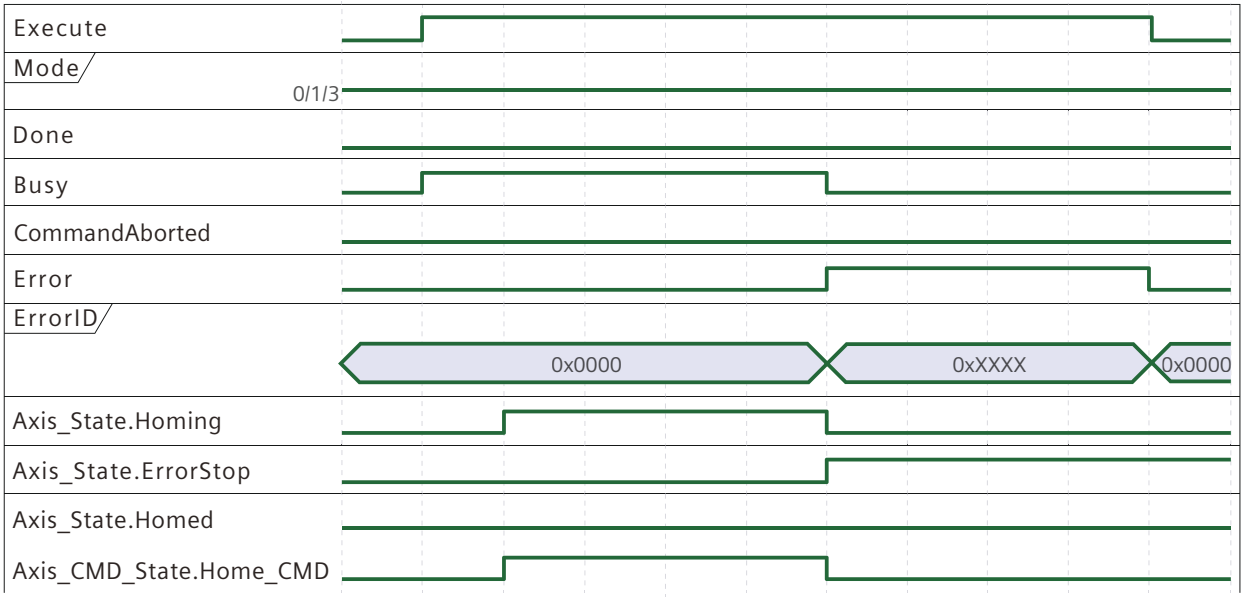
如果 MC_Home 重复执行，也就是在 Axis_State.Homed 状态下执行，那么会先拉低 Axis_State.Homed。



不满足 MC_Home 执行条件



MC_Home 执行中出错



超驰响应

- MC_Home 指令无法被 MC_Power（禁用）、MC_Stop 和 MC_ImmediateStop 以外的其他任何运动控制命令中止。
- MC_Home 指令无法被任何运动控制指令缓冲。
- MC_Home 指令无法中止任何运动控制指令。
- MC_Home 指令无法缓冲到任何运动控制指令。

参见

回原点 (页 871)

14.7.1.6 MC_Stop

该指令适用于 IRT 轴、PTO 轴和虚轴，用于命令受控停止运动，并将轴切换到 Stopping 状态。它会中断任何正在进行的运动 FB 执行。当轴处于 Stopping 状态时，除 MC_ImmediateStop 外，其他任何运动功能块均无法对该轴进行运动操作。一旦轴速度达到零（对于 IRT 轴，满足用户设定的停止判定条件即视为速度为零），Done 输出将立即置为 TRUE。在 Execute 仍为 TRUE 或速度尚未归零期间，轴将保持在 Stopping 状态。当 Done 已置为 TRUE 且 Execute 为 FALSE 时，轴将切换至 Standstill 状态。

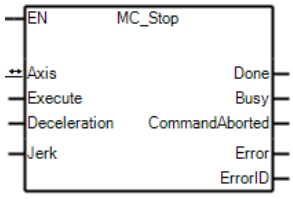
说明

只要 Execute 信号保持高电平，轴将持续处于 Stopping 状态。

说明

对于扭矩控制模式（MC_TorqueControl Mode 1），MC_STOP 不生效

参数

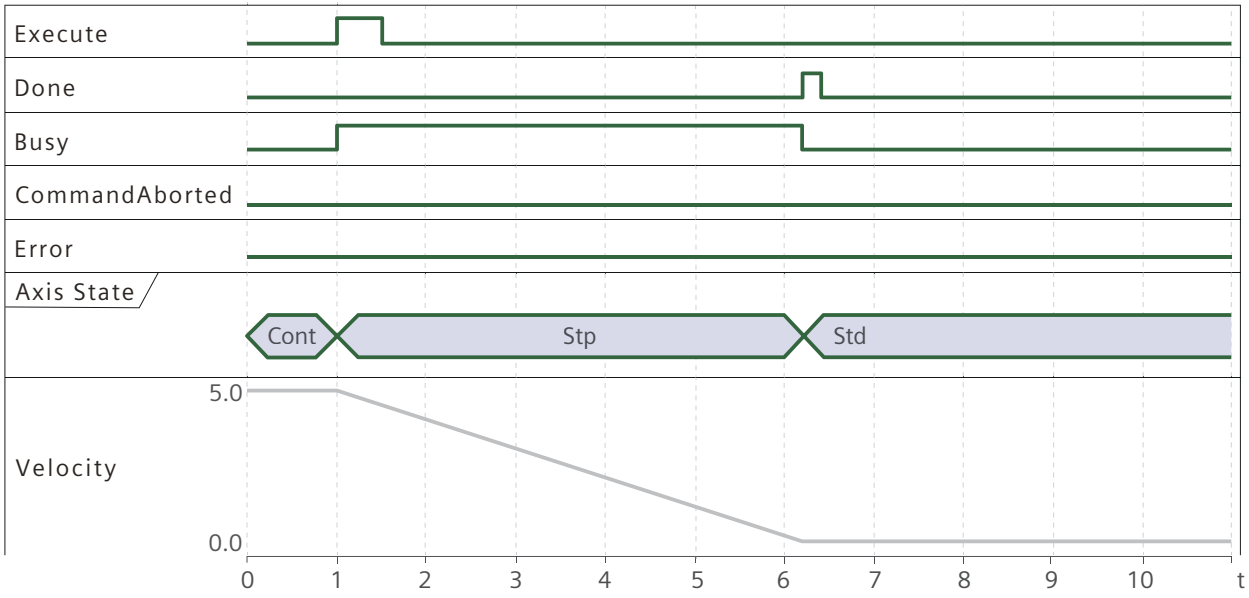
LAD/FBD	STL	说明
	`FBCALL MC_Stop, INSTANCE, Axis, Execute, Deceleration, Jerk, Done, Busy, CommandAborted, Error, ErrorID`	该指令命令轴减速停止，并将轴转换为 **Stopping** 状态。它会中止任何正在进行的指令执行（MC_ImmdiateStop 除外）。当轴处于 **Stopping** 状态时，任何其他指令都不能对同一轴执行任何运动。 当轴的 **Velocity** 为零时，**Done** 输出立即设置为 TRUE。 只要 **Execute** 仍然为 TRUE 或尚未达到 **Velocity** 零值，轴就会保持处于 **Stopping** 状态。**Done** 为 **TRUE** 且 **Execute** 为 FALSE 时，轴就会进入 **Standstill** 状态。

参数	声明	数据类型	初始值	说明
EN	IN	BOOL	-	能流
Axis	IN_OUT	SYS_AXIS	-	引用轴
Execute	IN	BOOL	-	上升沿触发该指令
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	减速度值
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	加加速度值
Done	OUT	BOOL	False	• 速度已达到零 • IRT 轴已满足用户设定的停止条件
Busy	OUT	BOOL	False	指令正在运行中
CommandAborted	OUT	BOOL	False	当前指令被另一个指令打断 • 当 Stopping 状态尚未解除时，如果被 MC_Power 或 MC_ImmediateStop 指令中断，则此引脚将被置为 True
Error	OUT	BOOL	False	表示指令内发生了错误
ErrorID	OUT	WORD	0	错误代码 (页 1068)

前提条件

- 轴已正确组态。
- 轴已使能。

执行流程



超驰响应

- 可以中止 MC_Stop 指令的运动控制指令：

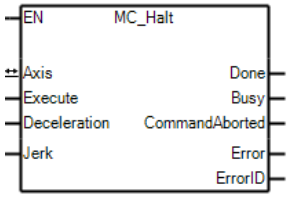
中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_Power MC_ImmediateStop	

- 新的 MC_Stop 指令可以中止以下运动控制指令：

中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_Home MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_MoveJog MC_CamIn MC_GearIn MC_CamOut MC_GearOut MC_PhasingRelative	

14.7.1.7 MC_Halt

表格 14-3 MC_Halt

LAD/FBD	STL	说明
	**FBCALL MC_Halt, INSTANCE, Axis, Execute, Deceleration, Jerk, Done, Busy, CommandAborted, Error, ErrorID**	该指令用于命令一个受控轴运动的停止。当执行该指令时，轴将进入 'Discrete' 状态，并从当前位置、沿当前运动方向，以设定的“Deceleration” 和“Jerk” 参数将当前速度平滑减小至零。

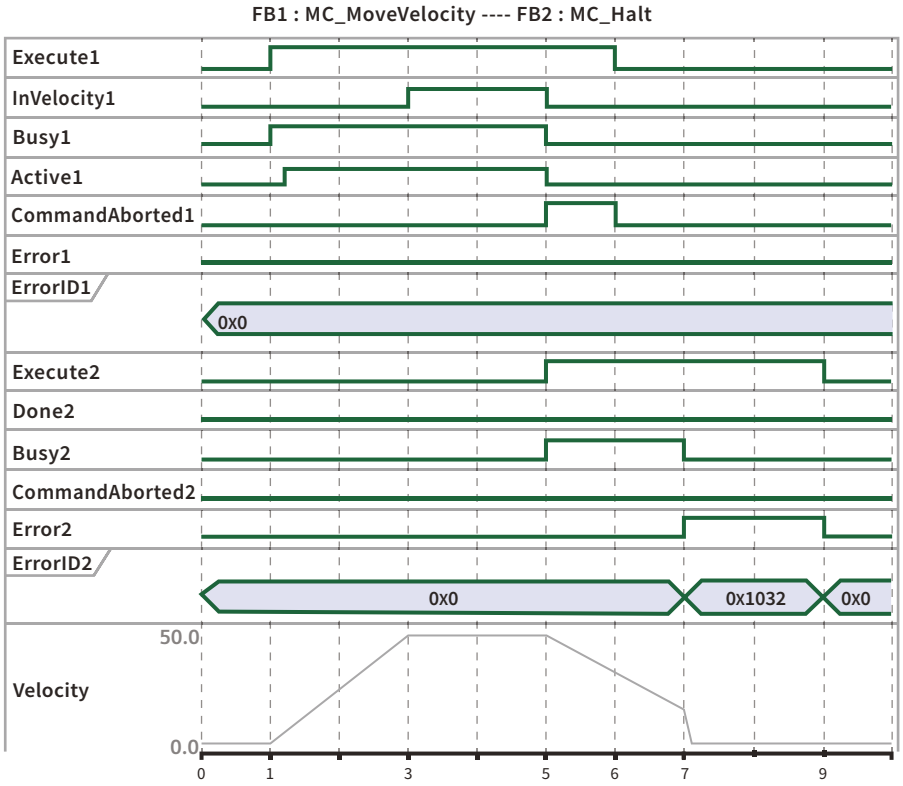
表格 14-4 MC_Halt 参数

参数	声明	数据类型	初始值	说明
EN	IN	BOOL	-	能流
Axis	IN_OUT	SYS_AXIS	-	引用轴
Execute	IN	BOOL	-	上升沿触发该指令

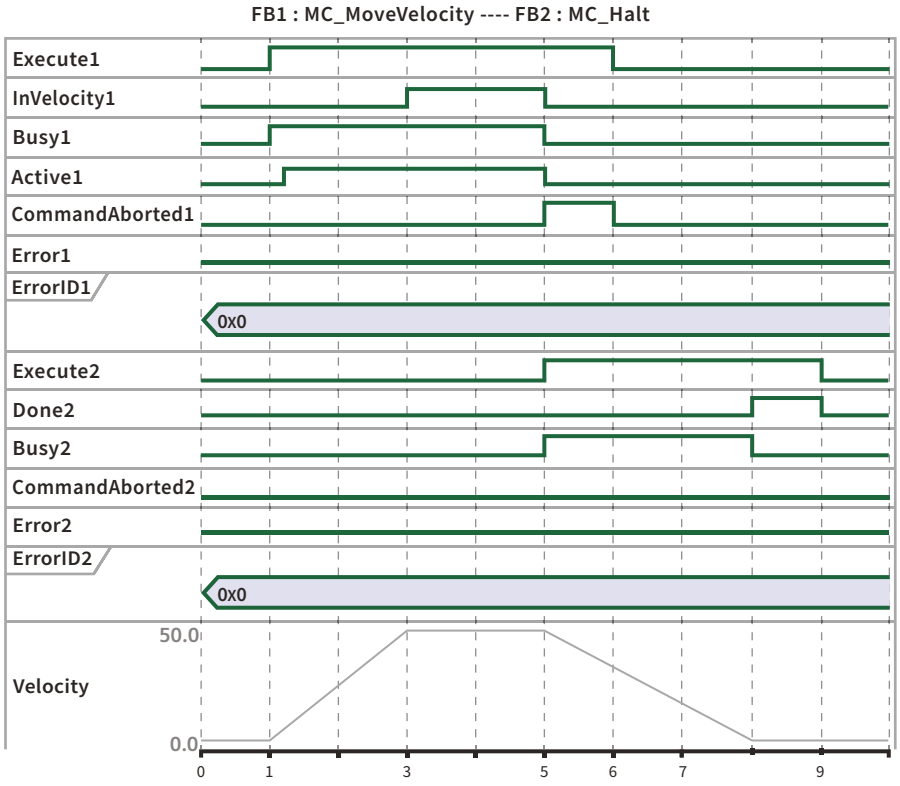
参数	声明	数据类型	初始值	说明
Deceleration	IN	REAL	Axis_Config. Dynamic_Defaul t. Default_Decel	减速度值，可缺省 • ＞0.0：使用指定值。
Jerk	IN	REAL	Axis_Config. Dynamic_Defaul t. Default_Jerk	加加速度值，可缺省 • ＞0.0：S 形速度曲线 • =0： 梯形速度曲线
Done	OUT	BOOL	False	True：速度已达到零。 对于 IRT 轴，Done 置位还表示已满足 StandStill 条件。
Busy	OUT	BOOL	False	指令正在运行中
CommandAborted	OUT	BOOL	False	当前指令被另一个指令打断
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1068)

时序图

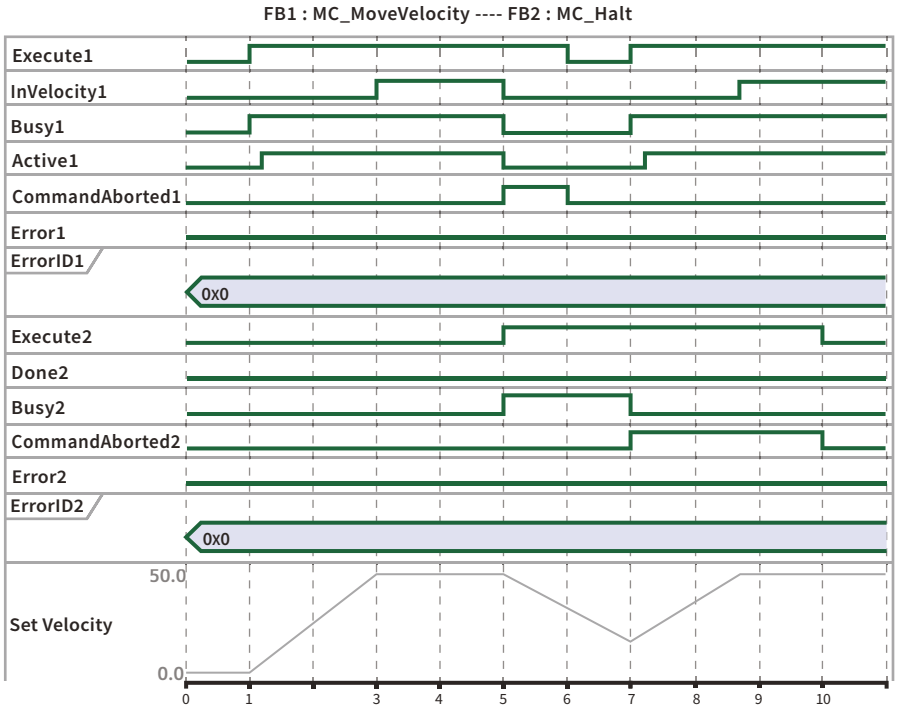
- 在 MC_Halt 指令执行减速至停止过程中，若发生轴错误等错误，系统会立即中断当前减速过程，并根据错误类型采取相应的错误处理措施。例如，遇到硬限位错误时，控制器将优先响应限位保护，停止轴的进一步运动，并报告对应的错误状态。



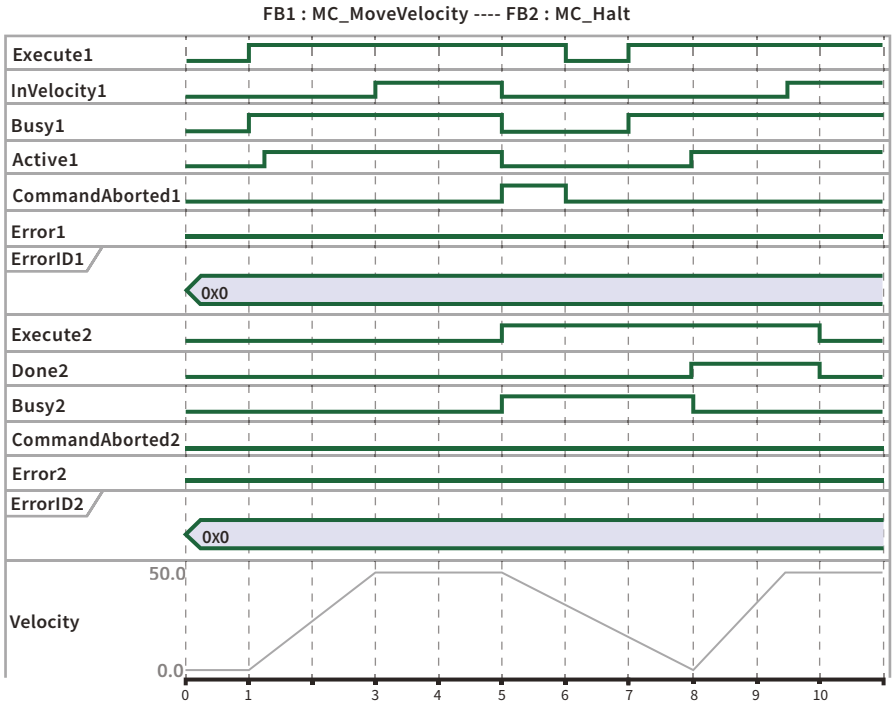
- 当触发 MC_MoveVelocity 并执行 T 型加减速运动曲线后，轴将在加速至设置的最大速度 InVelocity 时，处于 Continuous（连续运动）状态。此时如果触发 MC_Halt 指令，轴将按照设定的减速度和加加速度（Jerk）参数开始减速，直至停至零速，并最终进入 Standstill（静止）状态。


FB1 : MC_MoveVelocity ---- FB2 : MC_Halt
Execute1
InVelocity1
Busy1
Active1
CommandAborted1
Error1
ErrorID1
0x0
Execute2
Done2
Busy2
CommandAborted2
Error2
ErrorID2
0x0
Velocity
50.0
0.0
0
1
3
5
6
7
9

- 在 MC_Halt 指令执行减速停止运动期间，如果又收到处于 Aborting 模式的 MC_MoveVelocity 指令打断，则 MC_Halt 当前的减速过程会被中断，轴立即响应新的 MC_MoveVelocity 指令，按照其 T 型曲线参数执行加速或减速运动。此时，轴的运动状态和目标速度将依据新下发的 MC_MoveVelocity 指令进行调整。



- 在 MC_Halt 指令执行减速至停止的过程中，如果再次收到以 Buffered 模式下发的 MC_MoveVelocity 指令，该指令会被系统缓冲在命令队列。此时，轴仍继续按照 MC_Halt 的减速曲线直至完全停止。一旦减速过程结束并进入 Standstill 状态，系统会立即从队列中读取前面缓冲的 MC_MoveVelocity 指令，并按照其 T 型曲线参数执行加速运动。这样可以确保运动过程平滑切换，避免突变，满足运动连续性与安全性需求。



超驰响应

- 可以中止或缓冲 MC_Halt 指令的运动控制指令：

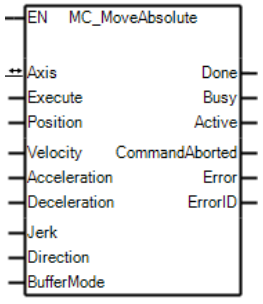
中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_Halt	MC_MoveAbsolute
MC_Stop	MC_MoveRelative
MC_ImmediateStop	MC_MoveVelocity
MC_MoveAbsolute	MC_CamIn
MC_MoveRelative	MC_GearIn
MC_MoveVelocity	
MC_MoveJog	
MC_CamIn	
MC_GearIn	

- 新的 MC_ Halt 指令可以中止以下运动控制指令：

中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_MoveJog MC_CamIn MC_GearIn MC_GearOut MC_CamOut MC_PhasingRelative	

14.7.1.8 MC_MoveAbsolute

MC_MoveAbsolute 指令通过绝对坐标指定目标位置，控制指定轴执行预期的轨迹运动。

LAD/FBD	STL	说明
	**FBCALL MC_MoveAbsolute, INSTANCE, Axis, Execute, Position, Velocity, Acceleration,Deceleration , Jerk, Direction, BufferMode,Done, Busy, Active, CommandAborted, Error, ErrorID**	该指令命令轴运动至指定的绝对位置。

参数	声明	数据类型	初始值	说明
EN	IN	BOOL	-	能流
Axis	IN_OUT	SYS_AXIS	-	引用轴
Execute	IN	BOOL	-	上升沿触发该指令
Position	IN	REAL	-	指定绝对坐标的目标位置
Velocity	IN	REAL	-	目标速度 取值范围：任意实数

参数	声明	数据类型	初始值	说明
Acceleration[1]	IN	REAL	Axis_Config. Dynamic_Default. Default_Acc	加速度值，可缺省 取值范围：> 0.0
Deceleration[1]	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	减速度值，可缺省 取值范围：> 0.0
Jerk[1]	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	加加速度值，可缺省 取值范围：>= 0.0
Direction[1]	IN	BYTE	0	轴的运动方向, 该参数只有在轴为旋转模式时才会有效。 • 0: 正向 • 1: 负向 • 2: 最短距离 • 3: 当前方向
BufferMode[1]	IN	BYTE	0	定义指令的时间顺序 • 0：打断模式 • 1：缓冲模式
Done	OUT	BOOL	False	轴到达目标位置，Done 输出置为 True
Busy	OUT	BOOL	False	指令正在运行中
Active	OUT	BOOL	False	指令正在对轴进行控制
CommandAborted	OUT	BOOL	False	该指令被另一个指令打断
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1068)

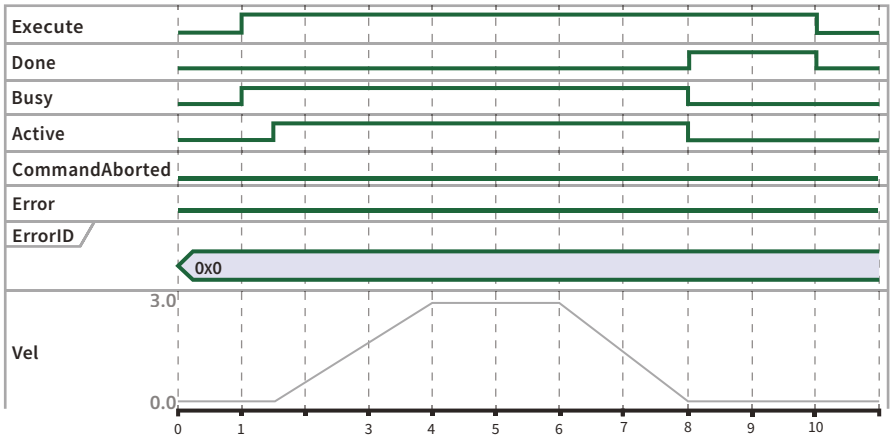
[1] 如果不为这些参数分配值，系统会使用默认初始值。

前置条件

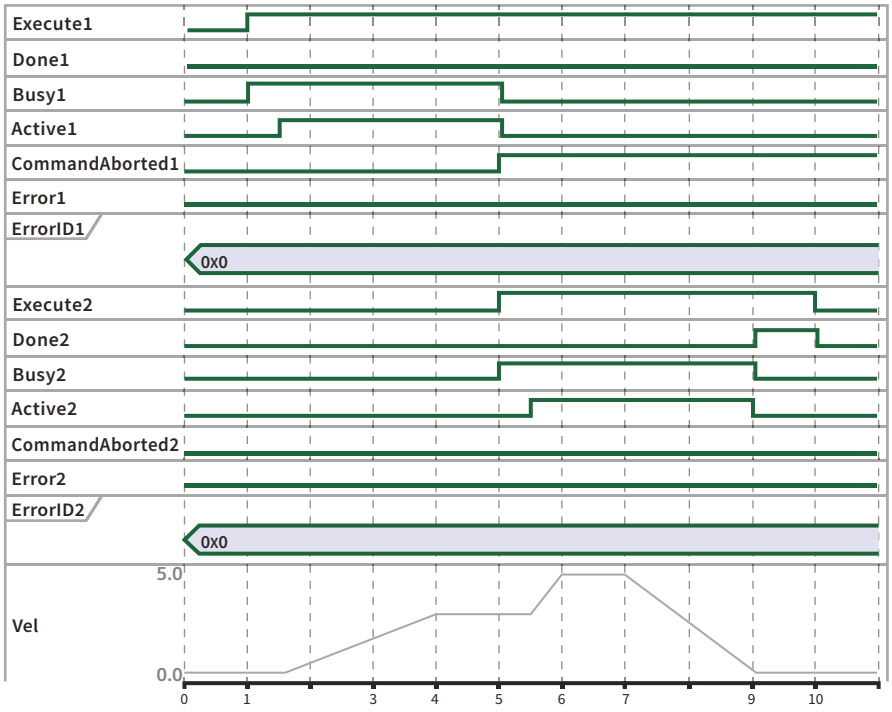
- Homed = 1

时序图

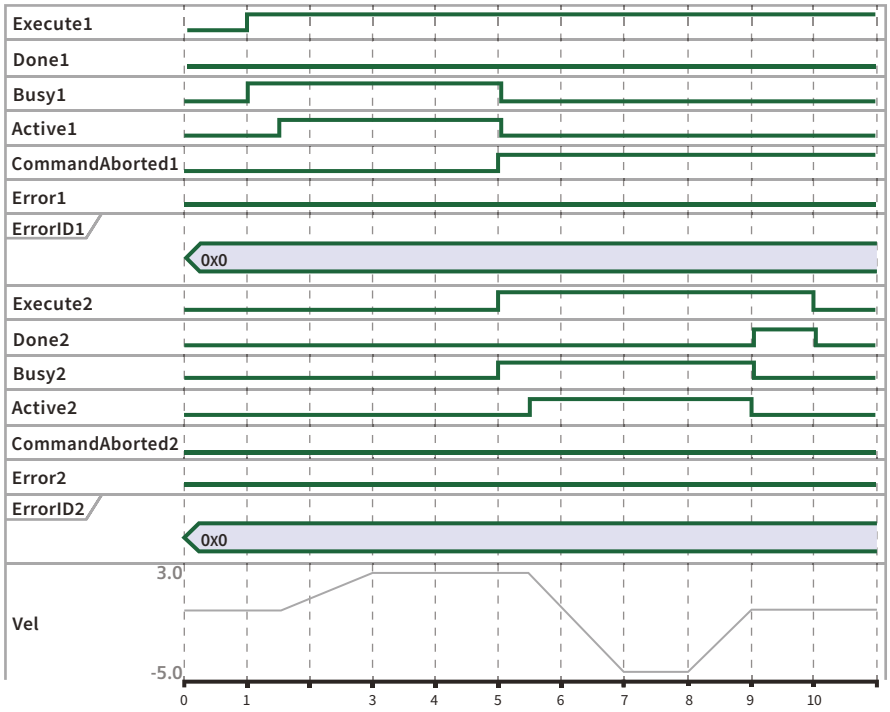
- 当轴状态处于 Standstill 和 Homed 时，触发 MC_MoveAbsolute 指令。



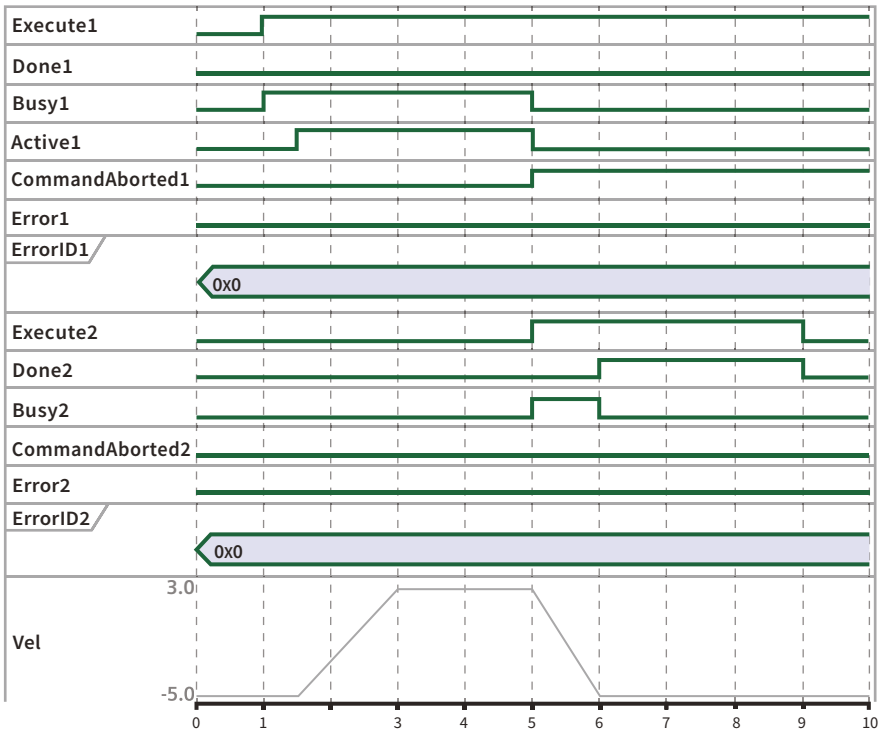
- 在梯形曲线执行至匀速阶段时，若接收到新的 MC_MoveAbsolute 指令，且目标位置与当前运动方向一致，且 BufferMode 设置为 0（无缓冲，立即打断），则当前运动会被新指令正向中断，轴根据新目标位置重新生成并执行新的运动曲线。



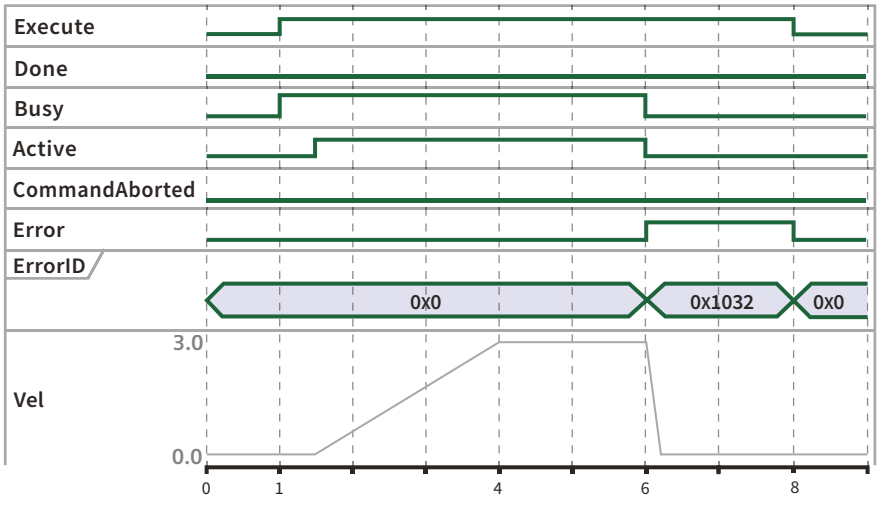
- 在梯形曲线执行至匀速阶段时，若接收到新的 MC_MoveAbsolute 指令，且目标位置与当前运动方向相反，且 BufferMode 设置为 0（无缓冲，立即打断），则当前运动会被新指令反向中断，轴根据新目标位置重新生成并执行反向运动曲线，直到完全停止。



- 在梯形曲线执行至匀速阶段时，若接收到 MC_Stop 指令，当前运动将被立即中断，轴按照预设的减速度执行停止运动。



- 在梯形曲线执行至匀速阶段时，若发生执行错误，当前运动将被强制中断，轴立即进入错误处理流程，相关故障状态和错误信息将被上报，运动停止以确保系统安全。



超驰响应

- 可以中止或缓冲 MC_MoveAbsolute 指令的运动控制指令：

中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_Halt MC_Stop MC_ImmediateStop MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_MoveJog MC_CamIn MC_GearIn	MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

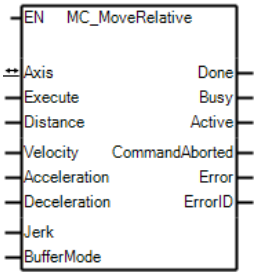
- 新的 MC_MoveAbsolute 指令可以中止或缓冲以下运动控制指令：

中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_MoveJog MC_CamOut MC_GearOut MC_CamIn MC_GearIn MC_PhasingRelative	MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

14.7.1.9 MC_MoveRelative

MC_MoveRelative 用于指定轴相对于当前位置移动指定的距离。支持指定轴配置为 PTO 轴、虚轴或 IRT 轴，支持线性轴和旋转轴。

参数

LAD/FBD	STL	说明
	**FBCALL MC_MoveRelative, INSTANCE, Axis, Execute, Distance, Velocity, Acceleration,Deceleration , Jerk, BufferMode,Done, Busy, Active, CommandAborted, Error, ErrorID**	该指令命令轴相对于当前位置移动指定的距离。

参数	声明	数据类型	初始值	说明
EN	IN	BOOL	-	能流
Axis	IN_OUT	SYS_AXIS	-	引用轴
Execute	IN	BOOL	-	上升沿处开始指令
Distance	IN	REAL	-	目标相对距离
Velocity	IN	REAL	-	目标速度 取值范围：任意实数
Acceleration	IN	REAL	Axis_Config.Dynamic_Default.Default_Acc	加速度值，可缺省 取值范围：> 0.0
Deceleration	IN	REAL	Axis_Config.Dynamic_Default.Default_Decel	减速度值，可缺省 取值范围：> 0.0
Jerk [1]	IN	REAL	Axis_Config.Dynamic_Default.Default_Jerk	加加速度值，可缺省 取值范围：>= 0.0
BufferMode	IN	BYTE	0	定义指令的时间顺序 • 0：打断模式，上一个运动功能块指令立即被中止，以打断时刻的位置和速度执行当前指令 • 1：缓冲模式，上一个运动功能块指令正常执行完后，无过渡的继续执行当前指令，两段运动中间无等待
Done	OUT	BOOL	False	轴到达目标位置，Done 输出置为 True
Busy	OUT	BOOL	False	指令正在运行中

参数	声明	数据类型	初始值	说明
Active	OUT	BOOL	False	指令正在对轴进行控制
CommandAborted	OUT	BOOL	False	该指令被另一个指令打断
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1068)

[1] 运动指令支持 T、三角形和 S 型速度曲线。

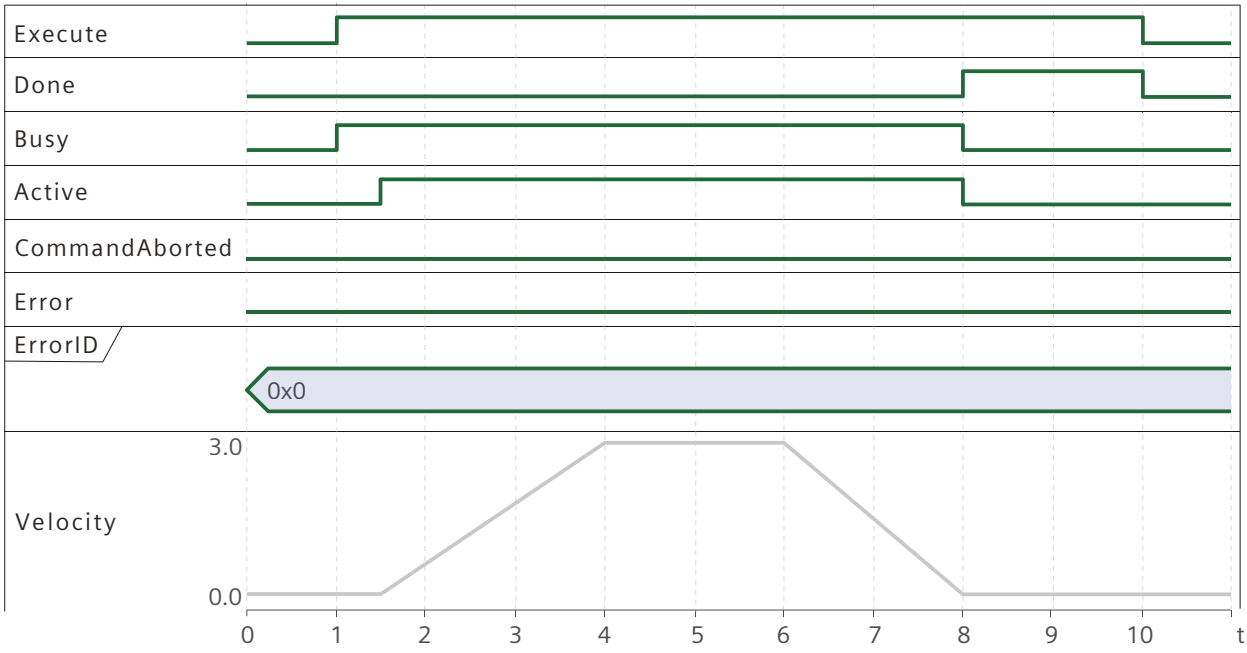
- 当 Jerk 设置为 0 时：
  系统将执行 T 型速度运动。如果由于距离限制，目标速度无法达到或刚好达到，导致没有匀速段，则速度曲线将退化为三角形曲线。
- 当 Jerk 设置为 >0 时：系统将执行 S 型速度运动，最多包含 7 个段。如果由于距离限制目标速度无法达到，S 型曲线因为无法在有限计算时间内找到最优的调整后的目标速度，S 型曲线会按照 T 型曲线来重新规划。

前置条件

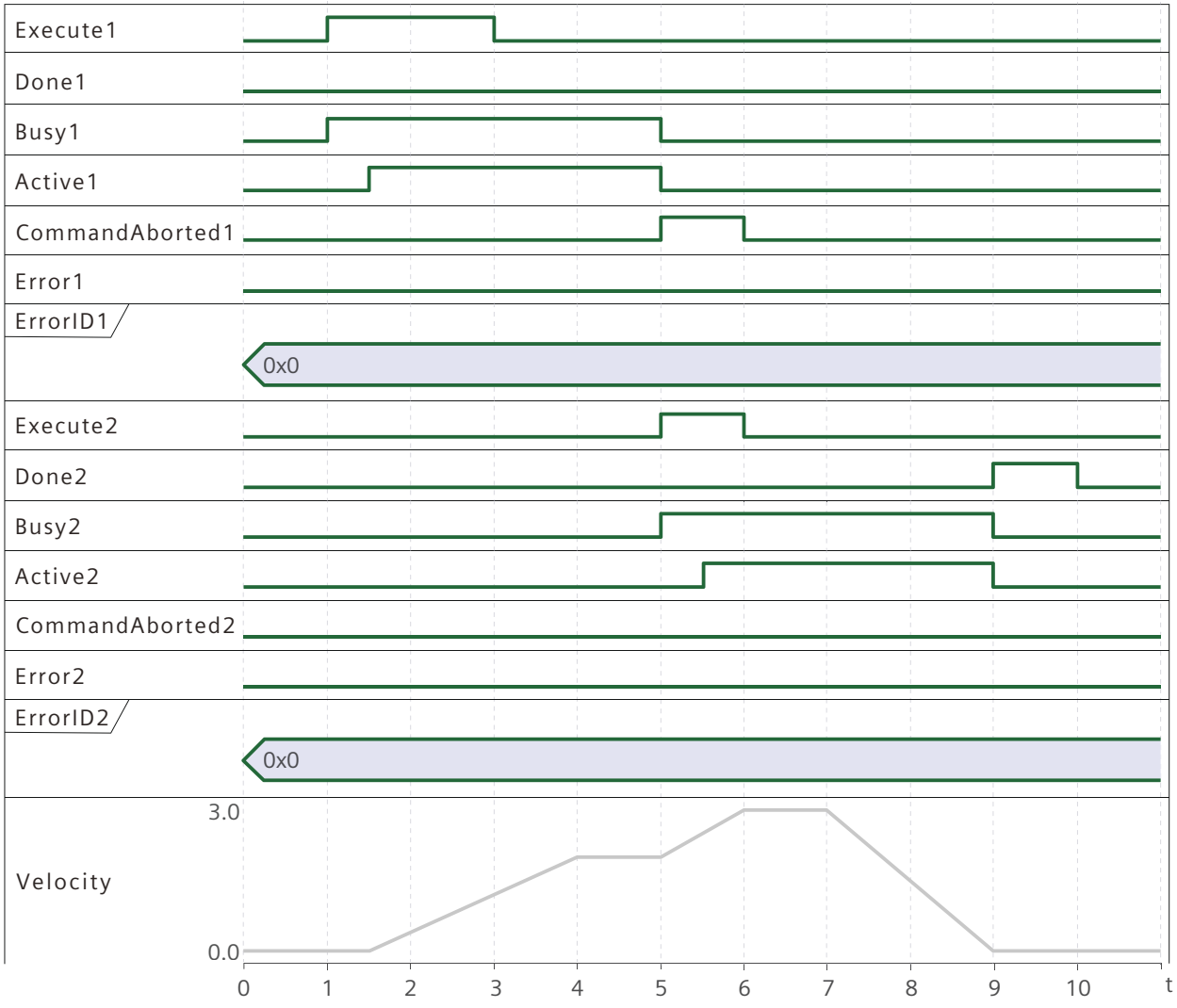
- 轴已正确组态。
- 轴已启用。

执行流程

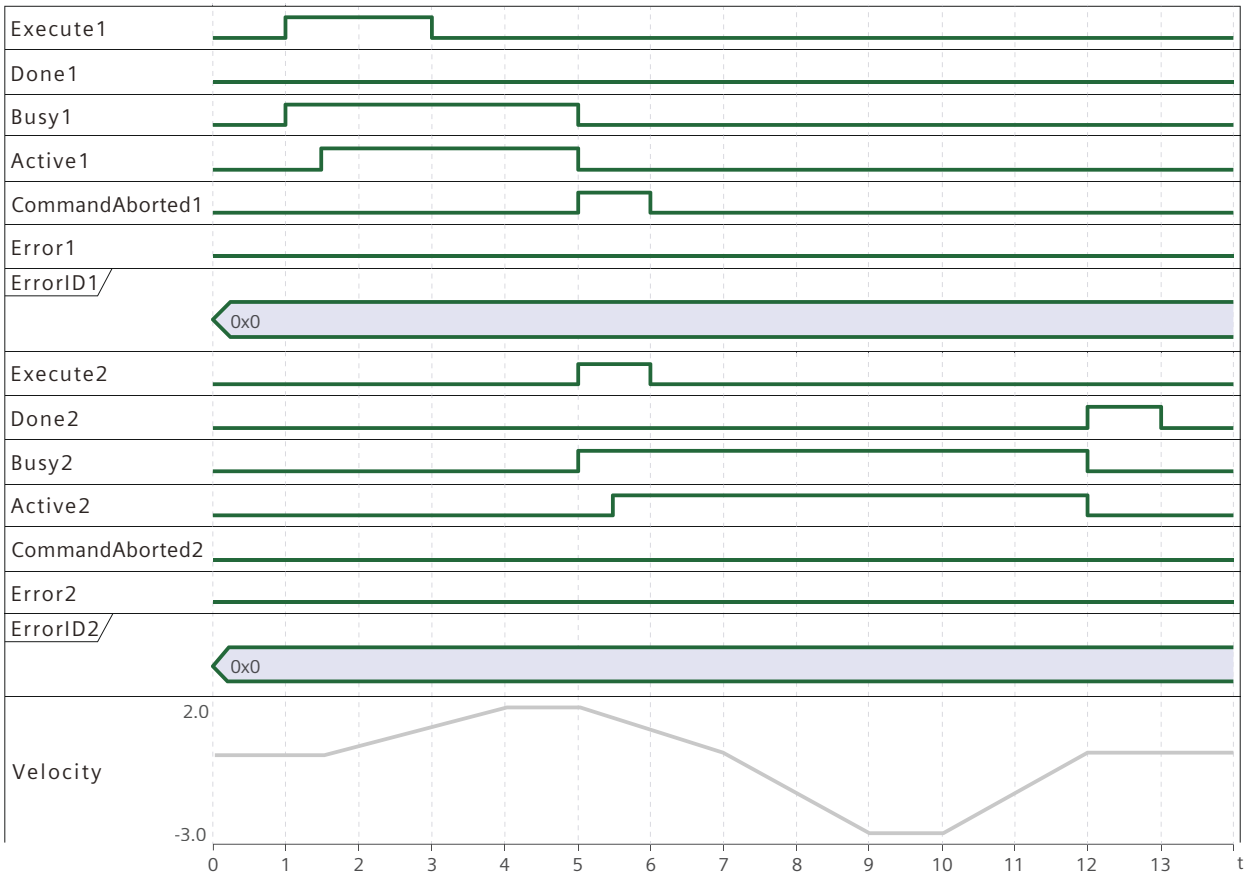
- 当轴处于 StandStill 状态时，调用此指令将执行 T 型速度曲线的相对位置运动。



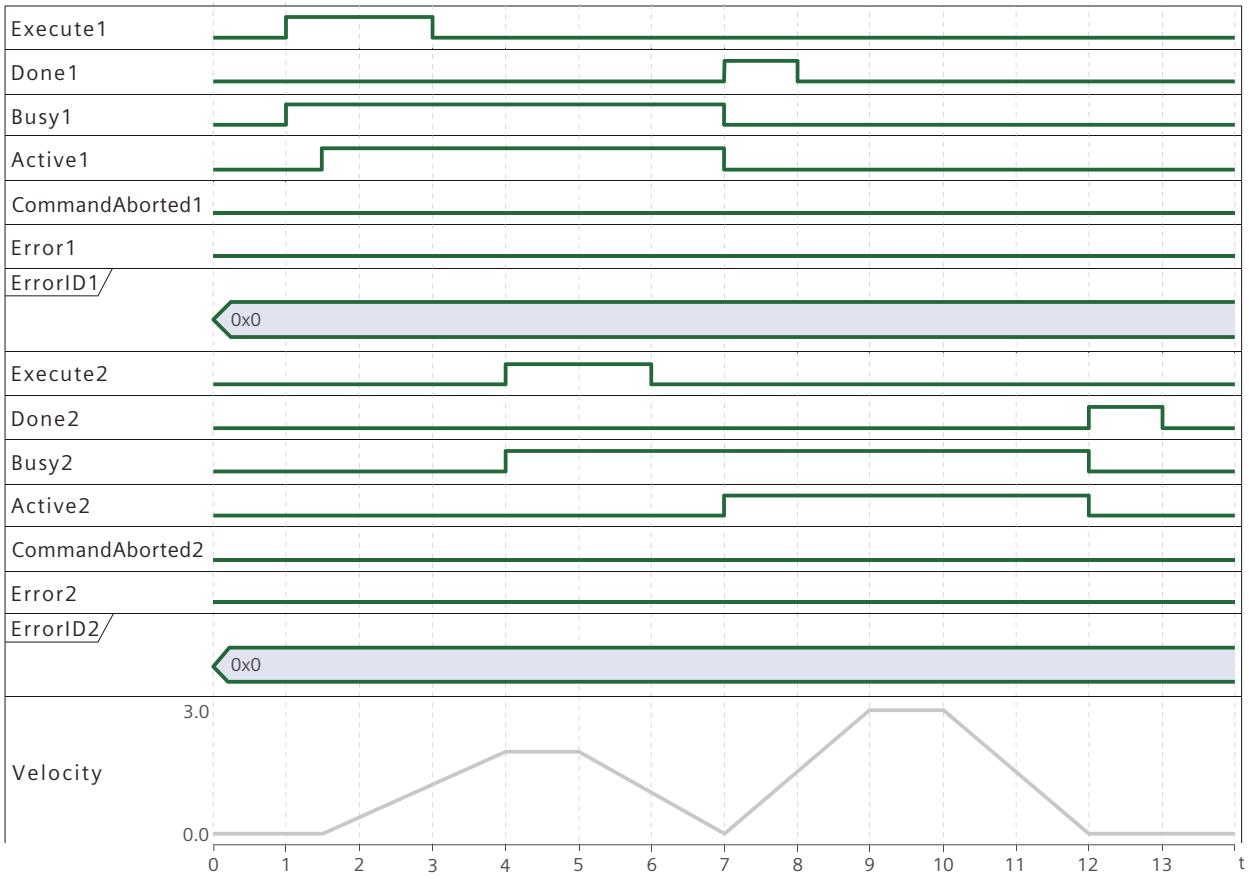
- 正在执行 T 型曲线相对运动时，触发同方向新的 T 型相对运动指令（BufferMode=0，正向打断）。


Execute1
Done1
Busy1
Active1
CommandAborted1
Error1
ErrorID1
0x0
Execute2
Done2
Busy2
Active2
CommandAborted2
Error2
ErrorID2
0x0
Velocity
3.0
0.0
0
1
2
3
4
5
6
7
8
9
10
t

- 正在执行 T 型曲线相对运动时，触发反方向新的 T 型相对运动指令（BufferMode=0，反向打断）。



- 正在执行 T 型曲线相对运动时，触发新的 T 型相对运动指令（BufferMode=1）。



超驰响应

- 可以中止或缓冲 MC_MoveRelative 指令的运动控制指令：

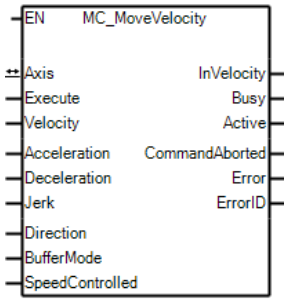
中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_Halt	MC_MoveAbsolute
MC_Stop	MC_MoveRelative
MC_ImmediateStop	MC_MoveVelocity
MC_MoveAbsolute	MC_CamIn
MC_MoveRelative	MC_GearIn
MC_MoveVelocity	
MC_MoveJog	
MC_CamIn	
MC_GearIn	

- 新的 MC_MoveRelative 指令可以中止或缓冲以下运动控制指令：

中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_MoveJog MC_CamIn MC_GearIn MC_CamOut MC_GearOut MC_PhasingRelative	MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

14.7.1.10 MC_MoveVelocity

MC_MoveVelocity 指令用于控制指定轴以设定速度进行持续的受控运动。它支持 PTO 轴、虚轴和 PN IRT 轴，适用于线性轴和旋转轴。

LAD/FBD	STL	说明
	**FBCALL MC_MoveVelocity, INSTANCE, Axis, Execute, Velocity, Acceleration, Deceleration, Jerk, Direction, BufferMode, SpeedControlled, Invelocity, Busy, Active, CommandAborted, Error, ErrorID**	该指令命令轴以指定的速度进行永不停止的受控运动。

表格 14-5 MC_MoveVelocity 参数

参数	声明	数据类型	初始值	说明
EN	IN	BOOL	-	能流
Axis	IN_OUT	SYS_AXIS	-	引用轴
Execute	IN	BOOL	-	上升沿触发指令
Velocity	IN	REAL	-	目标速度 取值范围：任意实数

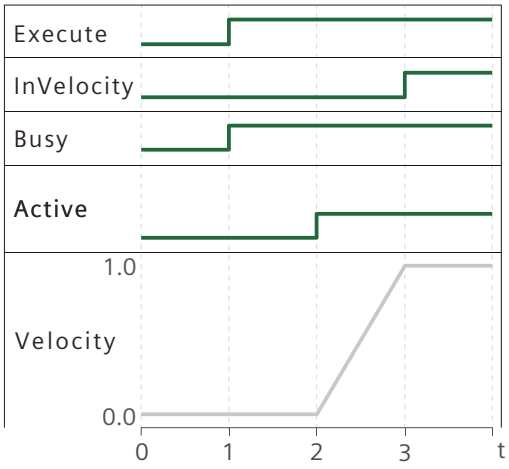
参数	声明	数据类型	初始值	说明
Acceleration	IN	REAL	Axis_Config.Dynamic_Default.Default_Acc	加速度值，可缺省 取值范围：> 0.0
Deceleration	IN	REAL	Axis_Config.Dynamic_Default.Default_Decel	减速度值，可缺省 取值范围：> 0.0
Jerk	IN	REAL	Axis_Config.Dynamic_Default.Default_Jerk	加加速度值，可缺省 取值范围：>= 0.0
Direction	IN	BYTE	0	移动方向 0 - 正向，1 - 负向。 此引脚与 Velocity 引脚的符号一起决定轴是朝正向还是负向移动。例如，如果 Direction = 1 且 Velocity = 10.0，则轴朝负向移动。如果 Direction = 1 且 Velocity = -10.0，则轴朝正向移动。
BufferMode	IN	BYTE	0	缓冲模式 • 0：打断模式 • 1：缓冲模式
SpeedControlled	IN	BOOL	0	控制模式 • 0：位置控制模式 • 1：速度控制模式
InVelocity	OUT	BOOL	False	给定速度等于目标速度时为 True
Busy	OUT	BOOL	False	指令正在运行中
Active	OUT	BOOL	False	指令正在对轴进行控制
CommandAborted	OUT	BOOL	False	该指令被另一个指令打断
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1068)

前置条件

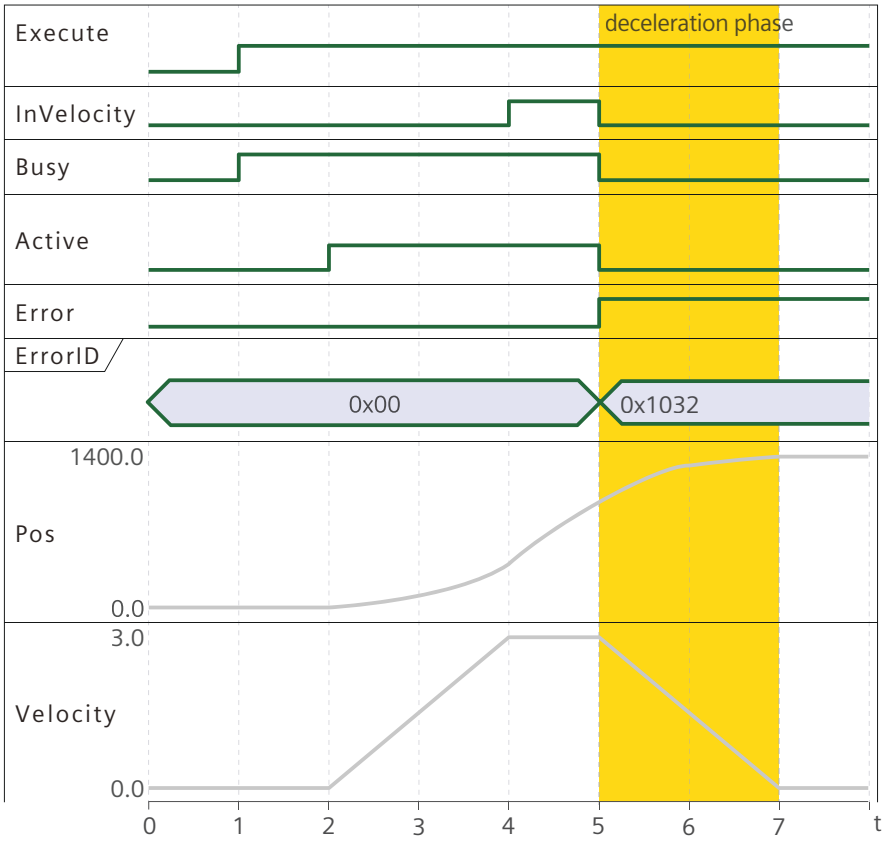
- 轴已正确组态。
- 轴已启用

时序图

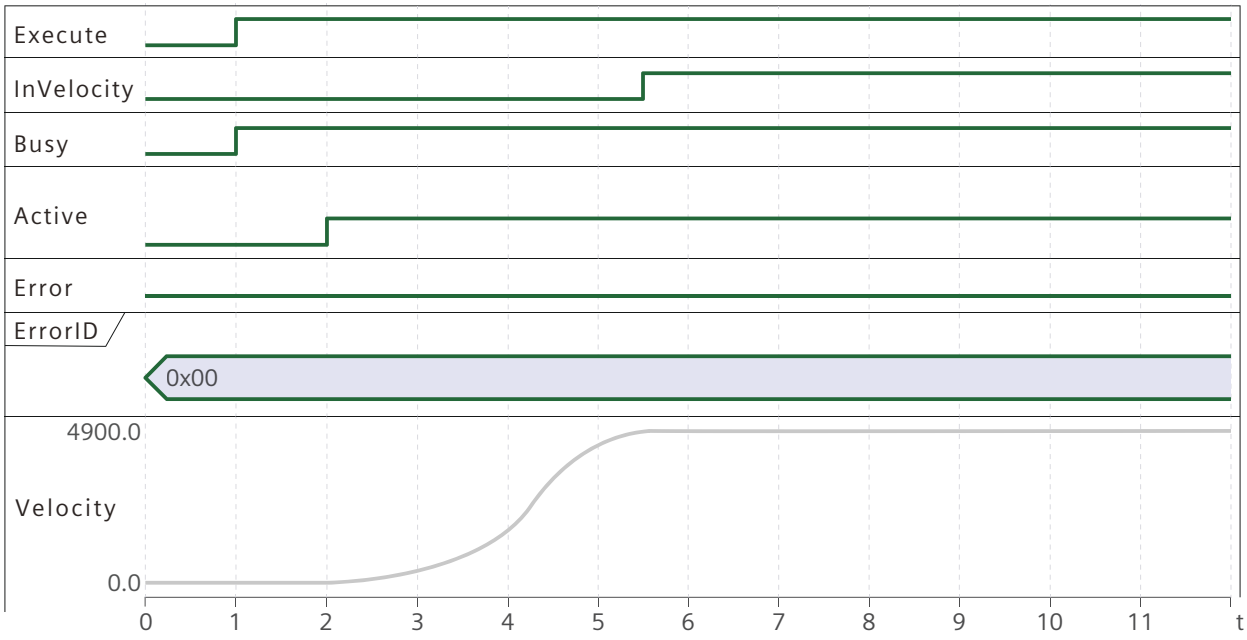
1. 加加速度为 0 的匀加速 T 型曲线



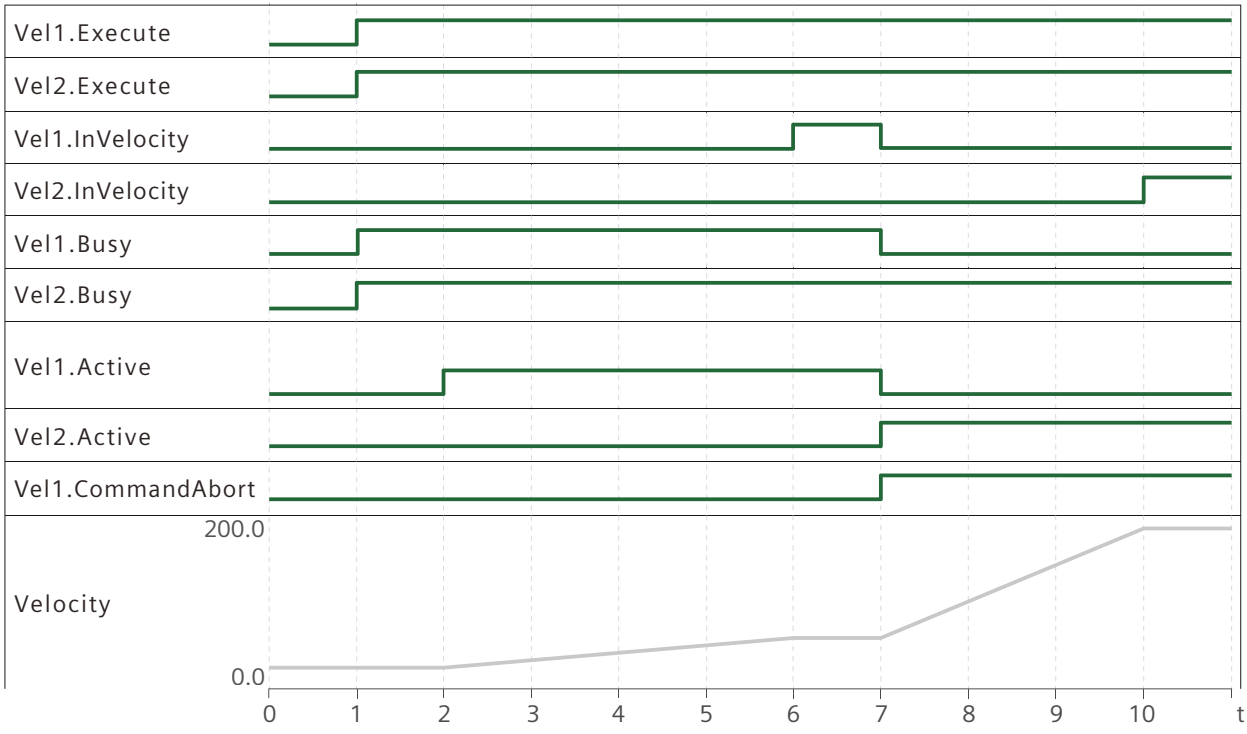
2. 加加速度为 0 的匀加速 T 型曲线（达到限位后停止）
配置了软限位，当 Pos 到达正向软限位设置值 999 后开始减速，最终在位置 1400 处停止



3. 加加速度大于 0 的变加速 S 型曲线



4. 在 MoveVelocity 的加速段缓冲 MoveVelocity
在当前 task InVelocity 后才开始执行 buffer 的 task



超驰响应

- MC_MoveVelocity 指令可由以下运动控制指令中止或缓冲：

中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_Halt MC_Stop MC_ImmediateStop MC_MoveVelocity MC_MoveAbsolute MC_MoveRelative MC_MoveJog MC_CamIn MC_GearIn	MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

- 新的 MC_MoveVelocity 指令可以中止或缓冲以下运动控制指令：

中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveJog MC_CamOut MC_GearOut MC_MoveVelocity MC_CamIn MC_GearIn MC_PhasingRelative	MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

在 MoveVelocity 之后 buffer 其他指令

- 在加速段 buffer 其他指令
  buffer 指令 Execute 拉高时 Busy 置位，等到被 buffer 的指令 task 达到 InVelocity 时自动结束退出，被 buffer 的指令 CommandAborted 置位，InVelocity/Busy/Active 拉低。此时 buffer 的指令开始执行，Active 置位。
- 在匀速段 buffer 其他指令
  此时被 buffer 的指令已经达到 InVelocity，buffer 的指令 Execute 拉高后被 buffer 的指令会立即结束退出，被 buffer 的指令 CommandAborted 置位，InVelocity/Busy/Active 拉低。buffer 的指令立即开始运行，Active 置位。

14.7.1.11 MC_ImmediateStop

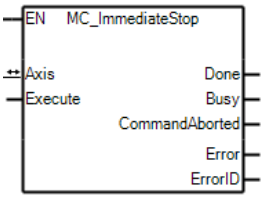
MC_ImmediateStop 指令用于停止轴的运动。

该指令支持三种轴类型： PTO 轴、虚拟轴和 PN IRT 轴，并根据轴类型提供不同的停止机制：

- PTO 轴：该指令会执行立即停止。
- PN IRT 轴、虚拟轴：该指令使用轴配置的最大减速度减速停止。

说明

对于扭矩控制模式（MC_TorqueControl Mode 1），MC_ImmediateStop 不生效。

LAD/FBD	STL	说明
	**FBCALL MC_ImmediateStop, INSTANCE, Axis, Execute, Done, Busy, CommandAborted, Error, ErrorID**	该指令命令轴立即停止，并将轴转换为 **Stopping** 状态。它会中止任何正在进行的指令执行。当轴处于 **Stopping** 状态时，任何其他指令都不能对同一轴执行任何运动。 当轴的 **Velocity** 为零时，**Done** 输出立即设置为 TRUE。 只要 **Execute** 仍然为 TRUE 或尚未达到零值，轴就会保持处于 Stopping 状态。**Done** 为 **TRUE** 且 **Execute** 为 FALSE 时，轴就会进入 **Standstill** 状态。

表格 14-6 MC_ImmediateStop 参数

参数	声明	数据类型	初始值	说明
Axis	IN_OUT	SYS_AXIS	-	引用轴
Execute	IN	BOOL	-	上升沿触发该指令
Done	OUT	BOOL	False	速度已达到零
Busy	OUT	BOOL	False	指令正在运行中
CommandAborted	OUT	BOOL	False	当前指令被另一个指令打断
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1068)

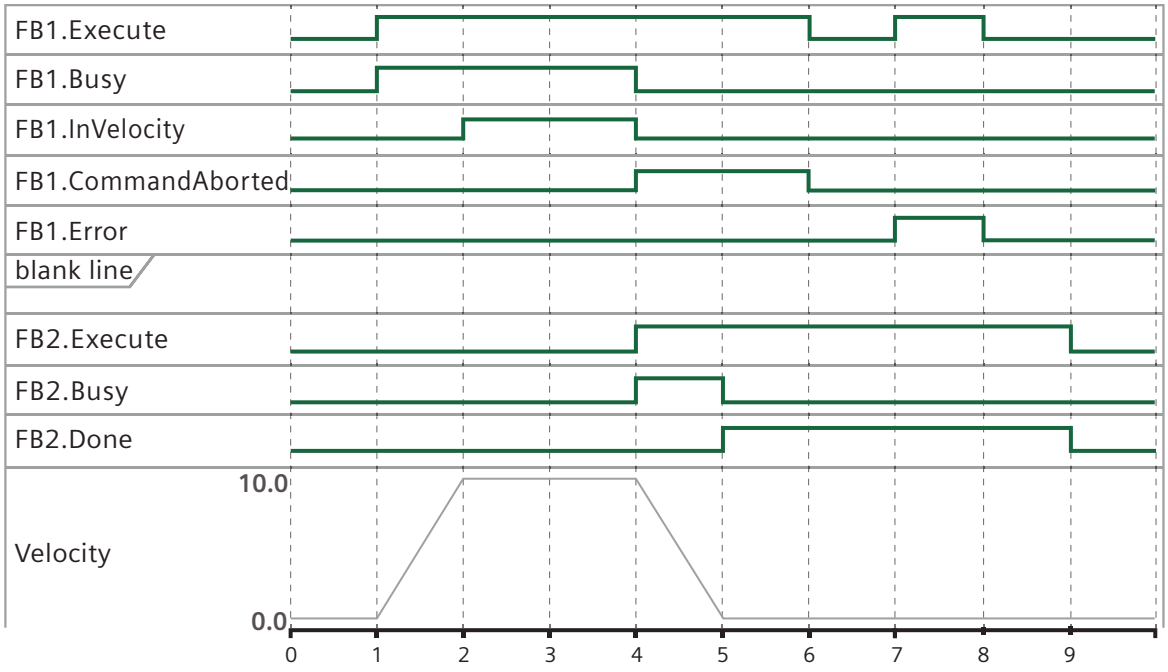
前置条件

- 轴已正确组态。
- 轴已启用

时序图

- MC_ImmediateStop 与 MC_Velocity 交互时：

FB1:MC_MoveVelocity --- FB2:MC_Immediate Stop



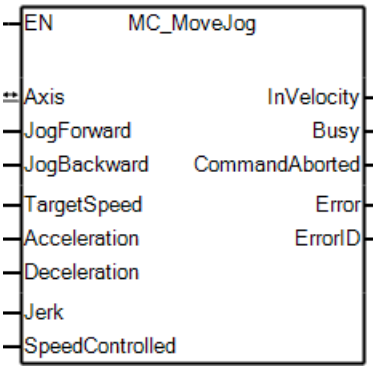
超驰响应

- MC_ImmediateStop 指令不可被任何运动控制指令中止。
- 新的 MC_ImmediateStop 指令可以中止以下运动控制指令：

中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_Home MC_Halt MC_Stop MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_MoveJog MC_CamOut MC_GearOut MC_CamIn MC_GearIn	

14.7.1.12 MC_MoveJog

表格 14-7 MC_MoveJog

LAD/FBD	STL	说明
	**FBCALL MC_MoveJog, INSTANCE, Axis, JogForward, JogBackword, TargetSpeed, Acceleration, Deceleration, Jerk, SpeedControlled, InVelocity, Busy, CommandAborted, Error, ErrorID**	MC_MoveJog 指令按照指定的目标速度点动轴。

表格 14-8 MC_MoveJog 参数

参数	声明	数据类型	初始值	说明
EN	IN	BOOL	-	能流
Axis	IN_OUT	SYS_AXIS	-	引用轴
JogForward	IN	BOOL	-	正向点动，高电平触发
JogBackward	IN	BOOL	-	反向点动，高电平触发
TargetSpeed	IN	REAL	-	目标速度

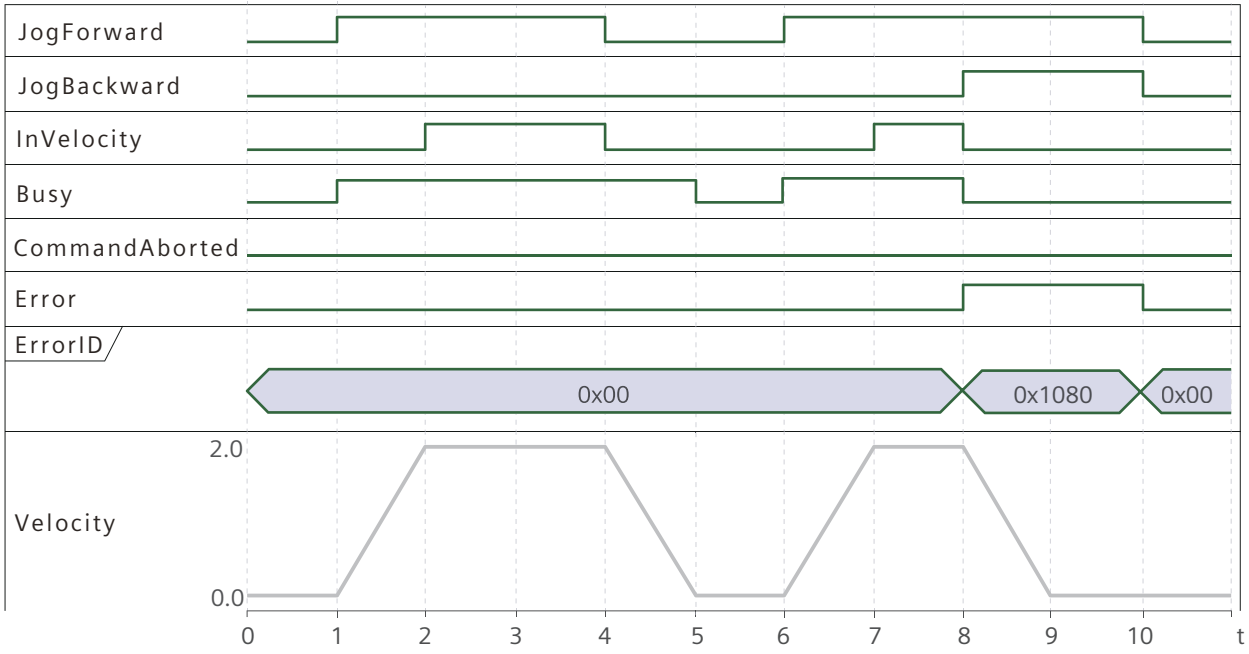
参数	声明	数据类型	初始值	说明
Acceleration	IN	REAL	Axis_Config. Dynamic_Defaul t. Default_Acc	加速度值
Deceleration	IN	REAL	Axis_Config. Dynamic_Defaul t. Default_Decel	减速度值
Jerk	IN	REAL	Axis_Config. Dynamic_Defaul t. Default_Jerk	加加速度值
SpeedControlled	IN	BOOL	0	控制模式 • 0：位置控制模式 • 1：速度控制模式
InVelocity	OUT	BOOL	False	给定速度等于目标速度时为 True
Busy	OUT	BOOL	False	指令正在运行中
CommandAborte d	OUT	BOOL	False	当前指令被另一个指令打断
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1068)

前置条件

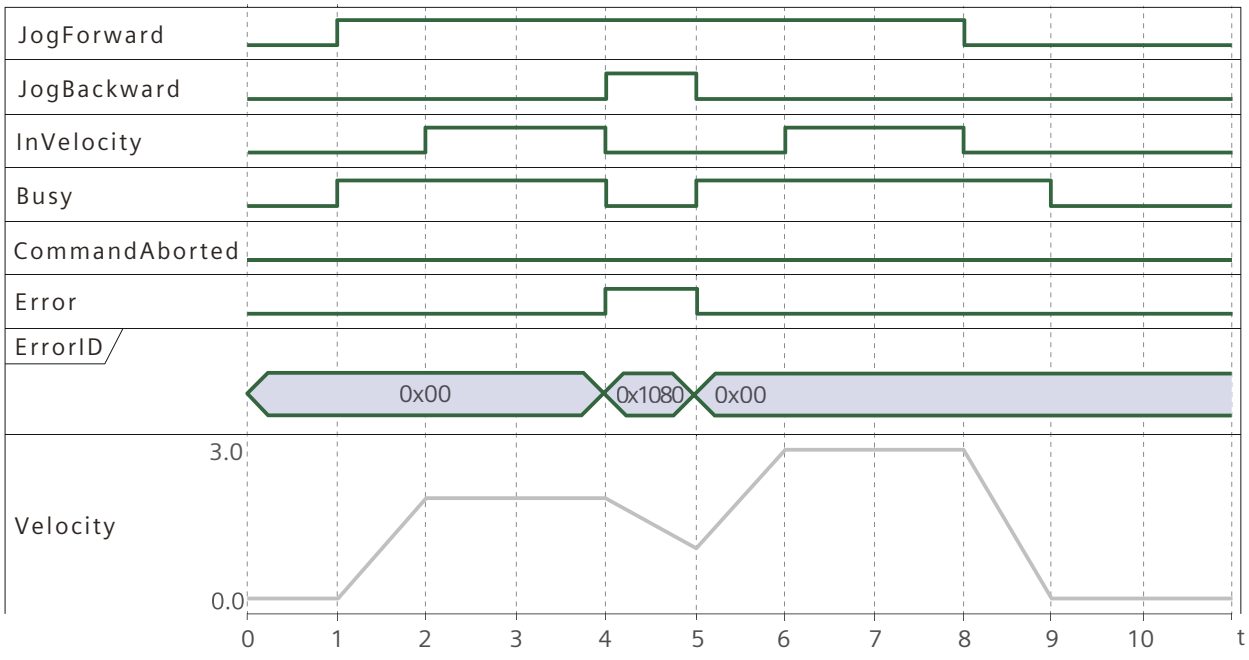
- 从轴已正确组态。
- 轴使能成功。通过用户程序中的 MC_Power 成功使能轴，轴状态机进入 Standstill 状态。

时序图

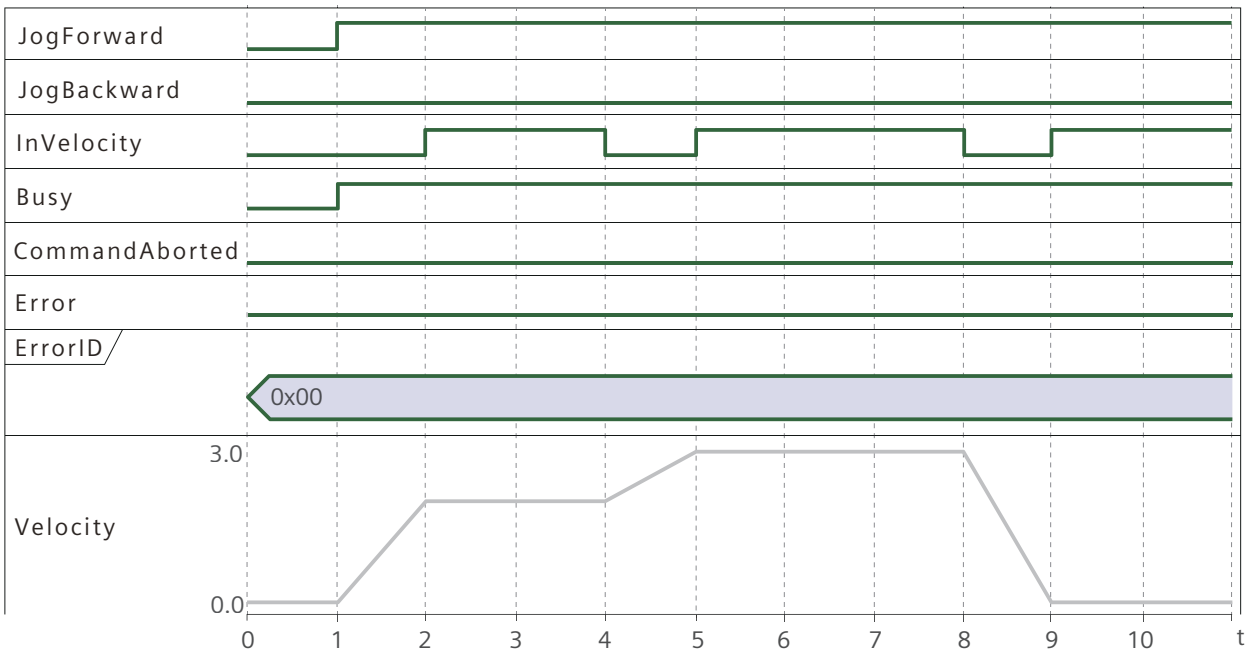
1. 正向点动过程中，同时触发反向点动



2. 0x1080 报错执行点动停止过程中，如果触发新的点动运动，会立即打断停止运动，并按照新的点动参数执行新的点动运动



3. 运动中进行 2 次调整速度


JogForward
JogBackward
InVelocity
Busy
CommandAborted
Error
ErrorID
0x00
Velocity
3.0
0.0
0
1
2
3
4
5
6
7
8
9
10
t

超驰响应

- MC_MoveJog 指令可由以下运动控制指令中止：

中止（**BufferMode = 0**）	缓冲（**BufferMode = 1**）
MC_Halt MC_Stop MC_MoveJog MC_ImmediateStop MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn	

– 新的 MC_MoveJog 指令可以中止以下运动控制指令：

中止（**BufferMode = 0**）	缓冲（**BufferMode = 1**）
MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamOut MC_GearOut MC_CamIn MC_GearIn MC_MoveJog MC_PhasingRelative	

说明

MC_MoveJog 没有缓冲行为，不支持缓冲到其他指令，也不支持被其他指令缓冲。

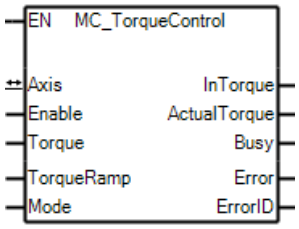
运动中修改输入引脚参数，会打断 MC_MoveJog 进程本身

小心

- 加速过程中修改加速度，如果修改前后加速度变化较大，机械容易出现冲击。
- 点动过程中出现轴错误，清除轴错误后，不会自动恢复点动，需要重新拉低 JogForward/JogBackward 后再次触发高电平。

14.7.1.13 MC_TorqueControl

表格 14-9 MC_TorqueControl

LAD/FBD	STL	说明
	`FBCALL MC_TorqueControl, Instance, Axis, Enable, Torque, TorqueRamp, Mode, InTorque, ActualTorque, Busy, Error, ErrorID`	使用 MC_TorqueControl 指令对 IRT 轴对应的驱动器进行扭矩控制。扭矩数据通过 750 报文传输。

表格 14-10 MC_TorqueControl 参数

参数	声明	数据类型	初始值	说明
EN	IN	BOOL	-	能流
Axis	IN_OUT	SYS_AXIS	-	引用轴
Enable	IN	BOOL	-	• Mode = 0 – 高电平：启用指令 – 低电平：取消指令 • Mode = 1 – 高电平：启用指令，同时启用扭矩控制模式 – 低电平：取消指令，同时关闭扭矩控制模式
Torque	IN	REAL	0.0	可缺省。 • Mode = 0，代表附加扭矩 • Mode = 1，代表设置的扭矩值 有效范围：驱动器参考扭矩的正负 200% 单位： Nm Enable 置 1 后可继续更新。
TorqueRamp [1]	IN	REAL	0.0	每秒扭矩变化。 有效范围：大于等于 0.0 单位： Nm/S Enable 置 1 后可继续更新。

参数	声明	数据类型	初始值	说明
Mode	IN	BYTE	0	指令模式。 • Mode = 0：速度模式，提供附加扭矩 指令执行期间不会启用驱动器的扭矩控制模式（STW1.14=0）。 • Mode = 1：扭矩控制模式 指令执行期间会启用驱动器的扭矩控制模式（STW1.14=1）。 Enable 置 1 后更新无效。
InTorque	OUT	BOOL	False	设定扭矩达到目标扭矩设定值。
ActualTorque	OUT	REAL	0.0	Enable 置 1 时为实际扭矩，Enable 置 0 时为 0.0
Busy	OUT	BOOL	False	指令正在运行中
Error	OUT	BOOL	False	指令发生错误
ErrorID [2]	OUT	WORD	0	错误代码 (页 1068)

[1] **注**：初始值为 0.0 时，Torque 值直接发送给驱动器，这种情况下可由用户决定 Torque 如何变化。初始值不为 0.0 时，从当前 Torque 以 TorqueRamp 逐步变化到目标 Torque。

[2] **注**：当 MC_TorqueControl 执行过程中报错，会将驱动器的扭矩恢复为 0.0。如果处于 Mode = 1 的扭矩控制模式，则会退出扭矩控制模式。

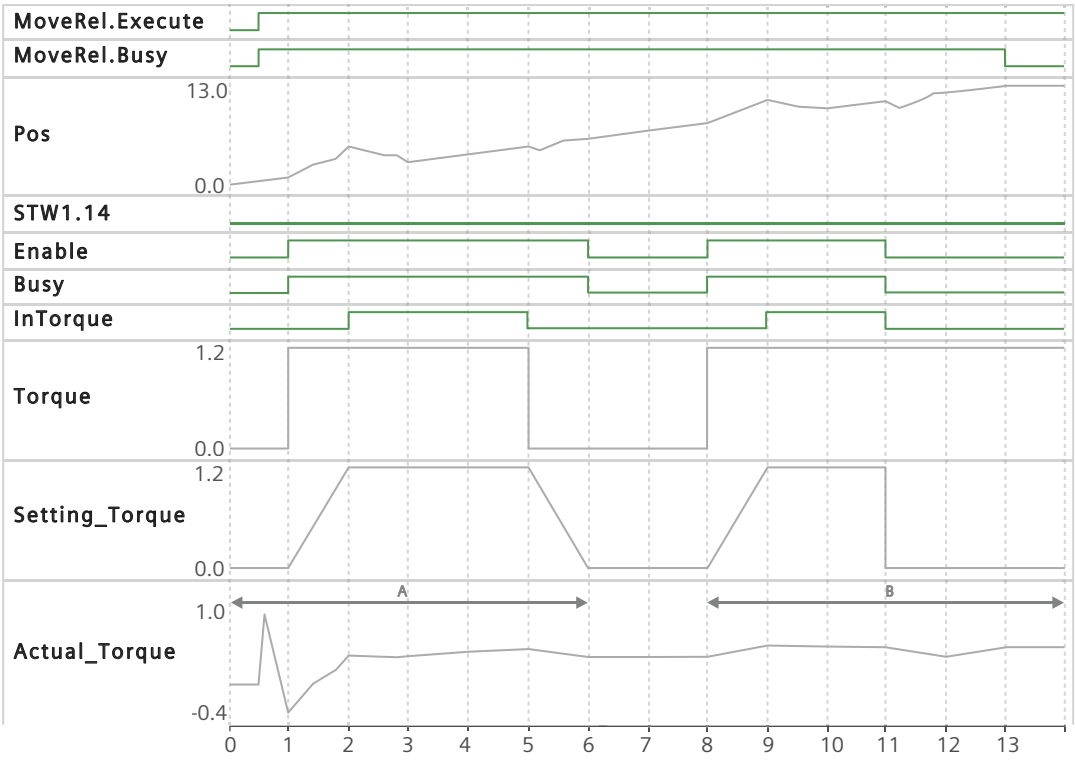
前置条件

（Mode = 0 或 Mode = 1）：

- IRT 轴正确配置
- 750 报文已经配置
- 轴对象中：
  - Drive Ready 为 1
  - Enable 为 1
  - 无 Error
- 轴 Power 成功

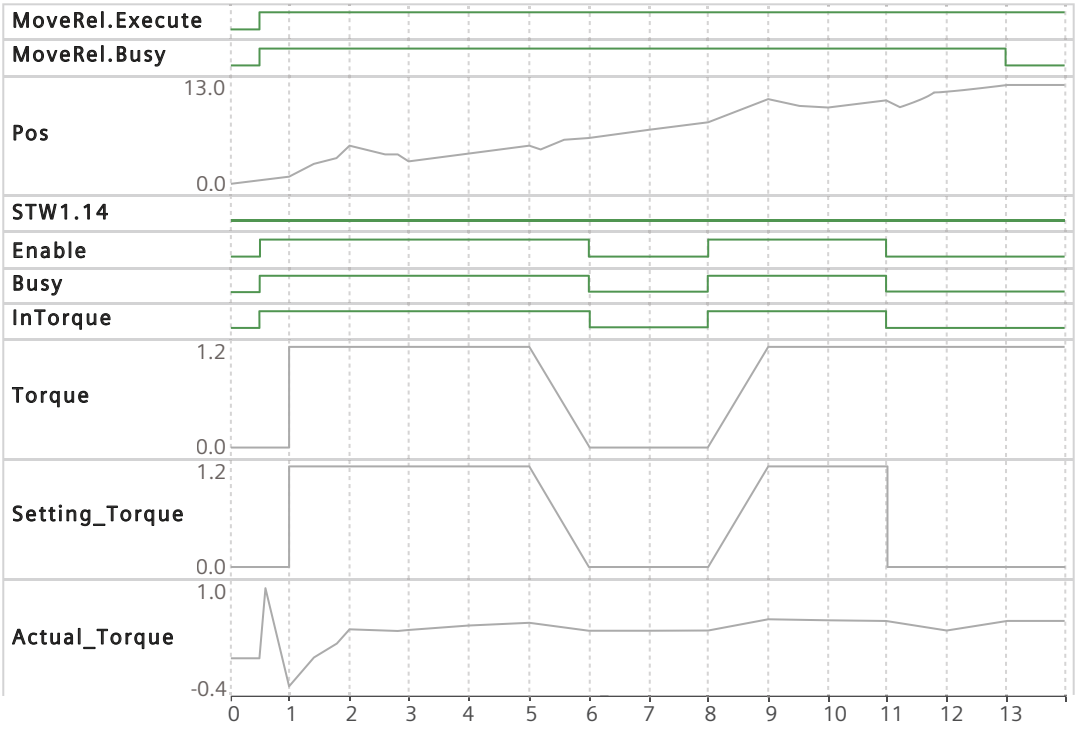
时序图

(1) 激活/取消附加扭矩（Mode=0，TorqueRamp>0.0）



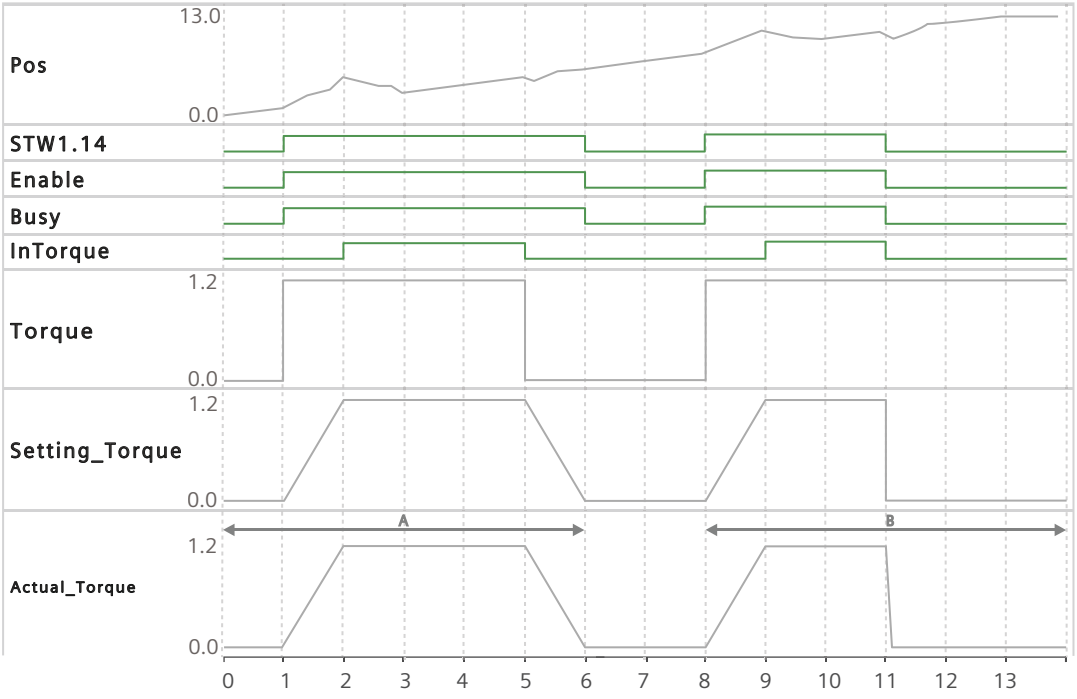
- 过程 A 中，用户通过在 1 时刻 Enable = TURE, 同时将 Torque 值改为 1.2。此时设定 Torque 值会根据 TorqueRamp 逐步从 0.0 增加到 1.2。
  同时由于没有开启扭矩模式，那么定位指令也可以执行。由于轴此时在运动，那么实际 Torque 就不为 0，不过会受到附加扭矩的影响，实际 Torque 会短暂变化。同时轴的位置会跳跃一下，之后因为还是位置闭环为主，所以会恢复到规划的位置。
  在 5 时刻，用户将 Torque 的值改为 0.0，此时设定扭矩值会根据 TorqueRamp 逐步减小到 0.0，由于扭矩的变化，位置同样会受到影响，只不过之后恢复到规划位置。
- 过程 B 中，用户通过在 8 时刻 Enable = TRUE，同时将 Torque 的值改为 1.2，此过程和 A 一致。然后在时刻 11 将 Enable 置为 FALSE，那么附加扭矩 0.0 会直接发给驱动器。位置的变化规律和过程 A 一致。

(2) 激活/取消附加扭矩（Mode=0，TorqueRamp=0.0）



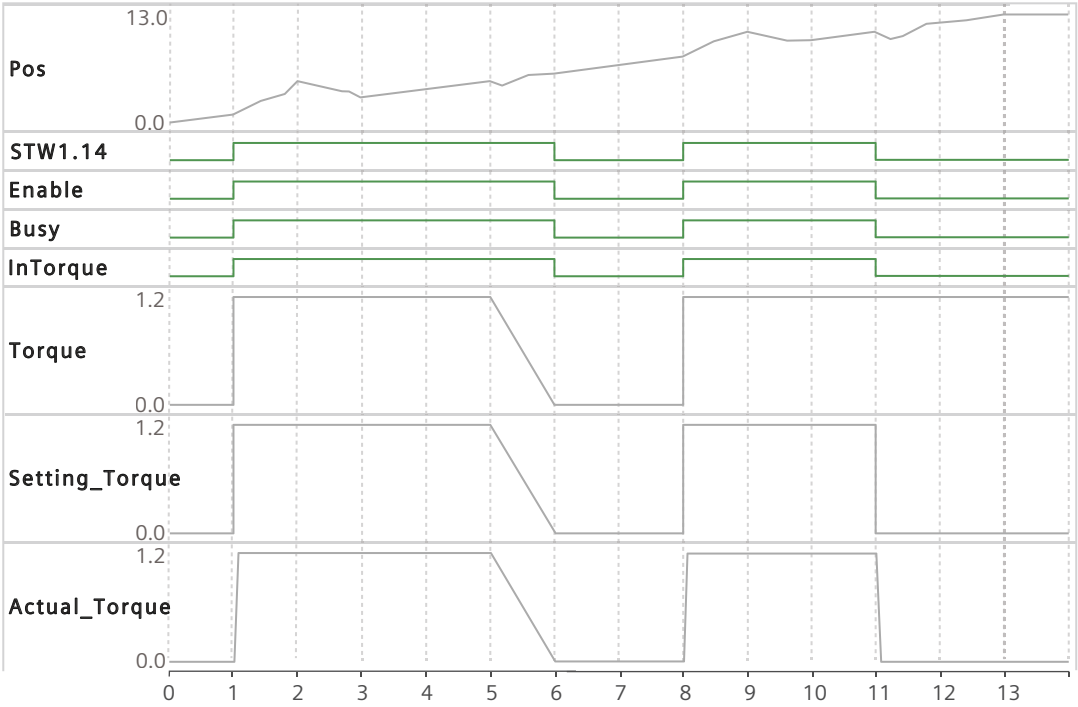
这个场景中，运动的行为和（1）中一致，变化在于 InTorque 管脚。因为 TorqueRamp=0.0，所以指令会直接把 Torque 管脚上的值给到驱动器没有 Ramp 过程，也就是设定扭矩值和目标值一致，那么 InTorque 管脚就为 1。其中 5 到 6 时刻和 8 到 9 时刻是用户写程序使得 Torque 管脚上的值产生变化，而不是 TorqueRamp 带来的。

(3) 激活/取消扭矩控制（Mode=1，TorqueRamp>0.0）



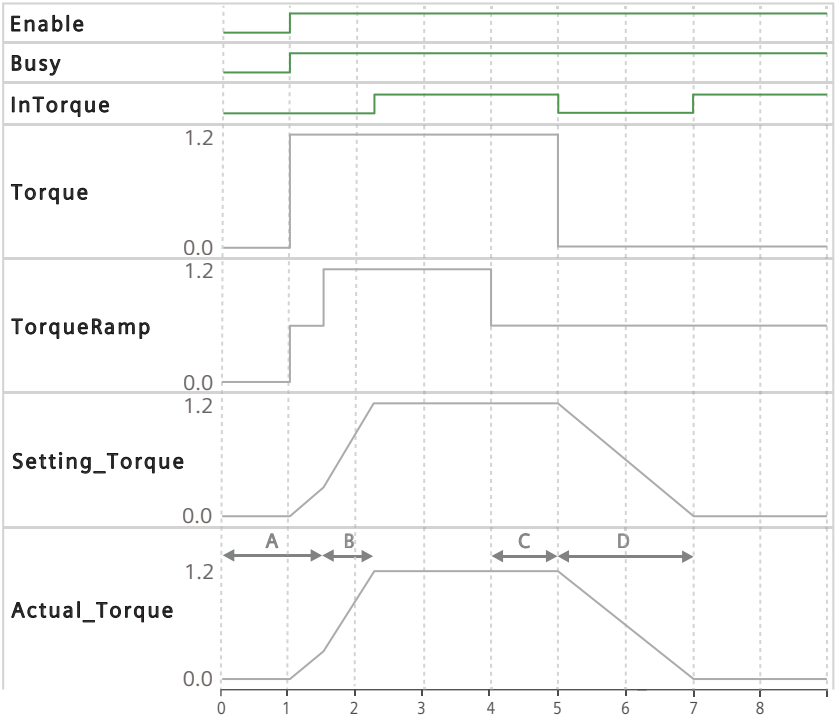
- 过程 A 中，用户通过在 1 时刻 Enable= TRUE，同时将 Torque 的值改为 1.2。此时设定扭矩值会根据 TorqueRamp 逐步从 0.0 增 加到 1.2，当设定扭矩达到目标值时 InTorque 置 1。由于开启了扭矩模式，实际扭矩会跟着设定扭矩变化。需要注意的是，此时位置的变化是因为扭矩产生的，和运动指令无关。
  在 5 时刻，用户将 Torque 的值改为 0.0，此时设定扭矩值会根据 TorqueRamp 逐步减小到 0.0，同样实际扭矩会跟着变化。
- 过程 B 中，用户通过在 8 时刻 Enable= TRUE，同时将 Torque 的值改为 1.2，此过程和 A 一致。然后在时刻 11 将 Enable 置为 FALSE，那么附加扭矩 0.0 会直接发给驱动器。位置的变化规律和过程 A 一致。

(4) 激活/取消扭矩控制（Mode=1，TorqueRamp=0.0）



这个场景定义的现象和（2）类似，除了 Torque 的实际值会紧跟设定值变化，因为此时处于扭矩控制模式。

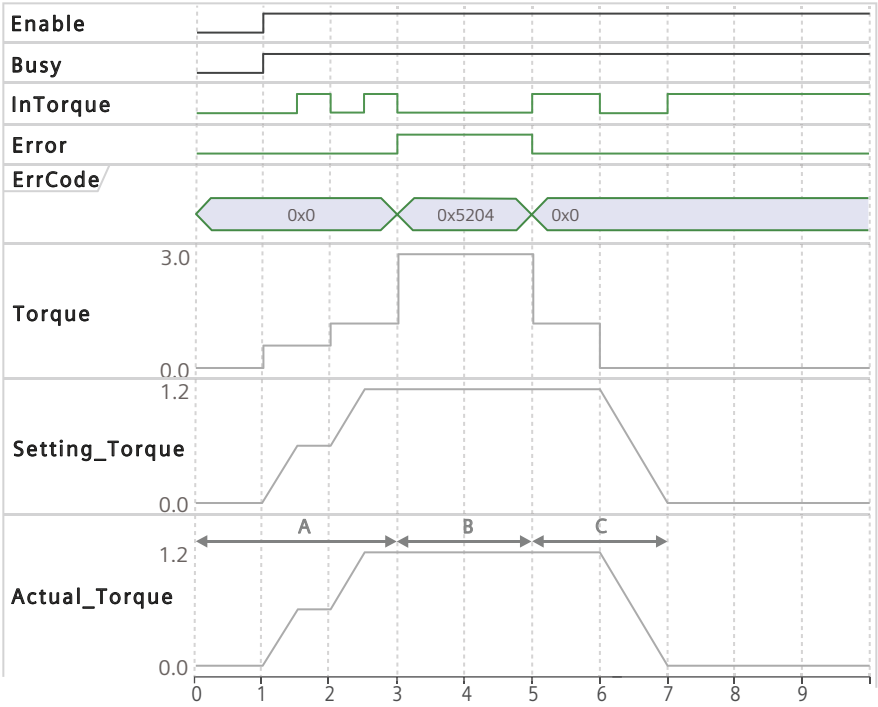
(5) 持续更新 TorqueRamp



以 Mode=1，Torque=1.2 为例。

- 过程 A 中，启动了 TorqueControl 指令，在 1 时刻设置 Ramp 为 0.6，Torque 设定值按照 0.6 的斜率上升。
- 过程 B 中，在 1.5 时刻用户将 Ramp 设置为 1.2， Torque 设定值从此开始按照 1.2 的斜率上升，在图中可以看到斜率的变化。
- 过程 C 中，由于用户没有修改目标 Torque，所以修改 Ramp 到 0.6 不会产生效果。
- 过程 D 中，由于用户修改目标 Torque 到 0.0，所以 Torque 设定值开始按照 0.6 的斜率下降，直到 0.0。

(6) Torque 等参数更新出错的报错行为



以 Mode=1，TorqueRamp=1.2 为例，驱动器参考扭矩为 1.2，扭矩范围 200% 。

- 过程 A 中，用户设定的 Torque 管脚值在范围内，所以正常运行。
- 过程 B 中，第 3 时刻，用户设置的 Torque 管脚值超过范围，此时报错，同时保持之前已经设置的 Torque 值。
- 过程 C 中，第 5 时刻，用户重新将 Torque 管脚值设置到范围内，管脚报错消失，正常运行。

使用说明

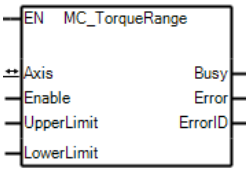
- 指令打断行为
  - Mode = 0
    MC_TorqueControl 不会打断任何正在执行的运动指令。任何运动指令也不会导致 MC_TorqueControl 的执行被中止。此模式下， MC_TorqueControl 和运行指令可以共存。
    管理指令的 MC_Power 的 power off 操作会中止运行中的 MC_TorqueControl。此时 Torque 设定值会被立即置为 0.0。
    轴报错（限位、驱动器报错等）都会中止运行中的 MC_TorqueControl。此时 Torque 设定值会被立即置为 0.0。
  - Mode = 1
    MC_TorqueControl 不会打断任何正在执行的运动指令。如果执行 MC_TorqueControl 时有正在运行的运动指令，那么 MC_TorqueControl 会报错。
    处于扭矩控制模式时（STW1.14=1）任何运动指令都不能执行。执行时会报错。
    管理指令的 MC_Power 的 power off 操作会中止运行中的 MC_TorqueControl。此时 Torque 设定值会被立即置为 0.0，同时会从扭矩控制模式中退出（ STW1.14=0）。
    轴报错（限位、驱动器报错等）都会中止运行中的 MC_TorqueControl。此时 Torque 设定值会被立即置为 0.0，同时会从扭矩控制模式中退出（ STW1.14=0）。
- 指令缓冲行为
  - Mode = 0
    指令启动后，附加在当前运动指令上，不会对当前运动指令有 buffer/abort 行为；并且当前运动指令后面执行的运动指令的 buffer/abort 行为，还是保持原样。
  - Mode = 1
    MC_TorqueControl 不能缓冲在其他指令之后； MC_TorqueControl 的后面也不能缓冲其他指令。
- PLCOpen 状态机
  - Mode = 0
    MC_TorqueControl 执行中，不引起状态机变化。
  - Mode = 1
    MC_TorqueControl 执行中，轴处于 Continuous 状态。
    用户正常停止 MC_TorqueControl 指令，轴处于 Standstill 状态。
    轴报错，轴处于 error stop 状态。

说明

当前已经激活 MC_ TorqueControl 情况下，执行新的 MC_ TorqueControl ，不会中止之前的，新的 MC_ TorqueControl 将报错。

14.7.1.14 MC_TorqueRange

表格 14-11 MC_TorqueRange

LAD/FBD	STL	说明
	**FBCALL MC_TorqueRange,INSTANCE , Axis, Enable, UpperLimit, LowerLimit, Busy, Error, ErrorID**	该指令用于设置驱动器实时扭矩的上下限值。该功能通过 SIEMENS 附加报文 750，将扭矩的上下限数值经 PROFIdrive 实时传输到驱动器。

参数	声明	数据类型	初始值	说明
EN	IN	BOOL	-	能流
Axis	IN_OUT	SYS_AXIS	-	引用轴
Enable	IN	BOOL	False	电平触发，Drive_Ready 置位为 1 后： • True：指令中的扭矩上下限值将实时写入驱动器，实现自定义扭矩限制。 • False：恢复使用默认扭矩上限和下限默认值，放弃本指令的限值设置。
UpperLimit	IN	REAL	1.0 E6	扭矩上限值，可缺省，有效值范围为[0, 200]，表示基准扭矩的百分比。超范围时，实际生效值为 2 × 扭矩基准值。 上限值不能为负数，表示电机正向转动。当 Enable 置位时，该参数可实时更新。
LowerLimit	IN	REAL	1.0 E6	扭矩下限值，可缺省，有效值范围为[-200, 0]，表示基准扭矩的百分比。超范围时，实际生效值为 -2 × 扭矩基准值。 下限值不能为正数，表示电机反向转动。当 Enable 置位时，该参数可实时更新。
Busy	OUT	BOOL	False	指令正在运行中
Error	OUT	BOOL	False	表示指令发生了错误
ErrorID	OUT	WORD	16#0000	错误代码 (页 1068)

说明

扭矩基准值说明

扭矩基准值优先从驱动器获取。如果无法获取到驱动器的扭矩基准值，且您在 STEP 7-Micro/WIN SMART V3 中勾选了“使用离线值”，则采用您在 STEP 7-Micro/WIN SMART V3 中配置的扭矩基准值。

如果您已勾选“使用离线值”，但配置的离线扭矩基准值与驱动器获取的在线扭矩基准值不一致，系统仍会使用驱动器的在线值，并触发轴警告 Axis_Status.Config_Warning。

若未能获取在线扭矩基准值，且用您勾选“使用离线值”，则会因配置错误导致无法置位驱动器为 Ready 状态。

时序图

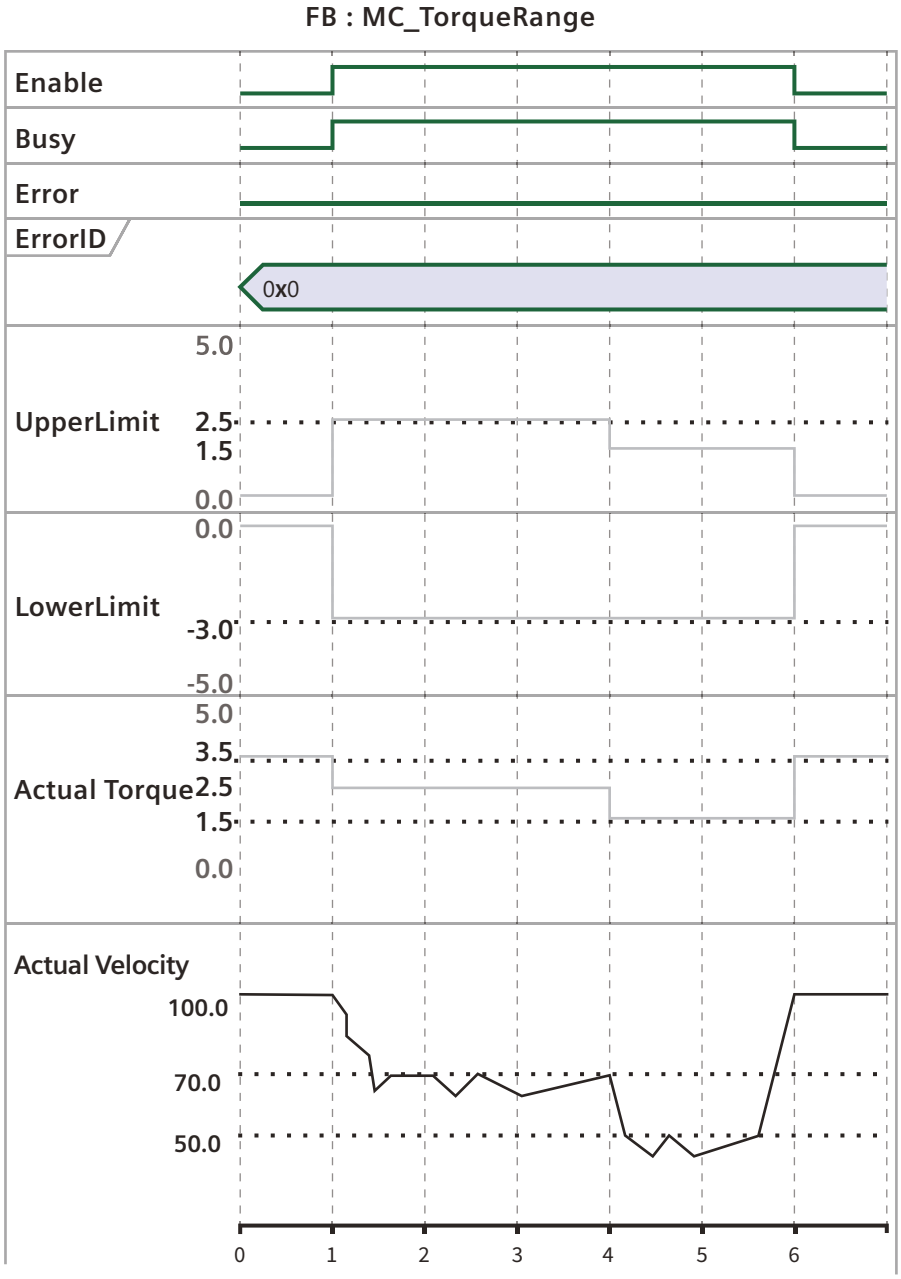
说明

运行过程中如参数设置错误，Error 引脚将置位为 1。再输入正确参数，Error 引脚会清零，并以新的正确参数生效。

说明

为了便于理解和对比，所有时序图中的 UpperLimit 和 LowerLimit 都被转换成了 Nm 的扭矩单位，实际的 FB 管脚使用的参考扭矩的百分比。

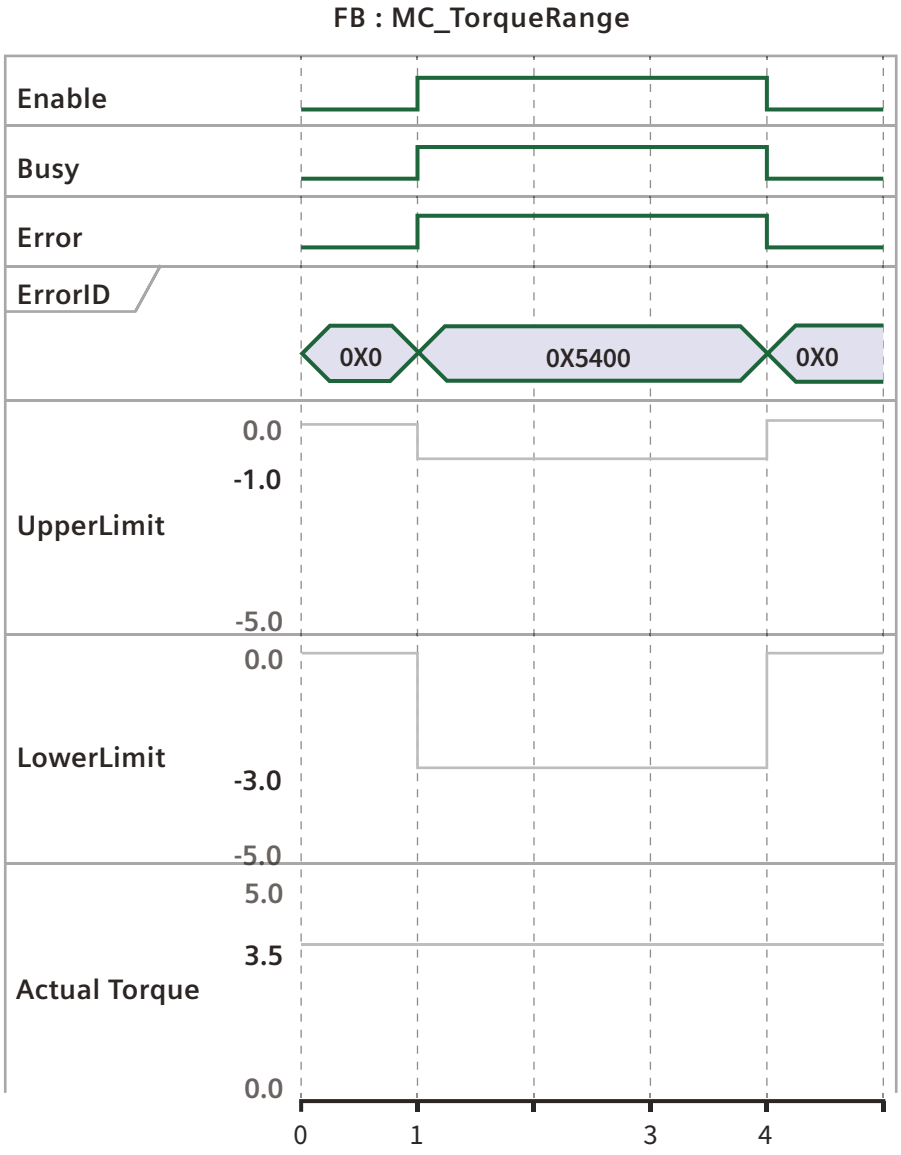
- 在电机运行并存在扭矩（如 InVelocity）时，采用电平触发方式使能 MC_TorqueRange，并设置正确的参数后，可实现扭矩限值的设定。在运行过程中，仍可实时更新扭矩限值，新的限值将立即生效。



说明

注：此处的 Actual Torque 仅表示受限值影响的一个峰值。实际扭矩为驱动器持续尝试降低扭矩过程中产生的波动值，并非固定不变。

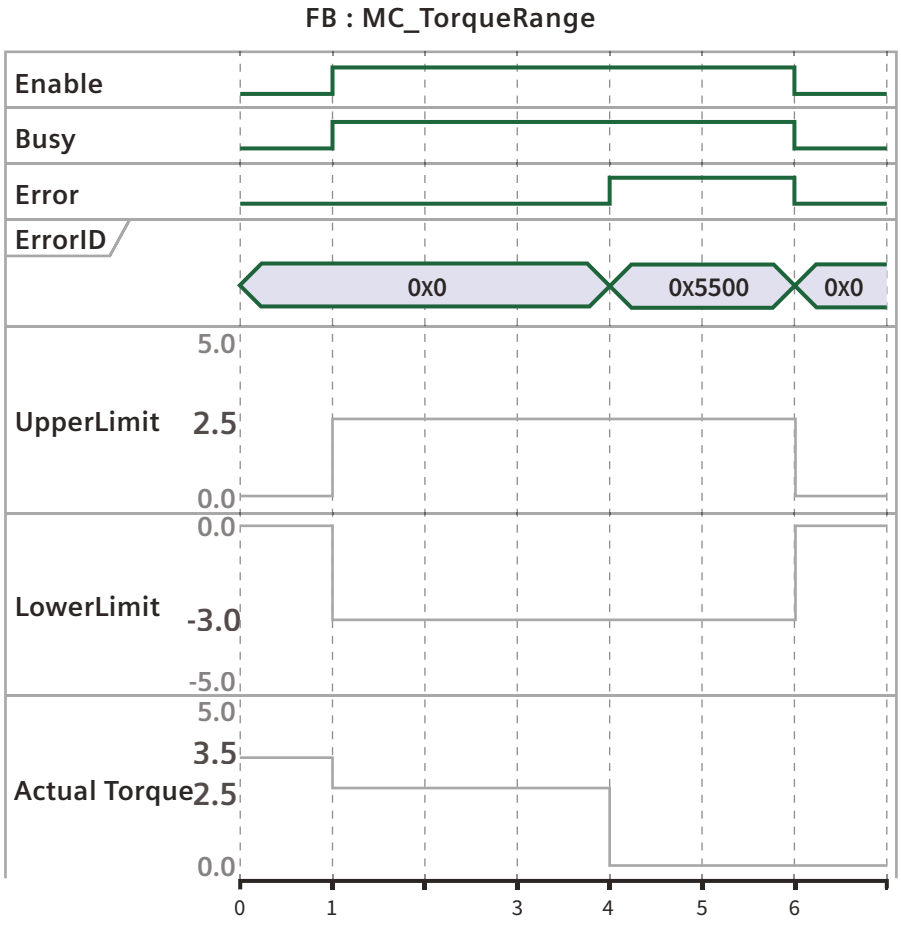
- 在电机运行并存在扭矩（如 InVelocity）时，采用电平触发方式使能 MC_TorqueRange，并设置扭矩限值，若设置非法参数，Error 引脚将置位为 1，当前扭矩限值设置不会生效。



- 在电机运行并存在扭矩（如 InVelocity）时，采用电平触发方式使能 MC_TorqueRange，并参数设置正确；若在运行过程中发生通信错误，Error 引脚将置位，扭矩限值控制会中断。

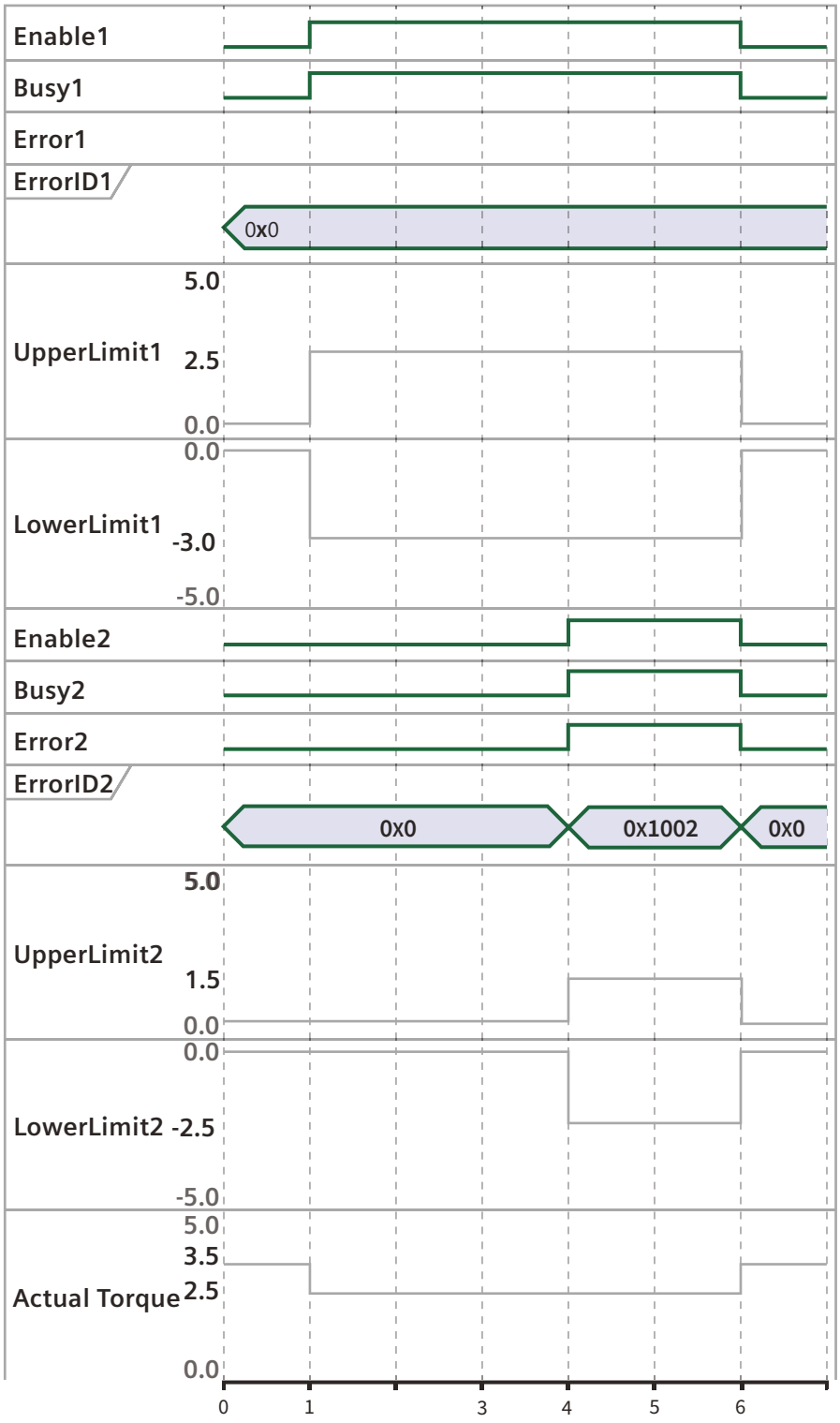
说明

发生通信错误时，系统将不再关注驱动器的实际扭矩值。通常情况下，驱动器会因故障停机，电机实际扭矩为 0.0。



- 在电机运行并存在扭矩（如 InVelocity）的情况下，通过电平触发使能第一个 MC_TorqueRange，并正确设置参数。运行过程中，若第二个 MC_TorqueRange 实例再次被触发，第二个报错。

FB1 : MC_TorqueRange1 - FB2 : MC_TorqueRange2



注意事项

- 指令打断与缓冲行为（超驰响应）：
  - 其它任何运动控制指令不会中止"MC_TorqueRange"的执行。
  - 新的"MC_TorqueRange"指令不会中止任何当前激活的运动控制指令。
  - 同一时间仅允许存在一个 MC_TorqueRange 指令实例，若再次激活，会报错。
- 跟随误差监控与参数限制：
  使用 MC_TorqueRange 会影响跟随误差监控，您需在 STEP 7-Micro/WIN SMART V3 中取消勾选"启用跟随误差监视"。对于容差检测和 Done 置位的超时，您须自行调整 STEP 7-Micro/WIN SMART V3 中的相关参数。
- 轴与驱动器相关注意事项：
  - MC_TorqueRange 为基于轴的指令，不同轴可独立使用各自的 Range 指令。
  - 您需提需提前设置驱动器堵转参数，例如 S200 驱动器的 P2175 参数设为合理值，否则会报错。
- 参数错误与恢复行为: 如运行中参数错误，将置位 Error 引脚，值恢复为默认。参数改为正确时，Error 引脚自动清零，新的参数立即生效。

14.7.1.15 MC_TorqueLimiting

通过运动控制指令 MC_TorqueLimiting，用户可以激活以下功能并分配参数：

- 扭矩/力（Torque/Force）限制
- 固定停止（Fixed Stop）检测

您可在轴配置中设定默认的百分比限制值。

MC_TorqueLimiting 指令支持在运动任务启动前或任务执行过程中进行激活和禁用。

说明

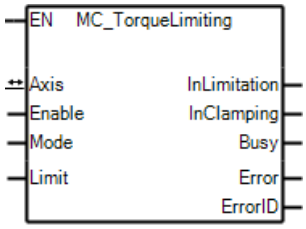
如果当前已有 MC_TorqueControl 或 MC_TorqueRange 等扭矩控制指令正在执行，则不允许调用该指令。

说明

该指令要求驱动器全面支持西门子 105 报文。

参数

下表列出了运动控制指令 MC_TorqueLimiting 的参数：

LAD/FBD	STL	说明
	`FBCALL MC_TorqueLimiting,INSTANCE, Axis, Enable, Mode, Limit, InLimitation, InClamping, Busy, Error, ErrorID`	MC_TorqueLimiting 指令提供以下功能： • 激活并配置扭矩/力限制 • 激活并监控固定停止检测

参数	声明	数据类型	初始值	说明
EN	IN	BOOL	-	能流
Axis	IN_OUT	SYS_AXIS	-	引用轴
Enable	IN	BOOL	-	开启/关闭输入参数 Mode 对应的功能 • TRUE: 启动指令 • FALSE: 尝试关闭指令 [1]
Mode[2]	IN	BYTE		• 0 ：扭矩/力限制 • 1： 固定停止检测
Limit [3]	IN	REAL	0.0	扭矩/力限制值，以百分比形式（%）进行配置。 • 当参数值在有效范围 [0, 100] 内时： 系统将采用该参数指定的值作为限制。 • 当未明确指定参数时： 系统将自动使用在 “Torque Limit or Fixed Stop Detection”配置窗口中预设的默认值。 相关数据标签： • 扭矩/力限制：<TO>.Torque_Lmt.Torque_Force 若提供的参数值超出有效范围 [0, 100]，系统将报告“参数非法”错误。
InLimitation	OUT	BOOL	False	Mode = 0 或 1 时：驱动器输出的实际扭矩大于等于扭矩阈值 2。
InClamping	OUT	BOOL	False	Mode = 1 ：驱动保持在固定停止（夹紧）。轴位置在定位容差范围内。
Busy	OUT	BOOL	False	指令正在运行中

参数	声明	数据类型	初始值	说明
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1068)

[1] 在 InClamping 状态下，不允许关闭指令

[2] 不允许在指令运行过程中动态修改。指令启动成功后，切换模式将被忽略。

[3] 允许在指令运行过程中动态修改。

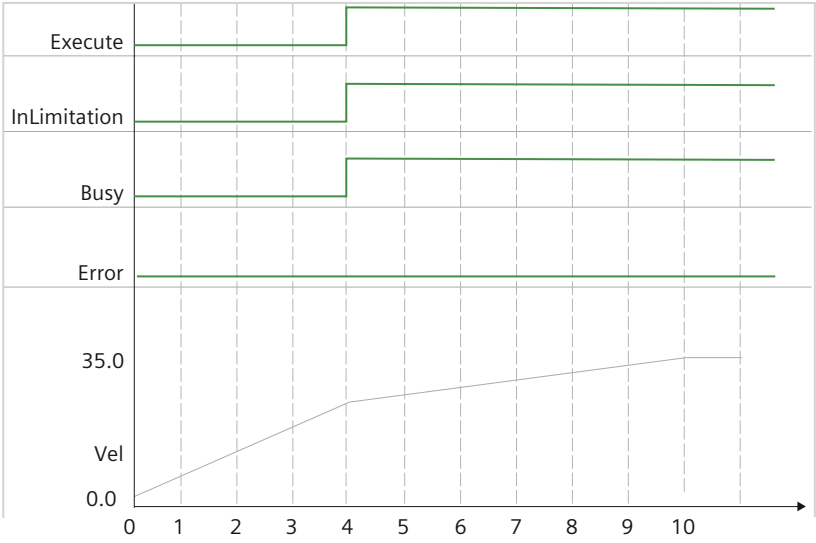
前提条件

- 轴已正确组态。
- 控制闭环已正确组态

扭矩限制模式

该模式下，指令用于实时控制驱动器扭矩上限，并监测实时扭矩是否达到设定阈值

- 加速运动过程中使能扭矩限制，运动加速度受到限制。

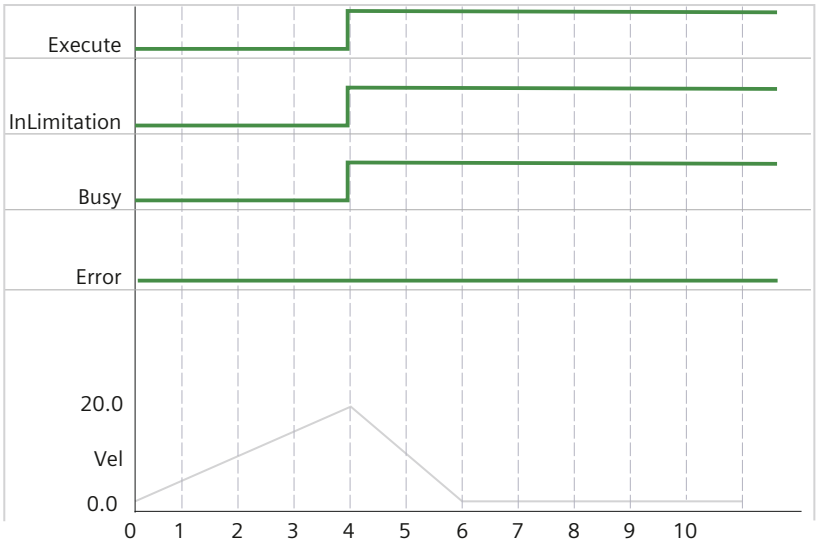


- 加速运动过程中，使能扭矩限制，扭矩不能克服摩擦力。此时，轴将无法继续加速，反而会开始减速，直至完全停止。

警告

“飞车”现象

在此减速过程中，如果立即解除扭矩限制，这可能导致轴发生不受控制的加速，即“飞车”现象（位置模式）。



固定停止模式

该模式下，该指令用于监测固定挡块，并且使轴以设定扭矩值夹紧至固定挡块处。监测并监测实时扭矩是否达到设定阈值以及轴是否处于夹紧状态。

说明

该模式需要在电机位置环生效，若电机工作在非位置环时，如扭矩控制模式时，该模式不生效。

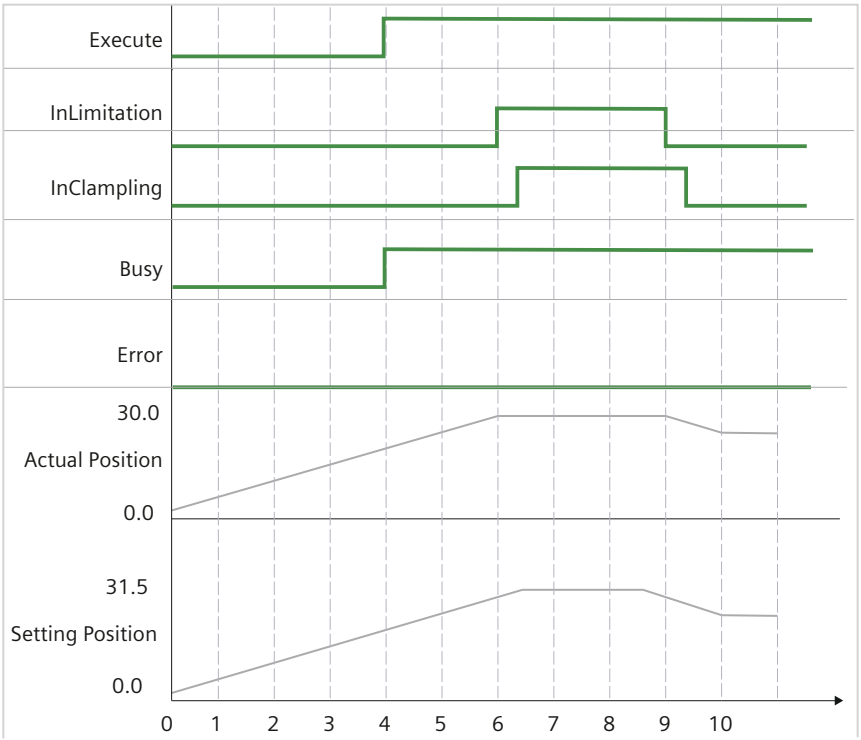
当轴在正常运行或预期操作中接触到机械挡块时，轴会停止，然后进入 InClamping 状态；如果此时启动反向指令，伴随反向运动的启动，InClamping 状态会自动解除。

使能指令前您需要在 Torque Limit or Fixed Stop Detection 窗口中，组态 Fixed Stop detection 相关配置 (页 891)。

异常脱离说明 (页 892)

说明

当设定位置在位置容差内，异常脱离无法被检测。



说明

当轴进入 InClamping 状态时，如果 Enable 信号被置为 False，FB 实例将触发一个错误。

状态保持与错误处理：

- 在此过程中，InLimitation、InClamping 和 Busy 等状态将保持其原有状态。
- 系统将报告相应的 Error 和 ErrorID。
- 值得注意的是，尽管有错误报告，但轴并不会进入 ErrorStop 状态，因此无需进行复位操作。

14.7.2 凸轮和齿轮指令

14.7.2.1 MC_CamIn

本指令可使相位（主轴）和位移（从轴）按照凸轮表进行同步凸轮动作。由本指令指定的凸轮表，必须事先使用凸轮编辑器制作，并下载到 PLC 中。在 PLC 运行过程中修改凸轮表，请参考 MC_GenerateCamTable（页 1051）。

在 Execute 的上升沿，本指令启动。

本指令对不同主轴和从轴的类型支持如下表：

从轴	主轴			
	IRT	**PTO**	**Virtual**	**Encoder**
IRT	✓	-	✓	✓
PTO	-	✓	✓	✓
Virtual	✓	✓	✓	✓

说明

PTO 和 IRT 无法严格同步，如果使用 Virtual 实现 Cam 联级，例如，IRT → Virtual → PTO，也无法保证 IRT 和 PTO 的同步关系

说明

从轴编号必须大于主轴编号。

说明

同步运动时，从轴变量中的 Setting_Info.Jerk 一直显示为 0。

说明

在配置凸轮主轴相位时，确保其值足够大，以避免同步效果不佳。

比如在如下配置中，在一个插补周期内，主轴的位移恰好等于凸轮的主轴总相位 1mm。这意味着从轴需要在 1ms 内完成整个凸轮曲线定义的运动。这对于大多数实际应用来说，是极难实现良好同步的。

- 凸轮主轴总相位： 1mm
- 运动插补周期： 1ms
- 主轴速度： 1000mm/s

警告

同步轴速度突然变化的潜在危险

如果在启动或执行同步操作时，主轴的速度发生急剧变化，则从轴的动作也将急剧变化这可能导致设备受到过大的力。

使用 MC_CamIn 时，尽量不要在主轴上执行 MC_Home 或 MC_ImmediateStop 指令。在主轴上执行 MC_Home/MC_ImmediateStop 指令可能导致从轴快速跟踪，这种情况十分危险。

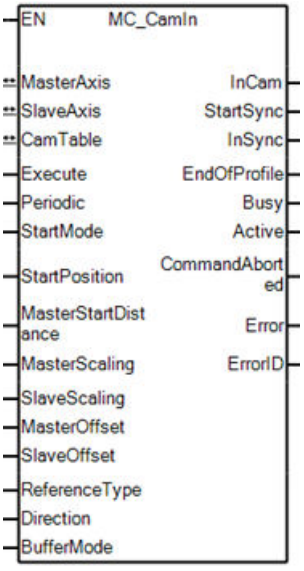
如果希望在主轴上使用 MC_Home/MC_ImmediateStop 指令，在执行该指令之前解除主轴与从轴之间的关系。

说明

组态 EndofProfile 引脚时，请注意以下几点：

- 对于非周期凸轮，在凸轮周期结束时， Endofprofile 将置位为 **TRUE** 并保持。
- 对于周期凸轮，在每个凸轮周期结束时，将置位 1 个扫描周期。
- PTO: EndOfProfile 相对于凸轮表终点最大提前 24 ms。
- IRT: EndOfProfile 相对于凸轮表终点不会提前，滞后最大不超过 2 个 IRT 周期。

参数

LAD/FBD	STL	说明
	**FBCALL MC_CamIn, INSTANCE, MasterAxis, SlaveAxis, CamTable, Execute, Periodic, StartMode, StartPosition, MasterStartDistance, MasterScaling, SlaveScaling, MasterOffset, SlaveOffset, ReferenceType, Direction, Buffermode, InCam, StartSync, InSync, EndOfProfile, Busy, Active, CommandAborted, Error, ErrorID**	MC_CamIn 指令通过使用指定凸轮表来启动凸轮运行。

参数	声明	数据类型	初始值	说明
MasterAxis	IN_OUT	SYS_AXIS	-	引用主轴
SlaveAxis	IN_OUT	SYS_AXIS	-	引用从轴
CamTable	IN_OUT	SYS_CAM	-	引用凸轮表
Execute	IN	BOOL	-	上升沿处开始指令
Periodic	IN	BOOL	False	指定是周期执行指定凸轮表还是仅执行一次。 • TRUE：周期 • FALSE：非周期
StartMode [1]	IN	WORD	0	指定 MasterStartDistance 的坐标。 • 0：绝对位置 • 1：相对位置

参数	声明	数据类型	初始值	说明
StartPosition	IN	REAL	0.0	指定凸轮表的起点（0 相位）
MasterStartDistance [1]	IN	REAL	0.0	指定轴开始凸轮运动时主轴的位置
MasterScaling	IN	REAL	1.0	以指定的比例放大/缩小主轴的相位。
SlavingScaling	IN	REAL	1.0	以指定的比例放大/缩小从轴的位移。
MasterOffset [1]	IN	REAL	0.0	主轴表的偏移量，主轴至凸轮的角偏移量。
SlaveOffset [1]	IN	REAL	0.0	从轴表的偏移量，锐化的凸轮（即更高的凸起和更深的凹陷）。
ReferenceType [2]	IN	BYTE	0	指定主轴的位置类型。 • 0：指令位置（同一任务周期下的计算值） • 1：反馈位置（同一任务周期下获取的主轴反馈位置） 当使用 Reference Type = 1 的非周期凸轮时，必须设置滞后值 (页 904)。这是为了防止由于 actual position 的波动，导致 CamIn 触发 EOP，引起凸轮同步退出。 • 2：指令位置（上个任务周期下的计算值）
Direction	IN	BYTE	0	主轴沿指定方向移动时，从轴凸轮移动。仅当主轴按指定方向反向移动时，从轴凸轮才会停止。 • 0：正方向 • 1：负方向 • 2：未指定方向
BufferMode	IN	BYTE	0	指定执行多个运动指令时的行为 • 0：打断模式 • 1：缓冲模式
InCam	OUT	BOOL	False	执行凸轮表起点时为 TRUE
StartSync	OUT	BOOL	False	从轴追赶主轴的行为 • 0：从轴没有追赶主轴 • 1：从轴开始追赶主轴，还未追上（未进入同步状态）
InSync	OUT	BOOL	False	从轴与主轴同步时为 TRUE

参数	声明	数据类型	初始值	说明
EndOfProfile	OUT	BOOL	False	执行凸轮表终点时为 TRUE
Busy	OUT	BOOL	False	指令未结束
Active	OUT	BOOL	False	表示该指令可对轴进行控制
CommandAborted	OUT	BOOL	False	Command 被另一个命令中止
Error	OUT	BOOL	False	表示指令内发生了错误
ErrorID	OUT	WORD	0	错误代码 (页 1074)

[1] PTO: 这四个参数的输入的值不会生效，使用默认参数。StartMode 的默认值为 1（相对运动模式），而其它参数的默认值为 0.0。

[2] 当从轴为 PTO 时，不支持设置此参数，默认为 0。当从轴为虚轴时，设置为 1 等同于设置为 0。

说明

对于使用实际值作为参考类型（ReferenceType=1）的单向凸轮，建议您在主轴“实际值推断”页面中设置合适的滞后值。

前置条件

- 主从轴已正确组态
- 通过用户程序中的 MC_Power 成功使能从轴。

启动凸轮移动

发出启动命令后，等待主轴到达 StartPosition（凸轮表的起始位置）。
当主轴通过 StartPosition 时，执行凸轮表的起始点，输出变量 InCam（凸轮移动进行中）变为 1。

凸轮表的相位和位移被指定为相对于零点的相对值。因此，每个轴在每个相位的绝对位置都是基于凸轮表每个轴的起点绝对位置的相对值。例如，主轴计数模式为 0 ~ 360° 的旋转模式，StartPosition 为 60°。如下图所示，主轴的绝对位置为凸轮表的相位加上 StartPosition 的值，从轴的绝对位置为凸轮表的位移加上凸轮表起点处从轴的绝对位置。

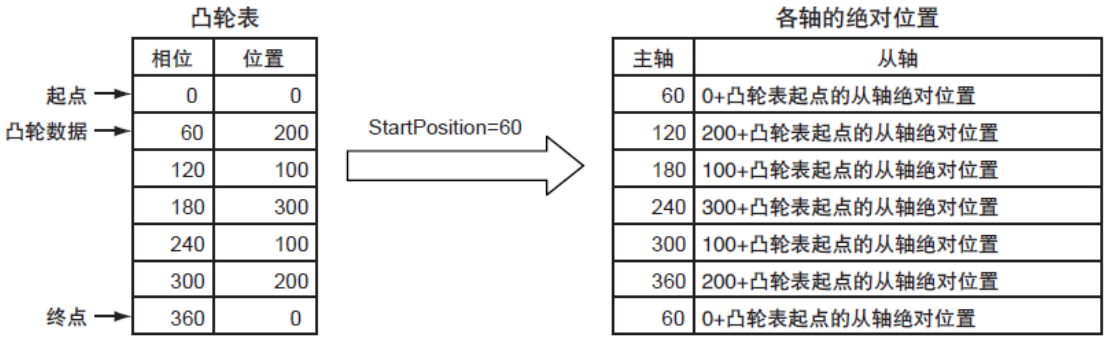
对于凸轮表，其相位和位移均以起点 “0.0” 起的相对量制定。关于主轴，从轴的指令位置。则以直线方式对两个凸轮数据之间进行插补，求得相当于相位（主轴）的位移（从轴）。凸轮

数据数量较少时，相位间隔增大，表现为精度较低的凸轮动作。相反，凸轮数据较多时，相位间隔减少，表现为精度较高的凸轮动作。

说明

达到 StartPosition 的判断条件

- 主轴绝对位置穿越 StartPosition（包括正向穿越和负向穿越）
- 主轴绝对位置正好在 StartPosition 上



并且，主轴通过 MasterStartDistance（主轴跟踪距离）时，开始从轴的凸轮动作，输出变量 StartSync（开始追赶）变 1。MasterStartDistance 以绝对位置或 StartPosition（凸轮表起始位置）起的相对距离来指定。绝对位置或相对位置通过 StartMode（起始位置指定方式）来设定。

示例：从轴的移动变化取决于 StartPosition

凸轮表设置如下：

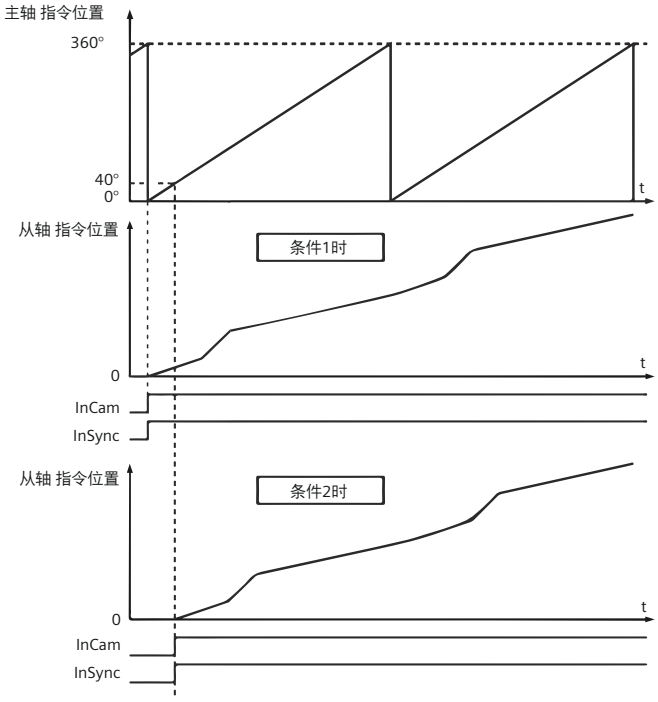
主轴	从轴	连接速度	连接加速度	凸轮曲线	相位间隔
0.000000	0.000000	360.000000	0.000000		
80.000000	80.000000	360.000000	0.000000	直线	0.100000
150.000000	220.000000	720.000000	0.000000	直线	0.100000
360.000000	360.000000	240.000000	0.000000	直线	0.100000

凸轮移动的起始条件如下：

输入变量	条件 1	条件 2
Periodic（重复模式）	TRUE（重复）	TRUE（重复）
StartMode（相对模式）	1	1

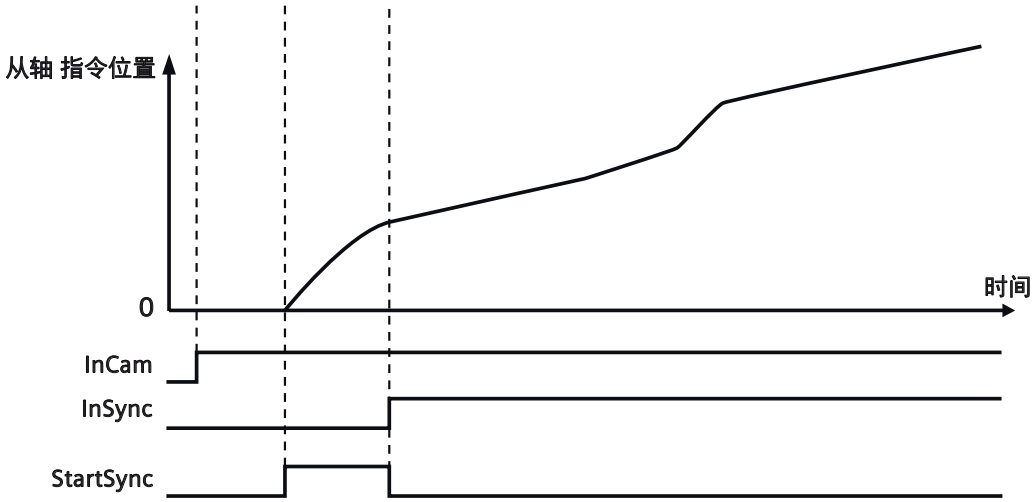
输入变量	条件 1	条件 2
StartPosition	0	40
MasterStartDistance	0	0

- 在条件 1 中，当主轴通过 0° 时，输出变量 InCam（凸轮移动进行中）和输出变量 InSync（同步进行中）都变为 TRUE，并且凸轮移动从轴开始。
- 在条件 2 中，当主轴通过 StartPosition（凸轮表起始位置）指定的 40° 时，输出变量 InCam（凸轮移动进行中）和输出变量 InSync（同步进行中）都变为 TRUE，并且从轴开始凸轮移动。



说明

MasterStartDistance 和 StartPosition 不在一个点时， 主轴在 MasterStartDistance 位置时从轴开始追赶。参考以下时序图：



Periodic（重复模式）

Periodic（重复模式）中设置了 TRUE [重复] 时，凸轮移动将从凸轮表的起点到终点重复执行。

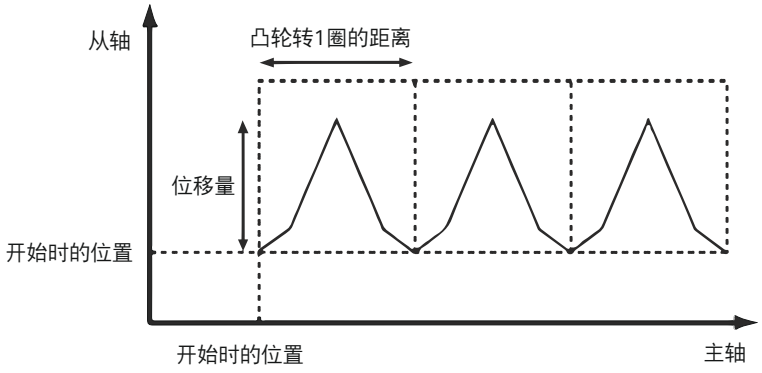
设置了 FALSE [无重复] 时，凸轮运动会在继续执行到凸轮表终点时结束。

如果从轴在凸轮表的起点和终点处的行程位置相同，则将以往复凸轮方式运动（请参见往复凸轮运动）。

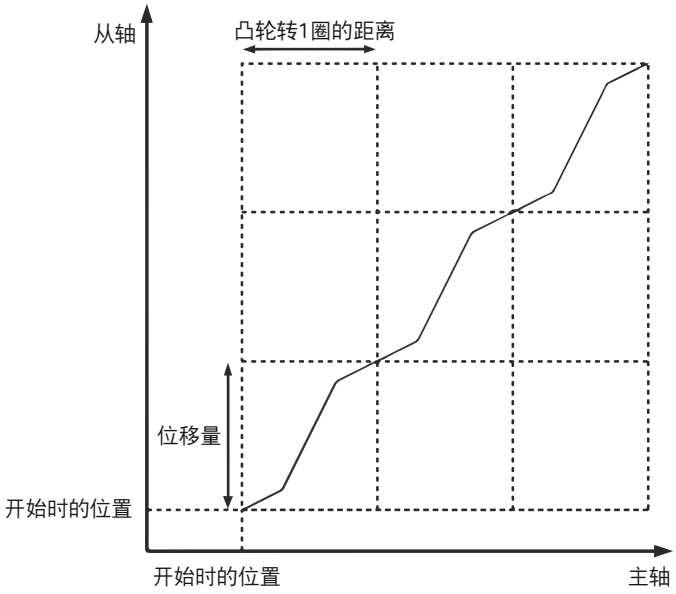
如果从轴在凸轮表的起点和终点处的行程位置不同，则将以传送凸轮方式运行（请参见传送凸轮移动）。

下图中，水平轴显示了主轴位置，垂直轴显示了从轴位置。

往复凸轮移动



传送凸轮移动



EndOfProfile （凸轮周期完成）

在输出凸轮动作的指令值的一个周期内，将 EndOfProfile（凸轮周期完成）设为 1。该凸轮动作使用了以凸轮表终点定义的相位、位移。

对 StartPosition（凸轮表起始位置）设定主轴的绝对位置，凸轮表即为该位置起的相对表。

说明

组态 EndOfProfile 引脚时，注意：

- 对于非周期凸轮，在凸轮周期结束时， EndOfProfile 将置位为 1 并保持。
- 对于周期凸轮，在每个凸轮周期结束时，将置位 1 个扫描周期。
- PTO: EndOfProfile 相对于凸轮表终点最大提前 24 ms。
- IRT: EndOfProfile 相对于凸轮表终点不会提前。滞后最大不超过 2 个 IRT 周期。

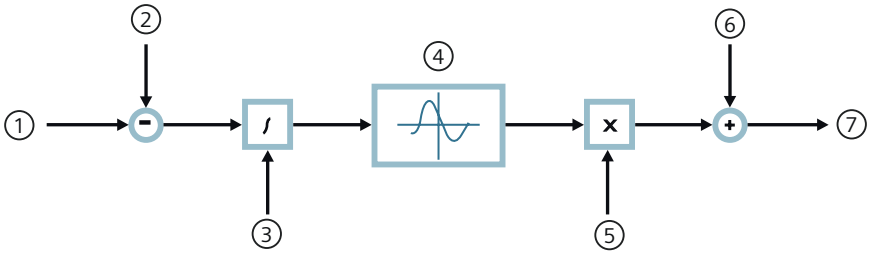
EndOfProfile（凸轮周期完成）即为表现凸轮表终点的输出。

结束凸轮移动

要中途停止凸轮，使用 MC_CamOut（凸轮解耦）、MC_ImmediateStop 或 MC_Stop（强制停止）。

缩放和偏移凸轮

下图展示了缩放和移动凸轮的一般顺序：



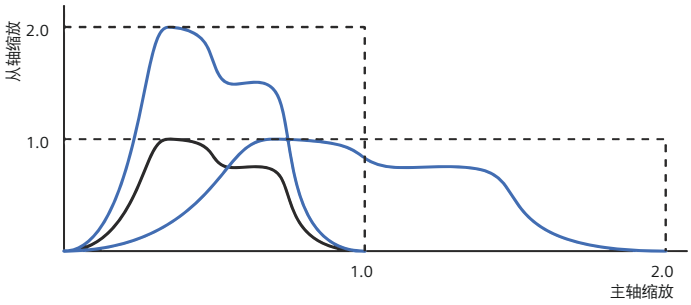
① 引导轴位置
② 主值偏移量
③ 值缩放
④ 凸轮
⑤ 从值缩放
⑥ 从值偏移量
⑦ 跟随轴位置

说明

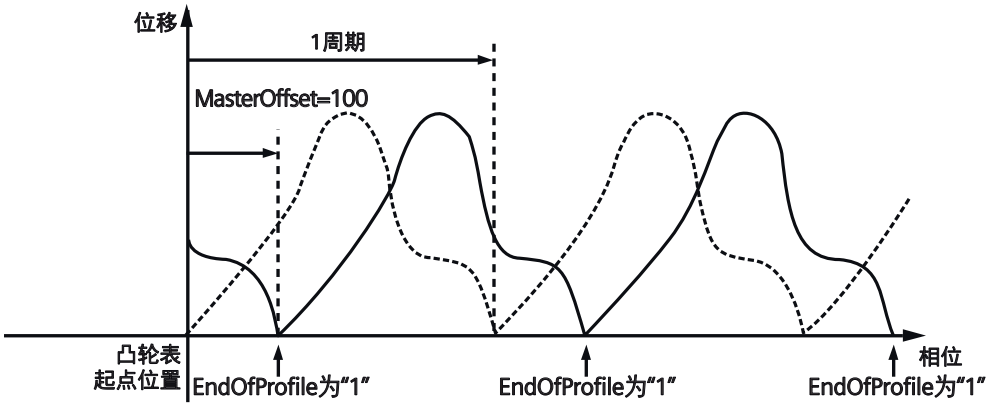
限制

PTO 功能不支持偏移

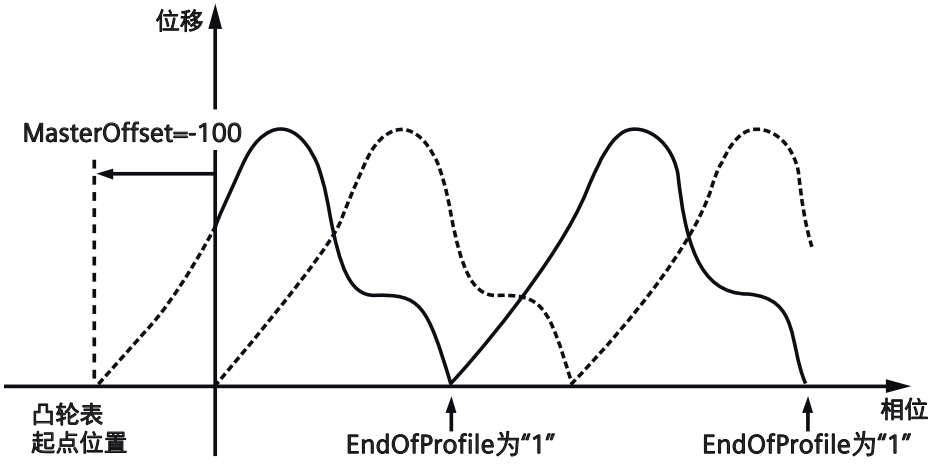
- 系数（缩放）
  针对指定凸轮表的主轴相位、从轴位移，可按照指定比例进行缩放。
  主轴、从轴都可以设定不同的倍率。



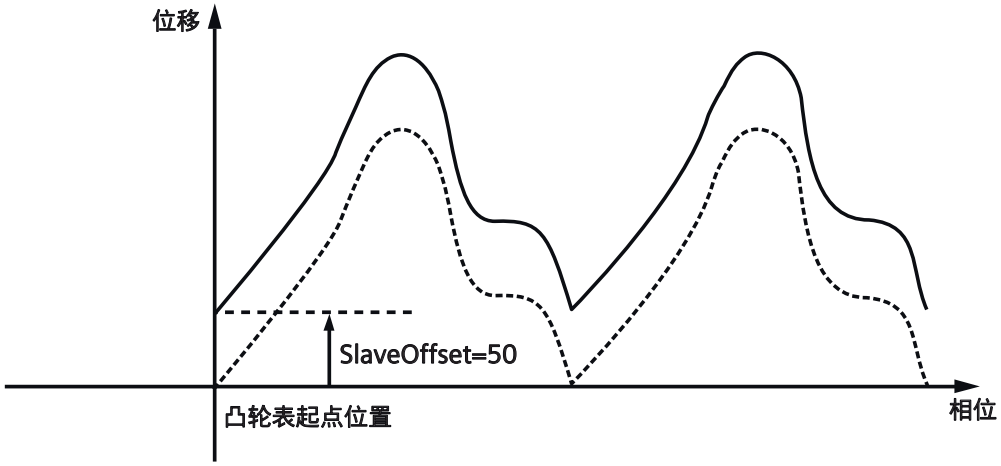
- 偏置
  针对指定凸轮表，可按照偏置量移动相位和位移的功能。 对主轴相位、从轴位移可指定不同的偏置量。
  – MasterOffset（主轴偏置）> 0 时：



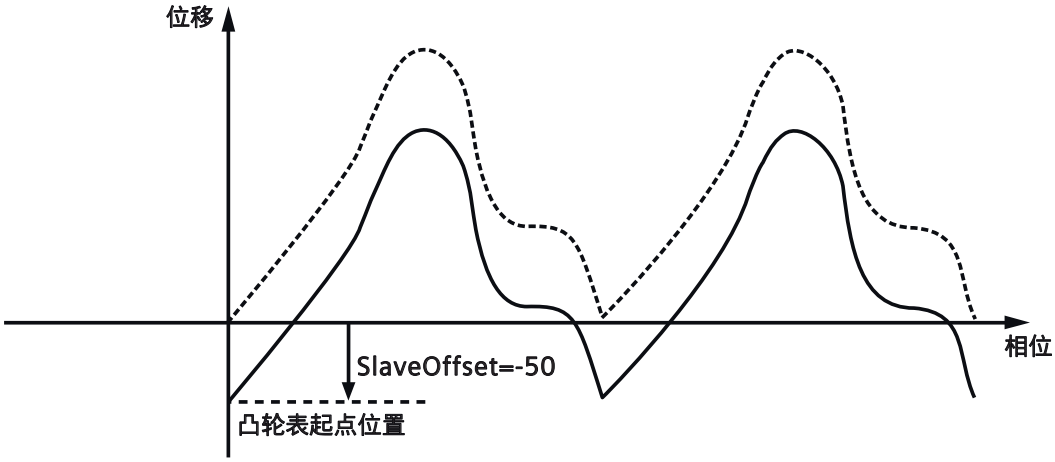
  – MasterOffset（主轴偏置）< 0 时：



– SlaveOffset（从轴偏置）> 0 时：



– SlaveOffset（从轴偏置）< 0 时：



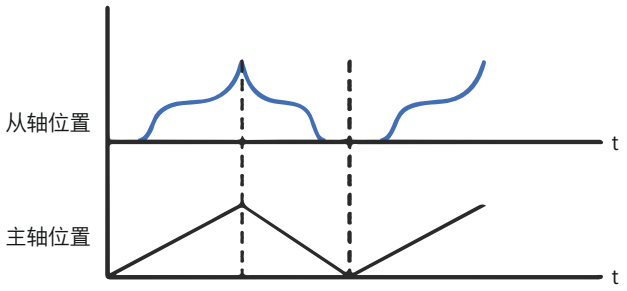
方向

只有当主轴移动方向与 Direction 相同时，才可启动从轴的凸轮移动。

Direction 在 InSync 为 1 时才有效。

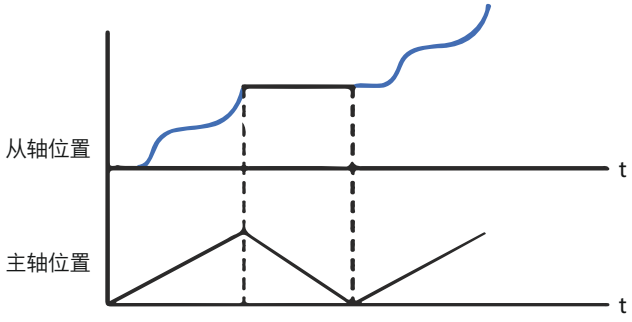
未指定方向

无论主轴朝正方向或负方向运动，凸轮都会开始移动。



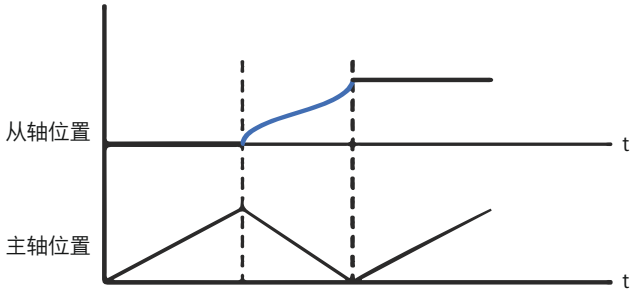
正方向

主轴正向移动时，凸轮开始移动。



指定为负方向

主轴负向移动时，凸轮开始移动。



从轴超速行为

轴类型	同步状态之前（超速行为）	达到同步状态之后
PN IRT 轴 虚轴	当从轴发生超速时，系统将立即进入追赶状态。从轴会加速以追赶主轴，直至重新达到同步状态。	周期凸轮 • 从轴超速后，将以轴配置中设定的最大速度持续运行。 • 此时，StartSync 标志位被置为 1，InSync 标志位被置为 0，EOP (End of Path) 标志位被置为 0。其他参数保持不变。从轴的 TO 中的 VelLimit 标志位将被置为 1，表示速度限制已激活。 • 一旦主从轴再次进入同步状态，StartSync 标志位将被置为 0，InSync 标志位将被置为 1。同时，从轴 TO 中的 VelLimit 标志位将被置为 0，系统恢复正常的同步运行状态。 非周期凸轮 • 如果从轴在超速运行期间，主轴到达了 EOP 点，从轴将继续追赶至该 EOP 点。 • 从轴到达 EOP 点后，将退出 CamIn 模式，并且 EOP 标志位将被置为 1（此行为类似于其他指令的 Done 位）。
PTO 轴	当从轴发生超速时，系统将立即报错并退出当前操作。	

超驰响应

MC_CamIn 指令可由以下运动控制指令中止或缓冲：

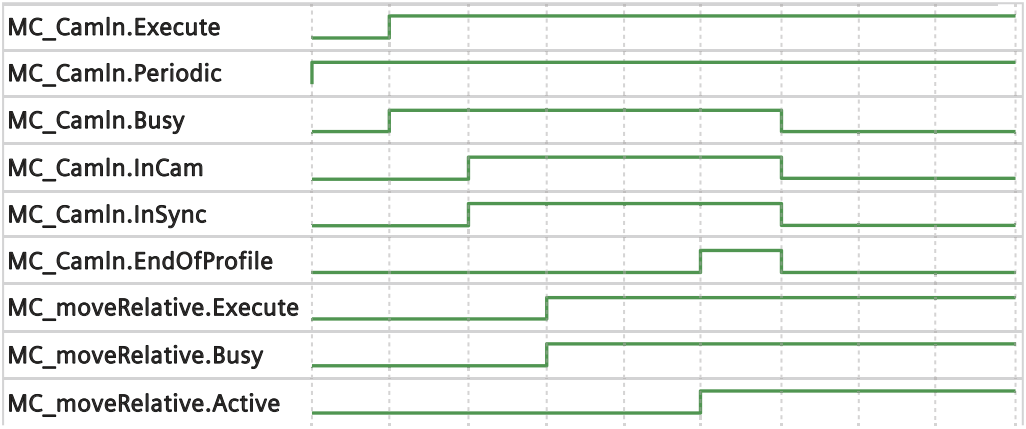
中止（**BufferMode = 0**）	缓冲（**BufferMode = 1**）
MC_Halt MC_Stop MC_ImmediateStop MC_MoveVelocity MC_MoveAbsolute MC_MoveRelative MC_MoveJog MC_CamIn MC_GearIn MC_CamOut	MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

- 新的 MC_CamIn 指令可以中止或缓冲以下运动控制指令：

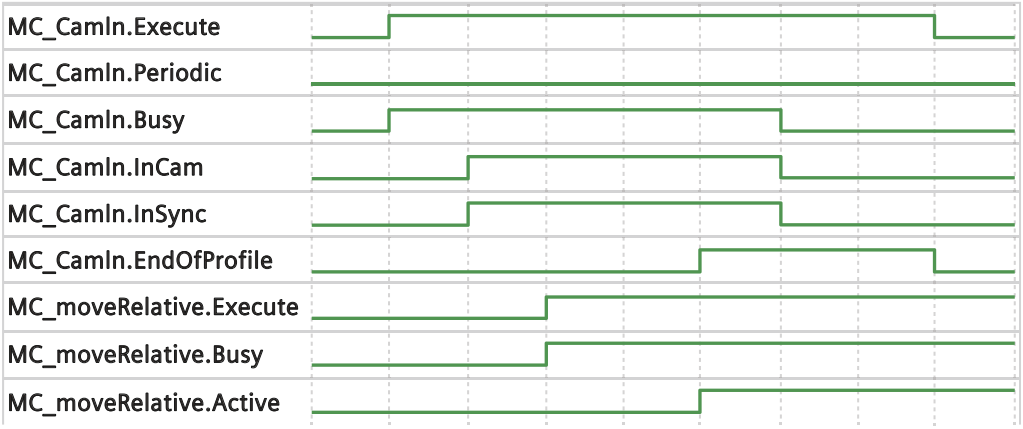
中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveJog MC_CamOut MC_GearOut MC_MoveVelocity MC_CamIn MC_GearIn MC_PhasingRelative	MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

MC_CamIn 被其他指令缓冲时，EOP 拉高和后续指令的 Active 拉高同周期并且输出管脚均为 False，CommandAborted 不会拉高。参考以下时序图：

- 周期凸轮被缓冲的时序图：



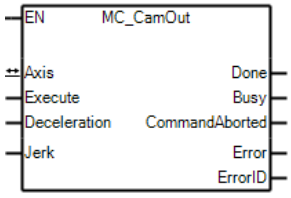
- 非周期凸轮被缓冲的时序图：



14.7.2.2 MC_CamOut

结束输入管脚（Axis）指定的轴的凸轮同步运动。

表格 14-12 MC_CamOut

LAD/FBD	STL	说明
	**FBCALL MC_CamOut, INSTANCE, SlaveAxis, Execute, Deceleration, Jerk, Done, Busy, CommandAborted, Error, ErrorID**	解除 MC_CamIn 凸轮行为，减速停止轴

表格 14-13 MC_CamOut 参数

参数	声明	数据类型	初始值	说明
EN	IN	BOOL	-	能流
Axis	IN_OUT	SYS_AXIS	-	轴名称
Execute	IN	BOOL	-	上升沿触发
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	减速度值
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	加加速度值
Done	OUT	BOOL	False	IRT 轴： 处于 standstill 状态时 PTO 轴、虚轴： 轴速度为 0 时
Busy	OUT	BOOL	False	指令正在运行中
Command Aborted	OUT	BOOL	False	该指令被另一个指令打断
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1074)

说明

Jerk 参数生效规则

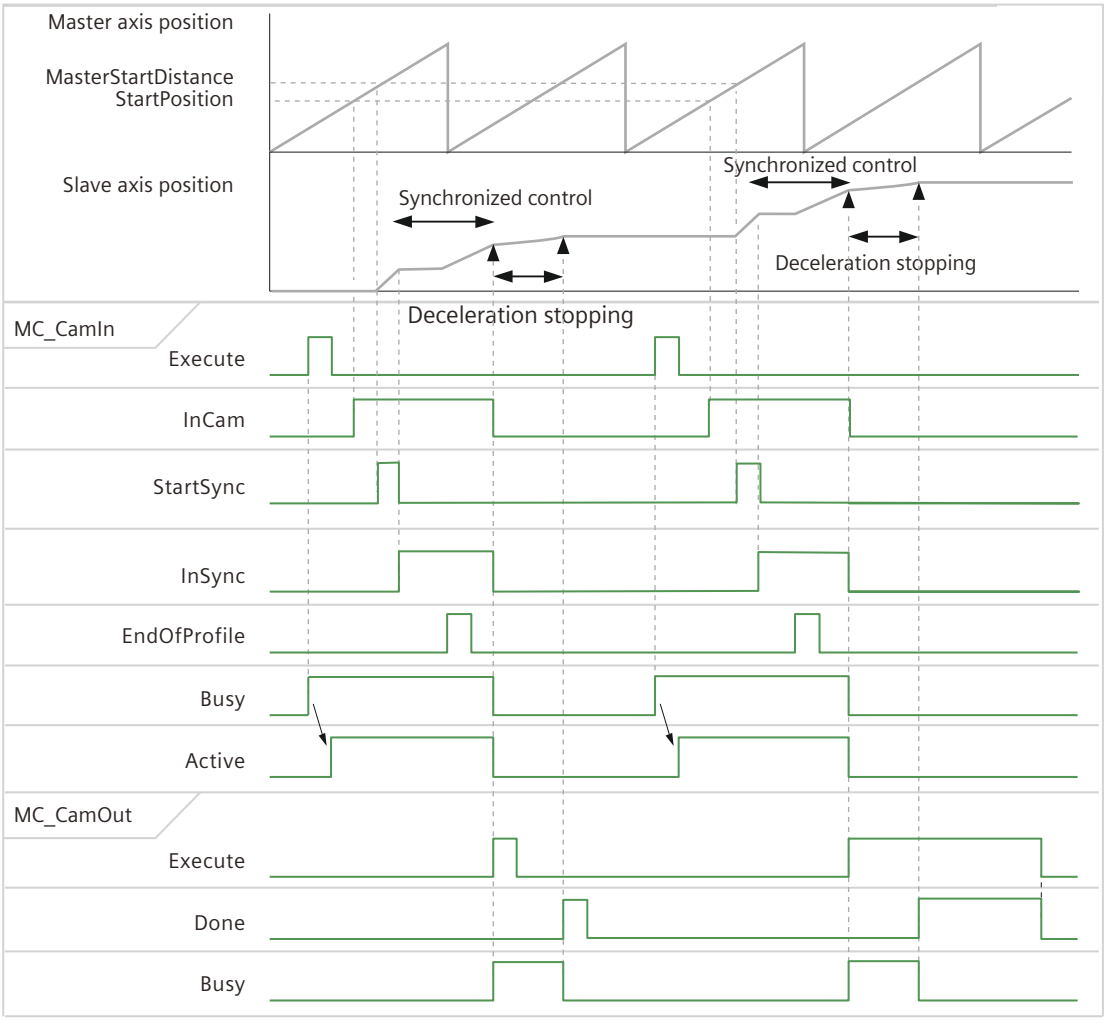
从 STEP 7 Micro/WIN SMART V3 (3.2 版) 开始，CamOut 功能的 Jerk 参数将生效，且适用于所有轴类型。

前置条件

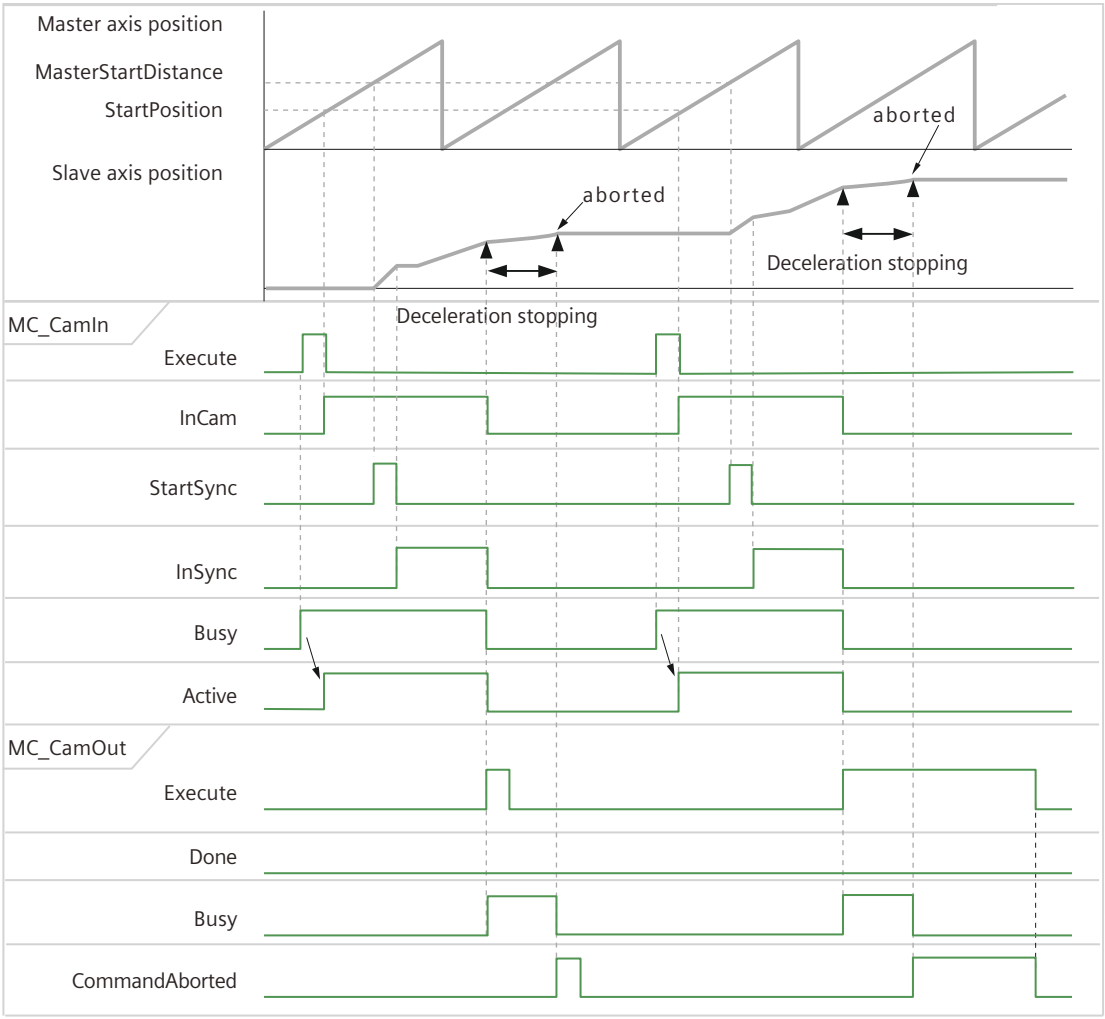
- 从轴已正确组态。
- 从轴正在运行 MC_CamIn 指令。

时序图

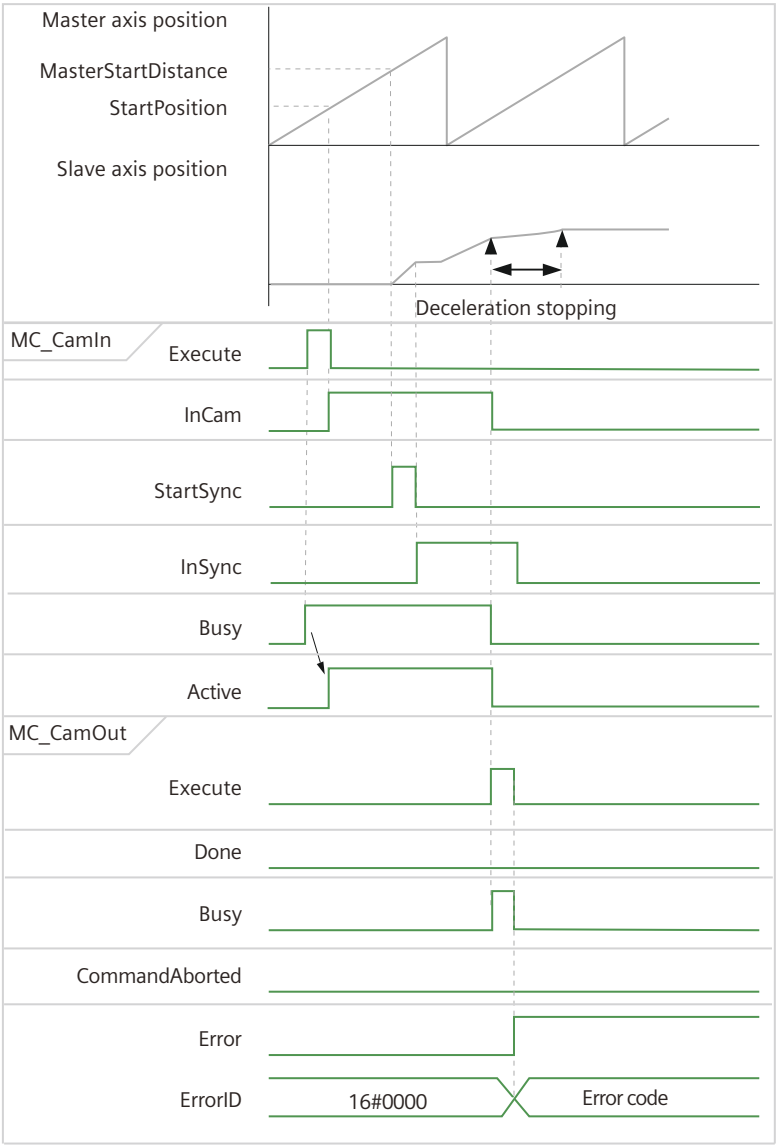
1. 上升沿触发和电平触发



2. 运行中被其他指令打断


Master axis position
MasterStartDistance
StartPosition
Slave axis position
aborted
aborted
Deceleration stopping
Deceleration stopping
MC_CamIn
Execute
InCam
StartSync
InSync
Busy
Active
MC_CamOut
Execute
Done
Busy
CommandAborted

3. 运动中发生轴错误


Master axis position
MasterStartDistance
StartPosition
Slave axis position
Deceleration stopping
MC_CamIn
Execute
InCam
StartSync
InSync
Busy
Active
MC_CamOut
Execute
Done
Busy
CommandAborted
Error
ErrorID
16#0000
Error code

使用注意

- 指令打断行为

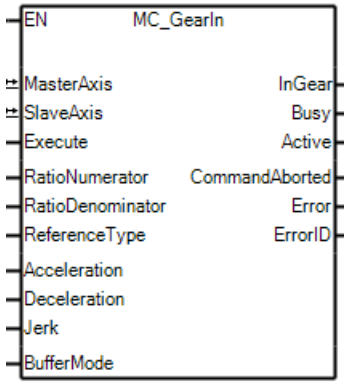
当前指令	后续指令
MC_CamOut	MC_Halt MC_Stop MC_MoveJog MC_ImmediateStop MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn
MC_CamIn	MC_CamOut

- 指令缓冲行为
  CamOut 不能 buffer 在任何指令后面。CamOut 后面也不能 buffer 任何指令。

14.7.2.3 MC_GearIn

MC_GearIn 功能说明

MC_GearIn 用于激活电子齿轮传动，可通过设定主轴和从轴之间的减速比，使从轴根据设定的减速比，同步跟随主轴的运动。

LAD/FBD	STL
	**FBCALL MC_GearIn, INSTANCE, MasterAxis,SlaveAxis, Execute, RatioNumerator, RatioDenominator, ReferenceType, Acceleration, Deceleration, Jerk, BufferMode, InGear, Busy, Active, CommandAborted, Error, ErrorID**

下表列出了 MC_GearIn 指令中主轴和从轴所支持的轴类型：

从轴	主轴			
	IRT	**PTO**	**Virtual**	**Encoder**
IRT	✓ [1]	-	✓	✓
PTO	- [2]	✓	✓	✓
Virtual 虚拟轴	✓	✓	✓	✓

[1] “ ✓ ”表示该轴类型受支持。

[2] “ - ”表示该轴类型不受支持。

说明

由于 PTO 控制的轴与通过 IRT 通信的轴无法实现严格同步，即使通过虚拟轴进行齿轮级联（例如 IRT 轴 → Virtual → PTO 轴），也无法确保 IRT 和 PTO 之间的精确同步。

说明

同步运动时，从轴变量中的 Setting_Info.Jerk 一直显示为 0。

参数

参数	声明	数据类型	初始值	说明
MasterAxis	IN_OUT	SYS_AXIS	-	引用主轴
SlaveAxis	IN_OUT	SYS_AXIS	-	引用从轴
Execute	IN	BOOL	-	上升沿触发该指令
RatioNumerator	IN	DINT	1	指定主轴和从轴之间的电子减速比分子
RatioDenominator	IN	DWORD	1	指定主轴和从轴之间的电子减速比分母

参数	声明	数据类型	初始值	说明
ReferenceType	IN	BYTE	0	在进行从轴同步时，您可以选择以下主轴速度类型： • 0：指令速度（同一任务周期下的主轴设定速度） • 1：反馈速度（同一任务周期下获取的主轴反馈速度） • 2：指令速度（上个任务周期下的设定速度） 不同类型主轴支持的速度类型如下： • IRT 轴： 支持 0，1，2 • 虚拟轴：支持 0，1，2 当主轴为虚拟轴时，设置为 1 等同于设置为 0。 • 编码器轴：1 • PTO 轴：0
Acceleration	IN	REAL	Axis_Config.Dynamic_Default.Default_Acc	加速度值
Deceleration	IN	REAL	Axis_Config.Dynamic_Default.Default_Decel	减速度值
Jerk	IN	REAL	Axis_Config.Dynamic_Default.Default_Jerk	加加速度值
BufferMode	IN	BYTE	0	指定执行多个运动指令时的行为 • 0：打断模式 • 1：缓冲模式
InGear	OUT	BOOL	False	从轴速度与主轴速度同步后，该参数设置为 1
Busy	OUT	BOOL	False	指令未结束
Active	OUT	BOOL	False	表示该指令可对轴进行控制
CommandAborted	OUT	BOOL	False	当前指令被另一个指令打断
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1074)

说明

从轴编号必须大于主轴编号。

警告

MC_GearIn 使用注意

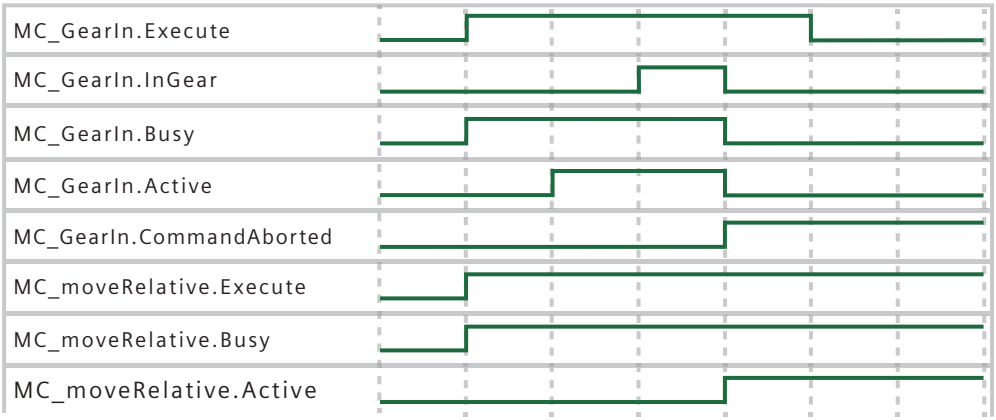
使用 MC_GearIn 时，尽量不要在主轴上执行 MC_Home 或 MC_ImmediateStop 指令，执行指令可能导致从轴快速运行（或轴报错），这种情况十分危险。

前置条件

- 主轴和从轴已经正确组态。
- 从轴已启用。

MC_GearIn 指令被缓冲执行时

当 MC_GearIn 指令被缓冲执行时，其 InGear 信号和 CommandAborted 信号将与缓冲指令的 Active 信号在同一个任务周期内同时变为 TRUE。



注意：如果 MC_GearIn 不是在 OB2 中调用，以上时序将无法严格保证。

超驰响应

- MC_GearIn 指令可由以下运动控制指令中止或缓冲：

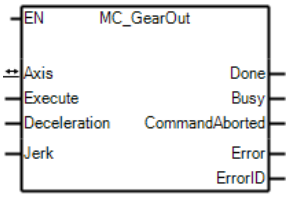
中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_Halt MC_Stop MC_ImmediateStop MC_MoveVelocity MC_MoveAbsolute MC_MoveRelative MC_MoveJog MC_CamIn MC_GearIn MC_GearOut	MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

- 新的 MC_GearIn 指令可以中止或缓冲以下运动控制指令：

中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveJog MC_CamOut MC_GearOut MC_MoveVelocity MC_CamIn MC_GearIn MC_PhasingRelative	MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

14.7.2.4 MC_GearOut

表格 14-14 MC_GearOut

LAD/FBD	STL	说明
	FBCALL MC_GearOut, INSTANCE, SlaveAxis, Exec, Dec, Jerk, Done, Busy, CommandAborted, Error, ErrorID	解除 MC_GearIn 齿轮行为，减速停止轴

表格 14-15 MC_GearOut 参数

参数	声明	数据类型	初始值	说明
EN	IN	BOOL	-	能流
Aixs	IN_OUT	SYS_AXIS	-	轴名称
Execute	IN	BOOL	-	上升沿触发该指令
Deceleration	IN	REAL	Axis_Config.Dynamic_Default.Default_Decel	减速度值
Jerk	IN	REAL	Axis_Config.Dynamic_Default.Default_Jerk	加加速度值
Done	OUT	BOOL	False	IRT 轴：处于 standstill 状态时 PTO 轴、虚轴：轴速度为 0 时
Busy	OUT	BOOL	False	指令正在运行中
CommandAborted	OUT	BOOL	False	该指令被另一个指令打断
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1074)

说明

Jerk 参数生效规则

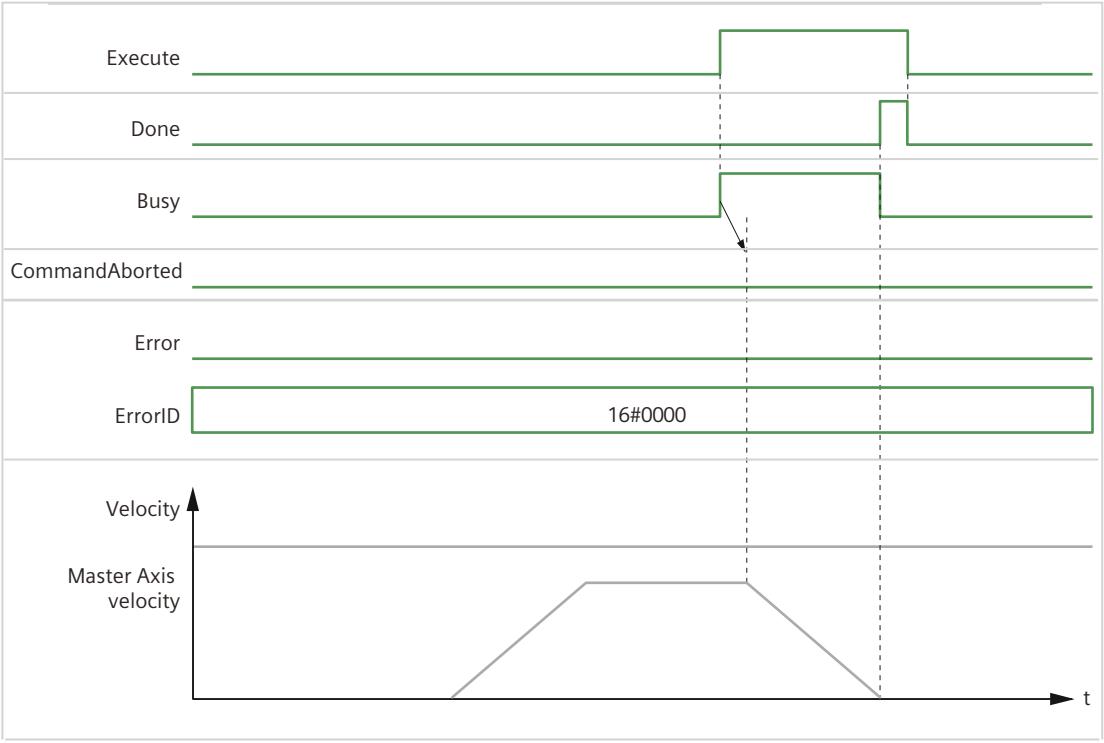
从 STEP 7 Micro/WIN SMART V3 (3.2 版) 开始，GearOut 功能的 Jerk 参数将生效，且适用于所有轴类型。

前置条件

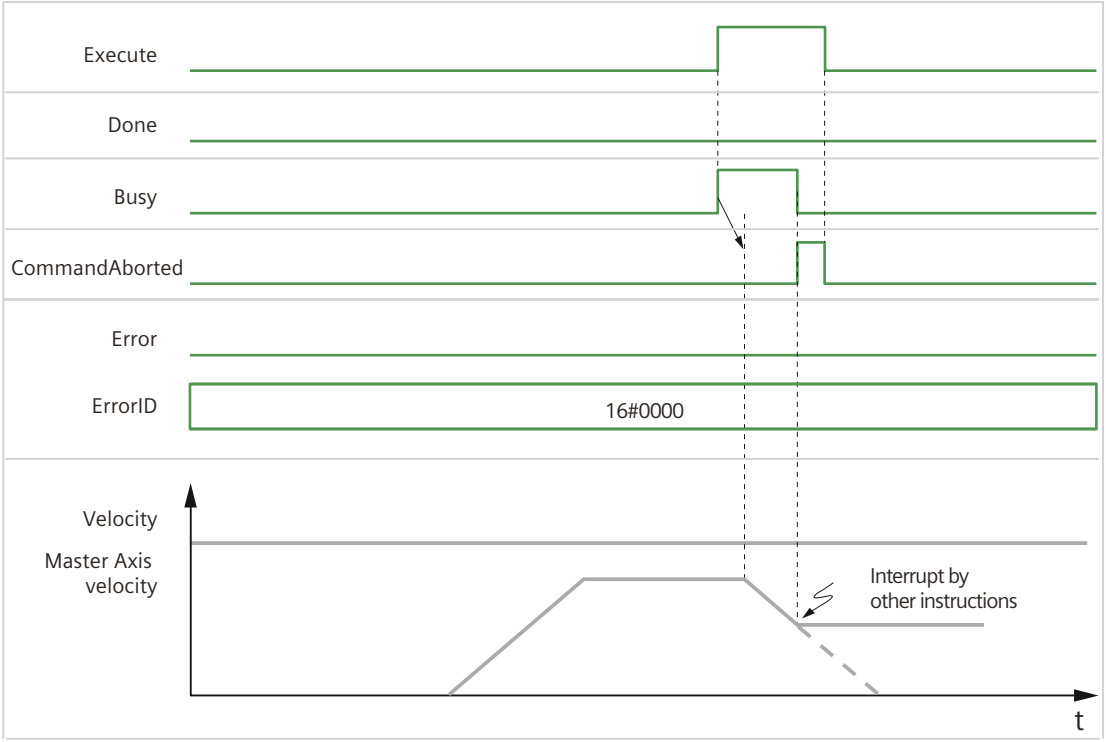
- 从轴已正确组态。
- 轴正在执行 GearIn 指令。

时序图

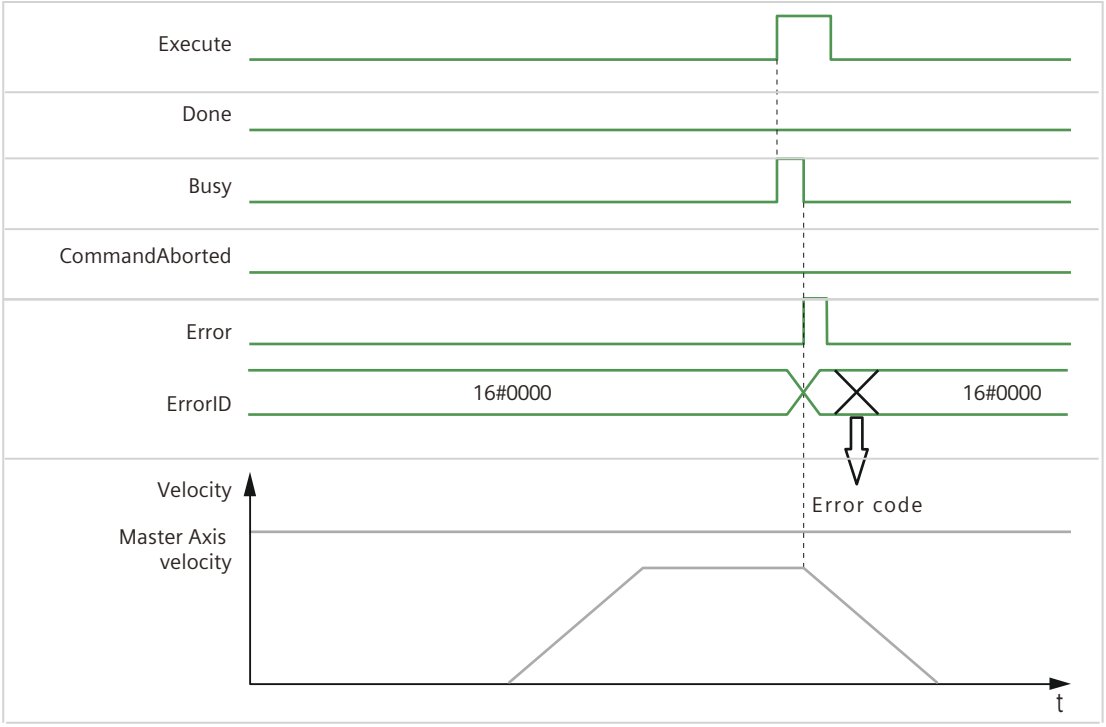
1. 上升沿触发



2. 运行中被其他指令打断


Execute
Done
Busy
CommandAborted
Error
ErrorID
16#0000
Velocity
Master Axis velocity
Interrupt by other instructions
t

3. 运动中发生轴错误


Execute
Done
Busy
CommandAborted
Error
ErrorID
16#0000
16#0000
Error code
Velocity
Master Axis velocity
t

超驰响应

- 指令打断行为

当前指令	后续指令
MC_GearOut	MC_Halt MC_Stop MC_MoveJog MC_ImmediateStop MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn
MC_GearIn	MC_GearOut

- 指令缓冲行为
  GearOut 不能 buffer 在任何指令后面。GearOut 后面也不能 buffer 任何指令。

说明

MC_GearOut 打断 MC_GearIn 时，缓冲队列中的 MC_GearIn 指令不能被打断。

说明

GearOut 指令不支持重启行为。

14.7.2.5 MC_PhasingRelative

MC_PhasingRelative 指令用于在同步运动（如 MC_CamIn 或 MC_GearIn）中，对从轴相对于主轴的当前相位进行补偿。通过此指令，您可以指定一个相对相位偏移量，从而调整从轴的同步位置，而无需中断当前的同步运行。

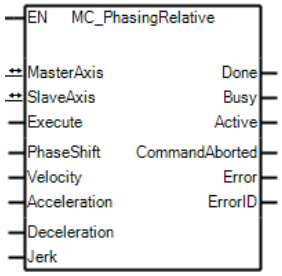
说明

主轴和从轴必须运行 MC_GearIn 或 MC_CamIn 指令。

说明

同步运动时，从轴变量中的 Setting_Info.Jerk 一直显示为 0。

参数

LAD/FBD	STL	说明
	`FBCALL MC_ PhasingRelative, INSTANCE, MasterAxis, SlaveAxis, Execute, PhaseShift,Velocity,Acceleration,Deceleration, Jerk, Done, Busy,Active, CommandAborted, Error, ErrorID`	该指令让 MC_CamIn\MC_GearIn 主轴相对于现在的相位，偏移指定的相位。

参数	声明	数据类型	初始值	说明
EN	IN	BOOL	-	能流
MasterAxis	IN_OUT	SYS_AXIS	-	引用主轴
SlaveAxis	IN_OUT	SYS_AXIS	-	引用从轴
Execute	IN	BOOL	-	上升沿处开始指令
PhaseShift	IN	REAL	-	相位补偿量 > 0.0 时 正向补偿 < 0.0 时 负向补偿
Velocity	IN	REAL	-	指定相移速度 取值范围：> 0.0
Acceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Acc	指定加速度 取值范围：> 0.0
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	指定减速度 取值范围：> 0.0
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	指定加加速度 取值范围：>= 0.0
Done	OUT	BOOL	False	指令完成为 True 主轴的补偿相位值和 FB 中配置的 PhasingShift 相等时置位
Busy	OUT	BOOL	False	指令未结束
Active	OUT	BOOL	False	指令正在对轴进行控制

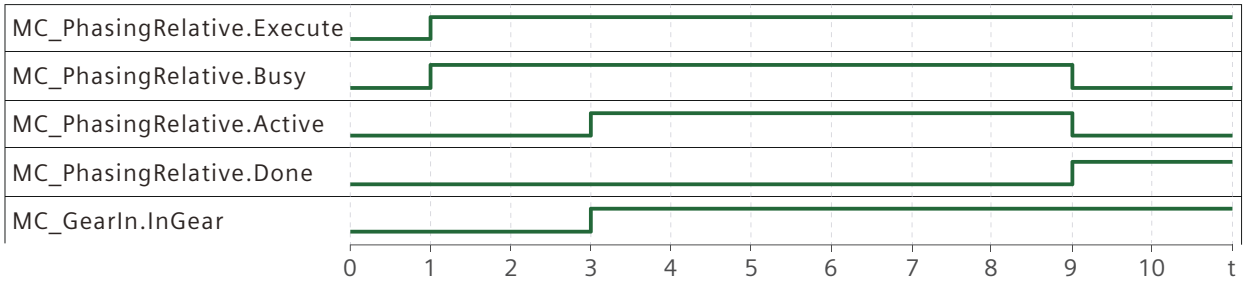
参数	声明	数据类型	初始值	说明
CommandAborted	OUT	BOOL	False	指令被中止
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1074)

前置条件

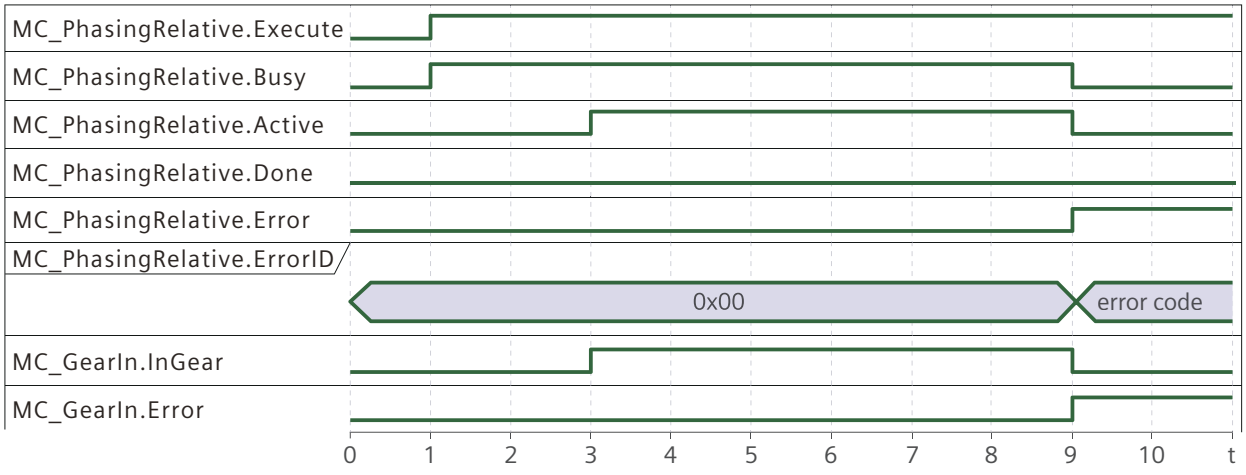
- 主轴和从轴已正确组态。
- 主轴和从轴已启用。
- 主轴和从轴必须运行 MC_GearIn 或者 MC_CamIn 指令，并且主轴和从轴必须一一对应 。

时序图

- 正常运行场景



- 错误导致运行失败场景

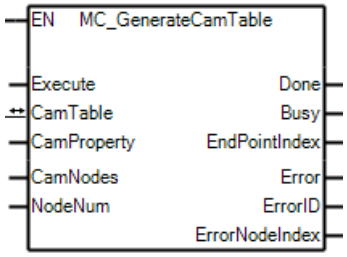


超驰响应

MC_PhasingRelative 只能在 CamIn/GearIn 执行时附加到对应的运动上运行。

14.7.2.6 MC_GenerateCamTable

表格 14-16 MC_GenerateCamTable

LAD/FBD	STL	说明
	**FBCALL MC_GenerateCamTable, INSTANCE, Exec, CamTable, CamProperty, CamNodes, NodeNum, Done, Busy, EndPointIndex, Error, ErrorID, ErrorNodeIndex**	可以使用 MC_GenerateCamTable 指令修改凸轮表。

表格 14-17 MC_GenerateCamTable 的参数

参数	声明	数据类型	初始值	说明
Execute	IN	BOOL	-	• 上升沿处开始指令 • 下降沿 – 当 Done=1 或 Error=1 时，将所有输出引脚设为 0 – 在所有其它情况下无操作
CamTable	INOUT	SYS_CAM	-	指定需要修改的凸轮表，该值不能为空
CamProperty	INOUT	SYS_CAM_PROPERTY	-	指定凸轮表属性，该值不能为空
CamNodes [1]	INOUT	SYS_CAM_NODE[361]	-	用于修改指定凸轮表的关键点数组，该值不能为空
NodeNum [1]	IN	WORD	0	指定关键点数组中的元素数。 值范围 [2, 361]
Done	OUT	BOOL	-	指令执行完成后设置为 1
Busy	OUT	BOOL	-	指令执行时设置为 1
EndPointIndex	OUT	WORD	-	指令执行完成后，Cam_Data 数组中最后一个数据的索引

参数	声明	数据类型	初始值	说明
Error	OUT	BOOL	-	执行期间出错时，设置为 1
ErrorID	OUT	WORD	-	执行出错时显示错误代码 (页 1074)
ErrorNodePointIndex	OUT	WORD	-	当指令执行失败时，Cam_Node 数组中错误关键点的索引

说明

数组的大小必须大于等于 NodeNum。

数据结构

表格 14-18 SYS_CAM_PROPERTY

	类型	是否参与计算？
m_unit	BYTE	N
s_unit	BYTE	N
period	REAL	Y
init_vel	REAL	Y
init_acc	REAL	Y
limit_vel	REAL	N
limit_acc	REAL	N
limit_jerk	REAL	N
empty_datanum	WORD	N

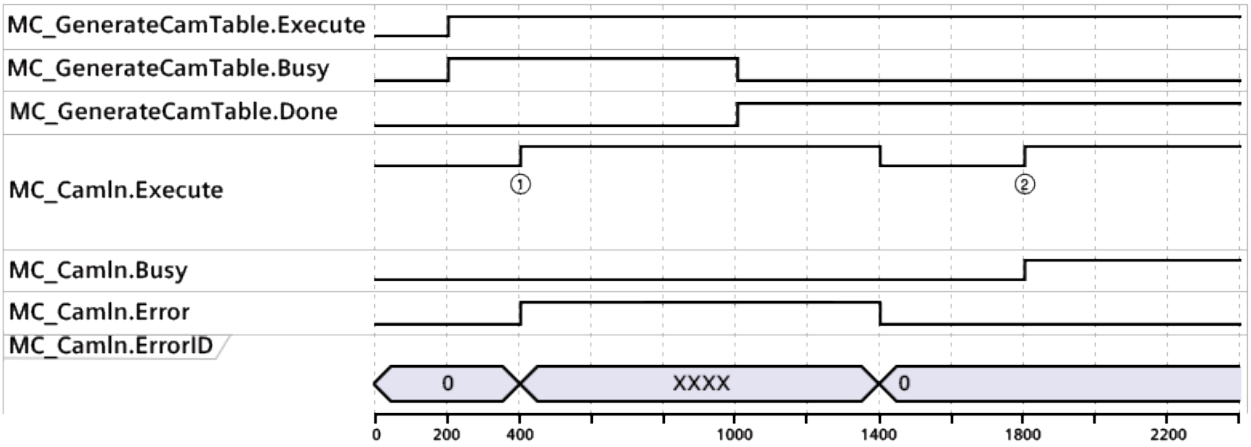
表格 14-19 SYS_CAM_NODE

	类型
phase	REAL
position	REAL
velocity	REAL
acceleration	REAL

	类型
curve （0：常数，1：直线，2：三次多项式，3：五次多项式，4：变形等速）	WORD
interval	REAL

MC_GenerateCamTable 和 MC_CamIn

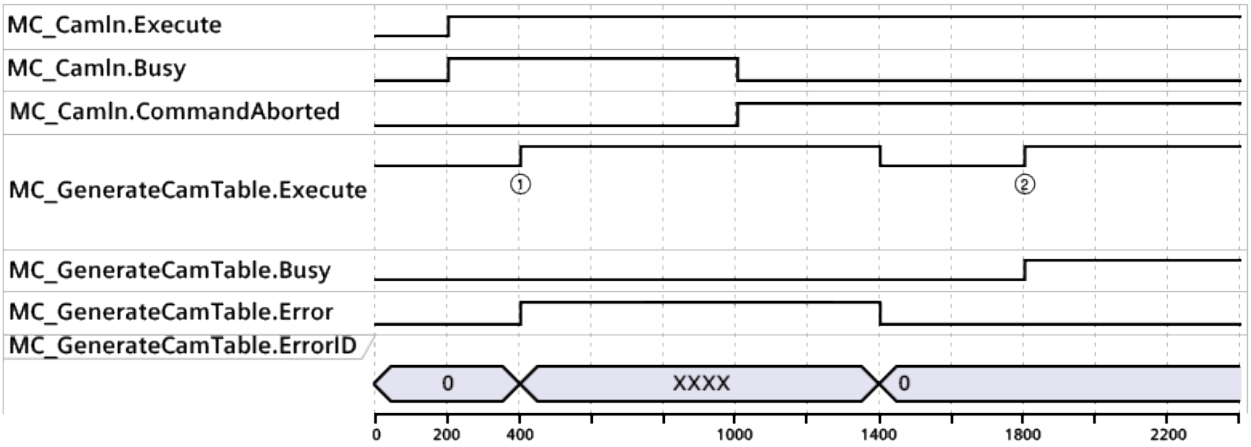
- 执行 MC_GenerateCamTable 时，无法执行 MC_CamIn 指令。



① 在繁忙状态下执行 MC_CamIn 时发生错误。

② MC_CamIn 可在 MC_GenerateCamTable 执行完成后执行。

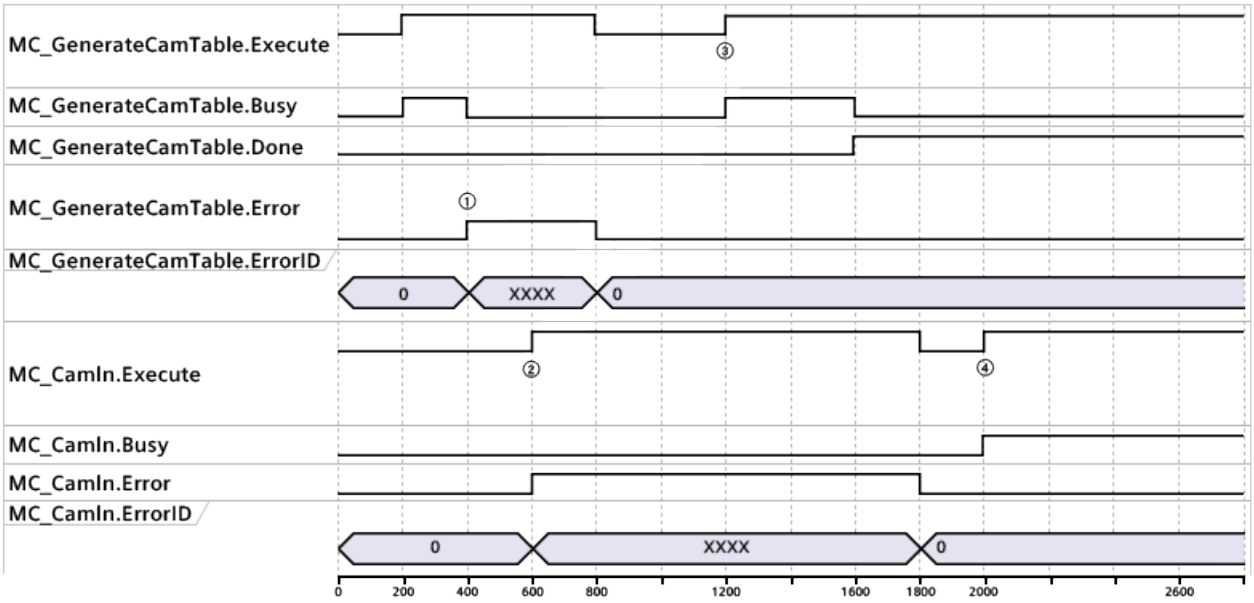
- 反之，执行 MC_CamIn 时，无法执行 MC_GenerateCamTable 指令。
  在下图中，CamIn.CommandAborted 表示使用 CamOut 取消轴同步运动。



① 在繁忙状态下执行 MC_GenerateCamTable 时发生错误。

② MC_GenerateCamTable 可在 MC_CamIn 执行完成后执行。

- 如果 MC_GenerateCamTable 报告错误，必须正确修改 CamProperty 和 CamNodes 数据，然后成功重新执行 MC_GenerateCamTable，才能再次执行 CamIn。



① 执行 MC_GenerateCamTable 时发生错误。

② 在错误状态下执行 MC_CamIn 时发生错误。

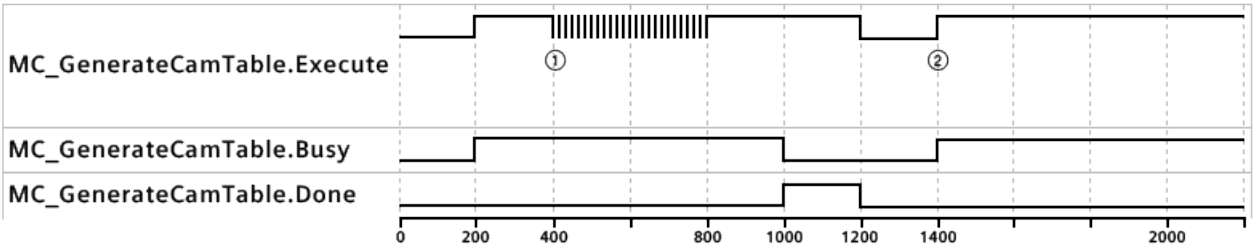
③ 处理错误并重新执行 MC_GenerateCamTable。

④ MC_CamIn 可在 MC_GenerateCamTable 执行完成后执行。

使用 MC_GenerateCamTable

对于由凸轮表引脚指定的同一凸轮表，不能同时执行多条 MC_GenerateCamTable 指令。

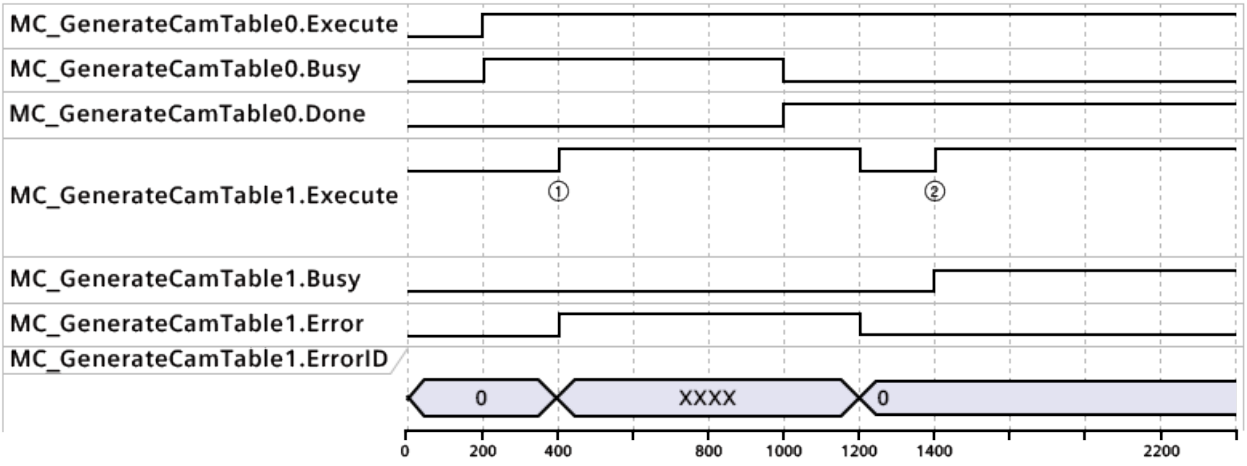
- 相同的 FB 实例



① 在执行 MC_GenerateCamTable 实例期间执行相同的 MC_GenerateCamTable 时会发生错误。

② MC_GenerateCamTable 可在执行完成后再次执行。

- 不同的 FB 实例



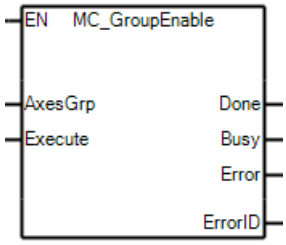
① 凸轮表在修改过程中无法执行。

② MC_GenerateCamTable 可在完成修改后再次执行。

14.7.3 轴组指令

14.7.3.1 MC_GroupEnable

表格 14-20 MC_GroupEnable

LAD/FBD	STL	说明
	**FBCALL MC_GroupEnable, INSTANCE, AxesGrp, Execute, Done, Busy, Error, ErrorID**	MC_GroupEnable 指令将轴组的状态从 GroupDisabled 更改为 GroupStandby。这条管理指令不会生成任何运动。

表格 14-21 MC_GroupEnable 参数

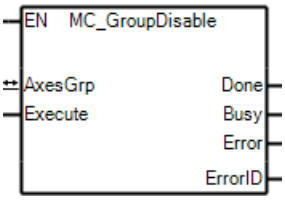
参数	声明	数据类型	初始值	说明
AxesGrp	IN_OUT	SYS_AXES_GROUP	-	引用轴组
Execute	IN	BOOL	-	在上升沿处开始状态改变
Done	OUT	BOOL	False	指令完成后，轴组处于 GroupStandby 状态。
Busy	OUT	BOOL	False	指令未结束
Error	OUT	BOOL	False	指令执行期间发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1076)

说明

1. 要求
   必须组态并启用组中的每个轴和组本身
2. 超驰响应
   MC_GroupEnable 无法被其它任何运动控制中止或缓冲。

14.7.3.2 MC_GroupDisable

表格 14-22 MC_GroupDisable

LAD/FBD	STL	说明
	**FBCALL MC_GroupDisable, INSTANCE, AxesGrp, Execute, Done, Busy, Error, ErrorID**	MC_GroupDisable 指令将轴组状态改为 GroupDisabled。由于没有产生任何移动，因此该指令为管理指令。

表格 14-23 MC_GroupDisable 参数

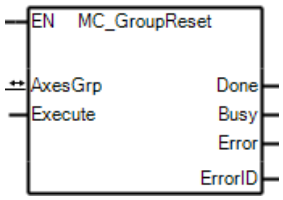
参数	声明	数据类型	初始值	说明
AxesGrp	IN_OUT	SYS_AXES_GROUP	-	引用轴
Execute	IN	BOOL	-	在上升沿处开始状态改变
Done	OUT	BOOL	False	指令完成后，轴组处于 GroupDisabled 状态
Busy	OUT	BOOL	False	指令未结束
Error	OUT	BOOL	False	指令执行期间发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1076)

说明

1. 要求
   - 必须正确组态组中的每个轴和组本身
   - 轴组没有错误
2. 超驰响应
   - MC_GroupDisable 指令无法被其它任何运动控制指令中止或缓冲。
   - 禁用轴组会立即中止关联轴组的所有运动控制指令。

14.7.3.3 MC_GroupReset

表格 14-24 MC_GroupReset

LAD/FBD	STL	说明
	**FBCALL MC_GroupReset, INSTANCE, AxesGrp, Execute, Done, Busy, Error, ErrorID**	MC_GroupReset 指令通过复位所有内部轴组相关错误将状态从 GroupErrorStop 转换为 GroupStandby。该指令不影响任何其它指令实例的输出。

表格 14-25 MC_GroupReset 的参数

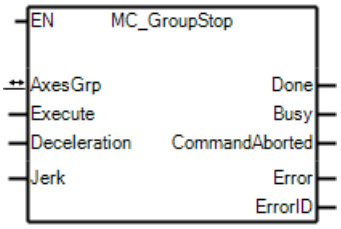
参数	声明	数据类型	初始值	说明
AxesGrp	IN_OUT	SYS_AXES_GROUP	-	引用轴组
Execute	IN	BOOL	-	上升沿触发该指令
Done	OUT	BOOL	False	轴组中所有轴的复位已完成
Busy	OUT	BOOL	False	指令正在运行中
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1076)

说明

1. 要求
   - 必须正确组态组中的每个轴以及组本身。
   - 轴组必须处于 ErrorStop 状态。
2. 超驰响应
   - MC_GroupReset 指令无法被其它任何运动控制指令中止或缓冲。
   - 新的 MC_GroupReset 指令不会中止或缓冲任何其它激活的运动控制指令。

14.7.3.4 MC_GroupStop

表格 14-26 MC_GroupStop

LAD/FBD	STL	说明
	**FBCALL MC_GroupStop, INSTANCE, AxesGrp, Execute, Deceleration, Jerk, Done, Busy, Active, CommandAborted, Error, ErrorID**	MC_GroupStop 指令命令轴组减速并停止，并将轴组转换为 GroupStopping 状态。 该指令会中止任何正在进行的指令执行。 当轴组处于 GroupStopping 状态时，其它任何指令都无法对该轴组执行任何运动命令。 轴组速度达到零后，Done 输出立即设置为 TRUE。 只要 Execute 输入仍为 TRUE，或者运动速度尚未达到零，该轴组就会保持在 GroupStopping 状态。 只要 Done 输出设为 1 且 Execute 输入为 0，轴组就会转换为 GroupStandby 状态。

表格 14-27 MC_GroupStop 的参数

参数	声明	数据类型	初始值	说明
AxesGrp	IN_OUT	SYS_AXES_GROUP	-	引用轴组
Execute	IN	BOOL	-	在上升沿开始执行
Deceleration	IN	REAL	-	Deceleration 参数的值
Jerk	IN	REAL	0.0	Jerk 参数的值
Done	OUT	BOOL	False	轴组运动已完成
Busy	OUT	BOOL	False	指令未结束
CommandAborted	OUT	BOOL	False	该命令被另一个指令中止
Error	OUT	BOOL	False	指令执行期间发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1076)

前置条件

- 定位运行轴组已正确组态。
- 轴组已启用。

超驰响应

- 可以中止 MC_GroupStop 指令的运动控制指令：

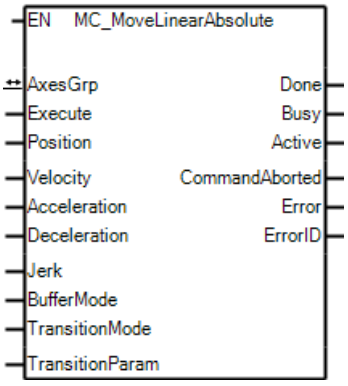
中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_GroupDisable MC_Power	

- 新的 MC_GroupStop 指令可以中止以下运动控制指令：

中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_MoveLinearAbsolute MC_MoveLinearRelative	

14.7.3.5 MC_MoveLinearAbsolute

表格 14-28 MC_MoveLinearAbsolute

LAD/FBD	STL	说明
	**FBCALL MC_MoveLinearAbsolute , INSTANCE, AxesGrp, Execute, Position, Velocity, Acceleration, Deceleration, Jerk, BufferMode, TransitionMode, TransitionParameter, Done, Busy, Active, CommandAborted, Error, ErrorID**	MC_MoveLinearAbsolute 指令让轴组在笛卡尔坐标系进行线性插补，从当前位置运动到指定的绝对位置。 该指令支持 2D 和 3D 线性运动。

表格 14-29 MC_MoveLinearAbsolute 的参数

参数	声明	数据类型	初始值	说明
AxesGrp	IN_OUT	SYS_AXES_GROUP	-	引用轴组
Execute	IN	BOOL	-	上升沿触发该指令
Position	IN	ARRAY [0..3] OF REAL	-	笛卡尔坐标系中每个方向（X、Y、Z 轴）的绝对终位的数组 [0..3]。 Array[3] 为保留数组
Velocity	IN	REAL	-	目标速度 (不一定要达到) 取值范围：任意实数
Acceleration	IN	REAL	-	加速度值，可缺省 取值范围：> 0.0
Deceleration	IN	REAL	-	减速度值，可缺省 取值范围：> 0.0
Jerk	IN	REAL	0.0	加加速度值，可缺省 取值范围：>= 0.0
BufferMode	IN	BYTE	0	• 0：中断模式。根据原命令的 Deceleration 的值，在原路径速度降为零。然后启动该命令。 • 1：缓冲模式。当前运动结束后启动指令
TransitionMode	IN	BYTE	0	预留
TransitionParam	IN	DWORD	-	预留
Done	OUT	BOOL	False	所有轴均已到达所命令的终位
Busy	OUT	BOOL	False	指令未结束
Active	OUT	BOOL	False	指令正在对轴组进行控制
CommandAborted	OUT	BOOL	False	该命令被另一个命令中止
Error	OUT	BOOL	False	指令执行期间发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1076)

前置条件

- 轴组组态正确。
- 轴组已启用。
- 轴组中所有的单轴都已经完成 MC_Home 指令。

超驰响应

- 可以中止或缓冲 MC_MoveLinearAbsolute 指令的运动控制指令：

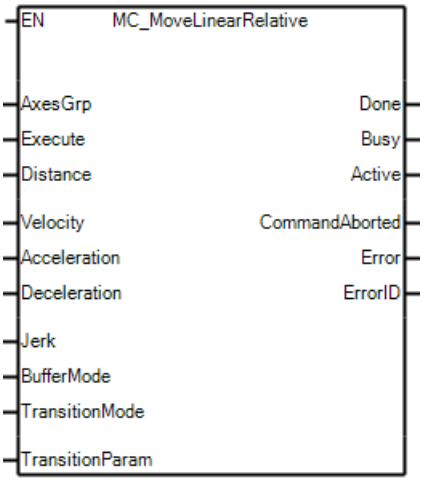
中止（**BufferMode = 0**）	缓冲（**BufferMode = 1**）
MC_Power MC_MoveLinearAbsolute MC_MoveLinearRelative MC_GroupDisable MC_GroupStop	MC_MoveLinearAbsolute MC_MoveLinearRelative

- 新的 MC_MoveLinearAbsolute 指令可以中止或缓冲以下运动控制指令：

中止（**BufferMode = 0**）	缓冲（**BufferMode = 1**）
MC_MoveLinearAbsolute MC_MoveLinearRelative	MC_MoveLinearAbsolute MC_MoveLinearRelative

14.7.3.6 MC_MoveLinearRelative

表格 14-30 MC_MoveLinearRelative

LAD/FBD	STL	说明
	**FBCALL MC_MoveLinearRelative , INSTANCE, AxesGrp, Execute, Distance, Velocity, Acceleration, Deceleration, Jerk, BufferMode, TransitionMode, TransitionParameter, Done, Busy, Active, CommandAborted, Error, ErrorID**	MC_MoveLinearRelative 指令让轴组在笛卡尔坐标系进行线性插补，从当前位置运动到指定的相对位置。 该指令支持 2D 和 3D 线性运动。

表格 14-31 MC_MoveLinearRelative 的参数

参数	声明	数据类型	初始值	说明
AxesGrp	IN_OUT	SYS_AXES_GROUP	-	引用轴组
Execute	IN	BOOL	-	上升沿触发该指令
Distance	IN	ARRAY [0..3] of REAL	-	笛卡尔坐标系中每个方向（X、Y、Z 轴）的相对距离的数组 [0..3]。Array[3] 为保留数组
Velocity	IN	REAL	-	目标速度 (不一定要达到) 取值范围：任意实数
Acceleration	IN	REAL	-	加速度值，可缺省 取值范围：> 0.0
Deceleration	IN	REAL	-	减速度值，可缺省 取值范围：> 0.0
Jerk	IN	REAL	0.0	加加速度值，可缺省 取值范围：>= 0.0

参数	声明	数据类型	初始值	说明
BufferMode	IN	BYTE	0	• 0：中断模式。根据原命令的 Deceleration 的值，在原路径速度降为零。然后启动该指令。 • 1：缓冲模式。当前运动结束后启动指令
TransitionMode	IN	BYTE	0	0
TransitionPara	IN	DWORD	-	预留
Done	OUT	BOOL	False	所有轴均已到达所命令的终位
Busy	OUT	BOOL	False	指令正在运行中
Active	OUT	BOOL	False	指令正在对轴组进行控制
CommandAborted	OUT	BOOL	False	当前指令被另一个指令打断
Error	OUT	BOOL	False	指令发生错误
ErrorID	OUT	WORD	0	错误代码 (页 1076)

前置条件

- 轴组组态正确。
- 轴组已启用。

超驰响应

- 可以中止或缓冲 MC_MoveLinearRelative 指令的运动控制指令：

中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_Power MC_MoveLinearAbsolute MC_MoveLinearRelative MC_GroupDisable MC_GroupStop	MC_MoveLinearAbsolute MC_MoveLinearRelative

- 新的 MC_MoveLinearRelative 指令可以中止或缓冲以下运动控制指令：

中止（BufferMode = 0）	缓冲（BufferMode = 1）
MC_MoveLinearAbsolute MC_MoveLinearRelative	MC_MoveLinearAbsolute MC_MoveLinearRelative

14.8 运动控制资源不足时的故障排除与优化

在复杂的自动化系统中，运动控制性能的稳定性和可靠性至关重要。当系统出现运动控制资源不足的情况时，会导致 CPU STOP，轴停止运动。本章将指导您如何识别和解决这类问题，确保运动控制系统的最佳运行。

识别运动控制资源不足

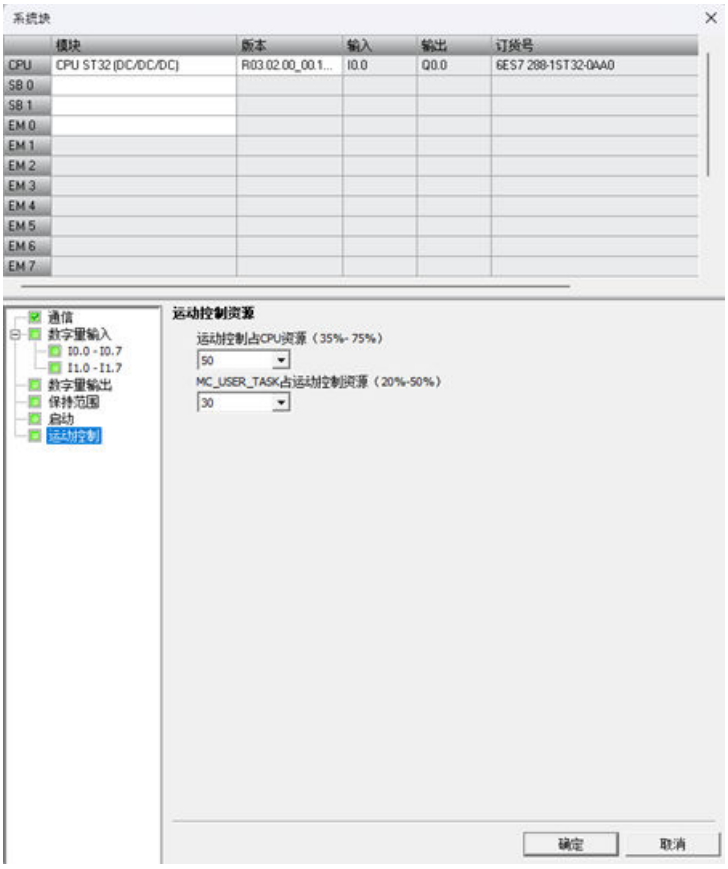
在进行故障排除之前，首先需要确认当前的问题是否确实由运动控制资源不足引起。以下是一些常见的诊断方法：

- 在 PLC -> 系统 -> CPU (页 182) 中，检查是否存在以下非致命错误:
  - Motion 资源不足
  - MC_USER_TASK 运行超时
- 扫描速率 (页 187)：使用扫描速率可以检查运动控制任务（Main, Motion, MC_USER_TASK）的实际执行时间 。如果实际周期显著长于配置的周期，或者周期波动较大，超时次数超限，则表明任务可能无法按时完成，导致资源不足。

- PLC 日志：在 PLC 日志中，仔细查找与程序退出或停止相关的事件。如果日志中出现“超时”、“资源不足”等关键字，这通常是运动控制资源不足的体现。这些信息可以帮助您定位资源不足的原因 。
- PLC 信息中的运控制资源占比检查：在系统块 -> 运动控制 (页 224)中，可以查看到运动控制任务所占用的 CPU 资源比例和 MC_USER_TASK 占用运动控制的资源比例。

运动控制资源不足的解决方案

- 增加运动控制任务的资源占比：在系统块 -> 运动控制 (页 224)中，可以调整系统配置，为运动控制任务分配更多的 CPU 时间。这是最直接且通常最有效的解决方案之一。



- 优化运动控制程序
  - 审查运动控制中的程序代码，移除不必要的计算、逻辑判断或数据处理。
  - 避免在等时任务中执行耗时较长的操作，例如复杂的浮点运算、字符串处理或大量数据块的读写。
  - 将非时间关键的代码移出 MC_USER_TASK，放入主扫描中执行。
  - 尽量使用优化的数据类型和数据访问方式。例如，直接访问数据块（DB）中的数据，而不是通过复杂的指针或间接寻址。
  - 避免在等时任务中频繁调用需要大量系统资源的函数块（FB）。

- **重新评估系统需求**
  - 重新评估当前项目中是否所有轴都必须同时进行高速、高精度运动。
  - 是否可以将某些轴的运动序列化，而不是并行执行？
  - 是否可以简化某些轴的运动轨迹或控制策略？
- **分阶段实施**
  如果项目允许，可以考虑将一个大型的多轴系统拆分为多个独立的、轴数量较少的运动控制单元，每个单元由独立的 PLC 或运动控制器控制。这会增加硬件成本和系统复杂性，但可以有效分散 CPU 负载。

14.9 错误代码

14.9.1 单轴运动的错误代码

表格 14-32 运动系统组态的错误代码

错误代码	说明
0x0001	轴类型无效 STEP 7 Micro/WIN SMART V3 支持 0 - 虚拟轴，1 - PTO 轴， 2 - 编码器轴， 3- PN IRT 轴
0x0002	无效轴模式
0x0003	无效急停触发类型
0x0004	无效的回零参考点标记类型
0x0005	无效 Homing 模式
0x0006	无效轴 TO 指针区域
0x0007	无效急停响应类型
0x0008	无效负硬限位响应类型
0x0009	无效正硬限位响应类型
0x000A	无效轴 TO 指针类型
0x000B	未知版本
0x000C	轴 SDB Record 长度错误
0x000D	轴 SDB Record ID 重复
0x000E	探针 0 输入点被占用或者输入点超出范围
0x000F	探针 1 输入点被占用或者输入点超出范围

0x0010	输出点配置错误
0x0011	轴 TO 地址指针检查错误
0x001A	轴 ID 错误
0x001B	正硬限位输入点被占用或者输入点超出范围
0x001C	负硬限位输入点被占用或者输入点超出范围
0x001D	急停输入点被占用或者输入点超出范围
0x001E	回零参考点输入点被占用或者输入点超出范围
0x001F	零位信号输入点被占用或者输入点超出范围
0x0020	无效输出类型
0x0021	编码器轴的输入模式设置错误
0x0022	编码器轴 A 相输入点被占用或者输入点超出范围（指第一个输入点）
0x0023	编码器轴 B 相输入点被占用或者输入点超出范围（指的是第二个输入点）
0x0024	编码器轴使用的 HSC 通道被占用或超出范围（例如 V3.0 中第一个编码器轴使用 HSC6，第二个编码器轴使用 HSC7）
0x0025	A 相输出点配置冲突
0x0026	B 相输出点配置冲突
0x0028	使用两个输入点时，编码器轴未组态在同一组中（I0.X 或 I1.X 指每个轴的组）
0x0029	无效轴 TO 指针元素数量
0x0030	PTO 轴的启停速度配置不合理 脉冲范围：[200 hz, 200k hz]
0x0031	编码器轴的速度滤波时间常量不合理 脉冲范围：[0 ms, 2000 ms]
0x0032	PTO 轴数量超限
0x0033	虚轴数量超限
0x0034	编码器轴数量超限
0x0035	PN 轴数量超限
0x0036	总轴数量超限
0x0037	PTO 输出点 0 配置错误
0x0038	PTO 输出点 1 配置错误
0x003A	PN 设备 ID 超限
0x003B	编码器类型配置错误
0x003C	主动回零模式配置错误

表格 14-33 运动轴组态的错误代码

错误代码	说明
0x0200	TO 版本错误
0x0201	测量单位不正确 STEP 7 Micro/WIN SMART V3 支持 0 - 脉冲，1 - 厘米，2 - 毫米，3 - 英寸，4 - 度
0x0202	PLS/REV 电机/编码器旋转一圈的脉冲数不正确，应该大于 0。 测量系统组态为脉冲，默认值为 1
0x0203	轴旋转一圈的移动距离不正确，应大于 0。 测量系统组态为脉冲，默认值为 1
0x0204	齿轮分子部分不正确，应大于 0。 测量系统组态为脉冲，默认值为 1
0x0205	齿轮分母部分不正确，应大于 0。 测量系统组态为脉冲，默认值为 1
0x0206	轴运动方式错误，只支持 0-线性轴，1-旋转轴
0x0207	模态轴从错误的位置启动。 模态轴的起始位置应大于单精度浮点数的最小值，模态轴的起始位置加上模式长度应小于单精度浮点数的最大值
0x0208	模态轴模长错误，应该大于 0
0x0209	编码器轴速度滤波时间常数错误， 滤波时间常数范围 [0 ms, 2000 ms]
0x020A	软件限位正开启且轴配为模态轴时, 位置 >= 模长+起始位置
0x020B	组态为直线轴时，下线位置 >= 上限位置 或组态为模态轴时，位置 >= 模式长度 + 起始位置 或组态为模态轴时，位置 < 起始位置
0x020C	默认加速度错误，加速度范围 [1 脉冲/s^2, 最大加速度]
0x020D	默认减速度错误，减速度范围 [1 脉冲/s^2, 最大减速度]
0x020E	默认加加速度错误，加加速度范围 [0, 2*10^8 脉冲/s^3]
0x020F	最大速度错误，进行脉冲相关转换后，应在范围 [200 Hz, 200 kHz] 内。
0x0210	最大加速度错误，加速度范围 [1 脉冲/s^2, 2*10^8 脉冲/s^2]
0x0211	最大减速度错误，减速度范围 [1 脉冲/s^2, 2*10^8 脉冲/s^2]
0x0212	STEP 7 Micro/WIN SMART V3 下发的 TO 中的轴 ID 非法

0x0213	STEP 7 Micro/WIN SMART V3 下发的 TO 中的轴类型非法
0x0215	该轴已激活轴急停位
0x0216	该轴已激活正向硬限位
0x0217	该轴已激活负向硬限位
0x0218	该轴已激活正向软限位
0x0219	该轴已激活负向软限位
0x021A	运动指令执行期间出错，请参见相应指令中的错误代码
0x0300	驱动器报错
0x0301	驱动器和 CPU 之间 SOL 信号丢失
0x0302	驱动器和 CPU 之间断联
0x0303	驱动器参数读取失败
0x0304	驱动器使能响应失败
0x0305	轴跟随误差超差
0x0310	轴位置监视错误
0x0311	轴夹紧错误

表格 14-34 指令引脚报告的常见错误的错误代码

错误代码	说明
0x1000	CPU 不支持该指令
0x1001	TO_AXIS 不支持该指令
0x1002	在同一 TO_Axis 上执行多个指令而导致的错误
0x1003	TO_AXIS 不存在或无效。
0x1004	TO_Axis 配置无效
0x1005	TO_AXIS 的 PLCopen 状态错误
0x1006	轴组在运行中，无法执行当前的单轴指令
0x1007	由于单轴指令正在运行而导致的错误
0x1008	由于零位未知（未归位）而导致的错误
0x1009	此指令不支持缓冲模式
0x1010	TO_AXIS 缓冲的指令过多
0x1011	Direction 参数非法
0x1015	距离/位置非法
0x1016	速度超出范围

0x1017	加速度/减速度超出范围
0x1018	加加速度超出范围
0x1019	指令被同时触发的更高优先级的任务中断
0x1020	指令的输入 ProbeID 无效（只能是 0 或 1）
0x1022	有窗口时，指令输入的 WindowStart 和 WindowEnd 非法，超过浮点数的最大最小限制
0x1023	中断沿触发模式非法（仅支持 0：上升沿和 1：下降沿。）
0x1024	探针未启用
0x1025	当 WindowOnly=true 时，模态轴输入的 FirstPostion 和 LastPosition 不在范围内（0-模长）
0x1028	减速停过程中不能执行其他运动指令
0x1029	打断非法
0x1030	触发已开启的正软限位开关
0x1031	触发已开启的负软限位开关
0x1032	检测到 TO 轴上有错误
0x1033	PN IRT 轴执行指令超时，轴未能在预设的容差时间内成功完成操作（包括轴未能达到目标位置或零速）并置位 Done
0x1035	在扭矩控制模式下执行了运动指令
0x1036	运动方向与夹紧方向相反
0x1037	指令不支持速度模式
0x1038	轴不支持速度模式
0x1039	无效的缓冲模式
0x1040	功能块执行期间轴 ID 已更改。轴 ID 重启错误
0x1080	Jogforward 和 JogBackward 都在上升沿
0x1FFF	内部意外错误代码

表格 14-35　单轴运动控制的错误代码

错误代码	说明
0x2000	不允许多实例调用 MC_Power
0x2001	轴使能失败：驱动器未就绪
0x2002	轴已被控制面板预定占用
0x2400	轴的速度不为 0，说明减速停没有完成

0x2500	轴配置中未使能 RPS
0x2501	不支持所选回零模式（输入了 0、1、3 以外的数字）
0x2502	当前轴正在执行回零
0x2503	回零过程中，信号出现非预期行为
0x2504	position 超出有效范围
0x2505	PN 编码器异常
0x2506	主动回零子模式错误
0x2507	触发回零指令时驱动器未就绪
0x2600	MC_Immediatestop 执行过程中不能再执行 MC_stop
0x2C00	当 MC_AbortTrigger 或 MC_TouchProbe 功能块实例的 Busy 状态为 TRUE 时，若再次调用该实例的 Execute 管脚，系统将触发此错误。
0x2C01	执行 MC_abort trigger 时发现没有运行 MC_touchprobe
0x2C02	执行 MC_abort trigger 或者 MC_touchprobe 时发现轴正在执行回零操作
0x2C03	编码器错误（仅供 PN IRT 轴使用）
0x2C04	驱动器不支持探针功能（仅供 PN IRT 轴使用）
0x1090	使用控制面板时，如果 PLC 与 STEP 7-Micro/Win SMART V3 之间的连接中断。
0x1091	控制面板接管控制权时执行功能块

表格 14-36　扭矩控制的错误代码

错误代码	说明
0x5200	轴组态没有配置 750 报文
0x5202	重复执行扭矩控制指令
0x5203	在非 PN IRT 轴上执行扭矩控制指令
0x5204	无效 Torque 参数
0x5205	无效 Ramp 参数
0x5206	执行 mode=1 的扭矩控制指令时，轴正在执行运动执行
0x5207	无效 mode 参数
0x5300	使用 MC_TorqueLimiting 指令需要配置 105 号报文
0x5301	固定挡块模式下，轴状态处于 Disabled
0x5302	无效参数
0x5303	InClamping 状态异常退出，例如固定挡块被移除或被回推

错误代码	说明
0x5304	Torque Limiting 功能块与其他力矩控制功能块（如 MC_TorqueControl 或 MC_TorqueRange）存在冲突
0x5305	InClamping 状态下，不得禁用 [Enable Pin]
0x5400	UpperLimit/LowerLimit 参数非法
0x5500	驱动器未就绪

表格 14-37 控制面板相关错误代码

错误代码	说明
0x8104	兼容旧版 PTO 控制面板错误代码，例如 sps7_COMMAND_NOT_SUPPORTED_ERROR
0x9000	CPU 未运行，无法使用控制面板
0x9001	当前功能块正在控制，无法使用控制面板
0x9002	TO 轴处于错误状态，无法使用控制面板
0x9003	控制面板指令无效
0x9004	当前轴未处于 Standstill 状态
0x9005	此轴不支持该命令

14.9.2 电子凸轮和齿轮的错误代码

表格 14-38 电子凸轮和齿轮的错误代码

错误代码	说明
0x0400	无效 Cam Table ID
0x0401	任务类型错误
0x0402	由于位置超限，任务被删除
0x4000	电子齿轮主轴轴号与从轴轴号相等
0x4001	电子齿轮主轴轴号需要小于从轴轴号
0x4002	PN IRT 主轴执行 Power Off 后，设定值失效
0x4005	GearIn 分母参数合法性校验
0x4006	GearIn 分子参数合法性校验
0x4007	追赶或者同步过程中速度超过了轴配置中的最大速度

0x4008	主轴速度类型选择非法
0x4100	当前执行的指令不是 GearIn
0x4200	电子凸轮主轴轴号与从轴轴号相等
0x4201	电子凸轮主轴轴号需要小于从轴轴号
0x4202	指令方向只能为 0 正向、1 负向、2 无方向
0x4203	startmode 只支持 0 绝对 1 相对
0x4204	CamIn StartPosition 参数合法性校验
0x4205	camin MasterStartDistance 参数合法性校验
0x4206	camin MasterScaling 参数合法性校验
0x4207	camin SlaveScaling 参数合法性校验
0x4208	camin MasterOffset 参数合法性校验
0x4209	camin SlaveOffset 参数合法性校验
0x4210	凸轮表初始化失败
0x4211	凸轮表初始化尚未完成
0x4212	凸轮表当前正在由 GenerateCamTable 使用。
0x4213	凸轮表数据错误
0x4300	执行 CamOut 时无 CamIn 指令
0x4301	无效凸轮主轴参考值
0x4400	同步运动凸轮/齿轮和相位均发生错误
0x4401	主轴 ID 不正确。当前主轴必须与当前齿轮或当前凸轮相同
0x4402	PhaseShift 超出范围，相位值不能为 INF 或者 NAN
0x4403	出现非预期的错误
0x4500	凸轮表插补计算资源被占用
0x4501	凸轮表正在被 MC_CamIn 使用
0x4502	凸轮表关键点数据为空
0x4503	凸轮表关键点数据地址错误
0x4504	凸轮表数据对象地址错误
0x4505	节点数小于 2 或大于 361。
0x4506	凸轮表计算超时
0x4507	property 中的 period 参数超出有效范围
0x4508	凸轮表曲线类型无效
0x4509	property 中的初始速度超出有效范围

0x450a	property 中的初始加速度超出有效范围
0x450b	主轴相位未按升序排序
0x450c	主轴相位过于接近
0x450d	主轴相位超出有效范围
0x450e	从轴位置超出有效范围
0x450f	从轴速度超出有效范围
0x4510	从轴加速度超出有效范围
0x4511	插值间隔超出范围
0x4512	计算结果需要的 datanum，超出了预留容量
0x4513	凸轮表插补数据地址为空
0x4514	凸轮表插补数据地址错误
0x4515	凸轮表插值被 CPU 停止中断
0x4517	GenerateCamTable 指令执行时发生内部错误

14.9.3 轴组的错误代码

错误代码	说明
0x0100	无效轴组配置字节
0x0101	无效轴组维度
0x0102	无效 X 轴轴 ID
0x0103	无效 Y 轴轴 ID
0x0104	无效 Z 轴轴 ID
0x0105	无效辅助轴轴 ID
0x0106	无效轴组 TO 指针区域
0x0107	无效轴组 TO 指针类型
0x0108	无效轴组 TO 指针元素数量
0x0109	轴组 ID 错误
0x010A	轴组 SDB Record ID 重复
0x010B	轴组 SDB Record 长度错误
0x010C	未知版本
0x010D	X 轴轴 ID 被占用

0x010E	Y 轴轴 ID 被占用
0x010F	Z 轴轴 ID 被占用
0x0110	辅助轴轴 ID 被占用
0x0111	轴组 TO 地址指针检查错误
0x0112	轴组数目超限（最多支持两个轴组）
0x0381	轴组因轴错误而停止
0x1026	轴组不存在
0x1027	轴组有错误，且不处于 ErrorStop 状态
0x5100	轴组指令因轴错误而执行失败
0x5101	有单轴同时在多个轴组中使能
0x5200	轴组态没有配置 750 报文

单轴运动的错误代码 (页 1068)

14.10 示例程序

14.10.1 运动控制示例

使用设备

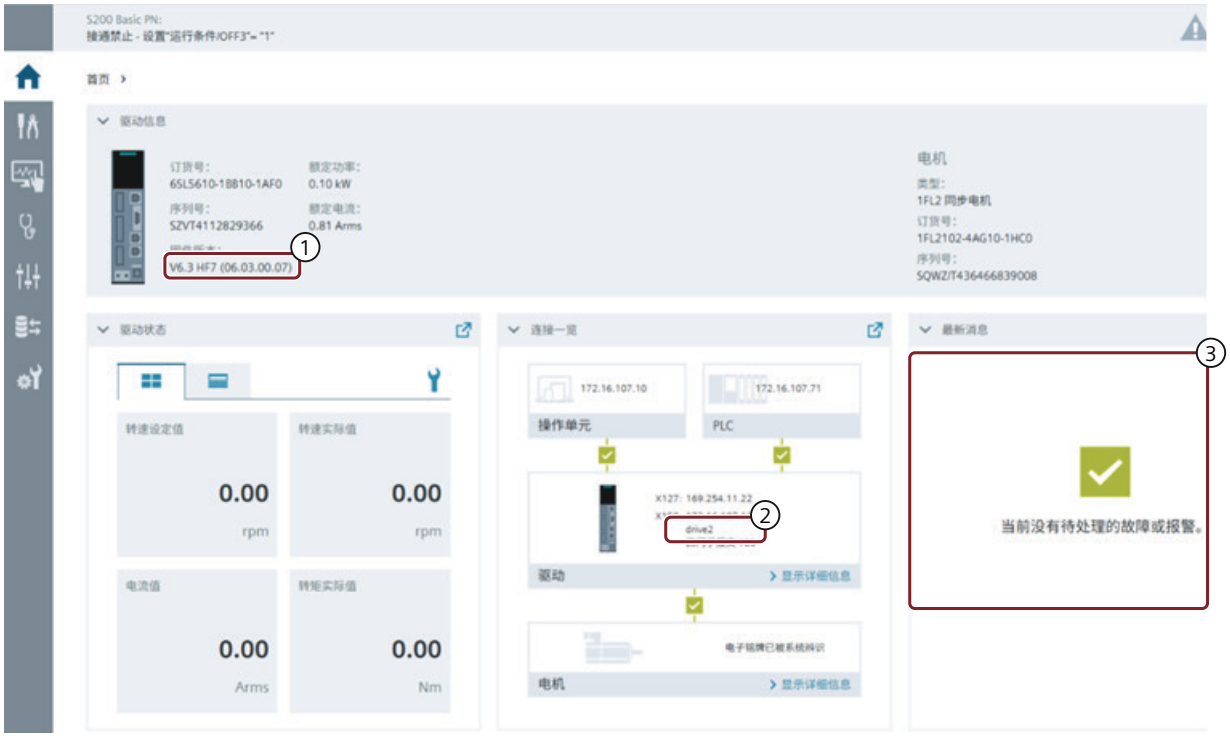
- CPU ST32
- 伺服驱动器 S200

设备连接

使用以太网线连接 CPU ST32 的 P2 端口 与 S200 驱动器的 P1 端口。

检查 S200 驱动器

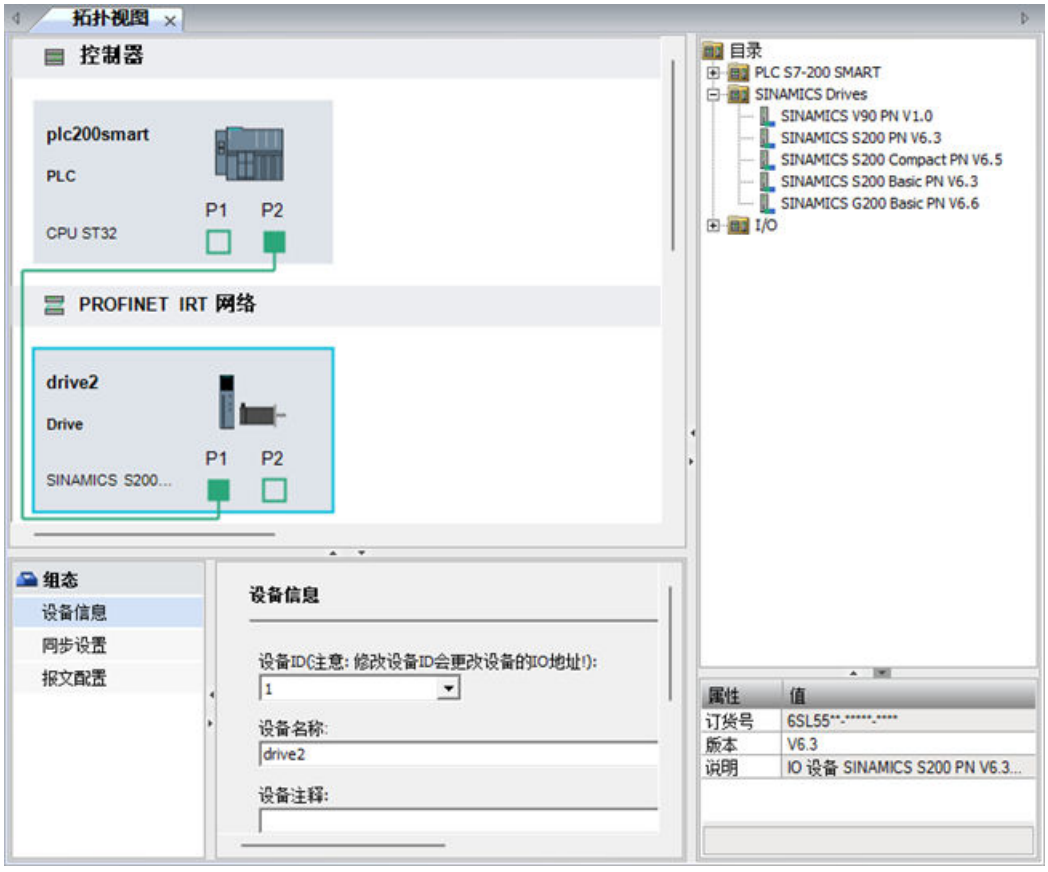
1. 将您的电脑通过以太网线连接到 S200 驱动器。
2. 打开浏览器，输入 S200 驱动器的 IP 地址。
   可查阅 S200 驱动器手册以获取 S200 驱动器默认信息：
   – S200 IP: 192.168.2.1
   – S200 用户名： Administrator
   – 密码：sinamicsS200
3. 在 S200 驱动器的网络服务器中，确认以下信息：
   – 固件版本：本例中为 V6.3
   – 设备名称：本例中为 driver2
   – 检查报警：确保 S200 驱动器没有显示任何报警或错误信息



组态 PROFINET 网络

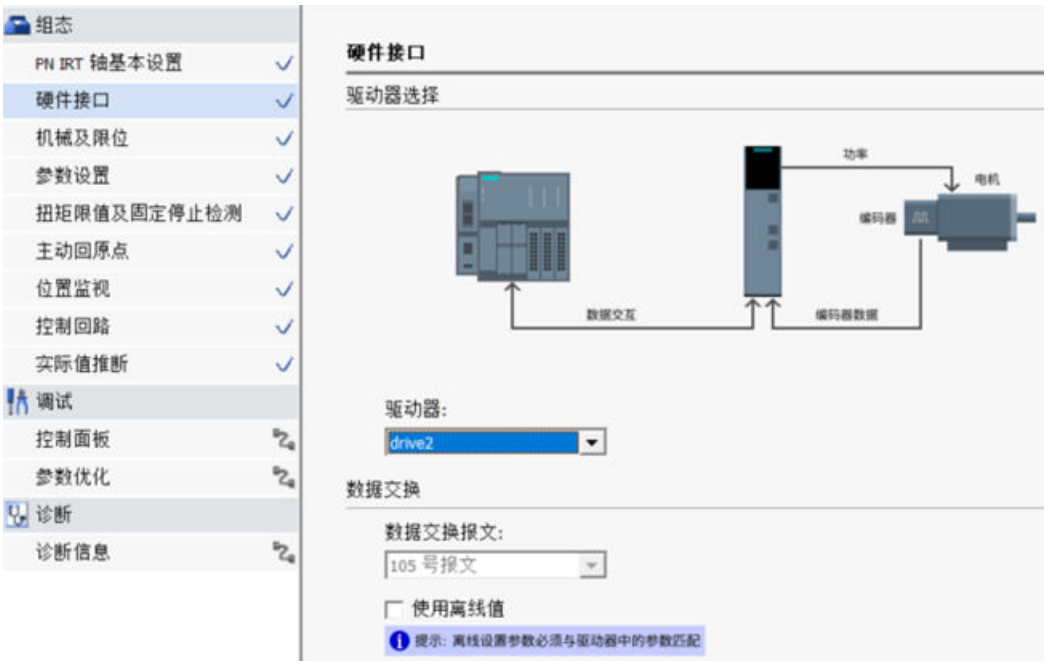
1. 打开 STEP 7-Micro/WIN SMART V3。
2. 在项目树中，双击“硬件网络拓扑 (页 833)”， 打开硬件网络拓扑 。
3. 在右侧的设备目录 "SINAMICS Drives" 中，选择固件版本为 V6.3 的 S200 PN 驱动器 "SINAMICS S200 PN V6.3"， 并将其拖拽到 PROFINET IRT 网络中。

4. 选中控制器，在组态窗口中设置其站名称和 IP 地址。
5. 选择刚刚添加的 S200 驱动器，在组态窗口中设置其设备名称 IP 地址和 报文。
   – 设备名称须和 S200 Webserver 中的名称保持一致，本例中为 driver2
   – 确保控制器和 S200 驱动器的 IP 地址在同一个 IP 网段内
   – 本例配置使用 105 报文



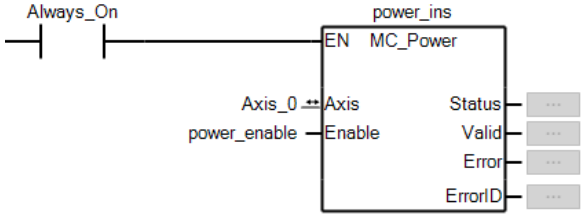
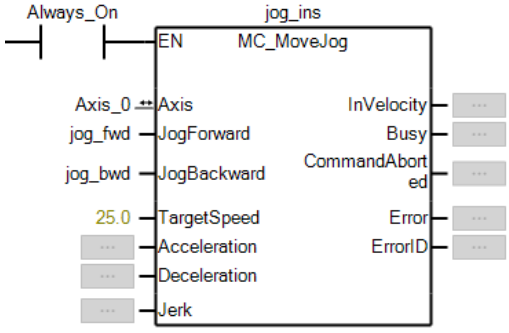
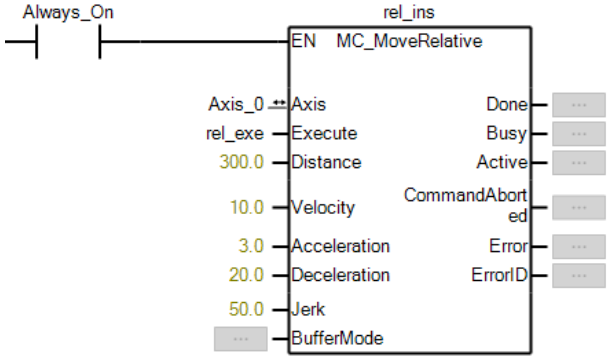
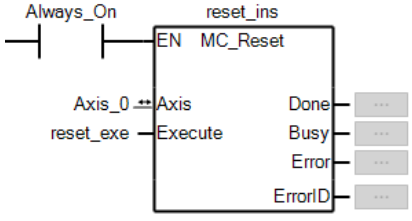
组态轴

1. 在项目树中，右键单击“运动控制”，然后选择“添加”>“轴”>“PN IRT 轴”。
2. 在 PN IRT 轴的组态界面 (页 884)中，点击“硬件接口”。
3. 将之前配置的 S200 驱动器（通过其设备名称）分配给该轴。



编写运动控制程序

1. 在项目树中，打开“程序块”文件夹，可右键单击 POU 打开 POU、编辑其属性、用密码对其进行保护或重命名。
2. 添加下列运动控制指令

LAD/FBD	说明
程序段 1：  	用于使能轴
程序段 2：  	Axis_0，点动功能
程序段 3：  	Axis_0，用于相对位移控制
程序段 4：  	用于复位 Axis_0

编译程序

点击工具栏上的“编译”按钮，确保没有编译错误。

如果出现编译错误，请根据错误信息进行排查和修正。

下载程序

1. 查找并登录 PLC。
2. 点击工具栏上的“下载”按钮，将编译好的程序下载到 PLC。

执行运动指令

> ⚠ **警告**
>
> **安全操作**
>
> 在执行任何运动控制程序前，请务必确认所有安全防护措施均已就位且功能正常，并确保操作区域内无人员或障碍物。不当操作可能导致设备损坏或人身伤害。

1. 在在线模式下，触发 MC_Power 以使能轴。
2. 触发 MC_MoveJog, MC_MoveRelative 等指令，观察轴的运动。

查看 POU 的程序状态

点击工具栏上的“程序状态 (页 773)”按钮，并监控程序块和轴的状态。

驱动器参数优化

在 PN IRT 轴的组态界面中，可在“参数优化 (页 918)”页面对 S200 驱动器的参数进行进一步的优化

跟踪轴运动轨迹

打开“轨迹 (页 192)”功能, 组态轨迹以记录轴的实际位置、速度等参数，以便分析运动轨迹。

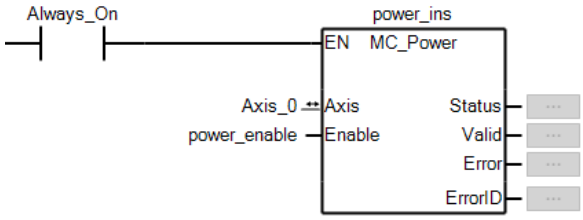
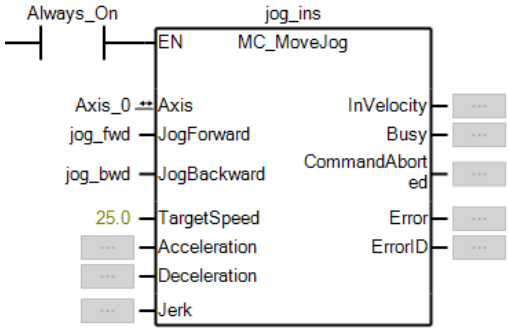
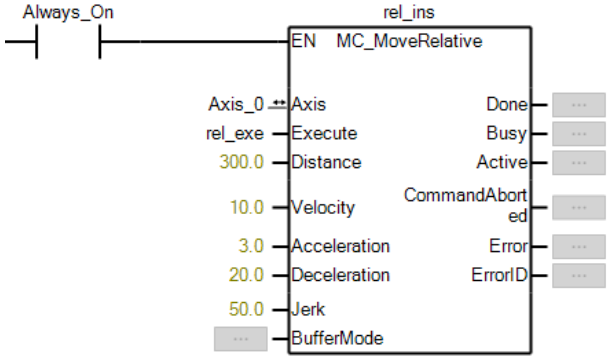
参见

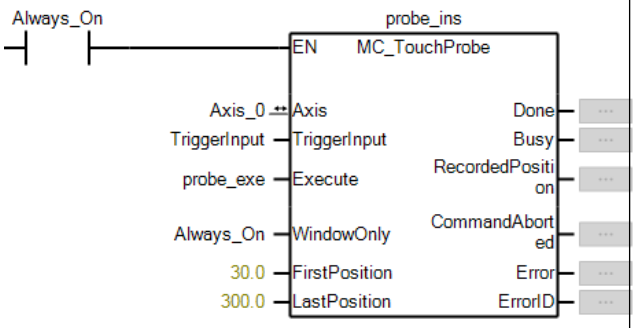
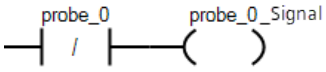
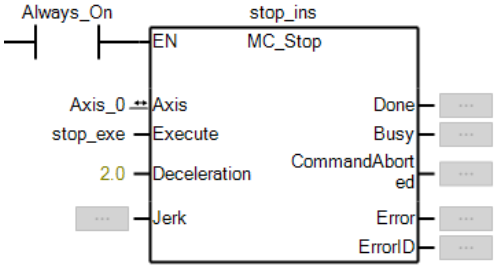
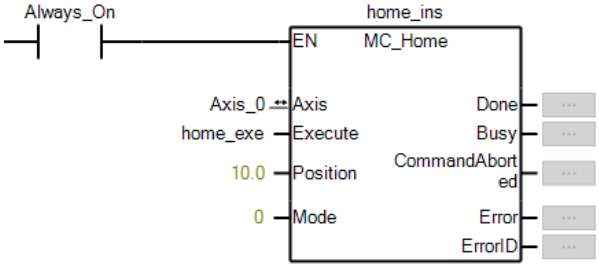
参数优化 (页 918)

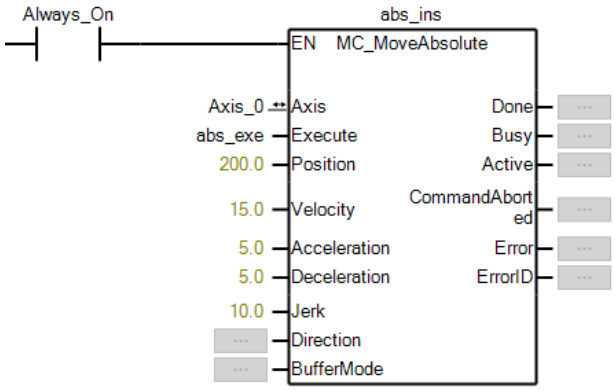
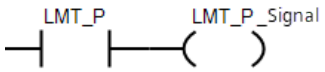
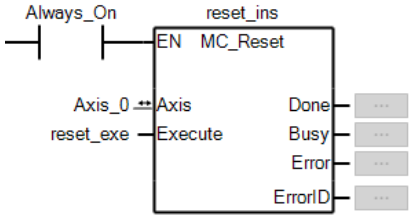
14.10.2 单轴示例

以下程序演示了多个单轴功能的调用，例如 Enable、Jog 和 Probe。

表格 14-39 示例：单轴

LAD/FBD	说明	STL
程序段 1：  	启用轴	LD Always_On FBCALL MC_Power, power_ins, Axis_0, power_enable, power_ins.Status, power_ins.Valid, power_ins.Error, power_ins.ErrorID
程序段 2：  	Axis_0，点动功能，点动目标速度为 25	LD Always_On FBCALL MC_MoveJog, jog_ins, Axis_0, jog_fwd, jog_bwd, 25.0, Axis_0.Axis_Config.Dynamic _Default.Default_Acc, Axis_0.Axis_Config.Dynamic _Default.Default_Decel, Axis_0.Axis_Config.Dynamic _Default.Default_Jerk, jog_ins.InVelocity, jog_ins.Busy, jog_ins.CommandAborted, jog_ins.Error, jog_ins.ErrorID
程序段 3：  	轴 Axis_0 将根据以下运动参数执行相对定位： 目标位置： 300.0 速度： 10.0 加速度： 3.0 减速度： 20.0 加加速度 ： 50.0	LD Always_On FBCALL MC_MoveRelative, rel_ins, Axis_0, rel_exe, 300.0, 10.0, 3.0, 20.0, 50.0, rel_ins.BufferMode, rel_ins.Done, rel_ins.Busy, rel_ins.Active, rel_ins.CommandAborted, rel_ins.Error, rel_ins.ErrorID

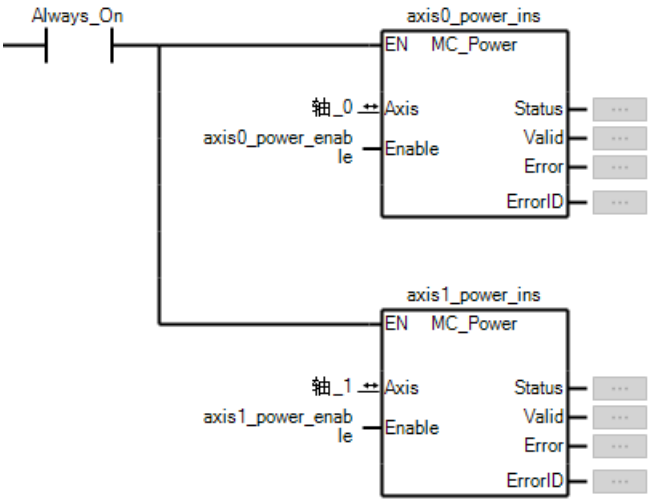
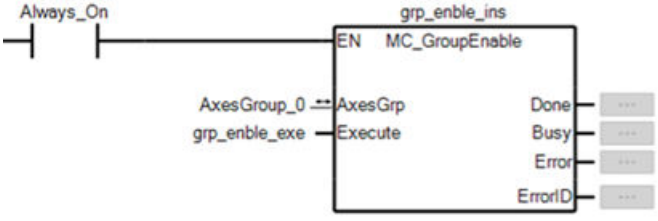
LAD/FBD	说明	STL
程序段 4：  	Axis_0，探针	LD Always_On FBCALL MC_TouchProbe, probe_ins, Axis_0, TriggerInput, probe_exe, Always_On, 30.0, 300.0, probe_ins.Done, probe_ins.Busy, probe_ins.RecordedPosition, probe_ins.CommandAborted, probe_ins.Error, probe_ins.ErrorID
程序段 5：  	触发探针	LDN probe_0 = CPU_Output11
程序段 6：  	Axis_0 以 2.0 的减速度减速至停止	ID Always_On FBCALL MC_Stop, stop_ins, Axis_0, stop_exe, 2.0, Axis_0.Axis_Config.Dynamic_Default.Default_Jerk, stop_ins.Done, stop_ins.Busy, stop_ins.CommandAborted, stop_ins.Error, stop_ins.ErrorID
程序段 7：  	操作模式：模式 0 原点偏移量：10.0 执行结果： Axis_0 的当前绝对位置被成功设置为 10.00	LD Always_On FBCALL MC_Home, home_ins, Axis_0, home_exe, 10.0, 0, home_ins.Done, home_ins.Busy, home_ins.CommandAborted, home_ins.Error, home_ins.ErrorID

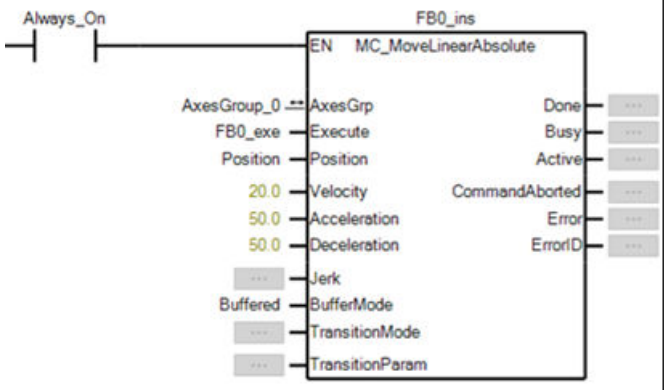
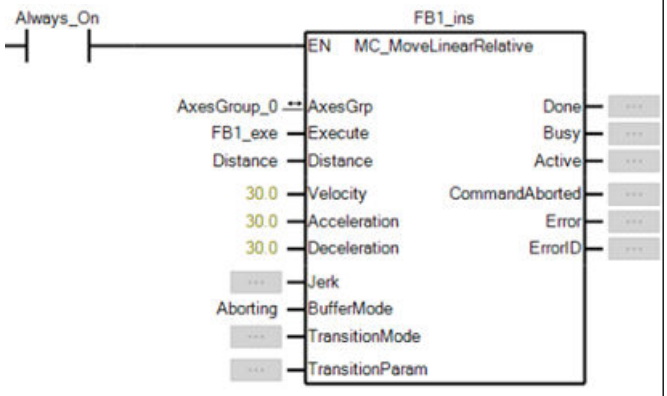
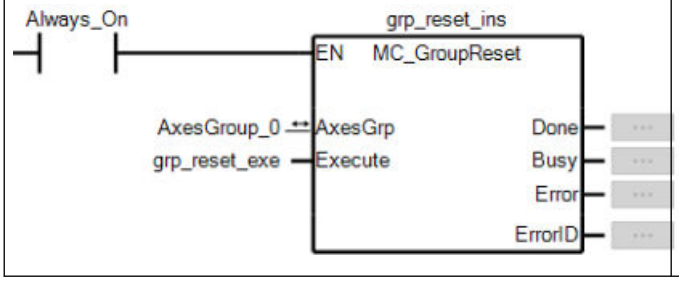
LAD/FBD	说明	STL
程序段 8：  	轴 Axis_0 将根据以下运动参数执行绝对定位： 目标位置： 200.0 速度： 15.0 加速度： 5.0 减速度： 5.0 加加速度 ： 10.0	**LD Always_On** **FBCALL MC_MoveAbsolute, abs_ins, Axis_0, abs_exe, 200.0, 15.0, 5.0, 5.0, 10.0, abs_ins.Direction, abs_ins.BufferMode, abs_ins.Done, abs_ins.Busy, abs_ins.Active, abs_ins.CommandAborted, abs_ins.Error, abs_ins.ErrorID**
程序段 9：  	触发正限位信号	**LD LMT_P** **= CPU_Output9**
程序段 10：  	复位 Axis_0	**LD Always_On** **FBCALL MC_Reset, reset_ins, Axis_0, reset_exe, reset_ins.Done, reset_ins.Busy, reset_ins.Error, reset_ins.ErrorID**

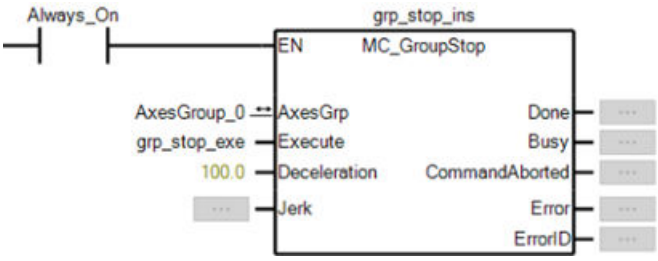
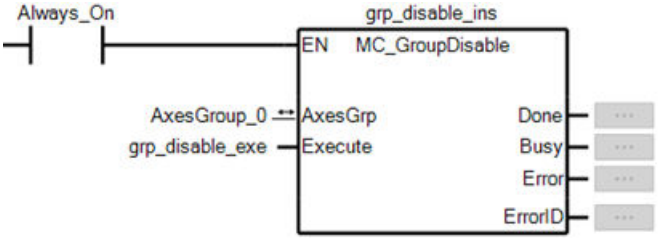
14.10.3 轴组示例

以下程序演示了轴组的调用。

表格 14-40 示例：轴组

LAD/FBD	说明	STL
程序段 1：  	使能轴组中的单轴	LD Always_On FBCALL MC_Power, axis0_power_ins, 轴_0, axis0_power_enable, axis0_power_ins.Status, axis0_power_ins.Valid, axis0_power_ins.Error, axis0_power_ins.ErrorID FBCALL MC_Power, axis1_power_ins, 轴_1, axis1_power_enable, axis1_power_ins.Status, axis1_power_ins.Valid, axis1_power_ins.Error, axis1_power_ins.ErrorID
程序段 2：  	使能轴组	LD Always_On FBCALL MC_GroupEnable, grp_enble_ins, AxesGroup_0, grp_enble_exe, grp_enble_ins.Done, grp_enble_ins.Busy, grp_enble_ins.Error, grp_enble_ins.ErrorID

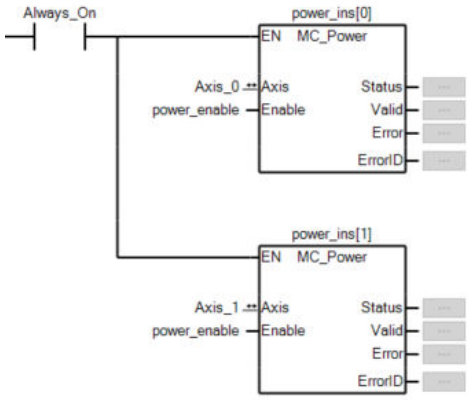
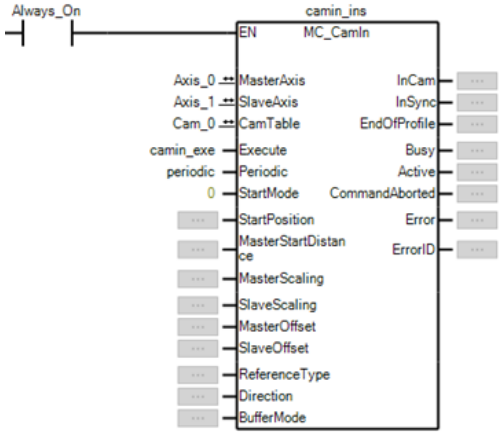
LAD/FBD	说明	STL
程序段 3：  	轴组 AxesGroup_0 将根据以下运动参数执行绝对定位： 速度： 20.0 加速度： 50.0 减速度： 50.0	LD Always_On FBCALL MC_MoveLinearAbsolute, FB0_ins, AxesGroup_0, FB0_exe, Position, 20.0, 50.0, 50.0, FB0_ins.Jerk, Buffered, FB0_ins.TransitionMode, FB0_ins.TransitionParam, FB0_ins.Done, FB0_ins.Busy, FB0_ins.Active, FB0_ins.CommandAborted, FB0_ins.Error, FB0_ins.ErrorID
程序段 4：  	轴组 AxesGroup_0 将根据以下运动参数执行相对定位： 速度： 30.0 加速度： 30.0 减速度： 0.0	LD Always_On FBCALL MC_MoveLinearRelative, FB1_ins, AxesGroup_0, FB1_exe, Distance, 30.0, 30.0, 30.0, FB1_ins.Jerk, Aborting, FB1_ins.TransitionMode, FB1_ins.TransitionParam, FB1_ins.Done, FB1_ins.Busy, FB1_ins.Active, FB1_ins.CommandAborted, FB1_ins.Error, FB1_ins.ErrorID
程序段 5：  	复位轴组	LD Always_On FBCALL MC_GroupReset, grp_reset_ins, AxesGroup_0, grp_reset_exe, grp_reset_ins.Done, grp_reset_ins.Busy, grp_reset_ins.Error, grp_reset_ins.ErrorID

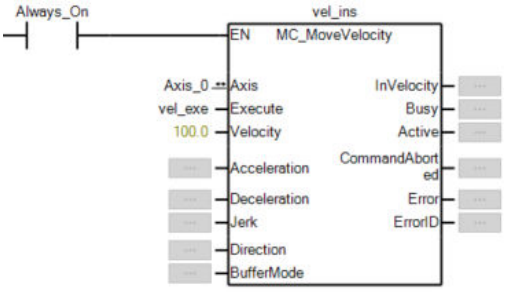
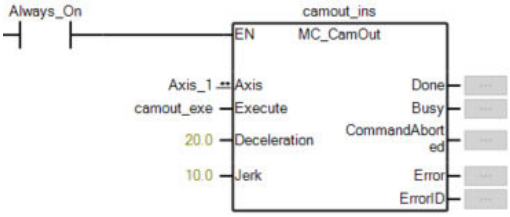
LAD/FBD	说明	STL
程序段 6：  	停止轴组	LD Always_On FBCALL MC_GroupStop, grp_stop_ins, AxesGroup_0, grp_stop_exe, 100.0, grp_stop_ins.Jerk, grp_stop_ins.Done, grp_stop_ins.Busy, grp_stop_ins.CommandAborted, grp_stop_ins.Error, grp_stop_ins.ErrorID
程序段 7：  	禁用轴组功能	LD Always_On FBCALL MC_GroupDisable, grp_disable_ins, AxesGroup_0, grp_disable_exe, grp_disable_ins.Done, grp_disable_ins.Busy, grp_disable_ins.Error, grp_disable_ins.ErrorID

14.10.4 凸轮示例

以下程序演示了如何调用凸轮功能，例如 Enable、CamIn 和 CamOut。

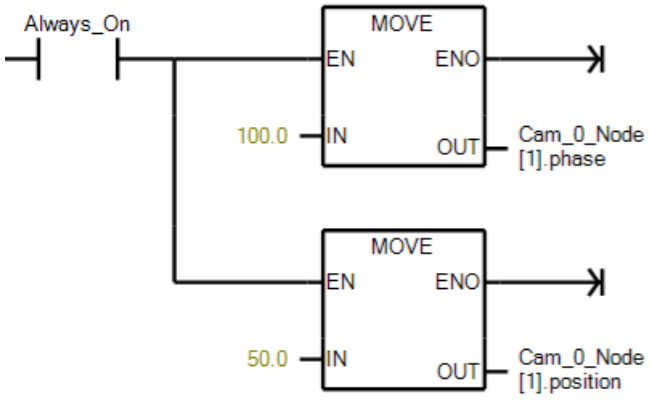
表格 14-41 示例：凸轮

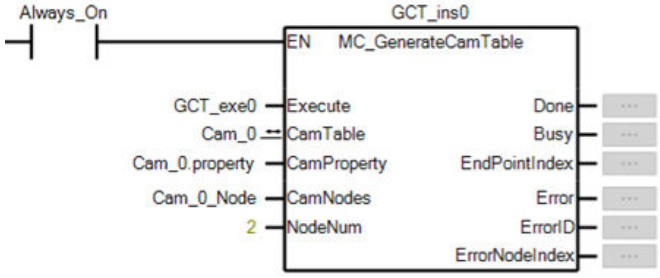
LAD/FBD	说明	STL
程序段 1：  	启用轴	LD Always_On FBCALL MC_Power, power_ins[0], Axis_0, power_enable, power_ins[0].Status, power_ins[0].Valid, power_ins[0].Error, power_ins[0].ErrorID FBCALL MC_Power, power_ins[1], Axis_1, power_enable, power_ins[1].Status, power_ins[1].Valid, power_ins[1].Error, power_ins[1].ErrorID
程序段 2：  	触发从轴 MC_CamIn	LD Always_On FBCALL MC_CamIn, camin_ins, Axis_0, Axis_1, Cam_0, camin_exe, periodic, 0, camin_ins.StartPosition, camin_ins.MasterStartDistance, camin_ins.MasterScaling, camin_ins.SlaveScaling, camin_ins.MasterOffset, camin_ins.SlaveOffset, camin_ins.ReferenceType, camin_ins.Direction, camin_ins.BufferMode, camin_ins.InCam, camin_ins.InSync, camin_ins.EndOfProfile, camin_ins.Busy, camin_ins.Active, camin_ins.CommandAborted, camin_ins.Error, camin_ins.ErrorID

LAD/FBD	说明	STL
程序段 3：  	触发主轴连续运动	LD Always_On FBCALL MC_MoveVelocity, vel_ins, Axis_0, vel_exe, 100.0, Axis_0.Axis_Config.Dynamic_Default.Default_Acc, Axis_0.Axis_Config.Dynamic_Default.Default_Decel, Axis_0.Axis_Config.Dynamic_Default.Default_Jerk, vel_ins.Direction, vel_ins.BufferMode, vel_ins.InVelocity, vel_ins.Busy, vel_ins.Active, vel_ins.CommandAborted, vel_ins.Error, vel_ins.ErrorID
程序段 4：  	解耦从轴凸轮	LD Always_On FBCALL MC_CamOut, camout_ins, Axis_1, camout_exe, 20.0, 10.0, camout_ins.Done, camout_ins.Busy, camout_ins.CommandAborted, camout_ins.Error, camout_ins.ErrorID

示例：使用系统生成的 CamProperty、 CamNode

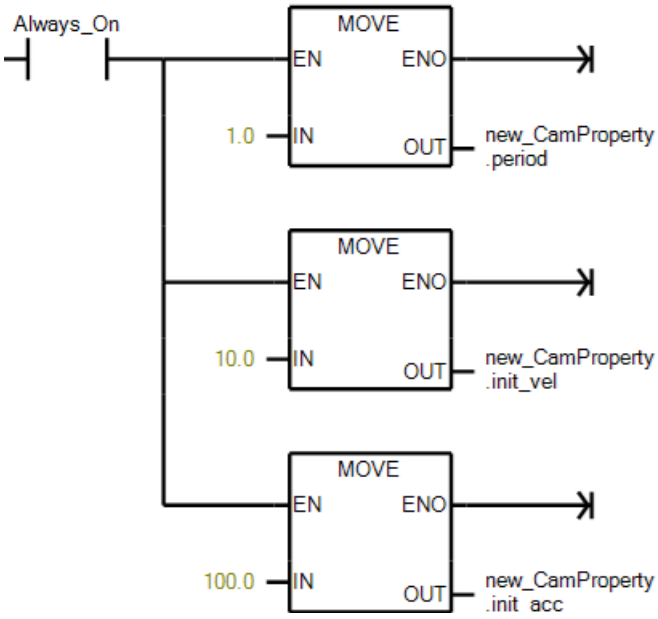
LAD/FBD	说明	STL

程序段 1：
 
 	
	修改 Cam_0_Node[1] 的主轴相位和从轴位置。	**LD Always_On**
		MOVE 100.0, Cam_0_Node[1].phase
		MOVE 50.0, Cam_0_Node[1].position

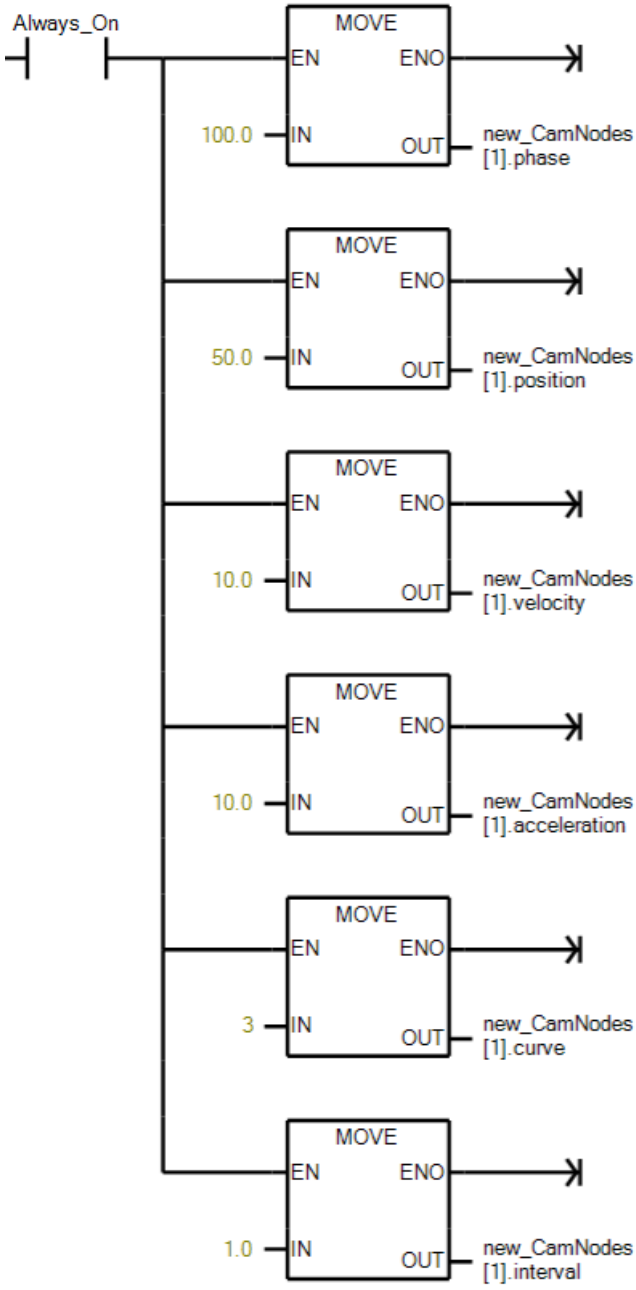
程序段 2：  	触发器 MC_GenerateCamTable	**LD Always_On** **FBCALL MC_GenerateCamTable, GCT_ins0, GCT_exe0, Cam_0, Cam_0.property, Cam_0_Node, 2, GCT_ins0.Done, GCT_ins0.Busy, GCT_ins0.EndPointIndex, GCT_ins0.Error, GCT_ins0.ErrorID, GCT_ins0.ErrorNodeIndex**

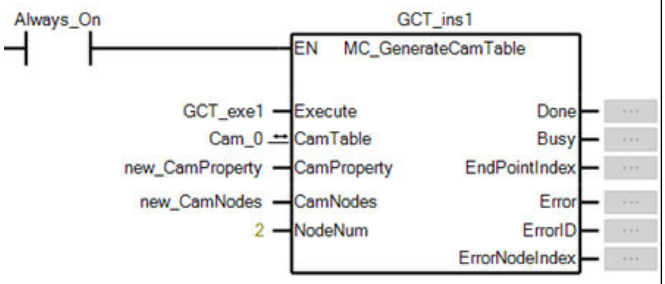
示例：使用创建的全局变量 CamProperty、CamNode

LAD/FBD	说明	STL

程序段 1：
 
 	
	1. 创建两个全局变量：
	– new_CamProperty.period（类型：SYS_CAM_PROPERTY）
	– new_CamNodes（类型：Array of SYS_CAM_NODES）
	2. 设置 new_CamProperty 的参数
		LD Always_On
		MOVE 1.0, new_CamProperty.period
		MOVE 10.0, new_CamProperty.init_vel
		MOVE 100.0, new_CamProperty.init_acc

LAD/FBD	说明	STL

程序段 2：
 
 	
	设置 new_CamNode 参数，将第一个区段组态为第 5 次曲线	**LD Always_On**
		MOVE 100.0, new_CamNodes[1].phase
		MOVE 50.0, new_CamNodes[1].position
		MOVE 10.0, new_CamNodes[1].velocity
		MOVE 10.0, new_CamNodes[1].acceleration
		MOVE 3, new_CamNodes[1].curve
		MOVE 1.0, new_CamNodes[1].interval

LAD/FBD	说明	STL
程序段 3：  	触发器 MC_GenerateCamTable	**LD Always_On** FBCALL MC_GenerateCamTable, GCT_ins1, GCT_exe1, Cam_0, new_CamProperty, new_CamNodes, 2, GCT_ins1.Done, GCT_ins1.Busy, GCT_ins1.EndPointIndex, GCT_ins1.Error, GCT_ins1.ErrorID, GCT_ins1.ErrorNodeIndex

开环运动控制（原有） 15

说明

运动向导默认情况下禁用

- 运动向导现仅适用于迁移使用 STEP 7-Micro/WIN SMART V2.8 创建的项目。
- 迁移后，可以使用运动向导修改使用 STEP 7-Micro/WIN SMART V2.8 创建的项目。
- 对于任何新项目，访问新的运动控制 (PLCopen) (页 805)进行组态。

通过运动向导，可定义运动轴的参数，并为当前应用定义一组位置曲线。

S7-200 SMART CPU 提供了四种开环运动控制方法：

- 脉冲串输出 (PTO)：内置在 CPU 的速度和位置控制。请参见脉冲输出指令。（注：无 PTO 向导可用。使用运动控制向导替代。）
- 脉宽调制 (PWM)：内置在 CPU 的速度、位置或负载循环控制。请参见脉冲输出指令。
- 运动轴：内置在 CPU 中的轴速度和位置控制方法。
- 运动轴组：内置在 CPU 中的轴组速度和位置控制方法。

CPU 提供了四个数字输出（Q0.0、Q0.1、Q0.2 和 Q0.3），可以通过 PLS 指令组态为 PTO 或 PWM 输出，通过 PWM 向导组态为 PWM 输出，或通过运动控制向导组态为运动控制输出。

说明

Q0.2 仅用于 ST60。

说明

只有 SR 和 ST CPU 型号支持运动控制。

CPU ST20 型号仅支持运动控制轴组进行 2D 直线插补运动；CPU ST30、CPU ST40 和 CPU ST60 型号支持运动控制轴组进行 2D 和 3D 直线插补运动。

为 PTO 操作组态输出后，CPU 会生成一个占空比为 50% 的脉冲串，用于对步进电机或伺服电机的速度和位置进行开环控制。内置 PTO 功能仅提供脉冲串输出。应用程序必须使用 PLC 内置的或扩展模块提供的 I/O 来提供方向和极限控制。

若组态 PWM 输出，CPU 将固定输出的循环时间，通过程序控制脉冲的持续时间或负载周期。可通过脉冲持续时间的变化来控制应用的转速或位置。

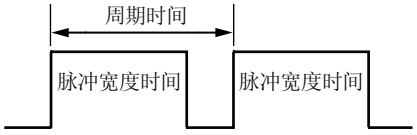
运动轴提供了带有集成方向控制和禁用输出的单脉冲串输出。运动轴还包括可编程输入，允许将 CPU 组态为包括自动参考点搜索在内的多种操作模式。运动轴为步进电机或伺服电机的速度和位置开环控制提供了统一的解决方案。

为了简化在应用中使用运动控制的流程，STEP 7-Micro/WIN SMART V3 提供了运动向导来组态运动轴和运动轴组，并提供了 PWM 向导来组态 PWM。这些向导会生成运动指令，用户可以使用这些指令在应用中对速度和运动进行动态控制。对于运动轴，STEP 7-Micro/WIN SMART V3 还提供了控制面板，可以通过该控制面板控制、监视和测试运动操作。

15.1 使用 PWM 输出

15.1.1 使用 PWM（脉宽调制）输出

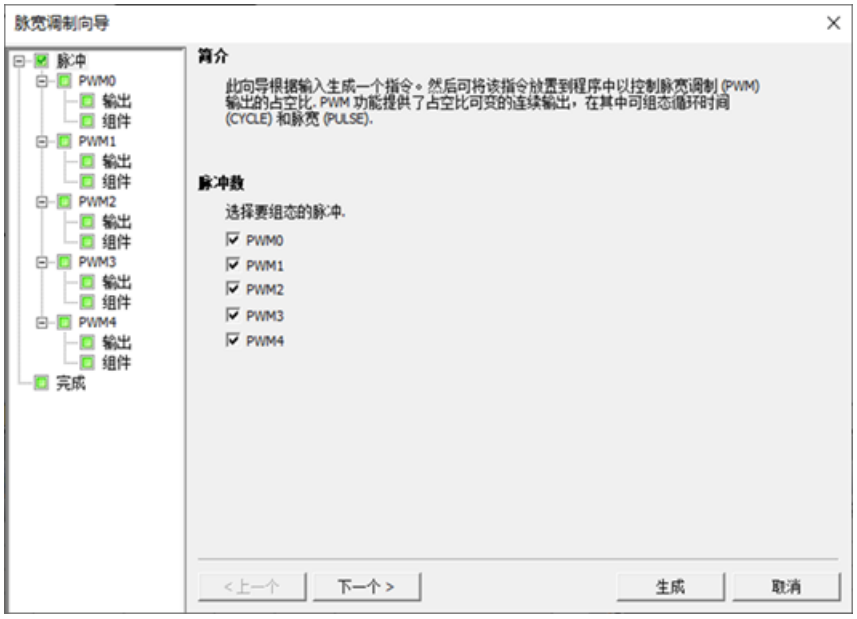
PWM 提供了占空比可变周期固定的输出。PWM 输出以指定频率（循环时间）启动之后将连续运行。可根据需要调节脉冲宽度，进而实现所需控制。占空比可表示为周期的百分比或对应于脉冲宽度的时间值。脉宽的变化范围为 0%（无脉冲，始终为低电平)）到 100%（无脉冲，始终为高电平）。请参见下图。



由于 PWM 输出可从 0% 变化到 100% ，因此在很多情况下，它可以提供一个类似于模拟量输出的数字量输出。例如，可 PWM 输出可用于电机从静止到全速运行的速度控制，或阀从关闭到全开的位置控制。

15.1.2 组态 PWM 输出

要为 PWM 组态其中一个内置输出，请使用 PWM 向导。



使用以下方法之一打开 PWM 向导：

- 在“工具”(Tools) 菜单的“向导”(Wizards) 区域单击“PWM”按钮。

- 在项目树中打开“向导”(Wizards) 文件夹，然后双击“PWM”，或选择“PWM”并按回车键。

1. 选择脉冲发生器。
2. 必要时，更改 PWM 通道的名称。
3. 组态 PWM 通道输出时基。
4. 生成项目组件。
5. 使用 PWMx_RUN 子例程控制 PWM 输出的占空比。

说明

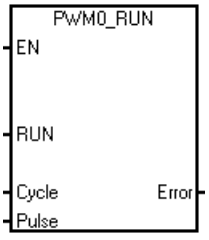
PWM 通道硬编码为具体输出：

- PWM0 已分配到 Q0.0。
- PWM1 已分配到 Q0.1。
- PWM2 已分配到 Q0.3。
- PWM3 已分配到 Q0.2。
- PWM3 已分配到 Q0.4。

15.1.3 PWMx_RUN 子例程

为在应用程序中简化脉宽调制 (PWM) 控制的使用，STEP 7-Micro/WIN SMART V3 提供一个用于组态板载 PWM 发生器和控制 PWM 输出占空比的 PWM 向导 (页 1097)。

PWMx_RUN 子程序用于在程序控制下执行 PWM。

LAD/FBD	STL	描述
	**CALL PWMx_RUN, Cycle, Pulse, Error**	PWMx_RUN 子例程允许您通过改变脉冲宽度（从 0 到周期时间的脉冲宽度）来控制输出占空比。

表格 15-1 PWMx_RUN 子例程的参数

输入/输出	数据类型	操作数
Cycle、Pulse	字	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*AC、*LD、常数
Error	字节	IB、QB、VB、MB、SMB、LB、AC、*VD、*AC、*LD、常数

Cycle 输入是一个字值，定义脉宽调制（PWM）输出的周期。时基为毫秒时，允许范围是 2 到 65535；时基为微秒时是 10 到 65535。

Pulse 输入是一个字值，用于定义 PWM 输出的脉宽（占空比）。允许的取值范围为 0 到 65535 个时基单元，时基是在向导中指定的，单位为微秒或毫秒。

Error 是 PWMx_RUN 子例程返回的字节值，用于指示执行结果。有关可能的错误代码的描述，请参见下表。

表格 15-2 PWMx_RUN 指令错误代码

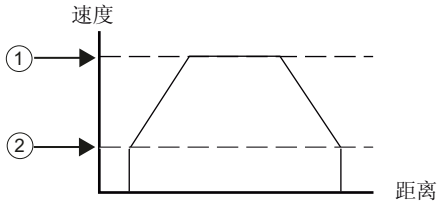
错误代码	描述
0	无错误，正常完成
131	脉冲生成器已被另一个 PWM 或运动轴占用，或者时基更改无效

15.2 使用运动控制

CPU 中内置的运动控制使用运动轴来控制步进电机或伺服电机的速度和运动。

使用运动轴需具备运动控制领域的专业知识。 本章的内容不用于培训。 而是提供基础信息，以帮助您使用运动向导为您的应用组态运动轴。

15.2.1 最大速度和启动/停止速度

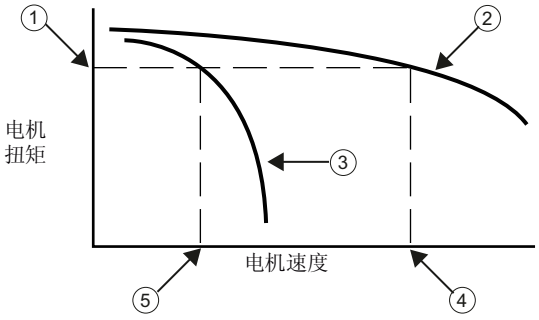
	运动控制向导会提示您应用的最大速度 (MAX_SPEED) 和启动/停止速度 (SS_SPEED)。

① MAX_SPEED

② SS_SPEED

- MAX_SPEED： 在电机力扭矩能力范围内，输入应用中最佳操作速度的数值。 驱动负载所需的扭矩由摩擦力、惯性以及加速/减速时间决定。
- 运动向导基于指定的 MAX_SPEED 计算和显示可由运动轴控制的最低速度。
- SS_SPEED： 在电机能力范围内输入一个数值，以便以较低的速度驱动负载。 如果 SS_SPEED 数值过低，电机和负载在运动的开始和结束时可能会摇摆或颤动。 如果 SS_SPEED 数值过高，电机会在启动时丢失脉冲，并且负载在试图停止时会使电机超速。

在电机数据单中，对于电机和给定负载，有不同的方式定义启动/停止（或拉入/拉出）速度。 通常，SS_SPEED 值是 MAX_SPEED 值的 5% 至 15%。 请参见电机数据单，为您的应用选择正确的速度。 下图显示了典型的电机扭矩/速度曲线。



① 驱动负载所需的扭矩

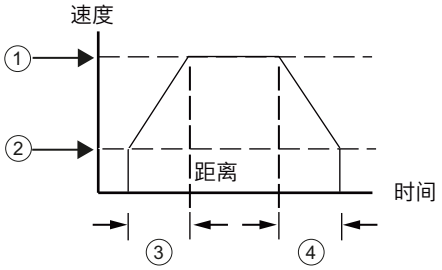
② 电机扭矩与速度特性

③ 启动/停止速度与扭矩： 此曲线随负载惯性增加向较低速度移动。

④ 电机可驱动负载的最大速度： MAX_SPEED 不应超过此值。

⑤ 此负载的启动/停止速度 (SS_SPEED)

15.2.2 输入加速和减速时间



① MAX_SPEED

② SS_SPEED

③ ACCEL_TIME

④ DECEL_TIME

作为组态的一部分，需要设置加速和减速时间。加速时间和减速时间的默认设置均为 1 秒。通常，电机可在不到 1 秒的时间内工作。请参见下图。

说明

电机的加速和减速时间要经过测试来确定。开始时，应输入一个较大的值。然后逐渐减小这个时间值，直到电机开始失速，这样可以优化应用中的这些设置。

以毫秒为单位指定下列时间：

- ACCEL_TIME：电机从 SS_SPEED 加速至 MAX_SPEED 所需要的时间。默认值 = 1000 ms
- DECEL_TIME：电机从 MAX_SPEED 减速至 SS_SPEED 所需要的时间。默认值 = 1000 ms

15.2.3 组态运动包络

曲线是一个预定义的移动描述，它包括一个或多个移动速度，影响着从起点到终点的位置变化。 不需要为了使用运动轴而定义曲线。 您可以使用运动向导提供的指令控制移动，无需运行曲线。

曲线由多个步组成，每一步包含一个达到目标速度的加速/减速过程和以目标速度匀速运行的一串固定数量的脉冲。 如果是单步运动或者是多步运动的最后一步，还应该包括一个由目标速度到停止的减速过程。

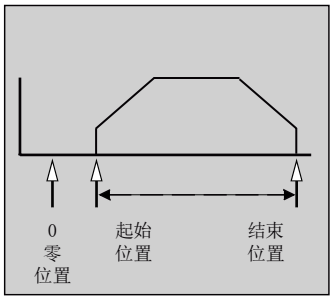
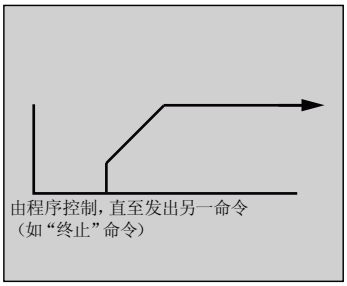
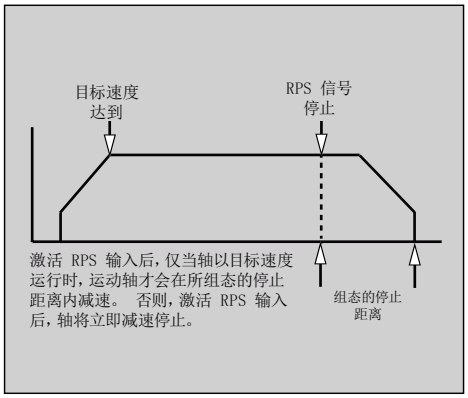
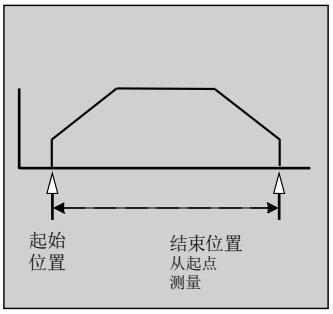
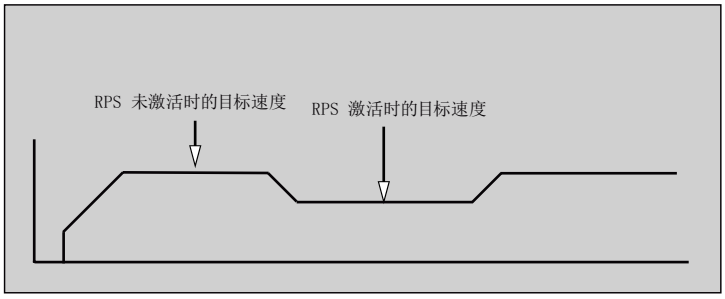
运动轴最多支持 32 个曲线。

定义运动曲线

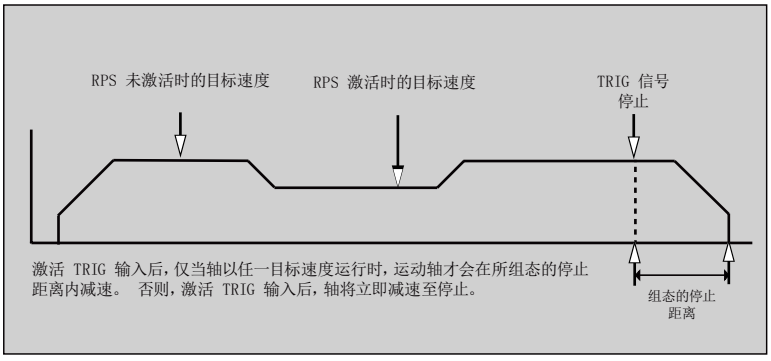
运动向导会指导您完成运动曲线定义，您可以在其中为应用定义每一个运动曲线。 对于每一个曲线，您可以选择操作模式并为曲线的每一步定义指标。 运动向导允许您为每个曲线定义符号名称，您只需在定义曲线时输入符号名称即可。

选择曲线的操作模式

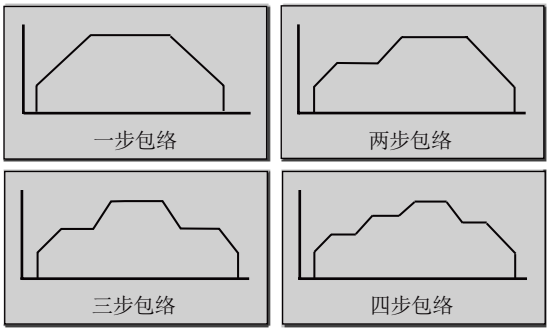
根据所需操作模式组态曲线。 运动轴支持绝对位置、相对位置、单速连续转动以及双速连续转动。 下图显示了不同的操作模式。

<table>
<tr><th colspan="2">运动轴的模式选择</th></tr>
<tr><td>绝对位置</td><td>单速连续旋转</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>单速连续旋转，并带有触发停止功能</td><td>相对位置</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">双速连续旋转</td></tr>
<tr><td colspan="2"> 
</td></tr>
</table>

运动轴的模式选择
双速连续旋转，并带有触发停止功能



创建曲线中的步



步是工具移动的固定距离，包括加速和减速时间内的距离。 运动轴支持每个曲线中最多有 16 个步。

指定每一步的目标速度、结束位置或脉冲数。 每次输入一步。 上图列出了一步、两步、三步和四步曲线。

请注意，一步曲线有一个匀速段，两步曲线有两个匀速段，依此类推。 曲线中的步数与曲线中匀速段的数目一致。

15.3 运动控制的特点

运动控制可在最多四个轴运动中提供开环位置控制所需的功能和性能：

- 提供高速控制，速度从每秒 20 个脉冲到每秒 100,000 个脉冲
- 支持急停（S 曲线）或线性加速及减速
- 提供可组态的测量系统，输入数据时既可以使用工程单位（如英寸或厘米），也可以使用脉冲数
- 提供可组态的换向间隙补偿

- 支持绝对、相对和手动位控方式
- 提供连续操作
- 提供多达 32 个移动曲线，每个曲线最多可有 16 种速度
- 提供四种不同的参考点搜索模式，每种模式都可对起始的寻找方向和最终的接近方向进行选择
- 提供 SINAMICS V90 驱动器的相关支持

使用 STEP 7-Micro/WIN SMART V3 可以创建运动轴所使用的全部组态和曲线信息。这些信息和用户的程序块一起下载到 CPU 中。

运动控制具有六个数字量输入和四个数字量输出，用于连接运动应用。请参见下表。这些输入和输出位于 CPU 上。CPU 技术规范提供了 CPU 的详细信息，并包括将每个 CPU 连接到一些常用的电机驱动器/放大单元的接线图。

表格 15-3　要组态的运动控制 CPU 输入

信号	说明
STP	STP 输入可使 CPU 停止正在进行的运动。在运动向导中可选择所需 STP 操作。
RPS	RPS（参考点切换）输入可为绝对运动操作建立参考点或零点位置。某些模式下，也可通过 RPS 输入使正在进行的运动在行进指定距离后停止。
ZP	ZP（零脉冲）输入可帮助建立参考点或零点位置。通常，电机每转一圈，电机驱动器/放大器就会产生一个 ZP 脉冲。 注：仅在 RP 搜索模式 3 和 4 中使用。
LMT+ LMT-	LMT+ 和 LMT- 输入是运动行程的最大限制。运动向导允许组态 LMT+ 和 LMT- 输入操作。
TRIG	某些模式下，TRIG（触发）输入会触发 CPU，使正在进行的运动在行进指定距离后停止。

说明

LMT+、LMT-、STP 输入信号会打断轴组直线插补运动。

如果已为轴组中的任何轴组态 LMT+/LMT-/STP 输入，CPU 将在接收到 LMT+/LMT-/STP 信号后打断正在进行的轴组直线插补运动。

> ⚠ **警告**
>
> **更改数字量输入通道的滤波时间存在的风险**
>
> 如果数字量输入通道的滤波时间更改自以前的设置，则新的“0”电平输入值可能需要保持长达 12.8 ms 的累积时间，然后滤波器才会完全响应新输入。在此时间段内，持续时间小于 12.8 ms 的短“0”脉冲事件可能无法检测到或无法计数。
>
> 滤波时间的这种更改会引发意外的机械或过程操作，这可能会导致人员死亡、重伤和/或设备损坏。
>
> 对 CPU 重新上电，以确保新的滤波时间立即生效。

表格 15-4　运动控制 CPU 硬编码输出

信号	说明
P0 P1	P0 和 P1 为脉冲输出，用于控制电机的运动和方向。
DIS	DIS 输出用于禁用或启用电机驱动器/放大器。

15.4 使用运动轴

15.4.1 编程运动轴

STEP 7-Micro/WIN SMART V3 提供了简单易用的工具来组态和编程运动轴。只需遵循以下步骤即可：

1. 组态运动轴：STEP 7-Micro/WIN SMART V3 提供了运动向导来创建组态/曲线表和位置指令。有关组态运动轴的信息，请参见“组态运动轴”。
2. 测试运动轴的操作：STEP 7-Micro/WIN SMART V3 提供了运动控制面板来测试输入和输出的接线、运动轴的组态以及运动曲线的操作。有关运动面板的信息，请参见“使用运动控制面板监视运动轴”。

3. 创建由 CPU 执行的程序：运动向导会自动创建运动指令，您可以将这些指令插入程序中。有关运动指令的信息，请参见“运动向导为运动轴创建的指令”。将下列指令插入程序中：
   - 要启用运动轴，插入 AXISx_CTRL 指令。使用 SM0.0（始终接通）可确保每次扫描时都会执行这条指令。
   - 要将电机移动到特定位置，请使用 AXISx_GOTO 或 AXISx_RUN 指令。AXISx_GOTO 指令会使电机运动到您在程序输入中指定的位置。AXISx_RUN 指令会使电机按照您在运动向导中所组态的曲线运动。
   - 要为运动使用绝对坐标，必须为应用建立零位置。使用 AXISx_RSEEK 或 AXISx_LDPOS 指令可以建立零位置。
   - 运动向导创建的其它指令可为典型应用提供需要的功能，对于您的特定应用，这些指令是可选的。
4. 编译程序并将系统块、数据块和程序块下载到 CPU。

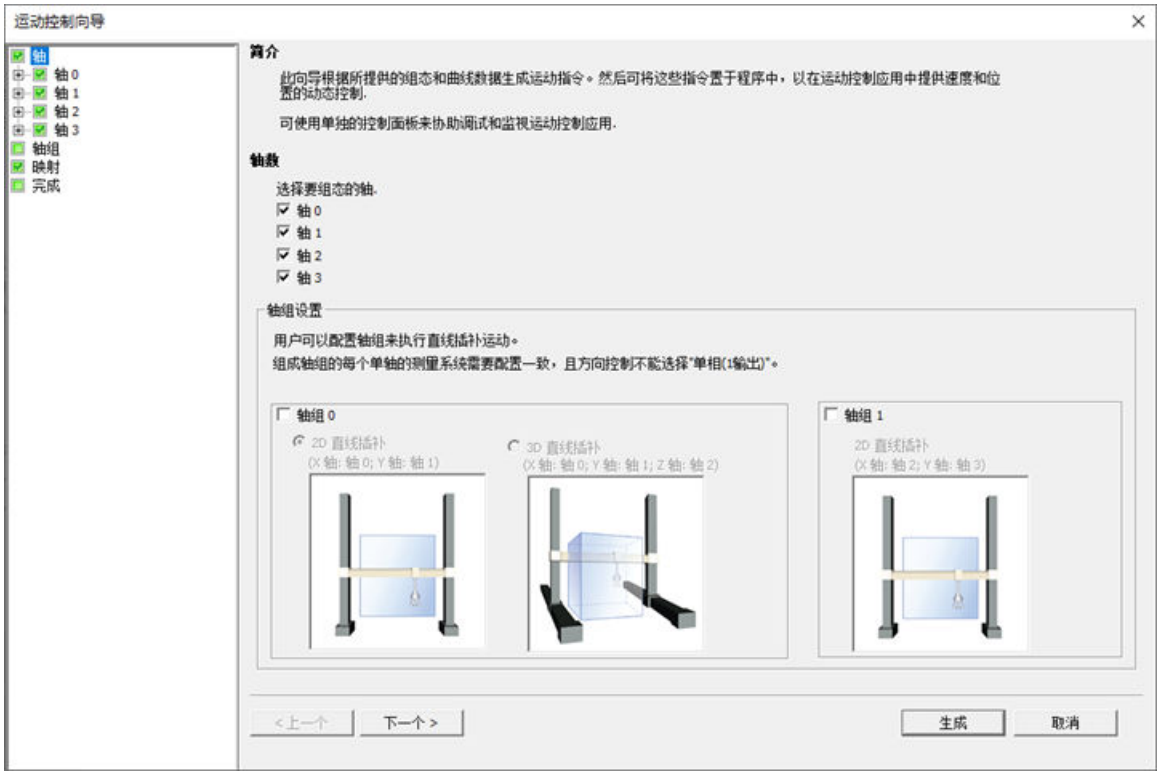
说明

确保测量系统的组态符合步进/伺服电机控制器系统的脉冲/转和距离/转的相关规范。

15.4.2 组态运动轴

组态/曲线表

为了使 CPU 能够控制运动应用，必须为运动轴创建组态/曲线表。基于运动向导中的逐步操作指导，组态过程更为轻松便捷。有关组态/曲线表的详细信息，请参见本章节的“高级主题”部分。



在运动向导中，可离线创建组态/曲线表。组态创建无需连接 CPU。

启动运动向导

要启动运动向导，请单击导航栏中的“工具”(Tools) 图标，然后双击“运动向导”(Motion Wizard) 图标；或者选择“工具 > 运动向导”(Tools> Motion Wizard) 菜单命令。

选择测量类型

选择测量系统：可以选择工程单位或脉冲：

- 如果选择脉冲，则不需要其它信息。
- 如果选择工程单位，则需要输入电机每转一圈产生的脉冲数（请参见电机或驱动器数据表）、测量基本单位（如英寸、英尺、毫米或厘米）以及电机转动一圈移动的距离。

如果之后对测量系统进行了更改，则需删除整个组态以及运动向导生成的所有指令。之后必须输入与新测量系统一致的选项。

组态输入引脚位置

可通过 SDB0 中的组态对与运动控制相关的输入进行编程，其中包括 STP、LMT-、LMT+、RPS、TRIG 和 ZP。

表格 15-5　STP、RPS、LMT+、LMT-、TRIG 和 ZP 的引脚存储单元

输入的引脚定义	说明
LMT+、LMT-、STP、RPS、TRIG	CPU 的输入引脚 0 可作为 LMT+、LMT-、STP、RPS、TRIG 输入 (I0.0)。
	CPU 的输入引脚 1 可作为 LMT+、LMT-、STP、RPS、TRIG 输入 (I0.1)。
	CPU 的输入引脚 2 可作为 LMT+、LMT-、STP、RPS、TRIG 输入 (I0.2)。
	CPU 的输入引脚 3 可作为 LMT+、LMT-、STP、RPS、TRIG 输入 (I0.3)。
	CPU 的输入引脚 4 可作为 LMT+、LMT-、STP、RPS、TRIG 输入 (I0.4)。
	CPU 的输入引脚 5 可作为 LMT+、LMT-、STP、RPS、TRIG 输入 (I0.5)。
	CPU 的输入引脚 6 可作为 LMT+、LMT-、STP、RPS、TRIG 输入 (I0.6)。
	CPU 的输入引脚 7 可作为 LMT+、LMT-、STP、RPS、TRIG 输入 (I0.7)。
	CPU 的输入引脚 8 可作为 LMT+、LMT-、STP、RPS、TRIG 输入 (I1.0)。
	CPU 的输入引脚 9 可作为 LMT+、LMT-、STP、RPS、TRIG 输入 (I1.1)。
	CPU 的输入引脚 10 可作为 LMT+、LMT-、STP、RPS、TRIG 输入 (I1.2)。
	CPU 的输入引脚 11 可作为 LMT+、LMT-、STP、RPS、TRIG 输入 (I1.3)。
ZP	ZP 输入 (I0.0)、(I0.1)、(I0.2) 和 (I0.3)。

说明

将某个输入组态给特定运动轴的指定功能（例如 RPS）后，不得将该输入用于任何其它的运动轴或用于任何其它的输入、计时器或中断功能。

说明

高速输入接线必须使用屏蔽电缆

连接 HSC 输入通道 I0.0、I0.1、I0.2 和 I0.3 时，所使用屏蔽电缆的长度不应超过 50 m。

映射 I/O

STEP 7-Micro/WIN SMART V3 为 PWM 和运动轴进行了固定输出分配。

P0 和 P1 输出

至少为已启用的任何轴组态 P0 输出引脚。如果“相”(Phase) 组态不是“单相（1 个输出）”，则还可能有 P1 输出。有关详细信息，请参见“编辑默认输入和输出组态”部分。根据以下标准，针对特定输出对这些输出引脚进行硬编码：

<table>
<tr><th>输出</th><th colspan="2">轴 0</th><th colspan="2">轴 1</th><th colspan="2">轴 2</th><th>轴 3</th></tr>
<tr><td>P0</td><td colspan="2">Q0.0</td><td colspan="2">Q0.1</td><td colspan="2">Q0.3</td><td>Q0.2</td></tr>
<tr><td rowspan="3">P1[1]</td><td rowspan="2">Q0.2</td><td>轴 0 组态为“双相（2 输出）”(Two-phase (2 output)) 或“AB 正交相位（2 输出）”(AB quadrature phase (2 output))。</td><td rowspan="2">Q0.3</td><td rowspan="2">轴 1 组态为“双相（2 输出）”(Two-phase (2 output)) 或“AB 正交相位（2 输出）”(AB quadrature phase (2 output))。</td><td colspan="2" rowspan="3">Q1.0：轴 2 组态为“单相（2 输出）”(Single phase (2 output))。
注：
如果轴 1 的“相”(Phase) 组态为“双相（2 输出）”(Two-phase (2 output)) 或“AB 正交相位（2 输出）”(AB quadrature phase (2 output))，则轴 2 不可用。</td><td rowspan="3">Q1.1：轴 3 组态为“单相（2 输出）”(Single phase (2 output))。
注：
如果轴 0 的“相”(Phase) 组态为“双相（2 输出）”(Two-phase (3 output)) 或“AB 正交相位（2 输出）”(AB quadrature phase (2 output))，则轴 3 不可用。</td></tr>
<tr><td>轴 0 组态为“单相（2 输出）”(Single-phase (2 output))，未组态轴 3</td></tr>
<tr><td>Q0.6</td><td>轴 0 组态为“单相（2 输出）”(Single-phase (2 output))，组态轴 3</td><td>Q0.7</td><td>轴 1 组态为“单相（2 输出）”(Single-phase (2 output))</td></tr>
<tr><td>DIS</td><td colspan="2">Q0.4</td><td colspan="2">Q0.5</td><td>Q0.6</td><td>未组态轴 3。</td><td rowspan="2">Q1.3</td></tr>
<tr><td colspan="2"></td><td colspan="2"></td><td>Q1.2</td><td>组态轴 3。</td></tr>
</table>

[1] 如果任意轴的“相”(Phase) 组态为“单相（1 个输出）”(Single-phase (1 output))，则不会分配 P1 输出。

DIS 输出

如果已为轴组态 DIS 输出，则在映射表中存在该输出的条目。DIS 输出也硬编码为具体输出，具体规则如下：

如果未组态轴 3：

- 轴 0：Q0.4
- 轴 1：Q0.5
- 轴 2：Q0.6

如果组态轴 3：

- 轴 0：Q0.4
- 轴 1：Q0.5
- 轴 2：Q1.2
- 轴 3：Q1.3

脉冲输出单元连接到标准 24V 输出。

编辑默认的输入和输出组态

要更改或查看集成输入/输出的默认组态，请选择所需输入/输出节点：

- 在“有效电平”(Active Levels) 字段中，使用下拉列表选择有效电平（高或低）。电平设为“高”(High) 时，当输入有电流时，将读取逻辑 1。如果电平设为“低”(Low)，则会在输入中没有电流时读取逻辑 1。逻辑 1 电平总是解释为有效条件。不管激活电平如何，输入中流入电流时，LED 就会点亮。（默认 = 高电平有效）
- 在“系统块”(System Block) 的“数字量输入”(Digital Inputs) 节点中，可选择 STP、RPS、LMT+ 、LMT- 和 TRIG 输入的滤波时间常数（0.20 ms 到 12.80 ms）。增加滤波器时间常数可消除更多噪声，但会降低对信号状态变化的响应时间。（默认值 = 6.4 ms）

  ⚠ 警告

  更改数字量输入通道的滤波时间存在的风险

  如果数字量输入通道的滤波时间更改自以前的设置，则新的“0”电平输入值可能需要保持长达 12.8 ms 的累积时间，然后滤波器才会完全响应新输入。在此时间段内，持续时间小于 12.8 ms 的短“0”脉冲事件可能无法检测到或无法计数。

  滤波时间的这种更改会引发意外的机械或过程操作，这可能会导致人员死亡、重伤和/或设备损坏。

  为了确保新的滤波时间立即生效，必须关闭 CPU 电源后再开启。

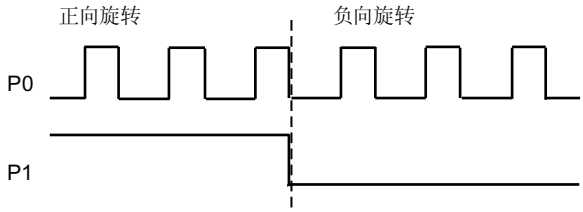
- 在“方向控制”(Directional Control) 节点，可选择下列“相位”(Phasing) 模式：
  - 单相（2 个输出）
  - 双相（2 个输出）
  - AB 正交相（2 个输出）
  - 单相（1 个输出）

  还可以选择输出的“极性”(Polarity)（正或负）。

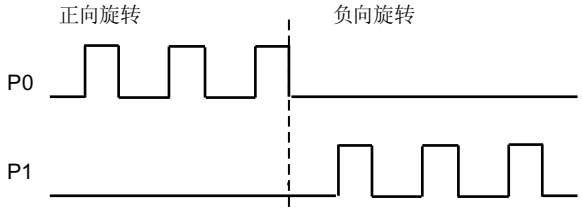
相位

步进电机/伺服驱动器的“相位”(Phasing) 接口有四个选项。这些选项如下：

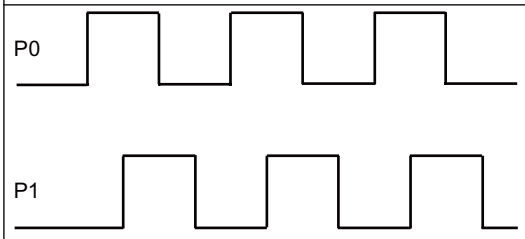
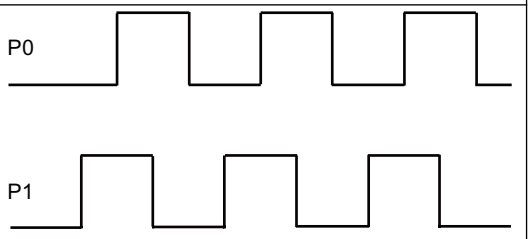
- 单相（2 个输出）：如果选择单相（2 个输出）选项，则一个输出 (P0) 控制脉冲，另一输出 (P1) 控制方向。如果脉冲处于正向，则 P1 为高（有效）。如果脉冲处于负向，则 P1 为低（无效）。单相（2 个输出）如下图所示（假设极性为正）：



- 双相（2 个输出）：如果选择双相（2 个输出）选项，则一个输出 (P0) 脉冲控制正方向，另一个输出脉冲 (P1) 控制负方向。双相（2 个输出）如下图所示（假设极性为正）：



- AB 正交相（2 个输出）：如果选择 AB 正交相（2 个输出）选项，则两个输出均以指定速度产生脉冲，但相位相差 90 度。AB 正交相（2 个输出）是一种 1X 组态，即会从输出的一个正跳变到下一个正跳变对生成的脉冲进行测量。在这种情况下，方向由首先跳变到高的输出确定。针对正向 P0 领先 P1。针对负向 P1 领先 P0。AB 正交相（2 个输出）如下图所示（假设极性为正）：

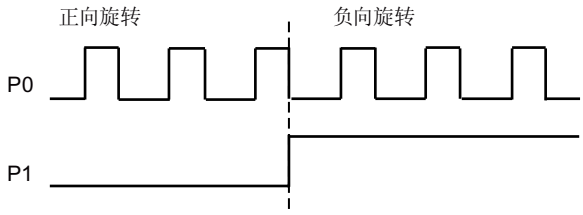
AB 正交相（2 个输出）	
（正极性）：正向旋转	（正极性）：反向旋转
	
P0 超前 P1	P1 超前 P0

- 单相（1 个输出）：如果选择单相（1 个输出）选项，则输出 (P0) 控制脉冲。此模式下，CPU 只接受正运动命令。选择此模式时，运动控制向导将限制进行非法负向组态。如果运动应用仅以一个方向进行，则可保存输出。单相（1 个输出）如下图所示（假设极性为正）：

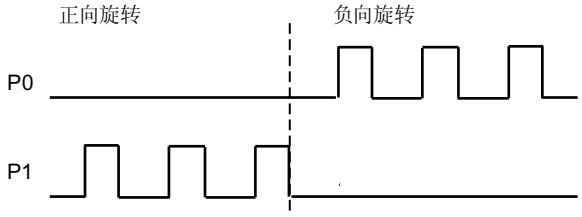
极性

可使用“极性”(Polarity) 参数切换正向和负向。如果电机的接线方向错误，则通常会进行此操作。此时，可以通过将此参数设置为负，避免对硬件进行重新接线。负设置如下更改输出操作：

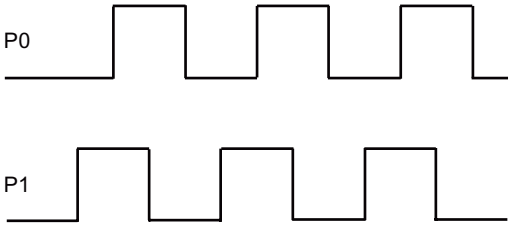
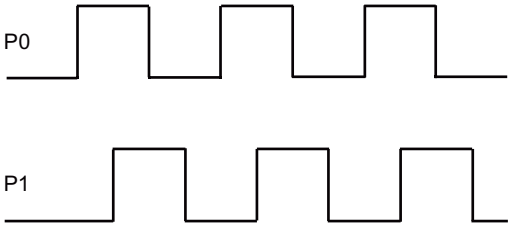
- 单相（2 个输出）：如果脉冲处于正向，则 P1 为低（无效）。如果脉冲处于负向，则 P1 为高（有效）。如下图所示：



- 双相（2 个输出）：P0 脉冲针对负向。P1 脉冲针对正向。如下图所示：



- AB 正交相（2 个输出）：针对负向 P0 领先 P1。针对正向 P1 领先 P0。如下图所示：

AB 正交相（2 个输出）	
（负极性）：正向旋转	（负极性）：反向旋转
	
P1 超前 P0	P0 超前 P1

- 单相（1 个输出）：在此相位模式下不允许负极性。

“方向控制”(Directional Control) 对话框的默认设置为“单相（2 个输出）”(Single phase (2 output)) 和“正极性”(Positive polarity)。

说明

无法选择 P0 和 P1 组态到哪个引脚；这被硬编码到特定引脚。有关引脚映射列表，请参见“映射 I/O”部分（本节前面）。

⚠ 警告

使用“运动轴”时的安全预防措施

轴控制中的限位和停止功能是通过电逻辑实现的，不能提供机电控制所能提供的保护等级。

控制设备和“运动轴”功能在不安全情况下可能会出现故障，进而导致受控设备的意外操作。此类意外操作可能会导致严重人身伤害，甚至死亡和 / 或财产损失。

请考虑使用独立于运动轴和 CPU 的急停功能、机电超控功能或冗余机电保护功能。

组态对物理输入的响应

1. 选择对 LMT+、LMT- 和 STP 输入的响应。
2. 使用下拉列表选择：减速至停止（默认设置）或立即停止。

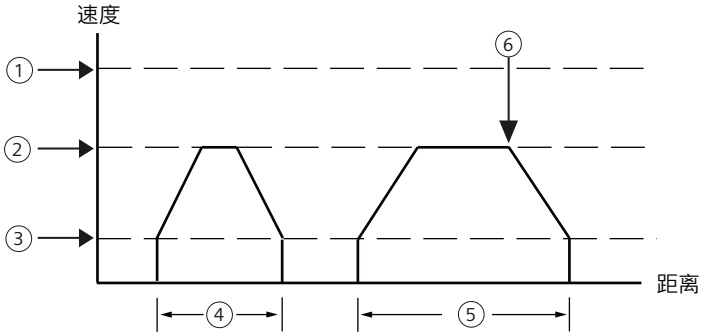
输入最大启动和停止速度

输入应用的最大速度 (MAX_SPEED) 和启动/停止速度 (SS_SPEED)。

输入点动参数

输入 JOG_SPEED 和 JOG_INCREMENT 值：

- JOG_SPEED：JOG_SPEED（电机的点动速度）是 JOG 命令仍然有效时所能实现的最大速度。
- JOG_INCREMENT：瞬时 JOG 命令移动工具的距离。
  下图显示了点动命令的操作。运动轴接收到点动命令后，将启动定时器。如果点动命令在 0.5 秒钟时间内结束，则运动轴将以 JOG_SPEED 中指定的速度将工件移动由 JOG_INCREMENT 指定的一段距离。如果 0.5 秒中后点动命令仍在运行，则运动轴将加速到 JOG_SPEED 速度，运动将继续，直至点动命令终止。运动轴随后减速至停止。点动命令可在运动控制面板中启用，也可通过运动指令启用。下图显示了 JOG 操作。



① MAX_SPEED
② JOG_SPEED
③ SS_SPEED
④ JOG_INCREMENT：JOG 命令处于激活状态不超过 0.5 秒。
⑤ JOG 命令处于激活状态超过 0.5 秒。
⑥ JOG 命令终止（开始从 JOG_SPEED 降至 SS_SPEED）。

输入加速和减速时间

在编辑框中输入加速和减速时间。

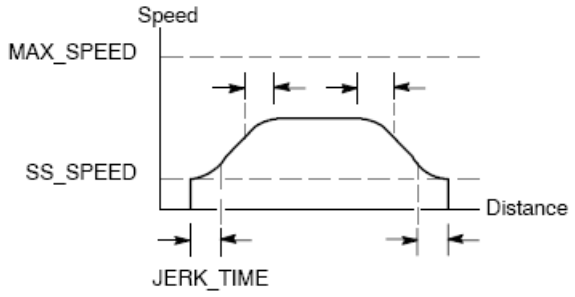
说明

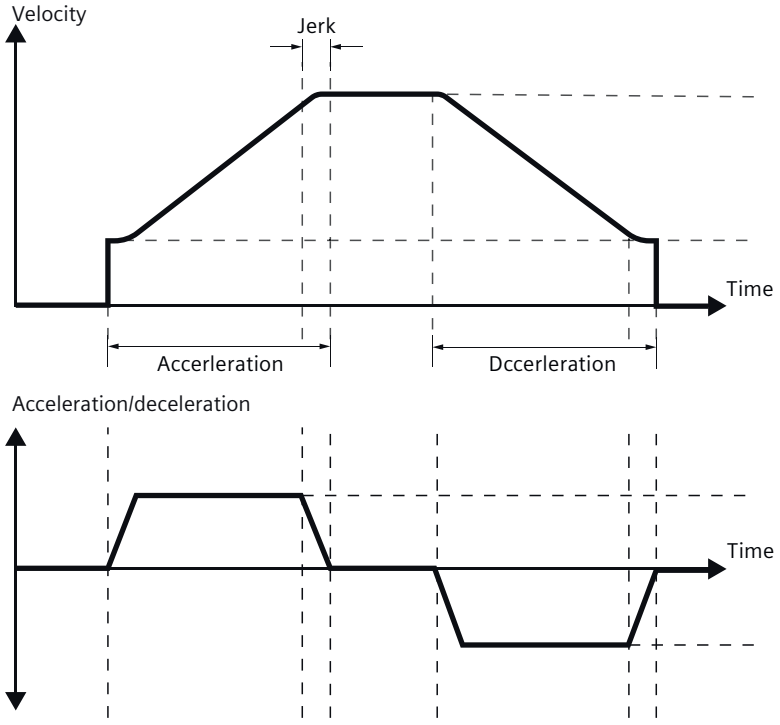
运动轴组

如果轴组中的所有轴均已组态了加速/减速时间，则该组态仅对运动轴生效，而对运动轴组无效。轴组直线插补运动的加速/减速时间可在 GRP0_2/3D_MOVELINEAR (页 1187) 指令的输入参数中，或运动向导中的轴组“线段规划 (页 1180)”(segment planning) 中组态。

输入急停时间

急停补偿对于某些移动类型可用，通过减少运动包络的加速和减速部分中的急停（速率变化），实现较平稳的位置控制。参见下图：





急停补偿又称作“S 曲线成型”。该补偿等效应用于加速和减速曲线的开始和结束部分。对于初始步和最终步期间的零速与 SS_SPEED 之间，则不会应用急停补偿。

可输入时间值 (JERK_TIME) 指定急停补偿。这是加速度从零变化为最大加速率所需的时间。急停时间越长，则运行越平稳，与减少 ACCEL_TIME 和 DECEL_TIME 的方式相比，总周期时间会略微增大。值为 0 ms（默认值）时，表示未应用任何补偿。

说明

JERK_TIME 值的最佳设置为 ACCEL_TIME 的 40%。

说明

急停补偿不可用于双速移动、手动更改速度移动、中止的移动和达到限制或 STP 输入时自动做出的减速反应。

说明

运动轴组

如果轴组中的所有轴均已组态了急停时间，则该组态仅对运动轴生效，而对运动轴组无效。轴组直线插补运动的急停时间可在 GRP0_2/3D_MOVELINEAR (页 1187) 指令的输入参数中，或运动向导中的轴组“线段规划 (页 1180)”(segment planning) 中组态。

组态反冲补偿

反冲补偿：在方向发生变化时，为消除系统中的反冲（滞慢），电机必须移动的距离。反冲补偿始终为正值：

- 默认值 = 0
- 选择参考点搜索序列来使用反冲。

说明

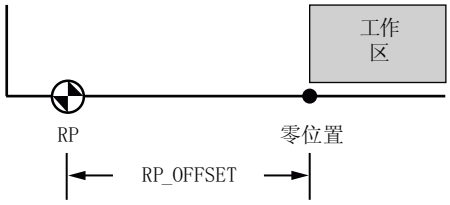
运动轴组

轴组中各个轴的换向间隙补偿组态不适用于进行轴组直线插补运动的电机。

组态参考点和搜索参数

1. 应用中选择使用参考点或不使用参考点：
   – 如果应用中要求从一个绝对位置处开始运动或以绝对位置作为参考，则需建立一个参考点 (RP) 或零点位置，该点可将位置测量固定到物理系统的一个已知点上。
   – 如果使用参考点，则需定义参考点自动重新定位的方式。自动定位参考点的过程称为“参考点搜索”。在向导中定义“参考点搜索”过程需要两步。
2. 输入参考点搜索速度（快速搜索速度和慢速搜索速度）：
   – **RP_FAST** 是模块执行 RP 搜索命令时使用的初始速度。通常，RP_FAST 值约为 MAX_SPEED 值的 2/3。
   – **RP_SLOW** 是接近 RP 的最终速度。接近 RP 时，会使用一个较慢的速度，以免错过。通常，RP_SLOW 的值为 SS_SPEED 值。

3. 定义初始搜索方向和最终参考点接近方向：
   - **RP_SEEK_DIR** 是 RP 搜索操作的初始方向。通常，这个方向是从工作区到 RP 附近。限位开关在确定 RP 的搜索区域时起着至关重要的作用。当执行 RP 搜索操作时，遇到限位开关会引起方向反转，使搜索能够继续下去。（默认方向 = 反向）
   - **RP_APPR_DIR** 是最终接近 RP 的方向。为了减小反冲并提供更高的精度，应该按照从 RP 移动到工作区所使用的方向来接近参考点。（默认方向 = 正向）
4. 运动向导提供高级参考点选项，可以指定 RP 偏移量 (RP_OFFSET)，即从 RP 到零点位置的距离。请参见下图：
   - **RP_OFFSET**：从 RP 到物理测量系统零点位置的距离。
   - 默认值 = 0



5. 运动轴提供了一个参考点开关 (RPS) 输入，在搜索 RP 的过程中使用。以 RPS 为参考确定一个准确的位置作为 RP。可以把 RPS 有效区域的中点或边沿作为 RP，也可以选择从 RPS 有效区域边沿开始，经过一定数量的 Z 脉冲 (ZP) 的位置作为 RP。
6. 用户可组态运动轴搜索参考点的顺序。下图显示了一个 RP 默认搜索顺序的简图。RP 搜索顺序可选择以下某种模式：
   - **RP 搜索模式 0**：不执行 RP 搜索序列
   - **RP 搜索模式 1**：这种模式将 RP 定位在靠近工作区一侧的 RPS 输入开始激活的位置。（默认）
   - **RP 搜索模式 2**：RP 在 RPS 输入有效区内居中。
   - **RP 搜索模式 3**：RP 位于 RPS 输入的有效区之外。RP_Z_CNT 指定 RPS 失效后应接收的 ZP（零脉冲）输入的数目。
   - **RP 搜索模式 4**：RP 通常位于 RPS 输入的有效区内。RP_Z_CNT 指定 RPS 激活后应接收的 ZP（零脉冲）输入的数目。

RP 搜索模式 1

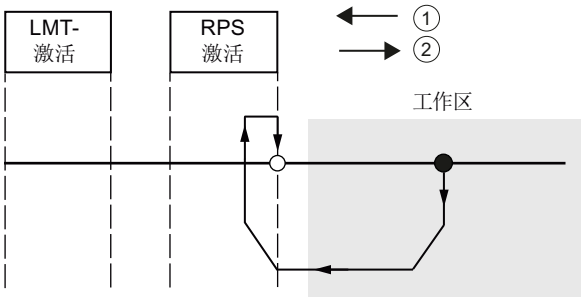


图 15-1　①：RP 搜索方向②：RP 接近方向

说明

RPS 有效区（RPS 输入保持激活的距离）必须大于从 RP_FAST 减速为 RP_SLOW 所需的距离。如果距离过短，运动轴将生成一个错误。

定义运动曲线

1. 在运动曲线定义画面中，单击新曲线按钮可启用定义曲线。
2. 选择需要的操作模式：
   - 对于绝对位置曲线：
     填写目标速度和结束位置。
     如果需要多步，请单击“新建步”(New Step) 按钮并根据需要输入步信息。
   - 对于相对位置曲线：
     填写目标速度和结束位置。
     如果需要多步，请单击“新建步”(New Step) 按钮并根据需要输入步信息。
   - 对于单速连续转动：
     在编辑框中输入目标速度值。
     选择旋转方向。
     如果希望使用 RPS 输入结束单速连续转动，请单击该复选框。
     填写 RPS 输入激活后要移动的距离（必须使能 RPS 输入）。
   - 对于双速连续旋转（必须使能 RPS 输入）：
     在编辑框中输入 RPS 未激活时的目标速度值。
     在编辑框中输入 RPS 激活时的目标速度值。
     选择旋转方向。
     若要使用 TRIG 输入结束双速连续转动，请单击该复选框。（必须使能 TRIG 输入）
     填写激活 TRIG 输入后要移动的距离。
3. 定义执行指定运动所需的任意多个曲线和步。

完成组态

1. 完成运动轴操作组态后，只需单击“生成”(Generate) 即可。
   运动向导会执行以下任务：
   - 将轴组态和曲线表插入 CPU 程序的系统块和数据块
   - 为运动参数创建一个全局符号表
   - 将运动指令子例程添加到项目程序块，之后即可在应用中使用这些指令
2. 要修改任何组态或曲线信息，可再次运行运动向导。

说明

由于运动向导会对程序块、数据块和系统块进行更改，因此必须将这三种块都下载到 CPU 中。否则，运动轴将无法得到正常操作所需的所有程序组件。

15.4.3 运动向导为运动轴创建的子程序

除了每次扫描时都必须激活的 AXISx_CTRL 外，每个动作都必须确保一次只有一个运动控制子例程处于激活状态。 每个运动子例程都有“AXISx_”前缀，其中“x”代表轴通道编号。共有 13 个运动控制子例程。

运动控制子例程	说明
AXISx_CTRL (页 1123)	提供轴的初始化和全面控制
AXISx_MAN (页 1124)	用于轴的手动模式操作
AXISx_GOTO (页 1126)	命令轴转到指定位置
AXISx_RUN (页 1128)	命令轴执行已组态的运动曲线
AXISx_RSEEK (页 1129)	启动参考点查找操作
AXISx_LDOFF (页 1130)	建立一个偏移参考点位置的新零点位置
AXISx_LDPOS (页 1131)	将轴位置更改为新值
AXISx_SRATE (页 1132)	修改已组态的加速、减速和急停补偿时间
AXISx_DIS (页 1133)	控制 DIS 输出
AXISx_CFG (页 1134)	根据需要读取组态块并更新轴设置
AXISx_CACHE (页 1135)	预先缓冲已组态的运动曲线
AXISx_RDPOS (页 1136)	返回当前轴位置
AXISx_ABSPOS (页 1137)	通过 SINAMICS V90 伺服驱动器读取绝对位置值

说明

运动控制子例程使程序所需的存储空间增加多达 1700 个字节。 可以删除未使用的运动控制子例程来降低所需的存储空间。 为防止生成不需要的运动控制子例程，请为运动控制向导的“组件”(Components) 节点中不需要的子例程，取消选中“生成”(Generate) 框。 要恢复生成特定的运动控制子例程，请再次启动运动控制向导，导航到“组件”(Components) 节点，然后为该子例程选中“生成”(Generate) 框。 单击“生成”(Generate) 按钮以重建由向导生成的子例程。

另请参见

使用运动控制向导

15.4.3.1 运动控制子例程使用准则

必须确保在同一时间仅有一条运动控制子例程激活。

只要循环调用中断，便可在中断例程中执行 AXISx_RUN 和 AXISx_GOTO。但是，如果运动轴正在处理另一个命令，则切勿尝试在中断例程中启动运动控制子例程。如果在中断例程中启动子例程，则可使用 AXISx_CTRL 子例程的输出来监视运动轴是否完成移动。

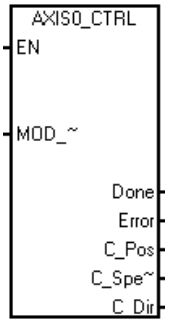
运动控制向导根据所选的度量系统自动组态速度参数（Speed 和 C_Speed）和位置参数（Pos 或 C_Pos）的值。对于脉冲，这些参数为 DINT 值。对于工程单位，这些参数是所选单位类型对应的 REAL 值。例如：如果选择厘米 (cm)，则以厘米为单位将位置参数存储为 REAL 值并以厘米/秒 (cm/sec) 为单位将速度参数存储为 REAL 值。

使用运动控制子例程时的一些“生成”准则如下：

- 要在每次扫描时执行子例程，请在程序中插入 AXISx_CTRL 子例程并使用 SM0.0 触点。
- 要指定运动到绝对位置，必须首先使用 AXISx_RSEEK 或 AXISx_LDPOS 子例程建立零位置。
- 要根据程序输入移动到特定位置，请使用 AXISx_GOTO 子例程。
- 要运行通过运动控制向导组态的运动曲线，请使用 AXISx_RUN 子例程。

15.4.3.2 AXISx_CTRL 子例程

表格 15-6 AXISx_CTRL

LAD/FBD	STL	说明
	**CALL AXISx_CTRL, MOD_EN, Done, Error, C_Pos, C_Speed, C_Dir**	AXISx_CTRL 子例程（控制）启用和初始化运动轴，方法是自动命令运动轴每次 CPU 更改为 RUN 模式时加载组态/曲线表。 项目中每条运动轴仅使用该子例程一次，并应确保程序在每次扫描时调用此子例程。使用 SM0.0（始终开启）作为 EN 参数的输入。 该子例程也用于初始化轴组中的运动轴。

表格 15-7 AXISx_CTRL 子例程的参数

输入/输出	数据类型	操作数
MOD_EN	BOOL	I、Q、V、M、SM、S、T、C、L、能流
Done、C_Dir	BOOL	I、Q、V、M、SM、S、T、C、L
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD
C_Pos、C_Speed	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD

MOD_EN 参数必须开启，才能启用其它运动控制子例程向运动轴发送命令。如果 MOD_EN 参数关闭，则运动轴将中止进行中的任何指令并执行减速停止。

说明

中止轴组直线插补运动

要中止轴组直线插补运动，可关闭任何 AXISx_CTRL 子例程的 MOD_EN 参数。轴组将沿规划的运动路径减速停止。

AXISx_CTRL 子例程的输出参数提供运动轴的当前状态。

当运动轴完成任何一个子例程时，Done 参数会开启。

Error 参数 (页 1161)包含该子例程的结果。

C_Pos 参数表示运动轴的当前位置。根据测量单位，该值是脉冲数 (DINT) 或工程单位数 (REAL)。

C_Speed 参数提供运动轴的当前速度。如果组态脉冲的运动轴测量系统，则 C_Speed 为一个 DINT 数值，具体为每秒钟的脉冲数目。如果组态工程组态单元的测量系统，C_Speed 是一个 REAL 数值，具体为每秒钟的所选工程组态单元数目 (REAL)。

C_Dir 参数表示电机的当前方向：

- 信号状态 0 = 正向
- 信号状态 1 = 反向

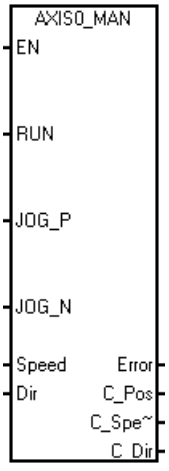
说明

运动轴仅在电源开启或接到指令加载组态时读取组态/曲线表。

- 使用运动控制向导修改组态时，AXISx_CTRL 子例程会自动命令运动轴在每次 CPU 更改为 RUN 模式时加载组态/曲线表。
- 使用运动控制面板修改组态时，单击“更新组态”(Update Configuration) 按钮，命令运动轴将加载新组态/曲线表。
- 使用其它方法修改组态时，还需向运动轴发出一个 AXISx_CFG 命令，加载组态/曲线表。否则，运动轴会继续使用旧组态/曲线表。

15.4.3.3 AXISx_MAN 子例程

表格 15-8 AXISx_MAN

LAD/FBD	STL	说明
	**CALL AXISx_MAN, RUN, JOG_P, JOG_N, Speed, Dir, Error, C_Pos, C_Speed, C_Dir**	AXISx_MAN 子例程（手动模式）将运动轴置为手动模式。 这允许电机按不同的速度运行，或沿正向或负向慢进。 您在同一时间仅能启用 RUN、JOG_P 或 JOG_N 输入之一。

表格 15-9　AXISx_MAN 子例程的参数

输入/输出	数据类型	操作数
RUN、JOG_P、JOG_N	BOOL	I、Q、V、M、SM、S、T、C、L、能流
Speed	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD、常数
Dir、C_Dir	BOOL	I、Q、V、M、SM、S、T、C、L
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD
C_Pos、C_Speed	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD

启用 RUN（运行/停止）参数会命令运动轴加速至指定的速度（Speed 参数）和方向（Dir 参数）。 您可以在电机运行时更改 Speed 参数，但 Dir 参数必须保持为常数。 禁用 RUN 参数会命令运动轴减速，直至电机停止。

启用 JOG_P（点动正向旋转）或 JOG_N（点动反向旋转）参数会命令运动轴正向或反向点动。 如果 JOG_P 或 JOG_N 参数保持启用的时间短于 0.5 秒，则运动轴将通过脉冲指示移动 JOG_INCREMENT 中指定的距离。 如果 JOG_P 或 JOG_N 参数保持启用的时间为 0.5 秒或更长，则运动轴将开始加速至指定的 JOG_SPEED。

Speed 参数决定启用 RUN 时的速度。 如果您针对脉冲组态运动轴的测量系统，则速度为 DINT 值（脉冲数/每秒）。 如果您针对工程单位组态运动轴的测量系统，则速度为 REAL 值（单位数/每秒）。 您可以在电机运行时更改该参数。

说明

运动轴可能不会对 Speed 参数的小幅更改做出响应，尤其是在组态的加速或减速时间非常短且组态的最大速度与启动/停止速度之间的差值较大时。

Dir 参数确定当 RUN 启用时移动的方向。 您可以在 RUN 参数启用时更改该数值。

Error 参数 (页 1161)包含该子例程的结果。

C_Pos 参数包含运动轴的当前位置。 根据所选的测量单位，该值是脉冲数 (DINT) 或工程单位数 (REAL)。

C_Speed 参数包含运动轴的当前速度。 根据所选的测量单位，该值是脉冲数/每秒 (DINT) 或工程单位数/每秒 (REAL)。

C_Dir 参数表示电机的当前方向：

- 信号状态 0 = 正向
- 信号状态 1 = 反向

15.4.3.4 AXISx_GOTO 子例程

表格 15-10 AXISx_GOTO

LAD/FBD	STL	说明
	CALL AXISx_GOTO, START, Pos, Speed, Mode, Abort, Done, Error, C_Pos, C_Speed	AXISx_GOTO 子例程命令运动轴转到所需位置。

表格 15-11 AXISx_GOTO 子例程的参数

输入/输出	数据类型	操作数
START	BOOL	I、Q、V、M、SM、S、T、C、L、能流
Pos、Speed	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD、常数
Mode	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD、常数
Abort、Done	BOOL	I、Q、V、M、SM、S、T、C、L
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD
C_Pos、C_Speed	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD

开启 EN 位会启用此子例程。确保 EN 位保持开启，直至 DONE 位指示子例程执行已经完成。

开启 START 参数会向运动轴发出 GOTO 命令。对于在 START 参数开启且运动轴当前不繁忙时执行的每次扫描，该子例程向运动轴发送一个 GOTO 命令。为了确保仅发送了一个 GOTO 命令，请使用边沿检测元素用脉冲方式开启 START 参数。

Pos 参数包含一个数值，指示要移动的位置（绝对移动）或要移动的距离（相对移动）。根据所选的测量单位，该值是脉冲数 (DINT) 或工程单位数 (REAL)。

Speed 参数确定该移动的最高速度。根据所选的测量单位，该值是脉冲数/每秒 (DINT) 或工程单位数/每秒 (REAL)。

Mode 参数选择移动的类型：

- 0：绝对位置
- 1：相对位置
- 2：单速连续正向旋转
- 3：单速连续反向旋转

当运动轴完成此子例程时，Done 参数会开启。

开启 Abort 参数会命令运动轴停止执行此命令并减速，直至电机停止。

Error 参数 (页 1161)包含该子例程的结果。

C_Pos 参数包含运动轴的当前位置。根据测量单位，该值是脉冲数 (DINT) 或工程单位数 (REAL)。

C_Speed 参数包含运动轴的当前速度。根据所选的测量单位，该值是脉冲数/每秒 (DINT) 或工程单位数/每秒 (REAL)。

超驰

正在运行的 AXISx_GOTO 指令可由其它 AXISx_GOTO 指令超驰。仅当在绝对或相对模式时，AXISx_GOTO 指令才支持超驰功能。

超驰功能可用于更改正在运行的 AXISx_GOTO 指令的速度、位置和运动模式。触发超驰后，

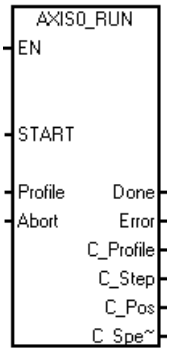
- 如果速度已更改，CPU 将加速或减速，以达到新的目标速度。
- 如果运动方向改为相反方向，CPU 将减速至停止，然后再沿更改的方向运动。
- 超驰中的无效组态，将打断当前的运动。

可通过以下两种方式触发超驰：

- 通过上升沿，重新激活正在运行的 AXISx_GOTO 指令。
- 运行 AXISx_GOTO 指令时，激活另一个 AIXSx_GOTO 指令。

15.4.3.5 AXISx_RUN 子例程

表格 15-12 AXISx_RUN

LAD/FBD	STL	说明
	**CALL AXISx_RUN, START, Profile, Abort, Done, Error, C_Profile, C_Step, C_Pos, C_Speed**	AXISx_RUN 子例程（运行曲线）命令运动轴按照存储在组态/曲线表的特定曲线执行运动操作。

表格 15-13 AXISx_RUN 子例程的参数

输入/输出	数据类型	操作数
START	BOOL	I、Q、V、M、SM、S、T、C、L、能流
Profile	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD、常数
Abort、Done	BOOL	I、Q、V、M、SM、S、T、C、L
Error、C_Profile、C_Step	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD
C_Pos、C_Speed	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD

开启 EN 位会启用此子例程。 确保 EN 位保持开启，直至 Done 位指示子例程执行已经完成。

开启 START 参数将向运动轴发出 RUN 命令。 对于在 START 参数开启且运动轴当前不繁忙时执行的每次扫描，该子例程向运动轴发送一个 RUN 命令。 为了确保仅发送了一个命令，请使用边沿检测元素用脉冲方式开启 START 参数。

Profile 参数包含运动曲线的编号或符号名称。 “Profile”输入必须介于 0 - 31。否则子例程将返回错误。

开启 Abort 参数会命令运动轴停止当前曲线并减速，直至电机停止。

当运动轴完成此子例程时，Done 参数会开启。

Error 参数 (页 1161)包含该子例程的结果。

C_Profile 参数包含运动轴当前执行的曲线。

C_Step 参数包含目前正在执行的曲线步。

C_Pos 参数包含运动轴的当前位置。 根据测量单位，该值是脉冲数 (DINT) 或工程单位数 (REAL)。

C_Speed 参数包含运动轴的当前速度。 根据所选的测量单位，该值是脉冲数/每秒 (DINT) 或工程单位数/每秒 (REAL)。

15.4.3.6 AXISx_RSEEK 子例程

表格 15-14 AXISx_RSEEK

LAD/FBD	STL	说明
	CALL AXISx_RSEEK, START, Done, Error	AXISx_RSEEK 子例程（搜索参考点位置）使用组态/曲线表中的搜索方法启动参考点搜索操作。运动轴找到参考点且运动停止后，运动轴将 RP_OFFSET 参数值载入当前位置。 在运动轴组中，该子例程也用于为轴组中的每个轴执行参考点搜索操作。

表格 15-15 AXISx_RSEEK 子例程的参数

输入/输出	数据类型	操作数
START	BOOL	I、Q、V、M、SM、S、T、C、L、能流
Done	BOOL	I、Q、V、M、SM、S、T、C、L
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD

RP_OFFSET 的默认值为 0。可使用运动向导、运动控制面板或 AXISx_LDOFF（加载偏移量）子例程来更改 RP_OFFSET 值。

开启 EN 位会启用此子例程。确保 EN 位保持开启，直至 Done 位指示子例程执行已经完成。

开启 START 参数将向运动轴发出 RSEEK 命令。对于在 START 参数开启且运动轴当前不繁忙时执行的每次扫描，该子例程向运动轴发送一个 RSEEK 命令。为了确保仅发送了一个命令，请使用边沿检测元素用脉冲方式开启 START 参数。

当运动轴完成此子例程时，Done 参数会开启。

Error 参数 (页 1161)包含该子例程的结果。

15.4.3.7 AXISx_LDOFF 子例程

表格 15-16 AXISx_LDOFF

LAD/FBD	STL	说明
	CALL AXISx_LDOFF, START, Done, Error	AXISx_LDOFF 子例程（加载参考点偏移量）建立一个与参考点处于不同位置的新的零位置。 在执行该子例程之前，您必须首先确定参考点的位置。 您还必须将机器移至起始位置。 当子例程发送 LDOFF 命令时，运动轴计算起始位置（当前位置）与参考点位置之间的偏移量。 运动轴然后将算出的偏移量存储到 RP_OFFSET 参数并将当前位置设为 0。这将起始位置建立为零位置。 如果电机失去对位置的追踪（例如断电或手动更换电机的位置），您可以使用 AXISx_RSEEK 子例程自动重新建立零位置。

表格 15-17 AXISx_LDOFF 子例程的参数

输入/输出	数据类型	操作数
START	BOOL	I、Q、V、M、SM、S、T、C、L、能流
Done	BOOL	I、Q、V、M、SM、S、T、C、L
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD

开启 EN 位会启用此子例程。 确保 EN 位保持开启，直至 Done 位指示子例程执行已经完成。

开启 START 参数将向运动轴发出 LDOFF 命令。 对于在 START 参数开启且运动轴当前不繁忙时执行的每次扫描，该子例程向运动轴发送一个 LDOFF 命令。 为了确保仅发送了一个命令，请使用边沿检测元素用脉冲方式开启 START 参数。

当运动轴完成此子例程时，Done 参数会开启。

Error 参数 (页 1161)包含该子例程的结果。

15.4.3.8 AXISx_LDPOS 子例程

表格 15-18 AXISx_LDPOS

LAD/FBD	STL	说明
	CALL AXISx_LDPOS, START, New_Pos, Done, Error, C_Pos	AXISx_LDPOS 子例程（加载位置）将运动轴中的当前位置值更改为新值。 您还可以使用本子例程为任何绝对移动命令建立一个新的零位置。

表格 15-19 AXISx_LDPOS 子例程的参数

输入/输出	数据类型	操作数
START	BOOL	I、Q、V、M、SM、S、T、C、L、能流
New_Pos、C_Pos	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD
Done	BOOL	I、Q、V、M、SM、S、T、C、L
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD

开启 EN 位会启用此子例程。 确保 EN 位保持开启，直至 Done 位指示子例程执行已经完成。

开启 START 参数将向运动轴发出 LDPOS 命令。 对于在 START 参数开启且运动轴当前不繁忙时执行的每次扫描，该子例程向运动轴发送一个 LDPOS 命令。 为了确保仅发送了一个命令，请使用边沿检测元素用脉冲方式开启 START 参数。

New_Pos 参数提供新值，用于取代运动轴报告和用于绝对移动的当前位置值。 根据测量单位，该值是脉冲数 (DINT) 或工程单位数 (REAL)。

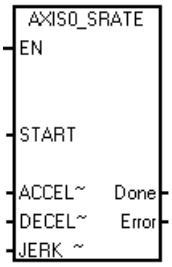
当运动轴完成此子例程时，Done 参数会开启。

Error 参数 (页 1161)包含该子例程的结果。

C_Pos 参数包含运动轴的当前位置。 根据测量单位，该值是脉冲数 (DINT) 或工程单位数 (REAL)。

15.4.3.9 AXISx_SRATE 子例程

表格 15-20 AXISx_SRATE

LAD/FBD	STL	说明
	**CALL AXISx_SRATE, START, ACCEL_Time, DECEL_Time, JERK_Time, Done, Error**	AXISx_SRATE 子例程（设置速率）命令运动轴更改加速、减速和急停时间。

表格 15-21 AXISx_SRATE 子例程的参数

输入/输出	数据类型	操作数
START	BOOL	I、Q、V、M、SM、S、T、C、L
ACCEL_Time、DECEL_Time、JERK_Time	DINT	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD、常数
Done	BOOL	I、Q、V、M、SM、S、T、C、L
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD

开启 EN 位会启用此子例程。 确保 EN 位保持开启，直至 Done 位指示子例程执行已经完成。

开启 START 参数会将新时间值复制到组态/曲线表中，并向运动轴发出一个 SRATE 命令。 对于在 START 参数开启且运动轴当前不繁忙时执行的每次扫描，该子例程向运动轴发送一个 SRATE 命令。 为了确保仅发送了一个命令，请使用边沿检测元素用脉冲方式开启 START 参数。

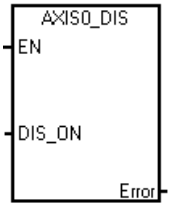
ACCEL_Time、DECEL_Time 和 JERK_Time 参数用于确定新的加速时间、减速时间以及急停时间，单位为毫秒 (ms)。

当运动轴完成此子例程时，Done 参数会开启。

Error 参数 (页 1161)包含该子例程的结果。

15.4.3.10 AXISx_DIS 子例程

表格 15-22 AXISx_DIS

LAD/FBD	STL	描述
	**CALL AXISx_DIS, DIS_ON, Error**	AXISx_DIS 子例程将运动轴的 DIS 输出打开或关闭。这样，可使用 DIS 输出禁用或启用电机控制器。如果在运动轴中使用 DIS 输出，可以在每次扫描时调用该子例程，或者仅在需要更改 DIS 输出值时进行调用。

表格 15-23 AXISx_DIS 子例程的参数

输入/输出	数据类型	操作数
DIS_ON	BOOL	I、Q、V、M、SM、S、T、C、L、能流
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD

当 EN 位打开以启用子例程时，DIS_ON 参数控制运动轴的 DIS 输出。

说明

如果未在运动控制向导中定义“DIS”输出，AXISx_DIS 子例程将返回错误。

说明

AXISx_DIS 子例程不能用于控制已组态的轴组。在轴组直线插补运动中使用 AXISx_DIS 子例程将提示错误消息。

Error 参数 (页 1161)包含该子例程的结果。

15.4.3.11 AXISx_CFG 子例程

表格 15-24 AXISx_CFG

LAD/FBD	STL	说明
	CALL AXISx_CFG, START, Done, Error	AXISx_CFG 子例程（重新加载组态）命令运动轴从组态/曲线表指针指定的位置读取组态块。 运动轴然后将新组态与现有组态进行比较，并执行任何所需的设置更改或重新计算。

表格 15-25 AXISx_CFG 子例程的参数

输入/输出	数据类型	操作数
START	BOOL	I、Q、V、M、SM、S、T、C、L、能流
Done	BOOL	I、Q、V、M、SM、S、T、C、L
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD

开启 EN 位会启用此子例程。 确保 EN 位保持开启，直至 Done 位指示子例程执行已经完成。

开启 START 参数将向运动轴发出 CFG 命令。 对于在 START 参数开启且运动轴当前不繁忙时执行的每次扫描，该子例程向运动轴发送一个 CFG 命令。 为了确保仅发送了一个命令，请使用边沿检测元素用脉冲方式开启 START 参数。

当运动轴完成此子例程时，Done 参数会开启。

Error 参数 (页 1161)包含该子例程的结果。

15.4.3.12 AXISx_CACHE 子例程

表格 15-26 AXISx_CACHE

LAD/FBD	STL	说明
	CALL AXISx_CACHE, START, Profile, Done, Error	AXISx_CACHE 子例程（缓冲曲线）命令运动曲线在执行前先缓冲。 这可在执行前预先缓冲所需命令。 预先缓冲可缩短从执行运动指令到开始运动的时间，并可带来一致性。

表格 15-27 AXISx_CACHE 子例程的参数

输入/输出	数据类型	操作数
START	BOOL	I、Q、V、M、SM、S、T、C、L、能流
Profile	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD、常数
Abort、Done	BOOL	I、Q、V、M、SM、S、T、C、L
Error、C_Profile、C_Step	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD
C_Pos、C_Speed	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD

开启 EN 位会启用此子例程。 确保 EN 位保持开启，直至 Done 位指示子例程执行已经完成。

开启 START 参数将向运动轴发出 CACHE 命令。 对于在 START 参数开启且运动轴当前不繁忙时执行的每次扫描，该子例程向运动轴发送一个 CACHE 命令。 为了确保仅发送了一个命令，请使用边沿检测元素用脉冲方式开启 START 参数。

Profile 参数包含运动曲线的编号或符号名称。 “Profile”输入必须介于 0 - 31。否则子例程将返回错误。

当运动轴完成此子例程时，Done 参数会开启。

Error 参数 (页 1161)包含该子例程的结果。

15.4.3.13 AXISx_RDPOS 子例程

表格 15-28 AXISx_RDPOS

LAD/FBD	STL	说明
	CALL AXISx_RDPOS, Error, I_Pos	AXISx_RDPOS 子例程返回当前运动轴位置。

表格 15-29 AXISx_RDPOS 子例程的参数

输入/输出	数据类型	操作数
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD
I_Pos	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD

开启 EN 位会启用此子例程。

Error 参数 (页 1161)包含该子例程的结果。

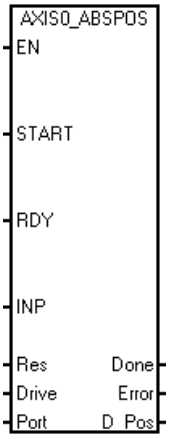
I_Pos 参数包含当前运动轴位置。

说明

执行此命令将返回轴当前的实际位置。 AXISx_CTRL 和 AXISx_GOTO 等其它运动控制子例程中提供的位置状态值将定期进行更新。 因此，通过其它命令报告的位置值与通过该命令报告的位置值可能会略有不同，但这是正常现象。

15.4.3.14 AXISx_ABSPOS 子例程

表格 15-30 AXISx_ABSPOS

LAD/FBD	STL	说明
	**CALL AXISx_ABSPOS, START, RDY, INP, Res, Drive, Port, Done, Error, D_Pos**	AXISx_ABSPOS 子例程通过特定的 Siemens 伺服驱动器（例如 V90）读取绝对位置。读取绝对位置值的目的是为了更新运动轴中的当前位置值。在将 SINAMICS V90 伺服驱动器与安装了绝对值编码器的 SIMOTICS-1FL6 伺服电机结合使用时，支持此功能。

表格 15-31 AXISx_ABSPOS 子例程的参数

输入/输出	数据类型	操作数
START	BOOL	I、Q、V、M、SM、S、T、C、L、能流
RDY、INP	BOOL	I、Q、V、M、SM、S、T、C、L
Res	DINT	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD、常数
Drive	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD、常数
Port	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD、常数
Done	BOOL	I、Q、V、M、SM、S、T、C、L
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD
D_Pos	REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD

开启 EN 位会启用此子例程。确保 EN 位保持开启，直至 DONE 位指示子例程执行已经完成。

开启 START 参数可通过指定驱动器获取当前绝对位置。为了确保仅执行一项当前位置读取操作，请使用边沿检测元素以脉冲方式开启 START 参数。

RDY 参数指示伺服驱动器处于就绪状态，而该状态通常通过驱动器的数字输出信号提供。仅当该参数开启时，此例程才会通过驱动器读取绝对位置。

INP 参数指示电机处于静止状态，而该状态通常通过驱动器的数字输出信号提供。仅当该参数开启时，此例程才会通过驱动器读取绝对位置。

Res 参数必须设置为与伺服电机相连的绝对编码器的分辨率。例如，连有绝对编码器的 SIMOTICS S-1FL6 伺服电机的单匝分辨率为 20 位或 1048576。

将 Drive 参数设置为与要通过该子例程访问的伺服驱动器的 RS485 地址相匹配。各驱动器的有效地址是 0 至 31。

将 Port 参数设置为指示要用于与伺服驱动器通信的 CPU 端口：

- 0：板载 RS485 端口（端口 0）
- 1：RS485/RS232 信号板（如存在，端口 1）

在子例程的工作完成时，Done 参数会开启。

Error 参数 (页 1161)包含该子例程的结果。

D_Pos 参数包含由伺服驱动器返回的当前绝对位置。

说明

要使用该子程序，必须根据工程单位组态运动轴的测量系统设置。

说明

附加子程序

在向导组态中启动位置读取功能时，运动向导还将创建 ABSPOS_SBR 和 ABSPOS_INT 子程序以从驱动读取绝对位置。

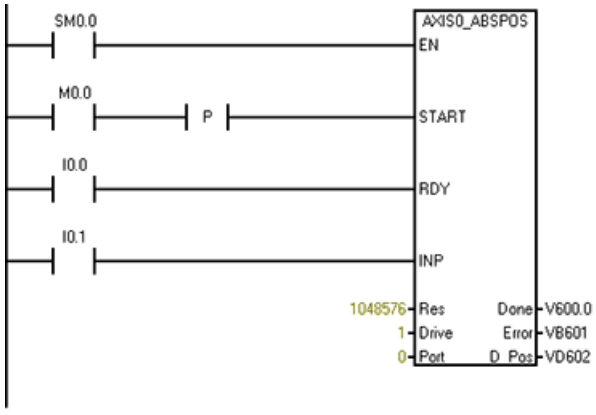
15.4.4 使用 AXISx_ABSPOS 子程序从 SINAMICS 伺服驱动器读取绝对位置

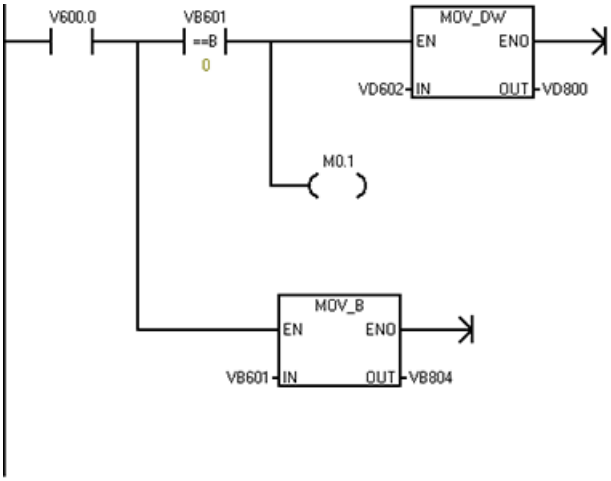
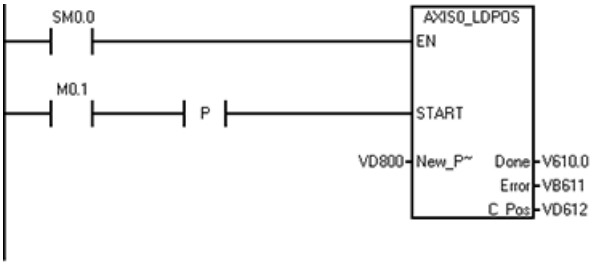
以下各节介绍如何在项目中使用 AXISx_ABSPOS 子程序从 SINAMICS V90 伺服驱动读取绝对位置。

15.4.4.1 AXISx_ABSPOS 和 AXISx_LDPOS 子程序应用示例

若是 START 参数启用，则只有成功执行 AXISx_ABSPOS 子程序（Done 参数 = ON 和 Error 参数 =“无错误”），绝对位置才有效。子程序在 START 输入禁用状态下执行时，Error 和 D_Pos 参数恢复为默认值，此时必须在程序中加入在子程序执行完成时捕获有效绝对位置值的指令。

表格 15-32 示例：使用 AXISx_ABSPOS 子程序从 SINAMICS V90 伺服驱动读取绝对位置。

LAD/FBD	说明	STL
程序段 1：  	从驱动读取伺服位置。	**LD SM0.0** **= L60.0** **LD M0.0** **EU** **= L63.7** **LD I0.0** **= L63.6** **LD I0.1** **= L63.5** **LD L60.0** **CALL AXIS0_ABSPOS, L63.7, L63.6, L63.5, 1048576,1, 0, V600.0, VB601, VD602**

程序段 2：  	操作完成后，捕获错误代码；如无错误，则捕获伺服位置。	**LD V600.0** **LPS** **AB= VB601, 0** **MOVD VD602, VD800** **= M0.1** **LPP** **MOVB VB601, VB804**
程序段 3：  	将运动轴的当前位置更新为捕获的伺服位置值。	**LD SM0.0** **= L60.0** **LD M0.1** **EU** **= L63.7** **LD L60.0** **CALL AXIS0_LDPOS, L63.7, VD800, V610.0, VB611, VD612**

15.4.4.2 互连

数字量 I/O

请参见 *SINAMICS V90/SIMOTICS S-1FL6 操作指令*文档中的“PLC 连接示例”部分，以获取在 S7-200 SMART CPU 和 V90 伺服驱动之间连接建议的数字量控制信号所用的接线图。

通信

AXISx_ABSPOS 子程序使用两个设备之间 RS485 链路上的串行通信从驱动获取位置数据。因此，使用电缆连接 S7-200 SMART CPU（或，用户的 CPU 型号支持时，连接 S7-200 SMART CM01 信号板）上的 RS485 端口和 V90 伺服驱动器上的 RS485 端口。

请参见 *S7-200 SMART 系统手册*和 *SINAMICS V90/SIMOTICS S-1FL6 操作指令*文档的相关部分，了解对 S7-200 SMART CPU 和 V90 伺服驱动上 RS485 端口的相关介绍。

15.4.4.3 调试

控制模式

“PTI”模式是一种驱动控制模式设置，它允许通过外部脉冲串对运动速度和距离进行控制。V90 伺服驱动中的默认控制模式是基本“PTI”模式，但可通过读取“p29003”参数的值并核对该值是否为“0”来检查该模式设置。也可借助 S7-200 SMART CPU 的脉冲串输出使用复合控制模式（PTI/S 和 PTI/T）。一些高级功能不属于本文档的讨论范围。请参见 *SINAMICS V90/SIMOTICS S-1FL6 操作指令*文档，获取这些功能的帮助信息。

设定值脉冲输入通道

为使用 S7-200 SMART CPU 的数字量输出正确操作，必须在 V90 伺服驱动中为设定值脉冲输入通道参数选择“24 V DC 单端脉冲串输入”（参数“p29014”= 1）。

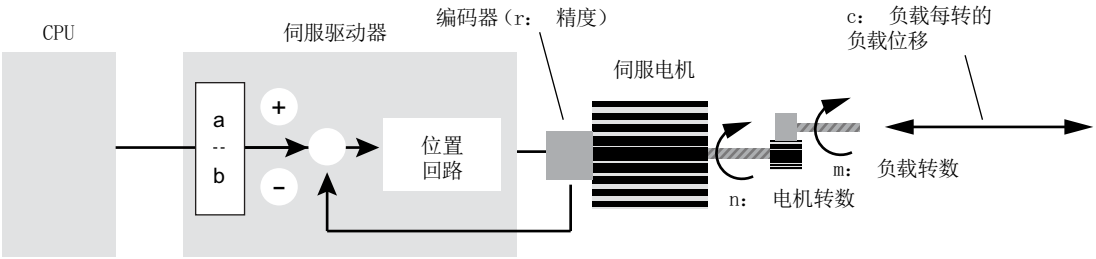
设定值脉冲串输入格式

确保 CPU 的运动轴输出相位和极性设置（在 STEP 7-Micro/WIN SMART V3 运动向导的“方向控制”(Directional Control) 对话框中完成）与 V90 伺服驱动器的设定点脉冲串输入格式设置（参数“p29010”）一致。

共同的工程单位基础

使用 S7-200 SMART CPU 上的运动轴控制伺服电机的运动速度和距离时，必须在运动轴 (CPU) 和驱动之间定义共同的工程单位。

下图显示了运动系统的组成部分：



要在 CPU 和伺服驱动之间定义共同的工程单位，必须在调试系统时考虑以下运动系统变量：

电子齿轮

在 V90 伺服驱动中，“a”和“b”的值决定驱动的电子齿轮传动比，它表示的是 CPU 发出的脉冲串的频率转换特性。由于来自 S7-200 SMART CPU 运动轴的最大脉冲频率是 100 kHz，而装有绝对编码器的 SIMOTICS S-1FL6 伺服电机的编码器分辨率为 2^20 脉冲/转，因此在许多应用中，借助驱动的电子齿轮特性可能实现更高的电机转速。例如，要相对于供给驱动的 CPU 脉冲串频率，在伺服电机内实现 10x 的设定值脉冲频率增量，则必须将电子齿轮传动比设置为“10:1”。

在 V90 伺服驱动中，通过参数“p29012[0]”设置电子齿轮传动比的分子（“a”），通过参数“p29013”设置传动比的分母（“b”）。另外，在使用电子齿轮时，请将参数“p29011”设为“0”。在 V90 伺服驱动中，电子齿轮传动比 (a/b) 的有效范围在“0.02”和“200”之间。

更多信息请参见 *SINAMICS V90/SIMOTICS S-1FL6 操作指令*文档的“电子齿轮传动比”部分。

机械因素

“m”和“n”值构成负载转数和电机转数间的机械关系，适用于使用齿轮机构的情况。若 V90 驱动处于“PTI”模式下，其内部机械齿轮传动比参数固定为“1:1”，但实际的“m”和“n”值对于为运动轴确定正确的工程单位转换因子非常重要，如下文所述。

“c”值构成负载运动（以指定工程单位表示）和负载转数的关系。“负载每转移动 20 cm”和“负载每转转动 360 度”便是该转换因子的示例。

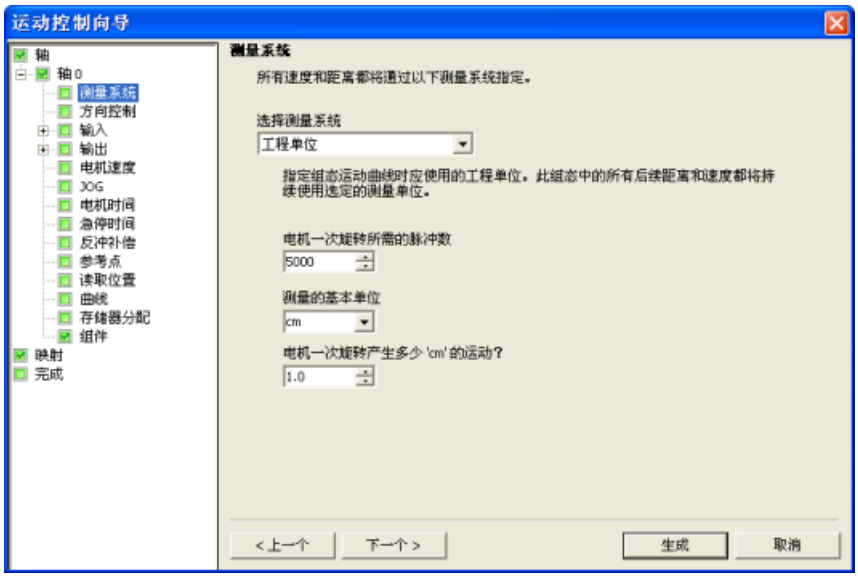
编码器分辨率

“r”值是伺服电机中绝对编码器的分辨率。如上所述，装有绝对编码器的 SIMOTICS S-1FL6 伺服电机的编码器分辨率为 2^20 脉冲/转或“1048576”。当 V90 伺服驱动配上带绝对编码器的电机时，驱动将自动检测编码器类型和获取编码器分辨率。不过，必须在 AXISx_ABSPOS 子程序的“Res”输入参数中以及下述一种工程单位换算因数计算中，指定该分辨率值。

运动向导中的测量系统设置

使用 STEP 7-Micro/WIN SMART V3 运动向导组态 CPU 运动轴的测量系统时，必须分配三个转换设置：

- 第一个设置：将 CPU 脉冲数转换为电机转数
- 第二个设置：创建基础工程单位名称
- 第三个设置：将电机转速转换为负载运动



设置 1：“电机旋转一周所需的脉冲数”(Number of pulses required for one motor revolution)

该设置定义 CPU 脉冲数和电机转数之间的关系。正确得出该设置值的相关方程如下：

(1) 电机旋转一周所需的脉冲数 = r * (b / a)

其中，“r”= 编码器分辨率，以编码器脉冲数/电机转表示，
“a"和“b”= 电子齿轮 (E-gear) 传动比参数（“a”= V90 参数“p29012[0]”的值；“b”=V90 参数“p29013”的值）

例如：如果所需的电子齿轮传动比为“128:1”并且电机绝对编码器分辨率为“2^20”或“1048576”，则：

“电机旋转一周所需的脉冲数”= **1048576 * (1 / 128) = 8192**

设置 2：“基本测量单位”

该设置通过运动向导为速度和距离设置创建基本工程单位名称。为避免混淆，选择的单位应与负载的工程单位匹配。例如，如果负载移动和速度以“cm”和“cm/秒” 表示，则应选择“cm”。

设置 3：“电机每旋转一周产生多少 "xxx" 的运动?”(One motor revolution produces how many "xxx" of motion?)

该设置定义电机转数和负载运动（以指定的工程单位如 cm 或 度表示）之间的关系。正确得出该设置值的相关方程如下：

(2) 电机每旋转一周产生多少 "xxx" 的运动 = c * (m / n)

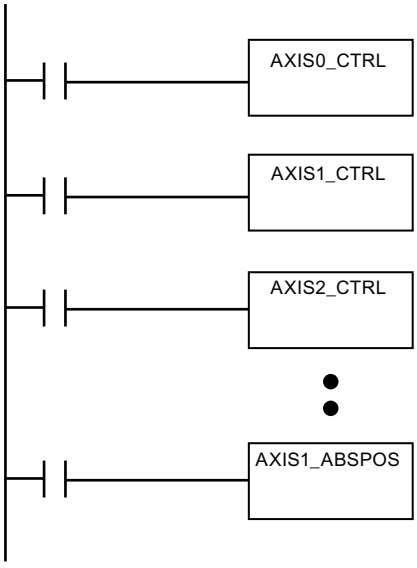
其中，“c”=负载运动（以指定的工程单位表示）/负载转，
“m/n”= 外部齿轮传动比（以负载转数/电机转表示）

例如，机械齿轮传动比为“1:2”并且负载每旋转一周移动 10 cm，则：

“电机每旋转一周产生多少 cm 的运动”=**10 * (1 / 2) = 5**

15.4.4.4 重要事项须知

- 请勿从中断例程内部或其中调用的子程序中调用 AXISx_ABSPOS 子程序
- 如在 CPU 项目中组态了多个运动轴，需确保控制所有轴的 AXISx_CTRL 子程序比控制任何轴的第一个 AXISx_ABSPOS 子程序优先执行。AXISx_CTRL 子程序包含初始化 V 存储区的代码，程序中 AXISx_ABSPOS 子程序的所有实例共用该存储区来管理与伺服驱动的通信。



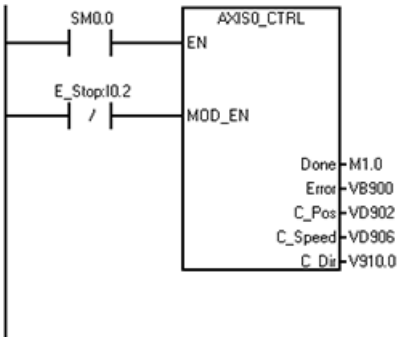
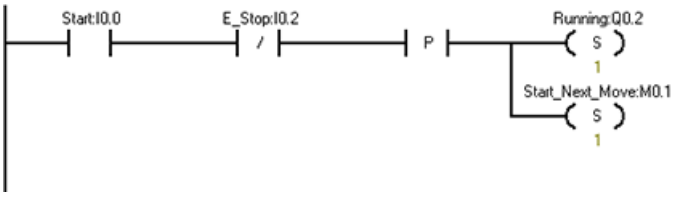
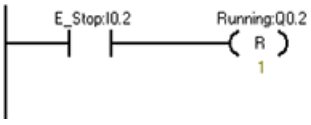
- 如果您根据“相对脉冲”设置而不是“工程单位”设置组态运动轴测量系统，则仍可以使用 AXISx_ABSPOS 子程序从 V90 伺服驱动返回位置信息。不过，需注意子程序“D_pos”参数中返回的位置值将是 DINT 类型的值，并且是伺服电机报告的实际位置值（没有对该值执行工程单位换算）。

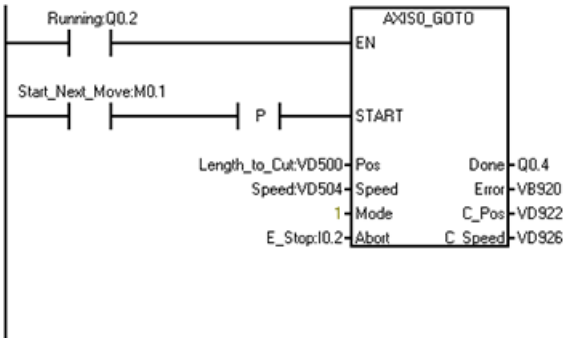
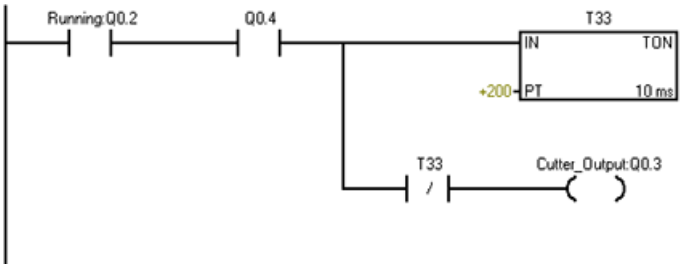
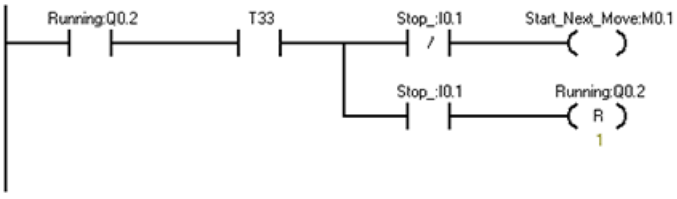
15.4.5 运动轴示例程序

15.4.5.1 运动轴简单相对移动（定长截断应用）示例

示例程序说明一个简单相对移动，使用 AXISx_CTRL 和 AXISx_GOTO 子程序执行定长截断操作。此程序不需要 RP 搜索模式或运动曲线，并且可以使用脉冲或工程单位测量长度。输入长度 (VD500) 和目标速度 (VD504)。I0.0（启动）打开时，会启动机器。I0.1（停止）打开时，机器会完成当前操作并停止。I0.2 (E_Stop) 打开时，机器会中止所有运动并立即停止。

表格 15-33 示例：运动轴简单相对移动（定长截断应用）

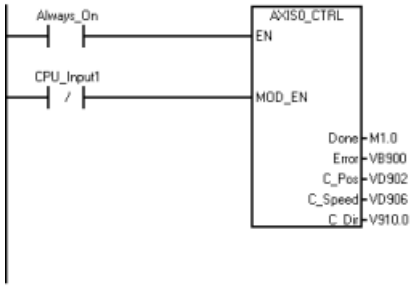
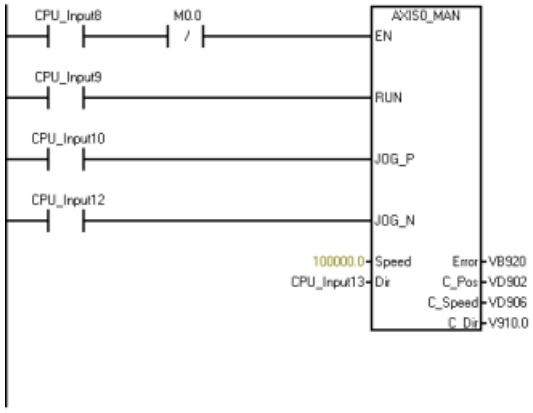
LAD/FBD	说明	STL
程序段 1：  	控制指令	LD SM0.0 = L60.0 LDN I0.2 = L63.7 LD L60.0 CALL AXIS0_CTRL, L63.7, M1.0, VB900, VD902, VD906, V910.0
程序段 2：  	开始使机器处于自动模式。	LD I0.0 AN I0.2 EU S Q0.2, 1 S M0.1, 1
程序段 3：  	E_Stop：立即停止并关闭自动模式。	LD I0.2 R Q0.2, 1

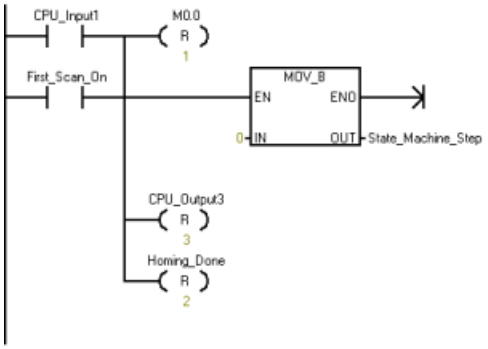
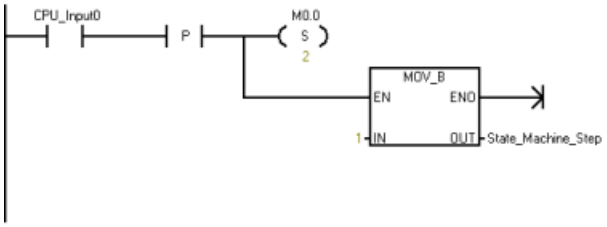
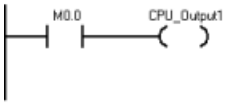
LAD/FBD	说明	STL
程序段 4：  	1. 移动到某个点： 2. 输入要剪切的长度。 3. 为目标速度输入“速度”。 4. 将模式设置为 1（相对模式）。	LD Q0.2 = L60.0 LD M0.1 EU = L63.7 LD L60.0 CALL AXIS0_GOTO, L63.7, VD500, VD504, 1, I0.2, Q0.4, VB920, VD922, VD926
程序段 5：  	就位时，开启刀具 2 秒钟以完成剪切。	LD Q0.2 A Q0.4 TON T33, +200 AN T33 = Q0.3
程序段 6：  	剪切完成时，重新启动，直至“停止”处于激活状态。	LD Q0.2 A T33 LPS AN I0.1 = M0.1 LPP A I0.1 R Q0.2, 1

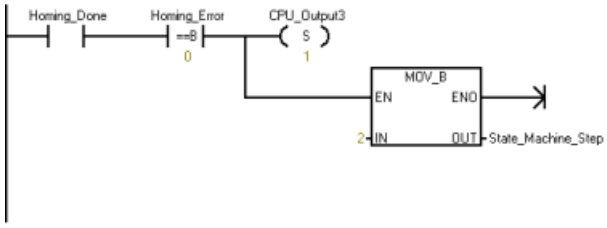
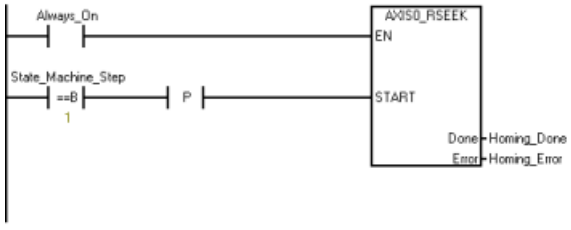
15.4.5.2 运动轴 AXISx_CTRL、AXISx_RUN、AXISx_SEEK 和 AXISx_MAN 示例

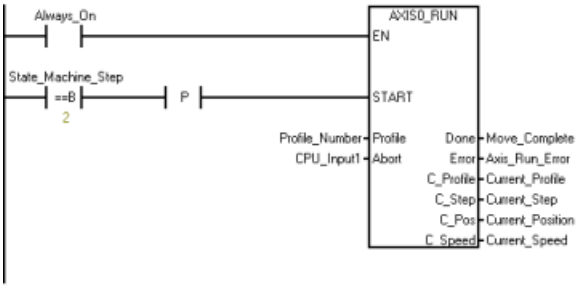
本程序提供 AXISx_CTRL、AXISx_RUN、AXISx_RSEEK 和 AXISx_MAN 子程序的示例。用户必须组态 RP 搜索模式和运动轨迹。

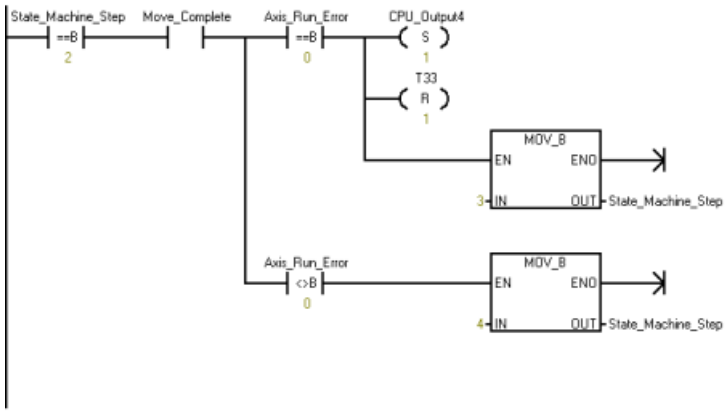
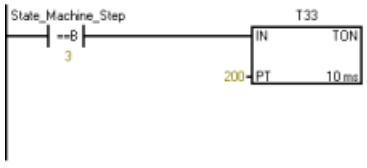
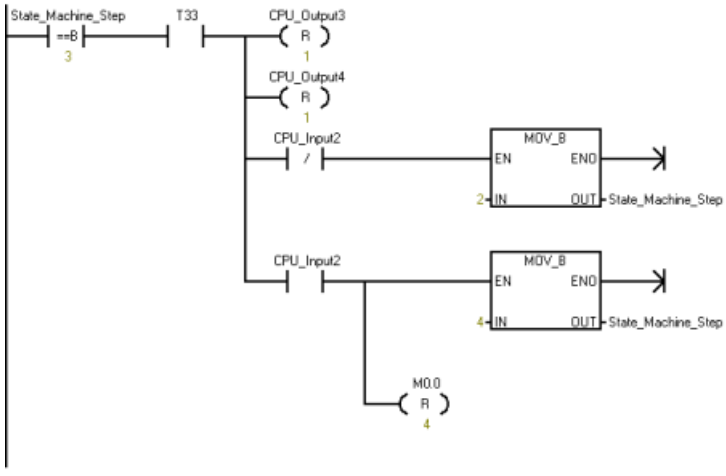
表格 15-34 示例：运动轴 AXISx_CTRL、AXISx_RUN、AXISx_SEEK 和 AXISx_MAN 子程序应用

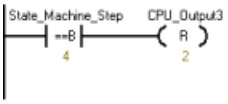
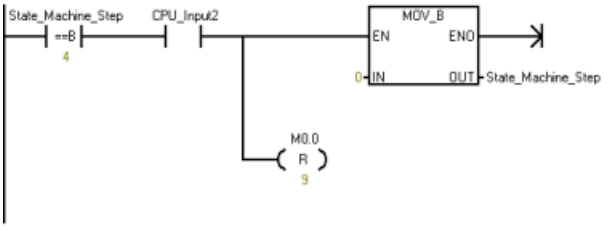
LAD/FBD	说明	STL
程序段 1  	通过关闭 CPU_Input1 启用轴。 符号和地址：[1] • Always_On = SM0.0 • AXIS0_CTRL = SBR1 • CPU_Input1 = I0.1	LD Always_On = L60.0 LDN CPU_Input1 = L63.7 LD L60.0 CALL AXIS0_CTRL, L63.7, M1.0, VB900, VD902, VD906, V910.0
程序段 2  	使用点动命令将轴移至已知位置。现在可手动移动轴。 符号和地址：[1] • AXIS0_MAN = SBR2 • CPU_Input10 = I1.2 • CPU_Input12 = I1.4 • CPU_Input13 = I1.5 • CPU_Input8 = I1.0 • CPU_Input9 = I1.1	LD CPU_Input8 AN M0.0 = L60.0 LD CPU_Input9 = L63.7 LD CPU_Input10 = L63.6 LD CPU_Input12 = L63.5 LD L60.0 CALL AXIS0_MAN, L63.7, L63.6, L63.5, 100000.0, CPU_Input13, VB920, VD902, VD906, V910.0

LAD/FBD	说明	STL
程序段 3  	复位过程。将初始步设为“0”。 符号和地址：[1] • CPU_Input1 = I0.1 • CPU_Output3 = Q0.3 • First_Scan_On = SM0.1 • Homing_Done = M1.1 • State_Machine_Step = VB1500	**LD CPU_Input1** **O First_Scan_On** **R M0.0, 1** **MOVB 0, State_Machine_Step** **R CPU_Output3, 3** **R Homing_Done, 2**
程序段 4  	启动过程。当 CPU_Input0 从关闭切换到开启时，会将 State_Machine_Step 设为“1”。 符号和地址：[1] • CPU_Input0 = I0.0 • State_Machine_Step = VB1500	**LD CPU_Input0** **EU** **S M0.0, 2** **MOVB 1, State_Machine_Step**
程序段 5  	该程序段用于开启 CPU_Input1。 符号和地址：[1] • CPU_Input1 = I0.1	**LD M0.0** **= CPU_Input1**

LAD/FBD	说明	STL
程序段 6  	完成归位后，移动至下一步。 符号和地址：[1] • CPU_Output3 = Q0.3 • Homing_Done = M1.1 • Homing_Error = VB930 • State_Machine_Step = VB1500	**LD Homing_Done** **AB= Homing_Error** **S CPU_Output3** **MOVB 2, State_Machine_Step**
程序段 7  	当状态机处于步 1 时，系统自动将轴归位。如果存在归位错误，Homing_Error 输出将显示错误代码。 符号和地址：[1] • Always_On = SM0.0 • AXIS0_RSEEK = SBR5 • Homing_Done = M1.1 • Homing_Error = VB930 • State_Machine_Step = VB1500	**LD Always_On** **= L60.0** **LDB State_Machine_Step, 1** **EU** **= L63.7** **LD L60.0** **CALL AXIS0_RSEEK, L63.7, Homing_Done, Homing_Error**

LAD/FBD	说明	STL
程序段 8  	当状态机进入步 2 时，将移动所选的曲线 符号和地址：[1] • Always_On = SM0.0 • AXIS0_RUN = SBR4 • Axis_Run_Error = VB940 • CPU_Input1 = I0.1 • Current_Position = VD914 • Current_Profile = VB941 • Current_Speed = VD948 • Current_Step = VB942 • Move_Complete = M1.2 • Profile_Number = VB228 • State_Machine_Step = VB1500	**LD Always_On** **= L60.0** **LDB= State_Machine_ Step, 2** **EU** **= L63.7** **LD L60.0** **CALL AXIS0_RUN, L63.7, Profile_Number, CPU_Input1, Move_Complete, Axis_Run_Error, Current_Profile, Current_Step, Current_Position , Current_Speed**

LAD/FBD	说明	STL
程序段 9  	当状态机处于步 2 并完成移动时，评估错误状态。如果没有错误，状态机转向步 3。如果存在错误，状态机转向步 4 处理误差。 符号和地址：[1] • Axis_Run_Error = VB940 • CPU_Output4 = Q0.4 • Move_Complete = M1.2 • State_Machine_Step = VB1500	LDB= State_Machine_ Step, 2 A Move_Complete LPS AB= Axis_Run_Error, 0 S CPU_Output4, 1 R T33, 1 MOVB 3, State_Machine_ Step LPP AB<> Axis_Run_Error, 0 MOVB 4, State_Machine_ Step
程序段 10  	等待步 3。 符号和地址：[1] • State_Machine_Step = VB1500	LDB= State_Machine_ Step, 3 TON T33, 200
程序段 11  	如果状态机移至步 3，等待 2 s。之后，评估开关状态，并重新开始移动或停止。 符号和地址：[1] • CPU_Input2 = I0.2 • CPU_Output3 = Q0.3 • CPU_Output4 = Q0.4 • State_Machine_Step = VB1500	LDB= State_Machine_ Step, 3 A T33 LPS R CPU_Output3, 1 R CPU_Output4, 1 AN CPU_Input2 MOVB 2, State_Machine_ Step LPP A CPU_Input2 MOVB 4, State_Machine_ Step R M0.0, 4

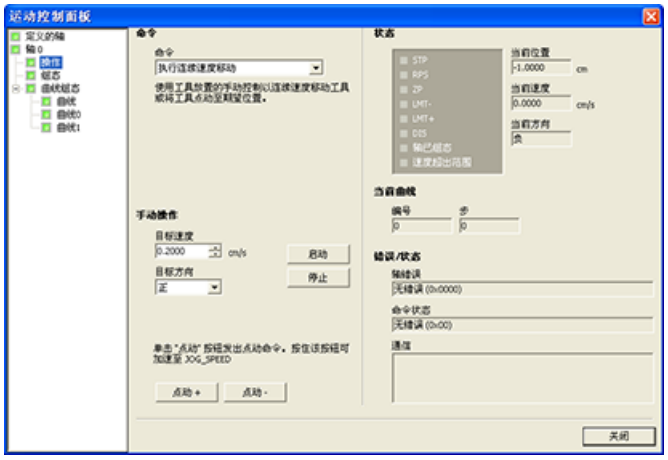
LAD/FBD	说明	STL
程序段 12  	如果状态机移至步 4，清空输出。 符号和地址：[1] • CPU_Output3 = Q0.3 • State_Machine_Step = VB1500	**LDB= State_Machine_ Step, 4** **R CPU_Output3, 2**
程序段 13 	状态机处于步 4 时，输出 5 闪烁已指示出现错误。 符号和地址：[1] • Clock_1s = SM0.5（时钟脉冲，工作周期时间为 1 秒时，接通 0.5 秒，关断 0.5 秒。） • CPU_Output5 = Q0.5 • State_Machine_Step = VB1500	**LDB= State_Machine_ Step, 4** **A Clock_1s** **= CPU_Output5**
程序段 14  	状态机处于步 4 时，必须通过切换输入 I0.2 的方式确认错误。该操作会将状态重置回步 0。 符号和地址：[1] • CPU_Input2 = I0.2 • State_Machine_Step = VB1500	**LDB= State_Machine_ Step, 4** **A CPU_Input2** **MOVB 0, State_Machine_ Step** **R M0.0, 9**

[1] 所示程序地址均为地址示例。您的程序地址可能有所不同。

15.4.6 监视运动轴

15.4.6.1 显示和控制运动轴的操作

在“操作”(Operation) 节点，您可与运动轴的操作进行交互。 控制面板显示运动轴的当前速度、当前位置和当前方向。 还可看到输入和输出 LED 的状态（脉冲 LED 除外）。



通过控制面板可与运动轴进行交互，您可以更改速度和方向、停止和启动运动以及使工具点动运行（如果 CPU 已停止）。

说明

在 CPU 运行时无法执行运动控制命令。 CPU 必须处于 STOP 模式才能要更改速度和方向、停止和启动运动以及使用点动工具。

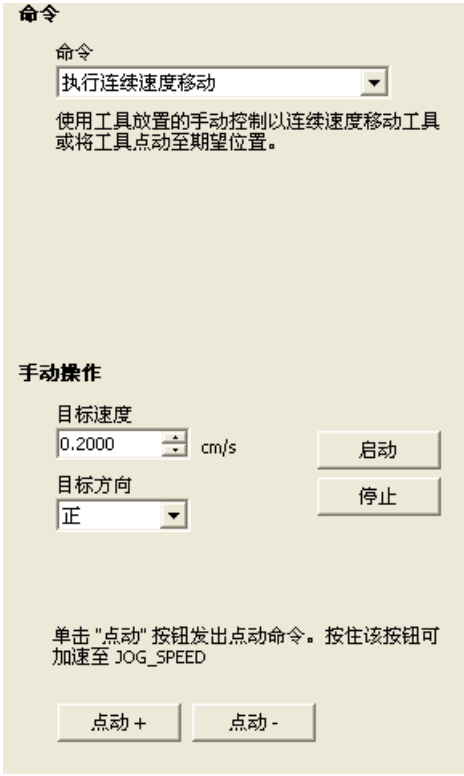
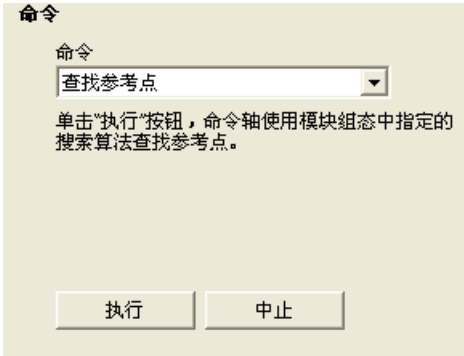
说明

运动命令激活时，退出运动控制面板或失去通信会使该轴在 5 秒超时过后停止运动。

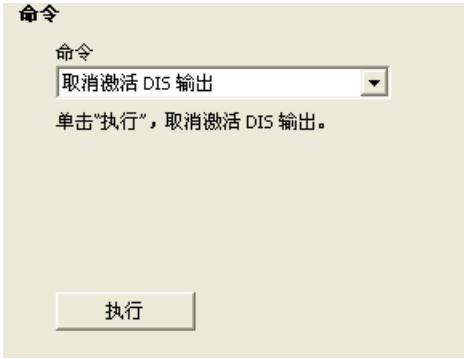
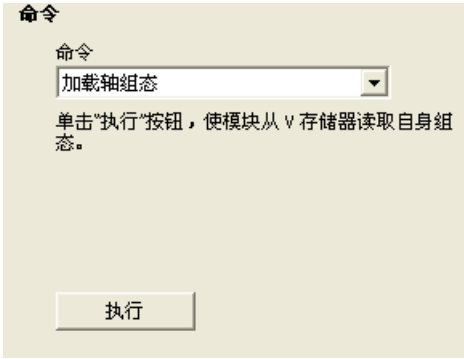
运动命令

还可生成下列运动命令：

表格 15-35　运动控制面板命令

命令	说明
	执行连续速度移动： 执行此命令后即可采用手动控制方式定位工具。 输入“目标速度”(Target Speed) 和“方向”(Direction)，然后单击“启动”(Start) 执行连续移动。 在单击“停止”(Stop)（或发生错误情况）前，运动将持续进行。
	查找参考点： 此命令使用组态的搜索模式查找参考点。 单击“执行”(Execute)，轴将使用轴组态中所指定的搜索算法发出“查找参考点”(Seek to Reference Point) 命令。 单击“中止”(Abort) 将在找到参考点之前中止查找过程。

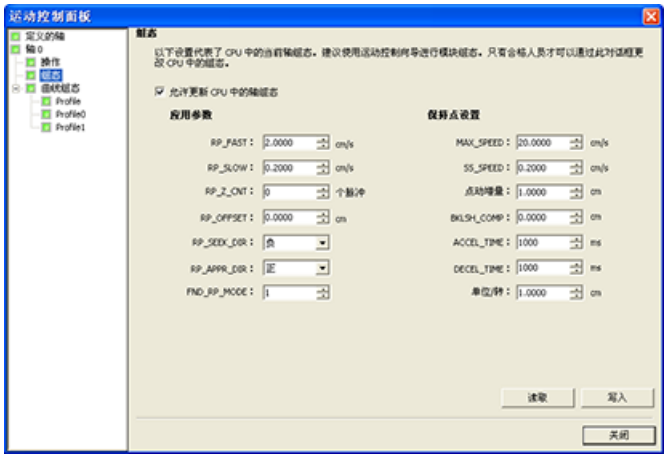
命令	说明
命令 命令 加载参考点偏移量 使用手动控制将工具放置在新零点位置。单击"执行"按钮将此位置保存为 RP_OFFSET。当前位置将为零。 执行 中止 手动操作 目标速度 0.2000 cm/s 启动 目标方向 停止 正 单击 "点动" 按钮发出点动命令。按住该按钮可加速至 JOG_SPEED 点动 + 点动 -	加载参考点偏移量： 采用手动控制方式使工具点动运行到新位置后，加载“参考点偏移量”(Reference Point Offset)。 使用手动控制将工具置于新位置。 单击“执行”(Execute) 将此位置另存为“RP_OFFSET”。 当前位置将设置为零点。
命令 命令 重新加载当前位置 输入位置进行设置，然后单击"执行"按钮更新当前位置。这样也会建立一个新零点位置。 新位置 1.0000 cm 执行	重新加载当前位置： 此命令更新当前位置值并建立新的零位置。 输入要设置的位置并单击“执行”(Execute) 更新当前位置。 这还将建立新的零位置。
命令 命令 激活 DIS 输出 单击"执行"按钮激活 DIS 输出。 执行	激活 DIS 输出： 此命令开启运动轴的 DIS 输出。 单击“执行”(Execute) 激活 DIS 输出。

命令	说明
	取消激活 DIS 输出： 此命令关闭运动轴的 DIS 输出。 单击“执行”(Execute) 取消激活 DIS 输出。
	加载轴组态： 此名令通过命令运动轴从 CPU 的 V 存储区读取组态块的方式加载新组态。 单击“执行”(Execute) 使轴从 V 储存器读取组态。
	移至绝对位置： 此命令允许以目标速度移至指定位置。 在使用此命令前，必须已建立零位置。 指定“目标速度”(Target Speed) 和要移至的“绝对位置”(Absolute Position)。 该选项要求定义零位置。

命令	说明
命令 命令 以相对量移动 指定目标速度和想要从当前位置移动的距离。您可指定正距离或负距离。 手动操作 目标速度 0.2000 cm/s 启动 停止 目标位置 1.0000 cm 单击 "点动" 按钮发出点动命令。按住该按钮可加速至 JOG_SPEED 点动 + 点动 -	移动相对量： 此命令允许以目标速度从当前位置移动指定距离。 可输入正向或负向距离。 指定“目标速度”(Target Speed) 和要移至的“目标位置”(Target Position)。
命令 命令 重置轴命令接口 当轴不响应命令时，此选项有用。此选项将清除轴的命令字节并设置位。 执行	复位轴命令接口： 此命令清除运动轴的轴命令接口并将“Done”置位。 如果运动轴出现不响应任何命令的情况，使用此命令。
命令 命令 执行曲线 设置曲线编号，然后单击"执行"按钮启动曲线运行。 曲线编号 Profile 执行 中止	执行曲线： 此命令允许您选择要执行的曲线。 控制面板显示运动轴正在执行的曲线的状态。 选择要执行的曲线，单击“执行”(Execute)，然后轴将执行该曲线。 注： 只有已在运动控制向导中定义曲线的情况下才可使用此命令。

15.4.6.2 显示和修改运动轴的组态

在“组态”(Configuration) 节点，您可查看和修改存储在 CPU 数据块中的运动轴组态设置。



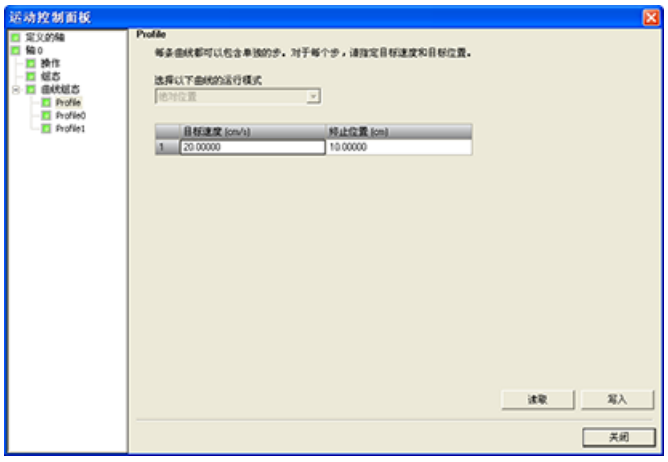
在修改组态设置后，只需单击写入按钮便可将数据值传到 CPU。 这些数据值不保存在 STEP 7-Micro/WIN SMART 项目中。 必须对项目进行可反映出这些字段最终值的手动更改。

15.4.6.3 显示运动轴的曲线组态

在“曲线组态”(Profile Configuration) 节点，您可查看运动轴每条曲线的组态。

单击各曲线可查看其工作模式和数据值。

可在此对话框中修改曲线的一部分数据值。 在修改组态设置后，只需单击写入按钮便可将数据值传到 CPU。 这些数据值不保存在 STEP 7-Micro/WIN SMART 项目中。 必须对项目进行可反映出这些字段最终值的手动更改。



15.4.6.4 运动轴错误代码（SMW620、SMW670、SMW720 或 SMW770 的 WORD）

表格 15-36 运动轴错误代码

错误代码	描述
0	无错误
1	预留
2	组态块不存在
3	组态块指针错误
4	组态块大小超出可用的 V 存储器
5	组态块格式非法
6	指定的曲线过多
7	指定的 STP_RSP 非法
8	指定的 LIM- 非法
9	指定的 LIM+ 非法
10	指定的 FILTER_TIME 非法
11	指定的 MEAS_SYS 非法
12	指定的 RP_CFG 非法
13	PLS/REV 值非法
14	UNITS/REV 值非法
15	RP_ZP_CNT 值非法

错误代码	描述
16	JOG_INCREMENT 值非法
17	MAX_SPEED 值非法
18	SS_SPD 值非法
19	RP_FAST 值非法
20	RP_SLOW 值非法
21	JOG_SPEED 值非法
22	ACCEL_TIME 值非法
23	DECEL_TIME 值非法
24	JERK_TIME 值非法
25	BKLSH_COMP 值非法
26	轴不可用
27	LMT+ 位置无效
28	LMT- 位置无效
29	STP 位置无效
30	RPS 位置无效
31	ZP 位置无效
32	输出相位非法
33	未定义 RPS 输入（如果已定义“归零”，则 RPS 也必须定义。）
34	触发器位置无效
35	预留
36	未定义 ZP 输入（如果已定义“归零”模式 3 或 4，则 ZP 也必须定义。）
37	相位 A (P0) 输出冲突
38	相位 B (P1) 输出冲突
39	DIS 输出冲突
40	预留
41	SDB0 记录大小无效
42	SDB0 格式非法
43 到 127	预留

要验证运动轴接线是否正确、调整组态数据以及测试各运动曲线，使用运动控制面板。

如果需要在运动轴中进行其它更改，转到运动控制向导。

15.4.6.5 运动轴指令的错误代码（SMB634、SMB684、SMB734 或 SMB784 的七个 LS 位）

每个轴的 SM 表中会预留一个字节，显示运动指令的结果（偏移量 34）。该字节指示指令完成时间和指令是否有错误。

表格 15-37 运动指令的错误代码

错误代码	描述
0	无错误
1	由用户打断
2	组态错误 （如果 SDB0 组态出错，则发生此错误。）
3	命令非法
4	因组态无效而打断 （如果组态表出错，则发生此错误。）
5	预留
6	由于无定义的参考点而打断
7	由于 STP 输入激活而打断
8	由于 LMT- 输入激活而打断
9	由于 LMT+ 输入激活而打断
10	由于执行运动时出错而打断
11	没有为指定曲线组态曲线块
12	非法操作模式
13	此命令不支持该操作模式
14	曲线块中的步数非法
15	方向更改非法
16	距离非法
17	达到目标速度前发生 RPS/TRIG 触发
18	RPS 有效区域宽度不足
19	速度超出范围
20	距离不足无法执行所需的速度更改
21	位置非法
22	零位置未知
23	未定义 DIS 输出
24	预留

错误代码	描述
25	因 CPU 转入 STOP 模式而打断
26	因运动控制面板活动结束而打断
27	AXIS_GOTO 超驰功能中的操作模式无效
28	轴组正在运行，因此禁止单轴运动
29 到 127	预留
128	运动轴无法处理此指令：要么是运行轴忙于执行另一指令，要么是没有此指令的启动脉冲。
129	预留
130	运动轴未启用
131	预留
132	预留
133	指定的曲线非法。AXISx_RUN 和 AXISx_CACHE 指令曲线编号的取值范围必须在 0 到 31 之间。
134	在 AXISx_GOTO 指令中指定了非法模式

要验证运动轴接线是否正确、调整组态数据以及测试各运动曲线，使用运动控制面板。

如果需要在运动轴中进行其它更改，转到运动控制向导。

15.4.7 高级主题

15.4.7.1 理解运动轴的组态/曲线表

概述

我们开发的运动控制向导能够根据您对运动控制系统相关问题所做的回答自动生成组态和曲线信息，从而可使运动应用更容易。组态/包络表信息供要创建个人运动控制例程的高级用户使用。

组态/包络表位于 S7-200 SMART CPU 的 V 存储区。如下表所示，组态设置存储在下列信息类型中：

- **组态块：**包含在准备执行位置命令时用于设置运动轴的信息
- **交互块：**支持由用户程序直接设置位置参数
- **曲线块：**描述运动轴要执行的预定义移动操作。最多可组态 32 个包络块。

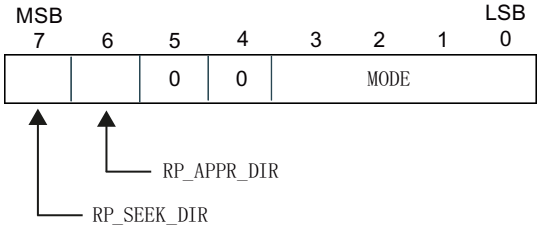
说明

组态/包络表的包络块可包含多达 32 个运动包络。要创建 32 个以上移动包络，可以通过更改存储在组态/包络表指针中的值来交换组态/包络表。

表格 15-38 组态/曲线表：组态块

组态/曲线表			
字节偏移量	**名称**	**功能说明**	**类型**
组态块			
0	MOD_ID	运动轴标识字段	--
5	CB_LEN	以字节为单位的组态块长度（1 字节）	--
6	IB_LEN	以字节为单位的交互块长度（1 字节）	--
7	PF_LEN	以字节为单位的单条曲线长度（1 字节）	--
8	STP_LEN	以字节为单位的单步长度（1 字节）	--
9	STEPS	每条曲线允许的步数（1 字节）	--
10	PROFILES	从 0 到 32 的包络数（1 字节）	--
11	--	保留：设置为 0	--
13	--	保留：设置为 0	--
14	STOP_RSP	指定驱动器对 STP 输入的响应（1 字节）： • 0：无动作。忽略输入条件。 • 1：减速至停止并指示 STP 输入激活。 • 2：终止脉冲并指示 STP 输入。 • 3 到 255：保留（如果指定，将出错）	--
15	LMT-_RSP	指定驱动器对负限位输入的响应（1 字节）： • 0：无动作。忽略输入条件。 • 1：减速至停止并指示到达限位。 • 2：终止脉冲并指示到达限位。 • 3 到 255：保留（如果指定，将出错）	--

组态/曲线表			
字节偏移量	名称	功能说明	类型
组态块			
16	LMT+_RSP	指定驱动器对正限位输入的响应（1 字节）： • 0：无动作。忽略输入条件。 • 1：减速至停止并指示到达限位。 • 2：终止脉冲并指示到达限位。 • 3 到 255：保留（如果指定，将出错）	--
17	--	保留：设置为 0	--
18	MEAS_SYS	指定描述移动的度量系统（1 字节）： • 0：脉冲（测得的速度为每秒脉冲数，测得的位置值为脉冲数；这些值为双整数） • 1：工程单位（测得的速度为每秒单位数，测得的位置值为单位数；这些值为单精度实数） • 2 到 255：保留（如果指定，将出错）	--
19	--	保留：设置为 0	--
20	PLS/REV	指定电机每转的脉冲数（仅当 MEAS_SYS 设置为 1 时可用）（4 字节）	DInt
24	UNITS/REV	指定电机每转的工程单位数（仅当 MEAS_SYS 设置为 1 时可用）（4 字节）	Real
28	UNITS	为 STEP 7-Micro/WIN SMART 保留，用于存储自定义单位字符串（4 字节）	--

组态/曲线表			
字节偏移量	名称	功能说明	类型
组态块			
32	RP_CFG	指定参考点搜索组态（1 字节）：  RP_SEEK_DIR：该位指定参考点搜索的起始方向（0 - 正向，1 - 反向）。 RP_APPR_DIR：该位指定终止参考点搜索的接近方向（0 - 正向，1 - 负向）。 MODE：指定参考点搜索方法 '0000'：参考点搜索被禁用。 '0001'：参考点在 RPS 输入变为激活状态的位置。 '0010'：参考点在 RPS 输入激活区域的中央。 '0011'：参考点在 RPS 输入激活区域外部。 '0100'：参考点在 RPS 输入激活区域内部。 '0101'至'1111'：保留（如果指定，将出错）	--
33	--	保留：设置为 0	--
34	RP_Z_CNT	用于定义参考点的 ZP 输入脉冲数（4 字节）	DInt
38	RP_FAST	RP 搜索操作快速：MAX_SPD 或更低（4 字节）	DInt/Real
42	RP_SLOW	RP 搜索操作慢速：电机可瞬时停止的最大速度或更低（4 字节）	DInt/Real
46	SS_SPEED	启动/停止速度。（4 字节）： 启动速度是电机可以立即从停止状态到达的最大速度以及电机可以立即进入停止状态的最大速度。允许低于该速度运行，但加速和减速时间不适用。	DInt/Real
50	MAX_SPEED	电机的最大运行速度（4 字节）	DInt/Real

组态/曲线表			
字节偏移量	名称	功能说明	类型
组态块			
54	JOG_SPEED	点动速度（4 字节）：MAX_SPEED 或更低（4 字节）	DInt/Real
58	JOG_INCREMENT	点动增量值：为响应一个点动脉冲而移动的距离（或脉冲数）（4 字节）。	DInt/Real
62	ACCEL_TIME	从最小速度加速到最大速度所需的时间（4 字节），单位为毫秒	DInt
66	DECEL_TIME	从最大速度减速到最小速度所需的时间（4 字节），单位为毫秒	DInt
70	BKLSH_COMP	间隙补偿：用于对方向改变时的系统反向间隙进行补偿的距离（4 字节）。	DInt/Real
74	JERK_TIME	急停补偿应用于加速 / 减速曲线（S 曲线）开始和结束部分的时间。指定一个 0 数值禁止急停补偿。急停时间以毫秒为单位。（范围：0 ms 到 32000 ms。（4 字节）	DInt

表格 15-39 组态/曲线表：交互块

组态/曲线表			
字节偏移量	名称	功能说明	类型
交互块			
78	MOVE_CMD	选择操作模式（1 字节）： • 0：绝对位置 • 1：相对位置 • 2：单速连续正向旋转 • 3：单速连续反向旋转 • 4：手动速度控制，正向旋转 • 5：手动速度控制，反向旋转 • 6：通过触发来停止的单速连续正向旋转（激活 RPS 可触发停止信号；TARGET_POS 包含要在信号发出后行进的距离） • 7：通过触发来停止的单速连续反向旋转（激活 RPS 可触发停止信号；TARGET_POS 包含要在信号发出后行进的距离） • 8 到 255：保留（如果指定，将出错）	--
79	--	保留：设置为 0	--
80	TGT_POS	本次移动的目标位置（4 字节）	DInt/Real
84	TGT_SPEED	本次移动的目标速度（4 字节）	DInt/Real
88	RP_OFFSET	参考点的绝对位置（4 字节）	DInt/Real

表格 15-40 组态/曲线表：曲线块 0

组态/曲线表			
字节偏移量	名称	功能说明	类型
曲线块 0			
92(+0)	STEPS	该移动序列中的步数（1 字节）	--
93(+1)	MODE	选择此曲线块的操作模式（1 字节）： • 0：绝对位置 • 1：相对位置 • 2：单速连续正向旋转 • 3：单速连续反向旋转 • 4：保留（如果指定，将出错） • 5：保留（如果指定，将出错） • 6：通过触发来停止的单速连续正向旋转（RPS 输入指示停止） • 7：通过触发来停止的单速连续反向旋转（RPS 输入指示停止） • 8：双速连续正向旋转（RPS 选择速度） • 9：双速连续反向旋转（RPS 选择速度） • 10：通过触发来停止的双速连续正向旋转（RPS 选择速度，TRIG 输入指示停止） • 11：通过触发来停止的双速连续反向旋转（RPS 选择速度，TRIG 输入指示停止） • 12 到 255：保留（如果指定，将出错）	--
94(+2)	第 0 步：POS	移动步 0 要到达的位置（4 字节）	DInt/Real
98(+6)	第 0 步：SPEED	移动步 0 的目标速度（4 字节）	DInt/Real
102(+10)	第 1 步：POS	移动步 1 要到达的位置（4 字节）	DInt/Real
106(+14)	第 1 步：SPEED	移动步 1 的目标速度（4 字节）	DInt/Real
110(+18)	第 2 步：POS	移动步 2 要到达的位置（4 字节）	DInt/Real
114(+22)	第 2 步：SPEED	移动步 2 的目标速度（4 字节）	DInt/Real
118(+26)	第 3 步：POS	移动步 3 要到达的位置（4 字节）	DInt/Real
122(+30)	第 3 步：SPEED	移动步 3 的目标速度（4 字节）	DInt/Real
...	步 ...：POS	移动步 ... 要到达的位置（4 字节）	DInt/Real
...	步 ...：SPEED	移动步 ... 的目标速度（4 字节）	DInt/Real

组态/曲线表			
字节偏移量	名称	功能说明	类型
曲线块 0			
214(+122)	第 15 步：POS	移动步 15 要到达的位置（4 字节）	DInt/Real
218(+126)	第 15 步：SPEED	移动步 15 的目标速度（4 字节）	DInt/Real

说明

在组态/曲线表的包络块 0 中可有 1 至 16 步。

表格 15-41 组态/曲线表：包络块 1

组态/曲线表			
字节偏移量	名称	功能说明	类型
包络块 1			
X [1]	STEPS	该移动序列中的步数（1 字节） **注**：最多可有 16 步。	--
(X + 1)	MODE	选择此包络块的操作模式（1 字节）	--
(X + 2)	第 0 步：POS	移动步 0 要到达的位置（4 字节）	DInt/Real
(X + 4)	第 0 步：SPEED	移动步 0 的目标速度（4 字节）	DInt/Real
…	…	…	…

[1] 包络块 1 与后续块的偏移量可变，取决于在最大曲线中组态的步数。偏移量由以下公式确定：包络块 x 的偏移量 = CB_LEN + IB_LEN + (x * PF_LEN)

表格 15-42 模式 0 的曲线详细信息（绝对位置）

距离包络起始位置的字节偏移量	步号	名称	域大小	值
+0		STEPS	字节	n = 此包络中组态的步数
+1		MODE	字节	0 = 绝对位置
+2	0	POS	dint/fp	步 0 中的目标位置

距离包络起始位置的字节偏移量	步号	名称	域大小	值
+6		SPEED	dint/fp	步 0 的目标速度
● ● ●				
(4 * n) + 2	n	POS	dint/fp	步 n 中的目标位置
($ * N) + 6		SPEED	dint/fp	步 n 的目标速度

表格 15-43　模式 1 的曲线详细信息（相对位置）

距离包络起始位置的字节偏移量	步号	名称	域大小	值
+0		STEPS	字节	n = 此包络中组态的步数
+1		MODE	字节	0 = 相对位置
+2	0	POS	dint/fp	步 0 中的行进距离
+6		SPEED	dint/fp	步 0 的目标速度
● ● ●				
(4 * n) + 2	n	POS	dint/fp	步 n 中的行进距离
($ * N) + 6		SPEED	dint/fp	步 n 的目标速度

表格 15-44　模式 2（单速连续正向旋转）和模式 3（单速连续反向旋转）包络的详细信息

距离包络起始位置的字节偏移量	步号	名称	域大小	值
+0		STEPS	字节	1
+1		MODE	字节	2 = 单速连续正向旋转或者 3 = 单速连续反向旋转
+2	0	POS	dint/fp	不适用（必须设为 0）
+6		SPEED	dint/fp	目标速度

表格 15-45 模式 6（通过触发来停止的单速连续正向旋转）和模式 7（通过触发来停止的单速连续反向旋转）包络的详细信息

距离包络起始位置的字节偏移量	步号	名称	域大小	值
+0		STEPS	字节	1
+1		MODE	字节	6 = 通过触发来停止的单速连续正向旋转或者 7 = 通过触发来停止的单速连续反向旋转
+2	0	POS	dint/fp	激活 RPS 信号后的行进距离（必须为正值）
+6		SPEED	dint/fp	目标速度

表格 15-46 模式 8（双速连续正向旋转）和模式 9（双速连续反向旋转）包络的详细信息

距离包络起始位置的字节偏移量	步号	名称	域大小	值
+0		STEPS	字节	2
+1		MODE	字节	8 = 双速连续正向旋转或者 9 = 双速连续反向旋转
+2	0	POS	dint/fp	不适用（必须设为 0）
+6		SPEED	dint/fp	RPS 信号未激活时的目标速度
+10	1	POS	dint/fp	不适用（必须设为 0）
+14		SPEED	dint/fp	RPS 信号激活时的目标速度

表格 15-47 模式 10（通过触发来停止的双速连续正向旋转）和模式 11（通过触发来停止的双速连续反向旋转）包络的详细信息

距离包络起始位置的字节偏移量	步号	名称	域大小	值
+0		STEPS	字节	2
+1		MODE	字节	10 = 通过触发来停止的双速连续正向旋转或者 11 = 通过触发来停止的双速连续反向旋转
+2	0	POS	dint/fp	激活 TRIG 信号后的行进距离（必须为正值）
+6		SPEED	dint/fp	RPS 信号未激活时的目标速度
+10	1	POS	dint/fp	不适用（必须设为 0）
+14		SPEED	dint/fp	RPS 信号激活时的目标速度

15.4.7.2 运动轴的特殊存储器 (SM) 位置

CPU 为每条运动轴分配 50 个字节的特殊存储器 (SM)。（请参见下表。）当运动轴检测到错误状态或数据状态变化时，运动轴会更新这些 SM 位置。第一个运动轴根据需要更新 SMB600 至 SMB649，报告错误状态，第二个运动轴更新 SMB650 至 SMB699，其余依此类推。

表格 15-48 专用存储器字节 SMB600 到 SMB799

运动轴的 SM 字节：			
运动轴 0	**运动轴 1**	**运动轴 2**	**运动轴 3**
SMB600 至 SMB649	SMB650 至 SMB699	SMB700 至 SMB749	SMB750 到 SMB799

下表显示了分配给运动轴的 SM 数据区域结构。定义以轴 0 为例进行说明。

表格 15-49 运动轴 0 的特殊存储区定义

SM 地址	描述
SMB600 至 SMB615	轴名称（16 个 ASCII 字符）。SMB600 是第一个字符：“轴 0”
SMB616 至 SMB619	预留
SMW620	轴 0：错误代码（请参见“运动轴错误代码” (页 1159)列表。）

SM 地址	描述
SMB622	轴 0：输入/输出状态：反映输入和输出的状态 MSB 7 6 5 4 3 2 1 LSB 0 DIS | 0 | TRIG | STP | LMT- | LMT+ | RPS | ZP • DIS（禁止输出）： – 0 = 无电流 – 1 = 有电流 • TRIG （停止输入）： – 0 = 无电流 – 1 = 有电流 • STP（停止输入）： – 0 = 无电流 – 1 = 有电流 • LMT-（负向限位输入）： – 0 = 无电流 – 1 = 有电流 • LMT+（正向限位输入）： – 0 = 无电流 – 1 = 有电流 • RPS（参考点开关输入）： – 0 = 无电流 – 1 = 有电流 • ZP（零脉冲输入）： – 0 = 无电流 – 1 = 有电流

SM 地址	描述
SMB623	轴 0 瞬时状态：反映组态状态和转向状态 MSB 7 6 5 4 3 2 1 LSB 0 0 0 0 0 0 OR R CFG • OR（目标速度超出范围）： – 0 = 未超出范围 – 1 = 超出范围 • R（旋转方向）： – 0 = 正向旋转 – 1 = 负向旋转 • CFG（模块组态）： – 0 = 未组态 – 1 = 已组态
SMB624	轴 0：CUR_PF 是一个指示当前正在执行的包络的字节。
SMB625	轴 0：CUR_STP 是一个指示当前正在包络中执行的步的字节。
SMD626	轴 0：CUR_POS 是指示运动轴当前位置的双字值。
SMD630	轴 0：CUR_SPD 是指示运动轴当前速度的双字值。
SMB634	轴 0：指令的结果。大于 127 的错误条件由运动控制向导创建的指令子例程生成。 MSB 7 6 LSB 0 D ERROR • D（Done 位）： – 0 = 正在运行 – 1 = 操作完成（初始化期间由运动轴置位） • ERROR：（请参见“运动指令错误代码” (页 1161)列表。）
SMB635 至 SMB645	预留
SMD646	轴 0：组态/曲线表的 V 存储位置的指针。指向 V 存储器以外区域的指针值无效。运动轴监视该位置，直至该位置收到一个非零的指针值。

15.4.8 了解运动轴的 RP 搜索模式

下图提供每个 RP 搜索模式的不同选项图：

- RP 搜索： 模式 1 显示 RP 搜索模式 1 的两个选项。此模式将 RP 定位在靠近工作区一侧的 RPS 输入开始激活的位置。
- RP 搜索： 模式 2 显示 RP 搜索模式 2 的两个选项。此模式将 RP 定位在 RPS 输入的激活区域的中心。
- RP 搜索： 模式 3 显示 RP 搜索模式 3 的两个选项。此模式将 RP 定位在 RPS 输入的激活区域外的指定数量的零脉冲 (ZP) 处。
- RP 搜索： 模式 4 显示 RP 搜索模式 4 的两个选项。此模式将 RP 定位在 RPS 输入的激活区域内的指定数量的零脉冲 (ZP) 处。

对于每种模式，存在四种 RP 搜索方向和 RP 接近方向组合。 （只显示了两种组合。） 这些组合决定了 RP 搜索操作的模式。 对于每种组合，也存在四种不同的起点：

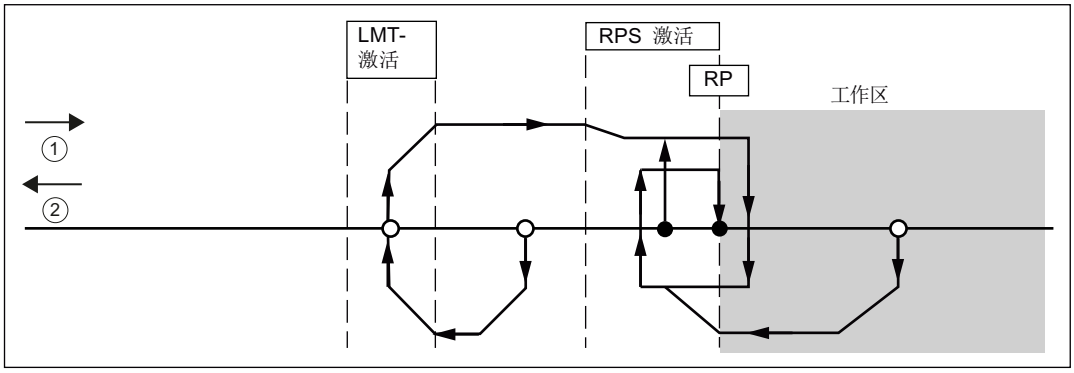
已定位每个图的工作区，以便从参考点移动到工作区需要以 RP 接近方向相同的方向移动。 通过在此方向选择工作区的位置，搜索参考点后移除了所有机械齿轮系统的反冲，以便第一次移动到工作区。

说明

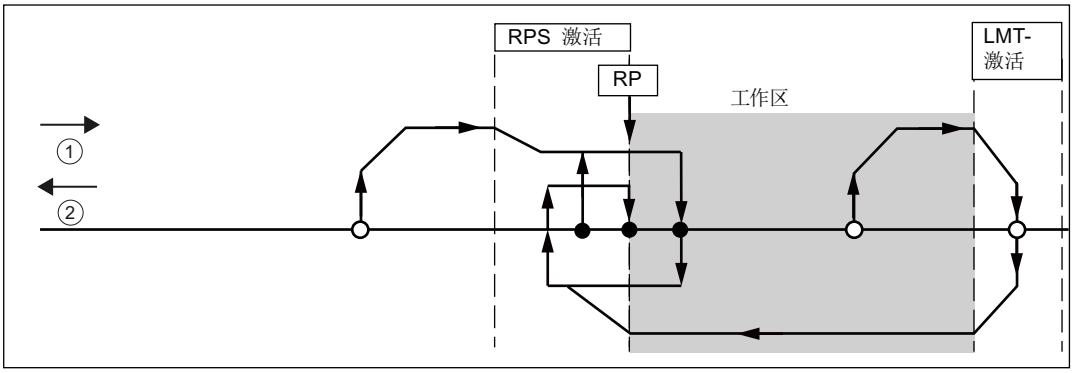
必须使能 RPS 输入才能使用 RP 搜索功能。 若要使用 RP 搜索模式 3 或模式 4，则还必须使能 ZP 输入，除非在进入 RPS 有效区后将要接收的 ZP 脉冲的数量组态为“0”。

RP 搜索模式 1

默认组态：RP 搜索方向：负方向，RP 接近方向：正方向



默认组态：RP 搜索方向：正方向，RP 接近方向：正方向

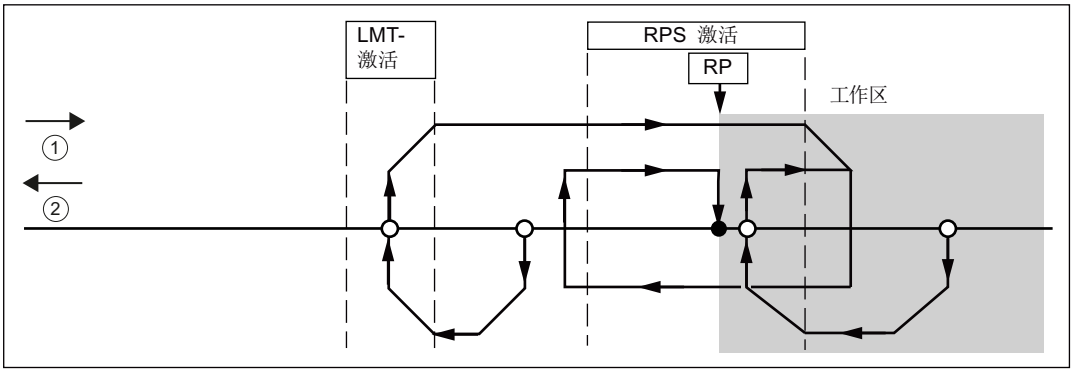


①: 正方向运动

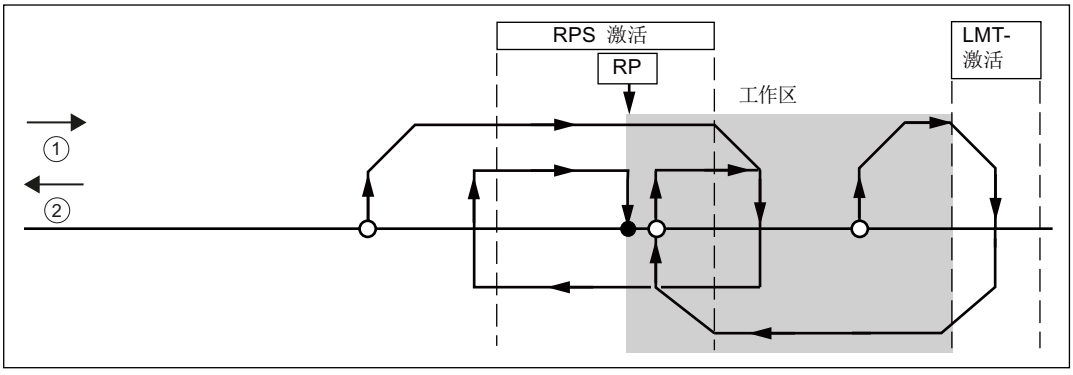
②: 负方向运动

RP 搜索模式 2

默认组态：RP 搜索方向：负方向，RP 接近方向：正方向



默认组态：RP 搜索方向：正方向，RP 接近方向：正方向

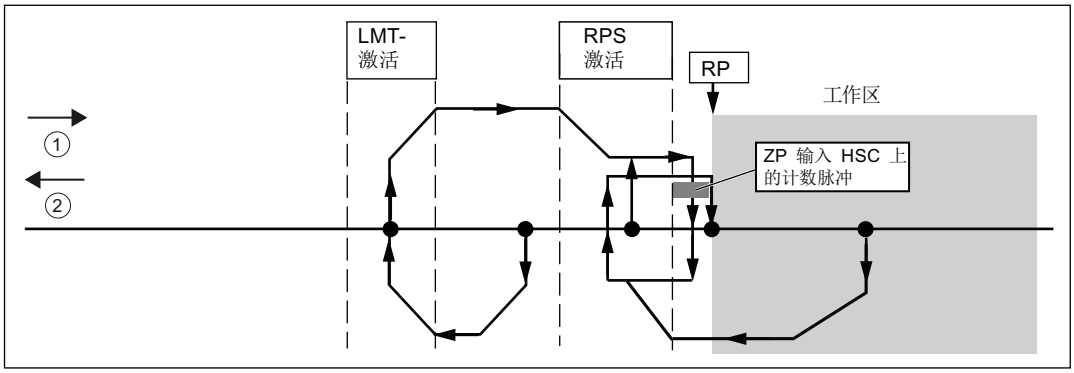


①: 正方向运动

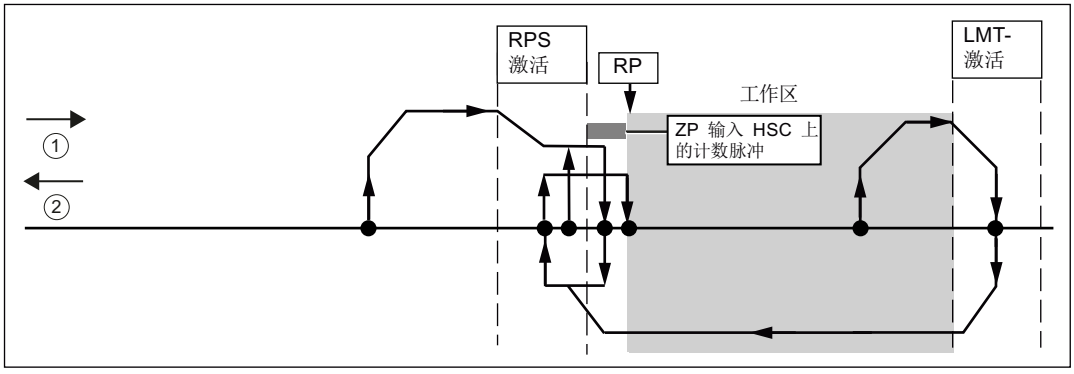
②: 负方向运动

RP 搜索模式 3

默认组态：RP 搜索方向：负方向，RP 接近方向：正方向



默认组态：RP 搜索方向：正方向，RP 接近方向：正方向

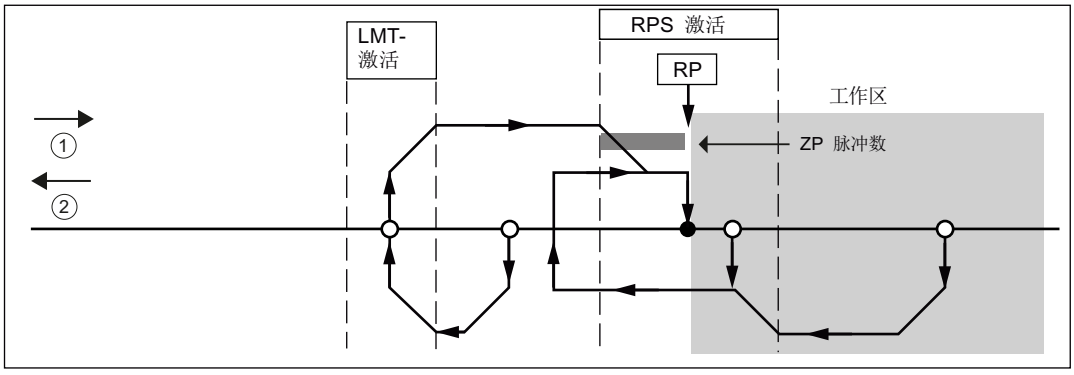


①: 正方向运动

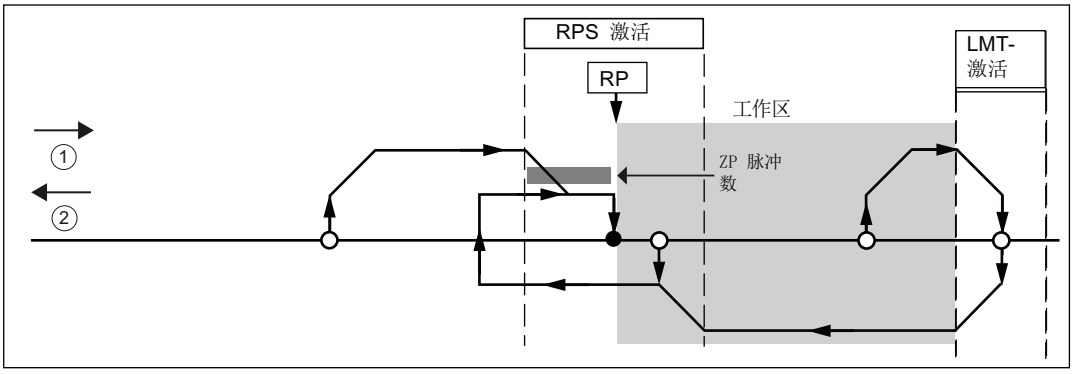
②: 负方向运动

RP 搜索模式 4

默认组态：RP 搜索方向：负方向，RP 接近方向：正方向



默认组态：RP 搜索方向：正方向，RP 接近方向：正方向



①: 正方向运动

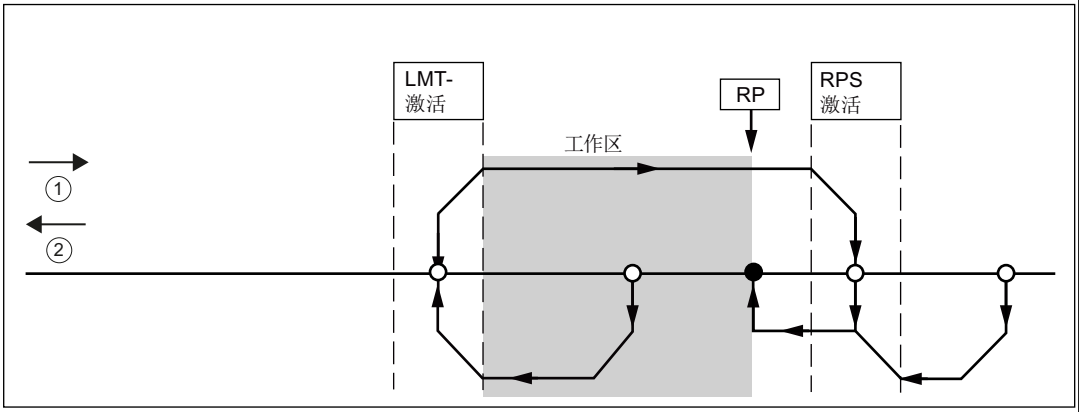
②: 负方向运动

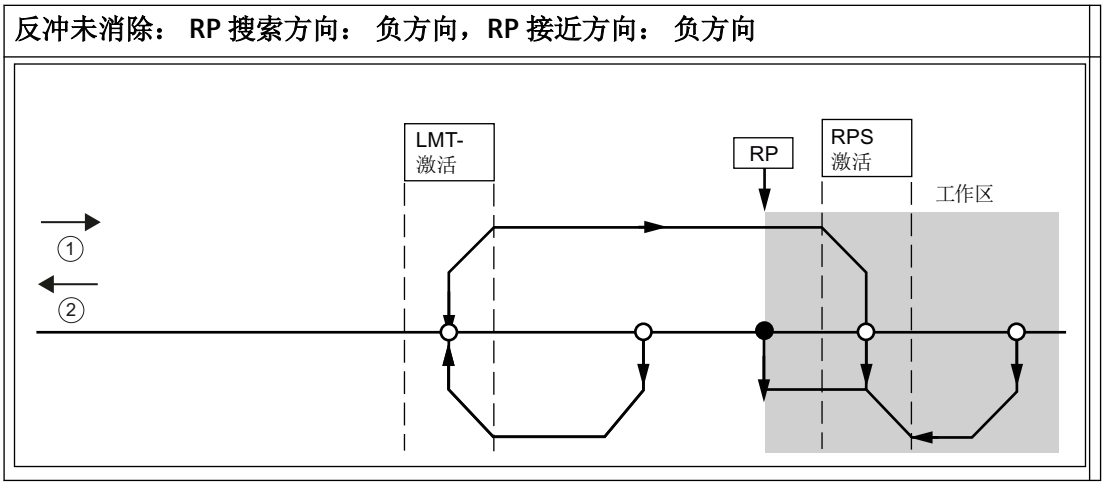
15.4.8.1 选择工作区位置以消除反冲

下图显示与消除反冲的接近方向的参考点 (RP)、RPS 激活区域和限位开关（LMT+ 和 LMT-）关联的工作区。 示图第二部分放置工作区，以便不消除反冲。 下图显示 RP 搜索模式 3。对于每个其他 RP 搜索模式的每个搜索序列，类似的工作区布置是可能的（尽管不建议）。

选择工作区位置以消除反冲

反冲已消除： RP 搜索方向： 负方向，RP 接近方向： 负方向





①: 正方向运动

② 负方向运动

15.5 使用运动轴组

15.5.1 为运动轴组编程

STEP 7-Micro/WIN SMART V3 提供了简单易用的工具来组态和编程运动轴组。只需遵循以下步骤即可：

1. 组态运动轴组：STEP 7-Micro/WIN SMART V3 提供了运动向导来组态轴组和运动路径。有关组态运动轴组的信息，请参见组态运动轴组 (页 1180)。
2. 创建由 CPU 执行的程序：运动向导会自动生成运动指令，可在程序中调用这些指令，以实现轴组功能。有关运动指令的信息，请参见运动向导为运动轴组创建的子例程 (页 1186)。在程序中调用以下指令：
   – 要启用运动轴组，插入 AXISx_CTRL 指令。使用 SM0.0（始终接通）可确保每次扫描时都会执行这条指令。
   – 要命令电机执行直线插补运动到达特定位置，使用 GRP0_2D_MOVELINEAR 或 GRP0_3D_MOVELINEAR 指令。
   – 要使电机按指定的运动路径运动，使用 GRP0_MOVEPATH 指令。
3. 编译程序并将系统块、数据块和程序块下载到 CPU。

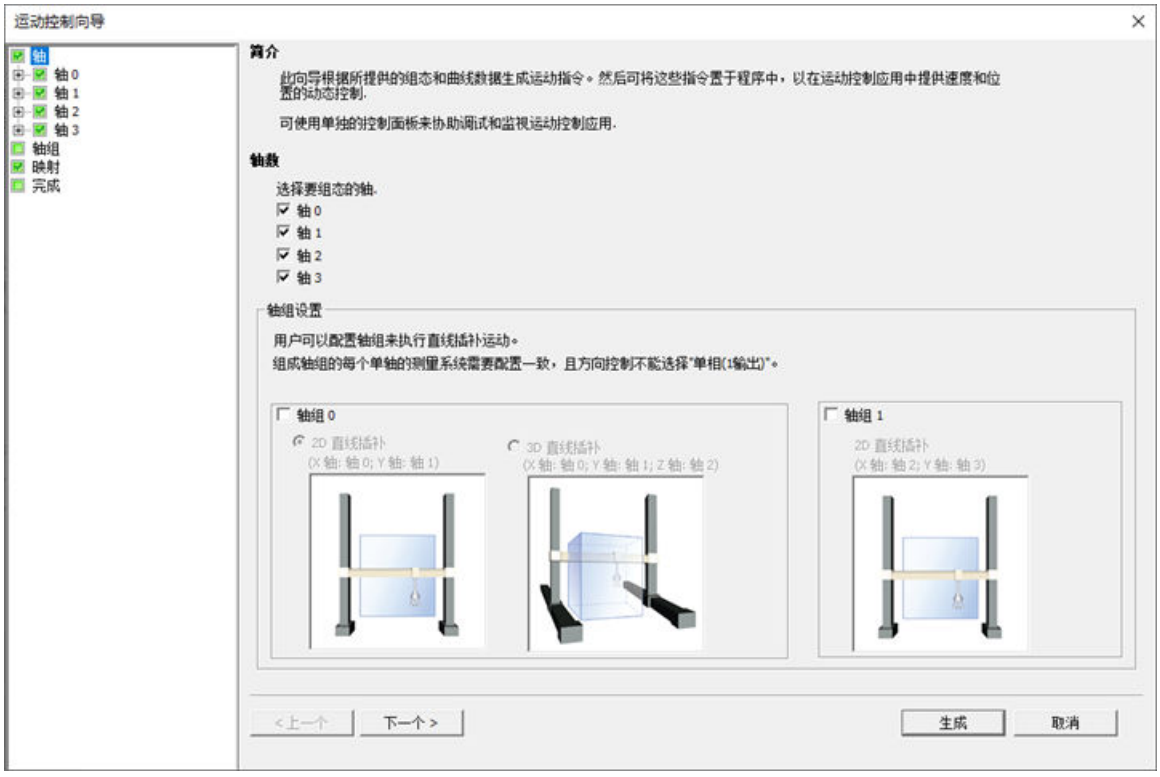
说明

确保测量系统的组态符合步进/伺服电机控制器系统的脉冲/转或距离/转的相关规范。

15.5.2 组态运动轴组

S7-200 SMART CPU 和 STEP 7-Micro/WIN SMART V3 为用户提供了运动向导来组态运动轴组。有关运动轴组态方法的信息，请参见“组态运动轴 (页 1106)”。

目前可为项目启用两个轴组。



启用轴组后，可重命名轴组，也可保留默认名称。下表显示了在组态轴组进行 2D 和 3D 直线插补运动时将启用的相应单轴。

直线插补运动	启用轴
2D 运动	• X 轴：轴 0 • Y 轴：轴 1
	• X 轴：轴 2 • Y 轴：轴 3
3D 运动	• X 轴：轴 0 • Y 轴：轴 1 • Z 轴：轴 2

说明

单轴的测量和相位

轴组中所有轴的测量系统和基本测量单位的组态需一致，方向控制的相位不能选择为“单相（1 输出）”(Single Phase (1 output))，否则轴组无法启用。

路径规划

STEP 7-Micro/WIN SMART V3 支持用户为运动控制轴组规划多段路径。

路径是预定义的空间曲线运动描述，轴组的刀具中心点 (TCP) 将沿该路径运动。路径由 2D/3D 轴组运动空间中的一系列连续位置和方向信息组成。

说明

在轴组的路径规划功能中，CPU 仅支持绝对位置直线插补运动。

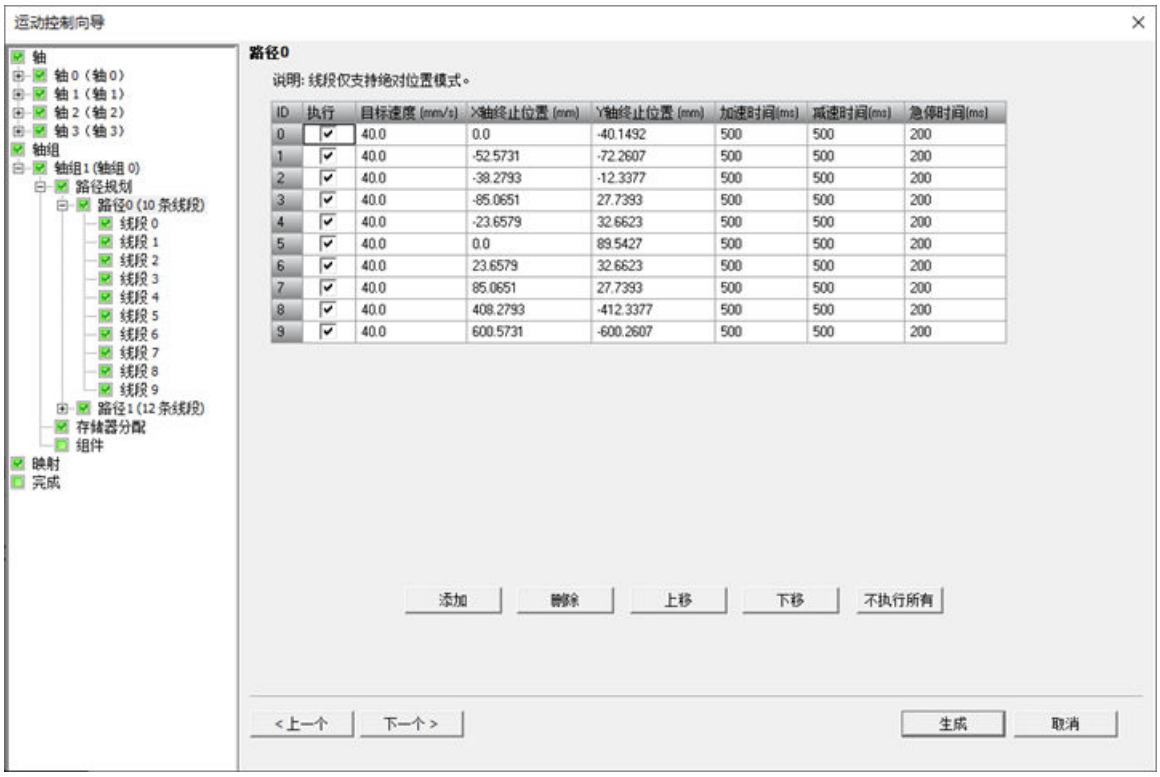
按照以下步骤规划路径：

1. 根据希望轴组运行的直线插补运动添加路径。可为每个路径创建一个名称并添加注释。最多可添加 128 个路径。

运动向导支持以下操作：

– 复制所选路径
– 通过在路径表中的上移或下移来更改所选路径的顺序

2. 为每个路径组态多个线段。在线段表中，需组态目标速度、X 轴终止位置、Y 轴终止位置、Z 轴终止位置（如果为 3D 直线插补运动启用了轴 2）、加速时间、减速时间和急停时间。



下表列出了线段输入的一些规定。如果输入不符合规定，会为输入加上红色下划线。

线段组态	输入规定
执行	可通过选中“执行”(Execution) 列中的复选框来执行线段的运动，或者通过取消选中“执行”(Execution) 列中的复选框不执行线段的运动。 **注：** • 必须选择线段 0。 • 必须连续选择线段。
目标速度	• 输入 > 0 • 如果在运动向导中组态的测量系统是相对脉冲，输入应为整数；如果组态的测量系统是工程单位，则输入应为浮点数
X/Y/Z 轴的终止位置	• 如果在运动向导中组态的测量系统是相对脉冲，输入应为整数；如果组态的测量系统是工程单位，则输入应为浮点数
加速时间	• 输入应为整数 • 20 ms ≤ 输入 ≤ 32000 ms
减速时间	• 输入应为整数 • 20 ms ≤ 输入 ≤ 32000 ms
急停时间	• 0 ms ≤ 输入 ≤ 32000 ms

运动向导支持通过以下操作组态线段：

– 使用快捷键 `Ctrl+C` 复制所选择的线段值，并使用快捷键 `Ctrl+V` 粘贴到所选行中
– 通过上移或下移更改所选线段的顺序
– 从 Excel 电子表格复制包含所有必需值的线段组态表，并在运动向导中使用快捷键 `Ctrl+V` 将其粘贴到线段组态表中。

说明

线段数量限值

- 路径中至少应创建 1 个线段，否则将弹出一条消息，提示用户删除不含线段的路径。
- 所有路径下的线段总数不得超过 128。

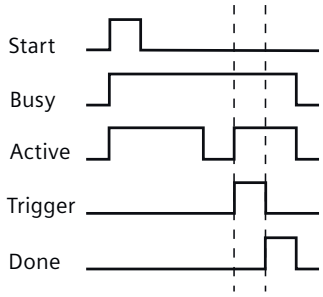
说明

从 Excel 粘贴到线段表中

- 线段表中的每个单元格最多支持输入 10 个字符。如果用户尝试粘贴的项超过 10 个字符，则仅会将前 10 个字符的内容粘贴到单元格中。
- 如果粘贴到线段中的项无效，将用红色下划线标出，提醒进行进一步检查。

---

3. 在线段组态页面中，组态每个线段的运动衔接方式和触发开始。
运动衔接方式
运动衔接方式定义功能块相对于上一个块的时间顺序。
– 缓冲模式：之前的线段运动完成后，下一个功能块将立即控制轴组。
– 打断模式：下一个功能块控制正在运动的线段减速直至停止，并控制新线段加速直至全部执行完成。
规划路径的第一个线段可选择“打断模式”(Aborting) 或“缓冲模式”(Buffered) 作为运动衔接方式；除第一个线段之外的其它线段，仅支持缓冲模式。
如果将路径的第一段组态为“正在中止”(Aborting)，则触发 GRP0_MOVEPATH 将导致 CPU 中止正在执行的直线插补运动，并开始执行所规划的路径运动。
如果将当前路径中的第一段组态为“缓冲模式”(Buffered)，则仅当上一个路径的最后两段正在执行或执行完成时，才能触发当前路径的执行。
触发开始
第一个线段之外的其它线段支持触发功能。
GRP0_MOVEPATH (页 1190) 子例程触发后，第一个线段将立即开始运行。
以下条件下，除第一个线段之外的其它线段开始运行：
– 如果当前线段没有组态为触发开始，则仅在之前的线段完成运行时，当前线段才开始运行。
– 如果当前线段组态为触发开始，则仅在之前的线段完成运行且当前线段出现触发上升沿时，当前线段才开始运行。
以下时序图举例说明了轴组运动中触发开始设置的输出状态。本示例中，该路径共组态了 3 个线段。前两个线段未组态为触发开始，而第三个线段组态为触发开始。



分配存储器

指定将组态置于数据块中时的起始位置。向导也可建议一个表示大小正确且未使用的 V 存储器块的地址。

生成组态

1. 组态运动轴组后，只需单击“生成”(Generate)。
   运动向导会执行以下任务：
   – 将轴组组态、路径和线段表插入 CPU 程序的系统块和数据块中
   – 为运动参数创建一个全局符号表
   – 将运动指令子例程添加到项目程序块，之后即可在应用中使用这些指令
2. 要修改任何组态或路径信息，可以再次打开运动向导。

说明

由于运动向导会对程序块、数据块和系统块进行更改，因此必须将这三种块都下载到 CPU 中。否则，运动轴组将无法得到正常操作所需的所有程序组件。

动态更改符号表中定义的线段参数

可更改在符号表中定义的每个线段参数。

可动态更改运动路径线段的参数，如下所示：

1. 为符号表中每个线段和路径的以下参数创建符号：

线段参数	符号
执行	AXESGRPx_Px_Sx_EXE
轴 X 的结束位置	AXESGRPx_Px_Sx_XPOS
轴 Y 的结束位置	AXESGRPx_Px_Sx_YPOS
轴 Z 的结束位置	AXESGRPx_Px_Sx_ZPOS
目标速度	AXESGRPx_Px_Sx_SPD
加速时间	AXESGRPx_Px_Sx_ACC
减速时间	AXESGRPx_Px_Sx_DEC
急停时间	AXESGRPx_Px_Sx_JERK

2. 根据需要更改每个符号的值。

说明

多个线段的规则

- 如果禁用连续线段中的一个线段，则已禁用的线段和后面的线段将不会运行。
- 如果一个线段正在运行或已完成运行，并且修改了该线段的位 “执行”(Execution) 的值，则修改将在下次运行路径时生效。

15.5.3 运动向导为运动轴组创建的子程序

共有 4 个运动轴组子例程。

运动控制子例程	描述
GRPx_2/3D_MOVELINEAR (页 1187)	命令轴组上的 2D/3D 直线插补运动从实际位置运动到目标绝对位置或相对位置
GRPx_MOVEPATH (页 1190)	命令轴组按照多线段组态中所指定的路径进行运动。
GRPx_RESET (页 1192)	命令轴组从 ErrorStop 状态转换为准备进行直线插补运动的状态。
GRPx_STOP (页 1193)	命令轴组停止运动。

15.5.3.1 运动控制子例程使用准则

使用运动轴组子例程时的一些准则如下：

- 可同时激活多个运动轴组子例程。CPU 最多可缓冲 32 个直线插补运动子例程。
- 将 AXISx_CTRL 子例程插入程序中，以启用和初始化轴组中的运动轴。
- 使用 GRP0_2/3D_MOVELINEAR 或 GRP0_MOVEPATH 命令 CPU 执行轴组运动。
- 如果为轴组中的单轴触发 LMT+/LMT-/STP 信号，或 AXISx_CTRL 指令中的 MOD_EN 参数关闭，正在运行的轴组直线插补运动将立即停止或减速停止。
- 如果 CPU 处于绝对模式，触发轴组运动之前，需要使用 AXISx_RSEEK/AXISx_LDPOS 为轴组建立零点位置。
- 使用 GRP0_RESET 子例程命令轴组从 ErrorStop 状态 (页 1192)转换为准备进行直线插补运动的状态。

15.5.3.2 GRPx_2/3D_MOVELINEAR 子例程

表格 15-50 GRPx_2/3D_MOVELINEAR

LAD/FBD	STL	描述
GRP0_3D_MOV~ EN START Segme~ Done Pos_X Busy Pos_Y Active Pos_Z Error Speed Mode Acc_Ti~ Decel_~ Jerk_Ti~ Buffer ~	**CALL GRPx_2D_MOVELINEAR, GRPx_3D_MOVELINEAR**	GRPx_3D_MOVELINEAR 子例程命令组态为进行直线插补运动的轴组从 TCP（刀具中心点）的实际位置运动到目标绝对位置或相对位置。 对于组态了 2 个单轴的轴组，CPU 将生成 GRPx_2D_MOVELINEAR 子例程指令。

表格 15-51 GRPx_2/3D_MOVELINEAR 子例程的参数

输入/输出	数据类型	操作数	描述
EN	BOOL	I、Q、V、M、SM、S、T、C、L	启用子例程。确保 EN 位保持开启，直至 Done 位指示子例程执行已经完成。
START	BOOL	I、Q、V、M、SM、S、T、C、L、能流	向 CPU 发送 GRPx_2/3D _MOVELINEAR 命令。
Segment_ID	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD、常数	用户为每个函数块指定的唯一参数，表示规划路径中的相应线段。
Pos_X、Pos_Y、Pos_Z	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD、常数	为该函数块设置的目标 X/Y/Z 位置参数。根据所选的测量单位，该值是脉冲数/每秒 (DINT) 或工程单位数/每秒 (REAL)。只有 3D 直线插补运动存在 Pos_Z 参数。
Speed	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD、常数	指定运动的最大复合速度，但不要求必须达到该速度。根据所选的测量单位，该值是脉冲数/每秒 (DINT) 或工程单位数/每秒 (REAL)。
Mode	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD、常数	指定运动模式： • 0：绝对模式 • 1：相对模式
Acc_Time	DINT	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD、常数	指定运动的加速时间 (ms)。
Decel_Time	DINT	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD、常数	指定运动的减速时间 (ms)。
Jerk_Time	DINT	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD、常数	指定运动的急停时间 (ms)。
BufferMode	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD、常数	定义函数块相对于上一个块的时间顺序。 可选择的 BufferMode： • 0：打断模式 • 1：缓冲模式
Done	BOOL	I、Q、V、M、SM、S、T、C、L	命令执行已成功完成时，参数启用。
Busy	BOOL	I、Q、V、M、SM、S、T、C、L	功能块触发但尚未完成时，参数启用。

输入/输出	数据类型	操作数	描述
Active	BOOL	I、Q、V、M、SM、S、T、C、L	函数块输出脉冲时，参数启动。
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD	指令执行时发生某些错误。

说明

START 参数

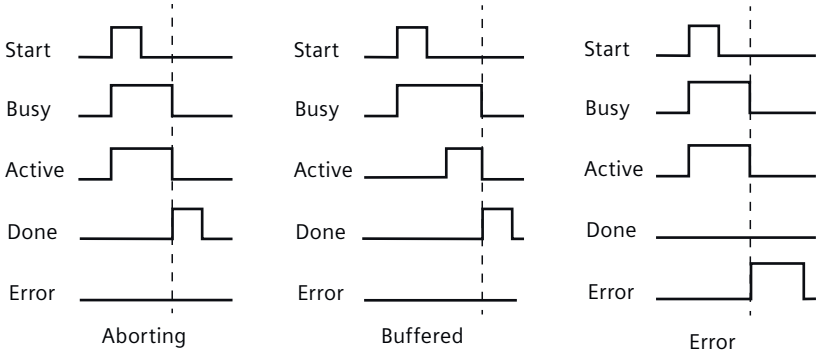
使用沿信号作为执行直线插补运动子例程的触发信号，确保同一直线插补运动子例程仅触发一次。

说明

Acc_Time/Decel_Time/Jerk_Time 参数

在该子例程中组态的时间相关参数适用于轴组直线插补运动，而在运动向导中为每个单轴组态的时间相关参数不适用于轴组直线插补运动。

轴组中的“完成”(Done)、“激活”(Active) 和“忙”(Busy) 输出状态显示如下：



Segment_ID

Segment_ID 参数可帮助用户形成具体的运动路径，表示运动路径中的每个对应的线段。使用 Segment_ID 时，需注意以下提示：

- Segment_ID 不表示线段执行顺序。
- Segment_ID 的范围是 0 到 127。
- Segment_ID 在同一程序中的不同运动线性子例程中是可以重复的。
  在一些应用场景中，需要编写包含多个子进程的复杂进程。为了显示子进程的执行顺序，可按数字顺序在子进程指令中命名 Segment_ID。在这种情况下，不同子进程中的 Segment_ID 是可以重复的。
  如果 CPU 在程序中缓冲多个 Segment_ID 相同的线段，这些重复的 Segment_ID 会共用相同的运行状态，即“完成”(Done)、“激活”(Active) 和“忙”(Busy)、“错误”(Error) 状态。

15.5.3.3 GRPx_MOVEPATH 子例程

表格 15-52 GRPx_MOVEPATH

LAD/FBD	STL	描述
	`CALL GRPx_MOVEPATH`	GRPx_MOVEPATH 子例程命令轴组按照多线段组态中指定的路径进行运动。

表格 15-53 GRPx_MOVEPATH 子例程的参数

输入/输出	数据类型	操作数	描述
EN	BOOL	I、Q、V、M、SM、S、T、C、L	启用子例程。确保 EN 位保持开启，直至 Done 位指示子例程执行已经完成。
START	BOOL	I、Q、V、M、SM、S、T、C、L、能流	向 CPU 发送 GRPx_MOVEPATH 命令。

输入/输出	数据类型	操作数	描述
Path_ID	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD、常数	指定路径 ID。ID 输入编号必须在 0 到 127 之间。否则子例程会返回错误
Done	BOOL	I、Q、V、M、SM、S、T、C、L	命令执行已成功完成时，参数启用。
Busy	BOOL	I、Q、V、M、SM、S、T、C、L	功能块触发但尚未完成时，参数启用。
Active	BOOL	I、Q、V、M、SM、S、T、C、L	函数块输出脉冲时，参数启动。
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD	指令执行时发生某些错误。
Err_Segment	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD	执行发生执行错误的线段的 ID。
Cur_Segment	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD	读取当前执行的线段的 ID

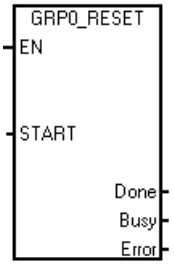
说明

START 参数

使用沿信号作为执行直线插补运动子例程的触发信号，确保同一直线插补运动子例程仅触发一次。

15.5.3.4 GRPx_RESET 子例程

表格 15-54 GRPx_MOVEPATH

LAD/FBD	STL	描述
	**CALL GRPx_RESET**	GRPx_RESET 子例程命令轴组从 ErrorStop 状态转换为准备好进行直线插补运动的状态。

表格 15-55 GRPx_RESET 子例程的参数

输入/输出	数据类型	操作数	描述
EN	BOOL	I、Q、V、M、SM、S、T、C、L	启用子例程。确保 EN 位保持开启，直至 Done 位指示子例程执行已经完成。
START	BOOL	I、Q、V、M、SM、S、T、C、L、能流	向 CPU 发送 GRPx_RESET 命令。
Done	BOOL	I、Q、V、M、SM、S、T、C、L	命令执行已成功完成时，参数启用。
Busy	BOOL	I、Q、V、M、SM、S、T、C、L	功能块触发但尚未完成时，参数启用。
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD	指令执行时发生某些错误。

说明

START 参数

使用沿信号作为执行直线插补运动子例程的触发信号，确保同一直线插补运动子例程仅触发一次。

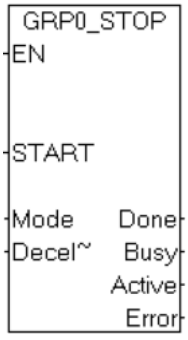
轴组处于 ErrorStop 状态

调用 GRPx_RESET 指令不影响其它 FB 实例的输出。只有在轴组处于 ErrorStop 状态时，GRPx_RESET 指令才会生效。如果调用该 FB 时轴组未处于 ErrorStop 状态，该 FB 将提示出错。以下情况下，轴组停止后，轴组会进入 ErrorStop 状态：

- 触发轴组直线插补运动指令（GRPx_2/3D_MOVELINEAR 或 GRPx_MOVEPATH）时使用的参数无效，例如操作模式无效、速度无效、加速时间无效等。
- 轴组直线插补运动期间出现 LMT+/LMT-/STP 信号。
- 在以下无效条件下触发轴组直线插补运动：
  - 在绝对模式下触发轴组直线插补运动，但未设置零点位置。
  - 超出运动缓冲限值 32 时缓冲轴组直线插补运动。
  - 运动缓冲队列忙时触发 GRPx_2/3D_MOVELINEAR 指令。
    GRPx_MOVEPATH 指令执行时运动缓冲队列忙，同时该路径中部分线段未添加到运动队列中。
  - 轴组中有一部分轴未处于就绪状态。
  - 轴组中至少有一个轴的组态无效。

15.5.3.5 GRPx_STOP 子例程

表格 15-56 GRPx_STOP

LAD/FBD	STL	描述
	**Call GRPx_STOP, START, Mode, Decel_Time, Done, Busy, Active, Error**	**GRPx_STOP** 子例程命令轴组停止运动。

表格 15-57 GRPx_STOP 子例程的参数

输入/输出	数据类型	操作数	描述
EN	BOOL	I、Q、V、M、SM、S、T、C、L	启用子例程。确保 EN 位保持开启，直至 Done 位指示子例程执行已经完成。
START	BOOL	I、Q、V、M、SM、S、T、C、L、能流	向 CPU 发送 GRPx_STOP 命令。
Mode	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD、常数	指定轴组的停止模式： • 0：减速停止 轴组运动在特定的减速时间内减速停止。 • 1：立即停止 轴组运动立即停止。
Decel_Time	DINT	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD、常数	指定停止运动的减速时间 (ms)。 **注**：减速时间，范围为 20 ms 到 32000 ms。 **注**：如果选择模式 0，则需要组态参数 Decel_Time。
Done	BOOL	I、Q、V、M、SM、S、T、C、L	命令执行已成功完成时，参数启用。
Busy	BOOL	I、Q、V、M、SM、S、T、C、L	功能块触发但尚未完成时，参数启用。
Active	BOOL	I、Q、V、M、SM、S、T、C、L	函数块停止轴组运动时，参数启动。
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD	指令执行时发生某些错误。

说明

如果轴组组态有 LMT+/LMT-/STP 输入信号或正在执行 GRPx_STOP 指令，正在进行的减速停止将被立即停止命令中断。

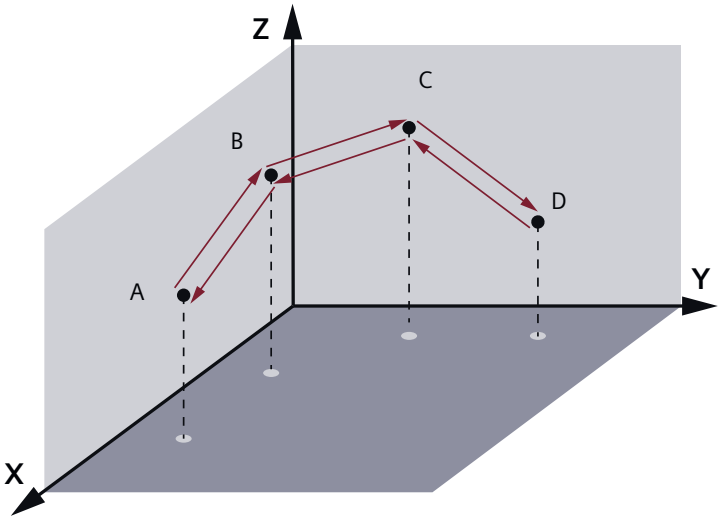
说明

触发 GRPx_STOP 命令后，轴组停止在当前运动所在的直线上。

15.5.4 运动轴组示例程序

示例程序显示使用 AXISx_CTRL, AXISx_RSEEK 和 GRPx_3D_MOVELINEAR 子例程执行取放操作的简单相对运动。该程序需要通过 RP 搜索轴组中的轴，移动距离可采用工程单位或相对脉冲进行测量。I0.0（启动）打开时，会启动机器。I0.2（轴 X 的 E_Stop）、或 I0.3（轴 Y 的 E_Stop）、或 I0.4（轴 Z 的 E_Stop）打开时，机器将打断所有运动并减速停止。

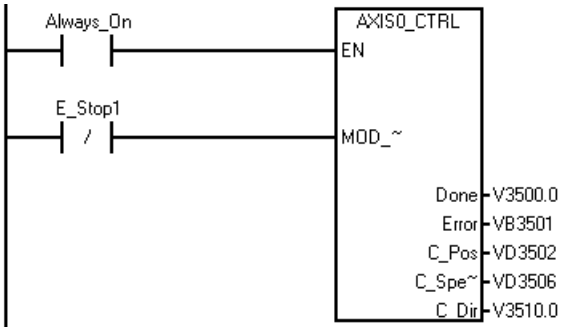
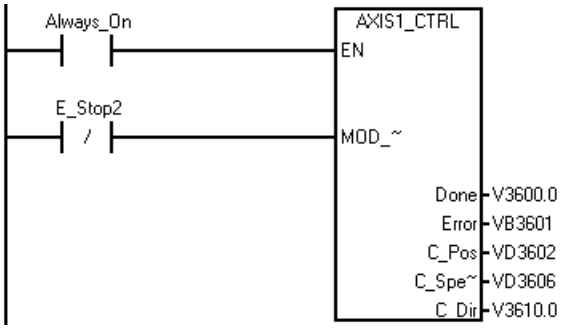
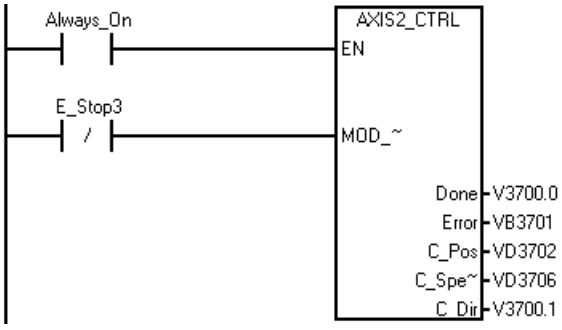
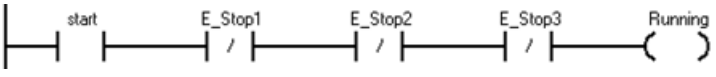
以下是取放机器夹具的运动路径：

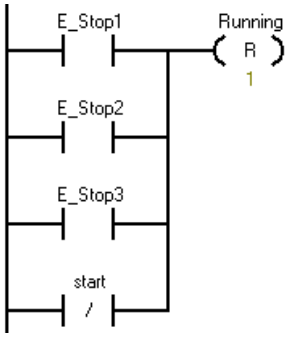
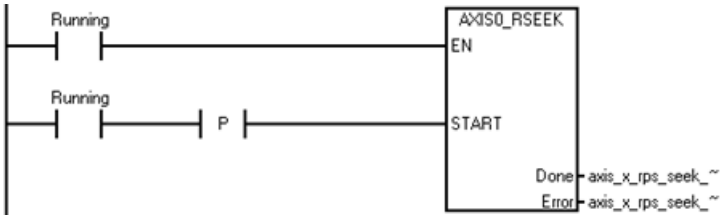
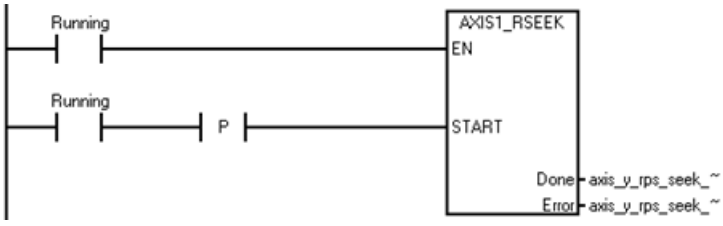
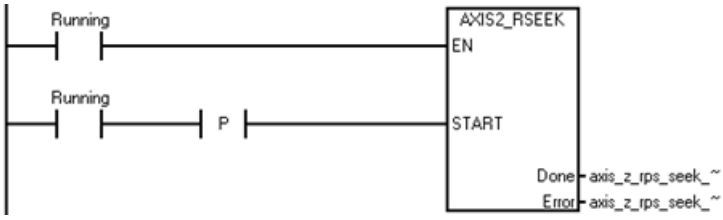


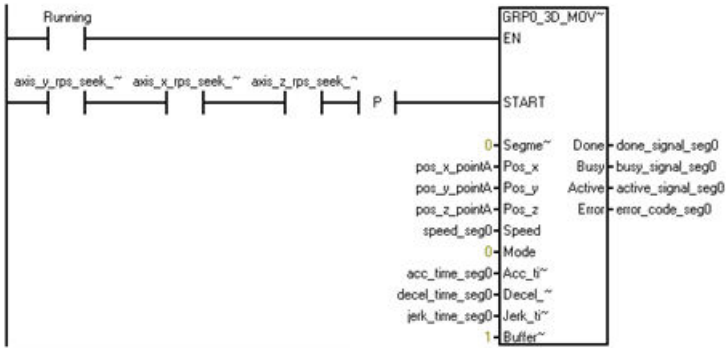
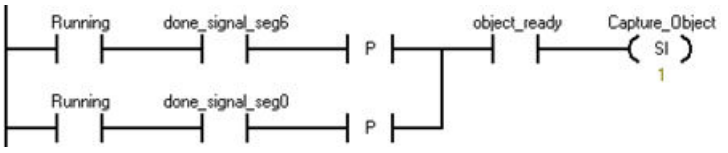
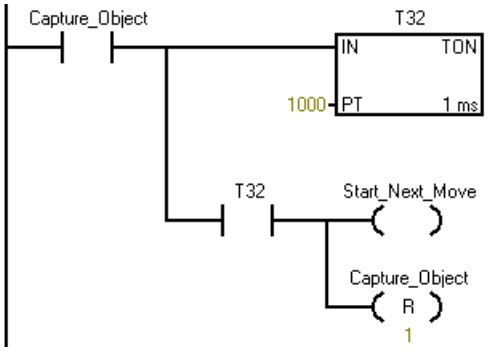
机器启动时，系统会先搜索零点位置，夹具运动到拾取位置 A。系统等待指示拾取物体已就绪的信号 I0.1。打开 Q1.1（时长 1 秒），夹具拾取物体，并沿轨迹 1、2 和 3 运动到位置 D。

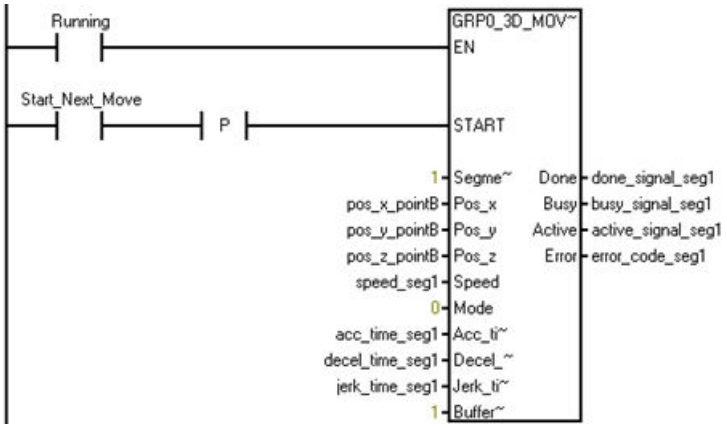
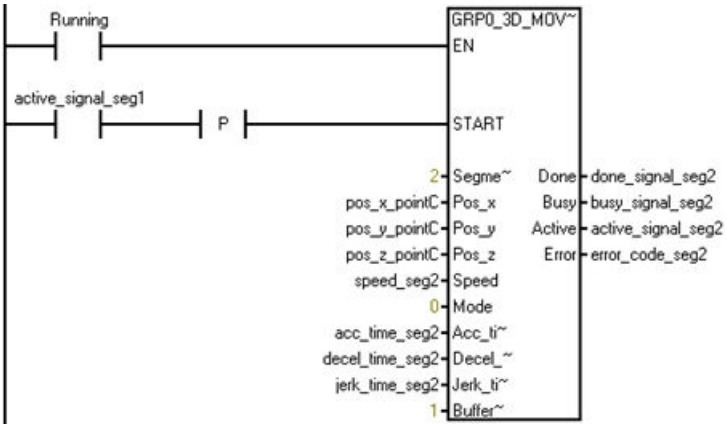
当夹具到达位置 D 时，打开 Q1.2（时长 1 秒），夹具释放物体。夹具沿轨迹 4、5 和 6 移动回位置 A，并等待下一次拾取。

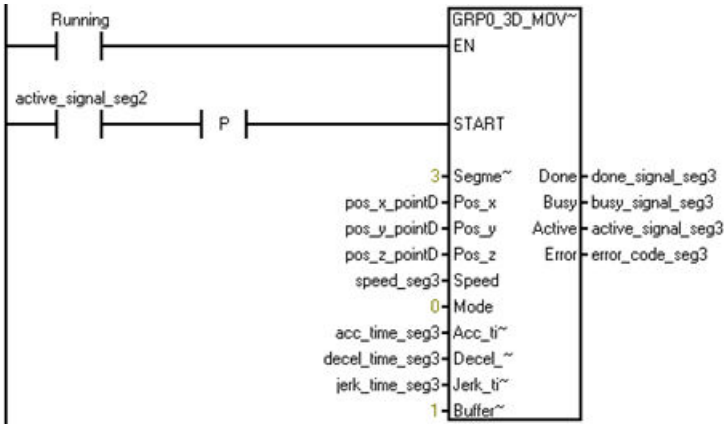
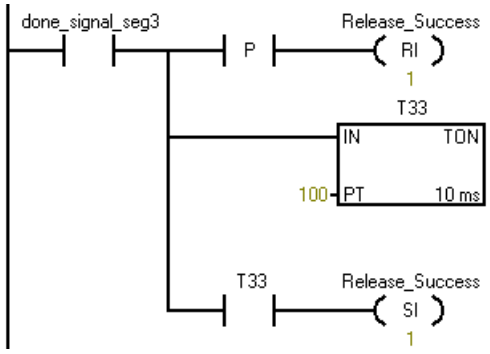
表格 15-58　示例：运动轴组简单绝对运动（取放应用）

LAD/FBD	描述	STL
程序段 1：  	初始化轴 X。	**LD Always_On** **= L60.0** **LDN E_Stop1** **= L63.7** **LD L60.0** **CALL AXIS0_CTRL, L63.7, V3500.0, VB3501, VD3502, VD3506, V3510.0**
程序段 2：  	初始化轴 Y。	**LD Always_On** **= L60.0** **LDN E_Stop2** **= L63.7** **LD L60.0** **CALL AXIS1_CTRL, L63.7, V3600.0, VB3601, VD3602, VD3606, V3610.0**
程序段 3：  	初始化轴 Z。	**LD Always_On** **= L60.0** **LDN E_Stop3** **= L63.7** **LD L60.0** **CALL AXIS2_CTRL, L63.7, V3700.0, VB3701, VD3702, VD3706, V3700.1**
程序段 4：  	启动信号为 ON，停止信号为 OFF 时，系统准备好运行。	**LD start** **AN E_Stop1** **AN E_Stop2** **AN E_Stops3** **= Running**

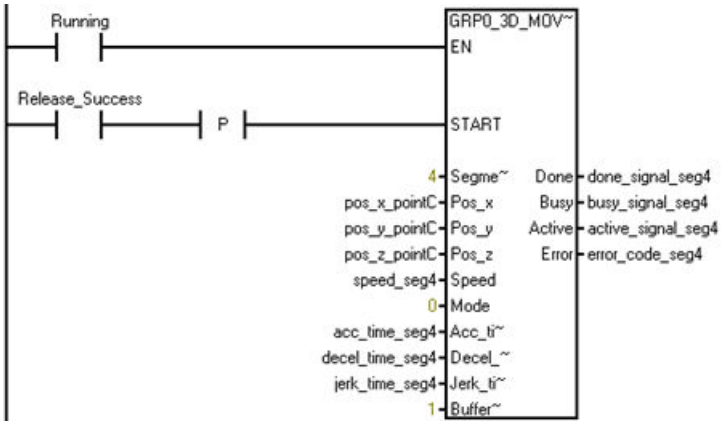
LAD/FBD	描述	STL
程序段 5：  	启动信号为 OFF，停止信号为 ON 时，系统停止运行。	LD start O E_Stop1 O E_Stop2 ON start = Running, 1
程序段 6：  	启动信号为 ON 时，系统开始搜索零点位置。该程序段用于使轴 X 搜索零点位置。	LD Running = L60.0 EU = L63.7 LD L60.0 CALL AXIS0_RSEEK, L63.7, axis_x_rps_seek_done, axis_x_rps_seek_error
程序段 7：  	启动信号为 ON 时，系统开始搜索零点位置。该程序段用于使轴 Y 搜索零点位置。	LD Running = L60.0 EU = L63.7 LD L60.0 CALL AXIS1_RSEEK, L63.7, axis_y_rps_seek_done, axis_y_rps_seek_error
程序段 8：  	启动信号为 ON 时，系统开始搜索零点位置。该程序段用于使轴 Z 搜索零点位置。	LD Running = L60.0 EU = L63.7 LD L60.0 CALL AXIS0_RSEEK, L63.7, axis_zx_rps_seek_done, axis_z_rps_seek_error

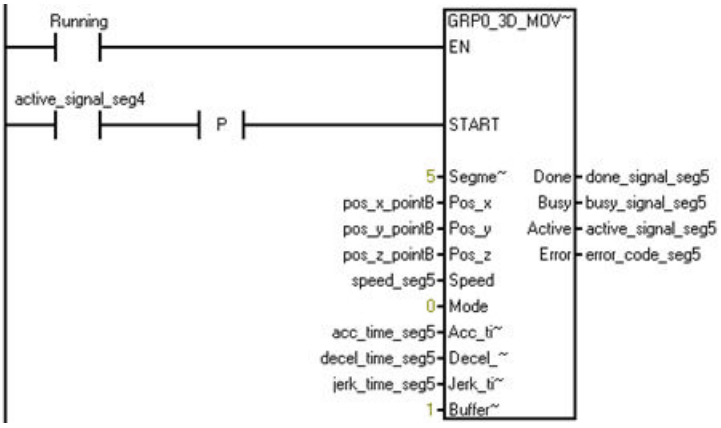
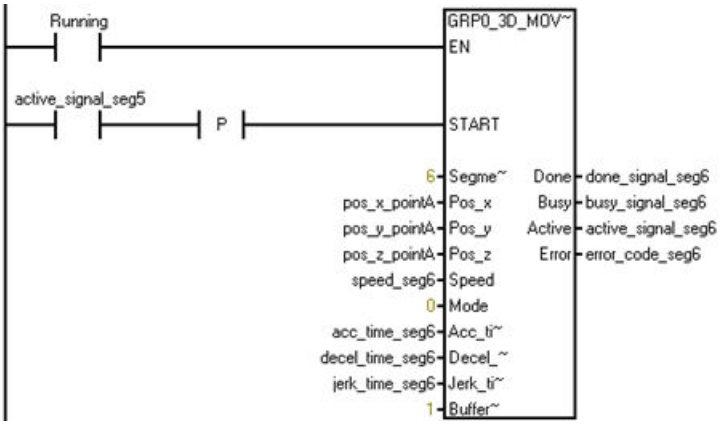
LAD/FBD	描述	STL
程序段 9：  	轴 X、轴 Y 和轴 Z 返回零点位置后，夹具会移动到超过点 A 的位置。	LD Running = L60.0 LD axis_y_rps_seek_done A axis_x_rps_seek_done A axis_z_rps_seek_done EU = L63.7 LD L60.0 CALL GRP0_3D_MOVELINEAR, L63.7, 0, pos_x_pointA, pos_y_pointA, pos_z_pointA, speed_seg0, 0, acc_time_seg0, decel_time_seg0, jerk_time_seg0, 1, done_signal_seg0, busy_signal_seg0, active_signal_seg0, error_code_seg0
程序段 10：  	拾取对象就绪后，拾取操作开始。	LD Running A done_signal_seg6 EU LD Running A done_signal_seg0 EU OLD A object_ready SI Capture_Object, 1
程序段 11：  	1 秒后拾取结束。	LD Capture_Object TON T32, 1000 A T32 = Start_Next_Move R Capture_Object, 1

LAD/FBD	描述	STL
程序段 12：  	拾取完成后，夹具从 3D 空间中的 A 点移动到 B 点。运动路径编号等于 Segment_ID，即 1。	LD Running = L60.0 LD Start_Next_Move EU = L63.7 LD L60.0 CALL GRP0_3D_MOVELINEAR, L63.7, 1, pos_x_pointB, pos_y_pointB, pos_z_pointB, speed_seg1, 0, acc_time_seg1, decel_time_seg1, jerk_time_seg1, 1, done_signal_seg1, busy_signal_seg1, active_signal_seg1, error_code_seg1
程序段 13：  	夹具从 3D 空间中的 B 点移动到 C 点。运动路径编号等于 Segment_ID，即 2。	LD Running = L60.0 LD active_signal_seg1 EU = L63.7 LD L60.0 CALL GRP0_3D_MOVELINEAR, L63.7, 2, pos_x_pointC, pos_y_pointC, pos_z_pointC, speed_seg2, 0, acc_time_seg2, decel_time_seg2, jerk_time_seg2, 1, done_signal_seg2, busy_signal_seg2, active_signal_seg2, error_code_seg2

LAD/FBD	描述	STL
程序段 14：  	夹具从 3D 空间中的 C 点移动到 D 点。运动路径编号等于 Segment_ID，即 3。	LD Running = L60.0 LD active_signal_seg2 EU = L63.7 LD L60.0 CALL GRP0_3D_MOVELINEAR, L63.7, 3, pos_x_pointD, pos_y_pointD, pos_z_pointD, speed_seg3, 0, acc_time_seg3, decel_time_seg3, jerk_time_seg3, 1, done_signal_seg3, busy_signal_seg3, active_signal_seg3, error_code_seg3
程序段 15：  	夹具开始放置拾取的物体。1 秒后放置结束。	LD done_signal_seg3 LPS EU RI Release_Success, 1 LRD TON T33, 100 LPP A T33 SI Release_Success, 1

LAD/FBD	描述	STL

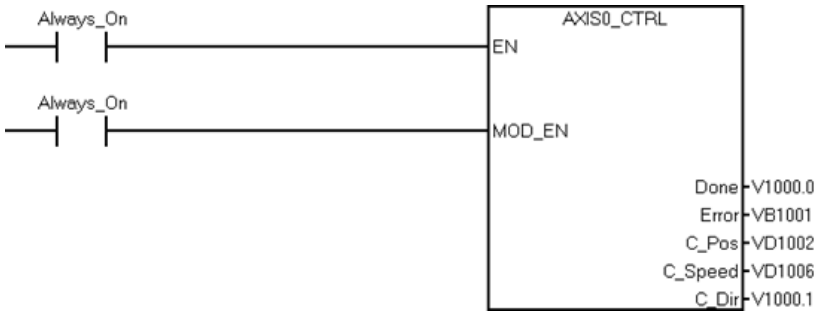
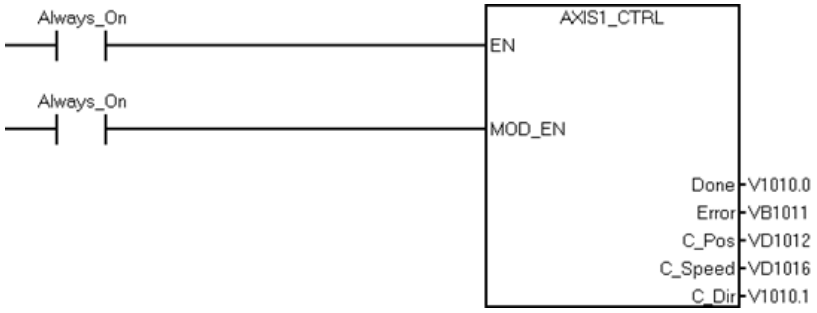
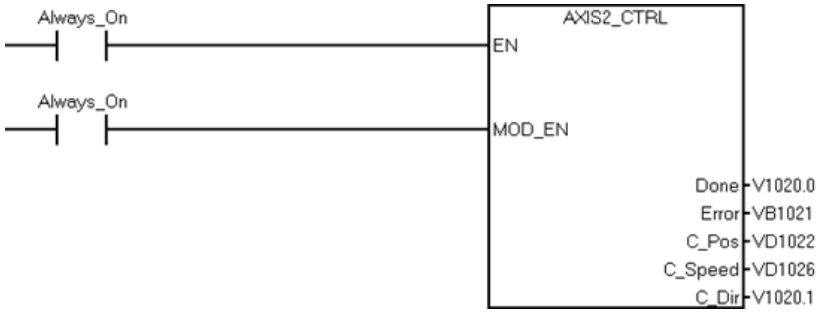
程序段 16：
 
 	
	放置完成后，夹具从 3D 空间中的 D 点移动到 C 点。运动路径编号等于 Segment_ID，即 4。	`LD Running`
		`= L60.0`
		`LD Release_Success`
		`EU`
		`= L63.7`
		`LD L60.0`
		`CALL GRP0_3D_MOVELINEAR, L63.7, 4, pos_x_pointC, pos_y_pointC, pos_z_pointC, speed_seg4, 0, acc_time_seg4, decel_time_seg4, jerk_time_seg4, 1, done_signal_seg4, busy_signal_seg4, active_signal_seg4, error_code_seg4`

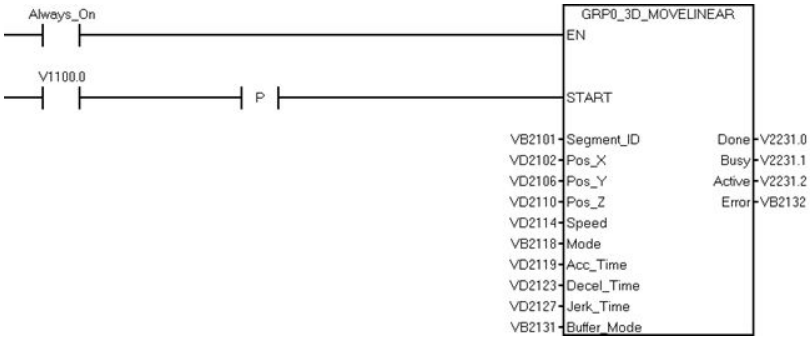
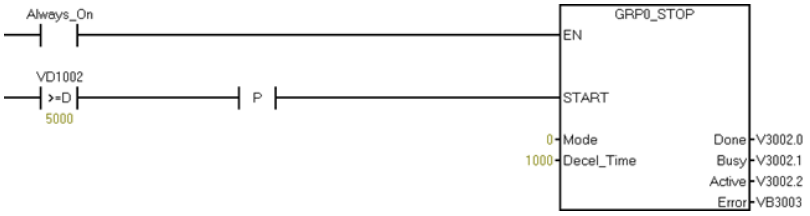
LAD/FBD	描述	STL
程序段 17：  	夹具从 3D 空间中的 C 点移动到 B 点。运动路径编号等于 Segment_ID，即 5。	LD Running = L60.0 LD active_signal_seg4 EU = L63.7 LD L60.0 CALL GRP0_3D_MOVELINEAR, L63.7, 5, pos_x_pointB, pos_y_pointB, pos_z_pointB, speed_seg5, 0, acc_time_seg5, decel_time_seg5, jerk_time_seg5, 1, done_signal_seg5, busy_signal_seg5, active_signal_seg5, error_code_seg5
程序段 18：  	夹具从3D空间中的B点移动到 A 点，并准备好执行下一次拾取操作。运动路径编号等于 Segment_ID，即 6。	LD Running = L60.0 LD active_signal_seg5 EU = L63.7 LD L60.0 CALL GRP0_3D_MOVELINEAR, L63.7, 6, pos_x_pointA, pos_y_pointA, pos_z_pointA, speed_seg6, 0, acc_time_seg6, decel_time_seg6, jerk_time_seg6, 1, done_signal_seg6, busy_signal_seg6, active_signal_seg6, error_code_seg6

15.5.4.1 使用 GPRx_STOP 子例程停止轴组运动的示例

示例程序显示了如何使用 GPR0_STOP 子例程来停止轴组运动。

表格 15-59 轴组停止示例

LAD/FBD	描述	STL
程序段 1：  	使能轴 0。	**LD Always_On:SM0.0** **= L60.0** **LD Always_On:SM0.0** **= L63.7** **LD L60.0** **CALL AXIS0_CTRL:SBR1, L63.7, axis0_ctl_done:V1000.0, VB1001, x_position:VD1002, VD1006, V1000.1**
程序段 2：  	使能轴 1。	**LD Always_On:SM0.0** **= L60.0** **LD Always_On:SM0.0** **= L63.7** **LD L60.0** **CALL AXIS1_CTRL:SBR12, L63.7, V1010.0, VB1011, VD1012, VD1016, V1010.1**
程序段 3：  	使能轴 2。	**LD Always_On:SM0.0** **= L60.0** **LD Always_On:SM0.0** **= L63.7** **LD L60.0** **CALL AXIS2_CTRL:SBR23, L63.7, V1020.0, VB1021, VD1022, VD1026, V1020.1**

LAD/FBD	描述	STL
程序段 4：  	触发轴组的 3D 线性运动。	**LD Always_On:SM0.0** **= L60.0** **LD start_movement:V1100.0** **EU** **= L63.7** **LD L60.0** **CALL GRP0_3D_MOVELINEAR:SBR34, L63.7, segment:VB2101, Pos_x:VD2102, Pos_y:VD2106, Pos_z:VD2110, speed:VD2114, mode:VB2118, acc:VD2119, decel:VD2123, jerk:VD2127, buffer_mode:VB2131, done:V2231.0, busy:V2231.1, active:V2231.2, error:VB2132**
程序段 5：  	X 轴移动 5000 个脉冲时，命令 GPRO-STOP 会触发减速停止。减速时间为 1000 ms。	**LD Always_On:SM0.0** **= L60.0** **LDD>= x_position:VD1002, 5000** **EU** **= L63.7** **LD L60.0** **CALL GRP0_STOP:SBR36, L63.7, 0, 1000, stp_done:V3002.0, stp_busy:V3002.1, stp_active:V3002.2, stp_error:VB3003**

15.6 停止轴组运动

可通过以下三种方法停止正在运行的轴组：

- 如果为轴组中的任一轴组态了 LMT+/LMT-/STP (页 1106) 输入信号，当信号到达时，根据组态，正在运行的轴组将立即停止，或沿规划的运动路径减速至停止。
- 用户也可以禁用 AXISx_CTRL 指令中的 MOD_EN 参数 (页 1123)，打断正在执行的轴组运动指令。
- 使用 GRPx_STOP (页 1193)子例程。

15.7 运动轴组错误代码（SMW820 或 SMW890 的 WORD）

如果组态下载到 CPU 后轴组组态中存在任何错误，对于轴组 0，错误代码将出现在 SMW820 WORD 中；对于轴组 1，错误代码将出现在 WORD SMB890 中。错误代码及其含义如下：

错误代码	描述
43	轴组组态块指针错误
44	轴组组态块大小错误
45	轴组的 SBD0 记录大小无效
46	轴组的 SDB0 格式非法
47	轴组的维度非法
48	轴组维度中的轴数非法
49	轴组运动模型非法
50	轴组中包含的轴不存在
51	组态块指针错误
52	轴组态重复
53	轴组不受该 CPU 类型的支持
54 到 127	预留

15.8 运动轴组指令的错误代码（SMB823 或 SMB893 的七个 LS 位）

如果 CPU 在直线插补运动期间检测到已触发的运动轴组中存在参数错误，相应的错误代码会出现在错误参数中。随后，该运动轴组将立即打断。由于运动打断，正在运行的指令和缓冲模式的指令将显示错误代码 66。

错误代码	描述
50	轴组运动指令由用户打断
51	轴组组态无效
52	由于正在执行路径指令，无法执行轴组指令。
53	非法操作模式
54	运动衔接方式非法
55	速度值非法
56	加速时间值非法
57	减速时间值非法
58	急停时间值非法
59	在绝对模式下运行时，轴组中有一部分轴未建立零点
60	轴组运动距离非法
61	不支持运动衔接方式
62	距离不足，无法执行指定的加速/减速
63	缓冲的直线插补运动已超过支持的最大缓冲数 32。
64	轴组中有一部分轴未处于就绪状态
65	轴组速度超出范围
66	由于轴组指令参数出错而打断
67	预留
68	线段 ID 超出范围 (0-127)
69	路径 ID 未组态或超出范围 (0-127)
70	由于单轴上的 STOP 信号已激活，轴组无法运行
71	由于已达到单轴上的正限值，轴组无法运行
72	由于已达到单轴上的负限值，轴组无法运行
73	轴组未处于 ErrorStop 状态时调用 GRPx_RESET。
74	减速停止正在进行中，不能通过另一个减速停止命令中止。
75	GRPx_STOP 指令的模式非法。仅支持模式 0（减速停止）和模式 1（立即停止）。

错误代码	描述
76	正在进行的运动被 GRPx_STOP 指令中止。
77	GRPx_STOP 指令触发的减速停止被立即停止中止。

15.9 轴和轴组之间的交互

轴组由 2 或 3 个已组态轴构成。单轴和轴组不能同时执行，但可调用某些单轴运动指令，以进行轴组运动准备。

单轴和轴组指令

单轴和轴组的指令不能同时执行。

GRP0_2/3D_MOVELINEAR 或 GRP0_MOVEPATH 指令在执行时，CPU 不会接受并执行单轴运动指令，而是将返回相应的错误代码并显示在单轴功能块中。轴组直线插补运动将继续执行。

相反的情况同样适用，发出轴组直线插补运动指令将不会中断正在运行的单轴指令。

轴组运动

在运动向导中进行组态后，按照以下步骤编写轴组运动程序。

1. 初始化轴组
   使用 AXISx_CTRL (页 1123) 指令初始化每个轴，从而激活轴组。
2. 搜索参考点位置或为轴组建立新位置
   使用 AXISx_RSEEK (页 1129) 或 AXISx_LDPOS (页 1131) 指令搜索参考点位置或为每个轴创建新位置，以用于轴组。
3. 编写轴组运动程序
   使用 GRP0_2/3D_MOVELINEAR (页 1187) 或 GRP0_MOVEPATH (页 1190) 命令 CPU 执行轴组运动。

技术规范 A

A.1 常规技术规范

遵守的标准

S7-200 SMART 自动化系统符合以下标准和测试规范。S7-200 SMART 自动化系统的测试标准均基于这些标准和测试规范。

CE 认证

S7-200 SMART 自动化系统满足下列 EC 指令的要求和安全相关目标，并符合欧盟官方公报中发布的可编程控制器的统一欧洲标准 (EN)。

- 2014/30/EU“电磁兼容性”（EMC 指令）
- 2011/65/EU“电气和电子设备中特定有害物质的使用限制”（RoHS 指令）

EC 符合性声明可从以下责任机构获得，并保存在以下地址：

Siemens Aktiengesellschaft

DI FA

Gleiwitzer Straße 555

DE-90475 Nürnberg

可以在 Internet (https://support.industry.siemens.com/cs/ww/zh/ps/14031/cert?ct=444&ci=526) 找到 EC 符合性声明。

英国合格认定标志

S7-200 SMART 自动化系统符合英国政府官方汇总清单中发布的专为可编程逻辑控制器制定的英国标准 (BS) 的要求。S7-200 SMART 自动化系统符合以下法规以及相关修订版的要求和保护目标：

- 电磁兼容性法规 2016 (EMC)
- 电气和电子设备中特定有害物质的使用限制规范 2012 (RoHS)

符合性声明可从以下责任机构获得，并保存在以下地址：

Siemens plc
Princess Road
Manchester
M20 2UR
United Kingdom

可以在 Internet (https://support.industry.siemens.com/cs/ww/zh/ps/14031/cert?ct=444&ci=5596) 找到 UK 符合性声明。

工业环境

S7-200 SMART 自动化系统设计用于工业环境。

表格 A-1　工业环境

应用现场	噪声辐射要求	抗噪声要求
工业	EN 61000-6-4	EN 61000-6-2

电磁兼容性

电磁兼容性 (EMC) 是电气设备在电磁环境中按预期运行以及运行时电磁干扰的发射水平 (EMI) 不会干扰周围其它电气设备的能力。

表格 A-2　抗扰度符合 EN 61000-6-2

电磁兼容性 - 抗扰度符合 EN 61000-6-2	
EN 61000-4-2 静电放电	所有表面的空气放电电压：±2kV；±4kV；±8 kV 暴露导电表面的接触放电电压：±4 kV
EN 61000-4-3 辐射、无线电频率、电磁场抗扰度测试	80 到 1000 MHz，10 V/m，1 kHz 时 80% AM 1.4 到 6.0 GHz，3 V/m，1 kHz 时 80% AM
EN 61000-4-4 快速瞬变脉冲	±2 kV，5 kHz 和 100 kHz
EN 61000-4-5 浪涌抗扰度	DC 系统 ±1 kV 共模，±0.5kV 差模

电磁兼容性 - 抗扰度符合 EN 61000-6-2	
EN 61000-4-6 抗传导干扰	150 kHz 到 80 MHz，10 V RMS，1 kHz 时 80% AM
EN 61000-4-29 电压突降	DC 系统（CPU 模块） 10ms

表格 A-3　传导和辐射发射符合 EN 61000-6-4

电磁兼容性 - 传导和辐射发射符合 EN 61000-6-4		
传导发射 EN 55011，A 类，1 组	0.15 MHz 到 0.5 MHz	<79dB (μV) 准峰值；<66 dB (μV) 均值
	0.5 MHz 到 30 MHz	<73dB (μV) 准峰值；<60 dB (μV) 均值
辐射发射 EN 55011，A 类，1 组	30 MHz 到 230 MHz	<50dB (μV/m) 准峰值；测量距离为 3 m
	230 MHz 到 1 GHz	<57dB (μV/m) 准峰值；测量距离为 3 m
	1 GHz 到 3 GHz	<76dB (μV/m) 峰值，56dB(μV/m) rms；测量距离为 3 m
	3 GHz 到 6 GHz	<80dB (μV/m) 峰值，60dB(μV/m) rms；测量距离为 3 m

环境条件

表格 A-4　运输与储存

环境条件 - 运输和存储	
EN 60068-2-2，测试 Bb，干热和 EN 60068-2-1，测试 Ab，寒冷	-40 °C 至 +70 °C
EN 60068-2-30，测试 Db，湿热	25 °C 到 55 °C，湿度 95%
EN 60068-2-14，测试 Na，温度骤变	-40 °C 到 +70 °C，停顿时间 3 小时，5 个周期
EN 60068-2-31，自由落体	0.3 m，5 次，产品包装
大气压	1139 到 660 hPa（相当于海拔 -1000 到 3500 m）

表格 A-5　运行条件

环境条件 - 运行	
环境温度范围 （设备下部 25 mm 进风距离）	-20 °C 至 60 °C 水平安装 -20 °C 至 50 °C 垂直安装 湿度 95%，不结露
大气压	1139 到 795 hPa（相当于海拔 -1000 到 2000 m）
污染物浓度	SO_2：< 0.5 ppm；H_2S：< 0.1 ppm；RH < 60% 不结露
EN 60068-2-14，测试 Nb，温度变化	-20 °C 到 60 °C，3 K/min
EN 60068-2-27 机械冲击	15 g，11 ms 脉冲，3 个轴向各 6 次冲击
EN 60068-2-6 正弦振动	DIN 导轨安装：5-8.4 Hz 时 3.5 mm，8.4 - 150 Hz 时 1g

表格 A-6　高电位绝缘测试

高电位绝缘测试	
24 V DC/5 V DC 标准电路	707 V DC（光隔离边界的型式测试）
230 V AC 电路接地和 24 V DC/5 V DC 电路	3000 V AC 或 4250 V DC
以太网端口对 24 V DC/5 V DC 电路和地 [1]	1500 V AC（仅型式测试）

[1] 以太网端口隔离旨在在危险电压引起短期网络故障时限制危险情况带来的影响。它不遵照常规 AC 线电压隔离的安全要求。

绝缘

绝缘根据 EN 61131-2 要求设计。

说明

对于具有 24 V DC 电源电压的模块，针对最大 60 VAC/75 VDC 设计电气隔离，根据 EN 61131-2 设计基本绝缘。

符合 IEC 61131-2 的污染等级/过压类别

- 污染等级 2
- 过压类别：II

符合 IEC 61131-2 的保护等级

- 保护等级 II 符合 EN 61131-2（不需要保护导线）

防护等级 IP20

- IP20 机械保护，EN 60529
- 防止手指接触经标准探针测试出的高压。需要针对灰尘、污物、水和直径小于 12.5mm 的异物施加外部保护。

额定电压

表格 A-7　　额定电压

模块	额定电压	容错
用于 DC/DC/DC 模块	24 V DC	20.4 V DC 到 28.8 V DC

说明

当某个机械触点将输出电源连接到 S7-200 SMART CPU 或其它数字量扩展模块，该触点将发送信号“1”到数字量输出并持续大约 150 ms。这可能引发意外的机械或过程操作，从而导致死亡、重伤和/或设备损坏。必须考虑这一点，尤其是使用响应短脉冲的设备时。

警告

机械触点接通输出电源的持续时间

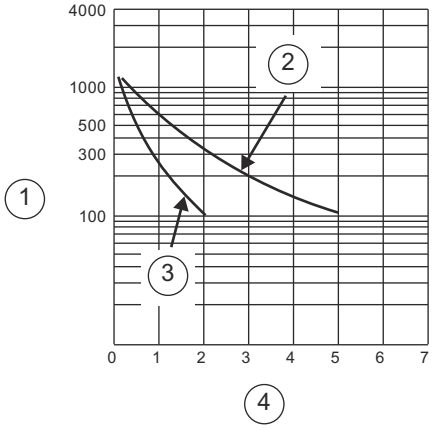
当某个机械触点将输出电源连接到 S7-200 SMART CPU 或其它数字量扩展模块，该触点将发送信号“1”到数字量输出并持续大约 150 ms。

由此会引发意外的机械或过程操作，进而可能会导致人员死亡或重伤和/或设备损坏。

必须将这一点考虑在内，尤其是使用响应短脉冲的设备时。

继电器电气使用寿命

继电器供应商提供的典型性能数据如下。根据具体应用，实际性能可能不同。使用适合于负载的外部保护电路可增强触点的使用寿命。



① 使用寿命（x 10^3 次动作）

② 250 V AC 电阻负载
30 V DC 电阻负载

③ 250 V AC 电感负载 (p.f = 0.4)
30 V DC 电感负载 (L/R = 7ms)

④ 额定工作电流 (A)

A.2 S7-200 SMART CPU

A.2.1 CPU ST32

A.2.1.1 常规规范和特性

CPU ST32 的常规规范和特性

表格 A-8 常规规范

技术数据	CPU ST32 DC/DC/DC
订货号	6ES7288-1ST32-0AA0
尺寸 W x H x D (mm)	100 x 118 x 75
重量	305 g
功耗	24 W
可用电流（EM 总线）	最大 1680 mA(5 V DC)

技术数据	CPU ST32 DC/DC/DC
可用电流 (24 V DC)	最大 300 mA（传感器电源）
数字量输入电流 消耗 (24 V DC)	**I0.0-I0.7:** • 10mA @24v • 6mA @5v **I1.0-1.7:** • 4mA @24v

表格 A-9　CPU 特征

技术数据		CPU ST32（V3.2) DC/DC/DC
用户存储器	程序	120 KB[1]
	用户定义区域	896 KB
	用户数据 (V)	100 KB
	保持性	最大 40 KB[2]
板载数字量 I/O		16 点输入/16 点输出
过程映像		584 位输入 (I)/584 位输出 (Q)
模拟映像		266 字输入 (AI)/266 字输出 (AQ)
位存储器 (M)		512 位
临时（局部）存储器 (L)		采用 LAD 或 FBD 进行编程时，主程序 1024 字节，每个子程序 1024 字节，中断程序 1008 字节（STEP 7-Micro/WIN 保留 4 字节）
顺序控制继电器 (S)		256 位
扩展模块扩展		最多 16 个
信号板扩展		最多 2 个
高速计数器	总数	8 个
	单相	8 个，200 kHz
	A/B 相	8 个，200 kHz
脉冲输出 [3]		• 单相： 5 个，200 kHz • A/B 相：4 个，200 kHz
脉冲捕捉输入		16 个
循环中断		10 个，1 ms 分辨率 包括中断号 10（定时中断 0）、中断号 11（定时中断 1），以及 8 个由 1ms 定时器（CT=PT）触发的中断

技术数据	CPU ST32（V3.2) DC/DC/DC
沿中断	4 个上升沿和 4 个下降沿（使用可选信号板时，各为 8 个）
存储卡	microSD 卡（最大 128 GB）
实时时钟精度	120 秒/月
实时时钟保持时间	通常为 7 天，25 °C 时最少为 6 天

[1] CPU ST32(V3.1)的用户程序大小为 100 KB。

[2] 可组态 V 存储器、M 存储器、C 存储器（当前值）的存储区、DB2 的存储区，以及 T 存储器（保持性定时器上的当前值）中要保持的部分，最高至最大指定量。

[3] 指定的最大脉冲频率仅适用于带晶体管输出的 CPU 型号。对于带有继电器输出的 CPU 型号，不建议进行脉冲输出操作。

表格 A-10 PROFINET 特性

说明	CPU ST32（V3.2) DC/DC/DC
支持用于 PROFINET 通信的控制器功能	✓
支持用于 PROFINET 通信的智能设备功能	✓
PROFINET 设备最大数	10
PROFINET 设备的设备编号	1 至 10
PN IRT 支持的设备最大数	8
每台 PROFINET 设备的最大输入大小	128 字节
每台 PROFINET 设备的最大输出大小	128 字节
PROFINET 智能设备的最大输入大小	128 字节
PROFINET 智能设备的最大输出大小	128 字节
最大模块数	64
PROFINET 支持的最小发送时钟	1 ms
PROFINET 过程映像输入寄存器的 CPU 地址范围	I128.0 至 I1535.7
PROFINET 过程映像输出寄存器的 CPU 地址范围	Q128.0 至 Q1535.7
1 号设备的 PROFINET 过程映像输入和输出寄存器的 CPU 地址	I128.0 至 I255.7 Q128.0 至 Q255.7
2 号设备的 PROFINET 过程映像输入和输出寄存器的 CPU 地址	I256.0 至 I383.7 Q256.0 至 Q383.7

说明	CPU ST32（V3.2) DC/DC/DC
3 号设备的 PROFINET 过程映像输入和输出寄存器的 CPU 地址	I384.0 至 I511.7 Q384.0 至 Q511.7
4 号设备的 PROFINET 过程映像输入和输出寄存器的 CPU 地址	I512.0 至 I639.7 Q512.0 至 Q639.7
5 号设备的 PROFINET 过程映像输入和输出寄存器的 CPU 地址	I640.0 至 I767.7 Q640.0 至 Q767.7
6 号设备的 PROFINET 过程映像输入和输出寄存器的 CPU 地址	I768.0 至 I895.7 Q768.0 至 Q895.7
7 号设备的 PROFINET 过程映像输入和输出寄存器的 CPU 地址	I896.0 至 I1023.7 Q896.0 至 Q1023.7
8 号设备的 PROFINET 过程映像输入和输出寄存器的 CPU 地址	I1024.0 至 I1151.7 Q1024.0 至 Q1151.7
智能设备的 PROFINET 过程映像输入和输出寄存器的 CPU 地址	I1152.0 至 I1279.7 Q1152.0 至 Q1279.7
9 号设备的 PROFINET 过程映像输入和输出寄存器的 CPU 地址 [1]	I1128.0 至 I1407.7 Q1128.0 至 Q1407.7
10 号设备的 PROFINET 过程映像输入和输出寄存器的 CPU 地址 [1]	I1408.0 至 I1535.7 Q1408.0 至 Q1535.7

[1] 不适用于 CPU ST32(V3.1) 。

表格 A-11 性能

指令类型	执行速度
布尔运算	90 ns/指令
移动字	0.7 μs/指令
实数数学运算	2.2 μs/指令

表格 A-12　支持的用户程序元素

元素		说明
POU	类型/数量	主程序：1 子例程：128（0 到 127） 功能块：128（0 到 127） 中断例程：128（0 到 127）
	嵌套深度	从主程序：8 级子程序/功能块 从中断例程：4 级子程序/功能块 注：功能块嵌套功能块示例遵循 UDT 实例嵌套逻辑（最多 4 个）
累加器	数量	4
定时器	类型/数量	非保持性（TON、TOF）：320 保持性 (TONR)：192
计数器	数量	512

表格 A-13　通信

技术数据	CPU ST32 DC/DC/DC
端口数	PROFINET (LAN)：2 串行端口：1 (RS485) 附加串行端口：2（带有可选 RS232/485 信号板）
HMI 设备	PROFINET (LAN)：8 个连接 串行端口：每个端口 4 个连接
编程设备 (PG)	PROFINET (LAN)：1 个连接 串行端口：1 个连接
CPU (PUT/GET)	PROFINET (LAN)：8 个客户端和 8 个服务器连接
开放式用户通信	PROFINET (LAN)：8 个主动和 8 个被动连接
数据传输率	PROFINET (LAN)：100 Mb/s RS485 系统协议：9600、19200 和 187500 b/s RS485 自由端口：1200 到 115200 b/s
隔离（外部信号与 PLC 逻辑侧）	PROFINET (LAN)：变压器隔离，1500 V AC RS485： RS485 信号与外壳接地，707 V DC； RS485 信号与 CPU 逻辑公共端，707 V DC

技术数据	CPU ST32 DC/DC/DC
电缆类型	PROFINET (LAN)：CAT5e 屏蔽电缆； RS485：PROFIBUS 网络电缆
PROFINET 通信	
PROFINET 控制器	✓
PROFINET 设备	✓
PROFINET 控制器	
服务	
PG/OP 通信	✓
S7 路由	✓
等时同步模式	✓
开放式 IE 通信	✓
IRT	✓
MRP	-
PROFIenergy	-
可为 RT 连接的 PROFINET 设备的最大数量	10
最大模块数量	64
更新时间	更新时间的最小值还取决于为 PROFINET 设置的通信组件、PROFINET 设备的数量以及已组态的用户数据量。
发送时钟	1 ms 到 4 ms

表格 A-14　电源

技术数据		CPU ST32 DC/DC/DC
电压范围		20.4 到 28.8 V DC [1]
电源频率		--
输入电流（最大负载时）	仅 CPU	24 V DC 时 120 mA（无 300 mA 传感器驱动功率） 24 V DC 时 420 mA（带 300 mA 传感器驱动功率）
	具有所有扩展附件的 CPU	24 V DC 时 1 A
保持时间（掉电）		24 V DC 时 10 ms
浪涌电流（最大）		28.8 V DC 时 15 A
隔离（输入电源与逻辑侧）		--

技术数据	CPU ST32 DC/DC/DC
漏地电流，交流线路对功能地	--
内部保险丝，用户不可更换	3.5 A，125 V，快速熔断

[1] 电源电压从 0 V 上升至正常工作电压范围的时间不得超过 1.5 秒。

表格 A-15 传感器电源

技术数据	CPU ST32 DC/DC/DC
电压范围	19.2 到 28.8 V DC
额定输出电流（最大）	300 mA
最大波纹噪声 (<10 MHz)	< 1 V 峰峰值
隔离（CPU 逻辑侧与传感器电源）	未隔离

A.2.1.2 数字量输入和输出

表格 A-16 数字量输入

技术数据	CPU ST32 DC/DC/DC
输入点数	16
类型	漏型/源型（IEC 1 类漏型，I0.0 到 I0.7 除外）
额定电压	**I0.0-I0.7:** • 10mA @24v • 6mA @5v **I1.0-1.7:** • 4mA @24v
允许的连续电压	30 V DC，最大值
浪涌电压	35 V DC，持续 0.5 s
逻辑 1 信号（最小）	• I0.0 到 I0.7：6 mA 时 4 V DC • 其它输入点：2.5 mA 时 15 V DC
逻辑 0 信号（最大）	• I0.0 到 I0.7：1 mA 时 1 V DC • 其它输入：1 mA 时 5 V DC
隔离（现场侧与逻辑侧）	707 V DC，持续 1 分钟
隔离组	2

技术数据	CPU ST32 DC/DC/DC
滤波时间	每个通道上可单独选择（点 I0.0 到 I1.7）： μs：0.2、0.4、0.8、1.6、3.2、6.4、12.8 ms：0.2、0.4、0.8、1.6、3.2、6.4、12.8
HSC 时钟输入频率	单相：8 个 HSC，200 kHz； A/B 相：8 个 HSC，200 kHz；
同时接通的输入数	16
电缆长度（最大值），以米为单位	I0.0 到 I1.7 ： 屏蔽（仅限此类）： • 50 m HSC（高速）输入
	非 HSC 使用： 屏蔽： • 500 m 正常输入

说明

提高 20kHz 以上高频方波信号质量

要获得 20 kHz 以上的干净方波，可考虑以下改进措施：

- 最大程度缩短电缆长度
- 从仅漏型/源型驱动器切换为推挽式驱动器
- 升级为更高质量的电缆
- 若输入信号的电平转换速度较慢，建议在输入侧添加下拉或上拉电阻，以加快信号边沿
- 建议使用双绞线，通过将信号线与回路线成对绞合，可有效降低信号串扰

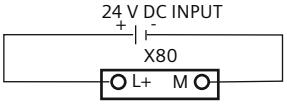
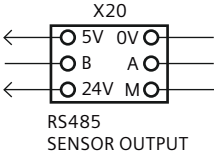
表格 A-17　数字量输出

技术数据	CPU ST32 DC/DC/DC
输出点数	16
类型	固态 - MOSFET（源型）
电压范围	20.4 到 28.8 V DC
最大电流时的逻辑 1 信号	20 V DC 最小
具有 10 KΩ 负载时的逻辑 0 信号	0.1 V DC 最大
每点的额定电流（最大）	0.5 A
每个公共端的额定电流（最大）	8 A

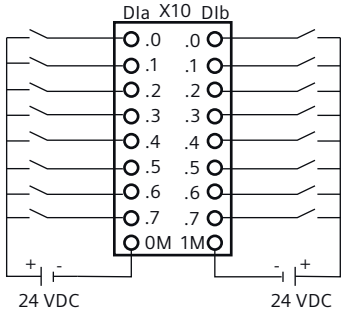
技术数据	CPU ST32 DC/DC/DC
灯负载	5 W
通态电阻	最大 0.4 Ω
每点的漏电流	最大 10 μA
浪涌电流	8 A，最长持续 100 ms
过载保护	-
隔离（现场侧与逻辑侧）	707 V DC，持续 1 分钟
隔离电阻	--
断开触点间的绝缘	--
隔离组	1
电感钳位电压	L+ - 48 V DC，1 W 损耗
开关延迟（Qa.0 到 Qb.7）	断开到接通最长为 0.5 μs 接通到断开最长为 2μs
继电器最大开关频率	--
机械寿命（无负载）	--
额定负载下的触点寿命	--
STOP 模式下的输出状态	上一个值或替换值（默认值为 0）
同时接通的输出数	16
电缆长度（最大值），以米为单位	屏蔽：500 m 非屏蔽：150 m

A.2.1.3 接线图

接线图和插针连接器位置如下所示：

CPU ST32 DC/DC/DC (6ES7288-1ST32-0AA0) 的连接器引脚位置				
	X80：直流电源接线端子			
	直流电源	L+	M	
	引脚编号			
	X20：RS485、传感器输出			
	直流电源	5 V	0 V	
	信号	B	A	
	传感器电源输出	24 V	M	

CPU ST32 DC/DC/DC (6ES7288-1ST32-0AA0) 的连接器引脚位置



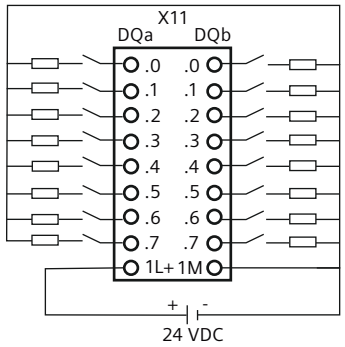
X10：直流输入接线端子

信号	DI a.0	DI a.1	DI a.2	DI a.3	DI a.4	DI a.5	DI a.6	DI a.7	0M
引脚编号	1	3	5	7	9	11	13	15	17
引脚编号	2	4	6	8	10	12	14	16	18
信号	DI b.0	DI b.1	DI b.2	DI b.3	DI b.4	DI b.5	DI b.6	DI b.7	1M

a, b 组均支持源型、漏型接法，互不影响

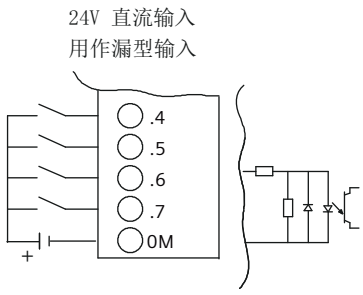
对于漏型输入，将 24 VDC “-” 连接到 “M”（如图所示）。

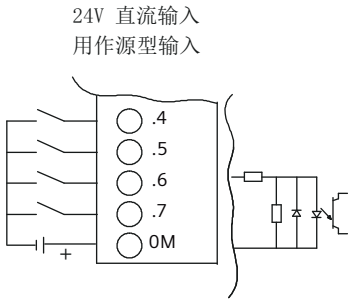
对于源型输入，将 24 VDC “+” 连接到 “M”

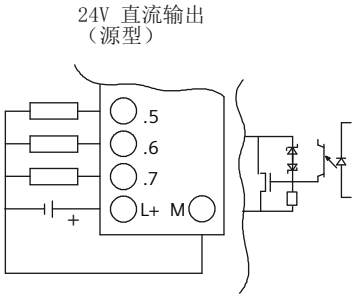


X11：直流输出接线端子

信号	DQ a.0	DQ a.1	DQ a.2	DQ a.3	DQ a.4	DQ a.5	DQ a.6	DQ a.7	L+
引脚编号	1	3	5	7	9	11	13	15	17
引脚编号	2	4	6	8	10	12	14	16	18
信号	DQ b.0	DQ b.1	DQ b.2	DQ b.3	DQ b.4	DQ b.5	DQ b.6	DQ b.7	M







A.3 数字量输入和输出扩展模块 (EM)

A.3.1 EM DE16 数字量输入规范

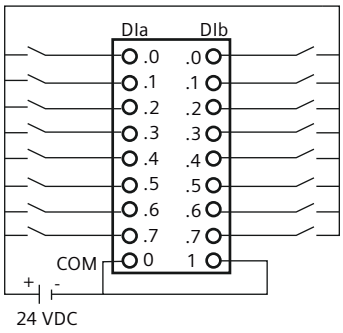
表格 A-18 常规规范

型号	EM 16 点数字量输入 (EM DE16)
产品编号	6ES7288-2DE16-0AA1
尺寸 W x H x D (mm)	15 x 118 x 74.5
重量	78 g
功耗	2.2 W
电流消耗（SM 总线）	80 mA
电流消耗 (24 V DC)	所用的每点输入 4 mA

表格 A-19 数字量输入

型号	EM 16 点数字量输入 (EM DE16)
输入点数	16
类型	漏型/源型（IEC 1 类漏型）
额定电压	4 mA 时 24 V DC，额定值
允许的连续电压	30 V DC，最大值
浪涌电压	35 V DC，持续 0.5 s
逻辑 1 信号（最小）	2.5 mA 时 15 V DC
逻辑 0 信号（最大）	1 mA 时 5 V DC
隔离（现场侧与逻辑侧）	707 V DC，持续 1 分钟
隔离组	2
滤波时间	0.2、0.4、0.8、1.6、3.2、6.4 和 12.8 ms （可选择，4 个为一组）
同时接通的输入数	16
电缆长度（最大值），以米为单位	屏蔽：500 m 正常输入 非屏蔽：300 m 正常输入

EM DE16 16 点数字量输入 (6ES7288-2DE16-0AA1) 的接线图



直流输入接线端子									
信号	DI a.0	DI a.1	DI a.2	DI a.3	DI a.4	DI a.5	DI a.6	DI a.7	0
引脚编号	1	3	5	7	9	11	13	15	17
引脚编号	2	4	6	8	10	12	14	16	18
信号	DI b.0	DI b.1	DI b.2	DI b.3	DI b.4	DI b.5	DI b.6	DI b.7	1

A.3.2 EM QT16 数字量输出规范

表格 A-20 常规规范

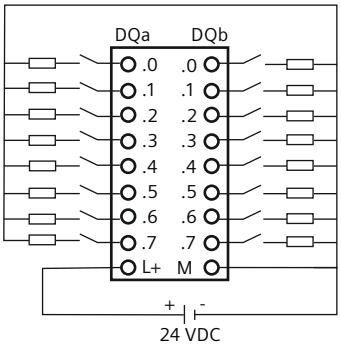
型号	**EM 16 点晶体管型 数字量输出 (EM QT16)**
订货号	6ES7288-2QT16-0AA1
尺寸 W x H x D (mm)	15 x 118 x 74.5
重量	79 g
功耗	1.5 W
电流消耗（SM 总线）	80 mA
电流消耗 (24 V DC)	控制电路最大 40 mA，L+ 最大 10 A

表格 A-21 数字量输出

型号	**EM 16 点晶体管型 数字量输出 (EM QT16)**
输出点数	16
类型	固态 - MOSFET（源型）
电压范围	20.4 到 28.8 V DC
最大电流时的逻辑 1 信号	20 V DC
具有 10 KΩ 负载时的逻辑 0 信号	0.1 V DC
每点的额定电流（最大）	0.75 A

型号	EM 16 点晶体管型 数字量输出 (EM QT16)
每个公共端的额定电流（最大）	10 A
灯负载	5 W
通态触点电阻	最大 0.6 Ω
每点的漏电流	10 μA
浪涌电流	-
过载保护	有
隔离（现场侧与逻辑侧）	707 V DC，持续 1 分钟
隔离电阻	-
断开触点间的绝缘	-
隔离组	1
电感钳位电压	L+ - 52 V，1 W 损耗
开关延时	接通延时小于 100 μs，断开延时小于 100 μS
STOP 模式下的输出行为	上一个值或替换值（默认值为 0）
同时接通的输出数	16
电缆长度（最大值），以米为单位	屏蔽：500 m 非屏蔽：150 m

EM QT16 16 点晶体管型数字量输出 (6ES7288-2QT16-0AA1)



直流输出接线端子									
信号	DQ a.0	DQ a.1	DQ a.2	DQ a.3	DQ a.4	DQ a.5	DQ a.6	DQ a.7	L+
引脚编号	1	3	5	7	9	11	13	15	17
引脚编号	2	4	6	8	10	12	14	16	18
信号	DQ b.0	DQ b.1	DQ b.2	DQ b.3	DQ b.4	DQ b.5	DQ b.6	DQ b.7	M

A.4 模拟量输入和输出扩展模块 (EM)

A.4.1 EM AE04 和 EM AE08 模拟量输入规范

表格 A-22 常规规范

型号	EM 4 点模拟量输入 (EM AE04)	EM 8 点模拟量输入 (EM AE08)
订货号	6ES7288-3AE04-0AA1	6ES7288-3AE08-0AA1
尺寸 W x H x D (mm)	15 x 118 x 74.5	15 x 118 x 74.5
重量	81 g	83 g
功耗	1.3 W	1.9 W
电流消耗（SM 总线）	70 mA	80 mA
电流消耗 (24 V DC)	40 mA	60 mA

表格 A-23 模拟量输入

型号	EM 4 点模拟量输入 (EM AE04)	EM 8 点模拟量输入 (EM AE08)
输入点数	4	8
类型	电压或电流（差分输入），可选 2 个为一组	电压或电流（差分输入），可选 2 个为一组
范围	±10 V、±5 V、±2.5 V 或 0 到 20 mA	±10 V、±5 V、±2.5 V 或 0 到 20 mA
满量程范围（数据字）	-27,648 到 27,648	-27,648 到 27,648
过冲/下冲范围（数据字）	电压：27,649 到 32,511/-27,649 到 -32,512 电流：27,649 到 32,511/-4864 到 0（请参见模拟量输入电压表示法和模拟量输入电流表示法 (页 1237)。）	电压：27,649 到 32,511/-27,649 到 -32,512 电流：27,649 到 32,511/-4864 到 0（请参见模拟量输入电压表示法和模拟量输入电流表示法 (页 1237)。）
上溢/下溢（数据字）	电压：32,512 到 32,767/-32,513 到 -32,768 电流：32,512 到 32,767/-4,865 到 -32,768（请参见模拟量输入电压表示法和模拟量输入电流表示法 (页 1237)。）	电压：32,512 到 32,767/-32,513 到 -32,768 电流：32,512 到 32,767/-4,865 到 -32,768（请参见模拟量输入电压表示法和模拟量输入电压表示法 (页 1237)。）

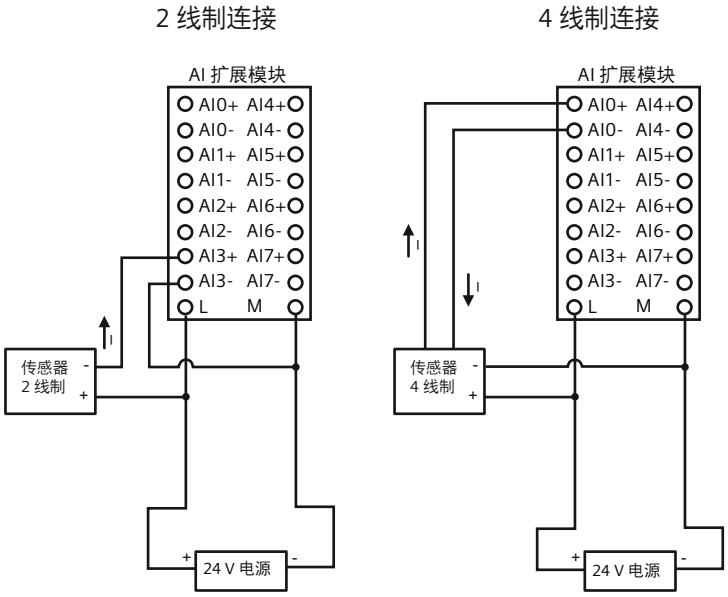
型号	EM 4 点模拟量输入 (EM AE04)	EM 8 点模拟量输入 (EM AE08)
分辨率	电压模式：12 位 + 符号位 电流模式：12 位	电压模式：12 位 + 符号位 电流模式：12 位
最大耐压/耐流	±35 V/±40 mA	±35 V/±40 mA
平滑化	无、弱、中或强（参见阶跃响应下的模拟量输入响应时间）(页 1236)	无、弱、中或强（参见阶跃响应下的模拟量输入响应时间）(页 1236)
噪声抑制	400、60、50 或 10 Hz	400、60、50 或 10 Hz
输入阻抗	≥1 M Ω（电压）/290 Ω（电流）	≥1 M Ω（电压）/290 Ω（电流）
隔离（现场侧与逻辑侧）	无	无
精度（25 °C/-20 到 60 °C）	电压模式：满量程的 ±0.1%/±0.2% 电流模式：满量程的 ±0.2%/±0.3%	电压模式：满量程的 ±0.1%/±0.2% 电流模式：满量程的 ±0.2%/±0.3%
测量原理	实际值转换	实际值转换
共模抑制	40 dB，DC 到 60 Hz	40 dB，DC 到 60 Hz
工作信号范围	信号加共模电压必须小于 +12 V 且大于 -12 V	信号加共模电压必须小于 +12 V 且大于 -12 V
电缆长度（最大值），以米为单位	100 m 屏蔽线	100 m 屏蔽线

表格 A-24　诊断

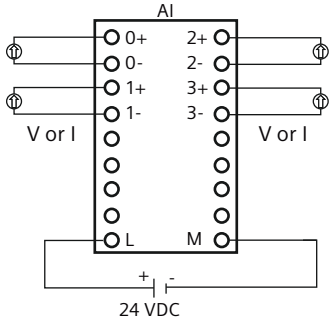
型号	EM 4 点模拟量输入 (EM AE04)	EM 8 点模拟量输入 (EM AE08)
上溢/下溢	✓	✓
24 V DC 低压	✓	✓

EM AE04 和 EM AE08 接线电流变送器

接线电流变送器可用作 2 线制变送器和 4 线制变送器，如下图所示。

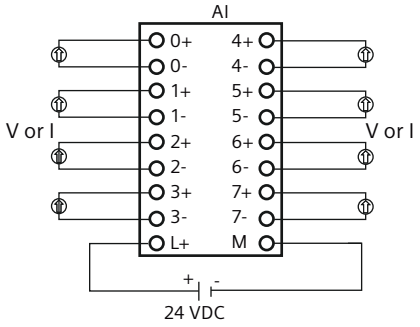


EM AE04 4 点模拟量输入 (6ES7288-3AE04-0AA1)



直流输入接线端子									
信号	AI 0+	AI 0-	AI 1+	AI 1-					L+
引脚编号	1	3	5	7	9	11	13	15	17
引脚编号	2	4	6	8	10	12	14	16	18
信号	AI 2+	AI 2-	AI 3+	AI 3-					M

EM AE08 8 点模拟量输入 (6ES7288-3AE08-0AA1)



直流输入接线端子									
信号	AI 0+	AI 0-	AI 1+	AI 1-	AI 2+	AI 2-	AI 3+	AI 3-	L+
引脚编号	1	3	5	7	9	11	13	15	17
引脚编号	2	4	6	8	10	12	14	16	18
信号	AI 4+	AI 4-	AI 5+	AI 5-	AI 6+	AI 6-	AI 7+	AI 7-	M

说明

应将未使用的模拟量输入短接。

A.4.2 EM AQ04 模拟量输出模块规范

表格 A-25 常规规范

技术数据	EM 4 点模拟量输出 (EM AQ04)
订货号	6ES7288-3AQ04-0AA1
尺寸 W x H x D (mm)	15 x 118 x 74.5
重量	79 g
功耗	1.3 W（无负载）
电流消耗（SM 总线）	60 mA
电流消耗 (24 V DC)	40 mA（无负载）
	120 mA（每个通道存在 20 mA 负载）

表格 A-26 模拟量输出

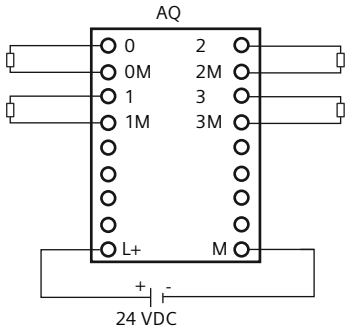
技术数据	EM 4 点模拟量输出 (EM AQ04)
输出点数	4
类型	电压或电流
范围	±10 V 或 0 到 20 mA
分辨率	电压模式：11 位 + 符号位 电流模式：11 位
满量程范围（数据字）	电压：-27,648 到 27,648 电流：0 到 27,648 （请参见电压和电流的输出范围 (页 1238)。）
精度（25 °C/-20 到 60 °C）	满量程的 ±0.5%/±1.0%
稳定时间（新值的 95%）	电压：300 µs (R)，750 µs (1 uF) 电流：600 µs (1 mH)，2 ms (10 mH)

技术数据	EM 4 点模拟量输出 (EM AQ04)
负载阻抗	电压 ：≥ 1000 Ω 电流：≤ 600 Ω
STOP 模式下的输出行为	上一个值或替换值（默认值为 0）
隔离（现场侧与逻辑侧）	无
电缆长度（最大值），以米为单位	100 m 屏蔽双绞线

表格 A-27　诊断

技术数据	EM 4 点模拟量输出 (EM AQ04)
上溢/下溢	✓
对地短路（仅限电压模式）	✓
断路（仅限电流模式）	✓
24 V DC 低压	✓

EM AQ04 4 点模拟量输出 (6ES7288-3AQ04-0AA1)



直流输入接线端子

信号	AQ 0	AQ 0M	AQ 1	AQ 1M					L+
引脚编号	1	3	5	7	9	11	13	15	17
引脚编号	2	4	6	8	10	12	14	16	18
信号	AQ 2	AQ 2M	AQ 3	AQ 3M					M

说明

保持未使用的通道处于默认电压模式（电压输出为 0V），无需接线。
如果未使用的通道设置为电流模式（电流输出为 0mA），请短接输出，否则通道将报警开路。

AQ04 模块电流模式降额趋势图

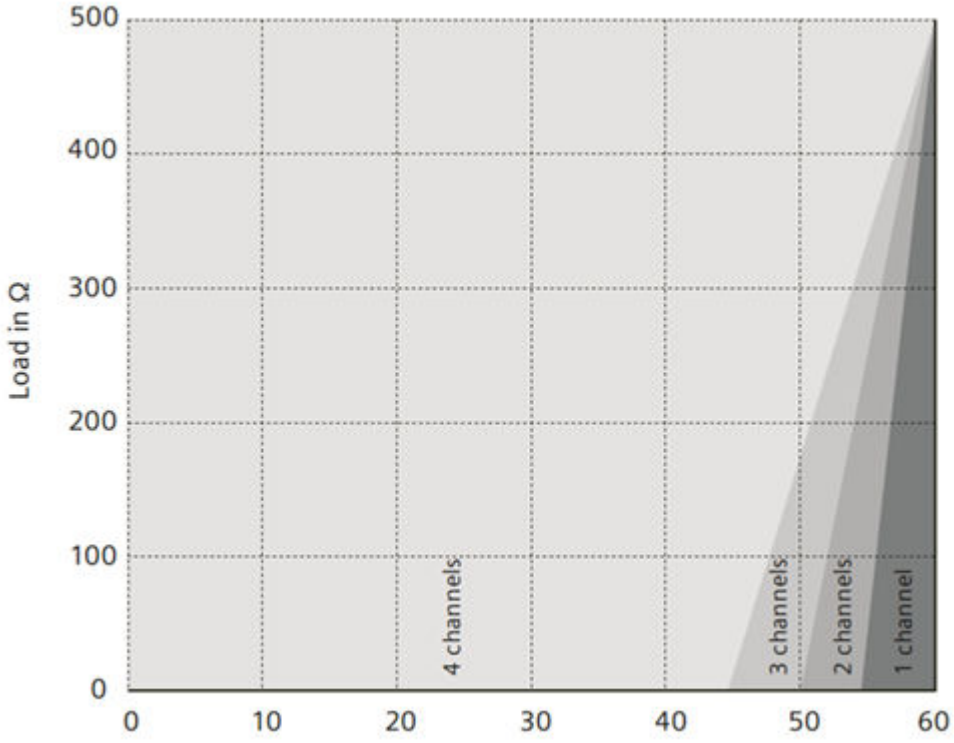


图 A-1　水平安装，环境温度（单位：℃）

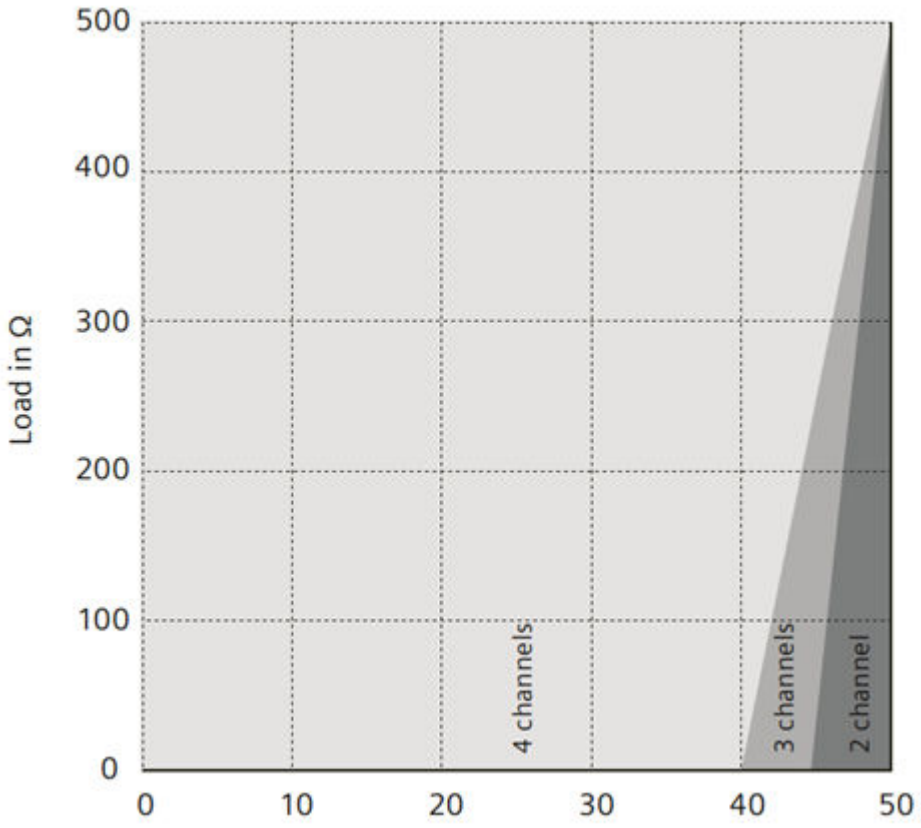


图 A-2　垂直安装，环境温度（单位：℃）

A.4.3　EM AM06 模拟量输入/输出模块规范

表格 A-28　常规规范

技术数据	EM 4 点模拟量输入/2 点模拟量输出 (AM06)
订货号	6ES7288-3AM06-0AA1
尺寸 W x H x D (mm)	15 x 118 x 74.5
重量	82 g
功耗	1.5 W（无负载）
电流消耗（SM 总线）	60 mA
电流消耗 (24 V DC)	50 mA（无负载）
	90 mA（每个通道存在 20 mA 负载）

表格 A-29　模拟量输入

型号	EM 4 点模拟量输入/2 点模拟量输出 (AM06)
输入点数	4
类型	电压或电流（差动）：可 2 个选为一组
范围	±10 V、±5 V、±2.5 V 或 0 到 20 mA
满量程范围（数据字）	-27,648 到 27,648
过冲/下冲范围（数据字）	电压：27,649 到 32,511/-27,649 到 -32,512 电流：27,649 到 32,511/-4,864 到 0 （请参见模拟量输入电压表示法和模拟量输入电流表示法 (页 1237)。）
上溢/下溢（数据字）	电压：32,512 到 32,767/-32,513 到 -32,768 电流：32,512 到 32,767/-4,865 到 -32,768 （请参见模拟量输入电压表示法和模拟量输入电流表示法 (页 1237)。）
分辨率	电压模式：12 位 + 符号 电流模式：12 位
最大耐压/耐流	±35 V/±40 mA
平滑化	无、弱、中或强 （参见阶跃响应下的模拟量输入响应时间） (页 1236)
噪声抑制	400、60、50 或 10 Hz
输入阻抗	≥1 M Ω（电压）/290 Ω（电流）
隔离（现场侧与逻辑侧）	无
精度（25 °C/-20 到 60 °C）	电压模式：满量程的 ±0.1%/±0.2% 电流模式：满量程的 ±0.2%/±0.3%
模数转换时间	625 μs（400 Hz 抑制）
共模抑制	40 dB，DC 到 60 Hz
工作信号范围	信号加共模电压必须小于 +12 V 且大于 -12 V
电缆长度（最大值），以米为单位	100 m 屏蔽线

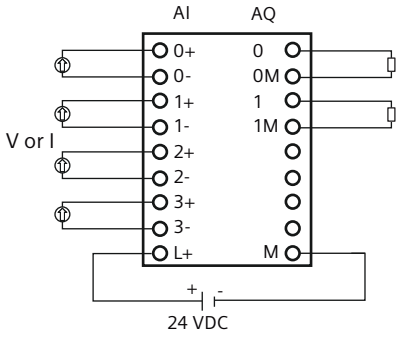
表格 A-30　模拟量输出

技术数据	EM 4 点模拟量输入/2 点模拟量输出 (AM06)
输出点数	2
类型	电压或电流
范围	±10 V 或 0 到 20 mA
分辨率	电压模式：11 位 + 符号 电流模式：11 位
满量程范围（数据字）	电压：-27,648 到 27,648 电流：0 到 27,648
精度（25 °C/-20 到 60 °C）	满量程的 ±0.5%/±1.0%
稳定时间（新值的 95%）	电压：300 μs (R)，750 μs (1 uF) 电流：600 μs (1 mH)，2 ms (10 mH)
负载阻抗	电压 ：≥ 1000 Ω 电流：≤ 00 Ω
STOP 模式下的输出行为	上一个值或替换值（默认值为 0）
隔离（现场侧与逻辑侧）	无
电缆长度（最大值），以米为单位	100 m 屏蔽线

表格 A-31　诊断

型号	EM 4 点模拟量输入/2 点模拟量输出 (AM06)
上溢/下溢	✓
对地短路（仅限电压模式）	✓
断路（仅限电流模式）	✓
24 V DC 低压	✓

EM AM06 4 点模拟量输入 /2 点模拟量输出 (6ES7288-3AM06-0AA1)



直流输入接线端子

信号	AI 0+	AI 0-	AI 1+	AI 1-	AI 2+	AI 2-	AI 3+	AI 3-	L+
引脚编号	1	3	5	7	9	11	13	15	17
引脚编号	2	4	6	8	10	12	14	16	18
信号	AQ 0	AQ 0M	AQ 1	AQ 1M					M

说明

保持为使用的输出通道处于默认电压模式（电压输出为 0V）,无需接线。
如果未使用的输出通道设置为电流模式（电流输出为 0mA）,请短接输出，否则通道将报警开路。

说明

应将未使用的模拟量输入短接。

A.4.4 模拟量输入的阶跃响应

表格 A-32 阶跃响应 (ms)，0 到满量程（在 95% 处测得）

平滑化选项（采样平均）	噪声消减/抑制频率（积分时间选项）			
	400 Hz (2.5 ms)	**60 Hz (16.6 ms)**	**50 Hz (20 ms)**	**10 Hz (100 ms)**
无（1 个周期）：不求平均值	4 ms	18 ms	22 ms	100 ms
弱（4 个周期）：4 次采样	9 ms	52 ms	63 ms	320 ms
中（16 个周期）：16 次采样	32 ms	203 ms	241 ms	1200 ms
强（32 个周期）：32 次采样	61 ms	400 ms	483 ms	2410 ms
采样时间 • 4 AI x 13 位 • 8 AI x 13 位	 • 0.625 ms • 1.25 ms	 • 4.17 ms • 4.17 ms	 • 5 ms • 5 ms	 • 25 ms • 25 ms

A.4.5 模拟量输入的采样时间和更新时间

表格 A-33 采样时间和更新时间

抑制频率（积分时间）	采样时间	所有通道的模块更新时间	
		4 通道 SM	**8 通道 SM**
400 Hz (2.5 ms)	• 4 通道 SM： 0.625 ms • 8 通道 SM： 1.250 ms	0.625 ms	1.250 ms
60 Hz (16.6 ms)	4.170 ms	4.17 ms	4.17 ms
50 Hz (20 ms)	5.000 ms	5 ms	5 ms
10 Hz (100 ms)	25.000 ms	25 ms	25 ms

A.4.6 模拟量输入的电压和电流测量范围（SB 和 SM）

表格 A-34 模拟量输入的电压表示法（SB 和 EM）

系统		电压测量范围				
十进制	十六进制	**±10 V**	**±5 V**	**±2.5 V**	**±1.25 V**	
32767	7FFF[1]	11.851 V	5.926 V	2.963 V	1.481 V	上溢
32512	7F00					
32511	7EFF	11.759 V	5.879 V	2.940 V	1.470 V	过冲范围
27649	6C01					
27648	6C00	10 V	5 V	2.5 V	1.250 V	额定范围
20736	5100	7.5 V	3.75 V	1.875 V	0.938 V	
1	1	361.7 μV	180.8 μV	90.4 μV	45.2 μV	
0	0	0 V	0 V	0 V	0 V	
-1	FFFF					
-20736	AF00	-7.5 V	-3.75 V	-1.875 V	-0.938 V	
-27648	9400	-10 V	-5 V	-2.5 V	-1.250 V	
-27649	93FF					下冲范围
-32512	8100	-11.759 V	-5.879 V	-2.940 V	-1.470 V	
-32513	80FF					下溢
-32768	8000	-11.851 V	-5.926 V	-2.963 V	-1.481 V	

[1] 返回 7FFF 可能由以下原因之一所致：上溢（如该表中所述）、有效值可用前（例如刚上电时）或者检测到断线时。

表格 A-35　模拟量输入的电流表示法（SB 和 EM）

系统		电流测量范围	
十进制	**十六进制**	**0 mA 到 20 mA**	
32767	7FFF	23.70 mA	上溢
32512	7F00		
32511	7EFF	23.52 mA	过冲范围
27649	6C01		
27648	6C00	20 mA	额定范围
20736	5100	15 mA	
1	1	723.4 nA	
0	0	0 mA	
-1	FFFF		下冲范围
-4864	ED00	-3.52 mA	
-4865	ECFF		下溢
-32768	8000		

A.4.7　模拟量输出的电压和电流测量范围（SB 和 EM）

表格 A-36　模拟量输出的电压表示法（SB 和 EM）

系统		电压输出范围	
十进制	**十六进制**	**±10 V**	
32767	7FFF	请参见注 1	上溢
32512	7F00	请参见注 1	
32511	7EFF	11.76 V	过冲范围
27649	6C01		
27648	6C00	10 V	额定范围
20736	5100	7.5 V	
1	1	361.7 μ V	
0	0	0 V	
-1	FFFF	-361.7 μ V	
-20736	AF00	-7.5 V	
-27648	9400	-10 V	

系统		电压输出范围	
十进制	十六进制	**±10 V**	
-27649	93FF		下冲范围
-32512	8100	-11.76 V	
-32513	80FF	请参见注 1	下溢
-32768	8000	请参见注 1	

[1] 在上溢或下溢情况下，模拟量输出将采用 STOP 模式的替代值。

表格 A-37 模拟量输出的电流表示法（SB 和 EM）

系统		当前输出范围	
十进制	十六进制	**0 mA 到 20 mA**	
32767	7FFF	请参见注 1	上溢
32512	7F00	请参见注 1	
32511	7EFF	23.52 mA	过冲范围
27649	6C01		
27648	6C00	20 mA	额定范围
20736	5100	15 mA	
1	1	723.4 nA	
0	0	0 mA	
-1	FFFF		下冲范围
-6912	E500		
-6913	E4FF		不可能。输出值限制在 0 mA。
-32512	8100		
-32513	80FF	请参见注 1	下溢
-32768	8000	请参见注 1	

[1] 在上溢或下溢情况下，模拟量输出将采用 STOP 模式的替代值。

A.5 热电偶扩展模块 (EM)

A.5.1 EM AT08 热电偶规范

表格 A-38 常规规范

型号	EM AT08 AI 8 x 16 位 TC
产品编号	6ES7288-3AT08-0AA1
尺寸 W x H x D (mm)	15 x 118 x 74.5
重量	79 g
功耗	0.525 W
电流消耗（SM 总线）	105 mA

表格 A-39 模拟量输入

型号		EM AT08 AI 8 x 16 位 TC
输入点数		8
范围 额定范围（数据字） 过量程/欠量程（数据字） 上溢/下溢（数据字）		请参见热电偶选型表。
分辨率	温度	0.1 °C/0.1 °F
	电压	15 位 + 符号
最大耐压		± 35 V
噪声抑制		对于所选滤波器设置 85 dB (10 Hz/50 Hz/60 Hz/400 Hz)
共模抑制		大于 120 dB
阻抗		≥ 10 MΩ
隔离	现场侧与逻辑侧	707 V DC
通道间隔离		120 V AC
精度		请参见热电偶选型表。
可重复性		±0.05% FS

型号	EM AT08 AI 8 x 16 位 TC
测量原理	积分型
模块更新时间	请参见滤波器选型表。
冷端误差	±2.5 °C
电缆长度（米）	至传感器最长 100 m
导线电阻	最大 100 Ω

表格 A-40　诊断

型号	EM AT08 AI 8 x 16 位 TC
上溢/下溢 [1]	√
断线 [2]	√

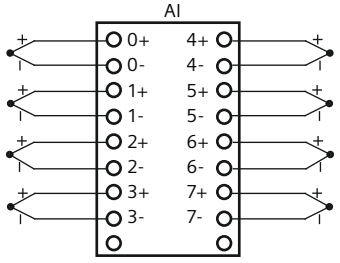
[1] 上溢、下溢和低压诊断报警信息将以模拟数据值的形式报告，即使在模块组态中禁用这些报警也会如此。

[2] 如果断线报警已禁用，但传感器接线存在开路情况，则模块可能会报告随机值。

EM AT08 热电偶 (TC) 模拟量扩展模块可测量连接到模块输入的电压值。温度测量类型可以是“热电偶”或“电压”类型。

- “热电偶”：将度数乘 10 得到该值（例如，25.3 度将报告为十进制数 253）。
- “电压”：额定范围的满量程值将是十进制数 27648。

EM AT08 RTD 热电偶 8 点 16 位 (6ES7288-3AT08-0AA1)



直流输入接线端子									
信号	AI 0+/ TC	AI 0-/ TC	AI 1+/ TC	AI 1-/ TC	AI 2+/ TC	AI 2-/ TC	AI 3+/ TC	AI 3-/ TC	
引脚编号	1	3	5	7	9	11	13	15	17
引脚编号	2	4	6	8	10	12	14	16	18
信号	AI 4+/ TC	AI 4-/ TC	AI 5+/ TC	AI 5-/ TC	AI 6+/ TC	AI 6-/ TC	AI 7+/ TC	AI 7-/ TC	

说明

应将未使用的模拟量输入短接。

两种不同的金属彼此之间存在电气连接时，便会形成热电偶。热电偶产生的电压与结点温度成正比。电压很小；一微伏能表示很多度。测量热电偶产生的电压，对额外的结点进行补偿，然后将测量结果线性化，这些是使用热电偶测量温度的基础。

将热电偶连接到 EM AT08 热电偶模块时，需将两条不同的金属线连接至模块的信号连接器上。这两条不同的金属线互相连接的位置即形成了传感器热电偶。

在这两条不同的金属线与信号连接器相连的位置，构成了另外二个热电偶。连接器温度会引起一定的电压，该电压将添加到传感器热电偶产生的电压中。如果不对该电压进行修正，结果报告的温度将偏离传感器温度。

冷端补偿便是用于对连接器热电偶进行补偿。热电偶表是基于参比端温度（通常是零摄氏度）得来的。冷端补偿用于将连接器温度修正为零摄氏度。冷端补偿可消除连接器热电偶增加的电压。模块的温度在内部测量，然后转换为数值并添加到传感器换算中。之后是使用热电偶表对修正后的传感器换算值进行线性化。

为使冷端补偿取得最佳效果，必须将热电偶模块安装在温度稳定的环境中。符合模块规范的模块环境温度的缓慢变化（低于 0.1 °C/分钟）能够被正确补偿。穿过模块的空气流动也会引起冷端补偿误差。

如果需要更佳的冷端误差补偿效果，则可使用外部 iso 热端子块。热电偶模块可以使用 0 °C 基准值或 50 °C 基准值端子块。

下表显示了 EM AT08 热电偶扩展模块支持的各种热电偶的测量范围和精度：

表格 A-41　EM AT08 热电偶选型表

类型	低于范围最小值 [1]	额定范围下限	额定范围上限	超出范围最大值 [2]	25 °C 时的额定范围 [3, 4] 精度	-20 °C 到 60 °C 时的额定范围 [1, 2] 精度
J	-210.0 °C	-150.0 °C	1200.0 °C	1450.0 °C	±0.6 °C	±1.2 °C
K	-270.0 °C	-200.0 °C	1372.0 °C	1622.0 °C	±0.7 °C	±1.5 °C
T	-270.0 °C	-200.0 °C	400.0 °C	540.0 °C	±0.7 °C	±1.5 °C
E	-270.0 °C	-200.0 °C	1000.0 °C	1200.0 °C	±0.6 °C	±1.2 °C
R & S	-50.0 °C	100.0 °C	1768.0 °C	2019.0 °C	±1.0 °C	±2.5 °C
B	0.0 °C	200.0 °C	800.0 °C	--	±2.0 °C	±3.5 °C
	--	800.0 °C	1820.0 °C	1820 °C	±1.0 °C	±2.3 °C
N	-270.0 °C	-200.0 °C	1300.0 °C	1550.0 °C	±1.0 °C	±1.6 °C
C	0.0 °C	100.0 °C	2315.0 °C	2500.0 °C	±0.7 °C	±2.7 °C

类型	低于范围最小值 [1]	额定范围下限	额定范围上限	超出范围最大值 [2]	25 °C 时的额定范围 [3, 4] 精度	-20 °C 到 60 °C 时的额定范围 [1, 2] 精度
TXK/XK(L)	-200.0 °C	-150.0 °C	800.0 °C	1050 °C	±0.6 °C	±1.2 °C
电压	-32512	-27648 -80mV	27648 80mV	32511	±0.05%	±0.1%

[1] “低于范围最小值”以下的热电偶值报告为 -32768。

[2] “超过范围最大值”以上的热电偶值报告为 32767

[3] 所有范围的内部冷端误差均为 ± 2.5 °C。此误差需添加到本表中的误差值上。模块需要至少 30 分钟的预热时间才能满足该规范。

说明

热电偶通道

热电偶扩展模块的每个通道均可使用不同型号的热电偶组态（可在模块组态期间在软件中选择）。

表格 A-42 EM AT08 热电偶的噪声消减和更新时间

抑制频率选择	积分时间	8 通道模块更新时间（秒）
400 Hz (2.5 ms)	10 ms [1]	0.286
60 Hz (16.6 ms)	16.67 ms	0.446
50 Hz (20 ms)	20 ms	0.526
10 Hz (100 ms)	100 ms	2.450

[1] 当选择 400 Hz 抑制时，为保证模块分辨率及精度，积分时间应当为 10 ms。同时，该选择也会抑制频率为 100 Hz 和 200 Hz 的噪声。

测量热电偶时建议使用 100 ms 的积分时间。使用更小的积分时间将增大温度读数的重复性误差。

说明

对模块上电后，模块将对模数转换器执行内部校准。在此期间，模块将报告每个通道的值为 32767，直到相应通道出现有效值为止。用户程序可能需要考虑这段初始化时间。由于模块的组态可能改变初始化时长，因此，应验证组态中模块的行为。如果需要，可以在用户程序中包含逻辑，以适应模块的初始化时间。

J 型热电偶模拟值的表示

J 型热电偶模拟值的表示如下表所示。

表格 A-43　J 型热电偶模拟值的表示

用 °C 表示的 J 型	功能单元		用 °F 表示的 J 型	功能单元		范围
	十进制	十六进制		十进制	十六进制	
> 1450.0	32767	7FFF	> 2642.0	32767	7FFF	溢出
1450.0 : 1200.1	14500 : 12001	38A4 : 2EE1	2642.0 : 2192.2	26420 : 21922	6734 : 55A2	超出上限
1200.0 : -150.0	12000 : -1500	2EE0 : FA24	2192.0 : -238.0	21920 : -2380	55A0 : F6B4	额定范围
-150.1 : -210.0	-1501 : -2100	FA23 : F7CC	-238.2 : -346.0	-2382 : -3460	F6B2 : F27C	超出下限
< -210.0	-32768	8000	< -346.0	-32768	8000	下溢 [1]

[1] 如果发生接线错误（例如极性接反或输入开路），或者传感器在负测量范围内出现故障（例如，热电偶类型错误），可能会导致热电偶模块信号超出下限。

A.6　RTD 扩展模块(EM)

A.6.1　EM AR04 RTD 规范

EM RTD 规范

表格 A-44　常规规范

技术数据	EM RTD 4 x 16 位 (EM AR04)
产品编号	6ES7288-3AR04-0AA1
尺寸 W x H x D (mm)	15 x 118 x 74.5
重量	77 g

技术数据	EM RTD 4 x 16 位 (EM AR04)
功耗	0.525 W
电流消耗（SM 总线）	105 mA

表格 A-45　模拟量输入

技术数据		EM RTD 4 x 16 位 (EM AR04)
输入点数		4
类型		模块参考 RTD 和 Ω
范围 额定范围（数据字） 过冲/下冲范围（数据字） 上溢/下溢（数据字）		请参见 RTD 传感器选型表。
分辨率	温度	0.1 °C/0.1 °F
	电阻	15 位 + 符号位
最大耐压		±35 V
噪声抑制		85 dB (10 Hz/50 Hz/60 Hz/400 Hz)
共模抑制		>120 dB
阻抗		≥10 M Ω
隔离	现场侧与逻辑侧	707 V DC
通道间隔离		无
精度		请参见 RTD 传感器选型表。
可重复性		±0.05% FS
最大传感器功耗		0.5 m W
测量原理		Sigma-delta
模块更新时间		请参见降噪选项表。
电缆长度（最大值），以米为单位		至传感器最长 100 m
导线电阻（最大）	10 Ω RTD 除外	20 Ω
	10 Ω RTD	2.7 Ω

表格 A-46　诊断

技术数据	EM RTD 4 x 16 位 (EM AR04)
上溢/下溢 [1,2]	有
断线 [3]	有

[1] 上溢、下溢和低压诊断报警信息将以模拟数据值的形式报告，即使在模块组态中禁用这些报警也会如此。

[2] 对于电阻范围，始终会禁用下溢检测。

[3] 如果断线报警已禁用，但传感器接线存在开路情况，则模块可能会报告随机值。

EM RTD 模拟量扩展模块可测量连接至模块输入的电阻的值。测量类型可选为“电阻”型或“热电阻”型。

- “电阻”：额定范围的满量程值将是十进制数 27648。
- “热电阻”：将度数乘 10 得到该值（例如，25.3 度将报告为十进制数 253）。

EM RTD 模块支持采用 2 线制、3 线制和 4 线制方式连接到传感器电阻进行测量。

表格 A-47　RTD 扩展模块支持的不同传感器的范围和精度

温度系数	RTD 类型	低于范围最小值 [1]	额定范围下限	额定范围上限	超出范围最大值 [2]	25 ℃ 时的额定范围精度	-20 ℃ 到 60 ℃ 时的额定范围精度
Pt 0.003850 ITS90 DIN EN 60751	Pt 10	-243.0 °C	-200.0 °C	850.0 °C	1000.0 °C	±1.0 °C	±2.0 °C
	Pt 50	-243.0 °C	-200.0 °C	850.0 °C	1000.0 °C	±0.5 °C	±1.0 °C
	Pt 100						
	Pt 200						
	Pt 500						
	Pt 1000						
Pt 0.003902 Pt 0.003916 Pt 0.003920	Pt 100	-243.0 °C	-200.0 °C	850.0 °C	1000.0 °C	± 0.5 °C	±1.0 °C
	Pt 200	-243.0 °C	-200.0 °C	850.0°C	1000.0°C	± 0.5 °C	±1.0 °C
	Pt 500						
	Pt 1000						
Pt 0.003910	Pt 10	-273.2 °C	-240.0 °C	1100.0°C	1295 °C	±1.0 °C	±2.0 °C
	Pt 50	-273.2 °C	-240.0 °C	1100.0 °C	1295 °C	±0.8 °C	±1.6 °C
	Pt 100						
	Pt 500						

温度系数	RTD 类型	低于范围最小值 [1]	额定范围下限	额定范围上限	超出范围最大值 [2]	25 ℃ 时的额定范围精度	-20 ℃ 到 60 ℃ 时的额定范围精度
Ni 0.006720 Ni 0.006180	Ni 100	-105.0 °C	-60.0 °C	250.0 °C	295.0 °C	±0.5 °C	±1.0 °C
	Ni 120						
	Ni 200						
	Ni 500						
	Ni 1000						
LG-Ni 0.005000	LG-Ni 1000	-105.0 °C	-60.0 °C	250.0 °C	295.0 °C	±0.5 °C	±1.0 °C
Ni 0.006170	Ni 100	-105.0 °C	-60.0 °C	180.0 °C	212.4 °C	±0.5 °C	±1.0 °C
Cu 0.004270	Cu 10	-240.0 °C	-200.0 °C	260.0 °C	312.0 °C	±1.0 °C	±2.0 °C
Cu 0.004260	Cu 10	-60.0 °C	-50.0 °C	200.0 °C	240.0°C	±1.0 °C	±2.0 °C
	Cu 50	-60.0 °C	-50.0 °C	200.0 °C	240.0°C	±0.6 °C	±1.2 °C
	Cu 100						
Cu 0.004280	Cu 10	-240.0 °C	-200.0 °C	200.0 °C	240.0°C	±1.0 °C	±2.0 °C
	Cu 50	-240.0 °C	-200.0 °C	200.0 °C	240.0°C	±0.7 °C	±1.4 °C
	Cu 100						

[1] “低于范围最小值”以下的 RTD 值报告为 -32768。

[2] 超出范围最大值以上的 RTD 值报告为 +32767。

表格 A-48 电阻

范围	低于范围最小值	额定范围下限	额定范围上限	超出范围最大值 [1]	25 ℃ 时的额定范围精度	-20 ℃ 到 60 ℃ 时的额定范围精度
150 Ω	不适用	0 (0 Ω)	27648 (150 Ω)	176.383 Ω	±0.05%	±0.1%
300 Ω	不适用	0 (0 Ω)	27648 (300 Ω)	352.767 Ω	±0.05%	±0.1%
600 Ω	不适用	0 (0 Ω)	27648 (600 Ω)	705.534 Ω	±0.05%	±0.1%
3000 Ω	不适用	0 (0 Ω)	27648 (3000 Ω)	3527.4 Ω	±0.05%	±0.1%

[1] 超出范围最大值以上的电阻值报告为 +32767。

说明

对于没有连接传感器的激活通道，模块会报告 32767。如果还启用了开路检测，模块会使相应的红色 LED 闪烁。

对于其它值较低的电阻使用 600 Ω 和 3000 Ω RTD 范围时，误差可能增加到指定误差的两倍。

2 线模式的连接线电阻会导致传感器读数误差，因此无法保证精度。

表格 A-49 RTD 模块的噪声消减和更新时间

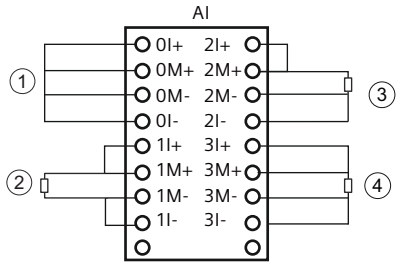
抑制频率选择	积分时间	更新时间（秒）
400 Hz (2.5 ms)	10 ms [1]	4/2 线制：0.142 3 线制：0.285
60 Hz (16.6 ms)	16.67 ms	4/2 线制：0.222 3 线制：0.445
50 Hz (20 ms)	20 ms	4/2 线制：0.262 3 线制：.505
10 Hz (100 ms)	100 ms	4/2 线制：1.222 3 线制：2.445

[1] 在选择 400 Hz 滤波器时，要维持模块的分辨率和精度，积分时间应为 10 ms。同时，该选择也会抑制频率为 100 Hz 和 200 Hz 的噪声。

说明

对模块上电后，模块将对模数转换器执行内部校准。在此期间，模块将报告每个通道的值为 32767，直到相应通道出现有效值为止。用户程序可能需要考虑这段初始化时间。由于模块的组态可能改变初始化时长，因此，应验证组态中模块的行为。如果需要，可以在用户程序中包含逻辑，以适应模块的初始化时间。

EM AR04 RTD 4 点 16 位 (6ES7288-3AR04-0AA1) 的接线图



直流输入接线端子

信号	AI 0I+	AI 0M +	AI 0M -	AI 0I-	AI 1I+	AI 1M +	AI 1M -	AI 1I-	
引脚编号	1	3	5	7	9	11	13	15	17
引脚编号	2	4	6	8	10	12	14	16	18
信号	AI 2I+	AI 2M +	AI 2M -	AI 2I-	AI 3I+	AI 3M +	AI 3M -	AI 3I-	

① 环接未使用的 RTD 输入② 2 线制 RTD ③ 3 线制 RTD ④ 4 线制 RTD

注意

外部电压参考源生成

该模块利用激励电流源生成外部电压参考。要顺利产生外部电压参考，外部电流回路必须连接良好。如果在自校准过程中出现断线等问题，激励电流源无法正常生成参考源，导致校准参数异常，进而无法正常采样数据。

A.7 RS485/RS232 信号板

A.7.1 SB RS485/RS232 规范

表格 A-50 常规规范

技术数据	SB RS485/RS232
产品编号	6ES7288-5CM01-0AA1
尺寸 W x H x D (mm)	14.5 x 62.45 x 51.5
重量	20 g
功耗	0.35 W

技术数据	SB RS485/RS232
电流消耗 (5 V DC)	30 mA[1]
电流消耗 (24 V DC)	8 mA[2]

[1] CPU 内部电源

[2] CPU 内部电源， 仅适用于 RS485 模式

表格 A-51 RS485 发送器和接收器

技术数据	SB RS485
共模电压范围	-7 V 到 +12 V，1 秒，3 VRMS 连续
发送器差动输出电压	RL = 100 Ω 时，最小 2 V RL = 54 Ω 时，最小 1.5 V
端接和偏置	TXD 上 4.7 K Ω 连接到 +5 V RXD 上 4.7 K Ω 连接到 GND
接收器输入阻抗	最小 96 K Ω
接收器阈值/灵敏度	20 mV/200 mV,典型滞后 40 mV
隔离	RS485 信号与外壳接地，707 V DC； RS485 信号与 CPU 逻辑公 共端，707 V DC;
电缆长度，屏蔽	带有隔离中继器：1000 m 最多 187.5 Kbps 不带有隔离中继器：50 m

表格 A-52 RS232 发送器和接收器

技术数据	SB RS232
发送器输出电压	RL = 3K Ω 时，最小 +/-5 V
传送输出电压	+/-5.5 V DC，最大值
接收器输入阻抗	最小 3 K Ω
接收器阈值/灵敏度	最低 0.8 V，最高 2.4 V 典型滞后 0.5 V
接收器输入电压	+/- 25 V DC ，最大值

技术数据	SB RS232
隔离	RS232 信号与外壳接地， 707 V DC； RS232 信号与 CPU 逻辑公共端，707 V DC
电缆长度，屏蔽	最长 10 m

SB CM01 RS485/RS232 (6ES7288-5CM01-0AA1)		
SB CM01 5V Tx/B Rx/A M RTS	信号	**X12： CM01 信号板 (SB) 端口（端口 1）**
	+5 V DC	+5 V，内部有 100 Ω 串联电阻
	Tx/B	RS232-Tx/RS485-B
	Rx/A	RS232-Rx/RS485-A
	M 接地	逻辑公共端
	RTS (TTL)	请求发送

计算功率预算

B

B.1 功率预算

CPU 有一个内部电源，用于为 CPU、扩展模块以及信号板供电，并可满足其它 24 V DC 用户的电源要求。请使用以下信息作为指导，确定 CPU 可为组态提供多少电能（或电流）。

请参见具体 CPU (页 1214) 的技术规范确定 24 V DC 传感器供电预算。请参考计算功率预算 (页 1214)，确定 CPU 能为您的组态提供多少电能（或电流）。

说明

如果超出 CPU 功率预算，则可能无法连接 CPU 允许的最大数量模块。

CPU 还提供 24 V DC 传感器电源，可以为输入点、扩展模块上的继电器线圈电源或其它要求供给 24 V DC。如果您的电源要求超出该传感器电源的预算，则必须给系统增加外部 24 V DC 电源。必须将 24 V DC 电源手动连接到输入点或继电器线圈。

如果需要外部 24 V DC 电源，请确保该电源不要与 CPU 的传感器电源并联。为提高电气噪声保护能力，建议将不同电源的公共端 (M) 连接在一起。

⚠ 警告

安全电源连接

将外部 24 V DC 电源与 CPU 的 24 V DC 传感器电源并联会导致这两个电源之间有冲突，因为每个电源都试图建立自己首选的输出电压电平。

该冲突可能导致其中一个电源或两个电源的寿命缩短或立即发生故障，从而导致 PLC 系统意外运行。意外运行可能导致人员死亡、重伤和/或设备损坏。

CPU 的直流传感器电源和任何外部电源应给不同点供电。允许将多个公共端连接到一起。

S7-200 SMART 系统中的一些 24 V DC 电源输入端口是互连的，并且通过一个公共逻辑电路连接多个 M 端子。例如，在数据表中指定为“非隔离”时，以下电路是互连的：CPU 的 24 V

DC 电源、EM 的继电器线圈的电源输入或非隔离模拟量输入的电源。所有非隔离的 M 端必须连接到同一个外部参考电位。

⚠ 警告
防止意外电流 将非隔离的 M 端子连接到不同参考电位将导致意外的电流，该电流可能导致 PLC 和任何连接设备损坏或运行不确定。 不遵守这些准则可能会导致设备损坏或运行不确定，而后者可能导致死亡、人员重伤和/或财产损失。 务必确保 S7-200 SMART 系统中的所有非隔离 M 端子都连接到同一个参考电位。

参见

功率要求计算示例 (页 1254)

B.2 功率要求计算示例

功率要求计算示例

下表给出了包括以下模块的 CPU 系统的功率要求计算例子：

- CPU ST32
- 1 个 EM 4 点模拟量输入 (AE04)
- 1 个 EM 8 点模拟量输入 (AE08)
- 1 个 EM 点模拟量输出 (AQ04)

本例中的 CPU 传感器电源为所有扩展模块提供了足够的 24 V DC 电流。I/O 需要 230 mA，但 CPU 提供了 300 mA。该安装不需要额外电源为扩展模块供电。

表格 B-1　示例组态功率预算计算

CPU 功率预算	**24 V DC**
CPU ST32 传感器电源	300 mA
减去	
系统要求	
插槽 0：EM AE04	40 mA

CPU 功率预算	24 V DC
插槽 1：EM AE08	50 mA
插槽 1：EM AQ04	150 mA
总要求	240 mA
等于	
电流差额	24 V DC
总电流差额	60 mA

B.3 计算功率要求

计算功率要求

通过下表可以确定 CPU 可为您的组态提供多少电能（或电流）。有关所用 CPU 型号的功率预算和数字量模块、模拟量模块或信号板的功率要求，请参见技术规范 (页 1214)。

表格 B-2 功率预算

功率预算	24 V DC
减去	
系统要求	24 V DC
总要求	
等于	
电流差额	24 V DC
总电流差额	

错误代码

C

C.1 时间戳不匹配

该警告信息表明项目的时间戳与 PLC 中程序的时间戳不匹配。这可能说明程序不同，在这种情况下，继续当前操作可能有危险。但是，程序可能在功能上完全相同，但时间戳不同。

哪些操作会修改程序时间戳？

每个程序都包含两个不同的时间戳：“创建”时间戳和“上次修改”时间戳。利用“新建项目”选项创建项目时，设置“创建”时间戳。任何用户编辑或程序编译操作都不会影响“创建”时间戳。

“上次修改”时间戳用于指示用户最后修改程序的时间。许多情况都会导致设置“最后修改”时间戳：

1. 在程序块编辑器中编辑指令或操作数。
2. 添加、删除或修改全局变量或局部变量。
3. 添加或删除 POU。
4. 编译程序块。
5. 下载程序块（执行该操作会自动编译程序块，因此会设置“最后修改”时间戳）。

请注意，虽然所有这些操作都会导致设置“最后修改”时间戳，但这并不意味着程序不同。因此，STEP 7-Micro/WIN SMART V3 提供“比较”(Compare) 选项，用于确定程序是否真的不同

如何确定程序是否确实不同？

可单击“比较”(Compare) 按钮，将 PLC 中的程序块与项目程序块进行比较。通过此比较结果可确定是否继续执行状态操作。

如何同步时间戳？

将新项目下载到 PLC 可同步时间戳，从而可以运行状态。

C.2 PLC 非致命错误代码

PLC 编译器和运行时间错误属于非致命错误。非致命错误可能降低 PLC 的某些性能，但不会导致 PLC 无法执行用户程序或更新 I/O。

- **运行时编程错误**是在程序执行过程中由用户程序造成的非致命错误条件。例如，间接地址指针，该指针在程序编译时有效，而在程序执行时则改为指向一个超出范围的地址。访问 PLC 菜单功能区的“PLC 信息”(PLC Information) (页 180) 可确定发生的错误类型。
  只有修改用户程序，才能纠正运行时间编程错误。下一次从 STOP 模式切换到 RUN 模式时，CPU 会清除运行时编程错误。
- **PLC 编译器错误**（或程序编译错误）会阻止将程序下载到 PLC。STEP 7-Micro/WIN SMART V3 在编译 (页 86)或下载 (页 86)程序时检测到编译错误，并会在输出窗口 (页 165)中显示错误。如果发生了编译错误，PLC 会保留驻留在 PLC 中的当前程序。

I/O 错误也是非致命错误。当 CPU 的 I/O、信号板和扩展模块出现问题时，PLC 在程序能够监视和评估的特殊存储器 (SM) 位 (页 1265)中记录错误信息。

非致命错误代码

十六进制错误代码	非致命 PLC 程序编译器错误
0080	该程序对于 CPU 而言过大；请减小程序大小
0081	逻辑堆栈下溢；请将该程序段分成多个程序段
0082	指令非法；检查指令助记符
0083	主程序结束前的指令非法；请移除错误指令
0085	FOR/NEXT 的组合非法；请添加 FOR 指令或删除 NEXT 指令
0086	FOR/NEXT 的组合非法；请添加 NEXT 指令或删除 FOR 指令
0087	缺少标签或 POU；请添加相应标签
0088	子例程结束前的指令非法；请在子例程末尾添加 RET 指令或者移除错误指令
0089	中断例程结束前的指令非法；请在中断例程末尾添加 RETI 指令或者移除错误指令
008B	SCR 段的跳转非法
008C	标签或 POU 名称重复
008D	超出了标签或 POU 的最大数量；请确保不超出允许的标签数
0090	操作数非法
0091	存储器范围错误；检查操作数范围
0092	计数操作数非法；验证最大计数大小

十六进制错误代码	非致命 PLC 程序编译器错误
0093	超出 FOR/NEXT 嵌套级别
0095	缺少 LSCR 指令
0096	缺少 SCRE 指令或 SCRE 前的指令非法
0099	受密码保护的 POU 过多
009D	在系统块中检测到非法参数
009F	程序组织非法

十六进制错误代码	禁止切换到 RUN 模式（运行禁止条件）
0070	由于插入存储卡而禁止运行
0071	由于缺少组态设备而禁止运行
0072	由于设备组态不匹配而禁止运行（注：此错误也包括设备参数化错误）
0073	由于尝试更新固件而禁止运行
0074	因扩展模块或信号板出现严重硬件错误，导致运行被禁止

十六进制错误代码	非致命运行时间编程问题
0000	不存在非致命错误
0001	在执行 HDEF 指令前启用 HSC 指令
0002	已将输入中断点分配给 HSC
0003	已将 HSC 输入点分配给输入中断或其它 HSC
0004	中断例程中不允许使用指令
0005	同时执行 HSC/PLS/运动指令
0006	间接寻址错误
0007	日时钟指令数据错误
0008	超出最大用户子例程嵌套级别
0009	在端口 0 上同时执行 XMT/RCV 指令
000A	执行之前组态的 HSC 的 HDEF 指令
000B	在端口 1 上同时执行 XMT/RCV 指令
000D	尝试在激活时重新定义脉冲输出
000E	PTO 包络段的数量设置为 0
000F	在比较触点指令中遇到非法数字值

十六进制错误代码	非致命运行时间编程问题
0010	当前 PTO 操作模式不允许该命令
0011	PTO 命令代码非法
0012	PTO 包络表非法
0013	PID 回路表非法
0014	数据日志错误： • 一次程序扫描中存在过多的 DATx_WRITE 子例程执行过程。每秒只能持续执行 10 到 15 个数据日志写操作。当每秒钟执行的 DATx-WRITE 操作过多时，已分配的存储器会满，并且将在一小段时间内不会存储任何新的数据日志记录。 • 在未事先通过数据日志向导组态数据日志的情况下执行数据日志写入子例程
0016	已将 HSC 或中断输入点分配给运动指令
0017	PTO/PWM 输出点已分配给运动功能
0019	“信号板”不存在或未组态
001A	扫描看门狗超时。
001B	尝试在启用的 PWM 上更改时基
001C	扩展模块或信号板出现严重硬件错误
001D	HSC 引脚映射无效
001E	时钟 A 和时钟 B 不在同一组引脚中
001F	在端口 2 上同时执行 XMT/RCV
0090	操作数非法
0091	操作数范围错误；检查操作数范围
0092	计数操作数非法；验证最大计数大小
0098	在 RUN 模式下执行非法程序编辑
009A	在用户中断例程中尝试切换到自由端口模式
009B	字符串操作的索引非法（用户请求索引 = 0）
00A0	Web 应用静态资源错误
00A1	Web OSS 静态资源错误
00A2	不使用 SDB0 调用轴组
00B0	安全性 UMLOG 文件错误
00B1	安全性用户管理器错误
00C0	Trace 0 配置校验失败
00C1	Trace 1 配置校验失败

十六进制错误代码	非致命运行时间编程问题
00C2	Trace 2 配置校验失败
00C3	Trace 3 配置校验失败
00C4	运动控制的轨迹任务加载失败
00CF	超过最大运动控制功能块实例数
00D0	Motion 资源不足
00D1	MC_USER_TASK 运行超时

C.3 PLC 非致命错误 SM 标志

概述

非致命错误是那些可能使 PLC 性能部分下降，但不会导致 PLC 无法执行用户程序以及更新 I/O 的错误。为帮助您调试程序，与错误状况相关的信息均保存在可通过用户程序访问的专用存储器 (SM) 单元 (页 1265)。例如，如果不希望在出现某些非致命错误条件时继续处于 RUN 模式，则可让用户程序在出现不良条件时强迫切换到 STOP 模式。

下表列举并说明了特殊存储器非致命错误信息。

SM 位	非致命错误说明	SM 字节	非致命错误说明
SM0.2	保持性数据丢失	SMB6	CPU 的 I/O 错误字节
SM0.7	RTC_Lost	SMB9	模块 0 I/O 错误字节
SM1.3	除数为零错误	SMB11	模块 1 I/O 错误字节
SM3.0	奇偶校验错误	SMB13	模块 2 I/O 错误字节
SM4.0	通信中断队列溢出	SMB15	模块 3 I/O 错误字节
SM4.1	输入中断队列溢出	SMB17	模块 4 I/O 错误字节
SM4.2	定时中断队列溢出	SMB19	模块 5 I/O 错误字节
SM4.3	运行时间编程问题	SMB132	模块 6 I/O 错误字节
SM5.0	I/O 错误（任何 I/O 错误位置位）	SMB134	模块 7 I/O 错误字节
		SMB2031	模块 8 I/O 错误字节
		SMB2033	模块 9 I/O 错误字节
		SMB2035	模块 10 I/O 错误字节
		SMB2037	模块 11 I/O 错误字节
		SMB2039	模块 12 I/O 错误字节

SM 位	非致命错误说明	SM 字节	非致命错误说明
		SMB2041	模块 13 I/O 错误字节
		SMB2043	模块 14 I/O 错误字节
		SMB2045	模块 15 I/O 错误字节
		SMB29	信号板 0 I/O 错误字节
		SMB33	信号板 1 I/O 错误字节

C.4 PLC 致命错误代码

概述

致命错误导致 PLC 停止执行程序。根据错误严重程度的不同，致命错误可能会导致 PLC 无法执行任一或全部功能。处理致命错误的目的是使 PLC 进入安全状态，这样 PLC 能对现有错误条件的询问做出响应。

检测到致命错误时，PLC 执行下列任务。

- 切换到 STOP 模式
- 接通系统故障 LED 和 STOP LED
- 关闭输出

PLC 一直处于该状态，直到致命错误得到纠正。下表列出了可从 PLC 读取的致命错误代码及其说明。

STEP 7-Micro/WIN SMART 在“PLC 信息”(PLC Information) 对话框中显示 PLC 生成的错误代码和简要说明。要访问 PLC 信息，可在 PLC 菜单功能区的“信息”(Information) 区域单击 PLC 按钮 PLC 。

在纠正了导致致命错误的条件后，对 PLC 循环上电或从 STEP 7-Micro/WIN SMART 执行暖启动。要执行暖启动，在 PLC 菜单功能区的“修改”(Modify) 区域单击“暖启动”(Warm Start) 按钮 暖启动 。

重新启动 PLC 会清除致命错误条件并引起上电诊断测试。如果出现另一个致命错误条件，PLC 会再次接通系统故障 LED；否则，PLC 开始正常操作。

有几种可能的错误条件会导致 PLC 无法通信，在这种情况下，无法查看 PLC 错误代码。此类错误表明硬件发生故障，需要修理 PLC 模块；更改程序或清空 PLC 存储器解决不了问题。

致命错误代码

十六进制错误代码	说明
0000	不存在致命错误
0001	系统固件校验和错误
0002	编译的用户程序校验和错误
0004	永久存储器出现故障
0005	用户程序发生永久存储器错误
0006	系统块发生永久存储器错误
0007	强制数据发生永久存储器错误
0009	用户数据 DB1 发生永久存储器错误
000A	存储卡出现故障
000B	用户程序发生存储卡错误
000C	系统块发生存储卡错误
000D	强制数据发生存储卡错误
000F	用户数据 DB1 发生存储卡错误
0010	内部固件错误
0015	上电时，用户程序发生编译错误
0016	上电时，用户数据发生编译错误
0017	上电时，系统块发生编译错误
0018	CPU HW 标识数据不可用或损坏
0019	HW 看门狗超时错误
001A	扫描看门狗超时
0020	内部闪存中 PN SDB 校验和错误
0028	内部闪存中检测到轨迹 SDB 校验和错误
0029	轨迹 SDB 编译失败

C.5 PROFINET 报警

概述

PROFINET 报警指示 PROFINET 通信期间出现错误。为了帮助用户调试程序，系统会将与错误条件相关的信息存储在特殊存储器 (SM) 位置 (页 1265)，用户程序可访问这些特殊存储单元。

下表列举并说明了 PROFINET 报警。

USI	报警数据	说明
0x1	0x1	设备运行时出错。

特殊存储器 (SM) 和系统符号名称

D

D.1 SM（特殊存储器）概述

S7-200 SMART CPU 提供包含系统数据的特殊存储器。SMW 表示指示特殊存储器字的前缀。SMB 表示指示特殊存储器字节的前缀。将各个位寻址为 SM<字节号>.<位号>。STEP 7-Micro/WIN SMART V3 中的系统定义变量表显示特殊存储器。

SMB0 至 SMB29、SMB32 至 SMB33、SMB250 至 SMB349、SMB480 至 SMB515、SMB950 至 SMB1999 以及 SMB2014 至 SMB2849（MBS7-200 SMART 只读特殊存储器）

S7-200 SMART CPU 将新更改内容写入存储在特殊存储器内的系统数据。

从 CPU 中读取系统状态

程序中的 SMB0 至 SMB29、SMB32 至 SMB33、SMB250 至 SMB349、SMB480 至 SMB515 以及 SMB950 至 SMB1999 为只读。如果程序包含用于写入只读 SM 地址的逻辑，则 STEP 7-Micro/WIN SMART V3 将无误编译程序。但是，CPU 程序编译器将拒绝程序，并显示“操作数范围错误，下载失败”(Operand range error, Download failed)。

程序可读取存储在特殊存储器地址的数据、评估当前系统状态和使用条件逻辑决定如何响应。在运行模式下，程序逻辑连续扫描提供对系统数据的连续监视功能。

- SMB0 (页 1268) 系统状态位
- SMB1 (页 1269) 指令执行状态位
- SMB2 (页 1269) 自由端口接收字符
- SMB3 (页 1270) 自由端口奇偶校验错误
- SMB4 (页 1270) 中断队列溢出、运行时程序错误、中断已启用、自由端口发送器空闲和强制值
- SMB5 (页 1272) I/O 错误状态位
- SMB6-SMB7 (页 1272) CPU ID、错误状态和数字量 I/O 点
- SMB8-SMB19 (页 1273)I/O 模块 ID 和错误
- SM20.0 (页 1270) 自由端口发送器空闲

- SMW22-SMW26 (页 1275) 扫描时间
- SMB28-SMB29 (页 1276) 信号板 0 ID 和错误
- SMB32-SMB33 (页 1276) 信号板 1 ID 和错误
- SMB220-SMB235 (页 1298) PID 控制错误代码
- SMB250-SMB349 (页 1308) EM（扩展模块）硬件/固件 ID
- SMB480-SMB515 (页 1300)数据日志状态（只读）
- SMB929-SMB936 轨迹
- SMB950-SMB999 (页 1307) SB（信号板 1）硬件/固件 ID
- SMB1000-SMB1049 (页 1306) CPU 硬件/固件 ID
- SMB1050-SMB1099 (页 1307) SB（信号板 0）硬件/固件 ID
- SMB1100-SMB1399 (页 1308) EM（扩展模块）硬件/固件 ID
- SMB1400-SMB1799 (页 1317) EM（扩展模块）模块特定的数据
- SMB1800-SMB1939 (页 1318) PROFINET 设备状态
- SMB1940-SMB1946 (页 1319) Web 服务器状态
- SMB2050-SMB2449 (页 1308) EM（扩展模块）硬件/固件 ID
- SMB2450-SMB2999 (页 1317) EM（扩展模块）模块特定的数据

SMB30 至 SMB194 、SMB566 至 SMB858、SMB870 至 SMB928 以及 SMB2014 至 SMB2028（S7-200 SMART 读/写特殊存储器）

S7-200 SMART CPU 执行以下操作：

- 从特殊存储器读取组态/控制数据
- 将新更改内容写入特殊存储器中的系统数据。

从 CPU 读取系统状态

程序可以读取和写入此范围内的所有 SM 地址。但 SM 数据的一般用法因每个地址的功能而异。

向 CPU 写入（发送）控制命令

SM 地址提供了一种访问系统状态数据、组态系统选项和控制系统功能的方法。在运行模式下，连续扫描程序，从而连续访问特殊系统功能。

- SMB30（端口 0）、SMB130（端口 1）和 SMB31（端口 2） (页 1277)集成 RS485 端口（端口 0）、CM01 信号板 (SB) RS232/RS485 端口（端口 1）和端口（端口 2）的端口组态
- SMB34-SMB35 (页 1279) 定时中断的时间间隔
- SMB195-SMB206 (页 1279) HSC 类型和频率测量采样周期
- SMB36-SMB45 (HSC0)、SMB46-SMB55 (HSC1)、SMB56-SM65 (HSC2)、SMB136-SMB145 (HSC3)、SMB146-SMB155 (HSC4)、SMB156-SMB165 (HSC5)、SMB360-SMB369 (HSC6) 和 SMB370-SMB379 (HSC7) (页 1279)：高速计数器组态和操作
- SMB66-SMB85 (页 1282) PLS0 和 PLS1 高速输出
- SMB86-SMB94 和 SMB186-SMB194 (页 1290) 接收消息控制
- SMW98 (页 1295) I/O 扩展总线通信错误
- SMW100-SMW120 (页 1296) 系统报警
- SMW2014-SMW2028 (页 1296) 系统报警
- CM01 信号板 (SB) RS232/RS485 端口（端口 1）的 SMB130 (页 1277) 端口组态（请参见 SMB30）
- SMB166-SMB169 (页 1282) PTO0 包络定义表
- SMB176-SMB179 (页 1282) PTO1 包络定义表
- SMB186-SMB194 (页 1290) 接收消息控制（请参见 SMB86-SMB94）
- SMB195-SMB206 (页 1298) HSC 类型和频率测量采样周期
- SMB380-SMB404 (页 1299) HSC 用户自定义引脚映射启用
- SMB566-SMB575 (页 1282) PLS2 高速输出
- SMB576-SMB579 (页 1282) PTO2 包络定义表
- SMB580-SMB589 (页 1282) PLS3 高速输出
- SMB590-SMB593 (页 1282) PTO3 包络定义表
- SMB600-SMB649 (页 1301) 轴 0 开环运动控制
- SMB650-SMB699 (页 1303) 轴 1 开环运动控制
- SMB700-SMB749 (页 1303) 轴 2 开环运动控制
- SMB750-SMB799 (页 1304) 轴 3 开环运动控制
- SMB800-SMB858 (页 1304) 轴组 0 开环运动控制
- SMB870-SMB928 (页 1305) 轴组 1 开环运动控制

D.2 SMB0： 系统状态

特殊存储器字节 0 (SM0.0 - SM0.7) 包含八个位，在各扫描周期结束时 S7-200 SMART CPU 会更新这些位。

表格 D-1 SMB0 系统状态位

S7-200 SMART 变量名	SM 地址	描述
Always_On	SM0.0	该位始终为 TRUE。
First_Scan_On	SM0.1	在第一个扫描周期，CPU 将该位设置为 TRUE，此后将其设置为 FALSE。该位的一个用途是调用初始化子例程。
Retentive_Lost	SM0.2	在以下操作后，CPU 将该位设置为 TRUE 并持续一个扫描周期： • 重置为出厂通信命令 • 重置为出厂存储卡评估 • 评估程序传送卡（在此评估过程中，会从程序传送卡中加载新系统块）。 • CPU 在上次断电时存储的保持性记录出现问题。 该位可用作错误存储器位或用作调用特殊启动顺序的机制。
RUN_Power_Up	SM0.3	通过上电或暖启动条件进入 RUN 模式时，CPU 将该位设置为 TRUE 并持续一个扫描周期。该位可用于在开始操作之前给机器提供预热时间。
Clock_60s	SM0.4	该位提供一个时钟脉冲。周期时间为一分钟时，该位有 30 秒的时间为 FALSE，有 30 秒的时间为 TRUE。该位可简单轻松地实现延时或提供一分钟时钟脉冲。
Clock_1s	SM0.5	该位提供一个时钟脉冲。周期时间为一秒时，该位有 0.5 秒的时间为 FALSE，然后有 0.5 秒的时间为 TRUE。该位可简单轻松地实现延时或提供一秒钟时钟脉冲。
Clock_Scan	SM0.6	该位是一个扫描周期时钟，其在一次扫描时为 TRUE，然后在下一次扫描时为 FALSE。在后续扫描中，该位交替为 TRUE 和 FALSE。该位可用作扫描计数器输入。
RTC_Lost	SM0.7	该位适用于具有实时时钟的 CPU 型号。如果实时时钟设备的时间在上电时复位或丢失，则 CPU 将该位设置为 TRUE 并持续一个扫描周期。程序可将该位用作错误存储器位或用来调用特殊启动序列。

D.3 SMB1： 指令执行状态

特殊存储器字节 1 (SM1.0 - SM1.7) 提供各种指令的执行状态，例如表格和数学运算。执行指令时由指令置位和复位这些位。

表格 D-2 SMB1 指令执行状态位

S7-200 SMART 变量名	SM 地址	描述
Result_0	SM1.0	如果运算结果为零，某些指令会将该位设置为 TRUE。
Overflow_Illegal	SM1.1	如果结果溢出或者指令检测到非法数值，某些指令会将该位设置为 TRUE。
Neg_Result	SM1.2	如果数学运算结果为负数，则此运算会将该位设置为 TRUE。
Divide_By_0	SM1.3	如果程序试图除以零，CPU 会将该位设置为 TRUE。
Table_Overflow	SM1.4	如果参考数据表已满，添表 (ATT) 指令会将该位设置为 TRUE。
Table_Empty	SM1.5	如果 LIFO 或 FIFO 指令尝试从空表读取，CPU 会将该位设置为 TRUE。
Not_BCD	SM1.6	BCD 值转换为二进制值时如果存在非法值（非 BCD），CPU 会将该位设置为 TRUE。
Not_Hex	SM1.7	将 ASCII 值转换为十六进制 (ATH) 值时，如果值非法（非十六进制 ASCII 数字），CPU 会将该位设置为 TRUE。

D.4 SMB2： 自由端口接收字符

特殊存储器字节 2 是自由端口接收字符缓冲区。在自由端口模式下，CPU 将其接收的每个字符均存储在该字节中，以便于程序访问。

表格 D-3 SMB2 自由端口接收字符

S7-200 SMART 变量名	SM 地址	描述
Receive_Char	SMB2	该字节包含在自由端口通信过程中 CPU 从端口 0、端口 1 或端口 2 接收的各个字符。

说明

端口 0、端口 1 和端口 2 共用 SMB2 和 SMB3

当 CPU 接收到端口 0 上的字符时，请注意以下事项：

- CPU 执行连接到该事件（中断事件 8）的中断例程。
- SMB2 包含在端口 0 上接收的字符。
- SMB3 包含所接收字符的奇偶校验状态。

同样，当 CPU 接收到端口 1 上的字符时，请注意以下事项：

- CPU 执行连接到该事件（中断事件 25）的中断例程。
- SMB2 包含在端口 1 上接收的字符。
- SMB3 包含所接收字符的奇偶校验状态。

同样，当 CPU 接收到端口 2 上的字符时，请注意以下事项：

- CPU 执行连接到该事件（中断事件 47）的中断例程。
- SMB2 包含在端口 2 上接收的字符。
- SMB3 包含所接收字符的奇偶校验状态。

D.5 SMB3： 自由端口字符错误

当自由端口通信检测到接收字符存在奇偶校验错误、组帧错误、断开错误或溢出错误时，CPU 将 SM3.0 设置为 TRUE。使用此位可丢弃消息。

表格 D-4 SMB3 自由端口字符错误

S7-200 SMART 变量名	SM 地址	描述
Parity_Err	SM3.0	该位指示 CPU 从端口 0 或端口 1 上收到奇偶校验错误、组帧错误、断开错误或溢出错误： • FALSE：无错误 • TRUE：错误

D.6 SMB4、SM20.0： 中断队列溢出、运行时程序错误、中断启用、自由端口发送器空闲和强制值

特殊存储器字节 4 (SM4.0 - SM4.7) 包含中断队列溢出位。这些位指示中断发生的速度比 CPU 能处理的速度快，或者指示全局中断禁用 (DISI) 指令 (页 413)禁用了中断。

其他位指示：

- 已启用或已禁用中断状态
- 运行时程序错误
- 自由端口发送器状态
- 一个或多个强制 PLC 存储器值

表格 D-5 SMB4 系统状态

S7-200 SMART 变量名	SM 地址	描述
Comm_Int_Ovr	** SM4.0	TRUE：通信中断队列已溢出。
Input_Int_Ovr	** SM4.1	TRUE：输入中断队列已溢出。
Timed_Int_Ovr	** SM4.2	TRUE：定时中断队列已溢出。
RUN_Err	SM4.3	TRUE：CPU 检测到非致命性运行时编程错误。
Int_Enable	SM4.4	TRUE：存在已启用中断
Xmit0_Idle	SM4.5	TRUE：端口 0 发送器空闲（FALSE：正在发送）。
Xmit1_Idle	SM4.6	TRUE：端口 1 发送器空闲（FALSE：正在发送）。
Force_On	SM4.7	TRUE：存在强制 PLC 存储器

** 只能在中断例程中使用状态位 SM4.0、SM4.1 和 SM4.2。如果 CPU 清空中断队列，并将控制权返还给主程序，则 CPU 会将这些状态位复位。

表格 D-6 SM20.0 系统状态

S7-200 SMART 变量名	SM 地址	描述
Xmit2_Idle	SM20.0	TRUE：端口 2 发送器空闲（FALSE：正在发送）。

D.7 SMB5： I/O 错误状态

特殊存储器字节 5 (SM5.0 - SM5.7) 包含一个状态位，用于指示 I/O 系统中的错误条件。

表格 D-7 SMB5 I/O 错误状态

S7-200 SMART 变量名	SM 地址	描述
IO_Err	SM5.0	如果存在任何 I/O 错误，该位将接通。

D.8 SMB6-SMB7： CPU ID、错误状态和数字量 I/O 点

特殊存储器字节 6 和 7 提供 CPU 信息。

S7-200 SMART 变量名	SM 地址	只读 SMB6 和 SMB7（CPU ID、错误状态和数字量 I/O 点）								
CPU_ID	SMB6	MSB							LSB	
		7							0	
		x	x	x	x	c	d	0	0	
	SM6.4 到 SM6.6	1	0	0	0	= CPU CR20s				
		1	0	0	1	= CPU CR40s				
		1	0	1	0	= CPU CR60s				
		1	0	1	1	= CPU SR20 / ST20				
		1	1	0	0	= CPU SR40 / ST40				
		1	1	0	1	= CPU SR60 / ST60				
		1	1	1	0	= CPU CR30s				
		1	1	1	1	= CPU SR30 / ST30				
		0	0	0	0	= CPU ST32				
	SM6.2 到 SM6.3					c				组态/参数化错误 0 = 无错误，1 = 错误
							d			诊断报警（请参见 SMW100 了解报警代码） 0 = 无错误，1 = 错误

S7-200 SMART 变量名	SM 地址	只读 SMB6 和 SMB7（CPU ID、错误状态和数字量 I/O 点）								
CPU_IO	SMB7	MSB							LSB	
		7							0	
		i	i	i	i	q	q	q	q	
	SM7.0 到 SM7.7	i	i	i	i					0 到 15 字节数字量输入点
						q	q	q	q	0 到 15 字节数字量输出点

另请参见 SMW100-SMW114 系统报警代码 (页 1296)

D.9 SMB8-SMB19、SMB131-SMB134、SMB2030-SMB2045：I/O 模块 ID 和错误

SMB8 至 SMB19 和 SMB131 至 SMB134 以字节对的形式组织，用于扩展模块 0 至 15。

每对字节的偶数字节是模块标识寄存器。这些字节标识模块类型、I/O 类型以及输入和输出点数。

每对字节的奇数字节是模块错误寄存器。这些字节提供在该模块 I/O 中检测到的任何错误。

S7-200 SMART 变量名	SM 地址	只读 SMB8-SMB21 模块 ID 和错误数据
EM0_ID	SMB8	扩展模块 0 ID 寄存器
EM0_Err	SMB9	扩展模块 0 错误寄存器（有关诊断报警代码，见 SMW104）
EM1_ID	SMB10	扩展模块 1 ID 寄存器
EM1_Err	SMB11	扩展模块 1 错误寄存器（有关诊断报警代码，见 SMW106）
EM2_ID	SMB12	扩展模块 2 ID 寄存器
EM2_Err	SMB13	扩展模块 2 错误寄存器（有关诊断报警代码，见 SMW108）
EM3_ID	SMB14	扩展模块 3 ID 寄存器
EM3_Err	SMB15	扩展模块 3 错误寄存器（有关诊断报警代码，见 SMW110）
EM4_ID	SMB16	扩展模块 4 ID 寄存器
EM4_Err	SMB17	扩展模块 4 错误寄存器（有关诊断报警代码，见 SMW112）
EM5_ID	SMB18	扩展模块 5 ID 寄存器

EM5_Err	SMB19	扩展模块 5 错误寄存器（有关诊断报警代码，见 SMW114）
EM6_ID	SMB131	扩展模块 6 ID 寄存器
EM6_Err	SMB132	扩展模块 6 错误寄存器（有关诊断报警代码，见 SMW116）
EM7_ID	SMB133	扩展模块 7 ID 寄存器
EM7_Err	SMB134	扩展模块 7 错误寄存器（有关诊断报警代码，见 SMW118）
EM8_ID	SMB2030	扩展模块 8 ID 寄存器
EM8_Err	SMB2031	扩展模块 8 错误寄存器（有关诊断报警代码，见 SMW2014）
EM9_ID	SMB2032	扩展模块 9 ID 寄存器
EM9_Err	SMB2033	扩展模块 9 错误寄存器（有关诊断报警代码，见 SMW2016）
EM10_ID	SMB2034	扩展模块 10 ID 寄存器
EM10_Err	SMB2035	扩展模块 10 错误寄存器（有关诊断报警代码，见 SMW2018）
EM11_ID	SMB2036	扩展模块 11 ID 寄存器
EM11_Err	SMB2037	扩展模块 11 错误寄存器（有关诊断报警代码，见 SMW2020）
EM12_ID	SMB2038	扩展模块 12 ID 寄存器
EM12_Err	SMB2039	扩展模块 12 错误寄存器（有关诊断报警代码，见 SMW2022）
EM13_ID	SMB2040	扩展模块 13 ID 寄存器
EM13_Err	SMB2041	扩展模块 13 错误寄存器（有关诊断报警代码，见 SMW2024）
EM14_ID	SMB2042	扩展模块 14 ID 寄存器
EM14_Err	SMB2043	扩展模块 14 错误寄存器（有关诊断报警代码，见 SMW2026）
EM15_ID	SMB2044	扩展模块 15 ID 寄存器
EM15_Err	SMB2045	扩展模块 15 错误寄存器（有关诊断报警代码，见 SMW2028）

	偶数字节 I/O 模块 ID 寄存器									奇数字节 I/O 模块错误寄存器									
	MSB								LSB		MSB								LSB
	7							0			7							0	
	m	0	0	a	i	i	q	q			c	d	0	b	0	0	0	m	
m：存在模块	0	= 存在								c：	0	无错误							
	1	= 不存在									1	组态/参数化错误							
										d：		0	无错误						
												1	诊断报警						

a：I/O 类型	0	= 数字量								
	1	= 模拟量			b：		0	无错误		
ii：输入		0	0	= 无输入			1	总线访问错误		
		0	1	= 2 AI 或 8 DI						
		1	0	= 4 AI 或 16 DI						
		1	1	= 8 AI 或 32 DI						
qq：输出			0	0	= 无输出	m：			0	正常
			0	1	= 2 AQ 或 8 DQ				1	组态模块丢失
			1	0	= 4 AQ 或 16 DQ					
			1	1	= 8 AQ 或 32 DQ					

另请参见 SMW100-SMW114 系统报警代码 (页 1296)

D.10 SMW22-SMW26：扫描时间

SMW22、SMW24 和 SMW26 包含扫描时间信息。可读取上次扫描时间、最小扫描时间和最大扫描时间（毫秒值）。

表格 D-8 SMW22-SMW26 PLC 扫描时间

S7-200 SMART 变量名	SM 地址	描述
Last_Scan	SMW22	最后一次扫描的扫描时间。
Minimum_Scan	SMW24	自进入 RUN 模式起或者自从通过“PLC 信息”(PLC Information) 对话框复位这些值起所记录的最小扫描时间。复位后，该值将变为 65535。
Maximum_Scan	SMW26	自进入 RUN 模式起或自从通过“PLC 信息”(PLC Information) 对话框复位这些值起所记录的最大扫描时间。复位后，该值将变为 0。

D.11 SMB28-SMB29、SMB32-SMB33：信号板 ID 和错误

SMB28-SMB29 字节地址存储信号板 0 类型和错误状态。

S7-200 SMART 变量名	SM 地址	只读 SMB28-SMB29 信号板 ID 和错误数据
SB_ID	SMB28	信号板 ID 寄存器
SB_Err	SMB29	信号板错误寄存器（有关诊断报警代码，见 SMW102）

<table>
<tr><td></td><td colspan="9">SMB28 信号板 ID 寄存器</td><td colspan="9">SMB29 信号板错误寄存器</td></tr>
<tr><td></td><td colspan="9">MSB LSB</td><td colspan="9">MSB LSB</td></tr>
<tr><td></td><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td><td>0</td><td></td><td></td><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td><td>0</td><td></td></tr>
<tr><td></td><td>m</td><td>0</td><td>0</td><td>a</td><td>i</td><td>i</td><td>q</td><td>q</td><td></td><td></td><td>c</td><td>d</td><td>0</td><td>b</td><td>0</td><td>0</td><td>0</td><td>m</td><td></td></tr>
<tr><td>m：存在模块</td><td>0</td><td colspan="8">= 存在</td><td>c：</td><td>0</td><td colspan="7">无错误</td></tr>
<tr><td></td><td>1</td><td colspan="8">= 不存在</td><td></td><td>1</td><td colspan="7">组态/参数化错误</td></tr>
<tr><td colspan="4"></td><td colspan="5"></td><td>d：</td><td>0</td><td colspan="7">无错误</td></tr>
<tr><td colspan="4"></td><td colspan="5"></td><td></td><td>1</td><td colspan="7">诊断报警</td></tr>
<tr><td colspan="4" rowspan="2">a：I/O 类型</td><td>0</td><td colspan="4">= 数字量</td><td colspan="3"></td><td colspan="6"></td></tr>
<tr><td>1</td><td colspan="4">= 模拟量</td><td colspan="3">b：</td><td>0</td><td colspan="6">无错误</td></tr>
<tr><td colspan="5" rowspan="4">ii：输入</td><td>0</td><td>0</td><td colspan="3">= 无输入</td><td colspan="3"></td><td>1</td><td colspan="6">总线访问错误</td></tr>
<tr><td>0</td><td>1</td><td colspan="3">= 2 AI 或 8 DI</td><td colspan="3" rowspan="3"></td><td colspan="3" rowspan="3"></td><td colspan="3"></td></tr>
<tr><td>1</td><td>0</td><td colspan="3">= 4 AI 或 16 DI</td><td colspan="3"></td></tr>
<tr><td>1</td><td>1</td><td colspan="3">= 8 AI 或 32 DI</td><td colspan="3"></td></tr>
<tr><td colspan="6" rowspan="4">qq：输出</td><td>0</td><td>0</td><td colspan="3">= 无输出</td><td colspan="6" rowspan="2">m：</td><td>0</td><td colspan="2">正常</td></tr>
<tr><td>0</td><td>1</td><td colspan="3">= 2 AQ 或 8 DQ</td><td>1</td><td colspan="2">组态模块丢失</td></tr>
<tr><td>1</td><td>0</td><td colspan="3">= 4 AQ 或 16 DQ</td><td colspan="9" rowspan="2"></td></tr>
<tr><td>1</td><td>1</td><td colspan="3">= 8 AQ 或 32 DQ</td></tr>
</table>

SMB32-SMB33 字节地址存储信号板 1 类型和错误状态。

S7-200 SMART 变量名	SM 地址	只读 SMB32-SMB33 信号板 ID 和错误数据
SB_ID	SMB32	信号板 ID 寄存器
SB_Err	SMB33	信号板错误寄存器（有关诊断报警代码，见 SMW102）

<table>
<tr><td rowspan="2"></td><td colspan="9">SMB32 信号板 ID 寄存器</td><td colspan="9">SMB33 信号板错误寄存器</td></tr>
<tr><td colspan="9">MSB LSB</td><td colspan="9">MSB LSB</td></tr>
<tr><td></td><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td><td>0</td><td></td><td></td><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td><td>0</td><td></td></tr>
<tr><td></td><td>m</td><td>0</td><td>0</td><td>a</td><td>i</td><td>i</td><td>q</td><td>q</td><td></td><td></td><td>c</td><td>d</td><td>0</td><td>b</td><td>0</td><td>0</td><td>0</td><td>m</td><td></td></tr>
<tr><td>m：存在模块</td><td>0</td><td colspan="8">= 存在</td><td>c：</td><td>0</td><td colspan="6">无错误</td></tr>
<tr><td></td><td>1</td><td colspan="8">= 不存在</td><td></td><td>1</td><td colspan="6">组态/参数化错误</td></tr>
<tr><td colspan="3"></td><td colspan="6"></td><td colspan="2" rowspan="2">d：</td><td>0</td><td colspan="5">无错误</td></tr>
<tr><td colspan="3"></td><td colspan="6"></td><td>1</td><td colspan="5">诊断报警</td></tr>
<tr><td colspan="3" rowspan="2">a：I/O 类型</td><td>0</td><td colspan="5">= 数字量</td><td colspan="2"></td><td colspan="6"></td></tr>
<tr><td>1</td><td colspan="5">= 模拟量</td><td colspan="2" rowspan="2">b：</td><td>0</td><td colspan="5">无错误</td></tr>
<tr><td colspan="3" rowspan="4">ii：输入</td><td>0</td><td>0</td><td colspan="4">= 无输入</td><td>1</td><td colspan="5">总线访问错误</td></tr>
<tr><td>0</td><td>1</td><td colspan="4">= 2 AI 或 8 DI</td><td colspan="4" rowspan="3"></td><td colspan="3" rowspan="3"></td><td></td></tr>
<tr><td>1</td><td>0</td><td colspan="4">= 4 AI 或 16 DI</td><td></td></tr>
<tr><td>1</td><td>1</td><td colspan="4">= 8 AI 或 32 DI</td><td></td></tr>
<tr><td colspan="3" rowspan="4">qq：输出</td><td>0</td><td>0</td><td colspan="4">= 无输出</td><td colspan="4" rowspan="2">m：</td><td>0</td><td>正常</td></tr>
<tr><td>0</td><td>1</td><td colspan="4">= 2 AQ 或 8 DQ</td><td>1</td><td>组态模块丢失</td></tr>
<tr><td>1</td><td>0</td><td colspan="4">= 4 AQ 或 16 DQ</td><td colspan="6" rowspan="2"></td></tr>
<tr><td>1</td><td>1</td><td colspan="4">= 8 AQ 或 32 DQ</td></tr>
</table>

另请参见 SMW100-SMW110 系统报警代码 (页 1296)

D.12 SMB30：（端口 0）、SMB130：（端口 1）和 SMB31：（端口 2）

SMB30 组态端口 0（板载 RS485 端口）。

SMB130 组态端口 1（可选 CM01 信号板）。

SMB31 组态端口 2（可选 CM01 信号板）。

可对 SMB30、SMB31 和 SMB130 进行读取和写入操作。这些字节配置相应通信端口进行自由端口操作，并提供自由端口或系统协议支持的选择。

S7-200 SMART 变量名	**SM 地址**			位格式	**位格式**								
					MSB								LSB
	端口 0	**端口 1**	**端口 2**		7							0	
P0_Config	**SMB30**				p	p	d	b	b	b	m	m	
P1_Config		**SMB130**											
P2_Config			**SMB31**										
	SM30.6 - SM30.7	SM130.6 - SM130.7	SM31.6 - SM31.7	pp：	0	0	= 无奇偶校验						
					0	1	= 偶校验						
					1	0	= 无奇偶校验						
					1	1	= 奇校验						
	SM30.5	SM130.5	SM31.5	d：	0	= 8 个数据位/字符							
					1	= 7 个数据位/字符							
	SM30.2 - SM30.4	SM130.2 - SM130.4	SM31.2 - SM31.4	bbb：	0	0	0	= 38,400 bps					
					0	0	1	= 19,200 bps					
					0	1	0	= 9,600 bps					
					0	1	1	= 4,800 bps					
					1	0	0	= 2,400 bps					
					1	0	1	= 1,200 bps					
					1	1	0	= 115,200 bps					
					1	1	1	= 57,600 bps					

P0_Config_0	SM30.0			mm：	0	0	= PPI 从站模式
P1_Config_1		SM130.0					
P1_Config_2			SM31.0		0	1	= 自由端口协议
	SM30.1	SM130.1	SM31.1		1	0	= 保留（默认为 PPI 从站模式）
					1	1	= 保留（默认为 PPI 从站模式）
					注：在 PPI 模式下位 2 到 7 被忽略		

D.13 SMB34-SMB35： 定时中断的时间间隔

特殊存储器字节 34 和 35 控制定时中断 0 和 1 (页 415) 的时间间隔。可以指定从 1 ms 到 255 ms 的时间间隔（以 1 ms 为增量）。当 CPU 将中断例程连接到对应的定时中断事件时，CPU 捕获时间间隔值。要更改时间间隔，必须将定时中断事件重新连接到相同或不同的中断例程。可以通过分离事件来终止定时中断事件。

表格 D-9 SMB34-SMB35 定时中断时间间隔

S7-200 SMART 变量名	SM 地址	描述
Time_0_Intrvl	SMB34	定时中断 0：时间间隔值（增量为 1 ms，取值范围是 1 ms 到 255 ms）。
Time_1_Intrvl	SMB35	定时中断 1：时间间隔值（增量为 1 ms，取值范围是 1 ms 到 255 ms）。

D.14 SMB36-SMB45 (HSC0)、SMB46-SMB55 (HSC1)、SMB56-SMB65 (HSC2)、SMB136-SMB145 (HSC3)、SMB146-SMB155 (HSC4)、SMB156-SMB165 (HSC5)、SMB360-SMB369 (HSC6)、SMB370-SMB379 (HSC7)：高速计数器

以下字节为高速计数器提供组态和操作信息：

- HSC0
- HSC1
- HSC2

D.14 SMB36-SMB45 (HSC0)、SMB46-SMB55 (HSC1)、SMB56-SMB65 (HSC2)、SMB136-SMB145 (HSC3)、SMB146-SMB155 (HSC4)、SMB156-SMB165 (HSC5)、SMB360-SMB369 (HSC6)、SMB370-SMB379 (HSC7)：高速计数器

- HSC3
- HSC4
- HSC5
- HSC6
- HSC7

SM 定义	S7-200 SMART 变量名	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7
HSC* 计数器状态 **注**：仅当 CPU 正在执行高速计数器事件所触发的中断例程时，计数器状态位才有效。	**HSC*_Status**								
当前计数方向状态位： • 0 = 减计数 • 1 = 加计数	HSC*_Status_5	SM36.5	SM46.5	SM56.5	SM136.5	SM146.5	SM156.5	SM360.5	SM370.5
当前值等于预设值状态位： • 0 = 不相等 • 1 = 相等	HSC*_Status_6	SM36.6	SM46.6	SM56.6	SM136.6	SM146.6	SM156.6	SM360.6	SM370.6
当前值大于预设值状态位： • 0 = 小于或等于 • 1 = 大于	HSC*_Status_7	SM36.7	SM46.7	SM56.7	SM136.7	SM146.7	SM156.7	SM360.7	SM370.7
HSC0 计数器控制	**HSC*_Ctrl**								
复位的触发电平控制位： • 0 = 高电平激活时复位 • 1 = 低电平激活时复位	HSC*_Reset_Level	SM37.0	SM47.0	SM57.0	SM137.0	SM147.0	SM157.0	SM361.0	SM371.0
正交计数器的计数速率选择： • 0 = 4x 计数速率 • 1 = 1x 计数速率	HSC*_Rate	SM37.2	SM47.2	SM57.2	SM137.2	SM147.2	SM157.2	SM361.2	SM371.2

D.14 SMB36-SMB45 (HSC0)、SMB46-SMB55 (HSC1)、SMB56-SMB65 (HSC2)、SMB136-SMB145 (HSC3)、SMB146-SMB155 (HSC4)、SMB156-SMB165 (HSC5)、SMB360-SMB369 (HSC6)、SMB370-SMB379 (HSC7)：高速计数器

SM 定义	S7-200 SMART 变量名	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7
方向控制位： • 0 = 减计数 • 1 = 加计数	HSC*_Dir	SM37.3	SM47.3	SM57.3	SM137.3	SM147.3	SM157.3	SM361.3	SM371.3
更新方向： • 0 = 无更新方向 • 1 = 更新方向	HSC*_Dir_Update	SM37.4	SM47.4	SM57.4	SM137.4	SM147.4	SM157.4	SM361.4	SM371.4
更新预设值： • 0 = 未将新预设值写入 HSC* 预设值 • 1：将新预设值写入 HSC* 预设值	HSC*_PV_Update	SM37.5	SM47.5	SM57.5	SM137.5	SM147.5	SM157.5	SM361.5	SM371.5
更新当前值： • 0 = 未将新预设值写入 HSC* 当前值 • 1：将新预设值写入 HSC* 当前值	HSC*_CV_Update	SM37.6	SM47.6	SM57.6	SM137.6	SM147.6	SM157.6	SM361.6	SM371.6
使能位： • 0 = 禁用 • 1 = 启用	HSC*_Enable	SM37.7	SM47.7	SM57.7	SM137.7	SM147.7	SM157.7	SM361.7	SM371.7

SM 定义	S7-200 SMART 变量名	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7
HSC* 新当前值 例如，SMD38 用于将 HSC0 当前值设置为所选的任何值。要更新当前值，可将新当前值写入 SMD38，将 1 写入 SM37.6，并执行该 HSC 指令。然后，该指令将新的当前值写入 HSC0 的当前计数寄存器中。	HSC*_CV	SMD38	SMD48	SMD58	SMD138	SMD148	SMD158	SMD362	SMD372
HSC* 新预设值 例如，SMD42 用于将 HSC0 预设值设置为所选的任何值。要更新预设值，可将新预设值写入 SMD42，将 1 写入 SM37.5，并执行该 HSC 指令。然后，该指令将新的预设值写入 HSC0 的预设寄存器中。	HSC*_PV	SMD42	SMD52	SMD62	SMD142	SMD152	SMD162	SMD366	SMD376

D.15 SMB66-SMB85 (PTO0/PWM0, PTO1/PWM1)、SMB166-SMB169 (PTO0)、SMB176-SMB179 (PTO1)、SMB566-SMB579 (PTO2/PWM2)、SMB580-SMB593 (PTO3/PWM3)、SMB2000-SMB2013 (PTO4/PWM4)：高速输出

S7-200 SMART CPU 使用以下特殊存储器监视和控制 PLS（脉冲）指令的脉冲串输出（PTO0 和 PTO1）和脉宽调制输出（PWM0 和 PWM1）：

CPU 使用 SMB566-SMB579 监视和控制脉冲串输出 PTO2 和脉宽调制输出 PWM2。

CPU 使用 SMB580-SMB593 监视和控制脉冲串输出 PTO3 和脉宽调制输出 PWM3。

D.15 SMB66-SMB85 (PTO0/PWM0, PTO1/PWM1)、SMB166-SMB169 (PTO0)、SMB176-SMB179 (PTO1)、SMB566-SMB579 (PTO2/PWM2)、SMB580-SMB593 (PTO3/PWM3)、SMB2000-SMB2013 (PTO4/PWM4)：高速输出

CPU 使用 SMB2000-SMB2013 监视和控制脉冲串输出 PTO4 和脉宽调制输出 PWM4。

表格 D-10　高速输出 0 组态和控制

S7-200 SMART 变量名	SM 地址	功能
PTO0_Status	**SMB66**	**PTO0 状态**
PLS0_Abort_AE	SM66.4	PTO0 包络因相加错误而中止：FALSE：不中止；TRUE：中止
PLS0_Disable_UC	SM66.5	PTO0 用户手动在 PTO 包络运行期间手动将其禁止： • FALSE：未禁用 • TRUE：手动禁用
PLS0_Ovr	SM66.6	PTO0 管道上溢/下溢，管道已满时装载管道或传输空管道： • FALSE：无上溢 • TRUE：管道上溢/下溢
PLS0_Idle	SM66.7	PTO0 空闲： • FALSE：PTO 正在进行 • TRUE：PTO 空闲
PLS0_Ctrl	**SMB67**	**为 Q0.0 监视和控制 PTO0（脉冲串输出）及 PWM0（脉宽调制）**
PLS0_Cycle_Update	SM67.0	PTO0/PWM0 更新周期时间或频率值： • FALSE：无更新 • TRUE：写入新周期/频率
PWM0_PW_Update	SM67.1	PWM0 更新脉宽值： • FALSE：无更新 • TRUE：写入新脉冲宽度
PTO0_PC_Update	SM67.2	PTO0 更新脉冲计数值： • FALSE：无更新 • TRUE：写入新脉冲计数
PWM0_TimeBase	SM67.3	PWM0 时基： • FALSE：1 μs/刻度 • TRUE：1 ms/刻度
PTO0_Override	SM67.4	PTO0 超控： • FALSE：禁止 • TRUE：启用
PTO0_Operation	SM67.5	PTO0 选择单/多段操作： • FALSE：单段 • TRUE：多段

D.15 SMB66-SMB85 (PTO0/PWM0, PTO1/PWM1)、SMB166-SMB169 (PTO0)、SMB176-SMB179 (PTO1)、SMB566-SMB579 (PTO2/PWM2)、SMB580-SMB593 (PTO3/PWM3)、SMB2000-SMB2013 (PTO4/PWM4)：高速输出

S7-200 SMART 变量名	SM 地址	功能
PLS0_Select	SM67.6	PTO0/PWM0 模式选择： • FALSE：PWM • TRUE：PTO
PLS0_Enable	SM67.7	PTO0/PWM0 使能： • FALSE：禁用 • TRUE：启用
PLS0_Cycle	**SMW68**	**字数据类型：PWM0 周期时间值（2 到 65,535 单位的时基）；PTO0 频率值（1 到 65,535 Hz）**
PWM0_PW	**SMW70**	**字数据类型：PWM0 脉宽值（0 到 65,535 单位的时基）**
PTO0_PC	**SMD72**	**双字数据类型：PTO0 脉冲计数值（1 到 2,147,483,647）**
PTO0_Seg_Num	**SMB166**	**字节数据类型：PTO0 包络中当前执行的段号**
PTO0_Profile_Offset	**SMW168**	**字数据类型：PTO0 包络表的起始单元（相对于 V0 的字节偏移量）**

表格 D-11　高速输出 1 组态和控制

S7-200 SMART 变量名	SM 地址	功能
PTO1_Status	**SMB76**	**PTO1 状态**
PLS1_Abort_AE	SM76.4	PTO1 包络因相加错误而中止： • FALSE：不中止 • TRUE：中止
PLS1_Disable_UC	SM76.5	PTO1 用户手动在 PTO 包络运行期间手动将其禁止： • FALSE：未禁用 • TRUE：手动禁用
PLS1_Ovr	SM76.6	PTO1 管道上溢/下溢，管道已满时装载管道或传输空管道： • FALSE：无上溢 • TRUE：管道上溢/下溢
PLS1_Idle	SM76.7	PTO1 空闲： • FALSE：PTO 正在进行 • TRUE：PTO 空闲
PLS1_Ctrl	**SMB77**	**为 Q0.1 监视和控制 PTO1（脉冲串输出）及 PWM1（脉宽调制）**

D.15 SMB66-SMB85 (PTO0/PWM0, PTO1/PWM1)、SMB166-SMB169 (PTO0)、SMB176-SMB179 (PTO1)、SMB566-SMB579 (PTO2/PWM2)、SMB580-SMB593 (PTO3/PWM3)、SMB2000-SMB2013 (PTO4/PWM4)：高速输出

S7-200 SMART 变量名	SM 地址	功能
PLS1_Cycle_Update	SM77.0	PTO1/PWM1 更新周期时间或频率值： • FALSE：无更新 • TRUE：写入新周期/频率
PWM1_PW_Update	SM77.1	PWM1 更新脉宽值： • FALSE：无更新 • TRUE：写入新脉冲宽度
PTO1_PC_Update	SM77.2	PTO1 更新脉冲计数值： • FALSE：无更新 • TRUE：写入新脉冲计数
PWM1_TimeBase	SM77.3	PWM1 时基： • FALSE：1 μs/刻度 • TRUE：1 ms/刻度
PTO1_Override	SM77.4	PTO1 超控： • FALSE：禁止 • TRUE：启用
PTO1_Operation	SM77.5	PTO1 选择单/多段操作： • FALSE：单段 • TRUE：多段
PLS1_Select	SM77.6	PTO1/PWM1 模式选择： • FALSE：PWM • TRUE：PTO
PLS1_Enable	SM77.7	PTO1/PWM1 使能： • FALSE：禁用 • TRUE：启用
PLS1_Cycle	**SMW78**	**字数据类型：PWM1 周期时间值（2 到 65,535 单位的时基）；PTO1 频率值（1 到 65,535 Hz）**
PWM1_PW	**SMW80**	**字数据类型：PWM1 脉宽值（0 到 65,535 单位的时基）**
PTO1_PC	**SMD82**	**双字数据类型：PTO1 脉冲计数值（1 到 2,147,483,647）**
PTO1_Seg_Num	**SMB176**	**字节数据类型：PTO1 包络中当前执行的段号**
PTO1_Profile_Offset	**SMW178**	**字数据类型：PTO1 包络表的起始单元（相对于 V0 的字节偏移量）**

表格 D-12　高速输出 2 组态和控制

S7-200 SMART 变量名	SM 地址	描述
PTO2_Status	**SMB566**	**PTO2 状态**
PLS2_Abort_AE	SM566.4	PTO2 包络因相加错误而中止： • FALSE：不中止 • TRUE：已中止
PLS2_Disable_UC	SM566.5	PTO2 用户手动在 PTO 包络运行期间手动将其禁止： • FALSE：未禁用 • TRUE：手动禁用
PLS2_Ovr	SM566.6	PTO2 管道上溢/下溢，管道已满时装载管道或传输空管道： • FALSE：无上溢 • TRUE：管道上溢/下溢
PLS2_Idle	SM566.7	PTO2 空闲： • FALSE：PTO 正在进行 • TRUE：PTO 空闲
PLS2_Ctrl	**SMB567**	**为 Q0.3 监视和控制 PTO2（脉冲串输出）及 PWM2（脉宽调制）**
PLS2_Cycle_Update	SM567.0	PTO2/PWM2 更新周期时间或频率值： • FALSE：无更新 • TRUE：写入新周期/频率
PWM2_PW_Update	SM567.1	PWM2 更新脉宽值： • FALSE：无更新 • TRUE：写入新脉冲宽度
PTO2_PC_Update	SM567.2	PTO2 更新脉冲计数值： • FALSE：无更新 • TRUE：写入新脉冲计数
PLS2_TimeBase	SM567.3	PWM2 时基： • FALSE：1 μs/刻度 • TRUE：1 ms/刻度
PTO2_Override	SM567.4	PTO2 超控： • FALSE：禁止 • TRUE：启用
PTO2_Operation	SM567.5	PTO2 选择单/多段操作： • FALSE：单段 • TRUE：多段

D.15 SMB66-SMB85 (PTO0/PWM0, PTO1/PWM1)、SMB166-SMB169 (PTO0)、SMB176-SMB179 (PTO1)、SMB566-SMB579 (PTO2/PWM2)、SMB580-SMB593 (PTO3/PWM3)、SMB2000-SMB2013 (PTO4/PWM4)：高速输出

S7-200 SMART 变量名	SM 地址	描述
PLS2_Select	SM567.6	PTO2/PWM2 模块选择： • FALSE：PWM • TRUE：PTO
PLS2_Enable	SM567.7	PTO2/PWM2 使能： • FALSE：禁用 • TRUE：启用
PLS2_Cycle	**SMW568**	**字数据类型：PWM2 周期时间值（2 到 65,535 单位的时基）；PTO2 频率值（1 到 65,535 Hz）**
PWM2_PW	**SMW570**	**字数据类型：PWM2 脉宽值（0 到 65,535 单位的时基）**
PTO2_PC	**SMD572**	**双字数据类型：PTO2 脉冲计数值（1 到 2,147,483,647）**
PTO2_Seg_Num	**SMB576**	**字节数据类型：PTO2 包络中当前执行的段号**
PTO2_Profile_Offset	**SMW578**	**字数据类型：PTO2 包络表的起始单元（相对于 V0 的字节偏移量）**

表格 D-13　高速输出 3 组态和控制

S7-200 SMART 变量名	SM 地址	描述
PTO3_Status	**SMB580**	**PTO3 状态**
PLS3_Abort_AE	SM580.4	PTO3 包络因相加错误而中止： • FALSE：不中止 • TRUE：已中止
PLS3_Disable_UC	SM580.5	PTO3 用户手动在 PTO 包络运行期间手动将其禁止： • FALSE：未禁用 • TRUE：手动禁用
PLS3_Ovr	SM580.6	PTO3 管道上溢/下溢，管道已满时装载管道或传输空管道： • FALSE：无上溢 • TRUE：管道上溢/下溢
PLS3_Idle	SM580.7	PTO3 空闲： • FALSE：PTO 正在进行 • TRUE：PTO 空闲
PLS3_Ctrl	**SMB581**	**为 Q0.2 监视和控制 PTO3（脉冲串输出）及 PWM3（脉宽调制）**

D.15 SMB66-SMB85 (PTO0/PWM0, PTO1/PWM1)、SMB166-SMB169 (PTO0)、SMB176-SMB179 (PTO1)、SMB566-SMB579 (PTO2/PWM2)、SMB580-SMB593 (PTO3/PWM3)、SMB2000-SMB2013 (PTO4/PWM4)：高速输出

S7-200 SMART 变量名	SM 地址	描述
PLS3_Cycle_Update	SM581.0	PTO3/PWM3 更新周期时间或频率值： • FALSE：无更新 • TRUE：写入新周期/频率
PWM3_PW_Update	SM581.1	PWM3 更新脉宽值： • FALSE：无更新 • TRUE：写入新脉冲宽度
PTO3_PC_Update	SM581.2	PTO3 更新脉冲计数值： • FALSE：无更新 • TRUE：写入新脉冲计数
PLS3_TimeBase	SM581.3	PWM3 时基： • FALSE：1 μs/刻度 • TRUE：1 ms/刻度
PTO3_Override	SM581.4	PTO3 超控： • FALSE：禁止 • TRUE：启用
PTO3_Operation	SM581.5	PTO3 选择单/多段操作： • FALSE：单段 • TRUE：多段
PLS3_Select	SM581.6	PTO3/PWM3 模式选择： • FALSE：PWM • TRUE：PTO
PLS3_Enable	SM581.7	PTO3/PWM3 使能： • FALSE：禁用 • TRUE：启用
PLS3_Cycle	**SMW582**	**字数据类型：PWM3 周期时间值（2 到 65,535 单位的时基）；PTO3 频率值（1 到 65,535 Hz）**
PWM3_PW	**SMW584**	**字数据类型：PWM3 脉宽值（0 到 65,535 单位的时基）**
PTO3_PC	**SMD586**	**双字数据类型：PTO3 脉冲计数值（1 到 2,147,483,647）**
PTO3_Seg_Num	**SMB590**	**字节数据类型：PTO3 包络中当前执行的段号**
PTO3_Profile_Offset	**SMW592**	**字数据类型：PTO3 包络表的起始单元（相对于 V0 的字节偏移量）**

D.15 SMB66-SMB85 (PTO0/PWM0, PTO1/PWM1)、SMB166-SMB169 (PTO0)、SMB176-SMB179 (PTO1)、SMB566-SMB579 (PTO2/PWM2)、SMB580-SMB593 (PTO3/PWM3)、SMB2000-SMB2013 (PTO4/PWM4)：高速输出

表格 D-14　高速输出 4 组态和控制

S7-200 SMART 变量名	SM 地址	描述
PTO4_Status	**SMB2000**	**PTO4 状态**
PLS4_Abort_AE	SM2000.4	PTO4 包络因相加错误而中止： • FALSE：不中止 • TRUE：已中止
PLS4_Disable_UC	SM2000.5	PTO4 用户手动在 PTO 包络运行期间手动将其禁止： • FALSE：未禁用 • TRUE：手动禁用
PLS4_Ovr	SM2000.6	PTO4 管道上溢/下溢，管道已满时装载管道或传输空管道： • FALSE：无上溢 • TRUE：管道上溢/下溢
PLS4_Idle	SM2000.7	PTO4 空闲： • FALSE：PTO 正在进行 • TRUE：PTO 空闲
PLS4_Ctrl	**SMB2001**	**为 Q0.4 监视和控制 PTO4（脉冲串输出）及 PWM4（脉宽调制）**
PLS4_Cycle_Update	SM2001.0	PTO4/PWM4 更新周期时间或频率值： • FALSE：无更新 • TRUE：写入新周期/频率
PWM4_PW_Update	SM2001.1	PWM4 更新脉宽值： • FALSE：无更新 • TRUE：写入新脉冲宽度
PTO4_PC_Update	SM2001.2	PTO4 更新脉冲计数值： • FALSE：无更新 • TRUE：写入新脉冲计数
PLS4_TimeBase	SM2001.3	PWM4 时基： • FALSE：1 μs/刻度 • TRUE：1 ms/刻度
PTO4_Override	SM2001.4	PTO4 超控： • FALSE：禁止 • TRUE：启用
PTO4_Operation	SM2001.5	PTO4 选择单/多段操作： • FALSE：单段 • TRUE：多段

S7-200 SMART 变量名	SM 地址	描述
PLS4_Select	SM2001.6	PTO4/PWM4 模式选择： • FALSE：PWM • TRUE：PTO
PLS4_Enable	SM2001.7	PTO4/PWM4 使能： • FALSE：禁用 • TRUE：启用
PLS4_Cycle	**SMW2002**	**字数据类型：PWM4 周期时间值（2 到 65,535 单位的时基）；PTO4 频率值（1 到 65,535 Hz）**
PWM4_PW	**SMW2004**	**字数据类型：PWM4 脉宽值（0 到 65,535 单位的时基）**
PTO4_PC	**SMD2006**	**双字数据类型：PTO4 脉冲计数值（1 到 2,147,483,647）**
PTO4_Seg_Num	**SMB2010**	**字节数据类型：PTO4 包络中当前执行的段号**
PTO4_Profile_Offset	**SMW2012**	**字数据类型：PTO4 包络表的起始单元（相对于 V0 的字节偏移量）**

D.16 SMB86-SMB94、SMB186-SMB194 和 SMB405-SMB413：接收信息控制

SMB86-SMB94、SMB405-SMB413 和 SMB186-SMB194 用于控制并读取 RCV（接收消息）指令的状态。

S7-200 SMART 变量名	SM 地址			位格式	接收消息状态字节								
					MSB								LSB
	端口 0	**端口 1**	**端口 2**		7							0	
P0_Stat_Rcv	**SMB86**				n	r	e	0	0	t	c	p	
P1_Stat_Rcv		**SMB186**											
P2_Stat_Rcv			**SMB405**										

P0_Stat_Rcv_7	SM86.7			n：	1	= 接收消息已由用户禁用命令终止 否则 n = 0
P1_Stat_Rcv_7		SM186.7				
P2_Stat_Rcv_7			SM405.7			
P0_Stat_Rcv_6	SM86.6			r：	1	= 接收消息终止：（未定义启动条件，字符计数为 0，或在传送激活的情况下执行接收消息） 否则 r = 0
P1_Stat_Rcv_6		SM186.6				
P2_Stat_Rcv_6			SM405.6			
P0_Stat_Rcv_5	SM86.5			e：	1	= 收到结束字符 否则 e = 0
P1_Stat_Rcv_5		SM186.5				
P2_Stat_Rcv_5			SM405.5			
P0_Stat_Rcv_2	SM86.2			t：	1	= 接收消息终止：定时器超时 否则 n = 0
P1_Stat_Rcv_2		SM186.2				
P2_Stat_Rcv_2			SM405.2			
P0_Stat_Rcv_1	SM86.1			c：	1	= 接收消息终止：达到最大字符计数 否则 c = 0
P1_Stat_Rcv_1		SM186.1				
P2_Stat_Rcv_1			SM405.1			

P0_Stat_Rcv_0	SM86.0			p：	1	= 接收消息因奇偶校验错误、帧错误、中断错误或溢出错误而终止 否则 p = 0
P1_Stat_Rcv_0		SM186.0				
P2_Stat_Rcv_0			SM405.0			

S7-200 SMART 变量名	**SM 地址**			位格式	**接收消息控制字节**								
					MSB							LSB	
	端口 0	**端口 1**	**端口 2**		7							0	
P0_Ctrl_Rcv	**SMB87**				en	sc	ec	il	c/m	Tmr	bk	0	
P1_Ctrl_Rcv		**SMB187**											
P2_Ctrl_Rcv			**SMB406**										
P0_Ctrl_Rcv_7	SM87.7			en：	0	= 接收消息功能禁用							
P1_Ctrl_Rcv_7		SM187.7			1	= 接收消息功能启用							
P2_Ctrl_Rcv_7			SM406.7										
P0_Ctrl_Rcv_6	SM87.6			sc：	0	= 忽略 SMB88 或 SMB188							
P1_Ctrl_Rcv_6		SM187.6			1	= 使用 SMB88 或 SMB188 的值检测消息的起始							
P2_Ctrl_Rcv_6			SM406.6										
P0_Ctrl_Rcv_5	SM87.5			ec：	0	= 忽略 SMB89 或 SMB189							
P1_Ctrl_Rcv_5		SM187.5			1	= 使用 SMB89 或 SMB189 的值检测消息的结束							
P2_Ctrl_Rcv_5			SM406.5										

P0_Ctrl_Rcv_4	SM87.4			il：	0	= 忽略 SMB90 或 SMB190
P1_Ctrl_Rcv_4		SM187.4			1	= 使用 SMB89 或 SMB189 的值检测空闲行条件
P2_Ctrl_Rcv_4			SM406.4			
P0_Ctrl_Rcv_3	SM87.3			c/m：	0	= 定时器是字符间定时器
P1_Ctrl_Rcv_3		SM187.3			1	= 定时器是消息定时器
P2_Ctrl_Rcv_3			SM406.3			
P0_Ctrl_Rcv_2	SM87.2			Tmr：	0	= 忽略 SMW92 或 SMW192
P1_Ctrl_Rcv_2		SM187.2			1	= 如果超出 SMW92 或 SMW192 中的时间段，则终止接收
P2_Ctrl_Rcv_2			SM406.2			
P0_Ctrl_Rcv_1	SM87.1				0	= 忽略断开条件
P1_Ctrl_Rcv_1		SM187.1			1	= 使用断开条件作为消息检测的起始
P2_Ctrl_Rcv_1			SM406.1			

使用消息控制字节的位定义识别信息的条件。消息控制字节还包括消息开始和消息结束条件。如果两组执行逻辑 AND（与）操作的消息开始条件为真且连续出现，则可确定消息开始。“连续”表示以下情况之一：

- 起始字符在空闲行后
- 起始字符在断行后

消息结束条件的逻辑 OR 操作可确定消息结束。以下是起始和结束条件的公式：

消息开始 = (il AND sc) OR (bk AND sc)

消息结束 = ec OR tmr OR 达到的最大字符计数

S7-200 SMART 变量名	**SM 地址**			位格式	**接收消息控制字节**								
					MSB							LSB	
	端口 0	**端口 1**	**端口 2**		7							0	
P0_Ctrl_Rcv	**SMB87**				en	sc	ec	il	c/m	Tmr	bk	0	
P1_Ctrl_Rcv		**SMB187**											
P2_Ctrl_Rcv			**SMB406**										
空闲线检测						0		1			0		SMW90 > 0
起始字符检测						1		0			0		SMW90 = 无关
断开检测						0		0			1		SMW90 = 无关
对请求的任何响应						0		1			0		SMW90 = 0（如果没有响应，可使用消息定时器终止接收）
断开和起始字符						1		0			1		SMW90 = 无关
空闲行和起始字符						1		1			0		SMW90 > 0
空闲行和起始字符（非法）						1		1			0		SMW90 = 0
注：接收因奇偶校验错误、帧错误、溢出错误或中断错误自动终止。													

S7-200 SMART变量名	**SM 地址**			
	端口 0	**端口 1**	**端口 2**	
P0_Stat_Char	**SMB88**			消息字符的开始
P1_Stat_Char		**SMB188**		
P2_Stat_Char			**SMB407**	

P0_End_Char	SMB89			消息字符的结束
P1_End_Char		SMB189		
P2_End_Char			SMB408	
P0_Idle_Time	SMW90			字数据：空闲线时间段以毫秒为单位指定。空闲线时间过后接收到的第一个字符为新消息的开始。
P1_Idle_Time		SMW190		
P2_Idle_Time			SMW409	
P0_Timeout	SMW92			字数据：字符间/消息定时器超时值以毫秒为单位指定。如果超出该时间段，接收消息终止。
P1_Timeout		SMW192		
P2_Timeout			SMW411	
P0_Max_Char	SMB94			要接收的最大字符数（1 至 255 字节）。 **注**：即使没有使用字符计数消息终止，此范围也必须设置为预期的最大缓冲区大小。
P1_Max_Char		SMB194		
P2_Max_Char			SMB413	

D.17 SMW98：扩展 I/O 总线通信错误

SMW98 提供有关扩展 I/O 总线错误的信息。

表格 D-15 SMW98 扩展 I/O 总线通信错误计数器

S7-200 SMART 变量名	SM 地址（读/写）	描述
EM_Parity_Err	SMW98	每次 CPU 在扩展 I/O 总线上检测到奇偶校验错误时，该字均会被递增。为 CPU 循环上电会清除该字，或向该字写入零。

D.18 SMW100-SMW120、SMW2014-SMW2028 系统报警

特殊存储器字 SMW100-SMW120 和 SMW2014-SMW2028 为 CPU、SB（信号板）和 EM（扩展模块）提供报警和诊断错误代码。

S7-200 SMART 变量名	SM 地址	当前诊断报警源
CPU_Alarm	SMW100	CPU 诊断报警代码
SB_Slot0_Alarm	SMW102	信号板 0 诊断报警代码
EM0_Alarm	SMW104	扩展模块总线插槽 0 诊断报警代码
EM1_Alarm	SMW106	扩展模块总线插槽 1 诊断报警代码
EM2_Alarm	SMW108	扩展模块总线插槽 2 诊断报警代码
EM3_Alarm	SMW110	扩展模块总线插槽 3 诊断报警代码
EM4_Alarm	SMW112	扩展模块总线插槽 4 诊断报警代码
EM5_Alarm	SMW114	扩展模块总线插槽 5 诊断报警代码
EM6_Alarm	SMW116	扩展模块总线插槽 6 诊断报警代码
EM7_Alarm	SMW118	扩展模块总线插槽 7 诊断报警代码
EM8_Alarm	SMW2014	扩展模块总线插槽 8 诊断报警代码
EM9_Alarm	SMW2016	扩展模块总线插槽 9 诊断报警代码
EM10_Alarm	SMW2018	扩展模块总线插槽 10 诊断报警代码
EM11_Alarm	SMW2020	扩展模块总线插槽 11 诊断报警代码
EM12_Alarm	SMW2022	扩展模块总线插槽 12 诊断报警代码
EM13_Alarm	SMW2024	扩展模块总线插槽 13 诊断报警代码
EM14_Alarm	SMW2026	扩展模块总线插槽 14 诊断报警代码
EM15_Alarm	SMW2028	扩展模块总线插槽 15 诊断报警代码
SB_Slot1_Alarm	SMW120	信号板 1 诊断报警代码

报警代码格式	MSB																LSB
	15	14	13					8	7							0	
	d	s	c	c	c	c	c	c	c	a	a	a	a	a	a	a	

d：报警方向	0	输入通道或不适用的														
	1	输出通道														
s：报警范围	0	在单个通道上														
	1	在单个通道上														
c：通道号	c	c	c	c	c	c	如果报警范围 =“在单个通道上”，则为受影响的通道数；如果报警范围 =“在整个模块上”，则为 0。									
a：报警类型							0	0	0	0	0	0	0	0	00H：无报警	
							0	0	0	0	0	0	0	1	01H：短路	
							0	0	0	0	0	x	x	x	02H 至 05H：预留	
							0	0	0	0	0	1	1	0	06H：断路	
							0	0	0	0	0	1	1	1	07H：超出上限	
							0	0	0	0	1	0	0	0	08H：超出下限	
							0	0	0	0	x	x	x	x	09H 至 0FH：预留	
							0	0	0	1	0	0	0	0	10H：参数化错误	
							0	0	0	1	0	0	0	1	11H：传感器或负载电压缺失	
							0	0	0	x	x	x	x	x	12H 至 1FH：预留	
							0	0	1	0	0	0	0	0	20H：内部错误（MID 问题）	
							0	0	1	0	0	0	0	1	21H：内部错误（IID 问题）	
							0	0	1	0	0	0	1	0	22H：预留	
							0	0	1	0	0	0	1	1	23H：组态错误	
							0	0	1	0	0	1	0	0	24H：预留	
							0	0	1	0	0	1	0	1	25H：固件损坏或缺失	
							0	0	1	0	x	x	x	x	26H 至 2AH：预留	
							0	0	1	0	1	0	1	1	2BH：电池电压低	
							x	x	x	x	x	x	x	x	2CH 至 FFH：预留	

D.19 SMB195-SMB210：HSC 类型和频率测量采样周期

SMB195 到 SMB210 被预留，用于设置 HSC 类型和频率测量采样周期。有关模块如何使用 SMB195 到 SMB210 的信息，请参见“高速输入降噪 (页 359)”

表格 D-16 专用存储器字节 SMB195 到 SMB210

SM 定义	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7
HSC 类型： • 0 = 计数模式 • 1 = 频率测量模式	SMB195	SMB197	SMB199	SMB201	SMB203	SMB205	SMB207	SMB209
HSC 频率测量采样周期，工作在频率测量模式下： • 0 = 采样周期 1 s • 1 = 采样周期 100 ms • 2 = 采样周期 10 ms	SMB196	SMB198	SMB200	SMB202	SMB204	SMB206	SMB208	SMB210

D.20 SMB220-SMB235：PID 控制错误代码

特殊存储器字节 SMB220 - SMB235 包含指示 PID 控制错误的状态位。

表格 D-17 SMB220 至 SMB 235

SM 地址	描述
SMB220 - SMB235 包含指示 PID 控制错误的状态位，均为从回路 0 到回路 15。 例如，SMB220 包含回路 0 的错误消息，SMB221 包含回路 1 的错误消息，依此类推。	• 0 - 没有错误 • 1 - PID 回路控制有错误

D.21 SMB380-SMB404：HSC 用户自定义引脚映射启用

SMB380 用于设置是想要传统的还是新的 HSC 引脚映射。如果启用了用户自定义引脚映射，则必须在适相应 SMB 中输入 HSC 的输入点。对于带输入点的 SMB381-SMB404 组态：I0.0-I0.7 对应于 16#00 - 16#07，I1.0-I1.7 对应于 16#08 - 16#0F。

SM 定义	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7
HSC 用户自定义引脚映射启用 [1] • 0 = 用户自定义引脚映射已禁用，请使用传统引脚映射 • 1 = 用户自定义引脚映射已启用	SM380.0	SM380.1	SM380.2	SM380.3	SM380.4	SM380.5		
时钟 A 的用户自定义输入点 [2]	SMB381	SMB384	SMB387	SMB390	SMB393	SMB396	SMB399	SMB402
时钟方向/时钟 B 的用户自定义输入点 [2]	SMB382	SMB385	SMB388	SMB391	SMB394	SMB397	SMB400	SMB403
预设值的用户自定义输入点 [2]	SMB383	SMB386	SMB389	SMB392	SMB395	SMB398	SMB401	SMB404

[1] HSC6 和 HSC7 不具备默认的引脚映射。使用 HSC6 和 HSC7 时，必须将该位置 1，并且必须为 HSC6 和 HSC7 分配输入点。否则，将在 PLC 信息中获得 `NF_HSC_PIN_MAP_INVALID`。

[2] 对于同一通道 HSC，时钟 A 和方向/时钟 B 只能在 IB0 中同时选择或在 IB1 中同时选择。

D.22 SMB480-SMB515： 数据日志状态

SMB480 至 SMB515 为只读特殊存储器地址，用于指示数据日志的操作状态。

S7-200 SMART 变量名	SM 地址	功能
DL0_InitResult	SMB480	数据日志 0 的初始化结果代码：在上电及下载系统块之后，CPU 执行数据日志分析。 • 00H：数据日志正常 • 01H：正在初始化 • 02H：未找到数据日志文件 • 03H：数据日志初始化出错 • 04H 至 FEH：预留 • FFH：数据日志未组态
DL1_InitResult	SMB481	数据日志 1 的初始化结果代码（结果代码请参见 SMB 480）
DL2_InitResult	SMB482	数据日志 2 的初始化结果代码（结果代码请参见 SMB 480）
DL3_InitResult	SMB483	数据日志 3 的初始化结果代码（结果代码请参见 SMB 480）
DL0_Maximum	SMW500	数据日志 0：允许的最大记录数的组态值
DL0_Current	SMW502	数据日志 0：允许的最大记录数的实际值
DL1_Maximum	SMW504	数据日志 1：允许的最大记录数的组态值
DL1_Current	SMW506	数据日志 1：允许的最大记录数的实际值
DL2_Maximum	SMW508	数据日志 2：允许的最大记录数的组态值
DL2_Current	SMW510	数据日志 2：允许的最大记录数的实际值
DL3_Maximum	SMW512	数据日志 3：允许的最大记录数的组态值
DL3_Current	SMW514	数据日志 3：允许的最大记录数的实际值

D.23 SMB600-SMB799：轴（0、1、2 和 3）开环运动控制（适用于运动控制 (V2.x)）

轴组态和控制 SM 地址

向导生成的程序代码会读写轴特殊存储器数据。

轴数据 SM 地址				轴功能
轴 0	**轴 1**	**轴 2**	**轴 3**	
SMB600 - SMB615	SMB650 - SMB665	SMB700- SMB715	SMB750- SMB765	轴名称（16 个 ASCII 字符）。第一个字符是序列中编号最小的字节。
SMB616 - SMB619	SMB666- SMB669	SMB716- SMB719	SMB766- SMB769	预留
SMW620	SMW670	SMW720	SMW770	错误代码 - 请参见运动轴错误代码

轴 0 SMB622	**轴 1 SMB672**	**轴 2 SMB722**	**轴 3 SMB772**		**MSB**							**LSB**	
					7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
					DIS	**0**	**TRIG**	**STP**	**LMT-**	**LMT+**	**RPS**	**ZP**	
SM622.7	SM672.7	SM722.7	SM772.7	**DIS**：禁用输出	0	未激活							
					1	激活							
SM622.5	SM672.5	SM722.5	SM772.5	**TRIG**：触发输入			0	未激活					
							1	激活					
SM622.4	SM672.4	SM722.4	SM772.4	**STP**：停止输入				0	未激活				
								1	激活				
SM622.3	SM672.3	SM722.3	SM772.3	**LMT-**：负向移动限制输入					0	未激活			
									1	激活			
SM622.2	SM672.2	SM722.2	SM772.2	**LMT+**：正向移动限制输入						0	未激活		
										1	激活		

轴 0 SMB622	轴 1 SMB672	轴 2 SMB722	轴 3 SMB772		MSB							LSB	
					7	6	5	4	3	2	1	0	
					DIS	0	TRIG	STP	LMT-	LMT+	RPS	ZP	
SM622.1	SM672.1	SM722.1	SM772.1	**RPS**：参考点开关输入							0	未激活	
											1	激活	
SM622.0	SM672.0	SM722.0	SM772.0	**ZP**：零脉冲输入								0	未激活
												1	激活

轴 0 SMB623	轴 1 SMB673	轴 2 SMB723	轴 3 SMB773		MSB							LSB	
					7	6	5	4	3	2	1	0	
					0	0	0	0	0	OR	R	CFG	
SM623.2	SM673.2	SM723.2	SM773.2	**OR**：目标速度超出范围						0	范围内		
										1	超出范围		
SM623.1	SM673.1	SM723.1	SM773.1	**R**：旋转方向							0	正向旋转	
											1	反向旋转	
SM623.0	SM673.0	SM723.0	SM773.0	**CFG**：轴已组态								0	未组态
												1	已组态

轴数据 SM 地址				轴功能
轴 0	轴 1	轴 2	轴 3	
SMB624	SMB674	SMB724	SMB774	**CUR_PF** 字节指示运动轴当前执行的曲线。
SMB625	SMB675	SMB725	SMB775	**CUR_STP** 字节指示运动轴当前执行的曲线。
SMD626	SMD676	SMD726	SMD776	**CUR_POS** 是指示轴当前位置的双字值。
SMD630	SMD680	SMD730	SMD780	**CUR_SPD** 是指示轴当前速度的双字值。

轴 0 SMB634	轴 1 SMB684	轴 2 SMB734	轴 3 SMB784		MSB						LSB	
					7	6	5	4	3	2	1	0
					D	Error						
SM634.7	SM684.7	SM734.7	SM784.7	**D：** Done 位	0	正在运行						
					1	操作完成（初始化期间由轴设置）						
SM634.0-SM634.6	SM684.0-SM684.6	SM734.0-SM734.6	SM784.0-SM784.6	**错误：**		请参见运动指令错误代码						

轴数据 SM 地址				轴功能
轴 0	**轴 1**	**轴 2**	**轴 3**	
SMB635-SMB645	SMB685-SMB695	SMB735-SMB745	SMB785-SMB795	预留
SMD646	SMD696	SMD746	SMD796	指向轴的组态/包络表的 V 存储器指针。指向 V 存储器以外区域的指针值无效。

D.24 SMB600-SMB649：轴 0 开环运动控制

请参见“SMB600-SMB799：轴（0、1、2 和 3）开环运动控制 (页 1301)”了解详细信息。

D.25 SMB650-SMB699：轴 1 开环运动控制

请参见“SMB600-SMB799：轴（0、1、2 和 3）开环运动控制 (页 1301)”了解详细信息。

D.26 SMB700-SMB749：轴 2 开环运动控制

请参见“SMB600-SMB799：轴（0、1、2 和 3）开环运动控制 (页 1301)”了解详细信息。

D.27 SMB750-SMB799：轴 3 开环运动控制

请参见“SMB600-SMB799：轴（0、1、2 和 3）开环运动控制 (页 1301)”了解详细信息。

D.28 SMB800-SMB858：轴组 0 开环运动控制

向导生成的程序代码会读写轴组 0 特殊存储器数据。

表格 D-18 轴组 0 SM 地址

<table>
<tr><th>SM 位</th><th colspan="8">描述</th></tr>
<tr><td>SMB800-SMB815</td><td colspan="8">轴组 0 的名称（16 个 ASCII 字符）。SMB800 是第一个字符。</td></tr>
<tr><td>SMB816-SMB819</td><td colspan="8">预留</td></tr>
<tr><td>SMW820</td><td colspan="8">错误代码 - 参见运动轴组错误代码</td></tr>
<tr><td>SMB822</td><td colspan="8">为瞬时轴组状态预留。</td></tr>
<tr><td rowspan="5">SMB823</td><td rowspan="3"></td><td>MSB</td><td colspan="6"></td><td>LSB</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Done</td><td colspan="7">轴组运动指令的错误代码。请参见轴组运动指令的错误代码。</td></tr>
<tr><td rowspan="2">Done</td><td>0</td><td colspan="7">轴组上的运动正在进行中。</td></tr>
<tr><td>1</td><td colspan="7">操作完成（初始化期间由轴组置位）。</td></tr>
<tr><td>SMB824</td><td colspan="8">当前路径</td></tr>
<tr><td>SMB825</td><td colspan="8">预留</td></tr>
<tr><td>SMB826</td><td colspan="8">当前线段</td></tr>
<tr><td>SMB827</td><td colspan="8">预留</td></tr>
<tr><td>SMD828</td><td colspan="8">轴组的当前复合速度</td></tr>
<tr><td>SMD832</td><td colspan="8">轴 X 的当前位置</td></tr>
<tr><td>SMD836</td><td colspan="8">轴 Y 的当前位置</td></tr>
<tr><td>SMD840</td><td colspan="8">轴 Z 的当前位置</td></tr>
<tr><td>SMD844</td><td colspan="8">预留</td></tr>
<tr><td>SMD848
V 存储器指针</td><td colspan="8">指向轴组的组态和路径表的 V 存储器指针。</td></tr>
<tr><td>SMB852</td><td colspan="8">轴组的缓冲区忙标志</td></tr>
<tr><td>SMB853</td><td colspan="8">正在执行或等待执行的路径 ID</td></tr>
</table>

SM 位	描述
SMB854	当前缓冲模式的线段数
SMB855	轴组状态
SMB856	轴启用状态
SMB857	上次完成的路径 ID
SMB858	上次完成的线段 ID

D.29 SMB870-SMB928：轴组 1 开环运动控制

表格 D-19 轴组 1 SM 地址

<table>
<tr><th>SM 位</th><th colspan="8">描述</th></tr>
<tr><td>SMB870-SMB885</td><td colspan="8">轴组 1 的名称（16 个 ASCII 字符）。SMB870 是第一个字符。</td></tr>
<tr><td>SMB886-SMB889</td><td colspan="8">预留</td></tr>
<tr><td>SMW890</td><td colspan="8">错误代码 - 参见运动轴组错误代码</td></tr>
<tr><td>SMB892</td><td colspan="8">为瞬时轴组状态预留。</td></tr>
<tr><td rowspan="6">SMB893</td><td rowspan="3"></td><td>MSB</td><td colspan="6"></td><td>LSB</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Done</td><td colspan="7">轴组运动指令的错误代码。请参见轴组运动指令的错误代码。</td></tr>
<tr><td rowspan="2">Done</td><td>0</td><td colspan="7">轴组上的运动正在进行中。</td></tr>
<tr><td>1</td><td colspan="7">操作完成（初始化期间由轴组置位）。</td></tr>
<tr><td>SMB894</td><td colspan="8">当前路径</td></tr>
<tr><td>SMB895</td><td colspan="8">预留</td></tr>
<tr><td>SMB896</td><td colspan="8">当前线段</td></tr>
<tr><td>SMB897</td><td colspan="8">预留</td></tr>
<tr><td>SMD898</td><td colspan="8">轴组 1 的当前复合速度</td></tr>
<tr><td>SMD902</td><td colspan="8">轴 X 的当前位置</td></tr>
<tr><td>SMD906</td><td colspan="8">轴 Y 的当前位置</td></tr>
<tr><td>SMD910</td><td colspan="8">轴 Z 的当前位置</td></tr>
<tr><td>SMD914</td><td colspan="8">预留</td></tr>
</table>

SM 位	描述
SMD918 V 存储器指针	指向轴组的组态和路径表的 V 存储器指针。
SMB922	轴组的缓冲区忙标志
SMB923	正在执行或等待执行的路径 ID
SMB924	当前缓冲模式的线段数
SMB925	轴组状态
SMB926	轴启用状态
SMB927	上次完成的路径 ID
SMB928	上次完成的线段 ID

D.30 SMB929-SMB936：轨迹控制和轨迹状态

S7-200 SMART CPU 使用以下特殊存储器监视和控制轨迹。

SM 定义	Trace 0	Trace 1	Trace 2	Trace 4
轨迹是否激活: • 0 = 否 • 1 = 是	SMB929	SMB931	SMB933	SMB935
轨迹状态： • 0 = 未配置 • 1 = 已配置; • 2 = 记录中 • 3 = 已触发 • 4 = 停止	SMB930	SMB932	SMB934	SMB936

D.31 SMB1000-SMB1049： CPU 硬件/固件 ID

在上电或热重启切换后，此 CPU 将信息写入特殊存储器。特殊存储器的 SMB1000-SMB1049 部分为只读。

SM 地址	说明
SMW1000	CPU 供应商 ID：（始终 0x002A）
SMB1002 至 SMB1021	CPU 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位

SM 地址	说明
SMB1022 至 SMB1037	CPU 序列号：ASCII 字符，在字段中左对齐，用空格补位
SMW1038	CPU 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。
SMD1040	CPU 固件版本： • 字节 0：ASCII ‘V’ • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW1044	CPU 固件版本计数器（范围 0x0000 至 0x00FF）
SMW1046	保留：始终为 0x0000
SMW1048	CPU 设备类型：始终为 0x0001

D.32 SMB950-SMB998、SMB1050-SMB1099：SB（信号板）硬件/固件 ID

在上电或热重启切换后，此 CPU 将信号板信息写入特殊存储器。特殊存储器的 SMB950-SMB999 和 SMB1050-SMB1099 部分为只读。

SM 地址		说明
信号板 0	信号板 1	
SMW1050	SMW950	信号板供应商 ID：如果存在 Siemens SB，则设置为 0x002A；如果没有 SB，则设为 0x0000
SMB1052 至 SMB1071	SMB952 到 SMB971	信号板订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位
SMB1072 至 SMB1087	SMB972 到 SMB987	信号板序列号：ASCII 字符，在字段中左对齐，用空格补位
SMW1088	SMW988	信号板硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。
SMD1090	SMD990	信号板固件版本： • 字节 0：ASCII ‘V’ • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW1094	SMW994	信号板固件版本计数器（范围 0x0000 至 0x00FF）

SM 地址		说明
信号板 0	信号板 1	
SMW1096	SMW996	保留，始终为 0x0000
SMW1098	SMW998	信号板设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

D.33 SMB1100-SMB1399、SMB250-SMB349、SMB2050-SMB2449：EM（扩展模块）硬件/固件 ID

在上电或热重启切换后，此 CPU 将扩展模块信息写入特殊存储器。特殊存储器的 SMB1100-SMB1399、SMB250 到 SMB349 SMB2050-SMB2449 为只读。

插槽 0 的 SM 地址	描述
SMW1100	EM 总线插槽 0 供应商 ID：如果存在 Siemens EM，则设置为 0x002A；如果没有 EM，则设为 0x0000
SMB1102 至 SMB1121	EM 总线插槽 0 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位
SMB1122 至 SMB1137	EM 总线插槽 0 序列号：ASCII 字符，在字段中左对齐，用空格补位
SMW1138	EM 总线插槽 0 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。
SMD1140	EM 总线插槽 0 固件版本： • 字节 0：ASCII ‘V’； • 字节 1：功能版本， • 字节 2：较小更改版本， • 字节 3：错误修复版本
SMW1144	EM 总线插槽 0 固件版本计数器（范围 0x0000 至 0x00FF）
SMW1146	保留，始终为 0x0000
SMW1148	EM 总线插槽 0 设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

插槽 1 的 SM 地址	描述
SMW1150	EM 总线插槽 1 供应商 ID：如果存在 Siemens EM，则设置为 0x002A；如果没有 EM，则设为 0x0000
SMB1152 至 SMB1171	EM 总线插槽 1 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位
SMB1172 至 SMB1187	EM 总线插槽 1：序列号：ASCII 字符，在字段中左对齐，用空格补位

插槽 1 的 SM 地址	描述
SMW1188	EM 总线插槽 1 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。
SMD1190	EM 总线插槽 1 固件版本： • 字节 0：ASCII 'V' • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW1194	EM 总线插槽 1 固件版本计数器（范围 0x0000 至 0x00FF）
SMW1196	保留，始终为 0x0000
SMW1198	EM 总线插槽 1 设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

插槽 2 的 SM 地址	描述
SMW1200	EM 总线插槽 2 供应商 ID：如果存在 Siemens EM，则设置为 0x002A；如果没有 EM，则设为 0x0000
SMB1202 至 SMB1221	EM 总线插槽 2 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位
SMB1222 至 SMB1237	EM 总线插槽 2 序列号：ASCII 字符，在字段中左对齐，用空格补位
SMW1238	EM 总线插槽 2 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。
SMD1240	EM 总线插槽 2 固件版本： • 字节 0：ASCII 'V' • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW1244	EM 总线插槽 2 固件版本计数器（范围 0x0000 至 0x00FF）
SMW1246	保留，始终为 0x0000
SMW1248	EM 总线插槽 2 设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

插槽 3 的 SM 地址	描述
SMW1250	EM 总线插槽 3 供应商 ID：如果存在 Siemens EM，则设置为 0x002A；如果没有 EM，则设为 0x0000
SMB1252 至 SMB1271	EM 总线插槽 3 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位
SMB1272 至 SMB1287	EM 总线插槽 3 序列号：ASCII 字符，在字段中左对齐，用空格补位

插槽 3 的 SM 地址	描述
SMW1288	EM 总线插槽 3 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。
SMD1290	EM 总线插槽 3 固件版本： • 字节 0：ASCII 'V' • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW1294	EM 总线插槽 3 固件版本计数器（范围 0x0000 至 0x00FF）
SMW1296	保留，始终为 0x0000
SMW1298	EM 总线插槽 3 设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

插槽 4 的 SM 地址	描述
SMW1300	EM 总线插槽 4 供应商 ID：如果存在 Siemens EM，则设置为 0x002A；如果没有 EM，则设为 0x0000
SMB1302 至 SMB1321	EM 总线插槽 4 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位
SMB1322 至 SMB1327	EM 总线插槽 4 序列号：ASCII 字符，在字段中左对齐，用空格补位
SMW1338	EM 总线插槽 4 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。
SMD1340	EM 总线插槽 4 固件版本： • 字节 0：ASCII 'V' • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW1344	EM 总线插槽 4 固件版本计数器（范围 0x0000 至 0x00FF）
SMW1346	保留，始终为 0x0000
SMW1348	EM 总线插槽 4 设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

插槽 5 的 SM 地址	描述
SMW1350	EM 总线插槽 5 供应商 ID：如果存在 Siemens EM，则设置为 0x002A；如果没有 EM，则设为 0x0000
SMB1352 至 SMB1371	EM 总线插槽 5 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位
SMB1372 至 SMB1387	EM 总线插槽 5 序列号：ASCII 字符，在字段中左对齐，用空格补位

插槽 5 的 SM 地址	描述
SMW1388	EM 总线插槽 5 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。
SMD1390	EM 总线插槽 5 固件版本： • 字节 0：ASCII 'V' • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW1394	EM 总线插槽 5 固件版本计数器（范围 0x0000 至 0x00FF）
SMW1396	保留，始终为 0x0000
SMW1398	EM 总线插槽 5 设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

插槽 6 的 SM 地址	描述
SMW250	EM 总线插槽 6 供应商 ID：如果存在 Siemens EM，则设置为 0x002A；如果没有 EM，则设为 0x0000
SMB252 到 SMB271	EM 总线插槽 6 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位
SMB272 到 SMB287	EM 总线插槽 6 序列号：ASCII 字符，在字段中左对齐，用空格补位
SMW288	EM 总线插槽 6 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。
SMD290	EM 总线插槽 6 固件版本： • 字节 0：ASCII 'V' • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW294	EM 总线插槽 6 固件版本计数器（范围 0x0000 至 0x00FF）
SMW296	保留，始终为 0x0000
SMW298	EM 总线插槽 6 设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

插槽 7 的 SM 地址	描述
SMW300	EM 总线插槽 7 供应商 ID：如果存在 Siemens EM，则设置为 0x002A；如果没有 EM，则设为 0x0000
SMB302 到 SMB321	EM 总线插槽 7 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位
SMB322 到 SMB337	EM 总线插槽 7 序列号：ASCII 字符，在字段中左对齐，用空格补位

插槽 7 的 SM 地址	描述
SMW338	EM 总线插槽 7 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。
SMD340	EM 总线插槽 7 固件版本： • 字节 0：ASCII ‘V’ • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW344	EM 总线插槽 7 固件版本计数器（范围 0x0000 至 0x00FF）
SMW346	保留，始终为 0x0000
SMW348	EM 总线插槽 7 设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

插槽 8 的 SM 地址	描述
SMW2050	EM 总线插槽 8 供应商 ID：如果存在 Siemens EM，则设置为 0x002A；如果没有 EM，则设为 0x0000
SMB2052 到 SMB2071	EM 总线插槽 8 供应商 ID：如果存在 Siemens 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位
SMB2072 到 SMB2087	EM 总线插槽 8 供应商 ID：如果存在 Siemens 订货号 (M 序列号：ASCII 字符，在字段中左对齐，用空格补位
SMW2088	EM 总线插槽 8 供应商 ID：如果存在 Siemens 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。
SMD2090	EM 总线插槽 8 固件版本： • 字节 0：ASCII ‘V’ • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW2094	EM 总线插槽 8 固件版本计数器（范围 0x0000 至 0x00FF）
SMW2096	保留，始终为 0x0000
SMW2098	EM 总线插槽 8 供应商 ID：如果存在 Sie 设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

插槽 9 的 SM 地址	描述
SMW2100	EM 总线插槽 9 供应商 ID：如果存在 Siemens EM，则设置为 0x002A；如果没有 EM，则设为 0x0000
SMB2102 到 SMB2121	EM 总线插槽 9 供应商 ID：如果存在 Siemens 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位
SMB2122 到 SMB2137	EM 总线插槽 9 供应商 ID：如果存在 Siemens 订货号 (M 序列号：ASCII 字符，在字段中左对齐，用空格补位
SMW2138	EM 总线插槽 9 供应商 ID：如果存在 Siemens 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。
SMD2140	EM 总线插槽 9 固件版本： • 字节 0：ASCII ‘V’ • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW2144	EM 总线插槽 固件版本计数器（范围 0x0000 至 0x00FF）
SMW2146	保留，始终为 0x0000
SMW2148	EM 总线插槽 9 设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

插槽 10 的 SM 地址	描述
SMW2150	EM 总线插槽 10 供应商 ID：如果存在 Siemens EM，则设置为 0x002A；如果没有 EM，则设为 0x0000
SMB2152 到 SMB2171	EM 总线插槽 10 供应商 ID：如果存在 Siemens 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位
SMB2172 到 SMB2187	EM 总线插槽 10 供应商 ID：如果存在 Siemens 订货号 (M 序列号：ASCII 字符，在字段中左对齐，用空格补位
SMW2188	EM 总线插槽 10 供应商 ID：如果存在 Siemens 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。

插槽 10 的 SM 地址	描述
SMD2190	EM 总线插槽 10 固件版本： • 字节 0：ASCII 'V' • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW2194	EM 总线插槽 10 固件版本计数器（范围 0x0000 至 0x00FF）
SMW2196	保留，始终为 0x0000
SMW2198	EM 总线插槽 10 设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

插槽 11 的 SM 地址	描述
SMW2200	EM 总线插槽 11 供应商 ID：如果存在 Siemens EM，则设置为 0x002A；如果没有 EM，则设为 0x0000
SMB2202 到 SMB2221	EM 总线插槽 11 供应商 ID：如果存在 Siemens 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位
SMB2222 到 SMB2237	EM 总线插槽 11 供应商 ID：如果存在 Siemens 订货号 (M 序列号：ASCII 字符，在字段中左对齐，用空格补位
SMW2238	EM 总线插槽 11 供应商 ID：如果存在 Siemens 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。
SMD2240	EM 总线插槽 11 固件版本： • 字节 0：ASCII 'V' • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW2244	EM 总线插槽 11 固件版本计数器（范围 0x0000 至 0x00FF）
SMW2246	保留，始终为 0x0000
SMW2248	EM 总线插槽 11 设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

插槽 12 的 SM 地址	描述
SMW2250	EM 总线插槽 12 供应商 ID：如果存在 Siemens EM，则设置为 0x002A；如果没有 EM，则设为 0x0000
SMB2252 到 SMB2271	EM 总线插槽 12 供应商 ID：如果存在 Siemens 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位

插槽 12 的 SM 地址	描述
SMB2272 到 SMB2287	EM 总线插槽 12 供应商 ID：如果存在 Siemens 订货号 (M 序列号：ASCII 字符，在字段中左对齐，用空格补位
SMW2288	EM 总线插槽 12 供应商 ID：如果存在 Siemens 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。
SMD2290	EM 总线插槽 12 固件版本： • 字节 0：ASCII 'V' • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW2294	EM 总线插槽 12 固件版本计数器（范围 0x0000 至 0x00FF）
SMW2296	保留，始终为 0x0000
SMW2298	EM 总线插槽 12 设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

插槽 13 的 SM 地址	描述
SMW2300	EM 总线插槽 13 供应商 ID：如果存在 Siemens EM，则设置为 0x002A；如果没有 EM，则设为 0x0000
SMB2302 到 SMB2321	EM 总线插槽 13 供应商 ID：如果存在 Siemens 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位
SMB2322 到 SMB2337	EM 总线插槽 13 供应商 ID：如果存在 Siemens 订货号 (M 序列号：ASCII 字符，在字段中左对齐，用空格补位
SMW2338	EM 总线插槽 13 供应商 ID：如果存在 Siemens 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。
SMD2340	EM 总线插槽 13 固件版本： • 字节 0：ASCII 'V' • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW2344	EM 总线插槽 13 固件版本计数器（范围 0x0000 至 0x00FF）
SMW2346	保留，始终为 0x0000
SMW2348	EM 总线插槽 13 设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

插槽 14 的 SM 地址	描述
SMW2350	EM 总线插槽 14 供应商 ID：如果存在 Siemens EM，则设置为 0x002A；如果没有 EM，则设为 0x0000
SMB2352 到 SMB2371	EM 总线插槽 14 供应商 ID：如果存在 Siemens 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位
SMB2372 到 SMB2387	EM 总线插槽 14 供应商 ID：如果存在 Siemens 订货号 (M 序列号：ASCII 字符，在字段中左对齐，用空格补位
SMW2388	EM 总线插槽 14 供应商 ID：如果存在 Siemens 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。
SMD2390	EM 总线插槽 14 固件版本： • 字节 0：ASCII ‘V’ • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW2394	EM 总线插槽 14 固件版本计数器（范围 0x0000 至 0x00FF）
SMW2396	保留，始终为 0x0000
SMW2398	EM 总线插槽 14 设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

插槽 15 的 SM 地址	描述
SMW2400	EM 总线插槽 15 供应商 ID：如果存在 Siemens EM，则设置为 0x002A；如果没有 EM，则设为 0x0000
SMB2402 到 SMB2421	EM 总线插槽 15 供应商 ID：如果存在 Siemens 订货号 (MLFB)：ASCII 字符，在字段中左对齐，用空格补位
SMB2422 到 SMB2437	EM 总线插槽 15 供应商 ID：如果存在 Siemens 订货号 (M 序列号：ASCII 字符，在字段中左对齐，用空格补位
SMW2438	EM 总线插槽 15 供应商 ID：如果存在 Siemens 硬件版本：表示硬件版本；范围 = 0x0001 至 0xFFFD 0x0000、0xFFFE 和 0xFFFF 是保留值。

插槽 15 的 SM 地址	描述
SMD2440	EM 总线插槽 15 固件版本： • 字节 0：ASCII 'V' • 字节 1：功能版本 • 字节 2：较小更改版本 • 字节 3：错误修复版本
SMW2444	EM 总线插槽 15 固件版本计数器（范围 0x0000 至 0x00FF）
SMW2446	保留，始终为 0x0000
SMW2448	EM 总线插槽 15 设备类型：I/O = 0x0003，通信 = 0x0004，所有其他值保留

D.34 SMB1400-SMB1799、SMB2450-2999：EM（扩展模块）模块特定的数据

CPU 为每个扩展模块保留额外的 50 字节，用于模块特定的只读数据：

SM 地址	描述
SMB1400 到 SMB1449	EM 总线槽 0：模块特定的信息
SMB1450 到 SMB1499	EM 总线槽 1：模块特定的信息
SMB1500 到 SMB1549	EM 总线槽 2：模块特定的信息
SMB1550 到 SMB1599	EM 总线槽 3：模块特定的信息
SMB1600 到 SMB1649	EM 总线槽 4：模块特定的信息
SMB1650 到 SMB1699	EM 总线槽 5：模块特定的信息
SMB1700 到 SMB1749	EM 总线槽 6：模块特定的信息
SMB1750 到 SMB1799	EM 总线槽 7：模块特定的信息
SMB2450 到 SMB2499	EM 总线槽 8：模块特定的信息
SMB2500 到 SMB2549	EM 总线槽 9：模块特定的信息
SMB2550 到 SMB2599	EM 总线槽 10：模块特定的信息
SMB2600 到 SMB2649	EM 总线槽 11：模块特定的信息
SMB2650 到 SMB2699	EM 总线槽 12：模块特定的信息
SMB2700 到 SMB2749	EM 总线槽 13：模块特定的信息
SMB2750 到 SMB2799	EM 总线槽 14：模块特定的信息
SMB2800 到 SMB2849	EM 总线槽 15：模块特定的信息

D.35 SMB1800-SMB1939：PROFINET 设备状态

S7-200 SMART CPU 用作控制器时，SMB1800-SMB1935 指示 IO 设备状态或循环数据。

SM 地址	说明
SMB1800 到 SMB1807	每个字节指示每个 PROFINET 设备的状态。 • 0x00：未组态。 • 0x80：正常。 • 0×81：未连接。（控制器无法连接设备。） • 0×82：诊断。（报告报警。）
SMB1808 到 SMB1871	每个模块的报警状态。SMB1808 到 SMB1815 有 64 位，显示了 1 号设备（设备编号 1）的报警状态。0 为“正常”，1 为“错误”。 例如，SM1808.0 表示第一个设备的第一个模块状态。SM1816.0 表示第二个设备的第一个模块状态。
SMB1872 到 SMB1935	IO 设备模块的 IO 数据状态。SMB1872 到 SMB1879 有 64 位，显示了 1 号设备（设备编号 1）中 IO 数据的报警状态。0 为“正常”，1 为 “IO 数据错误”。 例如，SM1872.0 表示第一个设备的第一个模块 IO 数据状态。SM1880.2 表示第二个设备的第三个模块 IO 数据状态。

S7-200 SMART CPU 用作智能设备时，SM1936.0- SM1937.0 表示智能设备与其上位控制器的连接状态以及智能设备的 IO 数据状态。

SM 地址	说明
SMB1936	该字节指示 PROFINET 智能设备与其上位控制器的连接状态。 • 0x00：未组态。 • 0x80：正常。 • 0×81：未连接。（智能设备未连接上位控制器。） • 0×82：诊断。（智能设备已连接上位控制器，但该智能设备的组态与上位控制器的组态不匹配。）
SM1937.0	智能设备 IO 数据状态。0 为“正常”，1 为 “IO 数据错误”。

D.36 SMB1940-SMB1946：Web 服务器状态

SMD1940 指示 Web 服务器的证书验证状态。

值	说明
0	证书正常
0x1	证书过期
0x0200	证书尚未生效
0x10000000	缺少证书
0x20000000	证书损坏
其它	保留的证书错误

SMB1944- SMB1946 部分显示 Web 服务器状态。

SM 地址	说明
SM1944.0	• 0：Web 服务器未激活。 • 1：Web 服务器已激活。
SM1944.1	• 0：Web OSS 关闭。 • 1：Web OSS 开启。
SM1944.2~7	保留
SMB1945	当前激活的 Web TCP 连接
SMB1946	最大激活的 Web TCP 连接

参考

E

E.1 常用特殊存储器位

在系统定义的变量表中，完整列出了可用于用户项目的预定义特殊存储器程序变量。

表格 E-1 常用特殊存储器位

SM 地址	变量名	描述
SM0.0	Always_On	该位始终为 TRUE。
SM0.1	First_Scan_On	在第一个扫描周期，CPU 将该位设置为 TRUE，此后将其设置为 FALSE。该位的一个用途是调用初始化子例程。
SM0.2	Retentive_Lost	如果保持数据丢失，一个扫描周期设置为 TRUE。
SM0.3	RUN_Power_Up	从上电进入 RUN 模式时，一个扫描周期设置为 TRUE。
SM0.4	Clock_60s	针对 1 分钟的周期时间，时钟脉冲 30 s 为 TRUE，断开 30 s。
SM0.5	Clock_1s	针对 1 s 的周期时间，时钟脉冲 0.5 s 为 TRUE，断开 0.5 s。
SM0.6	Clock_Scan	扫描周期时钟，一个扫描周期为 TRUE，下一个扫描周期关断
SM0.7	RTC_Lost	对于具有实时时钟的 CPU 型号，如果实时时钟设备的时间被复位或在上电时丢失，CPU 会在一个扫描周期将该位设置为 TRUE。程序可将该位用作错误存储器位或用来调用特殊启动序列。
SM1.0	Result_0	特定指令的操作结果 = 0 时，置位为 TRUE
SM1.1	Overflow_Illegal	特定指令执行结果溢出或数值非法时，置位为 TRUE
SM1.2	Neg_Result	当数学运算产生负数结果时，设置为 TRUE
SM1.3	Divide_By_0	尝试除以零时，设置为 TRUE
SM1.4	Table_Overflow	当填表指令尝试过度填充表格时，设置为 TRUE
SM1.5	Table_Empty	当 LIFO 或 FIFO 指令尝试读取空表时，设置为 TRUE
SM1.6	Not_BCD	尝试将非 BCD 值转换为二进制值时，设置为 TRUE
SM1.7	Not_Hex	当 ASCII 值无法转换为有效十六进制值时，设置为 TRUE

E.2 按优先级别顺序排列的中断事件

表格 E-2 中断事件的优先级顺序

优先级组	事件	描述
通信 最高优先级	8	端口 0 接收字符
	9	端口 0 发送完成
	23	端口 0 接收消息完成
	24	端口 1 接收消息完成
	25	端口 1 接收字符
	26	端口 1 发送完成
	46	端口 2 接收消息完成
	47	端口 2 接收字符
	48	端口 2 发送完成

优先级组	事件	描述
离散 中等优先级	19	PTO0 脉冲计数完成
	20	PTO1 脉冲计数完成
	34	PTO2 脉冲计数完成
	45	PTO3 脉冲计数完成
	67	PTO4 脉冲计数完成
	0	I0.0 上升沿
	2	I0.1 上升沿
	4	I0.2 上升沿
	6	I0.3 上升沿
	1	I0.0 下降沿
	3	I0.1 下降沿
	5	I0.2 下降沿
	7	I0.3 下降沿
	35	I7.0 上升沿（信号板 0，输入 0）
	36	I7.0 下降沿（信号板 0，输入 0）
	37	I7.1 上升沿（信号板 0，输入 1）
	38	I7.1 下降沿（信号板 0，输入 1）
	49	I72.0 上升沿（信号板 1，输入 0）
	50	I72.0 下降沿（信号板 1，输入 0）
	51	I72.1 上升沿（信号板 1，输入 1）
	52	I72.1 下降沿（信号板 1，输入 1）
	12	HSC0 CV=PV（当前值 = 预设值）
	27	HSC0 方向改变
	28	HSC0 外部复位
	13	HSC1 CV=PV（当前值 = 预设值）
	14	HSC1 方向改变
	15	HSC1 外部复位/视在阻抗相位
	16	HSC2 CV=PV（当前值 = 预设值）
	17	HSC2 方向改变
	18	HSC2 外部复位
	32	HSC3 CV=PV（当前值 = 预设值）

优先级组	事件	描述
	53	HSC3 方向改变
	54	HSC3 外部复位
	29	HSC4 CV=PV
	30	HSC4 方向改变
	31	HSC4 外部复位
	33	HSC5 CV=PV
	43	HSC5 方向改变
	44	HSC5 外部复位
	55	HSC6 CV=PV
	56	HSC6 方向改变
	57	HSC6 外部复位
	58	HSC7 CV=PV
	59	HSC7 方向改变
	60	HSC7 外部复位
定时 最低优先级	10	定时中断 0 SMB34
	11	定时中断 1 SMB35
	21	定时器 T32 CT = PT 中断
	22	定时器 T96 CT = PT 中断
	61	定时器 T262 CT = PT 中断
	62	定时器 T263 CT = PT 中断
	63	定时器 T264 CT = PT 中断
	64	定时器 T265 CT = PT 中断
	65	定时器 T266 CT = PT 中断
	66	定时器 T267 CT = PT 中断

E.3 高速计数器汇总

	时钟 A	方向/时钟 B	复位	单相/双相最大时钟/输入速率	AB 正交相最大时钟/输入速率
HSC0、HSC1、HSC2、HSC3、HSC4、HSC5、HSC6、HSC7	I0.0-I1.7 [1]			• 200 kHz	• 200 kHz • A/B 相位 (4X) 频率支持高达 800 kHZ

[1] 对于同一通道 HSC，时钟 A 和方向/时钟 B 只能在 IB0 (I0.0 ~I0.7) 或 IB1 (I1.0 ~I1.7) 中同时进行选择。

E.4 STL 指令

指令

STL 指令名称和说明显示在下表中。有关 LAD 和 FBD 指令的信息，请参见程序指令一章。

布尔指令	
STL	**描述**
LD 位 LDI 位 LDN 位 LDNI 位	加载 即时装载 装载非 即时装载非
A 位 AI 位 AN 位 ANI 位	与 即时与 与非 即时与非
O 位 OI 位 ON 位 ONI 位	或 立即或 或非 即时或非
LDBx IN1, IN2	加载字节比较结果 IN1 (x:<, <=,=, >=, >, <>I) IN2

布尔指令	
STL	**描述**
ABx IN1, IN2	对字节比较结果执行与运算 IN1 (x:<, <=,=, >=, >, <>) IN2
OBx IN1, IN2	对字节比较结果执行或运算 IN1 (x:<, <=,=, >=, >, <>) IN2
LDWx IN1, IN2	加载字比较结果 IN1 (x:<, <=,=, >=, >, <>) IN2
AWx IN1, IN2	对字比较结果执行与运算 IN1 (x:<, <=,=, >=, >, <>)I N2
OWxIN1, IN2	对字比较结果执行或运算 IN1 (x:<, <=,=, >=, >, <>) IN2
LDDx IN1, IN2	装载双字比较结果 IN1 (x:<, <=,=, >=, >, <>) IN2
ADx IN1, IN2	对双字比较结果执行与运算 IN1 (x:<, <=,=, >=, >, <>) IN2
ODx IN1, IN2	对双字比较结果执行或运算 IN1 (x:<, <=,=, >=, >, <>) IN2
LDRx IN1, IN2	加载实数比较结果 IN1 (x:<, <=,=, >=, >, <>) IN2
ARx IN1, IN2	对实数比较结果执行与运算 IN1 (x:<, <=,=, >=, >, <>) IN2
ORx IN1, IN2	对实数比较结果执行或运算 IN1 (x:<, <=,=, >=, >, <>) IN2
NOT	堆栈求非
EU TRIGR	上升沿检测
ED TRIGF	下降沿检测
= 位 =I bit	赋值 立即赋值

布尔指令	
STL	**描述**
S bit, N R bit, N SI bit, N RI bit, N	设置位范围 复位位范围 立即设置位范围 立即复位位范围
在 STL 中不可用	SR（置位优先双稳态触发器） RS（复位优先双稳态触发器）
LDSx IN1, IN2 ASx IN1, IN2 OSx IN1, IN2	加载字符串比较结果 IN1 (x: =, <>) IN2 对字符串比较结果执行与运算 IN1 (x: =, <>) IN2 对字符串比较结果执行或运算 IN1 (x: =, <>) IN2
ALD OLD	与装载 或装载
LPS LRD LPP LDS n	逻辑进栈（堆栈控制） 逻辑读取（堆栈控制） 逻辑出栈（堆栈控制） 装载堆栈（堆栈控制），n = 0 至 8
AENO	对 ENO 执行与运算

算术运算、递增和递减指令	
STL	**描述**
+I IN1、IN2、OUT +D IN1、IN2、OUT +R IN1、IN2、OUT	整数、双精度整数或实数加法 IN1 + IN2 = OUT
-I IN1、IN2、OUT -D IN1、IN2、OUT -R IN1、IN2、OUT	整数、双精度整数或实数减法 IN2 - IN1=OUT
MUL IN1、IN2、OUT	整数乘法（16*16->32）

算术运算、递增和递减指令	
STL	**描述**
*I IN1、IN2、OUT *D IN1、IN2、OUT *R IN1、IN2、IN2	整数、双精度整数或实数乘法 IN1 * IN2 = OUT
DIV IN1、IN2、OUT	整数除法（16/16->32）
/I IN1、IN2、OUT /D IN1、IN2、OUT /R IN1、IN2、OUT	整数、双精度整数或实数除法 IN2 / IN1 = OUT
SQRT IN, OUT	平方根
LN IN, OUT	自然对数
EXP IN, OUT	自然指数
SIN IN, OUT	正弦
COS IN, OUT	余弦
TAN IN, OUT	正切
INCB OUT INCW OUT INCD OUT	字节、字或双字递增
DECB OUT DECW OUT DECD OUT	字节、字或双字递减
PID TBL, LOOP	PID 回路

定时器和计数器指令	
STL	**描述**
TON Txxx, PT	接通延时定时器
TOF Txxx, PT	关断延时定时器
TONR Txxx, PT	保持性接通延时定时器
IECTON XXX, PT, ET	接通延时定时器（IEC)
IECTOF XXX, PT, ET	关断延时定时器（IEC)
IECTONR XXX, XXX.RESET PT, ET	保持性接通延时定时器（IEC)
BITIM OUT	启动间隔定时器
CITIM IN, OUT	计算间隔定时器
CTU Cxxx, PV	向上计数
CTD Cxxx, PV	向下计数
CTUD Cxxx, PV	向上/向下计数
CTUDINT XXX, XXX.RESET, XXX.PV, XXX.CV	向上计数（IEC)
CTDDINT XXX, XXX.LOAD, XXX.PV, XXX.CV	向下计数（IEC)
CTUDDINT XXX, XXX.CD, XXX.RESET, XXX.LOAD, XXX.PV, XXX.QD, XXX.CV	向上/向下计数（IEC)

高速指令	
STL	**描述**
HDEF HSC, MODE	定义高速计数器模式
HSC n	激活高速计数器
PLS n	脉冲输出：

实时时钟指令	
STL	**描述**
TODR T	读取日时钟
TODW T	写入日时钟
TODRX T	读取扩展实时时钟
TODWX T	设置扩展实时时钟

程序控制指令	
STL	**描述**
END	程序有条件结束
STOP	切换到 STOP 模式
WDR	看门狗复位 (500 ms)
JMP N LBL N	跳转至定义的标签 定义标签
CALL N [N1,...] CRET	调用子例程 [N1, ... 最多 16 个可选参数] 从 SBR 有条件返回
FOR INDX,INIT,FINAL NEXT	For/Next 循环
LSCR N SCRT N CSCRE SCRE	顺序控制继电器段的装载、转换、有条件结束和结束
GERR ECODE	获取错误代码

传送、移位和循环指令	
STL	**描述**
MOVE IN、OUT	移动字节、字、双字、实数、UDT、数组、字符串、FB 实例

MOVB IN, OUT
MOVW IN, OUT
MOVD IN, OUT
MOVR IN, OUT
	移动字节、字、双字和实数

传送、移位和循环指令	
STL	**描述**
BIR IN, OUT BIW IN, OUT	移动字节立即读取 移动字节立即写入
BMB IN, OUT, N BMW IN, OUT, N BMD IN, OUT, N	成块移动字节、字和双字
SWAP IN	字节交换
SHRB DATA, S_BIT, N	移位寄存器位
SRB IN、N、OUT SRW IN、N、OUT SRD IN、N、OUT	右移字节、字和双字
SLB IN、N、OUT SLW IN、N、OUT SLD IN、N、OUT	左移字节、字和双字
RRB IN、N、OUT RRW IN、N、OUT RRD IN、N、OUT	循环右移字节、字和双字
RLB IN、N、OUT RLW IN、N、OUT RLD IN、N、OUT	循环左移字节、字和双字

逻辑指令	
STL	**描述**
ANDB IN1、IN2、OUT ANDW IN1、IN2、OUT ANDD IN1、IN2、OUT	对字节、字和双字执行逻辑与运算
ORB IN1、IN2、OUT ORW IN1、IN2、OUT ORD IN1、IN2、OUT	对字节、字和双字执行逻辑或运算

逻辑指令	
STL	**描述**
XORB IN1、IN2、OUT XORW IN1、IN2、OUT XORD IN1、IN2、OUT	对字节、字和双字执行逻辑异或运算
NANDB IN1、IN2、OUT NANDW IN1、IN2、OUT NANDD IN1、IN2、OUT	对字节、字和双字执行逻辑“与非”运算
NORB IN1、IN2、OUT NORW IN1、IN2、OUT NORD IN1、IN2、OUT	对字节、字和双字执行逻辑“或非”运算
NXORB IN1、IN2、OUT NXORW IN1、IN2、OUT NXORD IN1、IN2、OUT	对字节、字和双字执行逻辑“异或非”运算
INVB OUT INVW OUT INVD OUT	对字节、字和双字取反 （1 的补码）

字符串指令	
STL	**描述**
SLEN IN, OUT	字符串长度
SCAT N, OUT	连接字符串
SCPY IN, OUT	复制字符串
SSCPY IN, INDX, N, OUT	从字符串复制子字符串
CFND IN1, IN2, OUT	在字符串中查找第一个字符
SFND IN1, IN2, OUT	在字符串中查找字符串

表格、查找和转换指令	
STL	**描述**
ATT DATA, TBL	将数据添加到表格
LIFO TBL, DATA FIFO TBL, DATA	从表格中获取数据

表格、查找和转换指令	
STL	**描述**
FND= TBL, PTN, INDX FND<> TBL, PTN, INDX FND< TBL, PTN, INDX FND> TBL, PTN, INDX	在表格中查找与比较相符的数据值
FILL IN, OUT, N	用格式填充存储器空间
BCDI OUT IBCD OUT	将 BCD 转换为整数 将整数转换为 BCD
BTI IN, OUT ITB IN, OUT ITD IN, OUT DTI IN, OUT	将字节转换为整数 将整数转换为字节 将整数转换为双精度整数 将双精度整数转换为整数
DTR IN, OUT TRUNC IN, OUT ROUND IN, OUT	将双字转换为实数 将实数转换为双精度整数 将实数转换为双精度整数
ATH IN, OUT, LEN HTA IN, OUT, LEN ITA IN, OUT, FMT DTA IN, OUT, FM RTA IN, OUT, FM	将 ASCII 转换为十六进制 将十六进制转换为 ASCII 将整数转换为 ASCII 将双精度整数转换为 ASCII 将实数转换为 ASCII
DECO IN, OUT ENCO IN, OUT	解码 编码
SEG IN, OUT	生成 7 段码显示格式
ITS IN, FMT, OUT DTS IN, FMT, OUT RTS IN, FMT, OUT	将整数转换为字符串 将双精度整数转换为字符串 将实数转换为字符串
STI STR, INDX, OUT STD STR, INDX, OUT STR STR, INDX, OUT	将子字符串转换为整数 将字符串转换为双精度整数 将子字符串转换为实数

中断指令	
STL	**描述**
CRETI	从中断有条件返回
ENI DISI	启用中断 禁用中断
ATCH INT, EVNT DTCH EVNT	连接中断程序到事件 分离事件
CEVENT EVNT	清除所有类型为 EVNT 的中断事件

通信指令	
STL	**LAD/FBD**
GET PUT	读取远程站数据 向远程站写入数据
XMT TBL, PORT RCV TBL, PORT	自由端口发送 自由端口接收消息
GIP ADDR, MASK, GATE SIP ADDR, MASK, GATE	获取 CPU 地址、子网掩码及网关 设置 CPU 地址、子网掩码及网关
GPA ADDR, PORT SPA ADDR, PORT	获得端口地址 设置端口地址

E.5 CPU 存储器范围和特性

下表列出了 S7-200 SMART CPU 固件版本 V3.2 (ST32) 支持的 CPU 存储器范围和特性。

说明	CPU ST32
用户程序大小	100 KB
用户数据大小	120 KB
变量存储器 (V)	VB0 到 VB102399
非绑定全局变量和 FB 实例的存储器 (DB2)	1 MB[1]
过程映像输入寄存器	I0.0 到 I72.7
过程映像输出寄存器	Q0.0 到 Q72.7
模拟量输入（只读）	AIW0 到 AIW530

说明		CPU ST32
模拟量输出（只写）		AQW0 到 AQW530
局部存储器 (L)[2]		LB0 到 LB1023
位存储器 (M)		M0.0 到 M63.7
特殊存储器 (SM)	总数	SMB0 至 SMB2999
	只读	SMB0 至 SMB29 SMB32 至 SMB33 SMB250 至 SMB349 SMB480 至 SMB515 SMB950 至 SMB1999 SMB2014 至 SMB2849
定时器		512（T0 到 T511）
时间累加器	1 ms	T0、T64、T256 到 T261
	10 ms	T1 到 T4、T65 到 T68、T268 到 T291
	100 ms	T5 到 T31、T69 到 T95、T316 到 T413
接通/断开延时	1 ms	T32、T96、T262 到 T267
	10 ms	T33 到 T36、T97 到 T100、T292 到 T315
	100 ms	T37 到 T63、T101 到 T255、T414 到 T511
计数器		512（C0 到 C511）
高速计数器		HC0 到 HC7
顺序控制继电器 (S)		S0.0 到 S31.7
累加器寄存器		AC0 到 AC3
跳转/标号		0 到 255
调用/子例程		0 到 127
中断例程		0 到 127
正/负跳变		1024
PID 控制回路		0 到 15
端口		2 x 以太网编程端口 集成的 RS485 端口（端口 0） CM01 信号板 (SB) RS232/RS485 端口（端口 1、端口 2）

[1] 前 896KB 存储为读/写，后 128KB 为只读。

[2] 基于 LAD 或 FBD 进行编程时，STEP 7-Micro/WIN SMART V3 会保留 LB1008 到 LB1024。

S7-200 SMART CPU 固件版本 V3.2 (ST32) 支持基于以下 CPU 型号进行 PROFINET 通信。有关参数和 PROFINET 过程映像的详细信息，请参见下表。

说明	CPU ST32
PROFINET 设备最大数	10
PROFINET 设备的设备编号	1 至 10
每台 PROFINET 设备的最大输入大小	128 字节
每台 PROFINET 设备的最大输出大小	128 字节
最大模块数	64
PROFINET 设备的最小循环更新时间	更新时间的最小值还取决于为 PROFINET 设置的通信组件、PROFINET 设备的数量以及已组态的用户数据量。
PROFINET 过程映像输入寄存器的 CPU 地址范围	I128.0 至 I1151.7
PROFINET 过程映像输出寄存器的 CPU 地址范围	Q128.0 至 Q1151.7
1 号设备的 PROFINET 过程映像输入寄存器的 CPU 地址	I128.0 至 I255.7
2 号设备的 PROFINET 过程映像输入寄存器的 CPU 地址	I256.0 至 I383.7
3 号设备的 PROFINET 过程映像输入寄存器的 CPU 地址	I384.0 至 I511.7
4 号设备的 PROFINET 过程映像输入寄存器的 CPU 地址	I512.0 至 I639.7
5 号设备的 PROFINET 过程映像输入寄存器的 CPU 地址	I640.0 至 I767.7
6 号设备的 PROFINET 过程映像输入寄存器的 CPU 地址	I768.0 至 I895.7
7 号设备的 PROFINET 过程映像输入寄存器的 CPU 地址	I896.0 至 I1023.7
8 号设备的 PROFINET 过程映像输入寄存器的 CPU 地址	I1024.0 至 I1151.7
9 号设备的 PROFINET 过程映像输入寄存器的 CPU 地址	I1128.0 至 I1407.7
10 号设备的 PROFINET 过程映像输入寄存器的 CPU 地址	I1408.0 至 I1535.7
智能设备的 PROFINET 过程映像输入寄存器的 CPU 地址	I1152.0 至 I1279.7
1 号设备的 PROFINET 过程映像输出寄存器的 CPU 地址	Q128.0 至 Q255.7
2 号设备的 PROFINET 过程映像输出寄存器的 CPU 地址	Q256.0 至 Q383.7
3 号设备的 PROFINET 过程映像输出寄存器的 CPU 地址	Q384.0 至 Q511.7
4 号设备的 PROFINET 过程映像输出寄存器的 CPU 地址	Q512.0 至 Q639.7
5 号设备的 PROFINET 过程映像输出寄存器的 CPU 地址	Q640.0 至 Q767.7
6 号设备的 PROFINET 过程映像输出寄存器的 CPU 地址	Q768.0 至 Q895.7

说明	CPU ST32
7 号设备的 PROFINET 过程映像输出寄存器的 CPU 地址	Q896.0 至 Q1023.7
8 号设备的 PROFINET 过程映像输出寄存器的 CPU 地址	Q1024.0 至 Q1151.7
9 号设备的 PROFINET 过程映像输出寄存器的 CPU 地址	Q1128.0 至 Q1407.7
10 号设备的 PROFINET 过程映像输出寄存器的 CPU 地址	Q1408.0 至 Q1535.7
智能设备的 PROFINET 过程映像输出寄存器的 CPU 地址	Q1152.0 至 Q1279.7

订购信息

F

F.1 CPU 模块

CPU 型号	订货号
CPU ST32，DC/DC/DC	6ES7288-1ST32-0AA0

F.2 扩展模块（EMs）和信号板（SBs）

扩展模块和信号板	订货号
EM 16 点数字量输入 (EM DE16)	6ES7288-2DE16-0AA1
EM 16 点晶体管型数字量输出 (EM QT16)	6ES7288-2QT16-0AA1
EM 4 点模拟量输入 (EM AE04)	6ES7288-3AE04-0AA1
EM 4 点模拟量输出 (EM AQ04)	6ES7288-3AQ04-0AA1
EM 8 点模拟量输入 (EM AE08)	6ES7288-3AE08-0AA1
EM 4 点模拟量输入/2 点模拟量输出 (EM AM06)	6ES7288-3AM06-0AA1
EM RTD 4 x 16 位 (EM AR04)	6ES7288-3AR04-0AA1
EM TC 8 x 16 位 (EM AT08)	6ES7288-3AT08-0AA1
SB RS485/RS232 (SB CM01)	6ES7288-5CM01-0AA1

F.3 编程软件

编程软件	订货号
STEP 7-Micro/WIN SMART V3 独立许可证 (CD-ROM)	6ES7288-8SW01-0AA1

F.4 备件和其它硬件

常规附件	订货号
末端保持器，钢制	8WA1805
端子块 24V	6ES7193-8ST00-1AN0
DIN 导轨 35 mm，镀锡钢条 [1]	
• 长度：483 mm	6ES5710-8MA11
• 长度：530 mm	6ES5710-8MA21
• 长度：830 mm	6ES5710-8MA31
• 长度： 2000 mm	6ES5710-8MA41
SIMATIC 系统导轨 [1]	
• 长度：483 mm	6ES7193-6MR00-0AA0
• 长度：530 mm	6ES7193-6MR00-0BA0
• 长度：830 mm	6ES7193-6MR00-0CA0
• 长度： 2000 mm	6ES7193-6MR00-0DA0

[1] DIN 导轨和 SIMATIC 系统导轨的长度可以根据需要缩短。

索引

-

-D（STL - 双整数减法）, 393
-I（STL - 整数减法）, 393
-R（STL - 实数减法）, 393

*****

*D（STL - 双整数乘法）, 393
*I（STL - 整数乘法）, 393
*R（STL - 实数乘法）, 393

.

.smart 文件, 153

/

/D（STL - 双整数除法）, 393
/I（STL - 整数除法）, 393
/R（STL - 实数除法）, 393

+

+D（STL - 双整数加法）, 393
+I（STL - 整数加法）, 393
+R（STL - 实数加法）, 393

<

<=B, 321
<=R, 321
<=W, 321
<>B, 321
<>R, 321
<>S, 325
<>W, 321
<B, 321
<R, 321
<W, 321

=

=（STL - 输出）, 270
==B, 321
==R, 321
==S, 325
==W, 321
=I（STL - 立即输出）, 270

>

>=B, 321
>=R, 321
>=W, 321
>B, 321
>R, 321
>W, 321

A

A（STL -“与”运算）, 261
AB<（STL - 与运算比较字节小于）, 321
AB<=（STL - 与运算比较字节小于或等于）, 321
AB<>（STL - 与运算比较字节不等于）, 321
AB=（STL - 与运算比较字节等于）, 321
AB>（STL - 与运算比较字节大于）, 321
AB>=（STL - 与运算比较字节大于或等于）, 321
ABS_DI, 410
ABS_I, 410
ABS_R, 410
ABSDI（STL-绝对双精度整数）, 345, 410
ABSI（STL-绝对整数）, 345, 410
ABSR（STL-绝对实数）, 345, 410
AC
    接地准则, 103
    接线和接地的先决条件, 102
    接线准则, 103
    绝缘准则, 102
AD<（STL - 与运算比较双字小于）, 321
AD<=（STL - 与运算比较双字小于或等于）, 321
AD<>（STL - 与运算比较双字不等于）, 321
AD=（STL - 与运算比较双字等于）, 321
AD>（STL - 与运算比较双字大于）, 321
AD>=（STL - 与运算比较双字大于或等于）, 321
ADD_DI, 393
ADD_I, 393
ADD_R, 393
AENO（STL 堆栈 - 与 ENO 位）
    逻辑堆栈概述, 265
    指令, 266
AI（STL - 立即与）, 263
ALD（STL 堆栈 - 与装载）, 266

ALT (STL), 274
ALTP
    STL, 274
AN（STL -“与非”运算）, 261
ANDB（STL-字节与）, 431
ANDD（STL-双字与）, 431
ANDW（STL-字与）, 431
ANI（STL - 取反后立即与）, 263
AR<（STL - 与运算比较实数小于）, 321
AR<=（STL - 与运算比较实数小于或等于）, 321
AR<>（STL - 与运算比较实数不等于）, 321
AR=（STL - 与运算比较实数等于）, 321
AR>（STL - 与运算比较实数大于）, 321
AR>=（STL - 与运算比较实数大于或等于）, 321
ARCCOS（反余弦）, 399
ARCSIN（反正弦）, 399
ARCTAN（反正切）, 399
AS<>（STL - 与运算字符串比较不等于）, 325
AS=（STL - 与运算字符串比较等于）, 325
ASCII 数组转换指令, 330
ATCH, 413
ATH, 330
ATT, 468
AVG_B, 408
AVG_DI, 408
AVG_I, 408
AVG_R, 408
AW<（STL - 与运算比较字小于）, 321
AW<=（STL - 与运算比较字小于或等于）, 321
AW<>（STL - 与运算比较字不等于）, 321
AW=（STL - 与运算比较字等于）, 321
AW>（STL - 与运算比较字大于）, 321
AW>=（STL - 与运算比较字大于或等于）, 321
AXISx_GOTO
    超驰, 1127

B

B_I, 326
BCD_I, 326
BCDI（STL - BCD 到整数）, 326
BGN_ITIME, 490
BIR（STL-字节立即读取）, 441
BITIM（STL-开始间隔时间）, 490
BIW（STL-字节立即写入）, 441
BLKMOV_B, 438
BLKMOV_D, 438
BLKMOV_W, 438
BMB（STL-字节块传送）, 438
BMD（STL-双字块传送）, 438
BMW（STL-字块传送）, 438
BTI（STL - 字节到整数）, 326

C

CAL_ITIME, 490
CALL, 491, 497
CE 认证, 1209
CFND（STL-查找字符）, 465
CHR_FIND, 465
CITIM（STL-计算间隔时间）, 490
CLR_EVNT, 413
CM01 信号板
    连接器引脚分配, 564
    偏置且端接, 565
COS（余弦）, 399
CPU
    DIN 导轨, 96
    IP 地址, 517
    LED 指示灯, 146
    MAC 地址, 524
    PPI 连接数, 553
    RS485 网络地址, 558
    RS485 网络端口, 558
    ST32 规范, 1214
    ST32 接线图, 1222
    安装, 95
    查看 PLC 信息, 180
    存储卡, 141
    访问数据, 119
    非致命错误存储单元, 1261
    过程映像寄存器, 117
    清除存储区, 239
    设置类型, 84
    通信类型, 34, 511
    通信连接数, 512
    系统块, 211
    以太网端口, 517
    以太网通信, 514
    支持的扩展模块, 31
    致命错误, 1262
    组态与 HMI 的通信, 526
CPU ID 寄存器 (SMB6-SMB7), 1272
CPU 和模块的设备组态, 211
CPU 角色
    控制器和智能设备, 551
CPU 硬件/固件 ID (SMB1000-SMB1049), 1306
CRET（STL-从功能块有条件返回）, 497
CRET（STL-从子例程有条件返回）, 491
CTD（向下计数）, 349
CTU（向上计数）, 349
CTUD（向上计数/向下计数）, 349

D

DC
接地准则, 103
接线和接地的先决条件, 102
接线准则, 103
绝缘准则, 102
DEC_B, 402
DEC_DW, 402
DEC_W, 402
DECB（STL - 字节递减）, 402
DECD（STL - 双字递减）, 402
DECO, 343
DECW（STL - 字递减）, 402
DI_I, 326
DI_R, 326
DI_S, 336
DIN 导轨, 95, 96
DISI, 413
DIV, 397
DIV_DI, 393
DIV_I, 393
DIV_R, 393
DSC（动态伺服控制）, 810, 811
DTA, 330
DTCH, 413
DTI（STL - 双精度整数到整数）, 326
DTR（STL - 双精度整数到实数）, 326
DTS（STL-双整数转换为字符串）, 336

E

ED（STL - 下降沿）, 269
EM DE16 规范, 1224
EM（扩展模块）硬件/固件 ID (SMB1100-SMB1299), 1308
ENCO, 343
END, 452
ENI, 413
EU（STL - 上升沿）, 269
EXP（自然指数）, 399

F

FBD 编辑器, 163
FIFO, 469
FILL（STL-表格填充）, 471
FILL_N, 471
FND=、<>、<、>（STL-表查找）, 472
FOR, 441

G

GET, 286
GET_ADDR, 305
GET_ERROR, 453
GET_ERROR（STL-获取非致命错误代码）, 453
GIP_ADDR, 306
GRPx_RESET
ErrorStop, 1193
GSDML, 533

H

HDEF（高速计数器定义）, 356
HMI
单主站 PPI 网络, 556
多主站和多从站 PPI 网络, 557
组态以太网通信, 526
HSC（高速计数器）, 241, 356
HTA, 330

I

I/O
阶跃响应时间 (SM), 1236
模拟量输出的电流表示法, 1239
模拟量输出的电压表示法, 1238
模拟量输入的电流表示法, 1238
模拟量输入的电压表示法, 1237
模拟量状态指示灯, 147
I/O 错误状态
PLC 非致命错误 SM 标志, 1261
PLC 非致命错误代码, 1258
SMB5, 1272
I/O 扩展总线 - 通信错误 (SMW98), 1295
I/O 模块 ID 和错误寄存器 (SMB8-SMB19), 1273
I/O 寻址, 129
I_B, 326
I_BCD, 326
I_DI, 326
I_S, 336
IBCD（STL-整数到 BCD）, 326
IEC_TOF（断开延时定时器）, 486
IEC_TON（接通延时定时器）, 486
IEC_TONR（保持型接通延时定时器）, 486
INC_B, 402
INC_DW, 402
INC_W, 402
INCB（STL - 字节递增）, 402
INCD（STL - 双字递增）, 402

INCW（STL - 字递增）, 402
INV_B, 430
INV_DW, 430
INV_W, 430
INVB（STL-字节取反）, 430
INVD（STL-双字取反), 430
INVW（STL-字取反）, 430
IP 地址, 516
    MAC 地址, 524
    分配, 515, 522
    组态, 517
IP 路由器, 517
ITA, 330
ITB（STL - 整数到字节）, 326
ITD（STL - 整数到双精度整数）, 326
ITS（STL-整数转换为字符串）, 336

J

JMP, 442

L

L 存储器, 171
LAD 编辑器
    特性, 162
LBL, 442
LD（STL - 装载）, 261
LD（STL 堆栈 - 取反后立即装载）, 265
LD（STL 堆栈 - 装载）, 265
LDB<（STL - 装载比较字节小于）, 321
LDB<=（STL - 装载比较字节小于或等于）, 321
LDB<>（STL - 装载比较字节不等于）, 321
LDB=（STL - 装载比较字节等于）, 321
LDB>（STL - 装载比较字节大于）, 321
LDB>=（STL - 装载比较字节大于或等于）, 321
LDD<（STL - 装载比较双字小于）, 321
LDD<=（STL - 装载比较双字小于或等于）, 321
LDD<>（STL - 装载比较双字不等于）, 321
LDD=（STL - 装载比较双字等于）, 321
LDD>（STL - 装载比较双字大于）, 321
LDD>=（STL - 装载比较双字大于或等于）, 321
LDI（STL - 立即装载）, 263
LDI（STL 堆栈 - 立即装载）, 265
LDN（STL - 取反后装载）, 261
LDN（STL 堆栈 - 取反后装载）, 265
LDNI（STL - 取反后立即装载）, 263
LDR<（STL - 装载比较实数小于）, 321
LDR<=（STL - 装载比较实数小于或等于）, 321
LDR<>（STL - 装载比较实数不等于）, 321
LDR=（STL - 装载比较实数等于）, 321
LDR>（STL - 装载比较实数大于）, 321
LDR>=（STL - 装载比较实数大于或等于）, 321
LDS（STL 堆栈 - 装载）, 266
LDS<>（STL - 装载字符串比较不等于）, 325
LDS=（STL - 装载字符串比较等于）, 325
LDW<（STL - 装载比较字小于）, 321
LDW<=（STL - 装载比较字小于或等于）, 321
LDW<>（STL - 装载比较字不等于）, 321
LDW=（STL - 装载比较字等于）, 321
LDW>（STL - 装载比较字大于）, 321
LDW>=（STL - 装载比较字大于或等于）, 321
LED 指示灯
    CPU 状态, 146
LIFO, 469
LN（自然对数）, 399
LPF, 411
LPF（STL 低通滤波器）, 411
LPP（STL 堆栈 - 逻辑出栈）, 266
LPS（STL 堆栈 - 逻辑进栈）, 266
LRD（STL 堆栈 - 逻辑读栈）, 266
LSCR（STL-装载 SCR）, 444

M

MAC 地址, 524
MBUS_CLIENT, 604
MBUS_CTRL（初始化 Modbus 主站通信）, 586
MBUS_INIT（初始化从站通信）, 594
MBUS_MSG/MB_MSG2（从 Modbus 主站发送消息）, 587
MBUS_SERVER, 609
MBUS_SLAVE（从站对主站消息的响应）, 596
MC_AbortTrigger, 959
MC_CamIn, 1020
MC_CamOut, 1035
MC_GearIn, 1040
MC_GearOut, 1044
MC_GroupDisable, 1058
MC_GroupEnable, 1057
MC_GroupReset, 1059
MC_GroupStop, 1060
MC_Halt, 971
MC_Home, 962
MC_ImmediateStop, 994
MC_MoveAbsolute, 977
MC_MoveJog, 996
MC_MoveLinearAbsolute, 1061
MC_MoveLinearRelative, 1064
MC_MoveRelative, 982
MC_MoveVelocity, 989
MC_PhasingRelative, 1048
MC_Power, 952

MC_Reset, 961
MC_Stop, 968
MC_TorqueControl, 1001
MC_TorqueLimiting, 1017
MC_TorqueRange, 1010
MC_TouchProbe, 955
MIN_MAX_B, 404
MIN_MAX_DI, 404
MIN_MAX_I, 404
MIN_MAX_R, 404
Modbus RTU 从站
    MBUS_INIT（初始化从站通信）, 594
    MBUS_SLAVE（从站对主站消息的响应）, 596
    使用指令, 592
    执行错误代码, 597, 598
Modbus RTU 库概述, 575
Modbus RTU 主站
    MBUS_CTRL（初始化主站通信）, 586
    MBUS_MSG/MB_MSG2（从主站发送消息）, 587
    使用指令, 584
    示例程序, 598
    执行错误代码, 590
Modbus TCP
    常规异常代码, 620
MODBUS TCP
    MBUS_CLIENT, 604
    MBUS_SERVER, 609
Modbus TCP 服务器
    执行错误代码, 611
Modbus TCP 客户端
    执行错误代码, 608
Modbus TCP 库
    示例, 611
Modbus TCP 库概述, 575
Modbus TCP 库特性, 602
Modbus 常规
    Modbus 协议的初始化和执行时间, 583
    高级用户信息, 600, 617
    库特性, 580
    使用 Modbus 指令的要求, 581
    寻址, 578
MOV_B, 437
MOV_BIR, 441
MOV_BIW, 441
MOV_DW, 437
MOV_R, 437
MOV_W, 437
MOVB（STL-字节传送）, 437
MOVD（STL-双字传送）, 437
MOVR（STL-实数传送）, 437
MOVW（STL-字传送）, 437
MUL, 397
MUL_DI, 393
MUL_I, 393
MUL_R, 393

N

NAND_B, 431
NAND_DW, 431
NAND_W, 431
NANDB（STL-NAND 字节）, 431
NANDD（STL-NAND 双字）, 431
NANDW（STL-NAND 字）, 431
NEXT, 441
NOP, 273
NOR_B, 431
NOR_DW, 431
NOR_W, 431
NORD（STL-NOR 双字）, 431
NORW（STL-NOR 字）, 431
NOT (STL), 268
NXOR_B, 431
NXOR_DW, 431
NXOR_W, 431
NXORB（STL-NXOR 字节）, 431
NXORD（STL-NXOR 双字）, 431
NXORW（STL-NXOR 字）, 431

O

O（STL -“或”运算）, 261
OB<（STL - 或运算比较字节小于）, 321
OB<=（STL - 或运算比较字节小于或等于）, 321
OB<>（STL - 或运算比较字节不等于）, 321
OB=（STL - 或运算比较字节等于）, 321
OB>（STL - 或运算比较字节大于）, 321
OB>=（STL - 或运算比较字节大于或等于）, 321
OB1, 151
OD<（STL - 或运算比较双字小于）, 321
OD<=（STL - 或运算比较双字小于或等于）, 321
OD<>（STL - 或运算比较双字不等于）, 321
OD=（STL - 或运算比较双字等于）, 321
OD>（STL - 或运算比较双字大于）, 321
OD>=（STL - 或运算比较双字大于或等于）, 321
OI（STL - 立即或）, 263
OLD（STL 堆栈 - 或装载）, 266
ON（STL -“或非”运算）, 261
ONI（STL - 取反后立即或）, 263
OR<（STL - 或运算比较实数小于）, 321
OR<=（STL - 或运算比较实数小于或等于）, 321
OR<>（STL - 或运算比较实数不等于）, 321
OR=（STL - 或运算比较实数等于）, 321

OR>（STL - 或运算比较实数大于）, 321
OR>=（STL - 或运算比较实数大于或等于）, 321
ORB（STL-NOR 字节）, 431
ORB（STL-字节或）, 431
ORD（STL-双字或）, 431
ORW（STL-字或）, 431
OS<>（STL - 或运算字符串比较不等于）, 325
OS=（STL - 或运算字符串比较等于）, 325
OUC 库示例
    CheckErrors 子例程程序, 656
    被动伙伴（服务器）程序, 658
    主动伙伴（客户端）程序, 647
OUC 库示例被动伙伴（服务器）程序
    变量表, 665
OUC 库示例主动伙伴（客户端）程序
    变量表, 657
OW<（STL - 或运算比较字小于）, 321
OW<=（STL - 或运算比较字小于或等于）, 321
OW<>（STL - 或运算比较字不等于）, 321
OW=（STL - 或运算比较字等于）, 321
OW>（STL - 或运算比较字大于）, 321
OW>=（STL - 或运算比较字大于或等于）, 321

P

PID 回路控制
    PID 整定控制面板, 800
    回路定义表, 790
PID 整定控制面板, 800
PID 自整定
    PV 超限, 799
    例外情况, 798
    先决条件, 796
    序列, 796
PLC
    安装, 95
    编译及运行时间错误, 1258
    查看 PLC 信息, 180
    存储卡, 141
    非致命错误存储单元, 1261
    清除存储区, 239
    系统块, 211
    致命错误, 1262
PLC 菜单
    上传, 136
    下载, 134
PLS
    超驰, 382
    用于监视和控制 PTO 和 PWM 输出的专用存储器, 1282
    指令, 381
PN IRT 轴
    诊断, 905
PN Read Write Record 库
    概述, 575
PN Read Write 库
    PN_RD_REC 指令, 667
    PN_WR_REC 指令,
POU
    查看属性, 69
    密码保护, 69
PPI 通信
    单主站网络, 556
    多主站和多从站 PPI 网络, 557
    切换到自由端口模式, 295
    系统块中的端口组态, 55, 215
PROFINET IO
    设备名称, 534
    指定设备名称, 534
PROFINET 设备状态
    SMB1800-SMB1999, 1318
PROFINET 指令
    BLKMOV_BIR, 508
    BLKMOV_BIW, 508
    BMIR, 508
    BMIW, 508
    RDREC, 504
    WRREC, 504
PTO
    超驰, 383
PTO0、PWM0、PTO1、PWM1、PTO2、PWM2、PTO3 和 PWM3 高速输出（SMB66-SMB85、SMB166-SMB169、SMB176-SMB179、SMB566-SMB579 和 SMB580-SMB593）, 1282
PUT, 286
PWM 向导
    PWMx_RUN 子例程, 1098
PWMx_RUN, 1098

R

R（STL - 复位）, 271
R_S, 336
RAND_B, 405
RAND_DI, 405
RAND_I, 405
RAND_R, 405
RCV（接收消息控制 SMB86-SMB94、SMB405-SMB413 和 SMB186-SMB194）, 1290
READ_RTC, 280
READ_RTCX, 282
RET, 491, 497
RETI, 413

RI（STL - 立即复位）, 271
ROUND, 326
RS（LAD/FBD 复位优先双稳态触发器）, 272
RS232, 572
    通信类型, 34, 511
    通信连接数, 512
    自由端口模式, 568
RS232/PPI 电缆, 568
RS485
    设置波特率和端口网络地址, 554
    通信端口组态, 55, 215
    通信概述, 552
    通信类型, 34, 511
    通信连接数, 512
    网络组态示例, 556
RS485 地址
    分配, 560
RS485 网络
    地址, 558
    组态 CPU 的 RS485 网络地址, 558
RS485 网络地址
    组态, 558
RTA, 330
RTD 模拟量输入
    RTD 类型, 231
    标定, 231
    电阻, 231
    平滑化, 231
    系数, 231
    系统块组态, 231
    抑制, 231
RTS（STL-实数转换为字符串）, 336
RUN 模式, 86, 145

S

S（STL - 置位）, 271
S_DI, 340
S_I, 340
S_ITR, 345
S_R, 340
S_RTI, 345
S_RTR, 345
S7-200 SMART G2
    组态, 25
S7-200 SMART 安全性
    网络安全, 49
S7-200 SMART 运动控制
    闭环控制, 810
S7-200 SMART G2 硬件故障排除, 782
SB（信号板）硬件/固件 ID (SMB1050-SMB1099), 1307
SCR, 444
SCRE, 444
SCRT, 444
SEG, 326
SET_ADDR, 305
SET_RTC, 280
SET_RTCX, 282
SFND（STL-查找字符串）, 465
SHL_B, 454
SHL_DW, 454
SHL_W, 454
SHR_B, 454
SHR_DW, 454
SHR_W, 454
SHRB, 459
SI（STL - 立即置位）, 271
Siemens 提供的库, 575
SIN（正弦）, 399
SINA_PARA_S 指令
    示例, 744
SINA_POS 指令
    操作模式选择, 700
    点动模式示例, 723
    恒速模式示例, 711
    绝对运动模式示例, 706
    设置参考点模式示例, 717
    相对运动模式示例, 702
    运行程序段模式示例, 720
    增量点动模式示例, 726
    主动回零模式示例, 714
SINA_SPEED 指令
    示例,
SINAMICS 库
    SINA_PARA_S 指令,
    SINA_POS 指令, 691
    SINA_SPEED 指令, 732
    SINAMICS_Control, 688
    SINAMICS_Parameter, 688
    概述, 575
SINAMICS 库特性, 687
SIP_ADDR, 306
SLB（STL-左移字节）, 454
SLD（STL-左移双字）, 454
SLW（STL-左移字）, 454
SM 存储器，PTO/PWM 操作, 385
SM 位置（运动轴）, 1172
SM（特殊存储器）分配及功能, 1265
SMB0 系统状态位, 1268
SMB1 指令执行状态位, 1269
SMB1000-SMB1049 CPU 硬件/固件 ID, 1306
SMB1050-SMB1099 SB（信号板）硬件/固件 ID, 1307

SMB1100-SMB1299 EM（扩展模块）硬件/固件 ID, 1308
SMB130 端口 1 组态, 1278
SMB136-SMB145 (HSC3) 高速计数器 3, 1280
SMB146-SMB155 (HSC4) 高速计数器 4, 1280
SMB156-SMB165 (HSC5) 高速计数器 5, 1280
SMB1800-SMB1999 PROFINET 设备状态, 1318
SMB186-SMB194 接收消息控制, 1290
SMB195 到 SMB210
    HSC, 1298
SMB2 自由端口接收字符, 1269
SMB2000-SMB2014：PTO4/PWM4 高速输出, 1283
SMB28-SMB29 信号板 ID 和错误寄存器, 1276
SMB3 自由端口字符错误, 1270
SMB30（端口 0）、SMB31（端口 2）和 SMB130（端口 1）组态, 1278
SMB32-SMB33 信号板 ID 和错误寄存器, 1277
SMB360-SMB369 (HSC6) 高速计数器 6, 1280
SMB36-SMB45 (HSC0) 高速计数器 0, 1280
SMB370-SMB379 (HSC7) 高速计数器 7, 1280
SMB380 至 SMB404
    HSC, 1299
SMB4 中断队列溢出、运行时程序错误、中断启用、自由端口发送器空闲和强制值, 1271
SMB46-SMB55 (HSC1) 高速计数器 1, 1280
SMB480-SMB515 数据日志状态, 1300
SMB5 I/O 错误状态位, 1272
SMB566-SMB579：PTO2 和 PWM2 高速输出, 1282
SMB56-SMB65 (HSC2) 高速计数器 2, 1280
SMB580-SMB593：PTO3 和 PWM3 高速输出, 1282
SMB580-SMB593：PTO3/PWM3 高速输出, 1282
SMB600-SMB799 轴 0、1、2 和 3 运动控制, 1301
SMB66-SMB85、SMB166-SMB169 和 SMB176-SMB179：PTO0、PWM0、PTO1 和 PWM1 高速输出, 1282
SMB66-SMB85、SMB166-SMB169、SMB176-SMB179、SMB566-SMB579 和 SMB580-SMB593：PTO0、PWM0、PTO1、PWM1、PTO2、PWM2、PTO3 和 PWM3 高速输出, 1282
SMB6-SMB7 CPU ID 寄存器, 1272
SMB86-SMB94 和 SMB186-SMB194 接收消息控制, 1290
SMB8-SMB19 I/O 模块 ID 和错误寄存器, 1273
SMB929，SMB931, SMB933, SMB935 轨迹控制, 1306
SMB930，SMB932, SMB934, SMB936 轨迹状态, 1306
SMB950-SMB998 SB（信号板）硬件/固件 ID, 1307
SMW100-SMW120 系统报警, 1296
SMW22-SMW26 扫描时间, 1275
SMW98 扩展 I/O 总线 - 通信错误, 1295
SORT_B, 407
SORT_DI, 407
SORT_I, 407
SORT_R, 407
SQRT（平方根）, 399
SR（LAD/FBD 置位优先双稳态触发器）, 272
SRB（STL-右移字节）, 454
SRD（STL-右移双字）, 454
SRW（STL-右移字）, 454
SSCPY（STL-复制子字符串）, 463
SSTR_CPY, 463
STD（STL-子字符串转换为双整数）, 340
STEP 7-Micro/WIN SMART
    RS485 网络端口组态, 558
    RUN 和 STOP 模式, 86, 145
    设备要求, 35
    以太网端口组态, 517
    与 CPU 连接, 79, 518
STEP 7-Micro/WIN（早期版）, 153
STL
    逻辑堆栈操作, 266
    状态选项, 776
STL 编辑器, 164
STOP 模式, 87, 145
    模拟量输出状态, 228
    数字量输出状态, 220
    写入和强制输出, 780
STOP（指令）, 452
STR（STL-子字符串转换为实数）, 340
STR（STL-子字符串转换为整数）, 340
STR_FIND, 465
SUB_DI, 393
SUB_I, 393
SUB_R, 393
SWAP, 439

T

TAN（正切）, 399
TBL_FIND, 472
TC 模拟量输入
    TC 类型, 235
    标定, 235
    平滑化, 235
    系统块组态, 235
    抑制, 235
TC 模拟量输入模块, 235
TODR（STL - 读取日时钟）, 280
TODRX（STL - 读取扩展日时钟）, 282
TODW（STL - 写入日时钟）, 280
TODWX（STL - 写入扩展日时钟）, 282
TOF（断开延时定时器）, 476
TON（接通延时定时器）, 476

TONR（保持型接通延时定时器）, 476
TRUNC, 326

U

USS 协议库
    概述, 575, 670
    计算通信时间, 672
    使用 USS 协议指令, 673
    要求, 671
    执行错误, 686
USS 协议指令
    USS_CTRL, 676
    USS_INIT, 674
    USS_RPM_x, 679
    USS_WPM_x, 682
    使用, 673
    示例程序, 684

V

V90 驱动器, 1103, 1137

W

WAND_B, 431
WAND_DW, 431
WAND_W, 431
WDR（看门狗定时器复位）, 452
Web API, 259
WOR_B, 431
WOR_DW, 431
WOR_W, 431
WXOR_B, 431
WXOR_DW, 431
WXOR_W, 431

X

XORB（STL-字节异或）, 431
XORD（STL-双字异或）, 431
XORW（STL-字异或）, 431

安

安全, 57
安全电路, 149
安全信息, 92
安装
    DIN 导轨, 96
    安装尺寸, 94
    尺寸, 94
    灯负载, 107
    发热区, 94
    概述, 89, 95
    感性负载, 108
    将设备与热源、高压和电气噪声隔离开, 90
    接地准则, 103
    接线和接地的先决条件, 102
    接线准则, 103
    剧烈振动环境, 89
    绝缘准则, 102
    扩展模块 (I/O), 99
    气流、间隙、冷却和热区域, 94
    气流和冷却间隙, 91
    信号板 (SB), 98
    准则, 89

按

按字访问, 120
按字节访问
    字节使用表, 772

版

版本（固件）, 180

包

包络表值
    PTO 生成器, 389

保

保持范围，系统块组态, 221
保持性存储器, 138
保存项目, 85
保护等级, 1213

报

报警
    来自系统 (SMW100-SMW120), 1296
    模拟量输入组态, 227

本

本地 I/O 和扩展 I/O 寻址, 129

本地/伙伴连接, 513

比

比较指令
    比较数值, 321
    比较字符串, 325

闭

闭环控制, 810
闭环位置控制, 810

边

边缘检测器, 269

编

编辑器
    FBD, 163
    LAD, 162
    STL, 164
编译程序
    PLC 非致命程序代码, 1258
    PLC 信息（错误）, 180
    下载, 134

变

变量（变量寻址）
    间接寻址, 170

标

标志存储器, 122

表

表格指令
    ATT, 468
    FIFO/LIFO, 469
    FILL_N, 471
    TBL_FIND, 472

波

波特率
    开关选择：RS232/PPI 多主站电缆, 569
    设置, 553
    通信, 55, 215

操

操作模式
    读取 PLC 信息, 180
操作站, 149

产

产品编号, 1339

常

常闭触点
    标准, 261
    立即, 263
常规技术规范, 1209
常开触点
    标准, 261
    立即, 263

程

程序
    存储卡, 141, 142
    元素, 151
    执行有限扫描, 781
    中断例程, 152
    状态图表, 777
    子例程, 151
程序编辑器
    STL 状态选项, 776
    调试和监视, 773
    类型, 161
    使用, 81
    书签, 771
程序传送卡, 139
程序控制指令
    END、STOP 和 WDR, 452
    FOR-NEXT 循环, 441
    GET_ERROR, 453
    JMP-LBL, 442
    SCR（顺控继电器）, 444

程序块, 151
程序指令
比较, 321, 325
表, 468, 469, 471
表格查找, 472
程序控制, 441, 442, 444, 452, 453
传送, 436, 437, 438, 439, 441
定时器, 476, 486, 490
功能块, 497, 500
计数器, 349, 356, 360, 371
逻辑运算, 345, 410, 411, 430, 431
频率模式, 377
时钟, 280, 282
数学, 393, 397, 399, 402, 404
位逻辑, 261, 263, 265, 266, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277
移位与循环移位, 454, 459
中断, 413
转换, 326, 330, 336, 340, 343
子例程, 491, 493
字符串, 463, 465
程序中的书签, 771
程序状态
构建状态图表, 777
在程序编辑器中显示, 773
执行有限次数的扫描, 781

出

出厂默认存储卡, 241

触

触点
NOT, 268
常闭, 261
常闭（立即）, 263
常开, 261
常开（立即）, 263
上升沿检测器, 269, 275
下降沿检测器, 269, 275

传

传输率, 561
传送卡, 139
传送指令
SWAP（交换字中的字节数据）, 439
传送（字节、字、双字、实数）, 437
传送（字节、字、双字、实数、UDT、数组、字符串、FB 实例）, 436
块传送（字节、字和双字）, 438
字节立即传送（读取和写入）, 441

窗

窗口
浮动、停放与合并, 158

创

创建
用户定义库, 763

从

从 RUN 切换到 STOP
模拟量输出状态, 228
数字量输出状态, 220
从自由端口收到的字符错误 (SMB3), 1270

存

存储卡
程序传送卡, 141, 142
复位为出厂默认值, 241
类型, 138
使用, 138
存储器, 1261
保持范围组态, 221
非致命错误指示器的地址, 1261
清除 PLC, 239

错

错误
GET_TABLE 和 PUT_TABLE 指令, 287
I/O 错误状态, 1272
I/O 模块 ID 和错误寄存器 (SMB8-SMB19), 1273
Modbus RTU 从站执行, 597, 598
Modbus RTU 主站执行, 590
Modbus TCP 常规通信异常代码, 620
Modbus TCP 常规异常代码, 620
Modbus TCP 服务器执行, 611
Modbus TCP 客户端执行, 608
PID 控制, 1298
PID 自整定, 798
PLC 信息, 180
PWMx_RUN 子例程, 1098
编译和运行时间错误（PLC 程序）, 1258
存储单元（PLC 非致命错误）, 1261

非致命错误对运行时执行的影响, 188
时间戳不匹配（PC/PLC 程序不同）, 1257
数据保持, 221
信号板 ID 和错误寄存器 (SMB28-SMB29), 1276
信号板 ID 和错误寄存器 (SMB32-SMB33), 1277
运动指令, 1161
运动轴, 1159
致命 (PLC), 1262
致命错误对运行时执行的影响, 189
自由端口通信期间收到的字符错误 (SMB3), 1270
错误代码，单轴, 1068
错误代码、凸轮和齿轮, 1074

打

打开早期版 STEP 7-Micro/WIN 项目, 153

带

带下级 PROFINET IO 系统的智能设备, 551

灯

灯负载, 107

点

点动模式, 722

电

电磁兼容性 (EMC), 1210
电源, 93, 1253

调

调试和监视
程序编辑器状态, 773
调试与监视
强制值, 779

订

订购信息, 1339

定

定时器指令
BITIM、CITIM, 490
IEC_TON、IEC_TONR、IEC_TOF, 486
TON、TONR、TOF, 476
编程提示和示例, 479, 487
中断, 420
定时中断组态 (SMB34-SMB35), 1279
定义
局部变量, 174

丢

丢失密码, 241

动

动态 IP 信息, 517
动态伺服控制 (DSC), 810

冻

冻结输出
模拟量输出组态, 228
数字量输出组态, 220

断

断电 (PLC), 221

队

队列中断溢出 (SMB4), 1271

多

多次扫描, 781

额

额定电压, 1213

发

发热区, 91, 94

发送指令
发送数据, 296
示例, 304

访

访问权限
CPU 安全, 57
密码权限级别, 57

非

非易失性存储器, 138
非致命 PLC 错误
编译及运行时间, 1258
特殊存储器位置, 1261
非致命错误对运行时执行的影响, 188

分

分配
局部变量, 174
局部变量（局部）, 171

服

服务与支持, 4

复

复位为出厂默认存储卡, 241

感

感性负载, 108

高

高速计数器, 360

更

更新固件, 180

工

工具（选项）
STL 状态, 776
执行状态颜色指示, 773
工具菜单
PID 整定控制面板, 800
工作模式
启动选项, 223
切换为 STOP, 87, 145

功

功率要求
CPU, 93, 1253
计算, 1255
功能块图编辑器, 163
功能块指令
CALL、RET, 497
调用参数和返回示例, 500

构

构建通信网络, 561
构建状态图表, 777

固

固件更新, 139, 180

故

故障排除
LED 指示灯, 146

管

管道化
PTO 脉冲, 383

规

规范
CE 认证, 1209
CPU ST32, 1214
EM AE04, 1227
EM AM06, 1234
EM AQ02, 1230

EM AR04 (RTD), 1245
EM AT08, 1240
EM QT16, 1225
SB CM01, 1250
保护, 1213
常规技术规范, 1209
电磁兼容性 (EMC), 1210
额定电压, 1213
工业环境, 1210
过压, 1212
环境条件, 1211
继电器电气使用寿命, 1214
阶跃响应时间 (SM), 1236
绝缘, 1212
模拟量输出的电流表示法, 1239
模拟量输出的电压表示法, 1238
模拟量输入的电流表示法, 1238
模拟量输入的电压表示法, 1237
污染等级, 1212
英国合格认定标志, 1209

过

过压, 1212

恒

恒速模式, 709

环

环境
工业环境, 1210
运输和存储条件, 1211
运行条件, 1211

恢

恢复出厂设置, 73

回

回路控制 (PID)
回路定义表, 790
回收和处置, 74

计

计数器指令
标准计数器, 349
高速计数器, 356

技

技术规范, 1209
技术支持, 4

继

继电器电气使用寿命, 1214

间

间接寻址
创建指针和使用间接地址, 130
全局变量表, 170
使用指针访问表中数据的示例, 132
使用指针偏移量访问数据的示例, 133

将

将 CPU 置于 RUN 模式　RUN 模式,

交

交叉引用, 771
字节使用, 772
交流感性负载, 108

接

接收指令
断开检测, 302
结束条件, 300
结束字符检测, 302
空闲线检测, 300
奇偶校验错误, 303
起始字符检测, 301
消息定时器, 303
用户终止, 303
字符间定时器, 302
最大字符计数, 303
接线图
CPU ST32, 1222

EM AE04, 1229
EM AE08, 1229
EM AM06, 1236
EM AQ04, 1231
EM AR04 (RTD), 1249
EM AT08, 1240, 1241
EM DE16, 1225
SB CM01, 1251
接线准则, 103
DIN 导轨, 96
安装, 89
灯负载, 107
感性负载, 108
将设备与热源、高压和电气噪声隔离开, 90
接地, 103
气流和冷却间隙, 91
先决条件, 102

静

静态 IP 信息, 517

局

局部变量, 171
局部变量表, 171

剧

剧烈振动环境, 89

绝

绝对运动模式, 705
绝缘, 1212
绝缘准则, 102

卡

卡（存储）, 141, 142

开

开放式用户通信 (OUC)
连接类型, 528
连接指令, 528
开放式用户通信 (OUC) 库
DISCONNECT 指令, 643
ISO_CONNECT 指令, 628
TCP_CONNECT 指令, 625
TCP_RECV 指令, 636
TCP_SEND 指令, 633
UDP_CONNECT 指令, 631
UDP_RECV 指令, 641
UDP_SEND 指令, 638
概述, 575
共用库指令参数, 622
指令错误代码, 645
开环控制, 1095
开源软件, 40
使用, 41

客

客户支持, 4

库

库
创建用户定义库, 763
从第三方添加, 752
类型, 575
删除, 752
项目中包括, 752
用户定义, 747

快

快速拆分器, 536, 833, 837
快速访问工具栏, 35

扩

扩展 I/O 和本地 I/O 寻址, 129
扩展 I/O 总线 - 通信错误 (SMW98), 1295
扩展模块, 31, 1272
EM DE08, 1224
PLC 信息, 180
模块 ID 和错误寄存器 (SMB8-SMB19), 1273
模块错误状态 (SMB5), 1272
扩展模块 (EM)
EM AE04 规范, 1227
EM AE04 接线图, 1229
EM AE08 接线图, 1229
EM AM06 规范, 1234
EM AM06 接线图, 1236
EM AQ02 规范, 1230
EM AQ04 接线图, 1231
EM AR0 (RTD) 规范, 1245

EM AR04 (RTD) 接线图, 1249
EM AT08 规范, 1240
EM AT08 接线图, 1241
EM DE08 规范, 1224
EM DE16 接线图, 1225
EM QT16 接线图, 1226
接线图, 1240
模拟量输出的电流表示法, 1239
模拟量输出的电压表示法, 1238
模拟量输入的电流表示法, 1238
模拟量输入的电压表示法, 1237
扩展模块 (I/O)
安装和拆除, 99

冷

冷端补偿
热电偶, 237, 1242
冷却, 91

立

立即 I/O 读取/写入, 117
立即指令
LAD、FBD 和 STL, 263

连

连接
连接数 (RS232), 512
连接数 (RS485), 512
连接数（以太网）, 512
通信类型, 34, 511

联

联系人
ALT, 274
ALTP, 274
联系信息, 4

逻

逻辑、控制, 115
逻辑堆栈
STL 堆栈操作, 266
STL 输入, 265
逻辑运算指令
A, 345
NAND、NOR、NXOR（字节、字和双字）, 431
低通滤波器, 411
绝对值, 410
取反, 430
与、或、异或（字节、字和双字）, 431

滤

滤波器，数字量输入组态, 218
滤波时间, 217

脉

脉冲捕捉位，系统块中的数字量输入组态, 218
脉冲串输出 (PTO)
脉冲输出指令 (PLS), 381
指令, 242
周期时间, 382
脉宽调制 (PWM)
脉冲输出, 1098
脉冲输出指令 (PLS), 381
输出, 1096, 1097
指令, 242
周期时间, 385

密

密码
丢失或忘记, 241
权限级别, 57
密码保护, 57
POU 和数据块, 69
项目, 68

模

模块
CPU ST32, 1214
EM AE04, 1227
EM AM06, 1234
EM AQ02, 1230
EM AR04 (RTD), 1245
EM DE08, 1224
EM DE16 规范, 1224
EM QT16, 1225
SB CM01, 1250
发热区, 94
模拟量 I/O
阶跃响应时间 (SM), 1236
输出的电流表示法, 1239
输出的电压表示法, 1238

输入的电流表示法, 1238
输入的电压表示法, 1237
状态指示灯, 147
模拟量输出
RUN/STOP 转换时的状态, 228
模拟量类型组态, 228
模拟量输入
模拟量类型组态, 225
平滑化, 225
系统块组态, 225
抑制, 225

默

默认网关 IP 地址, 516

偏

偏置 PID 回路, 413
偏置且端接
CM01 信号板, 565
网络电缆, 565

屏

屏蔽连接和接地准则, 103

启

启动
启动选项, 223
启动以太网通信
STEP 7-Micro/WIN SMART, 77

气

气流, 91
气流和冷却间隙, 91

强

强制
在 STOP 模式下写入和强制输出, 780
强制值指示器 (SMB4), 1271

清

清除, 73
清除 PLC 存储区, 239
清除存储卡, 241

驱

驱动器, 1103, 1137
驱动器通信
计算时间需求, 672

热

热电偶
EM AT04 热电偶滤波器选型表, 1242
EM AT04 热电偶选型表, 1242
基本操作, 237, 1242
冷端补偿, 237, 1242
热线, 4
热源、高压和电噪声, 90

日

日时钟
读写保护, 57
扩展时钟指令, 282
时钟指令, 280

软

软件调试, 771

扫

扫描周期
扫描时间 (SMW22-SMW26), 1275
扫描速率（PLC 信息）, 180
执行单次扫描, 781
执行多次扫描, 781

上

上传程序, 136
上电后恢复数据, 145

设

设备
PROFINET IO 设备名称, 534
插入, 843
交换, 843
移动, 842

设备视图
插入模块, 545, 849
设置 CPU 类型, 84
设置参考点模式, 716

时

时间戳不匹配（PC/PLC 程序不同）, 1257
时钟指令
READ_RTC, 280
READ_RTCX, 282
SET_RTC, 280
SET_RTCX, 282

实

实数值, 127

示

示例
DISCONNECT 指令, 645
GET 和 PUT 指令, 291
ISO_CONNECT 指令, 630
Modbus RTU 从站协议，编程, 597
TCP_CONNECT 指令, 627
TCP_RECV 指令, 638
UDP_CONNECT 指令, 632
UDP_RECV 指令, 643
UDP_SEND 指令, 641
表格, 474
表格查找 (TBL_FIND) 指令, 474
单轴, 1083
高速计数器编程, 360
高速计数器初始化顺序, 371
将实数缩放为整数, 758
将整数缩放为实数, 756
开放式用户通信 (OUC) 库, 647, 656, 658
实数转换为实数, 761
使用 AXISx_ABSPOS 子程序从 SINAMICS V90 伺服驱动读取绝对位置。, 1139
缩放模拟值, 755
凸轮, 1089
位逻辑输出, 277
位逻辑输入, 276
向上/向下计数计数器指令, 351
移位寄存器位 (SHRB) 指令, 461
用户定义库, 755
用于对模拟量输入值进行采样的子例程, 152
运动轴 AXISx_CTRL、AXISx_RUN、AXISx_SEEK 和 AXISx_MAN 子程序应用, 1147
运动轴简单相对移动（定长截断应用）, 1145
运动轴组简单绝对运动（取放应用）, 1195
轴组, 1086
转换指令, 329
示例 S7-200 SMART G2 组态, 25
示例控制程序, 80

事

事件，中断, 415

首

首次扫描，执行, 781
首次扫描标志 (SMB0), 1268

输

输出
实际情况和在程序中, 115
位逻辑示例, 277
线圈, 270
写入, 117
置位和复位位, 271
置位和复位优先双稳态触发器, 272
输出映像寄存器, 116
输入
ALT 触点, 274
ALTP 触点, 274
NOT 触点, 268
STL 逻辑堆栈, 265
边缘检测器, 269
标准触点, 261, 263
读取, 116
脉冲捕捉位（系统块）, 218
实际情况和在程序中, 115
位逻辑示例, 276
输入过程映像寄存器, 116
输入滤波时间, 217

属

属性
POU 和 DB, 69

数

数据
保持, 221
接收, 297

数据块
密码保护, 69
数据块 (DB), 152
数据日志
状态 (SMB480-SMB515), 1300
数学运算指令
产生双精度整数的整数乘法和带余数的整数除法, 397
带余数的整数除法, 397
递增和递减, 402
加法、减法、乘法和除法, 393
数学函数, 399, 404
数值转换为字符串的指令, 336
数字量输出, 220
数字量输入, 217
数字量输入滤波器, 218
数字输入滤波时间, 217

速

速度环响应时间, 811
速度前馈, 811

缩

缩放
调整缩放设置, 536, 834
实数转换为实数, 761
实数转换为整数, 758
整数转换为实数, 756

锁

锁定
项目树, 69

特

特殊存储器分配和功能, 1265
特性
支持的扩展模块, 31

通

通信
IP 地址, 517
RS485 网络地址, 558
RS485 组态, 552
点对点接口 (PPI) 协议, 552
对话框, 517, 559
连接数 (RS232), 512
连接数 (RS485), 512
连接数（以太网）, 512
网络, 76
限制写入, 57
选择, 552
以太网、RS485 和 RS232, 34, 511
以太网硬件连接, 77
在 CPU 上查找 MAC 地址, 524
通信端口, 513
连接器引脚分配, 563
系统块组态, 55, 215
自由端口模式, 566
通信指令
传送, 294
接收, 294

同

同步更新（PWM 指令）, 385

凸

凸轮表, 910

图

图表
创建, 777
打开, 777
构建, 777

网

网络（通信）
RS485 网络组态示例, 556
安全注意事项, 561
单主站 PPI, 557
地址, 554
计算网络距离, 561
偏置和端接网络电缆, 565
通信类型, 34, 511
通用构建指南, 561
网络组态, 552
选择网络电缆, 563
网络连接器引脚分配, 563
网络通信, 76
网络组态示例，RS485 设备, 556

忘

忘记密码, 57, 241

位

位逻辑指令
AENO（STL - 与 ENO）, 265
ALT, 274
ALTP, 274
NOP（空操作）, 273
NOT, 268
STL 逻辑堆栈操作, 266
边缘检测器, 269
触点, 261
触点（立即）, 263
输出示例, 277
输出线圈, 270
输入的 STL 逻辑堆栈指令, 265
输入示例, 276
置位和复位位, 271
置位和复位优先双稳态触发器, 272
位置环增益（Kv 因子）, 812

文

文件菜单
上传, 136
下载, 134

污

污染等级, 1212

西

西门子技术支持, 4

系

系统报警 (SMW100-SMW120), 1296
系统块, 152
CPU 的 IP 地址, 519
CPU 的 RS485 网络地址, 559
CPU 组态, 211
RS485/RS232 CM01 通信信号板, 238
RTD 模拟量输入模块, 231
TC 模拟量输入模块, 235
密码和安全, 57
启动选项, 223
数字量输入滤波器, 218
系统状态位 (SMB0), 1268

下

下载
程序, 134
示例程序, 86

先

先决条件
使用 SINA_PARA_S 指令, 737
使用 SINA_POS 指令, 689
使用 SINA_SPEED 指令, 730
先前版 STEP 7-Micro/WIN 项目, 153

显

显示状态, 773

线

线圈
复位位, 271
立即复位位, 271
立即置位位, 271
输出, 270
输出（立即）, 270
置位位, 271

相

相对运动模式, 701

向

向导
高速计数器 (HSC), 360

项

项目
密码保护, 68
迁移, 153

写

写入值
输出, 117
在 STOP 模式下写入和强制输出, 780

新

新版本的功能, 28
新功能, 28

信

信号板 (SB)
SB CM01 规范, 1250
SB CM01 接线图, 1251
安装和拆除, 98
模拟量输出的电流表示法, 1239
模拟量输出的电压表示法, 1238
输入的电流表示法, 1238
输入的电压表示法, 1237
信号板 ID 和错误寄存器 (SMB28-SMB29), 1276
信号板 ID 和错误寄存器 (SMB32-SMB33), 1277
信号模块 (SM)
阶跃响应时间, 1236

选

选项
STL 状态, 776
选择网络电缆, 563

寻

寻址
DB2 存储器, 121
V 存储器, 121
本地 I/O 和扩展 I/O, 129
变量存储器, 121
标志存储器, 122
创建指针和使用间接地址, 130
存储区, 120
定时器存储器, 122
高速计数器, 123
过程映像输出寄存器：, 121
计数器存储器, 123
局部存储器, 126
累加器, 123
模拟量输出, 127
模拟量输入, 126
使用指针访问表中数据的示例, 132
使用指针偏移量访问数据的示例, 133
顺序控制继电器 (SCR) 存储器, 127
特殊存储器 (SM) 位, 125

移

移位和循环移位指令
位 (SHRB), 459
字节、字、双字, 454

已

已嵌套子例程/FB, 491, 497

以

以太网
GET, 286
IP 地址, 516
ISO-on-TCP 协议, 527
MAC 地址, 524
TCP 协议, 527
TSAP, 529
UDP 协议, 527
端口, 528
通信类型, 34, 511
通信连接数, 512
组态 CPU 与 HMI 设备之间的通信, 526
以太网通信
STEP 7-Micro/WIN SMART V3 设置, 77
以太网网络
搜索 CPU, 523
组态 CPU 的 IP 地址, 517

抑

抑制电路, 108

应

应用程序用户参考手册, 158

英

英国合格认定标志, 1209

硬

硬件故障排除, 782
硬件配置
    添加模块, 545, 849
硬件网络拓扑
    功能, 535, 833
    设备视图, 837

用

用户参考手册, 158
用户定义库
    编程提示, 747
    示例, 755
用于定时中断的 SMB34-SMB35 时间间隔值, 1279

语

语句表编辑器 (STL), 164

元

元素使用, 771

运

运动控制
    定义运动曲线, 1120
    极性, 1114
    输入点动参数, 1116
    输入急停时间, 1117
    输入加速时间, 1116
    输入最大启动和停止速度, 1115
    相位, 1112
    映射 I/O, 1109
    运动控制资源, 224
    运动特性, 1102
    组态参考点和搜索参数, 1118
    组态反冲补偿, 1118
    组态输入引脚位置, 1108
运动控制面板
    显示和控制轴操作, 1153
    显示和修改轴组态, 1158
    显示运动轴的曲线组态, 1158
运动控制中的步进电机, 1098
运动曲线
    操作模式, 1101
    创建步, 1102
    定义, 1101
    组态, 1100
运动输入和输出
    CPU, 1103
运动衔接方式
    打断模式, 1184
    缓冲模式, 1184
运动向导
    SM 位置, 1172
    开环运动控制, 1095
    组态/曲线表, 1162
    最大速度和启动/停止速度, 1099
运动指令，错误代码, 1161
运动轴
    ACCEL_TIME, 1100
    AXISx_ABSPOS, 1137
    AXISx_CACHE, 1135
    AXISx_CFG, 1134
    AXISx_CTRL, 1123
    AXISx_DIS, 1133
    AXISx_GOTO, 1126
    AXISx_LDOFF, 1130
    AXISx_LDPOS, 1131
    AXISx_MAN, 1124
    AXISx_RDPOS, 1136
    AXISx_RSEEK, 1129
    AXISx_RUN, 1128
    AXISx_SRATE, 1132
    RP 搜索模式, 1175
    SM 位置, 1172
    编程, 1104
    错误代码, 1159
    定义运动曲线, 1120
    回原点, 873
    极性, 1114
    模数, 857, 865, 880, 886
    设置加加速度时间, 860, 871, 891
    设置加速和减速时间, 859, 870, 890
    输入点动参数, 1116
    输入急停时间, 1117
    输入加速时间, 1116
    输入最大启动和停止速度, 1115
    探针, 875, 881
    显示和控制轴操作, 1153
    显示和修改轴组态, 1158
    显示运动轴的曲线组态, 1158
    相位, 1112
    消除反冲, 1178
    选择测量系统, 857, 864, 879, 886
    选择方向控制和输出组态, 862
    选择方向控制和输入组态, 877
    选择轴名称, 856, 861, 877, 884

映射 I/O, 1109
原点漂移, 873
诊断, 860, 876, 883
指定电机速度, 859, 868, 890
轴状态, 861, 876, 883, 905
子例程, 1121
子例程准则, 1122
组态, 1106
组态 LMT- 输入, 866, 887
组态 LMT+ 输入, 866, 887
组态 RPS 输入, 871
组态 STP 输入, 874
组态 ZP 输入, 872
组态参考点和搜索参数, 1118
组态反冲补偿, 1118
组态输入引脚位置, 1108
运动轴组
GRPx_2D_MOVELINEAR, 1187
GRPx_3D_MOVELINEAR, 1187
GRPx_MOVEPATH, 1190
GRPx_RESET, 1192
编程, 1179
路径, 1181
线段, 1181
选择轴名称, 906
诊断, 908
轴组状态, 908
运行程序段模式, 719
运行时间和 PLC 编译错误, 1258

增

增量点动模式, 725

诊

诊断
LED 指示灯, 146

振

振动, 89

支

支持, 4

执

执行
程序, 117
单次或多次扫描, 781

直

直流感性负载, 108

指

指令
GET, 286
GET_ADDR, 305
GIP_ADDR, 306
MBUS_CLIENT, 604
MBUS_SERVER, 609
PUT, 286
SET_ADDR, 305
SIP_ADDR, 306
回路控制 (PID), 412
快速参考指南, 1325
指令执行状态位 (SMB1), 1269
指针
创建指针和使用间接地址, 130
使用指针访问表中数据的示例, 132
使用指针偏移量访问数据的示例, 133

致

致命错误 (PLC), 1262
致命错误对运行时执行的影响, 189

智

智能设备, 531
不带下级 PROFINET IO 系统的智能设备, 532
带下级 PROFINET IO 系统的智能设备, 532

置

置位和复位优先双稳态触发器指令, 272
置位和立即复位指令, 271

中

中断
CPU 型号的事件支持, 415
编程准则, 418
定时中断的时间间隔值 (SMB34-SMB35), 1279
概述, 415
连接/分离、启用/禁用、有条件返回和清除事件指令, 413
嵌套子例程/FB, 491, 497
全局中断启用状态 (SMB4), 1271
示例程序, 420
优先级和排队, 420
中断队列溢出 (SMB4), 1271
中断事件类型, 419
中断例程, 117
用户程序的元素, 152
中继器, 562

重

重叠
字节, 772
重复的 POU 名称分配, 69

轴

轴 0 运动控制 (SMB600-SMB649), 1301
轴 1 运动控制 (SMB650-SMB699), 1301
轴 2 运动控制 (SMB700-SMB749), 1301
轴 3 运动控制 (SMB750-SMB799), 1301
轴组 0 SM 地址 (SMB800-SMB848), 1304
轴组 1 SM 地址 (SMB870-SMB928), 1305
轴组的错误代码, 1205
轴组组态
运动衔接方式, 1184

主

主程序, 151
主动/被动通信伙伴, 514
主动回零模式, 713

注

注释
组态页面外观, 853

转

转换指令
ASCII 数组转换, 330
ASCII 子字符串转换为数值, 340
编码和解码, 343
标准转换, 326
数值转换为字符串, 336

状

状态
构建状态图表, 777
在程序编辑器中显示, 773
执行有限次数的扫描, 781
状态 LED
CPU, 146
PROFINET 网络, 534
扩展模块 (EM), 146
状态错误（时间戳不匹配）, 1257

准

准则
安装步骤, 95
隔离, 102
接地, 103
接线, 103
接线和接地的先决条件, 102

子

子例程
PWMx_RUN, 1098
用户程序元素, 151
运动轴, 1121
运动轴组, 1186
准则, 1122
子例程/FB
嵌套深度, 491, 497
总数, 491, 497
子例程/FB 递归, 491, 497
子例程指令
CALL、RET, 491
调用参数和返回示例, 493

自

自由端口模式
SMB405-SMB413 接收消息控制, 1290
SMB86-SMB94 和 SMB186-SMB194 接收消息控制, 1290
启用, 295
示例, 567
中断, 419
自由端口发送器空闲 (SMB4), 1271
自由端口接收字符 (SMB2), 1269
自由端口字符错误 (SMB3), 1270
自由端口组态（SMB30 端口 0、SMB31 端口 2 和 SMB130 端口 1）, 1278
字符中断控制, 303

字

字符串
表示方式, 128
格式, 128
字符串指令
查找字符串/字符, 465
复制子字符串, 463

组

组件属性, 69
组态
CPU，系统块, 211
IP 地址, 517
MAC 地址, 524
RS485 网络, 558
RS485 网络地址, 558
动态 IP 信息, 517
静态 IP 信息, 519
曲线表（运动轴）, 1162
以太网, 517
组态触发条件, 205
组态图, 149