SIEMENS

SIMATIC

S7
S7-200 SMART

系统手册

前言

产品概述 1

入门指南 2

安装 3

PLC 概念 4

编程概念 5

PLC 设备组态 6

程序指令 7

通信 8

库 9

调试和故障排除 10

PID 回路和整定 11

开环运动控制 12

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

11/2016

A5E03822234-AE

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。
由于具备相关培训及经验，合格人员可以察觉本产品/系统的风险，并避免可能的危险。

按规定使用Siemens 产品

请注意下列说明：

⚠警告
Siemens 产品只允许用于目录和相关技术文件中规定的使用情况。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 ® 的都是西门子股份有限公司的注册商标。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 AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
德国

A5E03822234-AE
℗ 10/2016 本公司保留更改的权利

前言

手册用途

S7-200 SMART 系列包括许多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 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 产品多种多样且提供基于 Windows 的编程工具，这使得您可以灵活地解决各种自动化问题。

本手册提供了有关 S7-200 SMART 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

要理解本手册，需要具备自动化和可编程逻辑控制器的基本知识。

本手册适用范围

本手册介绍了以下产品：

- STEP 7-Micro/WIN SMART V2.2
- S7-200 SMART CPU 固件版本 V2.2

要了解手册中 S7-200 SMART 产品和产品编号的完整列表，请参见技术规范 (页 719)。

证书、CE 标签和其它标准

有关详细信息，请参见技术规范。

服务与支持

除了文档之外，我们还在 Internet 的客户支持网站 (http://www.siemens.com/automation/)上提供了专业技术知识。

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。

安全信息

Siemens
为其产品及解决方案提供了工业安全功能，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安全保护机制。Siemens 的产品和解决方案仅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。只有在必要时并采取适当安全措施（例如，使用防火墙和网络分段）的情况下，才能将系统、机器和组件连接到企业网络或 Internet。

此外，应考虑遵循 Siemens
有关相应安全措施的指南。更多有关工业安全的信息，请访问
(http://www.siemens.com/industrialsecurity)。

Siemens 不断对产品和解决方案进行开发和完善以提高安全性。Siemens
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 Siemens 工业安全 RSS 源，网址为
(http://support.automation.siemens.com)。

目录

前言 3

1 产品概述 19

1.1 S7-200 SMART CPU 20

1.2 S7-200 SMART 扩展模块 23

1.3 适用于 S7-200 SMART 的 HMI 设备 24

1.4 通信选项 25

1.5 编程软件 26

1.6 新功能 27

2 入门指南 29

2.1 连接到 CPU 29
2.1.1 组态 CPU 以进行通信 30
2.1.1.1 概述 30
2.1.1.2 建立硬件通信连接 31
2.1.1.3 与 CPU 建立通信 32

2.2 创建示例程序 35
2.2.1 程序段 1： 启动定时器 37
2.2.2 程序段 2： 接通输出 38
2.2.3 程序段 3： 复位定时器 39
2.2.4 为项目设置 CPU 的类型和版本 40
2.2.5 保存示例项目 41

2.3 下载示例程序 42

2.4 更改 CPU 的工作模式 43

3 安装 45

3.1 S7-200 SMART 设备安装准则 45

3.2 功率预算 47

3.3 安装和拆卸步骤 49
3.3.1 S7-200 SMART 设备的安装尺寸 49
3.3.2 安装和拆卸 CPU 50
3.3.3 安装和拆卸信号板或电池板 53
3.3.4 拆卸和重新安装端子块连接器 54
3.3.5 安装和拆卸扩展模块 56
3.3.6 安装和卸下扩展电缆 57

3.4 接线准则 59

4 PLC 概念 67

4.1 控制逻辑的执行 67
4.1.1 读取输入和写入输出 69
4.1.2 立即读取或写入 I/O 70
4.1.3 执行用户程序 70

4.2 访问数据 72
4.2.1 访问存储区 73
4.2.2 实数格式 81
4.2.3 字符串格式 81
4.2.4 分配指令的常数值 82
4.2.5 对本地 I/O 和扩展 I/O 进行寻址 83
4.2.6 使用指针进行间接寻址 84
4.2.7 指针示例 85

4.3 保存和恢复数据 88
4.3.1 下载项目组件 88
4.3.2 上传项目组件 91
4.3.3 存储类型 92
4.3.4 使用存储卡 93
4.3.5 在 CPU 中插入存储卡 95
4.3.6 通过存储卡传送程序 95
4.3.7 上电后恢复数据 98

4.4 更改 CPU 的工作模式 98

4.5 状态 LED 99

5 编程概念 101

5.1 设计 PLC 系统的指南 101

5.2 用户程序的元素 103

5.3 创建用户程序 106
5.3.1 早期版本的 STEP 7-Micro/WIN 项目 106
5.3.2 使用 STEP 7-Micro/WIN SMART 用户界面 109
5.3.3 使用 STEP 7-Micro/WIN SMART 创建程序 110
5.3.4 借助向导创建控制程序 112
5.3.5 LAD 编辑器的特点 112
5.3.6 FBD 编辑器的特点 113
5.3.7 STL 编辑器的特点 114

5.4 数据块 (DB) 编辑器 115

5.5 符号表 117

5.6 变量表 121

5.7 PLC 错误响应 127
5.7.1 非致命错误和 I/O 错误 128
5.7.2 致命错误 130
5.8 在 RUN 模式下执行程序编辑 131
5.9 用于调试程序的功能 134
6 PLC 设备组态 135
6.1 组态 PLC 系统的运行 135
6.1.1 系统块 135
6.1.2 对通信进行组态 137
6.1.3 组态数字量输入 139
6.1.4 组态数字量输出 141
6.1.5 组态保持范围 142
6.1.6 组态系统安全 144
6.1.7 组态启动选项 148
6.1.8 组态模拟量输入 149
6.1.9 模拟量输入技术规范参考 151
6.1.10 组态模拟量输出 152
6.1.11 模拟量输出技术规范参考 154
6.1.12 组态 RTD 模拟量输入 154
6.1.13 组态 TC 模拟量输入 159
6.1.14 组态 RS485/RS232 CM01 通信信号板 163
6.1.15 组态 BA01 电池信号板 165
6.1.16 清除 PLC 存储区 166
6.1.17 创建复位为出厂默认存储卡。 168
6.2 高速 I/O 169
7 程序指令 171
7.1 位逻辑 171
7.1.1 标准输入 171
7.1.2 立即输入 173
7.1.3 逻辑堆栈概述 174
7.1.4 STL 逻辑堆栈指令 176
7.1.5 NOT 178
7.1.6 正跳变和负跳变检测器 179
7.1.7 线圈: 输出和立即输出指令 180
7.1.8 置位、复位、立即置位和立即复位功能 181
7.1.9 置位和复位优先双稳态触发器 182
7.1.10 NOP（空操作）指令 184
7.1.11 位逻辑输入示例 184
7.1.12 位逻辑输出示例 186
7.2 时钟 188
7.2.1 读取和设置实时时钟 188

7.2.2 读取和设置扩展实时时钟 191
7.3 通信 196
7.3.1 GET 和 PUT（以太网） 196
7.3.2 发送和接收（RS485/RS232 为自由端口） 203
7.3.3 获取端口地址和设置端口地址（RS485/RS232 上的 PPI 协议） 219
7.3.4 获取 IP 地址和设置 IP 地址（以太网） 220
7.3.5 开放式用户通信 221
7.3.5.1 OUC 指令 222
7.3.5.2 OUC 指令错误代码 234
7.4 比较 236
7.4.1 比较数值 236
7.4.2 比较字符串 240
7.5 转换 242
7.5.1 标准转换指令 242
7.5.2 ASCII 字符数组转换 246
7.5.3 数值转换为 ASCII 字符串 252
7.5.4 ASCII 子字符串转换为数值 257
7.5.5 编码和解码 260
7.6 计数器 262
7.6.1 计数器指令 262
7.6.2 高速计数器指令 267
7.6.3 高速输入降噪 271
7.6.4 高速计数器编程 273
7.6.5 高速计数器的初始化顺序示例 286
7.7 脉冲输出 294
7.7.1 脉冲输出指令 (PLS) 294
7.7.2 脉冲串输出 (PTO) 296
7.7.3 脉宽调制 (PWM) 298
7.7.4 使用 SM 位置组态和控制 PTO/PWM 操作 299
7.7.5 计算包络表值 304
7.8 数学 308
7.8.1 加法、减法、乘法和除法 308
7.8.2 产生双整数的整数乘法和带余数的整数除法 312
7.8.3 三角函数、自然对数/自然指数和平方根 314
7.8.4 递增和递减 317
7.9 PID 319
7.9.1 使用 PID 向导 321
7.9.2 PID 算法 326
7.9.3 转换和标准化回路输入 330
7.9.4 将回路输出转换为标定整数值 331
7.9.5 正作用或反作用回路 332

7.10 中断 335
7.10.1 中断指令 335
7.10.2 中断例程概述和 CPU 型号事件支持 337
7.10.3 中断编程准则 339
7.10.4 S7-200 SMART CPU 支持的中断事件类型 341
7.10.5 中断优先级、排队和示例程序 342

7.11 逻辑运算 348
7.11.1 取反 348
7.11.2 与、或和异或 349

7.12 传送 351
7.12.1 字节、字、双字或实数传送 351
7.12.2 块传送 352
7.12.3 交换字节 354
7.12.4 字节立即传送（读取和写入） 355

7.13 程序控制 356
7.13.1 FOR-NEXT 循环 356
7.13.2 JMP（跳转至标号） 358
7.13.3 SCR（顺控继电器） 359
7.13.4 END、STOP 和 WDR（看门狗定时器复位） 368
7.13.5 GET_ERROR（获取非致命错误代码） 370

7.14 移位与循环移位 371
7.14.1 移位和循环移位 371
7.14.2 移位寄存器位 375

7.15 字符串 377
7.15.1 字符串（获取长度、复制和连接） 377
7.15.2 从字符串中复制子字符串 379
7.15.3 在字符串中查找字符串和第一个字符 380

7.16 表 383
7.16.1 添表 383
7.16.2 先进先出和后进先出 385
7.16.3 存储器填充 387
7.16.4 查表 388

7.17 定时器 392
7.17.1 定时器指令 392
7.17.2 定时器编程提示和示例 396
7.17.3 时间间隔定时器 403

7.18 子例程 405
7.18.1 CALL（子例程）和 RET（有条件返回） 405

8 通信 413

8.1 CPU 通信连接 414
8.2 CPU 通信端口 415
8.3 HMI 和通信驱动程序 416
8.4 以太网 418
8.4.1 概述 418
8.4.2 本地/伙伴连接 418
8.4.3 以太网网络组态示例 419
8.4.4 分配 Internet 协议 (IP) 地址 420
8.4.4.1 为编程设备和网络设备分配 IP 地址 420
8.4.4.2 为项目中的 CPU 或设备组态或更改 IP 地址 422
8.4.4.3 搜索以太网网络上的 CPU 和设备 429
8.4.5 查找 CPU 上的以太网 (MAC) 地址 431
8.4.6 HMI 与 CPU 通信 432
8.4.7 开放式用户通信 434
8.4.7.1 协议 434
8.4.7.2 连接 435
8.4.7.3 端口和 TSAP 435
8.5 PROFIBUS 438
8.5.1 EM DP01 PROFIBUS DP 模块 440
8.5.1.1 分布式外设 (DP) 标准通信 440
8.5.1.2 使用 EM DP01 将 S7-200 SMART 连接为 DP 设备 441
8.5.1.3 组态 EM DP01 443
8.5.1.4 数据一致性 443
8.5.1.5 支持的组态 445
8.5.1.6 安装 EM DP01 GSD 文件 446
8.5.1.7 组态 EM DP01 I/O 448
8.5.1.8 V 存储器和 I/O 地址区域的示例 451
8.5.1.9 用户程序注意事项 452
8.5.1.10 EM DP01 PROFIBUS DP 的 LED 状态指示灯 455
8.5.1.11 使用 HMI 和配有 EM DP01 的 S7-CPU 457
8.5.1.12 设备数据库文件：GSD 458
8.5.1.13 与 CPU 进行 PROFIBUS DP 通信的示例程序 463
8.5.1.14 EM DP01 PROFIBUS DP 模块技术规范参考 465
8.6 RS485 466
8.6.1 PPI 协议 467
8.6.2 波特率和网络地址 468
8.6.2.1 波特率和网络地址定义 468
8.6.2.2 为 S7-200 SMART CPU 设置波特率和网络地址 469
8.6.3 RS485 网络组态示例 471
8.6.3.1 单主站 PPI 网络 471
8.6.3.2 多主站和多从站 PPI 网络 472

8.6.4 构建网络 473
8.6.4.1 通用准则 473
8.6.4.2 确定网络的距离、传输率和电缆长度 473
8.6.4.3 网络中的中继器 474
8.6.4.4 选择网络电缆 475
8.6.4.5 连接器引脚分配 476
8.6.4.6 偏置和端接网络电缆 477
8.6.4.7 偏置和端接 CM01 信号板 479
8.6.4.8 在 RS485 网络中使用 HMI 设备 479
8.6.5 自由端口模式 480
8.6.5.1 使用自由端口模式创建用户定义的协议 480
8.6.5.2 对 RS232 设备使用 RS232/PPI 多主站电缆和自由端口模式 483

8.7 RS232 485

9 库 487

9.1 库类型（Siemens 及用户定义） 487

9.2 Modbus 通信概述 489
9.2.1 Modbus 寻址 489
9.2.2 Modbus 读取和写入功能 491

9.3 Modbus RTU 库 492
9.3.1 Modbus 通信概述 492
9.3.1.1 Modbus RTU 库功能 492
9.3.1.2 使用 Modbus 指令的要求 493
9.3.1.3 Modbus 协议的初始化和执行时间 495
9.3.2 Modbus RTU 主站 496
9.3.2.1 使用 Modbus RTU 主站指令 496
9.3.2.2 MBUS_CTRL/MB_CTRL2 指令（初始化主站） 497
9.3.2.3 MBUS_MSG/MB_MSG2 指令 499
9.3.2.4 Modbus RTU 主站执行错误代码 503
9.3.3 Modbus RTU 从站 505
9.3.3.1 使用 Modbus RTU 从站指令 505
9.3.3.2 MBUS_INIT 指令（初始化从站） 507
9.3.3.3 MBUS_SLAVE 指令 509
9.3.3.4 Modbus RTU 从站执行错误代码 511
9.3.4 Modbus RTU 主站示例程序 512
9.3.5 Modbus RTU 高级用户信息 515

9.4 开放式用户通信库 517
9.4.1 OUC 库指令共用的参数 518
9.4.2 开放式用户通信库指令 521
9.4.2.1 TCP_CONNECT 指令 521
9.4.2.2 ISO_CONNECT 指令 524
9.4.2.3 UDP_CONNECT 指令 527
9.4.2.4 TCP_SEND 指令 529

9.4.2.5 TCP_RECV 指令 532
9.4.2.6 UDP_SEND 指令 536
9.4.2.7 UDP_RECV 指令 539
9.4.2.8 DISCONNECT 指令 543
9.4.3 开放式用户通信库指令错误代码 545
9.4.4 开放式用户通信库示例 547
9.4.4.1 主动伙伴（客户端） 547
9.4.4.2 CheckErrors 子例程 557
9.4.4.3 主动伙伴符号表 558
9.4.4.4 被动伙伴（服务器） 559
9.4.4.5 CheckErrors 子例程 567
9.4.4.6 被动伙伴符号表 568

9.5 USS 库 569
9.5.1 USS 通信概述 569
9.5.1.1 USS 协议概述 569
9.5.1.2 使用 USS 协议的要求 570
9.5.1.3 计算与驱动器通信所需的时间 571
9.5.2 USS 程序指令 572
9.5.2.1 使用 USS 协议指令 572
9.5.2.2 USS_INIT 指令 573
9.5.2.3 USS_CTRL 指令 576
9.5.2.4 USS_RPM_x 指令 581
9.5.2.5 USS_WPM_x 指令 584
9.5.2.6 USS 协议执行错误代码 588
9.5.2.7 USS 协议示例程序 589

9.6 创建用户定义的指令库 592

10 调试和故障排除 593

10.1 调试程序 593
10.1.1 书签功能 593
10.1.2 交叉引用表 594

10.2 显示程序状态 596
10.2.1 显示程序编辑器中的状态 596
10.2.2 组态 STL 状态选项 599

10.3 使用状态图以监视程序 600

10.4 强制特定值 603

10.5 在 STOP 模式下写入和强制输出 604

10.6 如何执行有限次数的扫描 605

10.7 硬件故障排除指南 607

11 PID 回路和整定 609

11.1 PID 回路定义表 610
11.2 先决条件 614
11.3 自滞后和自偏差 614
11.4 自整定序列 615
11.5 例外情况 617
11.6 关于过程变量超限的说明（结果代码 3） 618
11.7 PID 整定控制面板 619

12 开环运动控制 625

12.1 使用 PWM 输出 626
12.1.1 组态 PWM 输出 626
12.1.2 PWMx_RUN 子例程 628
12.2 使用运动控制 629
12.2.1 最大速度和启动/停止速度 629
12.2.2 输入加速和减速时间 630
12.2.3 组态运动包络 631
12.3 运动控制的特点 634
12.4 编程运动轴 636
12.5 组态运动轴 637
12.6 运动向导为运动轴创建的子例程 651
12.6.1 运动控制子例程使用准则 652
12.6.2 AXISx_CTRL 子例程 653
12.6.3 AXISx_MAN 子例程 654
12.6.4 AXISx_GOTO 子例程 656
12.6.5 AXISx_RUN 子例程 658
12.6.6 AXISx_RSEEK 子例程 659
12.6.7 AXISx_LDOFF 子例程 660
12.6.8 AXISx_LDPOS 子例程 662
12.6.9 AXISx_SRATE 子例程 663
12.6.10 AXISx_DIS 子例程 664
12.6.11 AXISx_CFG 子例程 665
12.6.12 AXISx_CACHE 子例程 666
12.6.13 AXISx_RDPOS 子例程 667
12.6.14 AXISx_ABSPOS 子例程 668
12.7 使用 AXISx_ABSPOS 子程序从 SINAMICS 伺服驱动读取绝对位置 670
12.7.1 AXISx_ABSPOS 和 AXISx_LDPOS 子程序应用示例 670
12.7.2 互连 672
12.7.3 调试 672

12.7.3.1 控制模式 672
12.7.3.2 设定值脉冲输入通道 672
12.7.3.3 设定值脉冲串输入格式 673
12.7.3.4 共同的工程单位基础 673
12.7.4 重要事项须知 676

12.8 运动轴示例程序 677
12.8.1 运动轴简单相对移动（定长截断应用）示例 677
12.8.2 运动轴 AXISx_CTRL、AXISx_RUN、AXISx_SEEK 和 AXISx_MAN 示例 679

12.9 监视运动轴 686
12.9.1 显示和控制运动轴的操作 687
12.9.2 显示和修改运动轴的组态 692
12.9.3 显示运动轴的曲线组态 693
12.9.4 运动轴错误代码（SMW620、SMW670 或 SMW720 的 WORD） 694
12.9.5 运动指令的错误代码（SMB634、SMB684 或 SMB734 的七个 LS 位） 696

12.10 高级主题 698
12.10.1 理解运动轴的组态/曲线表 698
12.10.2 运动轴的特殊存储器 (SM) 位置 709

12.11 了解运动轴的 RP 搜索模式 712
12.11.1 选择工作区位置以消除反冲 717

A 技术规范 719

A.1 常规规范 719
A.1.1 常规技术规范 719

A.2 S7-200 SMART CPU 724
A.2.1 CPU ST20 和 CPU SR20 724
A.2.1.1 常规规范和特性 724
A.2.1.2 数字量输入和输出 728
A.2.1.3 CPU ST20 和 CPU SR20 接线图 731
A.2.2 CPU ST30 和 CPU SR30 733
A.2.2.1 常规规范和特性 733
A.2.2.2 数字量输入和输出 738
A.2.2.3 CPU ST30 和 CPU SR30 接线图 741
A.2.3 CPU ST40、CPU SR40 和 CPU CR40 744
A.2.3.1 常规规范和特性 744
A.2.3.2 数字量输入和输出 748
A.2.3.3 CPU ST40、SR40 和 CR40 接线图 752
A.2.4 CPU ST60、CPU SR60 和 CPU CR60 757
A.2.4.1 常规规范和特性 757
A.2.4.2 数字量输入和输出 761
A.2.4.3 CPU ST60、SR60 和 CR60 接线图 765
A.2.5 漏型、源型输入和继电器输出的接线图 769

A.3 数字量输入和输出扩展模块 (EM) 770

A.3.1 EM DE08 和 EM DE16 数字量输入规范 770
A.3.2 EM DT08、EM DR08、EM QR16 和 EM QT16 数字量输出规范 773
A.3.3 EM DT16、EM DR16、EM DT32 和 EM DR32 数字量输入/输出规范 780

A.4 模拟量输入和输出扩展模块 (EM) 788
A.4.1 EM AE04 和 EM AE08 模拟量输入规范 788
A.4.2 EM AQ02 和 EM AQ04 模拟量输出模块规范 792
A.4.3 EM AM03 和 EM AM06 模拟量输入/输出模块规范 796
A.4.4 模拟量输入的阶跃响应 800
A.4.5 模拟量输入的采样时间和更新时间 801
A.4.6 模拟量输入的电压和电流测量范围（SB 和 SM） 801
A.4.7 模拟量输出的电压和电流测量范围（SB 和 EM） 803

A.5 热电偶模块和 RTD 扩展模块 (EM) 805
A.5.1 热电偶扩展模块 (EM) 805
A.5.1.1 EM AT04 热电偶规范 805
A.5.2 RTD 扩展模块 (EM) 811

A.6 数字信号板 818
A.6.1 SB DT04 数字量输入/输出规范 818

A.7 模拟信号板 821
A.7.1 SB AE01 模拟量输入规范 821
A.7.2 SB AQ01 模拟量输出规范 825

A.8 RS485/RS232 信号板 827
A.8.1 SB RS485/RS232 规范 827

A.9 电池板信号板 (SB) 830
A.9.1 SB BA01 电池板 830

A.10 EM DP01 PROFIBUS DP 模块 832
A.10.1 支持 EM DP01 PROFIBUS DP 模块的 S7-200 SMART CPU 834
A.10.2 EM DP01 连接器引脚分配 835
A.10.3 EM DP01 PROFIBUS DP 模块接线图 836

A.11 S7-200 SMART I/O 扩展电缆 837

B 计算功率预算 839

B.1 功率预算 839

B.2 功率要求计算示例 841

B.3 计算功率要求 842

C 错误代码 843
C.1 时间戳不匹配 843
C.2 PLC 非致命错误代码 844
C.3 PLC 非致命错误 SM 标志 847
C.4 PLC 致命错误代码 848
D 特殊存储器 (SM) 和系统符号名称 851
D.1 SM（特殊存储器）概述 851
D.2 SMB0： 系统状态 854
D.3 SMB1： 指令执行状态 855
D.4 SMB2： 自由端口接收字符 856
D.5 SMB3： 自由端口字符错误 857
D.6 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 858
D.7 SMB5： I/O 错误状态 859
D.8 SMB6-SMB7： CPU ID、错误状态和数字量 I/O 点 859
D.9 SMB8-SMB19： I/O 模块 ID 和错误 860
D.10 SMW22-SMW26： 扫描时间 861
D.11 SMB28-SMB29： 信号板 ID 和错误 862
D.12 SMB30： （端口 0）和 SMB130： （端口 1） 863
D.13 SMB34-SMB35： 定时中断的时间间隔 864
D.14 SMB36-45 (HSC0)、SMB46-55 (HSC1)、SMB56-65 (HSC2)、SMB136-145 (HSC3)：高速计数器 865
D.15 SMB66-SMB85、SMB166-SMB169、SMB176-SMB179 和 SMB566-SMB579：PTO0、PWM0、PTO1、PWM1、PTO2 和 PWM2 高速输出 870
D.16 SMB86-SMB94 和 SMB186-SMB194： 接收信息控制 874
D.17 SMW98： I/O 扩展总线通信错误 876
D.18 SMW100-SMW114 系统报警 877
D.19 SMB130： 端口 1 的自由端口控制（请参见 SMB30） 878
D.20 SMB136-SMB145： HSC3 高速计数器 878
D.21 SMB186-SMB194： 接收消息控制（请参见 SMB86-SMB94） 878
D.22 SMB480-SMB515： 数据日志状态 878
D.23 SMB600-SMB749： 轴（0、1 和 2）开环运动控制 879

D.24 SMB650-SMB699： 轴 1 开环运动控制（请参见 SMB600-SMB740） 881
D.25 SMB700-SMB749： 轴 2 开环运动控制（请参见 SMB600-SMB740） 881
D.26 SMB1000-SMB1049： CPU 硬件/固件 ID 882
D.27 SMB1050-SMB1099： SB（信号板）硬件/固件 ID 883
D.28 SMB1100-SMB1399： EM（扩展模块）硬件/固件 ID 884
D.29 SMB1400-SMB1699：EM（扩展模块）模块特定的数据 887
E 参考 889
E.1 常用特殊存储器位 889
E.2 按优先级别顺序排列的中断事件 890
E.3 高速计数器汇总 892
E.4 指令 893
E.5 存储器范围和特性 902
F 订购信息 905
F.1 CPU 模块 905
F.2 扩展模块 (EMs) 和信号板 (SBs) 906
F.3 编程软件 907
F.4 通信 907
F.5 备件和其它硬件 908
F.6 人机界面设备 911
索引 913

产品概述

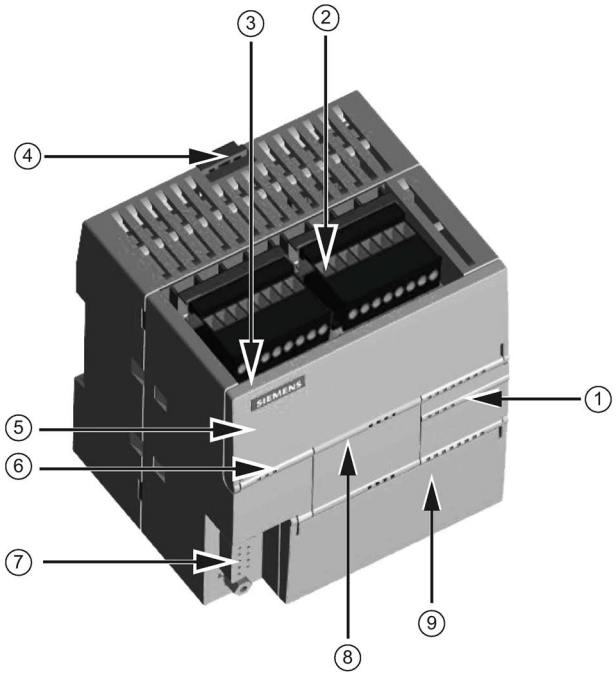
1

S7-200 SMART 系列微型可编程逻辑控制器 (Micro PLC, Micro Programmable Logic Controller) 可以控制各种设备以满足您的自动化控制需要。

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
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。

1.1 S7-200 SMART CPU

该 CPU
将集成电源、输入和输出电路组合到一个设计紧凑的外壳中来形成功能强大的微型 PLC。在您下载用户程序后，CPU 将包含监控应用中输入输出设备所需的逻辑。



① I/O 的 LED
② 端子连接器
③ 以太网通信端口
④ 用于在标准 (DIN) 导轨上安装的夹片
⑤ 以太网状态 LED（保护盖下面）：LINK，RX/TX
① 状态 LED：RUN、STOP 和 ERROR
① RS485 通信端口
① 可选信号板（仅限标准型）
① 存储卡连接（保护盖下面）

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。以下显示 CPU 的不同型号。有关特定 CPU 的详细信息，请参见技术规范 (页 724)。

表格 1- 1　S7-200 SMART CPU

	CR40	CR60	SR20	ST20	SR30	ST30	SR40	ST40	SR60	ST60
紧凑型，不可扩展	X	X								
标准，可扩展			X	X	X	X	X	X	X	X
继电器输出	X	X	X		X		X		X	
晶体管输出 (DC)				X		X		X		X
I/O 点（内置）	40	60	20	20	30	30	40	40	60	60

表格 1- 2　紧凑型不可扩展 CPU

特性		CPU CR40	CPU CR60
尺寸：W x H x D (mm)		125 x 100 x 81	175 x 100 x 81
用户存储器	程序	12 KB	12 KB
	用户数据	8 KB	8 KB
	保持性	最大 10 KB [1]	最大 10 KB [1]
板载数字量 I/O	• 输入 • 输出	• 24 DI • 16 DQ 继电器	36 DI 24 DQ 继电器
扩展模块		无	无
信号板		无	无
高速计数器		100 KHz 时 4 个，针对单相 或 50 KHz 时 2 个，针对 A/B 相	100 KHz 时 4 个，针对单相 或 50 KHz 时 2 个，针对 A/B 相
PID 回路		8	8
实时时钟，备用时间 7 天		无	无

[1] 可组态 V 存储器、M 存储器、C 存储器的存储区（当前值），以及 T 存储器要保持的部分（保持性定时器上的当前值），最大可为最大指定量。

表格 1- 3　标准型可扩展 CPU

特性		**CPU SR20、CPU ST20**	**CPU SR30、CPU ST30**	**CPU SR40、CPU ST40**	**CPU SR60、CPU ST60**
尺寸：W x H x D (mm)		90 x 100 x 81	110 x 100 x 81	125 x 100 x 81	175 x 100 x 81
用户存储器	程序	12 KB	18 KB	24 KB	30 KB
	用户数据	8 KB	12 KB	16 KB	20 KB
	保持性	最大 10 KB [1]	最大 10 KB [1]	最大 10 KB [1]	最大 10 KB [1]
板载数字量 I/O	• 输入 • 输出	• 12 DI • 8 DQ	• 18 DI • 12 DQ	• 24 DI • 16 DQ	• 36 DI • 24 DQ
扩展模块		最多 6 个	最多 6 个	最多 6 个	最多 6 个
信号板		1	1	1	1
高速计数器		200 KHz 时 4 个，针对单相 或 100 KHz 时 2 个，针对 A/B 相	200 KHz 时 4 个，针对单相 或 100 KHz 时 2 个，针对 A/B 相	200 KHz 时 4 个，针对单相 或 100 KHz 时 2 个，针对 A/B 相	200 KHz 时 4 个，针对单相 或 100 KHz 时 2 个，针对 A/B 相
脉冲输出 [2]		2 个，100 KHz	3 个，100 KHz	3 个，100 KHz	3 个，100 KHz
PID 回路		8	8	8	8
实时时钟，备用时间 7 天		有	有	有	有

[1] 可组态 V 存储器、M 存储器、C 存储器的存储区（当前值），以及 T 存储器要保持的部分（保持性定时器上的当前值），最大可为最大指定量。

[2] 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。

有关 CPU 和扩展模块的功率要求信息，请参见技术规范 (页 719)。使用附录 B 计算功率预算 (页 842)中的工作表计算功率预算。

1.2 S7-200 SMART 扩展模块

为更好的满足应用需求，S7-200 SMART 系列包括诸多扩展模块、信号板和通信模块。可将这些扩展模块与标准 CPU 型号（SR20、ST20、SR30、ST30、SR40、ST40、SR60 或 ST60）搭配使用，为 CPU 增加附加功能。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 719)。

表格 1- 4　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	• 8 个直流输入 • 16 个直流输入	• 8 个直流输出 • 8 个继电器输出 • 16 个继电器输出 • 16 个晶体管输出	• 8 个直流输入/8 个直流输出 • 8 个直流输入/8 个继电器输出 • 16 个直流输入/16 个直流输出 • 16 个直流输入/16 个继电器输出	
模拟量扩展模块	• 4 个模拟量输入 • 8 个模拟量输入 • 2 个 RTD 输入 • 4 个 RTD 输入 • 4 个热电偶输入	• 2 个模拟量输出 • 4 个模拟量输出	• 4 个模拟量输入/2 个模拟量输出 • 2 个模拟量输入/1 个模拟量输出	
信号板	• 1 个模拟量输入	• 1 个模拟量输出	• 2 个直流输入/2 个直流输出	• RS485/RS232 • 电池板

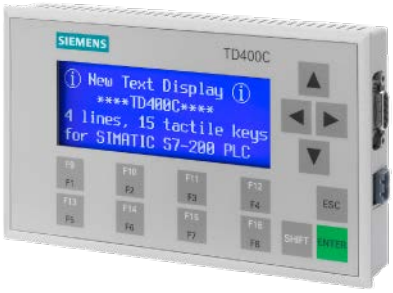
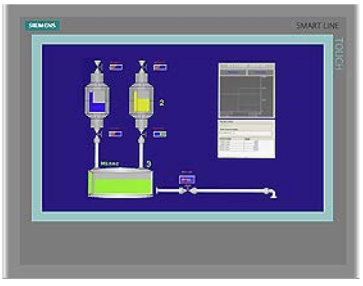
表格 1- 5　通信扩展模块

模块	类型	说明
通信扩展模块 (EM)	PROFIBUS DP SMART 模块	EM DP01 PROFIBUS DP

1.3 适用于 S7-200 SMART 的 HMI 设备

S7-200 SMART 支持 Comfort HMI、SMART HMI、Basic HMI 和 Micro HMI。以下显示 TD400C 和 SMART LINE
触摸面板。有关支持的设备和产品编号的完整列表，请参见附录 C“人机界面” (页 906)。

表格 1- 6　HMI 设备

	**文本显示单元：**TD400C 是一款显示设备，可以连接到 CPU。使用文本显示向导，可以轻松地对 CPU 进行编程，以显示文本信息和其它与您的应用有关的数据。 TD400C 设备可以作为应用的低成本接口，使用该设备可查看、监视和更改与应用有关的过程变量。
	**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。

STEP 7-Micro/WIN SMART 中的“文本显示”向导协助用户为 TD400C 快速且方便地组态文本显示消息。要启动“文本显示”向导，可从“工具”(Tools) 菜单选择“文本显示”(Text Display) 命令。

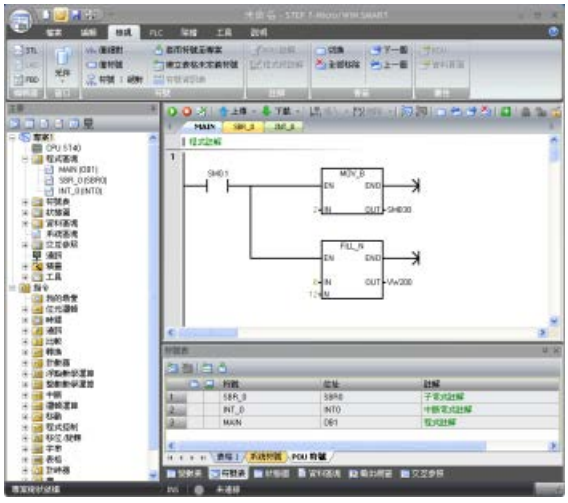
可从 Siemens 客户支持网站下载 SIMATIC 文本显示 (TD) 用户手册。

1.4 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 与其它 S7-200 SMART CPU 的对等通信
  - 与其它具有以太网功能的设备间的开放式用户通信 (OUC)
- PROFIBUS：
  - 适用于分布式 I/O 的高速通信（高达 12 Mbps）
  - 一个总线控制器连接许多 I/O 设备（支持 126 个可寻址设备）。
  - 主站和 I/O 设备间的数据交换
  - EM DP01 模块是 PROFIBUS I/O 设备。
- RS485：
  - 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）

1.5 编程软件



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

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 提供三种程序编辑器（LAD、FBD 和 STL），用于方便高效地开发适合用户应用的控制程序。

为帮助您找到所需信息，STEP7-Micro/WIN SMART 提供了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 XP SP3（仅 32 位）、Windows 7（支持 32 位和 64 位）
- 至少 350M 字节的空闲硬盘空间
- 鼠标（推荐）

安装 STEP 7-Micro/WIN SMART

将 STEP 7-Micro/WIN SMART CD 插入到计算机的 CD-ROM 驱动器中，或联系您的 Siemens 分销商或销售部门，从客户支持网站 (页 3)下载 STEP7-Micro/WIN SMART。安装程序将自动启动并引导您完成整个安装过程。 有关安装 STEP 7-Micro/WIN SMART 的详细信息，请参见自述文件。

说明

要在 Windows XP 或 Windows 7 操作系统上安装 STEP 7-Micro/WIN SMART，必须以管理员权限登录。

1.6 新功能

STEP 7-Micro/WIN SMART V2.2 和 S7-200 SMART V2.2 CPU 均引入了新功能，如下所示：

- 新模块：
  - EM 16 点数字输入 (EM DE16) (6ES7288-2DE16-0AA0) (页 770)
  - EM 16 点继电器型数字量输出 (EM QR16) (6ES7288-2QR16-0AA0) (页 773)
  - EM 16 点晶体管型数字量输出 (EM QT16) (6ES7288-2QT16-0AA0) (页 773)
  - 扩展电缆，1m (6ES7288-6EC01-0AA0) (页 837)
- 支持两个 Modbus RTU 主站 (页 492)
- 改进了添加自定义库 (页 592)的功能
- 开放式用户通信指令 (页 221)
- 开放式用户通信 (页 517)
- 提高了项目 (页 110)、POU 和数据块（数据页） (页 110)密码的安全性

入门指南

2

STEP 7-Micro/WIN SMART 可简化对 CPU 的编程。
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。

此示例需要以太网电缆、CPU 和运行 STEP 7-Micro/WIN SMART 编程软件的编程设备。

2.1 连接到 CPU

连接 CPU 十分容易。 在本例中，只需将电源连接到 CPU，然后用以太网通信电缆连接编程设备与 CPU。

将电源连接到 CPU

⚠警告

安装、接线或拆卸设备前，请确保电源关闭

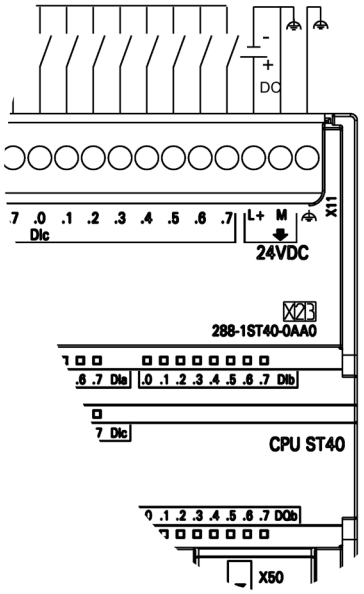
在安装或拆卸任何电气设备之前，请确保已切断该设备的电源。

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。

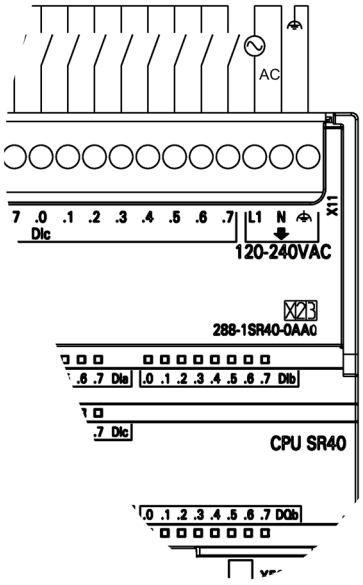
在安装或拆卸 CPU 或相关设备之前，必须采取合适的安全预防措施并确保切断该 CPU 的电源。

将 CPU 连接至电源。下图显示了直流和交流型 CPU 的接线。

直流安装

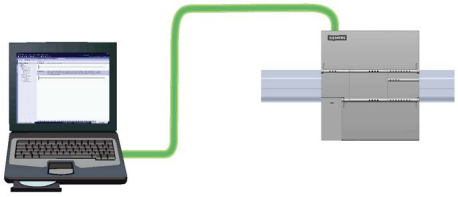


交流安装



2.1.1 组态 CPU 以进行通信

2.1.1.1 概述



CPU 可以与以太网上的 STEP 7-Micro/WIN SMART 编程设备进行通信。

在 CPU 和编程设备之间建立通信时请考虑以下几点：

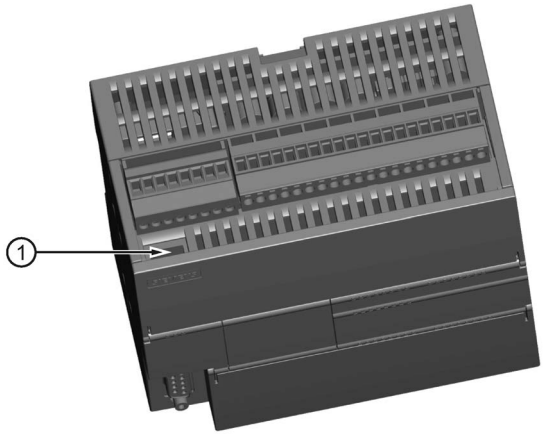
- 组态/设置： 单个 CPU 不需要硬件配置。 如果想要在同一个网络中安装多个 CPU，则必须将默认 IP 地址更改为新的唯一的 IP 地址。
- 一对一通信不需要以太网交换机；网络中有两个以上的设备时需要以太网交换机。

2.1.1.2 建立硬件通信连接

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 RJ45 连接盖从以太网端口卸下。收好盖以备再次使用。
3. 将以太网电缆插入 CPU 顶部的以太网端口中，如下所示。
4. 将以太网电缆连接到编程设备上。



① 以太网端口

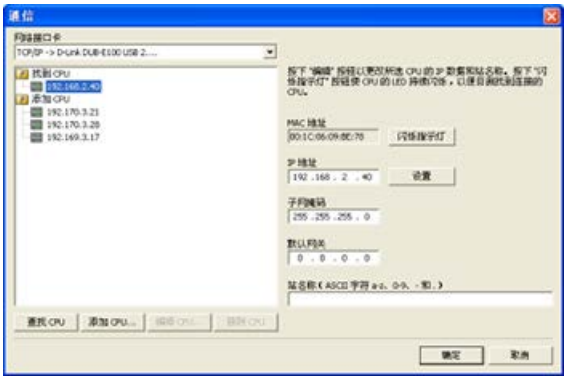
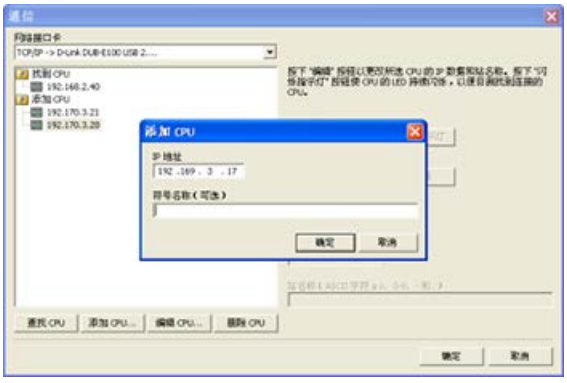
2.1.1.3 与 CPU 建立通信

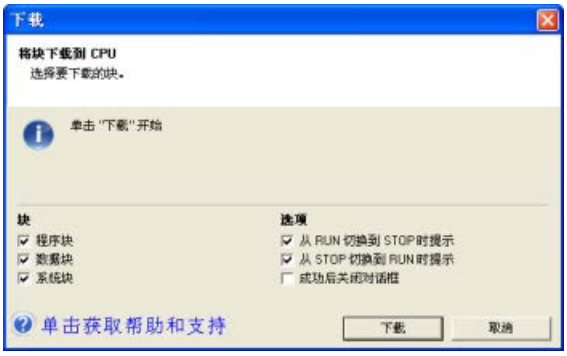
在 STEP 7-Micro/WIN SMART 中，使用以下方法之一显示“通信”(Communications) 对话框，组态与 CPU 的通信。

- 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来选择所要访问的 CPU：

- 单击“查找 CPU”(Find CPU) 按钮以使 STEP 7-Micro/WIN SMART 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。

	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： • 选择网络接口卡的 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（“已发现 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），可通过“通信对话框”(Communications dialog) 与您的 CPU 建立连接： • 选择网络接口卡的 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，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。

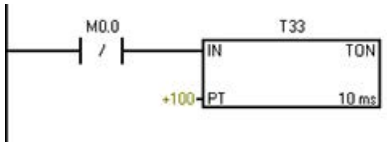
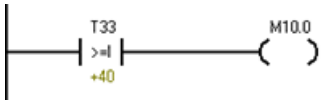
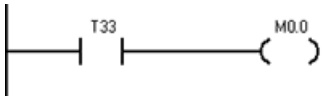
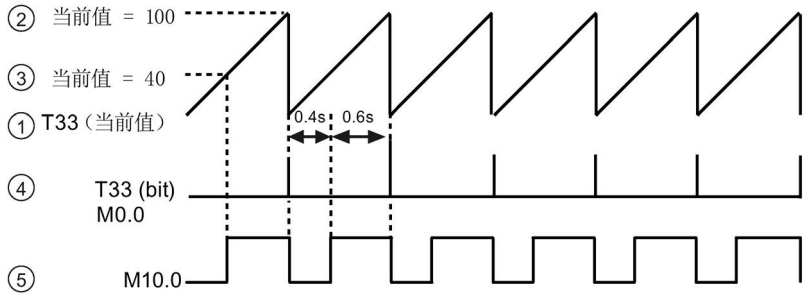
有关如何完成此任务的信息，请参见“为项目中的 CPU 或设备组态或更改 IP 地址”。

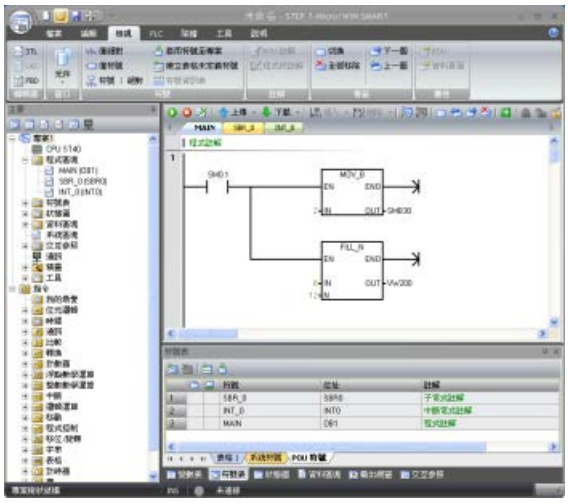
2.2 创建示例程序

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 2- 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。
	**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



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。

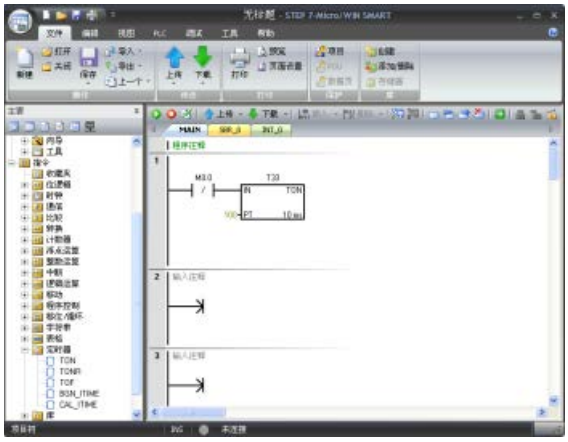
程序中的所有块均保存在项目树的程序块文件夹中。

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。

输入并保存程序之后，可以将程序下载到 CPU。

2.2.1 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：

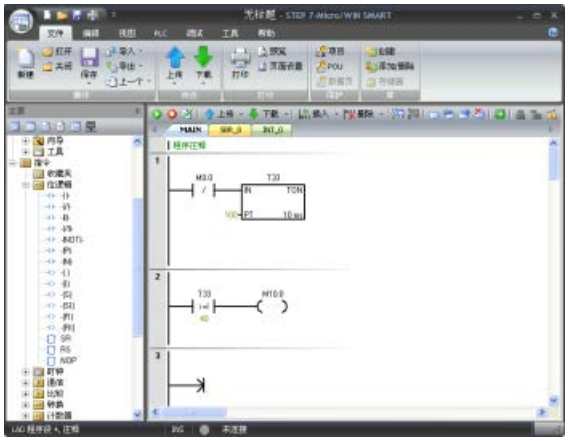
1. 双击“位逻辑”(Bit Logic) 图标或单击加号 (+) 以显示位逻辑指令。
2. 选择“常闭”触点。
3. 按住鼠标左键并将触点拖到第一个程序段中。
4. 为触点输入以下地址： M0.0
5. 按回车键即输入该触点地址。

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。

2.2.2 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：

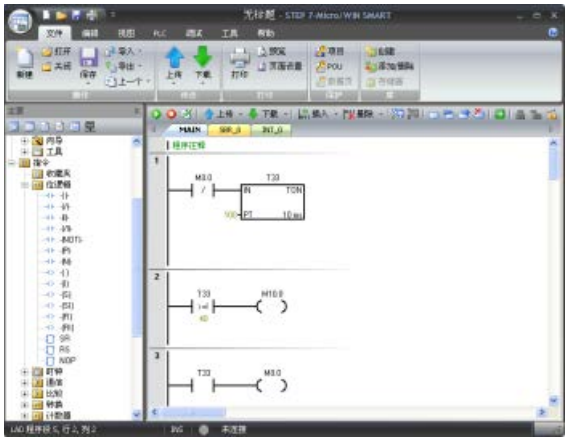
1. 双击“比较”(Compare) 图标以显示比较指令。 选择“**>=I**”指令（大于或等于整数）。
2. 按住鼠标左键并将比较指令拖到第二个程序段中。
3. 单击触点上方的“???”，然后输入以下定时器地址值： T33
4. 按回车键即输入定时器编号，光标将移动到将与定时器值进行比较的其它值。
5. 输入要与定时器数值比较的以下值： +40
6. 按回车键即输入该值。

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。

2.2.3 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。

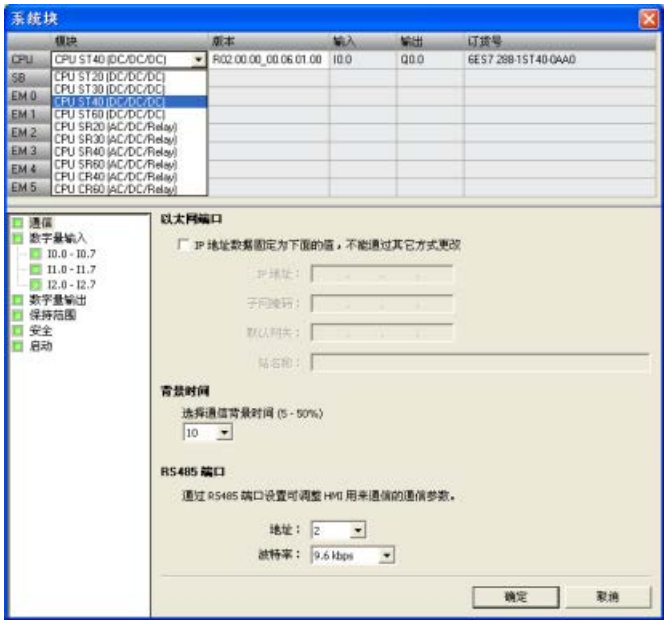
要输入用于接通 M0.0 的线圈：

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。

2.2.4 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。



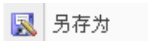
2.2.5 保存示例项目

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。

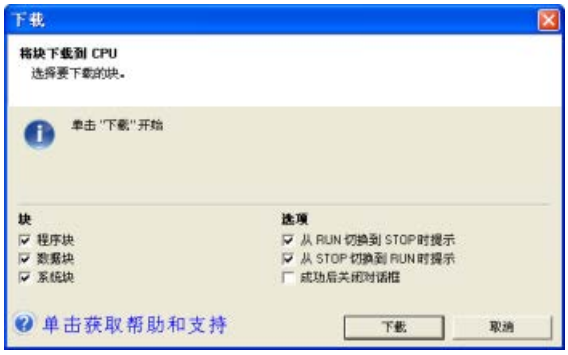


3. 在“另存为”(Save As) 对话框中输入项目名称。
4. 浏览到想要保存项目的位置。
5. 点击“保存”(Save) 以保存项目。

保存项目后，可下载程序到 CPU。

2.3 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 将完整程序或您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 类型不匹配，STEP 7-Micro/WIN SMART 将指示不匹配并提示您采取措施。

2.4 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 不执行任何程序，而用户可以下载程序块。在 RUN 模式下，CPU 会执行相关程序；但用户仍可下载程序块。

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. 提示时，单击“确定”(OK) 更改 CPU 的工作模式。

可监视 STEP 7-Micro/WIN SMART 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 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，以将 CPU 置于 STOP 模式。

安装 3

3.1 S7-200 SMART 设备安装准则

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

⚠警告

S7-200 SMART 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。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。

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。

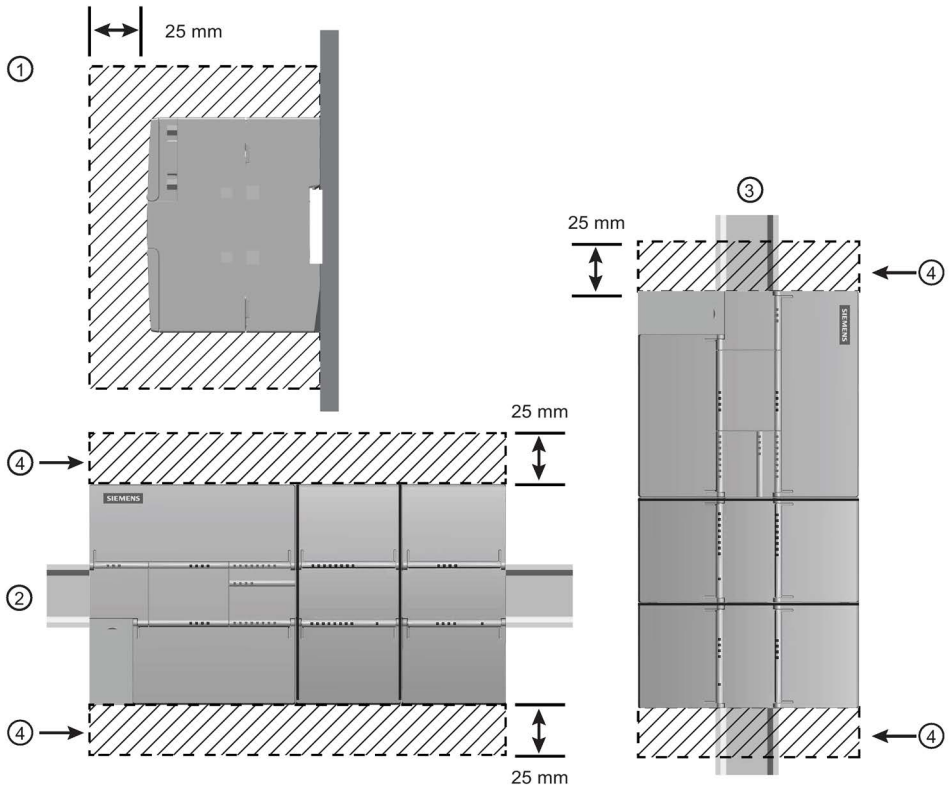
留出足够的间隙以便冷却和接线

S7-200 SMART 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。
将 CPU 安装在所有扩展模块的下方，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。

3.2 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 的技术规范确定 24 V DC 传感器供电预算、CPU 所提供的 5 V DC 逻辑预算以及扩展模块和信号板的 5 V DC 电源要求。请参考计算功率预算 (页 839)，确定 CPU 能为您的组态提供多少电能（或电流）。

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。要格外注意系统组态以确保 CPU 可以提供所选扩展模块所需的 5 V DC 电源。如果组态要求的电源超出 CPU 提供的电源范围，则必须拆下一些模块。

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 也提供 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 24 V DC 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。

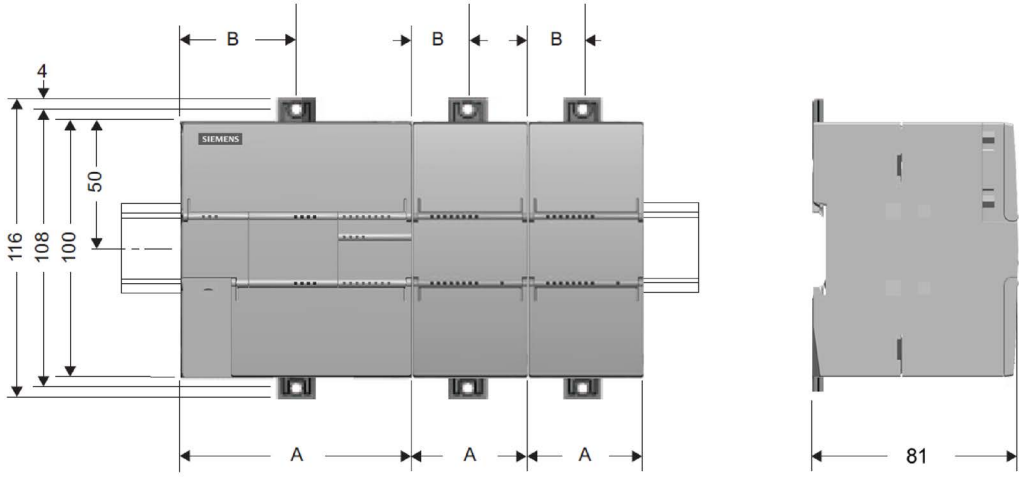
务必确保 S7-200 SMART 系统中的所有非隔离 M 端子都连接到同一个参考电位。

请参见具体 CPU 的技术规范确定 24 V DC 传感器供电预算、CPU 所提供的 5 V DC 逻辑预算以及扩展模块和信号板的 5 V DC 电源要求。

3.3 安装和拆卸步骤

3.3.1 S7-200 SMART 设备的安装尺寸

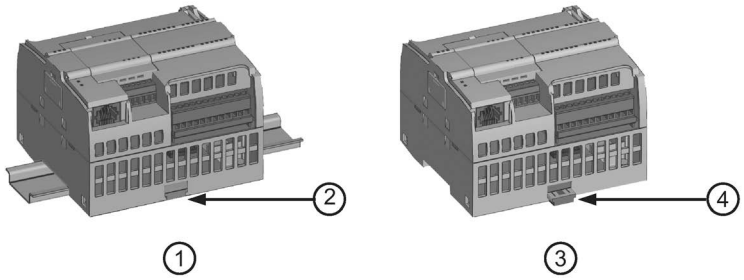
CPU 和扩展模块都有安装孔，可以很方便地安装到面板上。



S7-200 SMART 模块		宽度 A (mm)	宽度 B (mm)
CPU SR20 和 CPU ST20		90	45
CPU SR30 和 CPU ST30		110	55
CPU CR40、CPU SR40 和 CPU ST40		125	62.5
CPU CR60、CPU SR60 和 CPU ST60		175	87.5
扩展模块：	EM 4AI、EM 8AI、EM 2AQ、EM 4AQ、EM 8DI、EM 16DI、EM 8DQ、和 EM 8DQRLY、EM 16 DQRLY、和 EM 16DQ 晶体管	45	22.5
	EM 8DI/8DQ 和 EM 8DI/8DQRLY	45	22.5
	EM 16DI/16DQ 和 EM 16DI/16DQRLY	70	35
	EM 2AI/1AQ 和 EM 4AI/2AQ	45	22.5
	EM 2RTD、EM 4RTD	45	22.5
	EM 4TC	45	22.5
	EM DP01	70	35

3.3.2 安装和拆卸 CPU

CPU 可以很方便地安装到标准 DIN 导轨或面板上。 可使用 DIN 导轨卡夹将设备固定到 DIN 导轨上。
这些卡夹还能掰到一个伸出位置以提供用于对设备进行面板安装的螺钉安装位置。



① DIN 导轨安装
② DIN 导轨卡夹处于锁紧位置
③ 面板安装
④ 卡夹处于伸出位置

图 3-1 在 DIN 导轨或面板上安装

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。

说明

在安装 CPU 之后单独安装扩展模块。

将该单元安装到 DIN 导轨或面板上时，应考虑以下几点：

- 对于 DIN 导轨安装，确保 CPU 的上部 DIN 导轨卡夹处于锁紧（内部）位置而下部 DIN 导轨卡夹处于伸出位置。
- 将设备安装到 DIN 导轨上后，将下部 DIN 导轨卡夹推到锁紧位置以将设备锁定在 DIN 导轨上。
- 对于面板安装，确保将 DIN 导轨卡夹推到伸出位置。

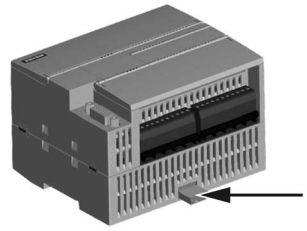
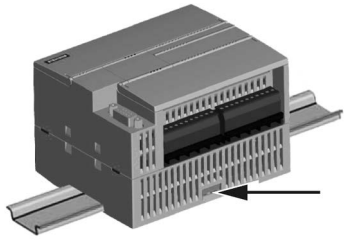
要在面板上安装 CPU，请按以下步骤操作：

1. 按照表安装尺寸 (mm) (页 49) 中的尺寸定位、钻孔并对安装孔攻螺纹（M4 或美国标准 8 号）。
2. 确保 CPU 和 S7-200 SMART 设备与电源断开连接。
3. 使用带弹簧和平垫圈的 Pan Head M4 螺钉将模块固定到面板上。
   不要使用平头螺钉。
4. 如果在使用扩展模块，则将其放在 CPU 旁，并一起滑动，直至连接器牢固连接。

说明

螺钉类型将由安装时的材料决定。 应施加适当的扭矩，直到弹簧垫圈变平。
避免对安装螺钉施加过多扭矩。 不要使用平头螺钉。

表格 3- 1　在 DIN 导轨上安装 CPU

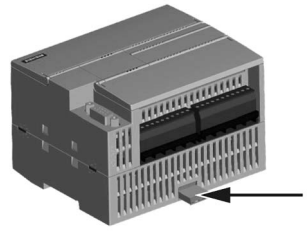
任务	步骤
	按照下面的步骤在 DIN 导轨上安装 CPU。 1. 每隔 75 mm 将导轨固定到安装板上。 2. 咔嚓一声打开模块底部的 DIN 夹片，并将模块背面卡在 DIN 导轨上。 3. 将模块向下旋转至 DIN 导轨，咔嚓一声闭合 DIN 夹片。 仔细检查夹片是否将模块牢牢地固定到导轨上。 为避免损坏模块，请按安装孔标记，而不要直接按模块前侧。
	

说明

当 CPU 的使用环境振动比较大或垂直安装时，使用 DIN 导轨挡块可能会有帮助。 在 DIN 导轨上使用端盖（8WA1 808 或 8WA1 805）以确保模块保持连接状态。

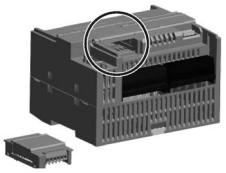
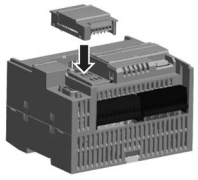
如果系统处于剧烈振动环境中，面板安装可给 CPU 提供较高的振动保护等级。

表格 3- 2　从 DIN 导轨上拆卸 CPU

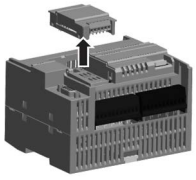
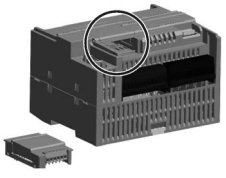
任务	步骤
	按照下面的步骤从 DIN 导轨上拆卸 CPU。 1. 切断 CPU 和连接的所有 I/O 模块的电源。 2. 断开连接到 CPU 的所有线缆。 CPU 和多数扩展模块都有可拆卸连接器，这使得该工作变得更加简单。 3. 拧下安装螺钉或咔嚓一声打开 DIN 夹片。 4. 如果连接了扩展模块，则向左滑动 CPU，将其从扩展模块连接器脱离。 注：拧下或解开扩展模块的 DIN 夹片可使分离 CPU 更容易。 5. 卸下 CPU。

3.3.3 安装和拆卸信号板或电池板

表格 3- 3 安装信号板

任务	步骤
	请按以下步骤安装信号板或电池板 1. 确保 CPU 和所有 S7-200 SMART 设备已与电源断开连接。 2. 卸下 CPU 上部和下部的端子块盖板。 3. 将螺丝刀插入 CPU 上部接线盒盖背面的槽中。 4. 轻轻将盖撬起并从 CPU 上卸下。 5. 将信号板或电池板直接向下放入 CPU 上部的安装位置中。 6. 用力将模块压入该位置直到卡入就位。 7. 重新装上端子块盖板。
	

表格 3- 4 拆卸信号板或电池板

任务	步骤
	请按以下步骤拆卸信号板或电池板 1. 确保 CPU 和所有 S7-200 SMART 设备与电源断开连接。 2. 卸下 CPU 上部和下部的端子块盖板。 3. 将螺丝刀插入模块上部的槽中。 4. 轻轻将模块撬起使其与 CPU 分离。 5. 将模块直接从 CPU 上部的安装位置中取出。 6. 将盖板重新装到 CPU 上。 7. 重新装上端子块盖板。
	

安装或更换 SB BA01 电池板中的电池

SB BA01 电池板所要求的电池型号为 CR1025。电池未随 SB BA01 一起提供，必须另行购买。

要安装电池，请按以下步骤操作：

1. 在 SB BA01 中，新电池的安装要求电池正极朝上，负极靠近印刷线路板。
2. 现在，已准备好将 SB BA01 安装到 CPU 中。请按照上述安装指示操作。

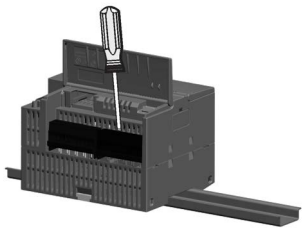
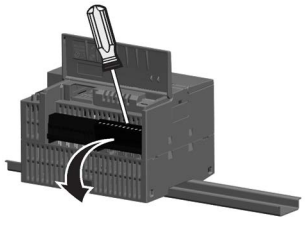
要更换电池，请按以下步骤操作：

1. 按照上述拆卸指示从 CPU 中取出 SB BA01。
2. 使用小号螺丝刀小心地取下旧电池。将电池从卡夹下部推出。
3. 安装新的 CR1025 替换电池时，要求电池正极朝上，负极靠近印刷线路板。
4. 按照上述安装指示重新安装 SB BA01 电池板。

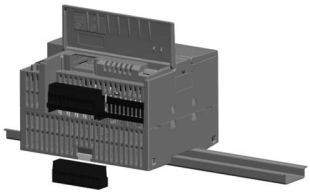
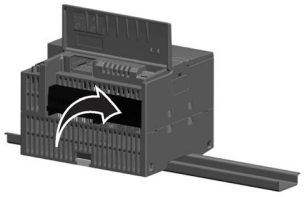
3.3.4 拆卸和重新安装端子块连接器

S7-200 SMART 模块具有可拆卸连接器，这简化了接线的连接。

表格 3- 5 拆卸连接器

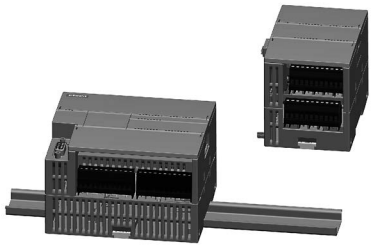
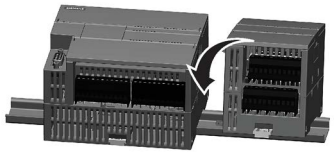
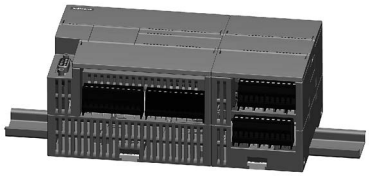
任务	步骤
	通过卸下 CPU 的电源并打开连接器上的盖子，准备从系统中拆卸端子块连接器。 1. 确保 CPU 和所有 S7-200 SMART 设备与电源断开连接。 2. 查看连接器的顶部并找到可插入螺丝刀头的槽。 3. 将小螺丝刀插入槽中。 4. 轻轻撬起连接器顶部使其与 CPU 分离。 连接器从夹紧位置脱离。 5. 抓住连接器并将其从 CPU 上卸下。
	

表格 3- 6　安装连接器

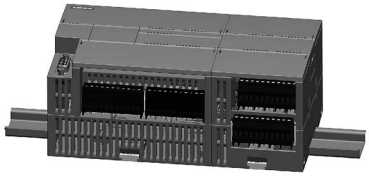
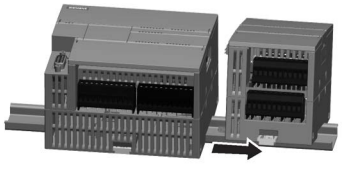
任务	步骤
	断开 CPU 电源并打开连接器上的盖子，准备安装接线盒组件。 1. 确保 CPU 和所有 S7-200 SMART 设备与电源断开连接。 2. 使连接器与单元上的插针对齐。 3. 将连接器的接线边对准连接器座沿的内侧。 4. 用力按下并转动连接器直到卡入到位。 仔细检查并确保连接器已正确对齐并且完全啮合。
	

3.3.5 安装和拆卸扩展模块

表格 3- 7 安装扩展模块

任务	步骤
	按照下面的步骤安装扩展模块： 1. 确保 CPU 和所有 S7-200 SMART 设备与电源断开连接。 2. 卸下 CPU 右侧的 I/O 总线连接器盖。 3. 将螺丝刀插入盖上方的插槽中。 4. 将其上方的盖轻轻撬出并卸下盖。保留该盖待重复使用。
	将扩展模块连接至 CPU。 1. 拉出下方的 DIN 导轨卡夹以便将扩展模块安装到导轨上。 2. 将扩展模块放置在 CPU 右侧。 3. 将扩展模块挂到 DIN 导轨上方。 4. 向左滑动扩展模块，直至 I/O 连接器与 CPU 右侧的连接器完全啮合，并推入下方的卡夹将扩展模块锁定到导轨上。
	

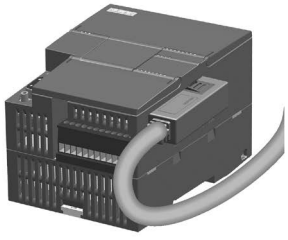
表格 3- 8 拆卸扩展模块

任务	步骤
	按照下面的步骤拆卸扩展模块： 1. 确保 CPU 和所有 S7-200 SMART 设备与电源断开连接。 2. 将 I/O 连接器和接线从扩展模块上卸下。拧松所有 S7-200 SMART 设备的 DIN 导轨卡夹。 3. 向右滑动扩展模块。
	

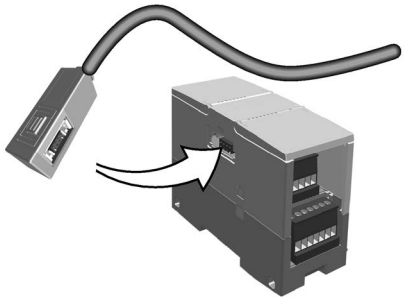
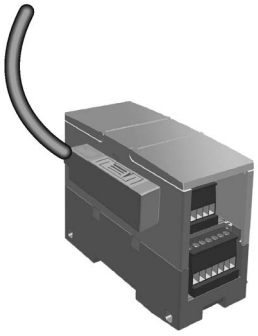
3.3.6 安装和卸下扩展电缆

S7-200 扩展电缆可用来更灵活地对 S7-200 系统的布局进行组态。每个 CPU 系统只允许使用一条扩展电缆。可以将扩展电缆安装在 CPU 和第一个 EM 之间，或者安装在任意两个 EM 之间。

表格 3- 9　安装和卸下扩展电缆的公连接器

任务	步骤
	要安装公连接器： 1. 确保 CPU 和所有 S7-200 SMART 设备与电源断开连接。 2. 将公连接器按压到扩展模块或 CPU 右侧的总线连接器中。 3. 公连接器完全插入槽中时卡入就位。 要卸下公连接器： 1. 确保 CPU 和所有 S7-200 SMART 设备与电源断开连接。 2. 用拇指按下公连接器上部的闩锁，使其从扩展模块或 CPU 中释放出来。 3. 直接拔出即可将公连接器从扩展模块或 CPU 上卸下。
	

表格 3- 10　安装和卸下扩展电缆的母连接器

任务	步骤
	要安装母连接器： 1. 确保 CPU 和所有 S7-200 SMART 设备与电源断开连接。 2. 将母连接器按压到扩展模块左侧的总线连接器中。 3. 母连接器完全插入槽中时卡入就位。 要卸下母连接器： 1. 确保 CPU 和所有 S7-200 SMART 设备与电源断开连接。 2. 用拇指按下母连接器上部的闩锁，使其从扩展模块中释放出来。 3. 直接拔出即可将母连接器从扩展模块上卸下。
	

说明

在振动环境中安装扩展电缆

如果将扩展电缆连接在移动或固定不牢的模块上，电缆头连接处可能会慢慢松动。

为了提供额外的应力消除作用，应使用电缆扎带将电缆头固定在 DIN 导轨（或其它位置）上。

安装期间拉拽电缆时应避免用力过猛。安装完成后，确保电缆与模块连接到位。

3.4 接线准则

对所有电气设备进行正确地接地和接线非常重要，这有助于确保系统最佳地运行以及为应用和 PLC 提供额外的电气噪声保护。 有关接线图的信息，请参见技术规范 (页 719)。

先决条件

在对任何电气设备进行接地或接线之前，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。

确保在对 PLC 和相关设备接线时遵守所有适用的电气准则。
根据所有适用的国家和地方标准来安装和操作所有设备。
联系当地管理部门确定哪些准则和标准适用于您的具体情况。

⚠警告

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 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 PLC 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。

规划 PLC 系统的接地和接线时，务必考虑安全问题。 电子控制设备（如
PLC）可能会失灵和导致正在控制或监视的设备出现意外操作。
因此，应采取一些独立于 PLC 的安全措施以防止发生可能的人员受伤或设备损坏。

⚠警告

控制设备在不安全情况下运行时可能会出现故障，从而导致受控设备的意外运行。
这种意外运行可能会导致人员死亡、重害和/或设备损坏。
应使用紧急停止功能、机电超控功能或其它独立于 PLC 的冗余安全功能。

绝缘准则

交流电源和 I/O
与交流电路的边界经过设计，经验证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。仅对达到交流线路电压的电路提供了与其它电路的安全隔离。24 V
直流电路间的隔离边界仅适于工作状态，不可用于安全目的。

根据 EN 61131-2，集成有交流电源的 S7-200
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。

要维持 S7-200 SMART 低压电路的安全特性，到通信端口、模拟电路以及所有 24 V
直流标称电源和 I/O 电路的外部连接必须由合格的电源供电，该电源必须满足各种标准对 SELV、PELV、2 类、限制电压或受限电源的要求。

⚠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。

接地准则

将应用设备接地的最佳方式是确保 PLC
和相关设备的所有公共端和接地连接在同一个点接地。该单点应当直接与系统的接地相连。

所有接地线必须尽可能地短且应使用大线径，例如，2 mm^2 (14 AWG)。

确定接地点时，请记住要考虑安全接地要求和保护性中断设备的正确工作。

接线准则

设计 S7-200 SMART CPU 的接线时，应提供一个可同时切断 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它限流装置以提供额外保护。

为所有可能遭受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。务必成对布线，将中性线或公共导线与带电导线或载有信号的导线成对布设。

使用尽可能短的电线并确保线径可承载要求的电流。

导线和电缆的额定温度应比 S7-200 SMART CPU 周边的环境温度高 30°C。（例如，环境温度为 55°C 时，导体的最小额定温度为 85°C。）对于其它接线类型和材料的要求，您需要根据具体的电路等级和安装环境来确定。

使用屏蔽线以最好地防止电气噪声。通常，在 S7-200 SMART CPU 端将屏蔽层接地可获得最佳效果。您应该将连接到 S7-200 SMART CPU 通信连接器外壳的通信电缆屏蔽层接地，可使用与电缆屏蔽层咬合的连接器接地，或是将通信电缆的屏蔽层单独接地。您应该使用夹板或环绕屏蔽层的铜条将其它电缆屏蔽层接地，这样可增大连接接地点的表面积。

在对由外部电源供电的输入电路进行接线时，应在该电路中安装过流保护装置。由 S7-200 SMART CPU 的 24 V 直流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 CPU 模块都有供用户接线的可拆卸连接器。要防止连接松动，请确保连接器固定牢靠且导线已牢固地安装到连接器中。

为了帮助避免安装设备中出现意外的电流，S7-200 SMART CPU 在某些点提供绝缘边界。在规划系统的接线时，应考虑这些绝缘边界。有关所提供的绝缘程度和绝缘边界位置的信息，请参见技术规范 (页 719)。对达到交流线路电压的电路提供了与其它电路的安全隔离。24 V 直流电路间的隔离边界仅适于工作状态，不可用于安全目的。

S7-200 SMART CPU、EM 和 SB 的接线规则总结如下：

表格 3- 11　S7-200 SMART CPU、EM 和 SB 的接线规则：

适用的接线规则...	CPU 和 EM 连接器	SB 连接器
标准导线可连接导体的截面积	2 mm² 到 0.3 mm²（14 AWG 到 22 AWG）	1.3 mm² 到 0.3 mm²（16 AWG 到 22 AWG）
每个连接的导线数	1 根导线或 2 根导线的组合，横截面积最多为 2 mm²（合计）	1 根导线或 2 根导线的组合，横截面积最多为 1.3 mm²（合计）
导线裸线长度	6.4 mm	6.3 至 7 mm
紧固扭矩*（最大）	0.56 N-m（5 英寸-磅）	0.33 N-m（3 英寸-磅）
工具	2.5 至 3.0 mm 一字螺丝刀	2.0 至 2.5 mm 一字螺丝刀

* 为避免损坏连接器，小心不要将螺丝紧固过紧。

说明

绞线上的保护套管或终端套管可降低杂散线股带来的短路风险。如果保护套管的长度大于推荐的裸线长度，则应加一个绝缘环，以防导体侧向移动造成短路。裸露导体的横截面积限定也适用于保护套管。

参见

常规技术规范 (页 719)

灯负载的使用准则

于接通浪涌电流大，灯负载会导致继电器触点损坏。 该浪涌电流通常是钨灯稳态电流的 10 到 15 倍。

对于在应用期间将进行大量开关操作的灯负载，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。抑制能力较差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
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 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。

一种良好的抑制解决方案是使用接触器或其它感性负载，制造商为这些感性负载提供了集成在负载设备中的抑制电路，或提供抑制电路作为可选附件。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。

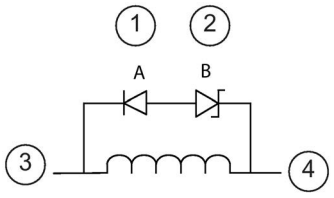
对于交流负载，可将金属氧化物变阻器 (MOV) 或其它电压钳制设备与并联 RC
电路配合使用，但不如单独使用有效。不带并联 RC 电路的 MOV
抑制器通常会导致出现高达钳位电压的显著高频噪声。

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

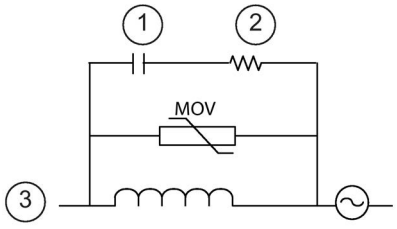
② 8.2 V
稳压二极管（直流输出），
36 V
稳压二极管（继电器输出）

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 3- 12　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性负载时，请务必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。

PLC 概念

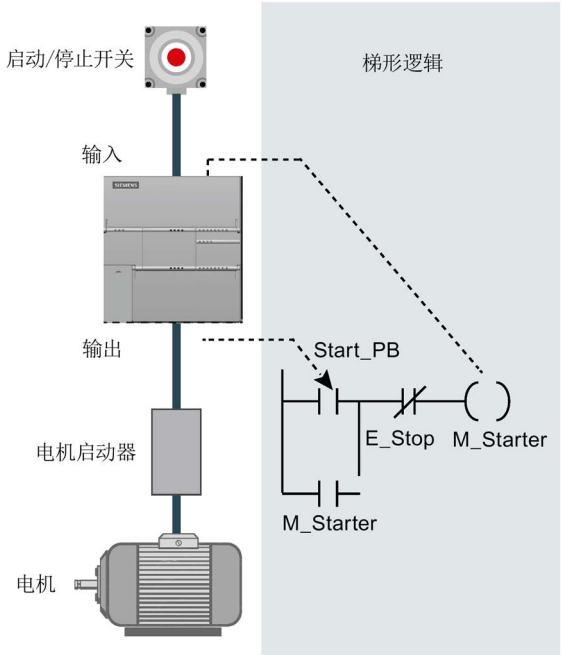
4

CPU 的基本功能是监视现场输入，并根据控制逻辑接通或断开现场输出设备。
本章介绍了有关程序执行、使用的各种存储器和存储器如何保持等方面的一些概念。

4.1 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。

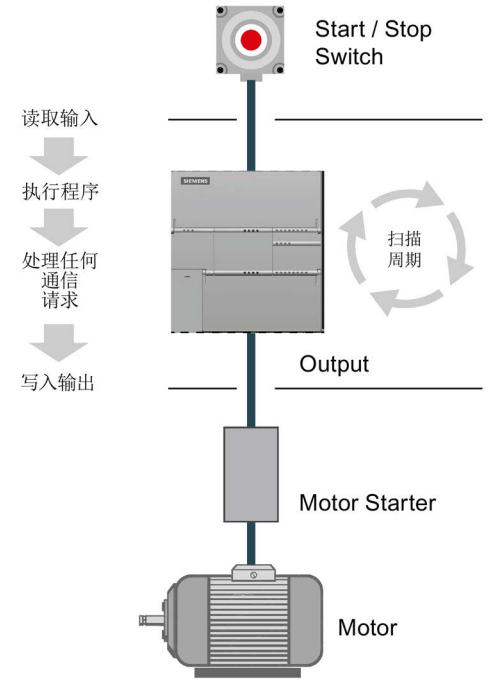


此图显示了电气继电器图与 CPU 关系的简图。在本例中，用于启动电机的开关的状态与其它输入的状态相结合。这些状态的计算结果决定用于控制电机启动执行器的输出的状态。

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 4- 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。

4.1.1 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。

4.1.2 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。

4.1.3 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、另一个子例程或中断例程调用子例程时，执行子例程。
从主程序调用时子例程的嵌套深度是 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、八个子例程嵌套级别（从主程序启动时）、一个中断例程和四个子例程嵌套级别（从中断程序启动时）。
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。

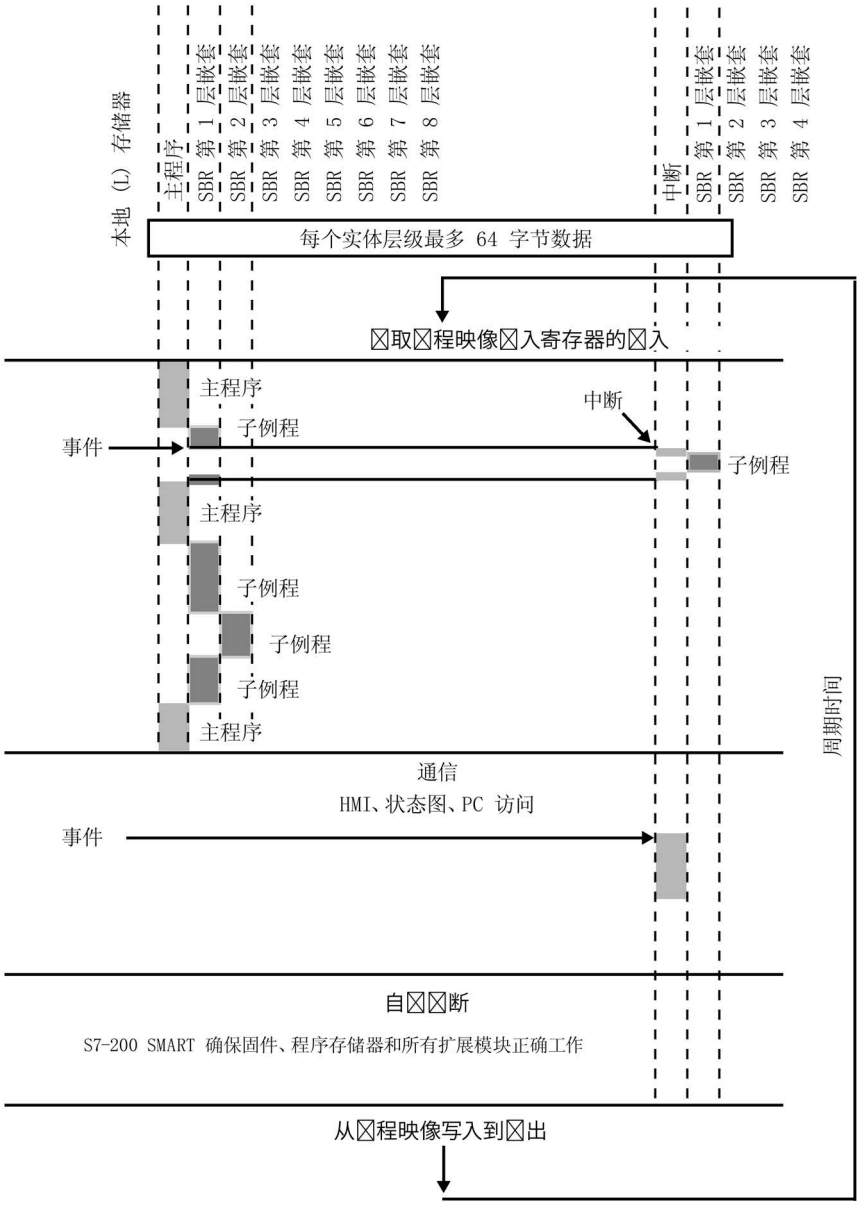
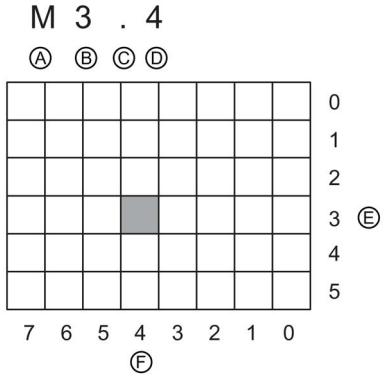


图 4-1　　典型扫描流程

4.2 访问数据

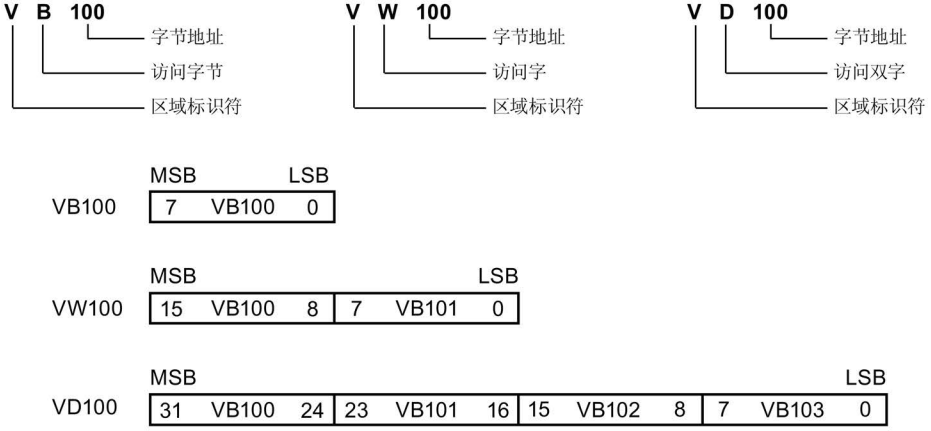
CPU 将信息存储在不同存储单元，每个位置均具有唯一的地址。
可以显式标识要访问的存储器地址。 这样程序将直接访问该信息。
要访问存储区中的位，必须指定地址，该地址包括存储器标识符、字节地址和位号（也称为“字节.位”寻址）。

表格 4- 2　位寻址

位地址元素	说明	
	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。

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。
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。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 4- 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。

4.2.1　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 4- 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 4- 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

V（变量存储器）

可以使用 V 存储器存储程序执行程序中控制逻辑操作的中间结果。 也可以使用 V 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 V 存储器：

表格 4- 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

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 4- 7　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 4- 8　T 存储器的绝对寻址

定时器：	T*[定时器编号]*	T24

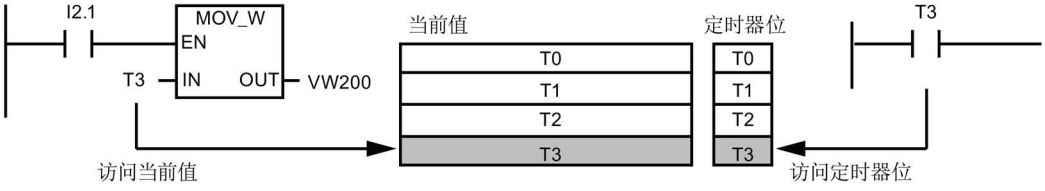


图 4-2　　访问定时器位或定时器的当前值

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 4- 9　C 存储器的绝对寻址

计数器	C*[计数器编号]*	C24

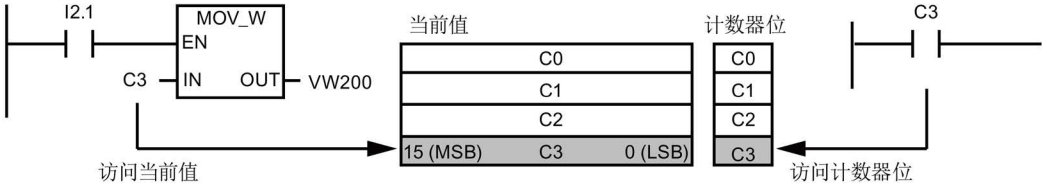


图 4-3　访问计数器位或计数器的当前值

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 4- 10　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）。
可以按位、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 335)。

表格 4- 11　AC 存储器的绝对寻址

累加器	AC*[累加器编号]*	AC0

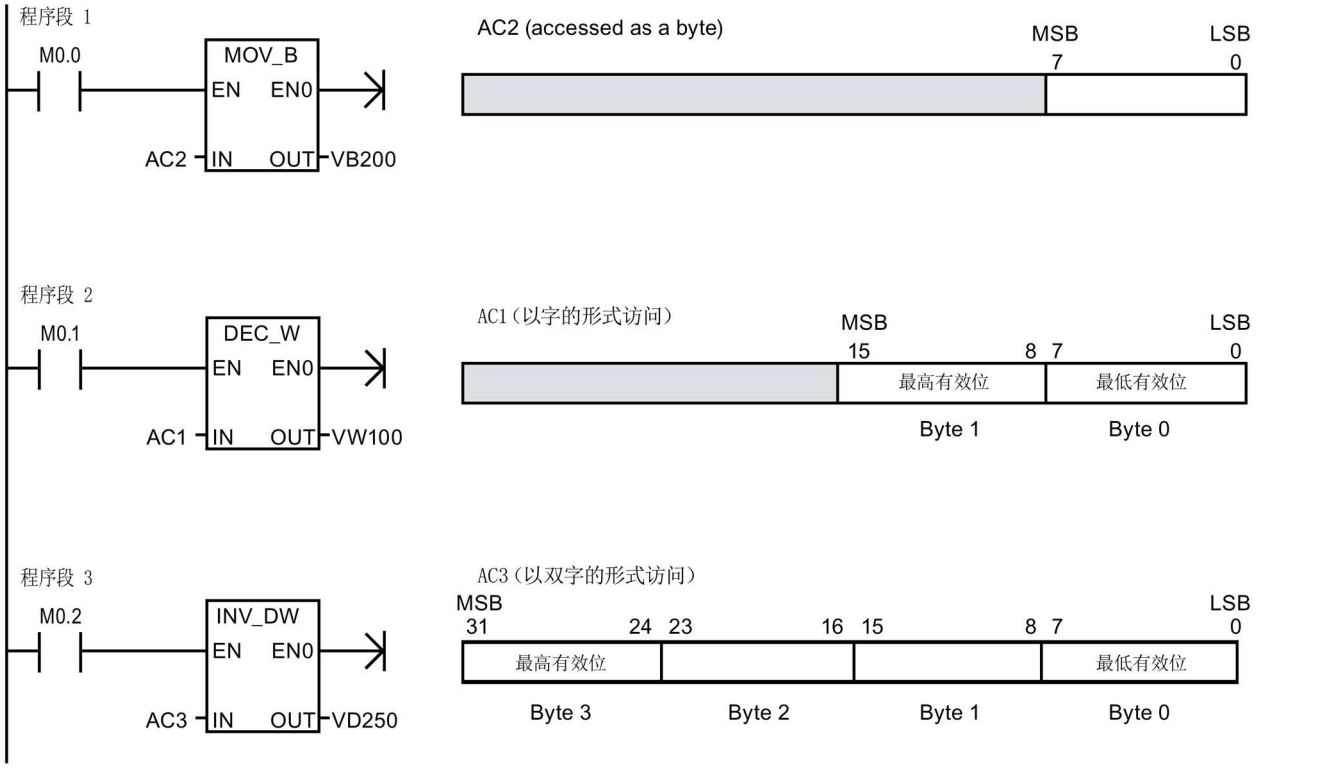


图 4-4　　访问累加器

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 4- 12　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 851)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 会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 4- 13　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 4- 14　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 4- 15　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 4- 16　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

4.2.2 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。

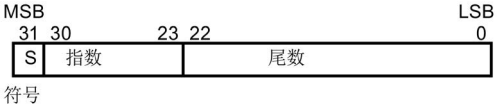


图 4-5 实数格式

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

4.2.3 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 4-6 字符串格式

4.2.4 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 4- 17 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。

4.2.5 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 4- 18 CPU 映射惯例

	CPU	信号板	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
起始地址	I0.0 Q0.0	I7.0 Q7.0 AI12 AQ12	I8.0 Q8.0 AI16 AQ16	I12.0 Q12.0 AI32 AQ32	I16.0 Q16.0 AI48 AQ48	I20.0 Q20.0 AI64 AQ64	I24.0 Q24.0 AI80 AQ80	I28.0 Q28.0 AI96 AQ96

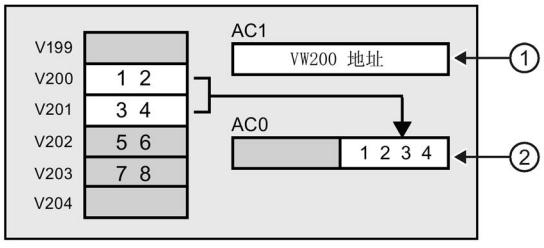
4.2.6 使用指针进行间接寻址

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）。
您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）。

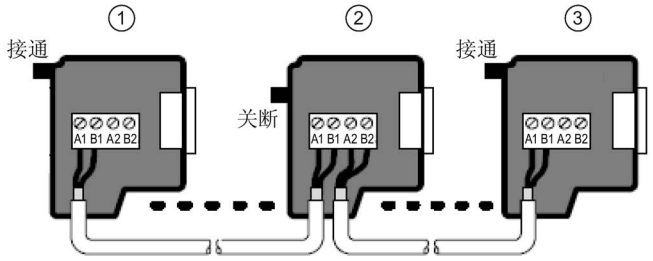
在指令操作数前面输入一个星号 (*) 可指定该操作数是一个指针。 如下图所示，输入 *AC1 表示 AC1 存储指向“移动字”(MOVW) 指令引用的字长度值的指针。
在该示例中，在 VB200 和 VB201 中存储的值被移至累加器 AC0。



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 4-7 创建和使用指针

如下图所示，您可以更改指针的值。 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 4-8 修改指针

说明

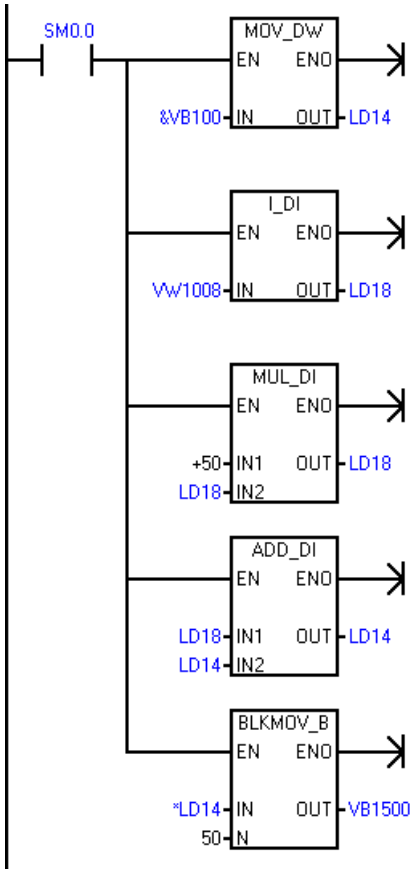
修改指针的值时，请记住调整所访问数据的大小： 访问字节时，指针值加 1；访问定时器或计数器的字或当前值时，指针值加 2；访问双字时，指针值加 4。

4.2.7 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 4- 19　示例： 使用指针访问表中数据

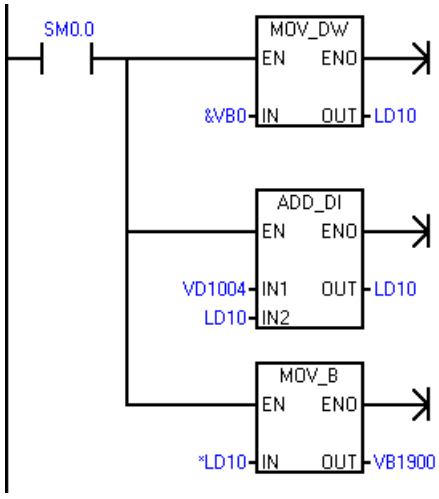
LAD		STL
	要传送配方表中的配方： • 每个配方的长度都是 50 字节。 • 索引参数 (VW1008) 标识要加载的配方。 创建指向配方表起始地址的指针。 将配方索引转换为双字值。 乘以偏移量，以容纳每个配方的大小。 将调整后的偏移量添加到指针。 将选定配方传送到 VB1500 至 VB1549	**Network 1** **LD SM0.0** **MOVD &VB100, LD14** **ITD VW1008, LD18** ***D +50, LD18** **+D LD18, LD14** **BMB *LD14, VB1500, 50**

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 4- 20　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

4.3 保存和恢复数据

4.3.1 下载项目组件

说明

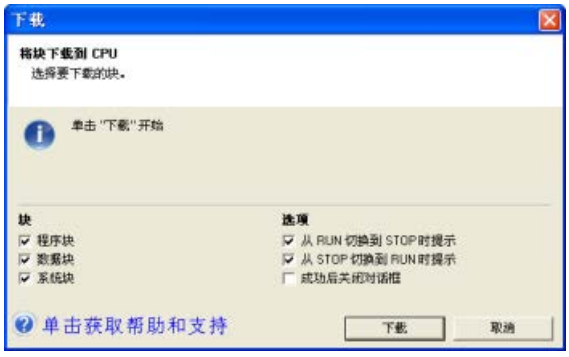
将程序块、数据块或系统块下载到 CPU 会彻底覆盖 CPU 中该块之前存在的任何内容。执行下载前，确定是要覆盖该块。

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

1. 确保网络硬件和 PLC 连接电缆运行正常 (页 31)，并确保 PLC 通信运行正常 (页 607)。
2. 将 CPU 置于 STOP 模式 (页 43)。
3. 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**。

4. 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要下载的特定项目组件（程序块、数据块或系统块）。
5. 单击“下载”(Download) 按钮后，如果弹出“通信”(Communications) 对话框，选择要下载到的 PLC 的网络接口卡和 IP 地址。
6. 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 43)和从 STOP 模式转换为 RUN 模式 (页 43)时您是否希望收到提示。

7. 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。
8. 单击“下载”(Download) 按钮。



STEP 7-Micro/WIN SMART 将完整程序或您所选择的程序组件复制到 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

最初创建的、用于固件版本为 V1.x 的 S7-200 SMART CPU 的项目组件可下载到固件版本为 V2.0 或更高版本的 CPU 中。然而，最初创建的、用于固件版本为 V2.0 或更高版本的 CPU 的项目组件可能无法成功下载到固件版本为 V1.x 的 CPU 中，在项目组件所用的功能不受固件版本 V1.x 支持时尤其如此。

下载过程

下载时，STEP 7-Micro/WIN SMART 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.	基于您所选择的下载对象，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您输入的新程序数据、保存并打开的 .smart 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 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 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。
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 848)和非致命运行错误 (页 844)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

在 RUN 模式下进行程序编辑

上传项目组件 (页 91)

4.3.2 上传项目组件

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

1. 确保网络硬件和 PLC 连接电缆运行正常 (页 31)，并确保 PLC 通信运行正常 (页 607)。
2. 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**。

3. 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（程序块、数据块或系统块）。
4. 如果弹出“通信”(Communications) 对话框，选择网络接口卡和要从中上传项目组件的 PLC 的 IP 地址。
5. 在“上传”(Upload) 对话框中，可改选要上传的块（如果已选择）。
6. （可选）如果想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 复制您选择从 PLC 上传到当前打开项目的完整程序或程序组件。
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。 PLC
不包含符号或状态图表信息；因此无法上传符号表或状态图表。

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，您不会意外损坏数据。
如果要使用位于另一项目的状态图表或符号表中的信息，可始终打开第二个 STEP 7-Micro/WIN SMART 实例，然后将该信息从另一项目文件复制过来 (页 110)。

如果要覆盖在下载 (页 88)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 项目时，才使用此选项。

STEP 7-Micro/WIN SMART
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。

4.3.3 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： 可拆卸 microSDHC 卡，用于作为程序传送卡 (页 95)存储项目块，作为恢复为出厂默认设置的卡 (页 168)完全擦除 PLC，或作为固件更新卡 (页 93)更新 PLC 和扩展模块固件。

4.3.4 使用存储卡

使用存储卡

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

- 用户程序传送 (页 95)
- 将 CPU 重置为出厂默认状态 (页 168)
- 支持 CPU 和连接的扩展模块的固件更新

可使用任何容量为 4GB 到 16GB 的标准型商业 microSDHC 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 4- 21　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。

要用于重置为出厂默认目的，按以下方式组织存储卡：

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。

要用于固件更新目的，按以下方式组织存储卡：

表格 4- 22　用于固件更新目的的存储卡

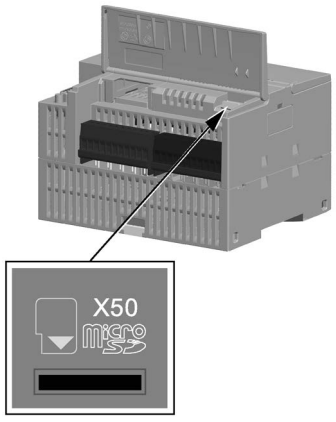
在卡的根级别	
文件： S7_JOB.S7S	包含字 FWUPDATE 的文本文件
文件夹： FWUPDATE.S7S	包含要更新的每个设备的更新文件 (.upd) 的文件夹

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，则用更新文件内包含的固件内容替换该设备的固件。

4.3.5 在 CPU 中插入存储卡

表格 4- 23　在 CPU 中插入和拔出存储卡

任务	步骤
	按照下面的步骤将 microSDHC 存储卡插入 CPU 中。 1. 打开下部的端子块连接器盖。 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. 在插入卡后重新装上端子块连接器盖，以确保该卡牢固。 按照下面的步骤从 CPU 中取下 microSDHC 存储卡。 1. 打开下部的端子块连接器盖。 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. 重新装上下部的端子块盖板。

4.3.6 通过存储卡传送程序

S7-200 SMART CPU 使用 FAT32 文件系统格式支持容量处于 4 GB 到 16 GB 范围内的标准商用 microSDHC 卡。可将 microSDHC 卡用作程序传送卡，实现程序和项目数据的便携式存储。

⚠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 自动转换到 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 43)。

创建程序传送存储卡

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 32)。
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 42)到 PLC。
4. 在 PLC 菜单功能区的“存储卡”(Memory Card) 区域单击“程序”(Program) 按钮。

5. 选择将以下哪些（或全部）块存储于存储卡：
   - 程序块
   - 数据块
   - 系统块（PLC 组态）
6. 单击“编程”(Program) 按钮。

7. 如果需要密码才能对存储卡进行编程，输入密码 (页 144)。

说明

STEP 7-Micro/WIN SMART 首先擦除卡中任何 SIMATIC 内容，然后再将程序传入卡中。使用读卡器和 Windows 资源管理器存入卡中的任何其它数据都保持原样。

另请注意，如果已插入存储卡，无法将 CPU 更改为 RUN 模式。

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 135)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

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。

参见

创建复位为出厂默认存储卡。 (页 168)

4.3.7 上电后恢复数据

CPU 上电时：

- CPU 从永久存储器中恢复程序块和系统块。
- 最多可恢复 10 KB 的保持性存储器分配。
- 而 V 存储器的非保持性部分将根据永久存储器中的数据块内容来恢复。
- 其它存储区的非保持性部分则会被清空。

4.4 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 不执行任何程序，而用户可以下载程序块。在 RUN 模式下，CPU 会执行相关程序；但用户仍可下载程序块。

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. 提示时，单击“确定”(OK) 更改 CPU 的工作模式。

可监视 STEP 7-Micro/WIN SMART 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 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 368)，以将 CPU 置于 STOP 模式。

4.5 状态 LED

CPU 使用 LED 提供有关 CPU 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
RUN	RUN：开 STOP、ERROR：灭	当 CPU 处于 RUN 模式时适用
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。
故障	STOP、ERROR：开 RUN：灭	当 CPU 处于故障模式时适用
Ping	STOP、RUN：以 2 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

编程概念

5

5.1 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。

5.2 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和若干子例程或中断例程组成。 代码已编译并下载到 CPU 中。
可以使用程序组织单元（主程序、子例程和中断例程）来结构化用户程序。

- 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。

- 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。

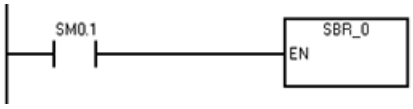
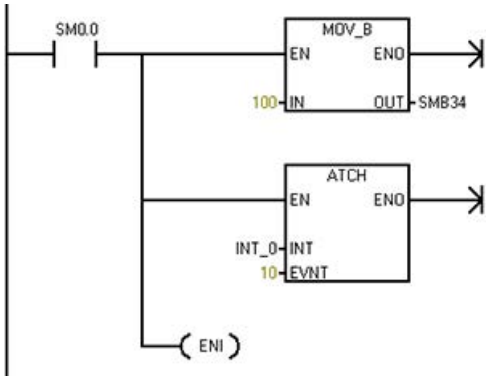
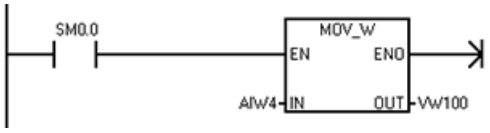
  使用中断例程的局部变量表可确保中断例程仅使用临时存储器，从而不会覆盖程序其它位置使用的数据。

  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 335)的说明。

- 其它块中包含 CPU 的信息。 下载程序时，您可以选择下载这些块：
  - 系统块： 系统块允许您为 CPU 组态不同的硬件选项。
  - 数据块： DB 存储程序使用的不同变量的初始值（V 存储器）。

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 5- 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。

5.3 创建用户程序

STEP 7-Micro/WIN SMART
的用户界面为创建用户项目程序提供了一个便捷的工作环境。
（STEP 7-Micro/WIN SMART 项目是带有 .smart 扩展名的文件。）
要打开用户界面，请双击 STEP 7-Micro/WIN SMART 图标，或者从“开始”(Start) 菜单的“SIMATIC”组件中选择“STEP 7-MicroWIN SMART”。

5.3.1 早期版本的 STEP 7-Micro/WIN 项目

要使用在 4.0 或更高版本的 STEP 7-Micro/WIN 中创建的项目，按以下步骤操作：

- 在“文件”(File) 菜单功能区的“操作”(Operations) 区域单击“打开”(Open) 按钮，然后选择所需项目。

- 根据需要更正程序。

无法打开使用旧版本（早于 STEP 7-Micro/WIN 4.0
版）创建的项目。如果试图打开此类项目，STEP 7-Micro/WIN SMART
通知您无法打开。

说明

打开用旧版程序创建的项目

- 由早期版 STEP 7-Micro/WIN（.mwp 文件）创建的项目可能包含一种或多种 STEP 7-Micro/WIN SMART（.smart 文件）不支持的逻辑结构。如果旧版项目包含 STEP 7-Micro/WIN SMART 不支持的指令，则在 STEP 7-Micro/WIN SMART 中打开项目时，会将这些指令从项目中忽略。必须仔细检查项目，并对忽略逻辑的部分进行重新设计。
- STEP 7-Micro/WIN SMART 忽略旧版项目的系统块，对打开的项目使用默认系统块。
- STEP 7-Micro/WIN SMART 会忽略旧项目中所有向导生成的程序块。
- 如果较早版本的 STEP 7-Micro/WIN（.mwp 文件）使用 OB 中的符号 SM 寻址，且已生成系统符号表，则符号将正确映射到新地址。但是，如果 .mwp 文件使用 OB 中的绝对 SM 寻址，则那些绝对 SM 地址将不会映射到新 SM 地址。更多相关信息，请参见符号表 (页 117)或特殊存储器 (页 851)。
- 您不能使用“打开”命令打开位于 PLC 中的项目；项目文件必须位于您的个人计算机/编程设备中。
- 只可为每个 STEP 7-Micro/WIN SMART 实例打开一个项目。必须运行两个 STEP 7-Micro/WIN SMART 实例才能同时打开两个项目。打开两个实例时，可在其间复制和粘贴 LAD/FBD 程序元素和 STL 文本。
- 可定义一个默认路径，指向用于打开和保存新 STEP 7-Micro/WIN 项目的具体文件目录。在“工具”(Tools) 菜单功能区的“设置”(Settings) 区域单击“选项”(Options) 按钮；单击“常规”(General) 选项，然后通过“默认值”(Defaults) 选项卡输入默认文件位置。

⚠警告

使用绝对特殊存储器 (SM) 寻址的 STEP 7-Micro/WIN 版本 4.0 或更高版本（.mwp 文件）存在风险

如果较早版本的 STEP 7-Micro/WIN（.mwp 文件）使用 OB 中的符号 SM 寻址，且已生成系统符号表，则符号将正确映射到新地址。但是，如果 .mwp 文件使用 OB 中的绝对 SM 寻址，则那些绝对 SM 地址将不会映射到新 SM 地址。

如果 SM 地址的映射错误，则会导致意外的机械或过程操作，从而可能导致人员死亡、重伤和/或设备损坏。

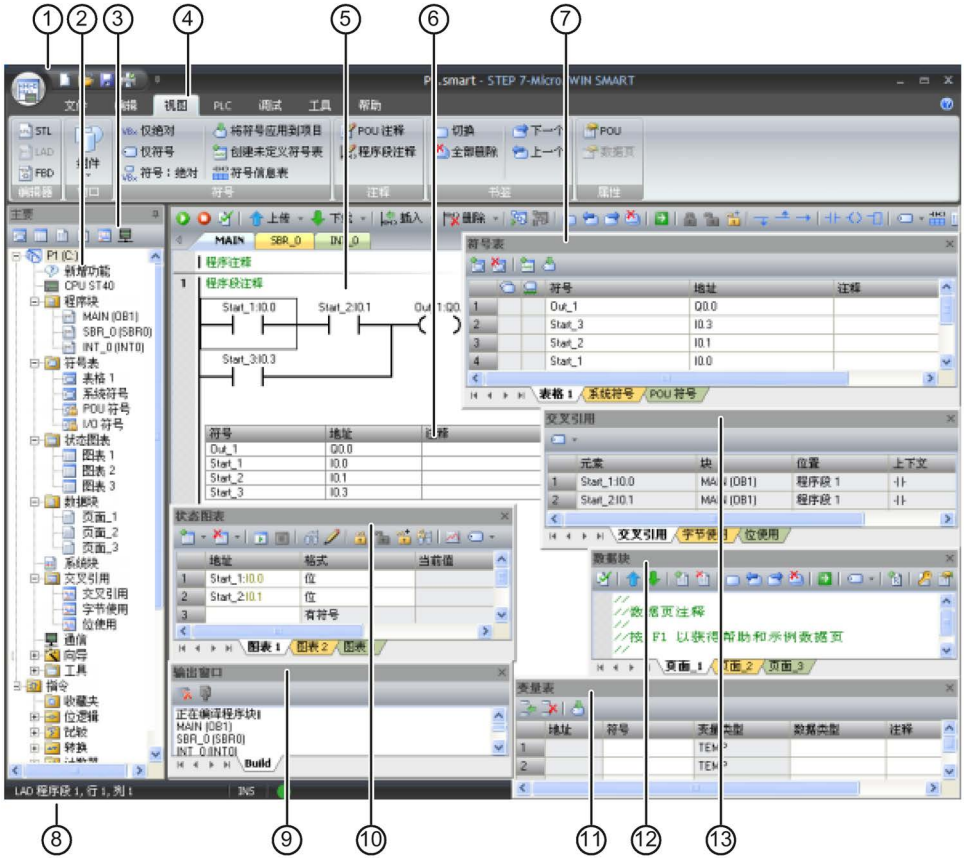
删除“S7-200 符号”表并生成 SMART“系统符号”表。OB 中的符号将映射到 SMART 系统符号表中的新 SM 地址方案。

5.3.2 使用 STEP 7-Micro/WIN SMART 用户界面

STEP 7-Micro/WIN SMART

用户界面如下所示。请注意，每个编辑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（如下所示），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 110)
② 项目树 (页 110)
③ 导航栏 (页 110)
④ 菜单 (页 110)
⑤ 程序编辑器 (页 110)
⑥ 符号信息表 (页 117)
⑦ 符号表 (页 117)
⑧ 状态栏 (页 110)
⑨ 输出窗口 (页 110)
⑩ 状态图表 (页 600)
⑪ 变量表 (页 121)
⑫ 数据块 (页 115)
⑬ 交叉引用 (页 594)

5.3.3 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正上方。通过快速访问文件按钮可简单快速地访问“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) 文件夹插入新的子例程和中断例程。
- 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“符号表”(Symbol Table) 文件夹，插入新图或新表。
- 打开“状态图”(Status Chart) 或“符号表”(Symbol 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 V2.2
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。

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。导航栏具有几组图标，用于访问
STEP 7-Micro/WIN SMART 的不同编程功能。

菜单功能区

STEP 7-Micro/WIN SMART
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。子例程和中断例程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 提供了三个用于创建程序的编辑器：

- 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，显示在 STEP 7-Micro/WIN SMART 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。

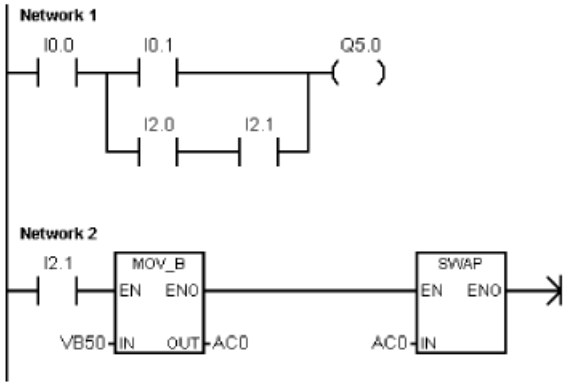
5.3.4 借助向导创建控制程序

STEP 7-Micro/WIN SMART 提供以下向导使编程变得更自动更容易：

- 高速计数器
- 运动
- PID
- PWM（脉宽调制）
- 文本显示
- Get/Put
- 数据日志

要启动向导，可在 STEP 7-Micro/WIN SMART“工具”(Tools)
菜单功能区或在项目树下的向导节点中选中此向导。 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。

5.3.5 LAD 编辑器的特点



LAD
编辑器以图形方式显示程序，与电气接线图类似。

LAD
程序仿真来自电源的电流通过一系列的逻辑输入条件，进而决定是否启用逻辑输出。

LAD 程序包括已通电的左侧电源导轨。
闭合触点允许能量通过它们流到下一元件，而断开的触点则阻止能量的流动。
逻辑分成不同的程序段。
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。

各种指令通过图形符号表示，包括三个基本形式：

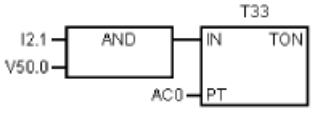
- 触点表示逻辑输入条件，如开关、按钮或内部条件。
- 线圈通常表示逻辑输出结果，如指示灯、电机启动器、干预继电器或内部输出条件。
- 方框表示其它指令，如定时器、计数器或数学指令。

选择 LAD 编辑器时，请考虑以下要点：

- 梯形图逻辑易于初学者使用。
- 图形表示法通常易于理解，且全世界通用。
- 可以使用 STL 编辑器显示所有用 SIMATIC LAD 编辑器编写的程序。

5.3.6 FBD 编辑器的特点

FBD
编辑器以图形方式显示程序，类似于通用逻辑门图。
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。



FBD 不使用左右侧电源导轨的概念，因此，术语“逻辑流”用于表达流过 FBD 逻辑块的控制流的类似概念。

通过 FBD 元件的逻辑“1”称为逻辑流。
逻辑流输入的起点和逻辑流输出的终点可以直接分配给操作数。

程序逻辑由这些框指令之间的连接决定。 即，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。这一连接概念使能能够解决各种各样的逻辑问题。

选择 FBD 编辑器时，请考虑以下要点：

- 图形逻辑门表示样式对跟随程序流有益。
- 可以使用 STL 编辑器显示所有用 SIMATIC FBD 编辑器编写的程序。

5.3.7 STL 编辑器的特点

STL 编辑器以文本语言的形式显示程序。 STL 编辑器允许您输入指令助记符来创建控制程序。 STL 编辑还允许您创建用 LAD 或 FBD 编辑器无法创建的程序。 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绘图。 如下例所示，这种基于文本的概念与汇编语言编程十分相似。

表格 5- 2　STL 用户程序示例

LD	I0.0	// 读取一个输入 (I0.0)。
A	I0.1	// 与另一个输入 (Q1.0) 进行"与"运算。
=	Q1.0	// 将值写入输出 1。

CPU 按照程序指示的顺序，从顶部至底部执行每条指令，然后再从头重新开始。

STL 使用逻辑栈解析控制逻辑。 插入 STL 指令来处理堆栈操作。

选择 STL 编辑器时，请考虑以下要点：

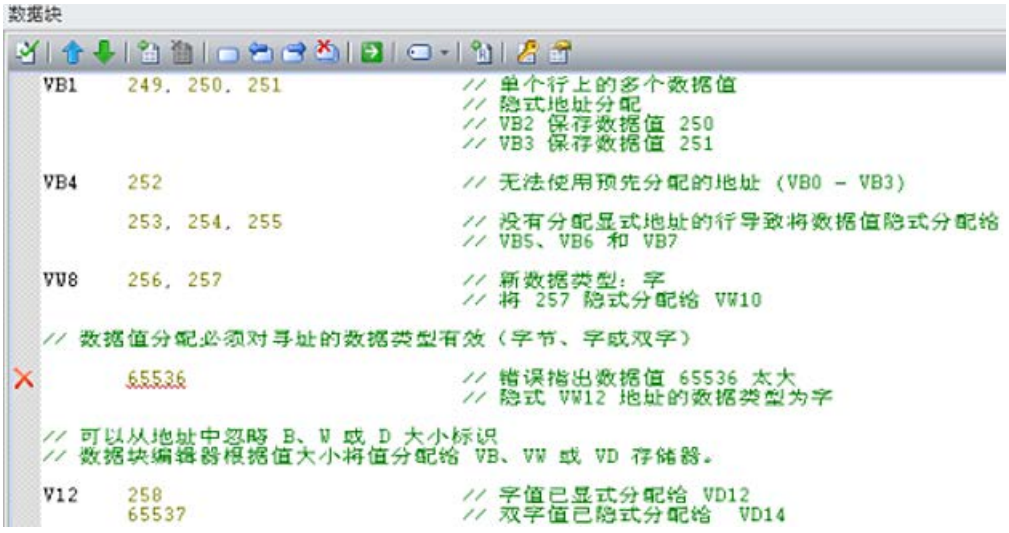
- STL 对经验丰富的程序员最适合。
- STL 有时可以解决无法用 LAD 或 FBD 编辑器轻易解决的问题。
- 虽然您可以使用 STL 编辑器查看或编辑用 LAD 或 FBD 编辑器创建的程序，但反过来不一定成立。 LAD 或 FBD 编辑器不一定总能显示所有用 STL 编辑器编写的程序。

5.4 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 82)（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 //。

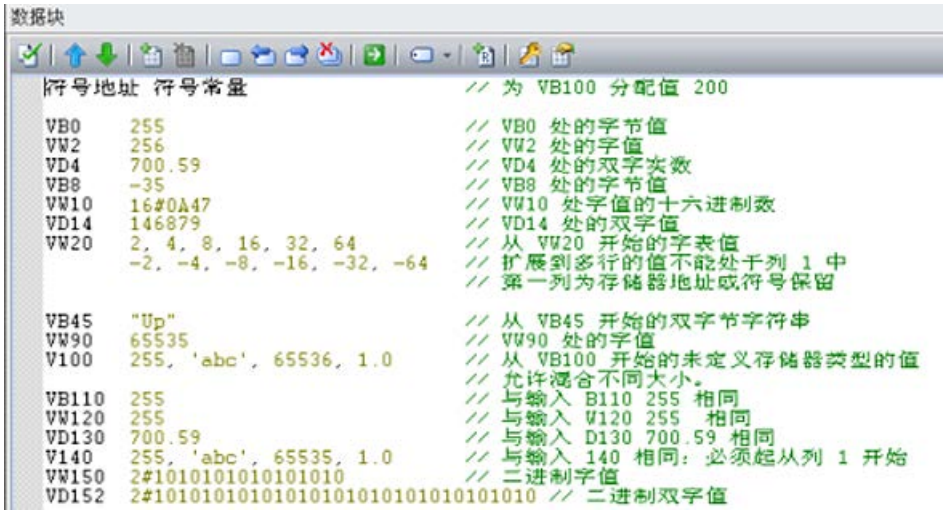
- 数据块的第一行必须分配显式地址。可使用存储器地址（绝对地址）或符号表 (页 117)中以前分配给地址的符号名称（符号地址）。
- 后续行可分配显式地址或隐式地址。当您在单个地址分配后键入多个数据值时，或键入仅包含数据值的一行时，编辑器会自动进行隐性地址分配。编辑器根据先前的地址分配及数据值大小（字节、字或双字），指定适当数量的 V 存储区。
- 数据块编辑器是一种自由格式文本编辑器；但是，它预期地址或符号名称出现在第一个位置。如果继续输入一个隐式数据值条目，输入隐式赋值前在地址位置输入至少一个空格。键入一行后，按 ENTER
  键，数据块编辑器格式化该行（对齐地址列、数据和注释；大写 V
  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。

示例：数据块页面

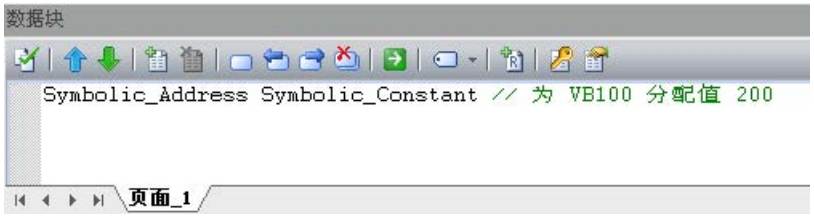


注：在输入非显性地址的行上的数据值前输入一个空格。

示例：直接地址和数字值



示例：符号地址和符号数分配



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。

5.5 符号表

符号是可为存储器地址或常量指定的符号名称。您可为下列存储器类型创建符号名：I、Q、M、SM、AI、AQ、V,
S、C、T、HC。在符号表中定义的符号适用于全局。已定义的符号可在程序的所有程序组织单元 (页 103) (POU) 中使用。如果在变量表
(页 121)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。此类符号被称为“局部变量”，与适用于全局范围的符号有区别。符号可在创建程序逻辑之前或之后进行定义。

⚠警告

使用绝对特殊存储器 (SM) 寻址的 STEP 7-Micro/WIN 版本 4.0 或更高版本（.mwp 文件）存在风险

如果较早版本的 STEP 7-Micro/WIN（.mwp 文件）使用 OB 中的符号 SM 寻址，且已生成系统符号表，则符号将正确映射到新地址。但是，如果 .mwp 文件使用 OB 中的绝对 SM 寻址，则那些绝对 SM 地址将不会映射到新 SM 地址。

如果 SM
地址的映射错误，则会导致意外的机械或过程操作，从而可能导致人员死亡、重伤和/或设备损坏。

删除“S7-200 符号”表并生成 SMART“系统符号”表。OB 中的符号将映射到 SMART 系统符号表中的新 SM 地址方案。

打开符号表

要打开符号表，可使用以下方法之一：

- 单击导航栏 (页 26)中的“符号表”(Symbol Table) 按钮。
- 在“视图”(View) 菜单的“窗口”(Windows) 区域中，从“组件”(Component) 下拉列表中选择“符号表”(Symbol Table)。
- 在项目树 (页 35)中打开“符号表”(Symbol Table) 文件夹，选择一个表名称，然后按下 Enter 或者双击表名称。

还可在项目中使用系统符号表中的符号。预定义的系统符号表提供了对常用 PLC 系统功能的访问。PLC 系统符号将功能名称与用于调用该功能的 PLC 特殊存储位置相关联。

在符号表中分配符号

要将符号分配给地址或常数值，请按以下步骤操作：

1. 打开符号表。
2. 在“符号”(Symbol) 列中键入符号名（例如，Input1）。符号名可包含的最大字符数为 23 个单字节字符。

说明

在为符号指定地址或常数值之前，该符号一直显示为未定义符号（绿色波浪下划线）。在完成“地址”(Address) 列分配后，STEP 7-Micro/WIN SMART 将移除绿色波浪下划线。

如果已选择同时显示项目操作数的符号视图和绝对视图，则程序编辑器中较长的符号名将以波浪号 (~) 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“地址”(Address) 列中键入地址或常数值（例如，VB0 或 123）。请注意，在为符号分配字符串常量时，需要用双引号将该字符串常量括起来。
4. 也可以键入最长为 79 个字符的注释。

可根据需要在符号表编辑器中调整列宽。

说明

可创建多个符号表；但是，在进行全局符号分配时，不可多次使用同一符号名。相反，可在变量表中重复使用符号名。

语法规则和错误指示

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

符号	地址
2name	I0.0
	VBB0
Begin	I0.2

红色文本表示语法无效。
符号不能以数字开头。
VBB0 为无效地址。
Begin 为预留的字，是无效的符号名。

符号	地址
Pump1	I0.0
Pump1	I0.0
SymConstant	1234
SymConstant	5678

红色波浪下划线表示用法无效。
Pump1 和 SymConstant 是重复的符号名。
I0.0 是重复的地址。

符号	地址
Pump1	

绿色波浪下划线表示未定义符号。
Pump1 没有地址。

定义符号时应遵守以下语法规则：

- 符号名可包含字母数字字符、下划线以及从 ASCII 128 到 ASCII 255 的扩充字符。第一个字符不能为数字。
- 使用双引号将指定给符号名的 ASCII 常量字符串括起来。
- 使用单引号将字节、字或双字存储器中的 ASCII 字符常量括起来。
- 不要使用关键字作为符号名。
- 符号名的最大长度为 23 个字符。

说明

在更正错误的符号名或地址后，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符号时，可以像直接地址一样对符号名使用间接记号（& 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。

查看重叠符号和未使用的符号

STEP 7-Micro/WIN SMART 以 图标指示重叠符号，以 图标指示未使用的符号。在下面的符号表中，符号 S1 和 S2 重复使用 VB0 存储器地址。另外，符号 S1 未在项目中使用。

插入附加行

使用以下方法之一可在符号表中插入附加行：

- 右键单击符号表中的单元格，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。
- 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 将新行插入到符号表中光标所在位置上方。
- 要在符号表底部插入新行，可将光标放在最后一行的任意一个单元格中，然后按“下箭头”键。

对符号表排序

可以基于“符号”(Symbol) 或“地址”(Address) 列按字母升序或降序对符号表进行排序。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“符号”(Symbol) 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 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) 区域打印符号表。

可通过显示符号信息表来按网络查看符号。

5.6 变量表

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

- 您要创建不引用绝对地址或全局符号的可移值子例程。
- 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。
- 您要为子例程定义输入和输出。

如果以上描述对您的具体情况不适用，则无需使用局部变量；可在符号表 (页 117)中定义符号值，从而将其全部设置为全局变量。

了解局部变量

您可以使用程序编辑器的变量表来分配对个别子例程或中断例程唯一的变量。

局部变量可用作传递至子例程的参数，并可用于增加子例程的移植性或重新使用子例程。

程序中的每个 POU (页 103) 都有自身的变量表，并占 L 存储器的 64 个字节（如果在 LAD 或 FBD 中编程，则占 60 个字节）。借助局部变量表，可对特定范围内的变量进行定义：局部变量仅在创建时所处的 POU 内部有效。相反，在每个 POU 中均有效的全局符号只能在符号表中定义。当您为全局符号和局部变量使用相同的符号名时（例如 INPUT1），在定义局部变量的 POU 中局部定义优先，在其他 POU 中使用全局定义。

在局部变量表中进行分配时，指定声明类型（TEMP、IN、IN_OUT 或 OUT）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。

变量表符号地址分配将符号名称与存储相关数据值的 L 存储器地址进行关联。局部变量表不支持对符号名称直接赋值的符号常数（这在符号/全局变量表中是允许的）。

说明

PLC 不会将本地数据值初始化为零。您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 (OB1)、中断例程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间进行传递，具体如下所述：

- 如果要将数据值传递至子例程，则在子例程变量表中创建一个变量，并将其声明类型指定为 IN。
- 如果要将子例程中建立的数据值传回至调用例程，则在子例程的变量表中创建一个变量，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，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，然后在子例程变量表中创建一个变量，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。

局部变量的数据类型检查

将局部变量作为子例程参数传递时，在该子例程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

您从 OB1 调用 SBR0，将称为 INPUT1 的全局符号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 OB1 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。

查看变量表

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

说明

可将变量表放在快速访问工具栏 (页 103)上以便于访问。

在变量表中赋值

说明

在程序中使用局部变量**之前**，先在变量表中赋值。在程序中使用符号名时，程序编辑器首先检查相应 POU
的局部变量表，然后检查符号表。如果符号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符号；此类符号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变量表并对您的程序逻辑做出更正。如果以后对该符号名称的数据类型分配进行定义（在局部变量表中），必须在符号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
16。如果尝试下载一个超出此项限制的程序，STEP 7-Micro/WIN SMART 返回错误。

要在变量表中赋值，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 都有自己的变量表，所以需要确保对正确的 POU 赋值。）
2. 如果变量表尚不可见，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“变量表”(Variable Table)。
3. 选择变量类型与要定义的变量类型相符的行，然后在“符号”(Symbol) 字段输入变量名称。如果在 OB1 或中断例程中赋值，变量表只含 TEMP 变量。如果在子例程中赋值，变量表包含 IN、IN_OUT、OUT 和 TEMP 变量。在变量表中不要在名称前加上星号。井号只用在程序代码中的局部变量前。

说明

局部变量名称允许包含字母数字字符和下划线的数量最多为 23
个，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符号名，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参数时，必须确保分配给局部变量的数据类型不与子例程调用中正在使用的操作数发生冲突。

5. 也可提供注释，描述局部变量。

为“符号”(Symbol) 和“数据类型”(Data Type) 字段提供值后，程序编辑器自动将 L 存储器地址分配给局部变量。

输入附加变量

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 > 下一行”(Insert > Row Below)** 来添加行。

删除变量

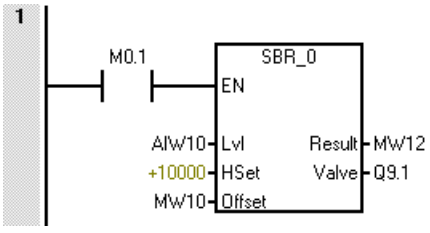
要删除局部变量，需在变量表中选中此变量，然后单击“删除”(Delete) 按钮 
。也可删除一行，方法是右键单击该行，然后从上下文菜单中选择**“删除 > 行”(Delete > Row)**。

变量表示例

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

变量表

	地址	符号	变量类型	数据类型	注释
1		EN	IN	BOOL	
2	LW0	Lvl	IN	INT	罐发送器
3	LW2	Hset	IN	INT	高设定值
4	LW4	Offset	IN	INT	偏移量
5			IN		
6			IN_OUT		
7	LW6	Result	OUT	INT	结果值
8	L8.0	Valve	OUT	BOOL	控制阀
9			OUT		



参见

编程软件 (页 26)

5.7 PLC 错误响应

在 PLC 菜单功能区的“信息”(Information) 部分单击 PLC 按钮可查看当前错误状态。

要识别具体错误，请参见错误代码列表 (页 843)。

选择了树的“设备”(Devices) 条目后，将显示 CPU 和任何扩展模块的简要错误状态。要显示每个设备的详细错误信息，在该树中选择设备名称。

错误和状态信息：

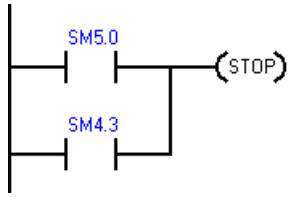
- “上一致命错误”(Last Fatal) 字段显示的是 CPU 生成的上一个致命错误代码。在上电循环过程中保留此值。 在清除 CPU 的全部存储器时始终清除此位置。
- 在树中选择“事件”(Event) 日志条目会显示 CPU 的已存储事件历史，包括上电、掉电、错误和模式转换。 还会列出事件发生的时间。
- PLC 还提供 SM 位用于错误的已编程响应。 请参见 SM 位 (页 851)的列表。
- GET_ERROR（获取非致命错误代码）程序指令将返回 PLC 当前的非致命错误代码并清除 PLC 中锁存的非致命错误信息。 有关详细信息，请参见 GET_ERROR 指令 (页 370)。

5.7.1 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 5- 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。 可以使用 STEP 7-Micro/WIN SMART 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 的 PLC 的当前错误状态。

表格 5- 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。 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，程序编译时有效的间接地址指针在程序执行过程中可能会改为指向非法地址。这是一个运行程序问题的例子。 发生运行程序故障时，SM4.3 会置位，并会在 CPU 处于 RUN 模式期间一直保持置位。 可通过执行 GET_ERROR 指令获取任何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 844)。

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 851)的描述。

5.7.2 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，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，需要修理 PLC；更改程序或清空 PLC 存储器无法解决这些问题。

有关详细信息，请参见致命错误代码列表 (页 848)。

5.8 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（相对仿真过程而言，程序调试期间可能会进行模拟仿真），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，子例程和中断）。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。
- 运行 STEP 7-Micro/WIN SMART 的计算机与 PLC 之间必须已成功建立通信。
- 目标 PLC 的固件必须支持在 RUN 模式下进行程序编辑。仅带有 V2.0 版或更新版固件的 S7-200 SMART CPU 支持在 RUN 模式下进行程序编辑。
- 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。

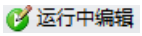
处理正负跳变

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 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 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 会提示您保存项目。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 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变更。

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 中的其它任务。

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 137)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。

5.9 用于调试程序的功能

STEP 7-Micro/WIN SMART 提供了下列功能来帮助您调试程序：

- 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 594)跟踪程序中的引用
- 使用状态图 (页 600)显示 PLC 数据值和状态
- 显示程序编辑器中的状态 (页 596)

有关调试程序的详细信息，请参见诊断和故障排除 (页 593)章节。

PLC 设备组态

6

6.1 组态 PLC 系统的运行

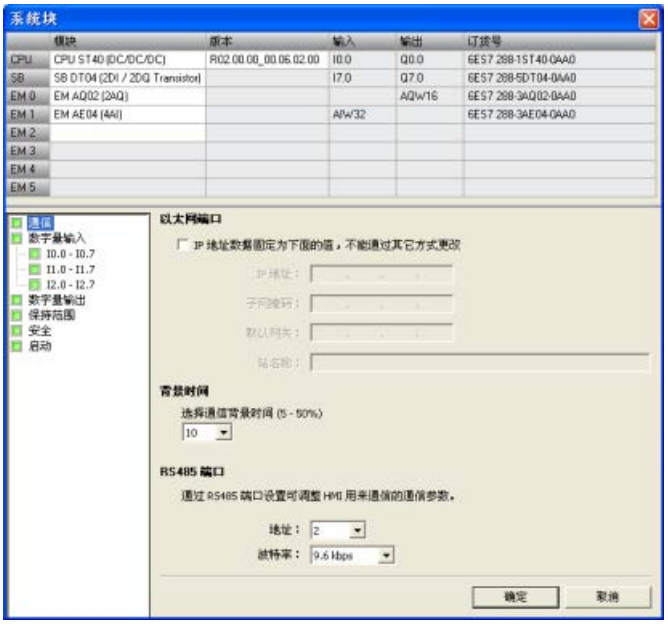
6.1.1 系统块

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 (页 26)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 (页 26)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 (页 26)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（V1 或 V2）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。

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 137)
- 数字量输入和脉冲捕捉位 (页 139)
- 数字量输出 (页 141)
- 保持范围 (页 142)
- 安全 (页 144)
- 启动 (页 148)

其它设备（如模拟量输入 (页 149)、模拟量输出 (页 152)、RTD 模拟量输入
(页 154)、热电偶 (TC) 模拟量输入 (页 159)、RS485/RS232 CM01 通信信号板
(页 163)、电池 BA01 信号板
(页 165)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。

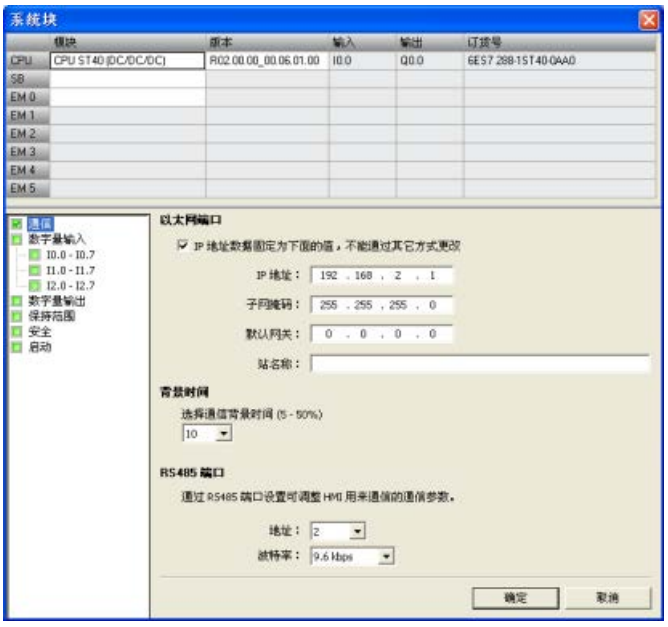
在下载或上传系统块之前，必须在 STEP 7-Micro/WIN SMART 与 CPU 之间建立通信。

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 (页 42)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。

6.1.2 对通信进行组态

单击“系统块”(System Block) (页 135) 对话框的“通信”(Communication) 节点组态以太网端口、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它网络发送消息，这些网络可能还隐含着其它 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。

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它设备间的连接。如果有 HMI 设备或其它的 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。

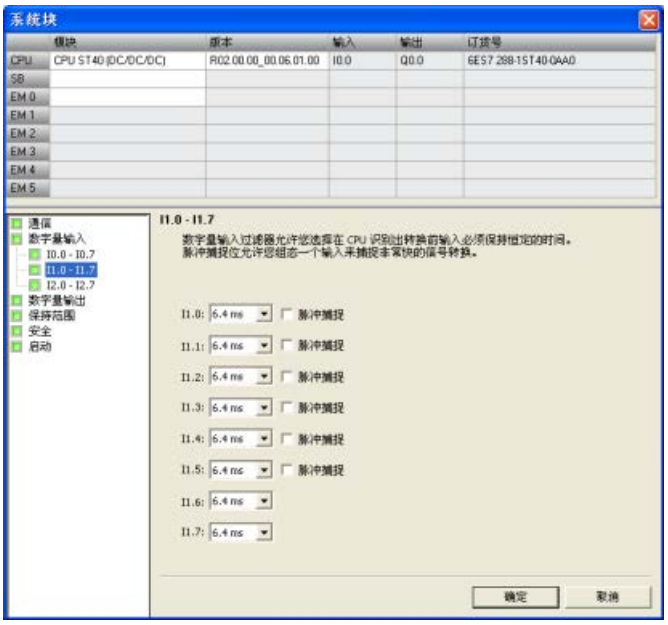
RS485 端口

使用以下设置对板载 RS485 端口调整系统协议通信参数。连接 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）。

6.1.3 组态数字量输入

单击“系统块”(System Block) (页 135) 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，用户可以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 (页 20)。

前十四个输入点（I0.0 到 I0.7 以及 I1.0 到 I1.5）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）。 其余输入点（I1.6 及以上）仅支持输入延迟选项的限定设置（6.4 ms、12.8 ms 或者不过滤）。

例如，CPU SR20 的所有十二个输入点均支持输入延迟设置的扩展列表。 对于 CPU ST40，输入延迟选项的扩展列表适用于其前十四个输入点，其余十个输入点则仅支持限定列表。

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。
在此期间，可能不会检测到持续时间少于 12.8 ms 的短“0”脉冲事件或对其计数。

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

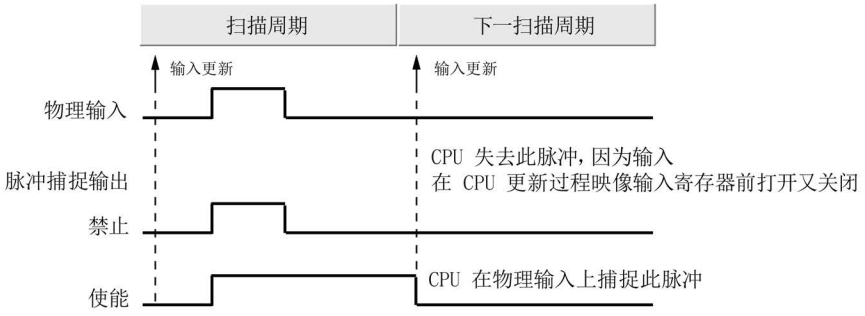
S7-200 SMART CPU 为数字量输入点提供脉冲捕捉功能。
通过脉冲捕捉功能可以捕捉高电平脉冲或低电平脉冲。此类脉冲出现的时间极短，CPU 在扫描周期开始读取数字量输入时，可能无法始终看到此类脉冲。

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。

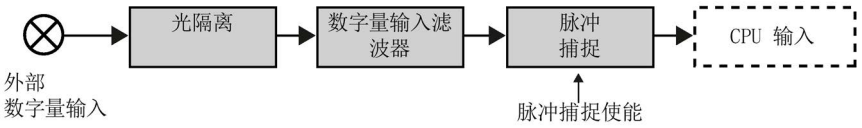
可根据 CPU 型号 (页 20)单独启用前十四个数字量输入点（I0.0 至 I0.7 以及 I1.0 至 I1.5）的脉冲捕捉操作。

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。

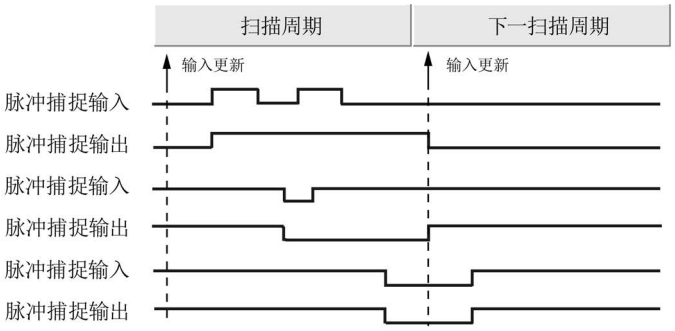
下图显示 S7-200 SMART CPU（脉冲捕捉启用和未启用）的基本操作状况：



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：

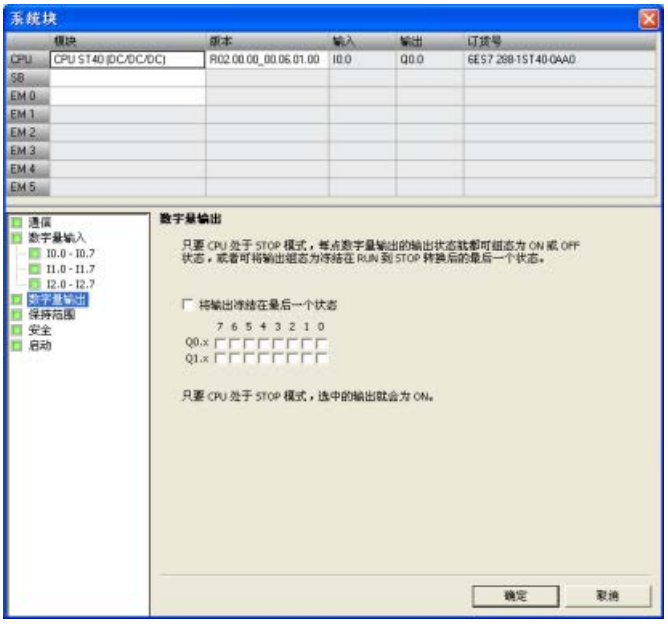


下图显示启用脉冲捕捉功能时对各种不同输入条件的响应。
如果在某一特定扫描中存在一个以上脉冲，仅读取第一个脉冲。
如果在某一特定扫描中有多个脉冲，则应当使用上升/下降沿中断事件：



6.1.4 组态数字量输出

单击“系统块”(System Block) (页 135) 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)。

6.1.5 组态保持范围

单击“系统块”(System Block) (页 135) 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 和 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 中并未定义保持区域，但可组态保持范围以保持最多 10 KB 的存储器空间。

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：**

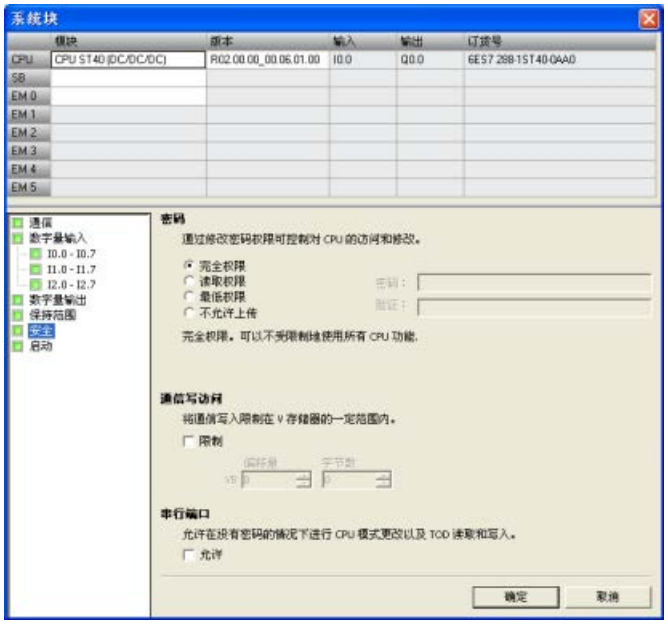
  CPU 先将 V、M、C 和 T 存储器清零，将所有初始值都从数据块复制到 V 存储器，然后将保存的保持值从永久存储器复制到 RAM。

S7-200 SMART CPU 存储器地址保持范围

数据类型	描述	CPU CR40	CPU CR60	CPU SR20 CPU ST20	CPU SR30 CPU ST30	CPU SR40 CPU ST40	CPU SR60 CPU ST60
V	数据存储器	VB0-VB8191	VB0-VB8191	VB0-VB8191	VB0-VB12281	VB0-VB16383	VB0-VB20479
T	定时器	T0-T31、T64-T95	T0-T31、T64-T95	T0-T31、T64-T95	T0-T31、T64-T95	T0-T31、T64-T95	T0-T31、T64-T95
C	计数器	C0-C255	C0-C255	C0-C255	C0-C255	C0-C255	C0-C255
M	标志位	MB0-MB31	MB0-MB31	MB0-MB31	MB0-MB31	MB0-MB31	MB0-MB31

6.1.6 组态系统安全

单击“系统块”(System Block) (页 135) 对话框的“安全”(Security) 节点组态 CPU 的密码及安全设置。



密码可以是字母、数字和符号的任意组合，区分大小写。

密码保护权限级别

CPU 提供四级密码保护，“完全权限”（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。S7-200 SMART CPU 的默认密码级别是“完全权限”（1 级）。

CPU 密码授权访问 CPU 功能和存储器。未下载 CPU 密码（“完全权限”（1 级））情况下，S7-200 SMART CPU 允许无限制访问。如果已组态比“完全权限”（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 6- 1　S7-200 SMART CPU 密码保护权限级别

操作说明	**完全权限（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 的启动、停止和上电复位	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
下载程序块、数据块或系统块	允许	有限制	有限制	有限制 注：如果存在用户程序块，不允许对系统块进行操作。
复位为出厂默认设置	允许	有限制	有限制	有限制
删除程序块、数据块或系统块	允许	有限制	有限制	有限制 注：如果存在用户程序块，不允许对系统块进行操作。
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
强制状态图中的数据	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

操作说明	完全权限（1 级）	读取权限（2 级）	最低权限（3 级）	不允许上传（4 级）
程序状态	允许	允许	有限制	不允许
项目比较	允许	允许	有限制	不允许

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，禁止对其它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”(Restrict) 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 和其它 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 和 AQ。

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（内置 RS485 和 RS485/RS232 信号板）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”(Serial Ports) 部分选中“允许”(Allow) 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 更改工作模式和进行 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 更改工作模式和进行 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 更改工作模式和进行 TOD 写入。

访问受密码保护的 CPU

说明

输入受密码保护的 CPU 的密码后，当编程设备从 S7-200 SMART CPU 断开后，该密码的授权级别最多可保持一分钟有效时间。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

由于 1 级权限允许所有不受限制的 CPU 访问，因此可通过将权限级别 4、3 或 2 更改为“完全权限”（1 级）来禁用密码。

说明

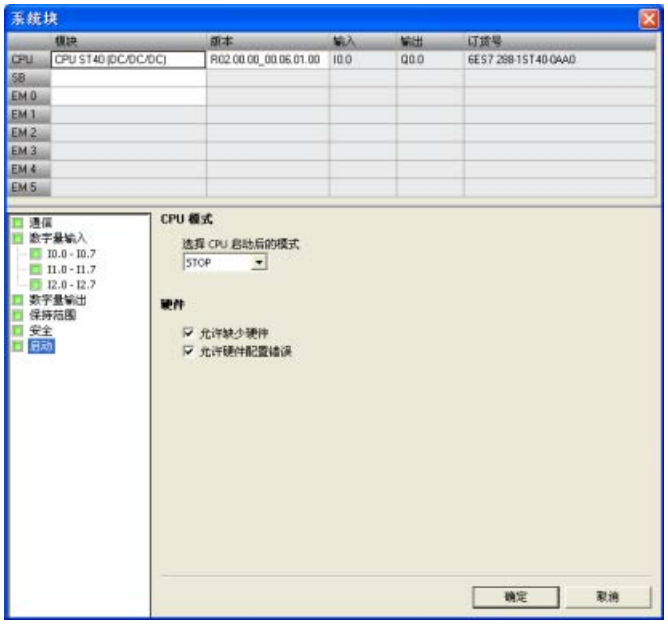
如果权限级别为“不允许上传”（4 级），存在有效用户程序时无法通过新的密码级别下载新的系统块。必须首先删除该用户程序，然后才能下载更新的系统块。

忘记密码怎么办

如果忘记密码，只有一种选择：使用“复位为出厂默认存储卡”(Reset-to-factory-defaults memory card)。（有关详细信息，请参见清除 PLC 存储器 (页 166)。）

6.1.7 组态启动选项

单击“系统块”(System Block) (页 135) 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 (页 43)。

硬件选项

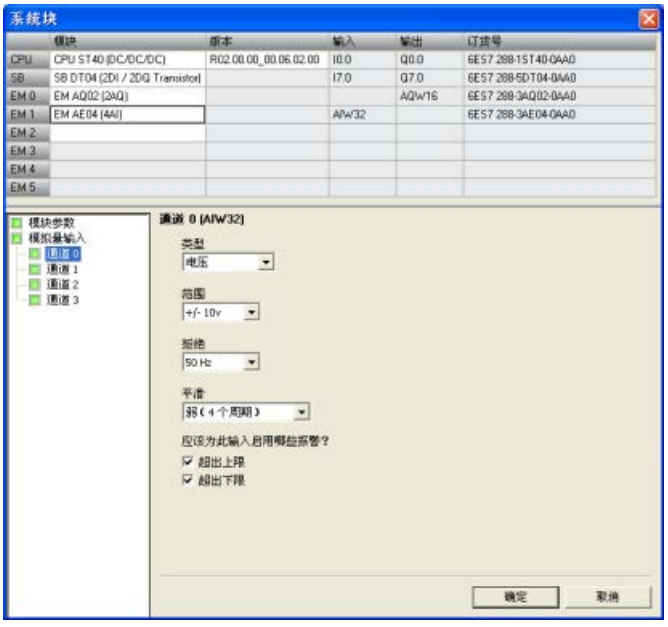
还可组态 CPU 以允许在以下硬件条件下以 RUN 模式运行：

- 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。

6.1.8 组态模拟量输入

单击“系统块”(System Block) (页 135) 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)

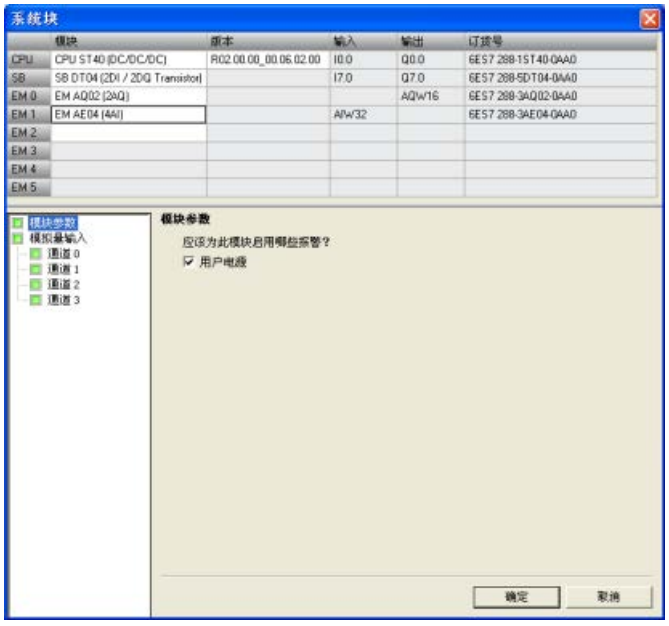
可组态模块在组态的周期数内平滑模拟量输入信号，从而将一个平均值传送给程序逻辑。有四种平滑算法可供选择：

- 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。）



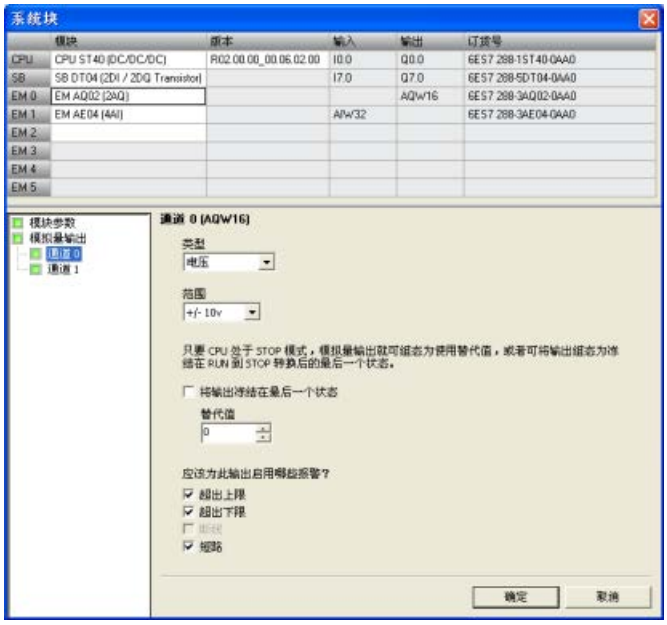
6.1.9 模拟量输入技术规范参考

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 801)
- 抑制：“模拟量输入的采样时间和更新时间” (页 801)
- 滤波：“模拟量输入的阶跃响应” (页 800)

6.1.10 组态模拟量输出

单击“系统块”(System Block) (页 135) 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。

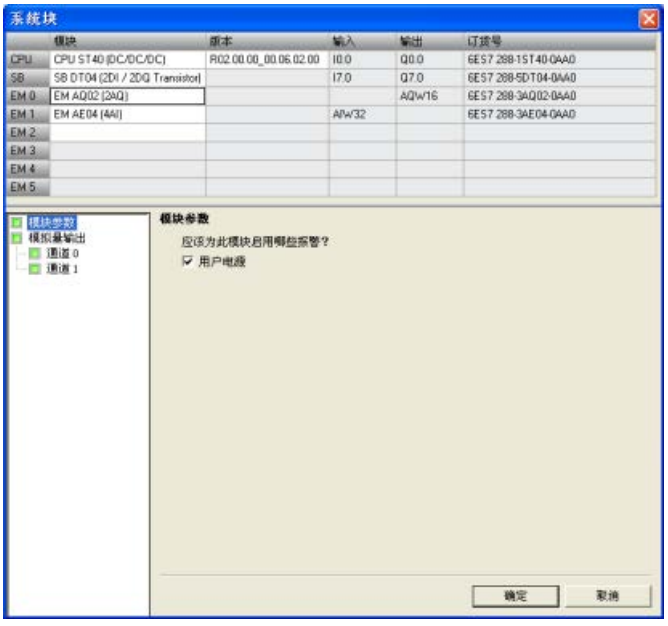
STOP 模式下，有两种方法可用于设置模拟量输出行为：

- “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。）

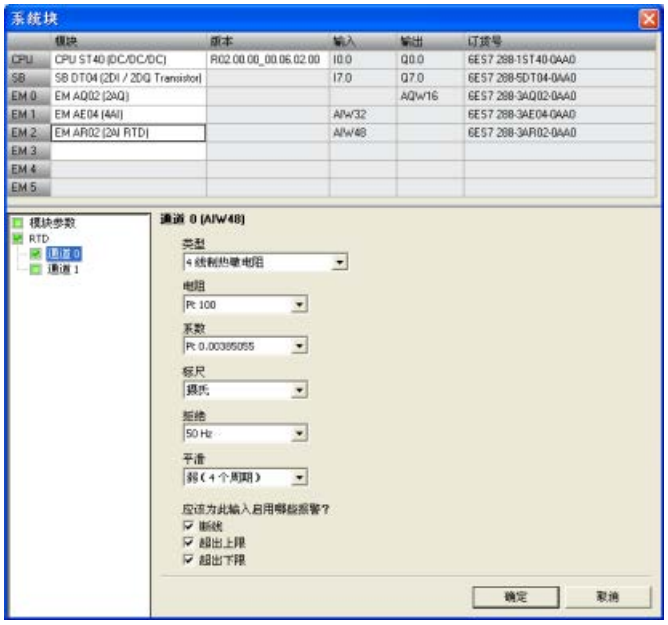


6.1.11 模拟量输出技术规范参考

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 803)技术规范

6.1.12 组态 RTD 模拟量输入

在“系统块”(System Block) (页 135) 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 6- 2　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。	• 48 欧姆 • 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
• 48 欧姆 • 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
• 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

RTD 电阻	RTD 温度系数
• 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

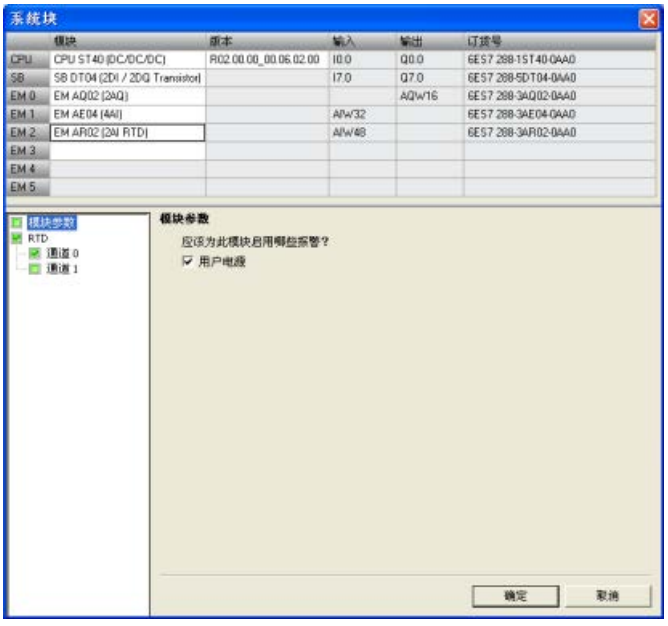
用户可对模块进行组态，在组态的周期数内平滑 RTD 模拟量输入信号，然后将平均值传送至程序逻辑。有四种平滑算法可供选择：

- 无
- 弱
- 中
- 强

报警组态

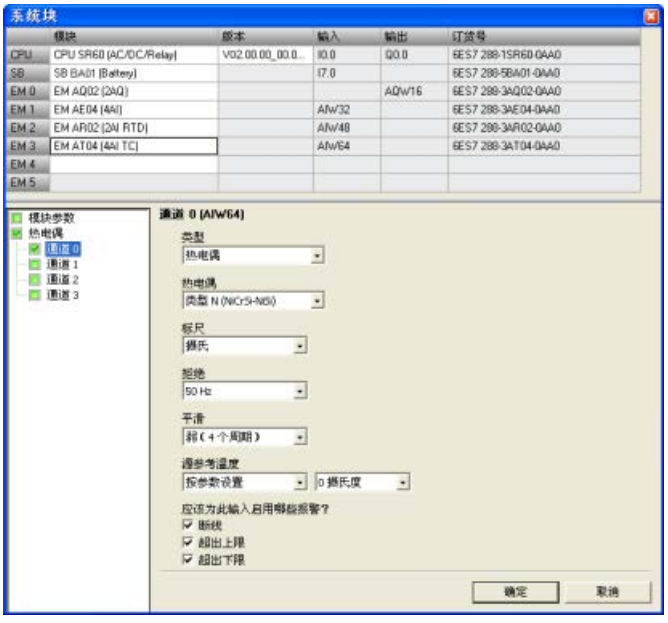
可针对所选 RTD 模块的选定通道，选择启用或禁用下列报警：

- 断路
- 超出上限
- 超出下限
- 用户电源（在系统块的“模块参数”(Module Parameters) 节点下组态，参见下图。）



6.1.13 组态 TC 模拟量输入

在“系统块”(System Block) (页 135) 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

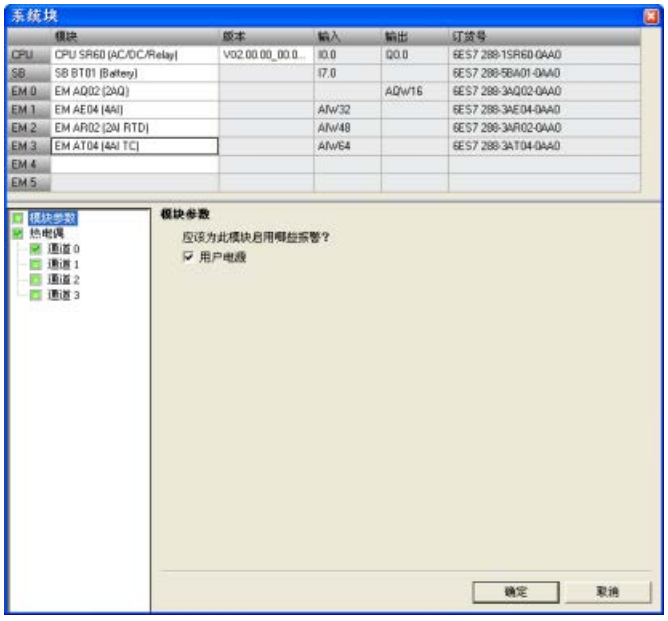
选择以下任一选项，组态各 TC 模拟量输入模块通道的源参考温度：

- 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
- 用户电源（在系统块的“模块参数”(Module Parameters) 节点下组态，参见下图。）



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。

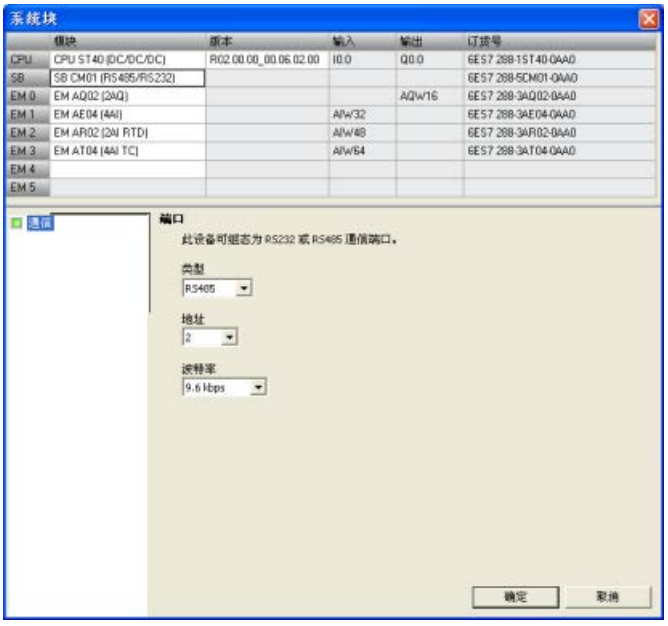
冷端补偿便是用于对连接器热电偶进行补偿。热电偶表是基于参比端温度（通常是零摄氏度）得来的。冷端补偿用于将连接器温度修正为零摄氏度。冷端补偿可消除连接器热电偶增加的电压。模块的温度在内部测量，然后转换为数值并添加到传感器换算中。之后是使用热电偶表对修正后的传感器换算值进行线性化。

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。

6.1.14 组态 RS485/RS232 CM01 通信信号板

在“系统块”(System Block) (页 135) 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。



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。

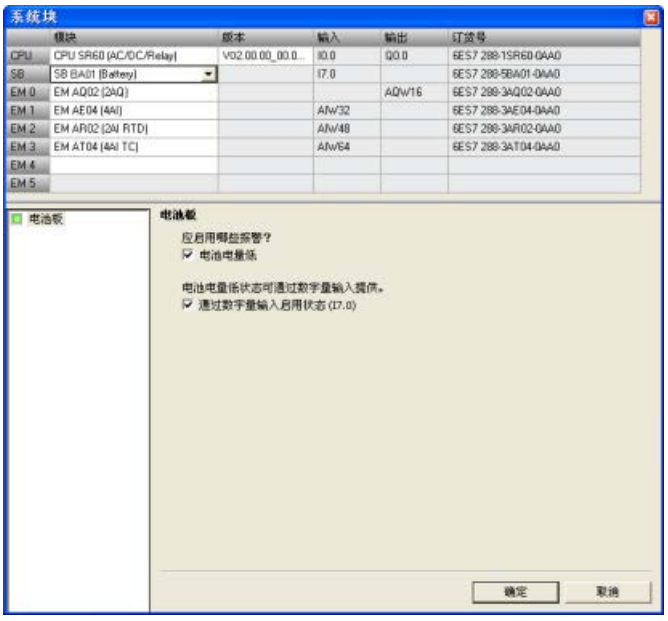
波特率

从下拉列表中选择所需数据波特率：

- 9.6 kbps
- 19.2 kbps
- 187.5 kbps

6.1.15 组态 BA01 电池信号板

单击“系统块”(System Block) (页 135) 对话框中的 BA01 电池信号板节点，对在顶部选择的 BA01 电池信号板的相关选项进行组态。



启用不良诊断报警

单击“启用不良诊断报警”(Enable bad diagnostic alarm) 复选框，当电池出现故障时便触发报警。

启用数字量输入的状态

单击“启用数字量输入的状态”(Enable status in digital input)，启用数字量输入监视信号板的状态。

电池 (BA01) 信号板的相关操作

电池信号板上有一个红色 LED，可为用户提供有关电池健康状况的视觉指示。LED 亮表示电池电量不足。

无论系统块是否包含信号板的组态，CPU 都会自动执行 RTC 交换、电池测试和电池健康状况 LED 操作。

借助电池信号板系统块组态中的相关选项，用户可以选择以诊断报警的方式报告电池电量不足，和/或在组态的设备映像寄存器输入字节的 LSB（例如，I7.0）位报告电池状态（1= 电池电量不足、0= 电池电量充足）。用户必须选择系统块组态中的电池信号板，这样才可以使用附加电池健康状况报告选项。

6.1.16 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换值 0。如果已为数字量或模拟量输出定义不为 0 的替换值或选择“冻结”(Freeze)，则在删除系统块时需要注意，您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。
直至您下载新的程序块，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 -
   程序块、数据块、系统块或所有块，或选择“复位为出厂默认设置”(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 复位为出厂默认设置。 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，“复位为出厂默认设置”(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”都复位为初始值。

忘记 PLC 密码怎么办

如果忘记 PLC 密码 (页 144)，必须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 168) 清除 PLC 存储器。

⚠警告

将存储卡插入 CPU

在 RUN 模式下将存储卡插入 CPU 导致 CPU 自动转换到 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。

要使用此卡清除 PLC，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 复位为出厂默认设置。 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 95)。

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。

6.1.17 创建复位为出厂默认存储卡。

可创建一个将 CPU 返回到出厂默认状态的存储卡。 如果要清除 CPU 的内容，可使用这张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。

6.2 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 19)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 267)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 19)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 294)。有关使用 PWM (页 625) 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 19)章节。

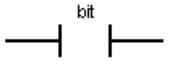
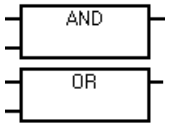
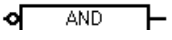
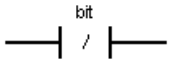
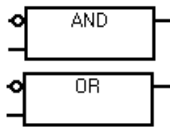
有关 CPU 中各运动功能的完整介绍，请参见开环运动控制 (页 625)的相关章节。

程序指令

7

7.1 位逻辑

7.1.1 标准输入

LAD	FBD	STL	说明
		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  ，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、A 和 O 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）、A（与非）和 O（或非）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。
		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。

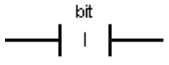
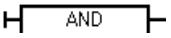
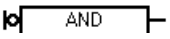
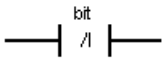
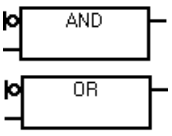
输入选项	光标放置	工具按钮	快捷键
添加输入	在功能框上		+
移除输入	在功能框和底部输入上		-

另请参见

位逻辑输入示例 (页 184)

逻辑堆栈概述 (页 174)

7.1.2 立即输入

LAD	FBD	STL	说明
		**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  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  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。
		**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 184)

逻辑堆栈概述 (页 174)

7.1.3 逻辑堆栈概述

STEP 7-Micro/WIN SMART 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。

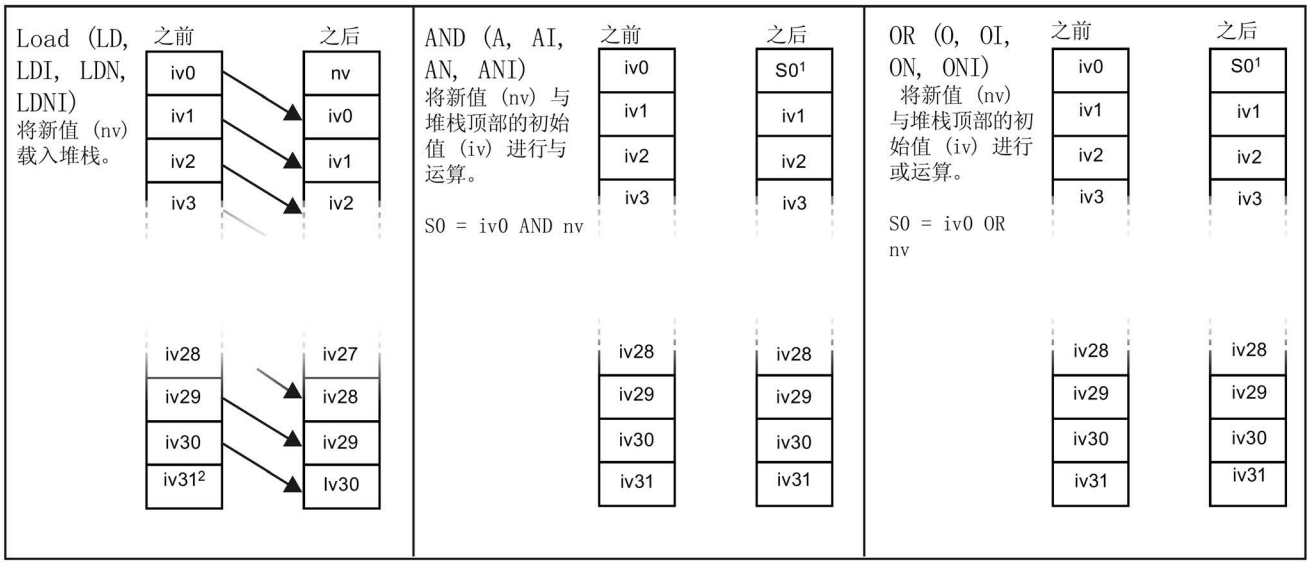
对于 LAD 和 FBD 编辑，会自动生成 STL 逻辑堆栈指令，并且程序员不需要使用逻辑堆栈指令。

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	说明
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。

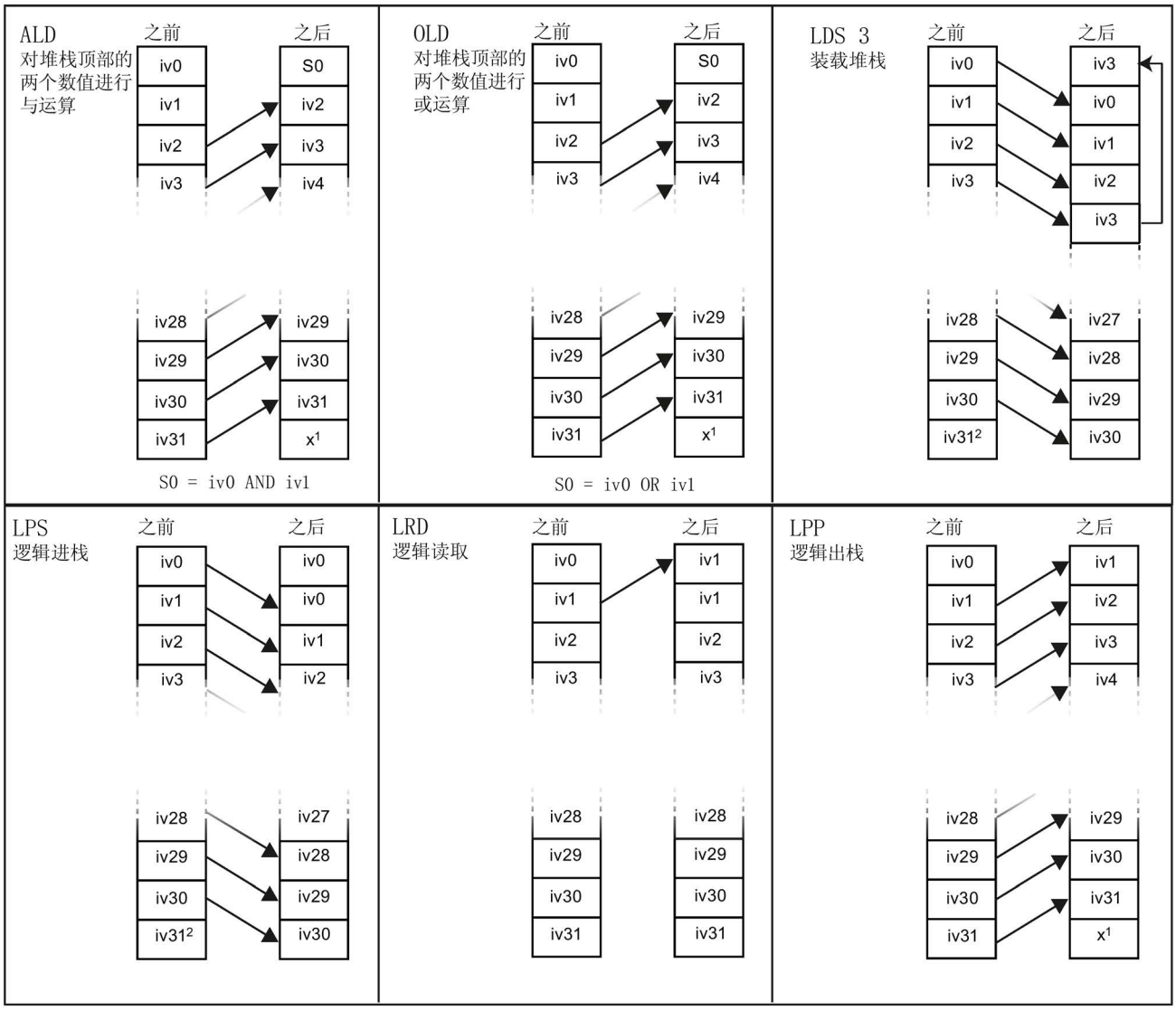
7.1.4 STL 逻辑堆栈指令

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）

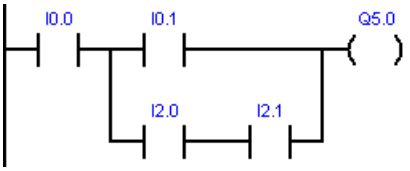
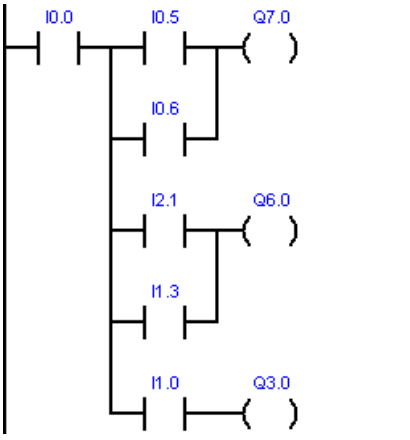
如下图所示，CPU 使用逻辑堆栈来解决控制逻辑。
在这些示例中，“iv0”至“iv31”用于标识逻辑堆栈的初始值，“nv”用于标识指令提供的新值，“S0”用于标识存储在逻辑堆栈中的计算值。



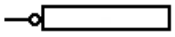
[1] 该值未知（可以是 a 0 或 a 1）。

[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

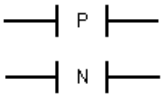
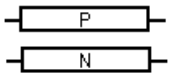
7.1.5 NOT

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 184)

7.1.6 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

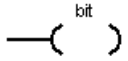
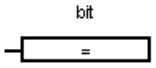
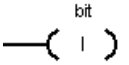
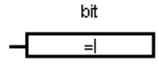
说明

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 184)

7.1.7 线圈: 输出和立即输出指令

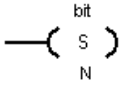
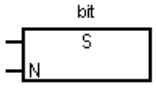
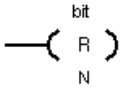
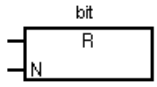
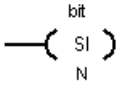
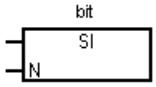
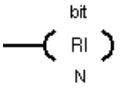
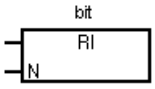
LAD	FBD	STL	说明
		`= bit`	该输出指令将输出位的新值写入过程映像寄存器。 **LAD** 和 **FBD**：该输出指令执行时，S7-200 将打开或关闭过程映像寄存器中的输出位。分配的位被设置为等于能流状态。 **STL**：堆栈顶值复制到分配的位。
		`=I bit`	该立即输出指令执行时，指令会将新值写入物理输出和相应的过程映像寄存器单元。 **LAD** 和 **FBD**：执行立即输出指令时，物理输出点（位）立即被设置为等于能流状态。“I”表示一个立即地址引用；新值将写入物理输出点和相应的过程映像寄存器地址。这不同于非立即地址引用仅将新值写入过程映像寄存器。 **STL**：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 186)

7.1.8 置位、复位、立即置位和立即复位功能

LAD	FBD	STL	说明
		**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。
		**R bit, 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。
		**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

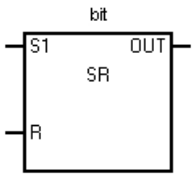
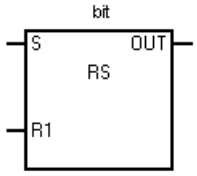
输入/输出	数据类型	操作数
位	BOOL	I、Q、V、M、SM、S、T、C、L
位（立即）	BOOL	Q
N	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、常数、*VD、*AC、*LD
输入 (LAD)	BOOL	能流
输入 (FBD)	BOOL	I、Q、V、M、SM、S、T、C、L、逻辑流

另请参见

位逻辑输入示例 (页 184)

位逻辑输出示例 (页 186)

7.1.9 置位和复位优先双稳态触发器

LAD/FBD [1]	说明
	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

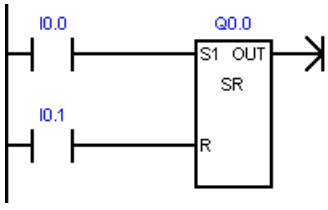
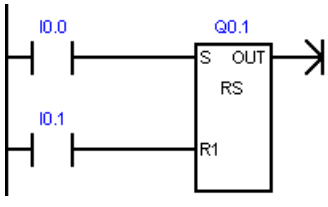
SR 真值表

S1	R	输出（位）
0	0	先前状态
0	1	0
1	0	1
1	1	1

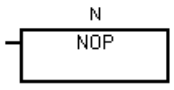
RS 真值表

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

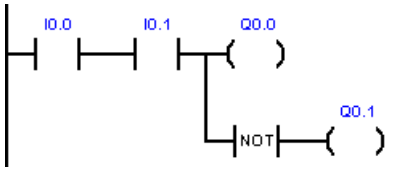
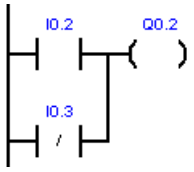
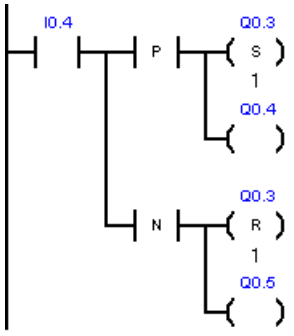
LAD	STL
	Network 1 LD I0.0 LD I0.1 NOT A Q0.0 OLD = Q0.0
	Network 2 LD I0.0 LD I0.1 NOT LPS A Q0.1 = Q0.1 LPP ALD O Q0.1 = Q0.1

7.1.10 NOP（空操作）指令

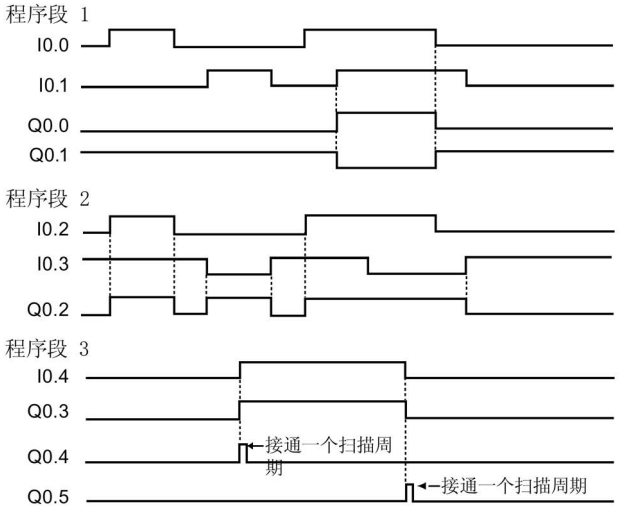
LAD	STL	说明
	NOP N	空操作 (NOP) 指令不影响用户程序的执行。 在 FBD 模式下，该指令不可用。 操作数 N 为 0 至 255 之间的数。

输入/输出	数据类型	操作数
N（LAD、STL）	BYTE	N： 常数（0 到 255）

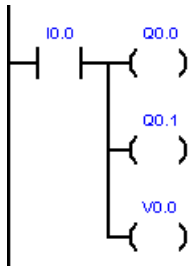
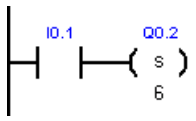
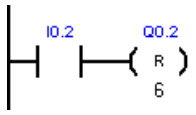
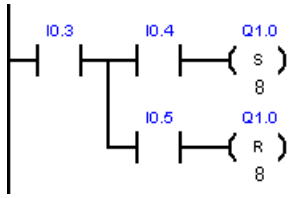
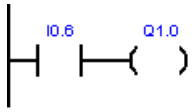
7.1.11 位逻辑输入示例

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

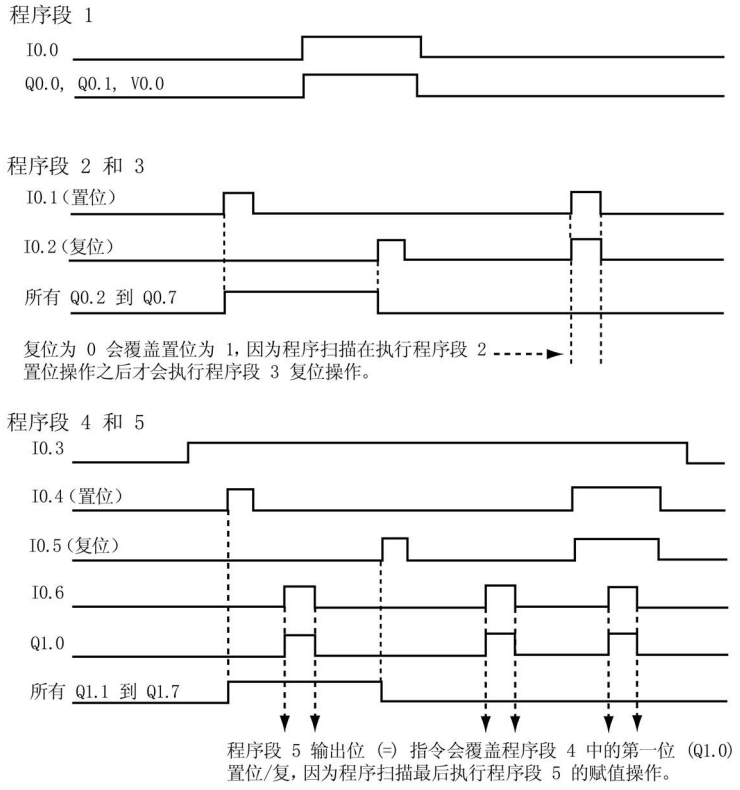
运行模式下的输入定时示例



7.1.12 位逻辑输出示例

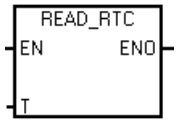
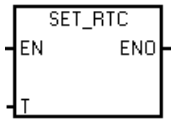
LAD		STL
	输出指令将位值分配给外部 I/O（I、Q）和内部存储器（M、SM、T、C、V、S、L）。	Network 1 LD I0.0 = Q0.0 = Q0.1 = V0.0
	将一组连续的 6 个位设置为值 1。指定起始位地址和要置位的位数。第一个位 (Q0.2) 的值为 1 时，“置位”指令的程序状态指示器为 ON。	Network 2 LD I0.1 S Q0.2, 6
	将一组连续的 6 个位复位为值 0。指定起始位地址和要复位的位数。第一个位 (Q0.2) 的值为 0 时，“复位”指令的程序状态指示器为 ON。	Network 3 LD I0.2 R Q0.2, 6
	成组置位和复位 8 个输出位（Q1.0 至 Q1.7）。	Network 4 LD I0.3 LPS A I0.4 S Q1.0, 8 LPP A I0.5 R Q1.0, 8
	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，Network 4 成组置位和复位八个输出位（Q1.0 至 Q1.7）。 在 RUN 模式下，Network 5 会改写 Q1.0 位值并控制 Network 4 中的置位/复位程序状态指示器。	Network 5 LD I0.6 = Q1.0

运行模式下的输出定时示例



7.2 时钟

7.2.1 读取和设置实时时钟

LAD/FBD	STL	说明
	**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。
	**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

不接受无效日期。 例如，如果您输入 2 月 30 日，则将发生非致命日时钟错误 (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，系统标志位 SM4.3 会置位，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 2000 年表示为 00。但使用年份值的用户程序必须考虑两位数的表示法。

2099 年之前的闰年均可正确处理。

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。

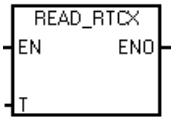
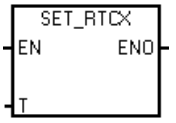
日期	时间	星期几
2000 年 1 月 1 日	00:00:00	星期六

说明

紧凑型 S7-200 SMART CPU 型号 CR40 和 CR60 不含 RTC（实时时钟）或超级电容

可借助 READ_RTC 和 SET_RTC 指令设置 CPU 型号 CR40 和 CR60 的年份、日期和时间值，但这些值会在下一次 CPU 断电再重新上电时丢失。上电时，日期和时间将初始化为 2000 年 1 月 1 日。

7.2.2 读取和设置扩展实时时钟

LAD/FBD	STL	说明
	**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。
	**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

不接受无效日期。 例如，如果您输入 2 月 30 日，则将发生非致命日时钟错误 (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，系统标志位 SM4.3 会置位，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 2000 年表示为 00。但使用年份值的用户程序必须考虑两位数的表示法。

2099 年之前的闰年均可正确处理。

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）

T 字节	说明	数据值
12	DST 开始星期	1 至 5（BCD 值）[5]
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 S7-200 SMART CPU 型号 CR40 和 CR60 不含 RTC（实时时钟）或超级电容

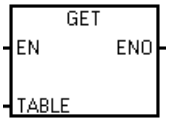
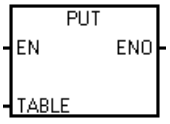
可借助 READ_RTC 和 SET_RTC 指令设置 CPU 型号 CR40 和 CR60 的年份、日期和时间值，但这些值会在下一次 CPU 断电再重新上电时丢失。上电时，日期和时间将初始化为 2000 年 1 月 1 日。

7.3 通信

7.3.1 GET 和 PUT（以太网）

GET 和 PUT 指令适用于通过以太网进行的 S7-200 SMART CPU 之间的通信：

表格 7- 1 GET 和 PUT 指令

LAD/FBD	STL	说明
	**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 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它设备的连接时，会需要额外的后台通信时间（参见“组态通信” (页 137)）。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 7- 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 7- 3　GET 和 PUT 指令 TABLE 参数的定义

字节偏移量	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
0	D [1]	A [2]	E [3]	0	错误代码			
1	远程 站 IP 地址 [4]							
2								
3								
4								
5	保留 = 0（必须设置为零）							
6	保留 = 0（必须设置为零）							
7	指向远程站（此 CPU） 中数据区的 指针 （I、Q、M、V 或 DB1）[5]							
8								
9								
10								

字节偏移量	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
11	数据长度 [6]							
12	指向本地站（此 CPU）中数据区的指针（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）。

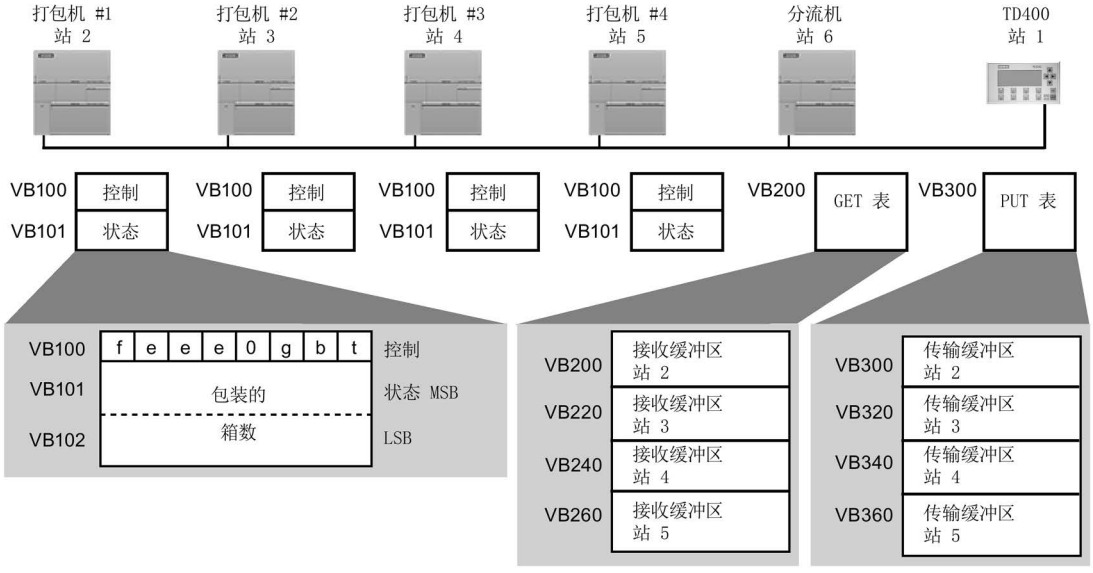
7 指向本地站中数据区的指针：指向本地站（此 CPU）中将要访问的数据的间接指针。

表格 7- 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

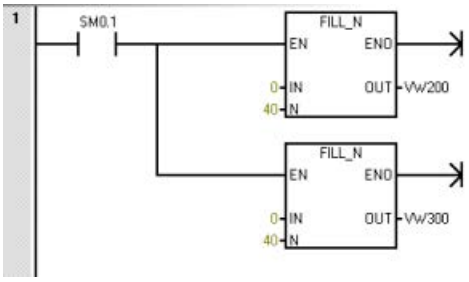
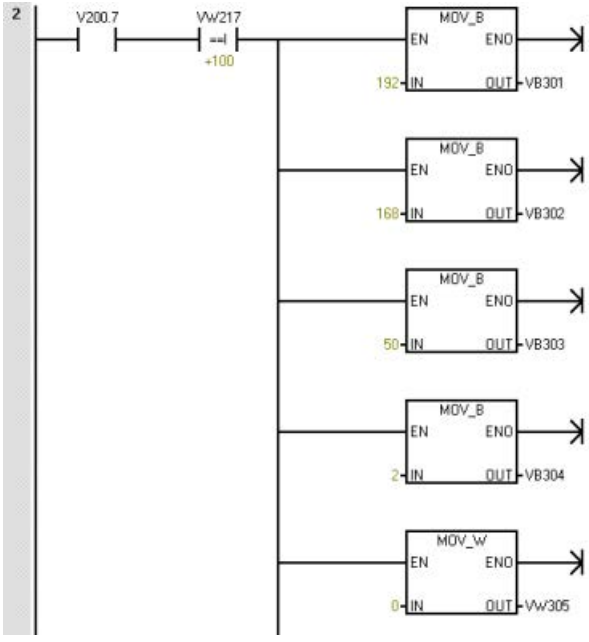
下图显示访问站 2 中数据所用的 GET 表格 (VB200) 和 PUT 表格 (VB300)。分流 CPU 使用 GET 指令连续读取来自每个装箱机的控制和状态信息。每当打包机装完 100 箱时，分流机都会注意到并通过 PUT 指令发送相应消息清除状态字。

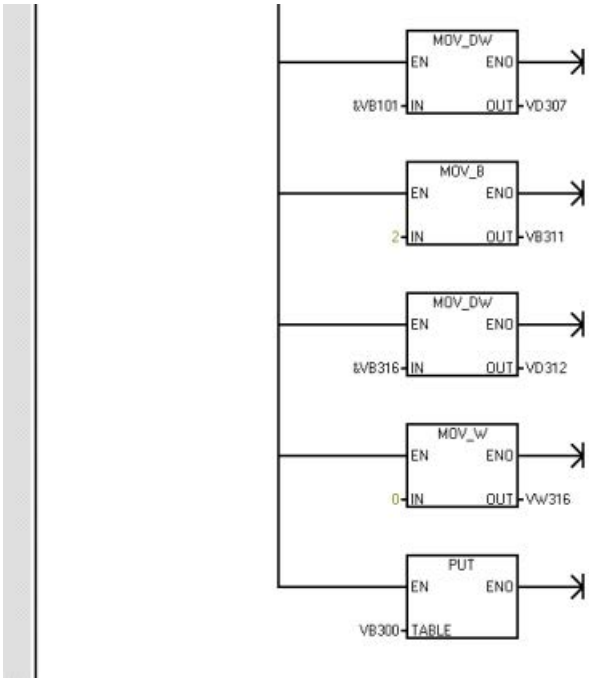
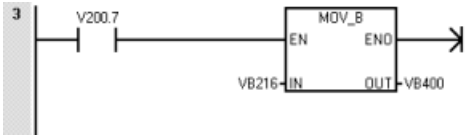
表格 7- 5　用于读取和清除打包机 1 计数的 GET 和 PUT 指令缓冲区

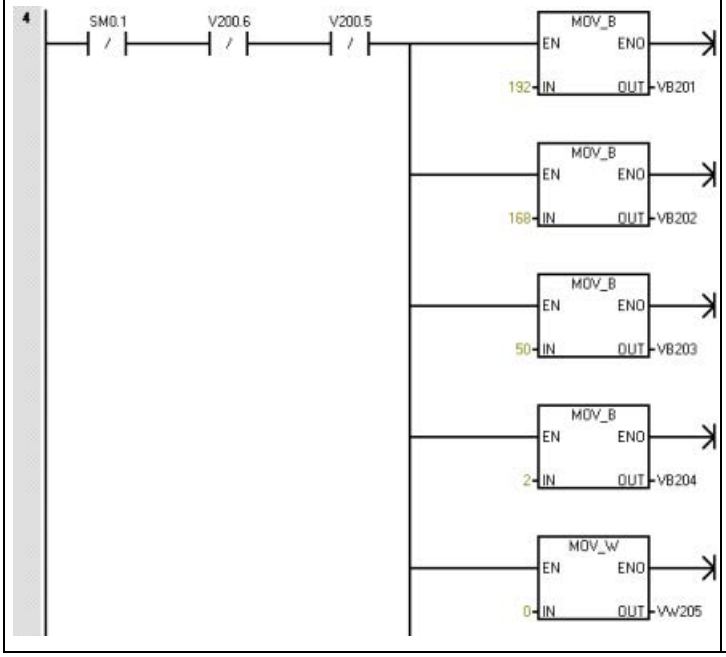
GET_TABLE 缓冲区	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0	PUT_TABLE 缓冲区	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
VB200	D	A	E	0	错误代码				VB300	D	A	E	0	错误代码			
VB201	远程站 IP 地址 = 192.								VB301	远程站 IP 地址 = 192.							
VB202	168.								VB302	168.							
VB203	50.								VB303	50.							
VB204	2								VB304	2							
VB205	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	指向远程站								VB307	指向远程站							
VB208	中数据区的								VB308	中数据区的							
VB209	指针 =								VB309	指针 =							
VB210	(&VB100)								VB310	(&VB101)							
VB211	数据长度 = 3 个字节								VB311	数据长度 = 2 个字节							
VB212	指向本地站（此 CPU）								VB312	指向本地站（此 CPU）							
VB213	中数据区的								VB313	中数据区的							
VB214	指针 =								VB314	指针 =							
VB215	(&VB216)								VB315	(&VB316)							
VB216	控制								VB316	0							
VB217	状态 MSB								VB317	0							
VB218	状态 LSB																

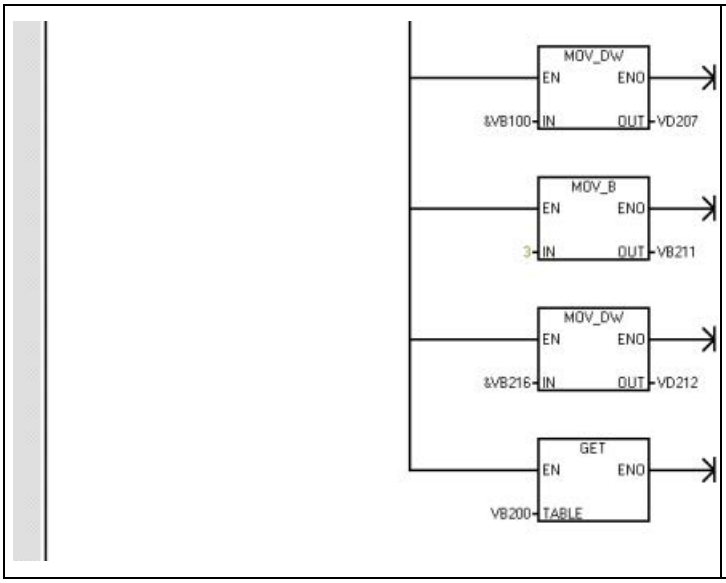
在本示例中，数据紧随 PUT 和 GET 表的变化而变化。由于表中本地站的指针指向该数据，因此可将该数据置于 CPU 存储器中的任意位置（例如，VB212 - VB215）。

表格 7- 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, VB305** **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

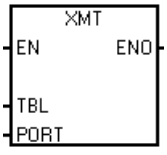
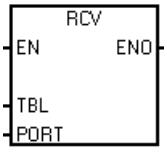
		
	Network 3 LD V200.7 MOVB VB216, VB400	当 GET“完成”位置位时， 保存打包机 1 中的控制数据。

	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, VB205
	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。

7.3.2 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 和其它设备之间进行通信。每个 S7-200 SMART CPU 都提供集成的 RS485 端口（端口 0）。标准 CPU 额外支持可选 CM01 信号板 (SB) RS232/RS485 端口（端口 1）。必须在用户程序中执行通信协议。

LAD/FBD	STL	说明
	**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 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） • 0090H 端口号无效 • 接收参数错误置位 SM86.6 或 SM186.6 • 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

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 或 1 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）

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。处于自由端口模式时，通信协议完全由用户程序控制。SMB30 和 SMB130 用于选择波特率和奇偶校验。

向两个物理端口分配两个特殊存储器字节：

- 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

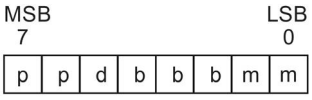
CPU 处于 STOP 模式时，会禁用自由端口模式，并会重新建立正常通信（例如，HMI 设备访问）。

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。其它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，才能进行自由端口通信。要启用自由端口模式，请在 SMB30（端口 0）或 SMB130（端口 1）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

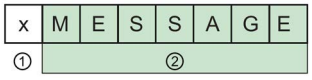
SMB30 和 SMB130 分别组态通信端口 0 和 1
以进行自由端口操作，并提供波特率、奇偶校验和数据位数的选择。下图显示了自由端口控制字节。对于所有组态，都生成一个停止位。



pp	奇偶校验选择		d	每个字符的数据位数	
	00 = 01 = 10 = 11 =	无奇偶校验 偶校验 无奇偶校验 奇校验		0 = 1 =	每个字符 8 位 每个字符 7 位
bbb	自由端口波特率		mm	协议选择	
	000 = 001 = 010 = 011 = 100 = 101 = 110 = 111 =	38400 19200 9600 4800 2400 1200 115200 57600		00 = 01 = 10 = 11 =	PPI 从站模式 自由端口模式 保留（默认为 PPI 从站模式） 保留（默认为 PPI 从站模式）

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。



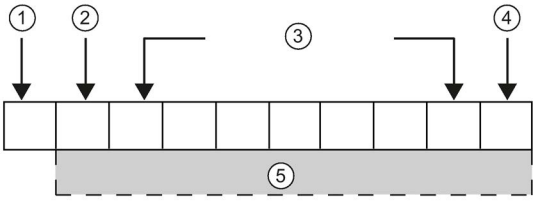
① 要发送的字节数
② 消息字符

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）。

您可以不使用中断，而通过监视 SM4.5（端口 0）或 SM4.6（端口
1）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它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 或 SM4.6 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）。

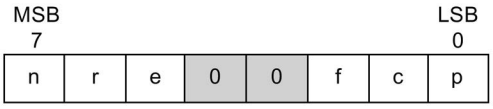
可以不使用中断，而通过监视 SMB86（端口 0）或 SMB186（端口
1）来接收消息。如果接收指令未激活或已终止，该字节不为零。正在接收时，该字节为零。

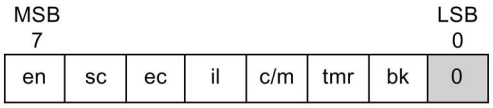
如下表所示，接收指令允许您选择消息开始和结束条件，对于端口 0 使用 SMB86 到 SMB94，对于端口 1 使用 SMB186 到 SMB194。

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。

接收缓冲区格式（SMB86 至 SMB94，以及 SMB186 至 SMB194）

端口 0	端口 1	说明
SMB86	SMB186	接收消息状态字节  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	说明
SMB87	SMB187	接收消息控制字节  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 或 SMB188。 1 = 使用 SMB88 或 SMB188 的值检测消息的起始。 ec： 0 = 忽略 SMB89 或 SMB189。 1 = 使用 SMB89 或 SMB189 的值检测消息的结束。 il： 0 = 忽略 SMB90 或 SMB190。 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 或 SMW192。 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	消息字符开始。
SMB89	SMB189	消息字符结束。
SMW90	SMW190	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	说明
SMW92	SMW192	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。
SMB94	SMB194	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 或 SMB187）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它设备的通信，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 或 SMW190
   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 或 SMW190
   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 = 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 或 SMB188
   中指定的起始字符时，启动消息。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 = 不相关，SMB88/SMB188 = 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它信息的协议，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 > 0，SMB88/SMB188 = 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 = 不相关，SMB88/SMB188 = 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 = 不相关，SMB88/SMB188 = 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 或 SMW190）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 = 0，SMB88/SMB188 = 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。如果未满足其它结束条件，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 = 0，SMB88/SMB188 = 不相关，c/m = 1，tmr = 1，SMW92 = 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。

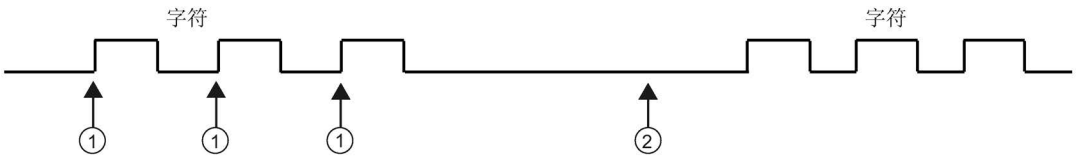
   设置：ec = 1，SMB89/SMB189 = 结束字符

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 或 SMW192 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 = 超时（毫秒）



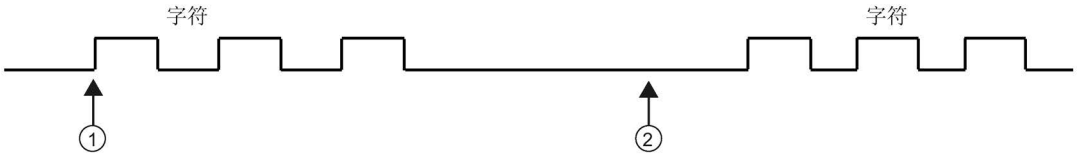
① 重新启动字符间定时器
② 字符间定时器时间到：终止消息并生成接收消息中断

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 或 SMW192 中指定的毫秒数后，消息定时器时间到。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 = 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 或 SMB194）。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 或 SMB130 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 或 SMB187 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

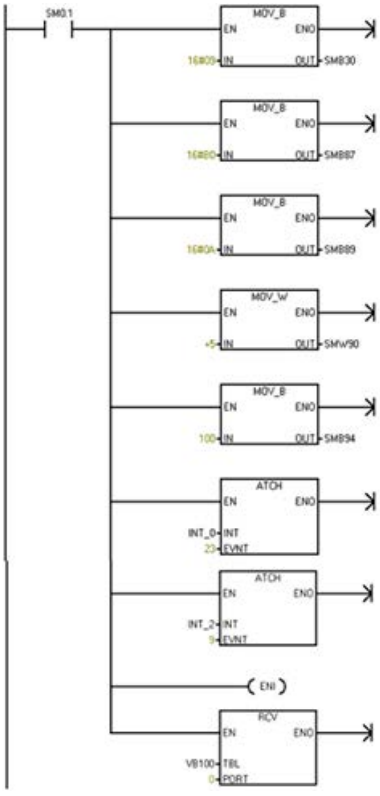
为了完全适应对各种协议的支持，您还可以使用字符中断控制来接收数据。接收每个字符时都会产生中断。执行连接到接收字符事件的中断例程之前，接收到的字符存入 SMB2，奇偶校验状态（若已启用）存入 SM3.0。SMB2 是自由端口接收字符缓冲区。自由端口模式下接收到的每一个字符都会存入这一位置，便于用户程序访问。SMB3 用于自由端口模式，包含一个奇偶校验错误位，如果在接收到的字符中检测到奇偶校验错误、组帧错误、超限错误或断开错误，该位将置位。保留该字节的所有其它位。可使用奇偶校验位丢弃消息或向该消息发送否定确认。

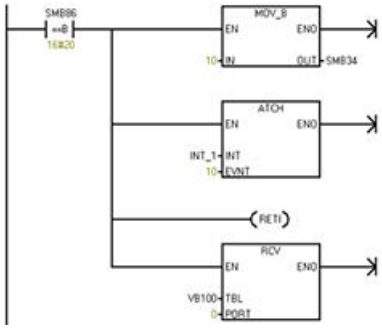
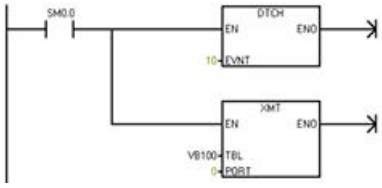
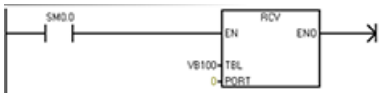
以较高波特率（38.4K 到 115.2K）使用字符中断时，中断之间的时间间隔会非常短。例如，波特率为 38.4K 时的字符中断为 260 微秒，57.6K 时为 173 微秒，115.2K 时为 86 微秒。确保中断例程足够短，以避免字符丢失，否则请使用接收指令。

说明

SMB2 和 SMB3 可供端口 0 和端口 1 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。

示例：发送和接收指令

<table>
<tr><td>MAIN</td><td>Network 1</td><td colspan="2">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。</td></tr>
<tr><td rowspan="11"></td><td rowspan="11"> 
</td><td>LD SM0.1
MOVB 16#09, SMB30</td><td>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。</td></tr>
<tr><td>MOVB 16#B0, SMB87</td><td>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。</td></tr>
<tr><td>MOVB 16#0A, SMB89</td><td>3.将消息结束字符设为十六进制 0A（换行）。</td></tr>
<tr><td>MOVW +5, SMW90</td><td>4.将空闲线超时设为 5 ms。</td></tr>
<tr><td>MOVB 100, SMB94</td><td>5.将最大字符数设为 100。</td></tr>
<tr><td>ATCH INT_0, 23</td><td>6.将中断 0
连接到接收完成事件。</td></tr>
<tr><td>ATCH INT_2, 9</td><td>7.将中断 2
连接到发送完成事件。</td></tr>
<tr><td>ENI</td><td>8.启用用户中断。</td></tr>
<tr><td>RCV VB100, 0</td><td>9.启用具有 VB100
缓冲区的接收功能框。</td></tr>
</table>

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.如果因其它原因完成接收，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。

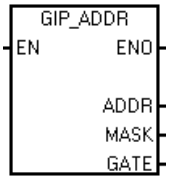
7.3.3 获取端口地址和设置端口地址（RS485/RS232 上的 PPI 协议）

LAD/FBD	STL	说明
	**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。
	**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上一地址（即通过系统块下载的地址）。

ENO = 0 时的非致命错误条件	受影响的 SM 位
• 0006H 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 或 1 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）

7.3.4 获取 IP 地址和设置 IP 地址（以太网）

LAD/FBD	STL	说明
	**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, 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
• 0006H 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
MASK	DWORD	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC
GATE	DWORD	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC

说明

- 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，不能通过其它方式更改"**(IP address data is fixed to the values below and cannot be changed by other means) 复选框。
  该组态复选框位于"系统块"(System Block) "通信"(Communication) 节点下的"以太网"(Ethernet) 字段中。
- IP 地址、子网掩码和网关值被写入永久性存储器。

7.3.5 开放式用户通信

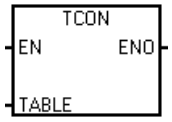
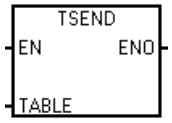
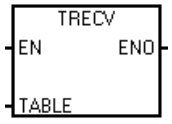
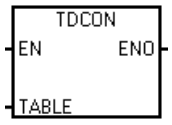
开放式用户通信 (OUC)
指令可使您的程序通过以太网与另一个支持以太网的设备进行通信。对方以太网设备可以是另一个 S7-200 SMART CPU 或是另一个支持 UDP、TCP、或 ISO-on-TCP 协议的第三方设备。您的程序对通信进行全方位的控制，包括选择协议、发起连接、发送数据、接收数据和终止连接。

7.3.5.1 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 7- 7 OUC 指令

LAD/FBD	STL	描述
	**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 table**	TSEND 用于将数据发送到另一个设备。
	**TRECV table**	TRECV 用于检索通过现有通信连接接收到的数据。
	**TDCON table**	TCON 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” (页 137)。

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” (页 517)。

表格 7- 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 234)。有关端口数限制及更多信息，请参见“端口和 TSAP” (页 435)：

- 状态：表的第一个字节将操作的状态返回给用户。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 234)。
- 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 7- 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								

字节偏移量	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
9	远程端口 [4]							
10								
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 7- 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								
26	本地 TSAP [5] 2 到 16 字符的字符串（3 到 17 个字节）							
到								
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。

TECV

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 TECV 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 234)。有关端口数限制及更多信息，请参见“端口和 TSAP” (页 435)：

- 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 234)。
- 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 7- 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								
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 7- 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 IP 地址 [4]							
11								
12								
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 234)。有关端口数限制及更多信息，请参见“端口和 TSAP” (页 435)：

- 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 234)。
- 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 7- 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								

7.3.5.2 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

错误代码	描述	TCON	TSEND	TRECV	TDCON
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			
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

7.4 比较

7.4.1 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

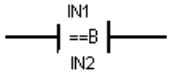
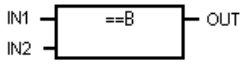
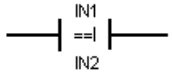
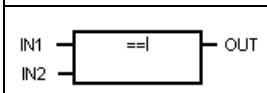
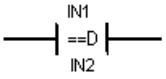
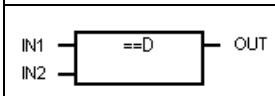
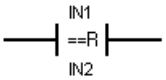
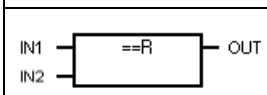
有六种比较类型可用：

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
 	**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
 	**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

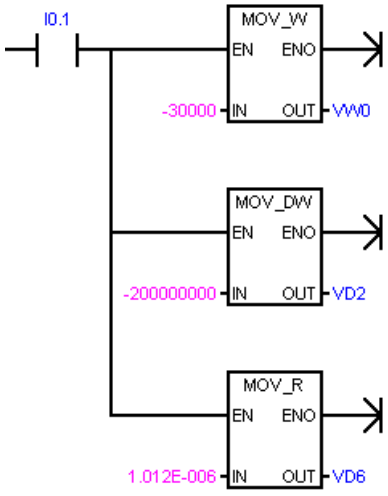
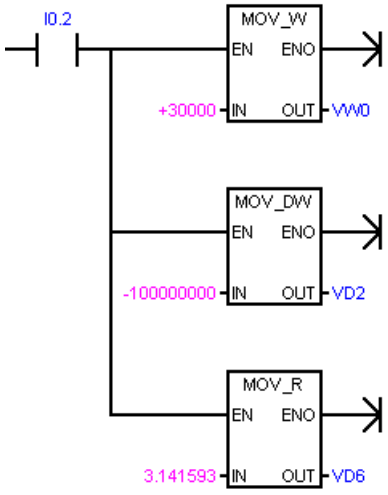
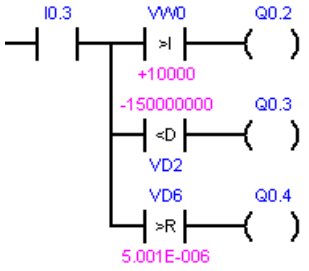
- 遇到非法间接地址（任意比较指令）
- 比较实数指令遇到非法实数（例如 NaN）

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 (页 82)

7.4.2 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)。

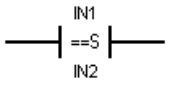
对于 **STL**： 比较结果为 TRUE 时，比较指令可装载 1、将 1
与逻辑栈顶中的值进行“与”运算或者“或”运算。

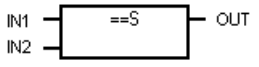
可以在两个变量或一个常数和一个变量之间进行比较。
如果比较中使用了常数，则它必须为顶部参数（LAD 触点/ FBD 功能框）或第一参数 (STL)。

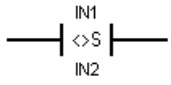
在程序编辑器中，常数字符串参数赋值必须以双引号字符开始和结束。
常数字符串条目的最大长度是 126 个字符（字节）。

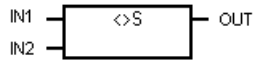
相反，变量字符串由初始长度字节的字节地址引用，字符字节存储在下一个字节地址处。
变量字符串的最大长度为 254
个字符（字节），并且可在数据块编辑器进行初始化（前后带双引号字符）。

LAD 触点 FBD 功能框	STL	说明

	`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。

	`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。

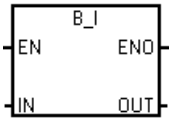
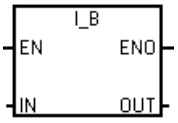
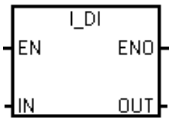
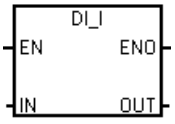
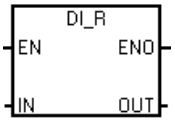
另请参见常数 (页 82)

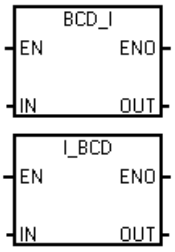
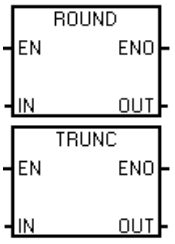
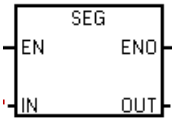
7.5 转换

7.5.1 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	说明
	**BTI IN, OUT**	**字符转换为整数：** 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。
	**ITB IN, OUT**	**整数转换为字节：** 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。
	**ITD IN, OUT**	**整数转换为双精度整数：** 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。
	**DTI IN, OUT**	**双精度整数转换为整数：** 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。
	**DTR IN, OUT**	**双整数转换为实数：** 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。

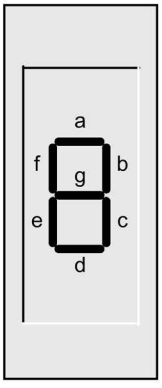
LAD/FBD	STL	说明
	**BCDI OUT** **IBCD OUT**	**BCD 转换为整数：** 将二进制编码的十进制 WORD 数据类型值 IN 转换为整数 WORD 数据类型的值，并将结果加载至分配给 OUT 的地址中。IN 的有效范围为 0 到 9999 的 BCD 码。 **整数码转换为 BCD：** 将输入整数 WORD 数据类型值 IN 转换为二进制编码的十进制 WORD 数据类型，并将结果加载至分配给 OUT 的地址中。IN 的有效范围为 0 到 9999 的整数。 对于 STL，IN 和 OUT 参数使用同一地址。
	**ROUND IN, OUT** **TRUNC 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 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。

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

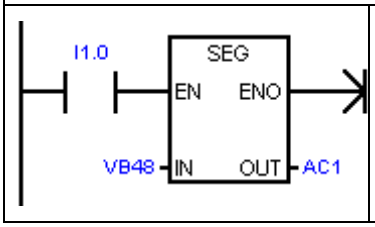
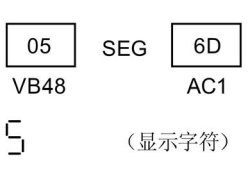
七段显示器的编码

(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

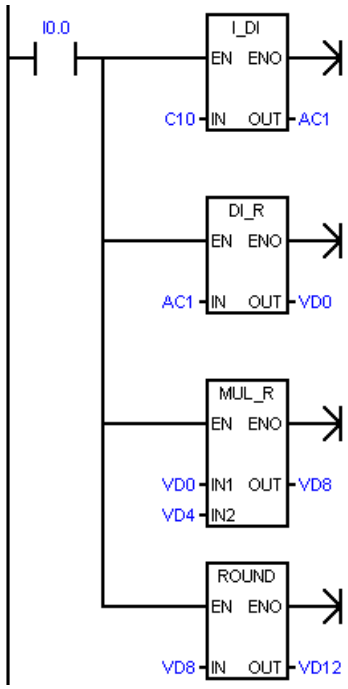
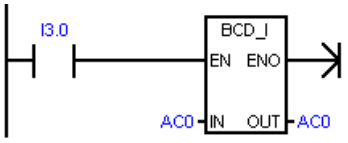


(IN) LSD	分段显示	(OUT) - gfe dcba
8	8	0111 1111
9	9	0110 0111
A	A	0111 0111
B	b	0111 1100
C	C	0011 1001
D	d	0101 1110
E	E	0111 1001
F	F	0111 0001

示例：使用 SEG 在七段显示屏上显示数值 5

LAD		STL
		Network 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）。	**Network 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）。	**Network 2** **LD I0.3** **BCDI AC0**

另请参见

分配指令的常数值

参见

分配指令的常数值 (页 82)

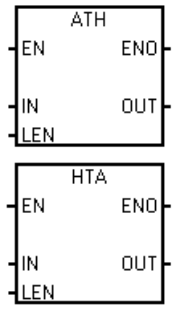
7.5.2 ASCII 字符数组转换

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

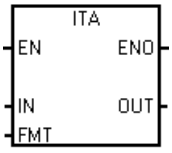
输入/输出	数据类型	操作数
IN, OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC
LEN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant

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 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。

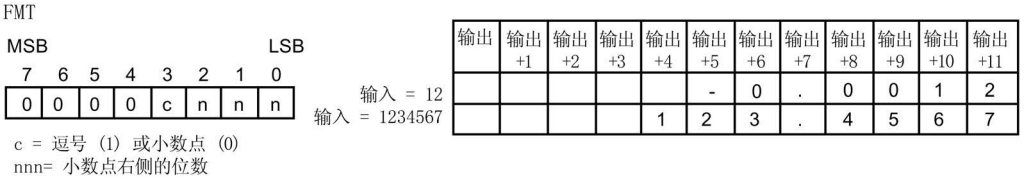
ENO = 0 时的非致命错误条件	受影响的 SM 位
• 0006H 无效间接地址 • 0091H 操作数超出范围 • FMT 字节的 4 个最高有效位的 FMT 位不为零 • nnn > 5	• 无

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。

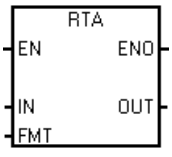
下图给出了一个数值作为示例，其格式为使用小数点 (c=0)，小数点右侧有四位 (nnn=100)。

双整数转换为 ASCII (DTA) 指令的 FMT 操作数



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
					-	0	.	0	0	1	2
				1	2	3	.	4	5	6	7

实数转换为 ASCII

LAD/FBD	STL	说明
	**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

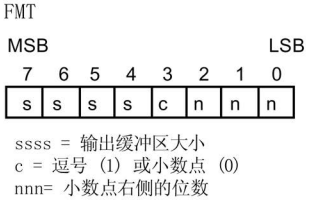
得出的 ASCII 字符数（或长度）就是输出缓冲区的大小，它的值在 3 到 15 个字节或字符之间。

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。

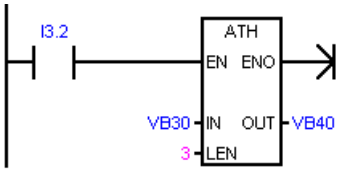
下图给出了一个数值作为示例，其格式为使用小数点 (c=0)、小数点右侧有一位 (nnn=001)、缓冲区的大小为六个字节 (ssss=0110)。

实数转换为 ASCII (RTA) 指令的 FMT 操作数



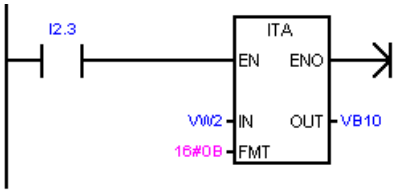
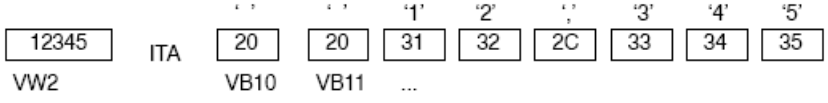
	输出	输出 +1	输出 +2	输出 +3	输出 +4	输出 +5
in = 1234.5	1	2	3	4	.	5
in = -0.0004				0	.	0
in = -3.67526			-	3	.	7
in = 1.95				2	.	0

示例：ASCII 转换为十六进制

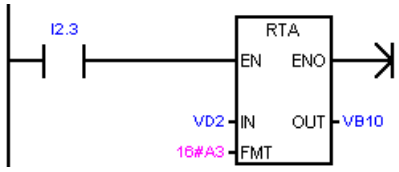
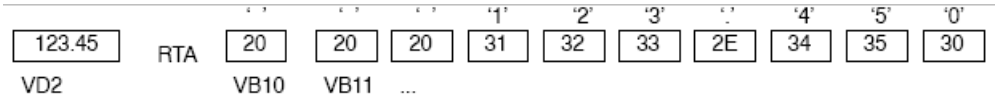
LAD	STL
	**Network 1** **LD I3.2** **ATH VB30, VB40, 3**
	

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 (页 82)

7.5.3 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。

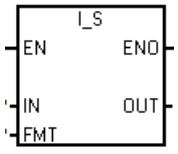
下图所示为存储器中以变量形式存储的 STRING 数据类型。字符串的长度可以是 0 到 254 个字符。变量字符串的最大存储要求为 255 个字节（长度字节加上 254 个字符）。

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
	`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

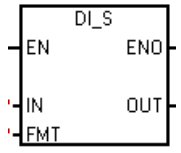
FMT

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

输入 = 12
输入 = 1234567

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
12							0	.	0	0	1	2
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

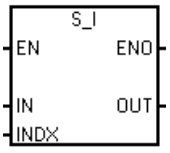
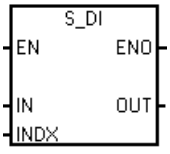
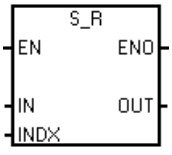
ssss = 输出字符串的长度
c = 逗号 (1) 或小数点 (0)
nnn= 小数点右侧的位数

	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 (页 82)

7.5.4 ASCII 子字符串转换为数值

LAD/FBD	STL	说明
	**STI IN, INDX, OUT**	ASCII 子字符串转换为整数值
	**STD IN, INDX, OUT**	ASCII 子字符串转换为双整数值
	**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

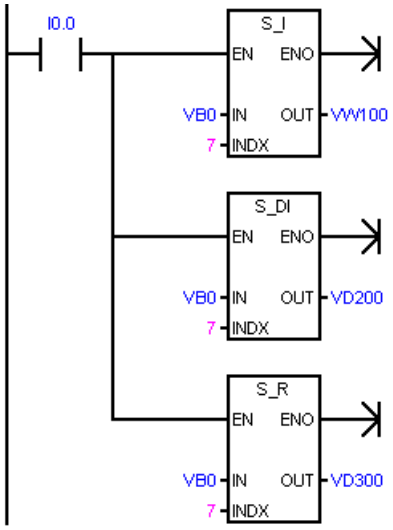
输入字符串	输出实数
'123'	123.0
'-00456'	-456.0
'123.45'	123.45
'+2345'	2345.0
'00.000000123'	0.000000123

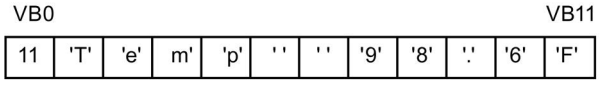
有效输入字符串

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		STL

	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。
		`Network 1`
		`LD I0.0`
		`STI VB0, 7, VW100`
		`STD VB0, 7, VD200`
		`STR VB0, 7, VD300`

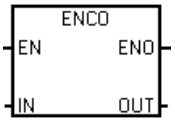
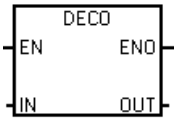


执行该程序段后：
VW100（整数）= 98
VD200（双精度整数）= 98
VD300（实际）= 98.6

另请参见

分配指令的常数值 (页 82)

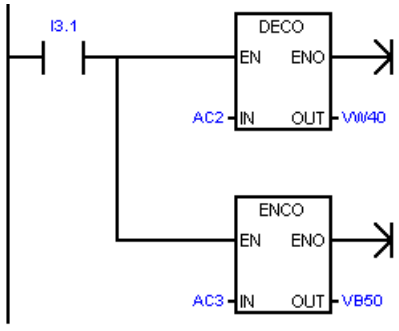
7.5.5 编码和解码

LAD/FBD	STL	说明
	**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 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

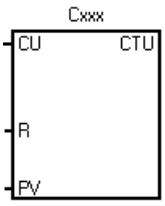
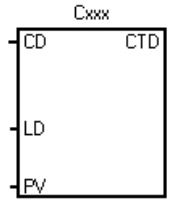
输入/输出	数据类型	操作数
IN	WORD (ENCO)	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*LD、*AC、常数
	BYTE (DECO)	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数
OUT	BYTE (ENCO)	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC
	WORD (DECO)	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AQW、*VD、*LD、*AC

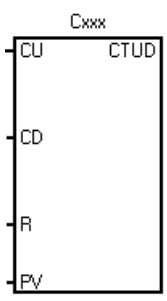
示例：编码和解码

<table>
<tr><th>LAD</th><th></th><th>STL</th></tr>
<tr>
<td> 
</td>
<td>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。</td>
<td>Network 1
LD I3.1
DECO AC2, VW40
ENCO AC3, VB50</td>
</tr>
<tr>
<td colspan="3"> 
</td>
</tr>
</table>

7.6 计数器

7.6.1 计数器指令

LAD/FBD	STL	说明
	**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

LAD/FBD	STL	说明
	**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，加计数输入处的下一上升沿导致当前计数值变为最小值 -32,768。达到最小值 -32,768 时，减计数输入处的下一上升沿导致当前计数值变为最大值 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 C255）
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。

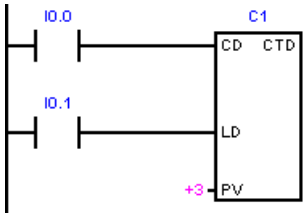
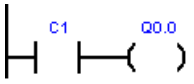
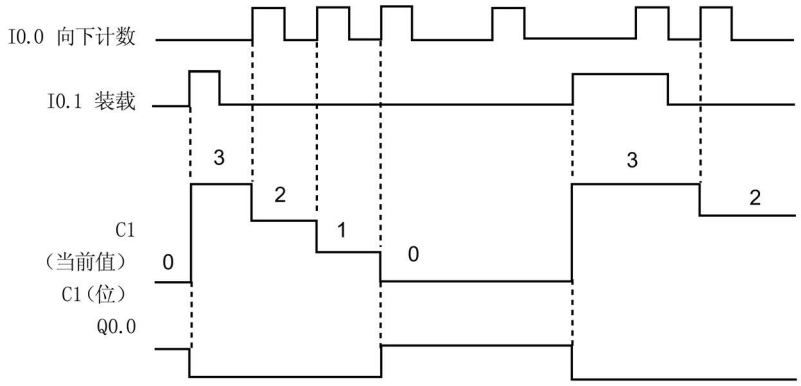
另请参见组态保持范围 - 系统块组态

计数器操作

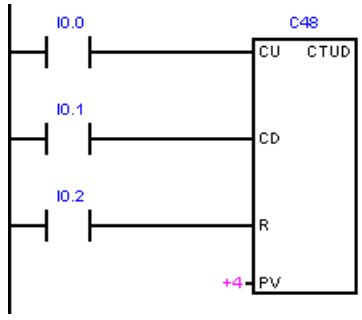
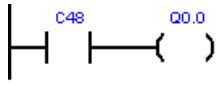
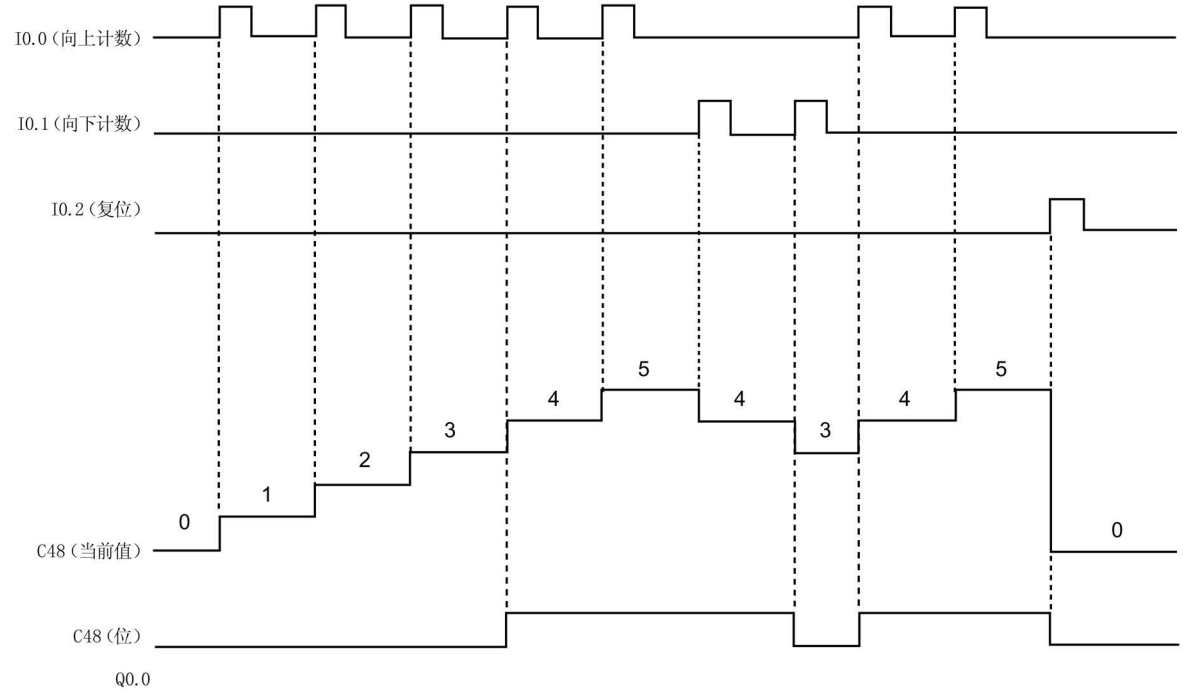
类型	操作	计数器位	上电循环/首次扫描
CTU	• CU 增加当前值。 • 当前值持续增加，直至达到 32,767。	以下情况下，计数器位接通：当前值 >= 预设值	• 计数器位关断。 • 当前值可保留 [1]
CTD	• CD 减少当前值，直至当前值达到 0。	以下情况下，计数器位接通：当前值 = 0	• 计数器位关断。 • 当前值可保留 [1]
CTUD	• CU 增加当前值。 • CD 减少当前值。 • 当前值持续增加或减少，直至计数器复位。	以下情况下，计数器位接通：当前值 >= 预设值	• 计数器位关断。 • 当前值可保留 [1]

[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

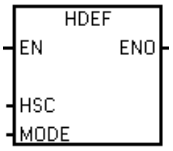
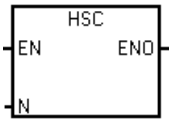
<table>
<tr><th>LAD</th><th></th><th>STL</th></tr>
<tr><td> 
</td><td>减计数器 C1 当前值从 3 计数至 0
当 I0.1 关断时，I0.0 的上升沿会减少 C1 当前值
I0.1 接通会装载减计数预设值 3</td><td>Network 1
LD I0.0
LD I0.1
CTD C1, +3</td></tr>
<tr><td> 
</td><td>当计数器 C1 当前值 = 0 时，C1 位接通</td><td>Network 2
LD C1
= Q0.0</td></tr>
<tr><td colspan="3">时序图
 
</td></tr>
</table>

CTUD 加计数/减计数示例

<table>
<tr><th colspan="2">LAD</th><th>STL</th></tr>
<tr><td> 
</td><td>I0.0 加计数
I0.1 减计数
I0.2 将当前值复位为 0</td><td>Network 1
LD I0.0
LD I0.1
LD I0.2
CTUD C48, +4</td></tr>
<tr><td> 
</td><td>当前值 >= 4
时，加/减计数计数器 C48 接通
C48 位</td><td>Network 2
LD C48
= Q0.0</td></tr>
<tr><td colspan="3">时序图
 
</td></tr>
</table>

7.6.2 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	说明
	**HDEF HSC, MODE**	高速计数器定义指令 (HDEF) 选择特定高速计数器 (HSC0-3) 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。 必须为多达四个激活的高速计数器各使用一条高速计数器定义指令。
	**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 或 3）
MODE	BYTE	模式编号常数：八种可能的模式（0、1、3、4、6、7、9 或 10）
N	WORD	HSC 编号常数（0、1、2 或 3）

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

所有高速计算器的运行方式与相同操作模式一样，但对于每一个 HSC 编号来说，并不支持每一种模式。HSC 输入连接（时钟、方向和复位）必须使用 CPU 的集成输入通道，如下表所示。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 271)”。

	时钟 A	Dir/时钟 B	复位	单相最大时钟/输入速率	双相/AB 正交相最大时钟/输入速率
HSC0	I0.0	I0.1	I0.4	200 kHz（S 型号 CPU）[1] 100 kHz（C 型号 CPU）[2]	S 型号 CPU： 100 kHz = 最大 1 倍计数速率 400 kHz = 最大 4 倍计数速率 C 型号 CPU： 50 kHz = 最大 1 倍计数速率 200 kHz = 最大 4 倍计数速率
HSC1	I0.1			200 kHz（S 型号 CPU） 100 kHz（C 型号 CPU）	
HSC2	I0.2	I0.3	I0.5	200 kHz（S 型号 CPU） 100 kHz（C 型号 CPU）	S 型号 CPU： 100 kHz = 最大 1 倍计数速率 400 kHz = 最大 4 倍计数速率 C 型号 CPU： 50 kHz = 最大 1 倍计数速率 200 kHz = 最大 4 倍计数速率
HSC3	I0.3			200 kHz（S 型号 CPU） 100 kHz（C 型号 CPU）	

[1] S 型号 CPU：SR20、ST20、SR30、ST30、SR40、ST40、SR60、ST60

[2] C 型号 CPU：CR40、CR60

HSC 计数模式支持

- 共可使用四个 HSC 设备（HSC0、HSC1、HSC2 和 HSC3）
- HSC0 和 HSC2 支持八种计数模式（模式 0、1、3、4、6、7、9 和 10）
- HSC1 和 HSC3 只支持一种计数模式（模式 0）

可用的 HSC 计数器类型

- 具有内部方向控制功能的单相时钟计数器
  - 模式 0：
  - 模式 1：具有外部复位功能
- 具有外部方向控制功能的单相时钟计数器
  - 模式 3：
  - 模式 4：具有外部复位功能
- 具有 2 路时钟输入（加时钟和减时钟）的双相时钟计数器
  - 模式 6：
  - 模式 7：具有外部复位功能
- AB 正交相计数器
  - 模式 9：
  - 模式 10：具有外部复位功能

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。

另请参见

高速计数器编程示例 (页 273)

高速计数器初始化顺序示例 (页 286)

高速输入降噪 (页 271)

7.6.3 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、I0.1、I0.2 和 I0.3 时，所使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 S7-200 SMART CPU 中。在 HSC 通道对脉冲进行计数前应用输入滤波。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。这包括方向和复位输入。下表所示为各种输入滤波组态可检测到的最大输入频率。

输入滤波时间	可检测到的最大频率
0.2 µs	200 kHz（S 型号 CPU）[1] 100 kHz（C 型号 CPU）[2]
0.4 µs	200 kHz（S 型号 CPU） 100 kHz（C 型号 CPU）
0.8 µs	200 kHz（S 型号 CPU） 100 kHz（C 型号 CPU）
1.6 µs	200 kHz（S 型号 CPU） 100 kHz（C 型号 CPU）
3.2 µs	156 kHz（S 型号 CPU） 100 kHz（C 型号 CPU）
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

输入滤波时间	可检测到的最大频率
3.2 ms	156 Hz
6.4 ms	78 Hz
12.8 ms	39 Hz

[1] S 型号 CPU：SR20、ST20、SR30、ST30、SR40、ST40、SR60、ST60

[2] C 型号 CPU：CR40、CR60

- 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。信号将转换为低电平状态，但所需时间将取决于电路的输入电阻和电容。这种情况可能导致脉冲丢失。可通过将下拉电阻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 24V，因此电阻的额定功率必须为高功率。100 欧 5 瓦的电阻是一个合适的选择。

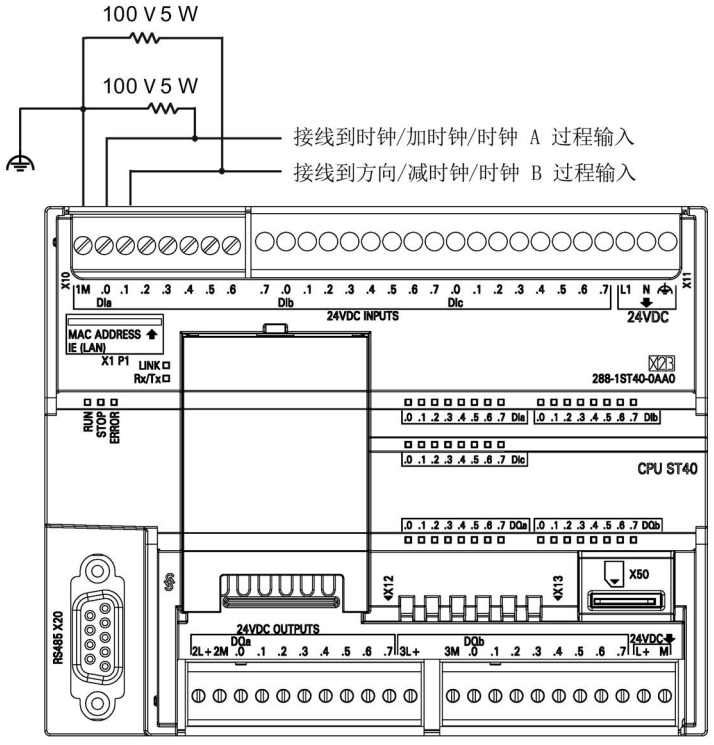


图 7-1　　为集电极开路 HSC 输入驱动接线下拉电阻

7.6.4 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。该向导可帮助您选择计数器类型/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 271)”。

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：在项目树的“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（起始值）。
- 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它用途。例如，如果在使用 I0.0 和 I0.4 的模式 1 下使用 HSC0，则 I0.1、I0.2 和 I0.3 可用于沿中断、HSC3 或运动控制输入。

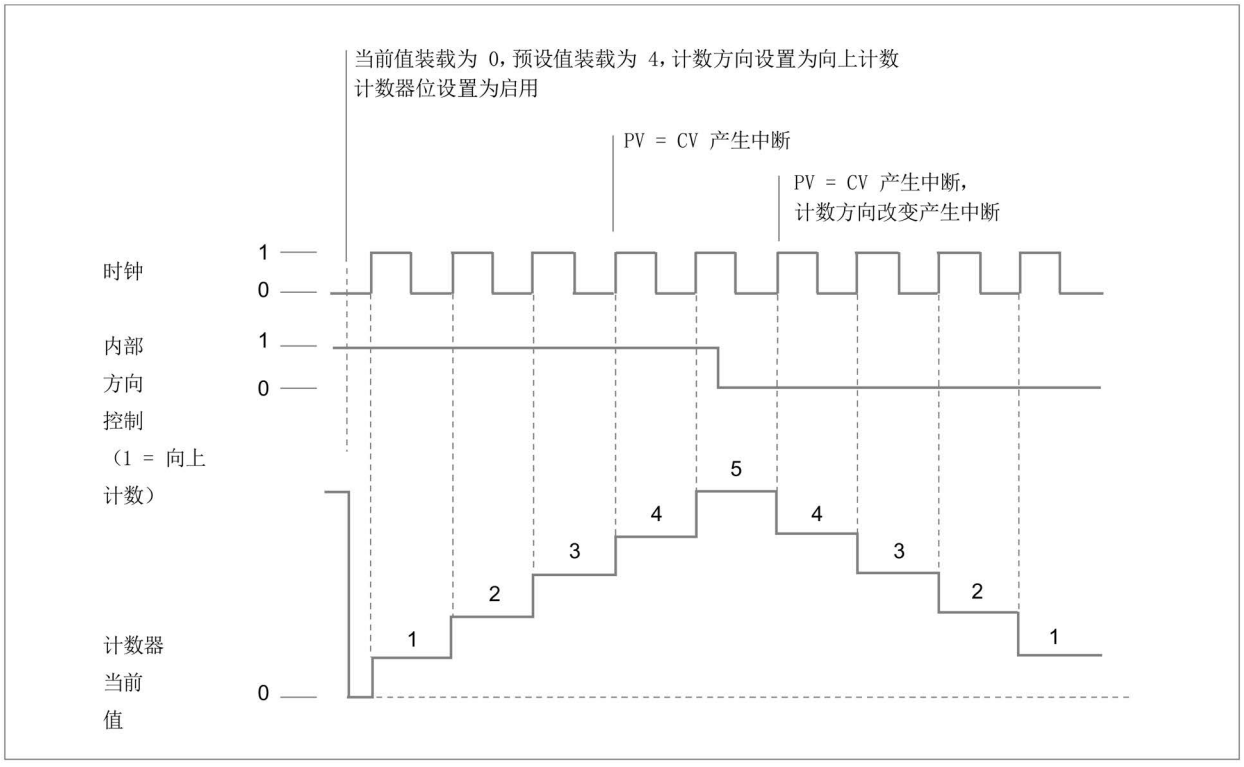
说明

HSC0 的所有计数模式始终使用 I0.0，而 HSC2 的所有模式始终使用 I0.2，因此使用这些计数器时，这些输入绝不会用于其它用途。

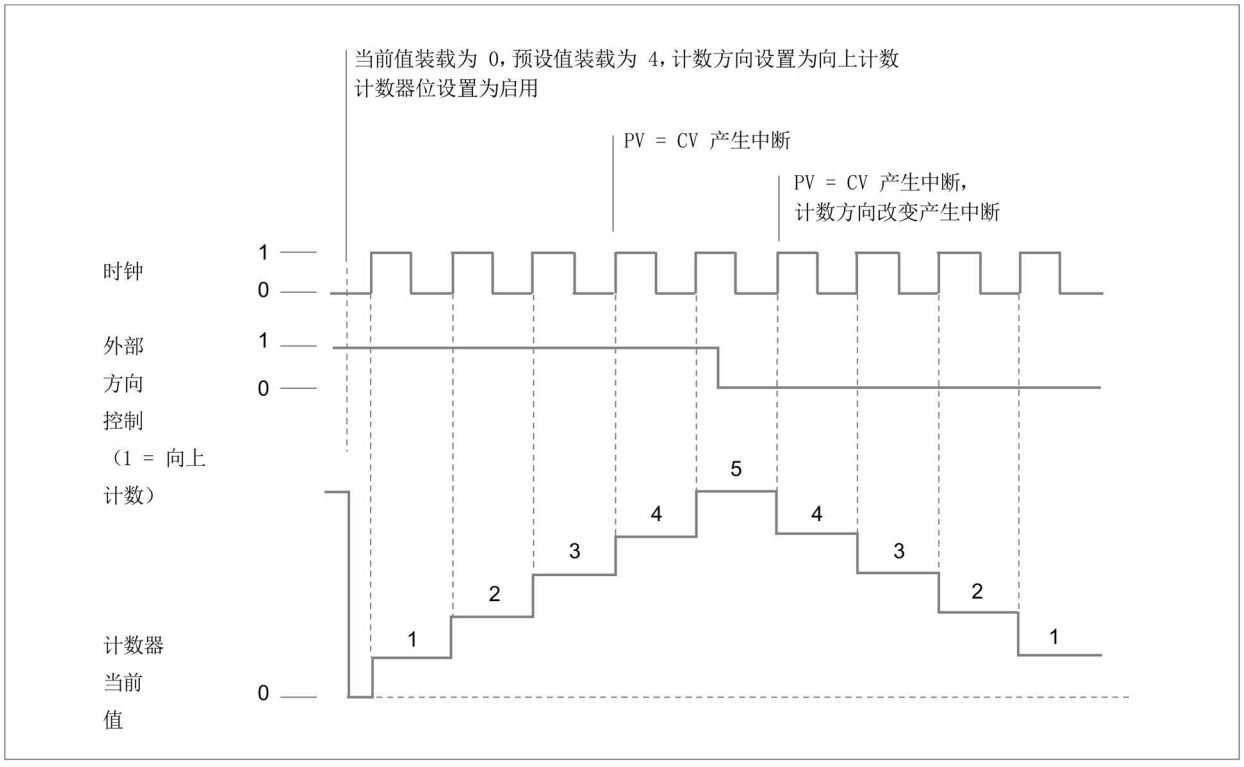
模式	说明	输入分配		
	HSC0	I0.0	I0.1	I0.4
	HSC1	I0.1		
	HSC2	I0.2	I0.3	I0.5
	HSC3	I0.3		
0	具有内部方向控制的单相计数器	时钟		
1		时钟		复位
3	具有外部方向控制的单相计数器	时钟	方向	
4		时钟	方向	复位
6	具有 2 个时钟输入的双相计数器	加时钟	减时钟	
7		加时钟	减时钟	复位
9	AB 正交相计数器	时钟 A	时钟 B	
10		时钟 A	时钟 B	复位

模式选择对计数操作的影响

HSC 模式 0 和 1

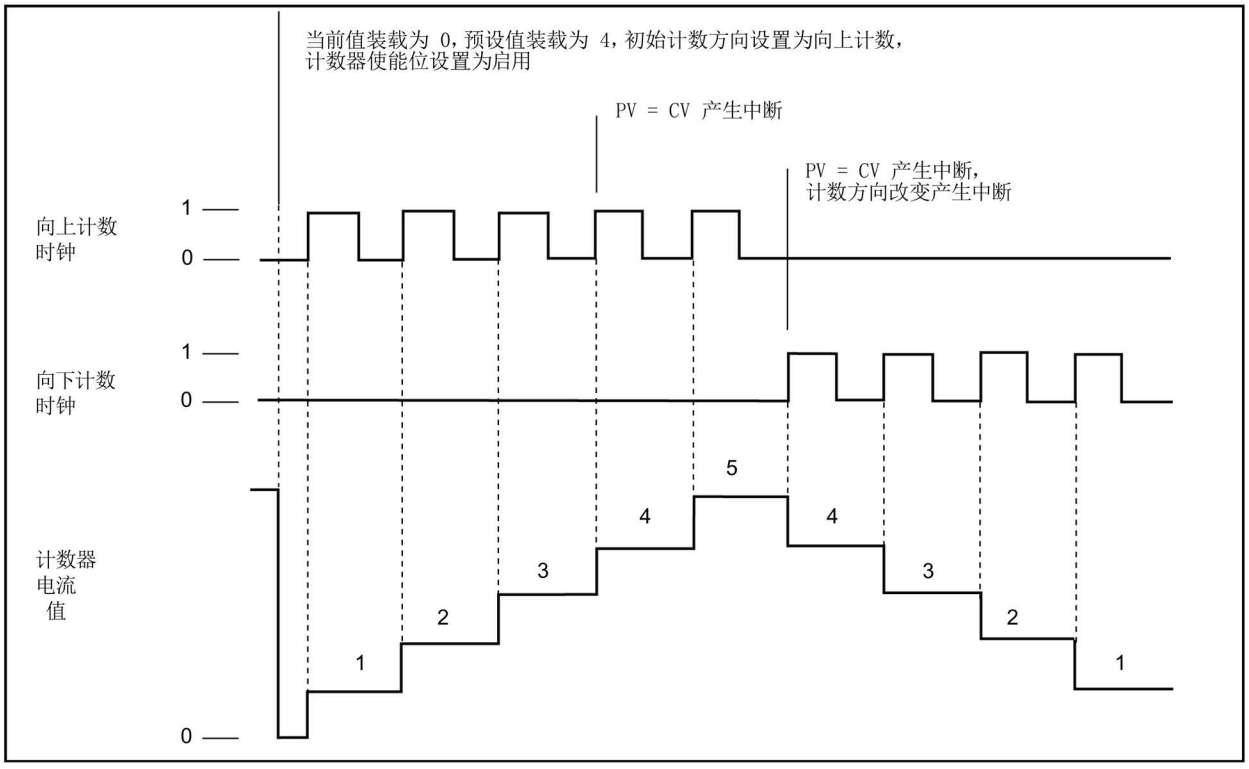


HSC 模式 3 和 4

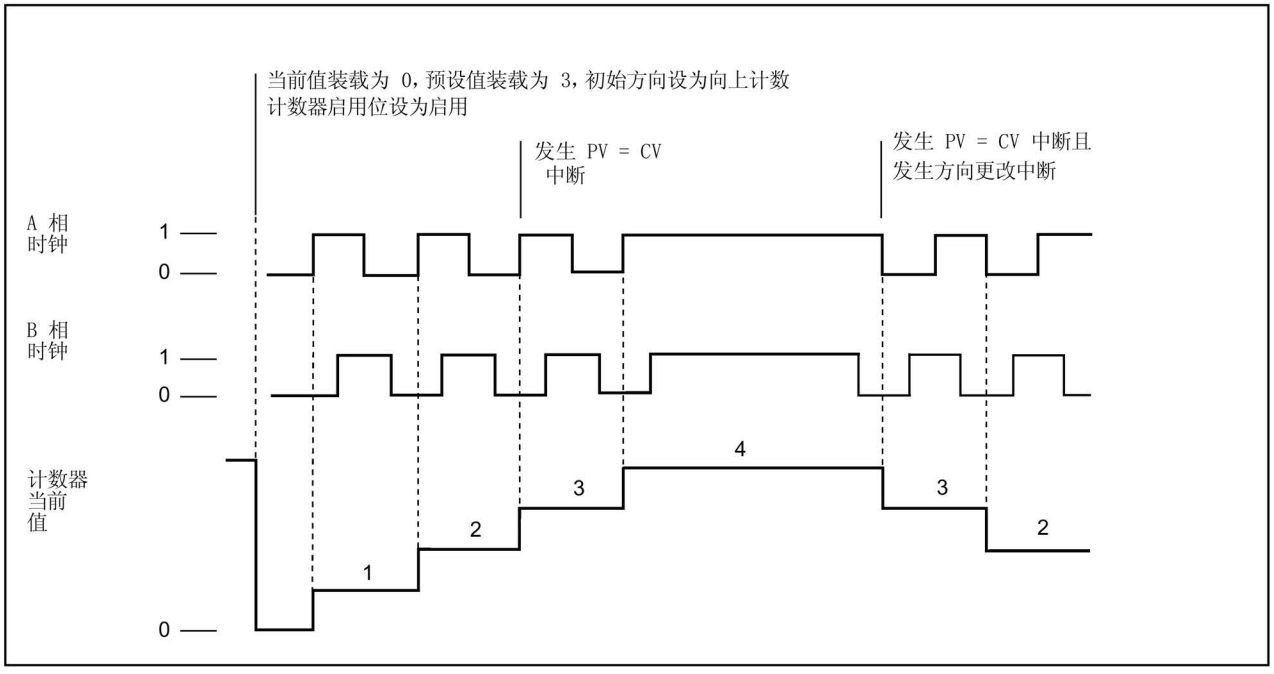

当前值装载为 0，预设值装载为 4，计数方向设置为向上计数
计数器位设置为启用
PV = CV 产生中断
PV = CV 产生中断，
计数方向改变产生中断
时钟
外部
方向
控制
（1 = 向上
计数）
计数器
当前
值
1
0
1
0
5
4
4
3
3
2
2
1
1
0

HSC 模式 6 和 7

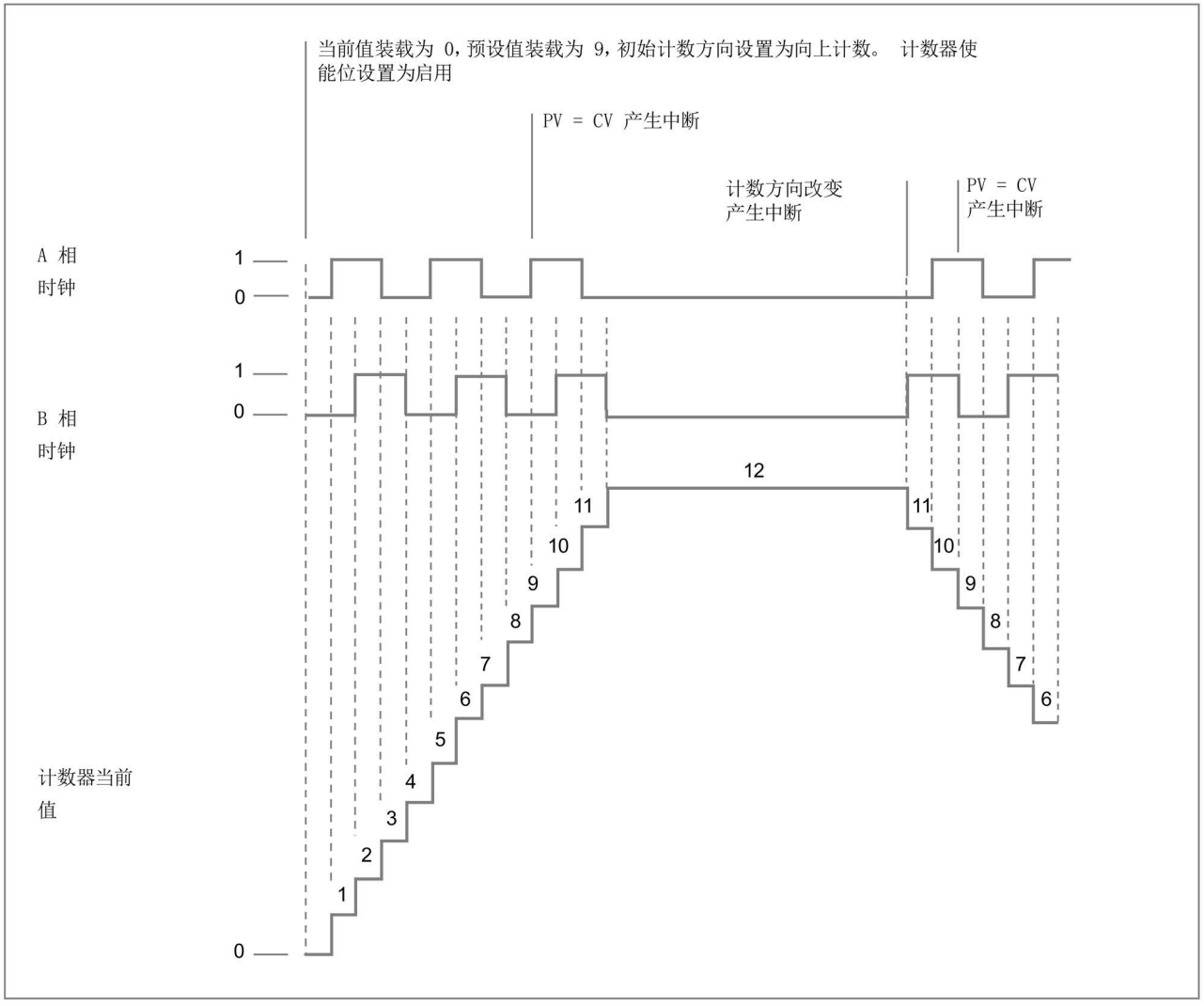
使用计数模式 6 或 7 时，如果加时钟和减时钟输入的上升沿在 0.3
微秒内发生，高速计数器可能认为这些事件同时发生。如果发生这种情况，当前值不改变，而且计数方向不改变。只要加时钟和减时钟输入的上升沿之间的间隔大于该时段，高速计数器就能够单独捕获每个事件。在两种情况下，均不生成错误，而且计数器保持正确计数值。



HSC 模式 9 和 10（AB 正交相位 1X）



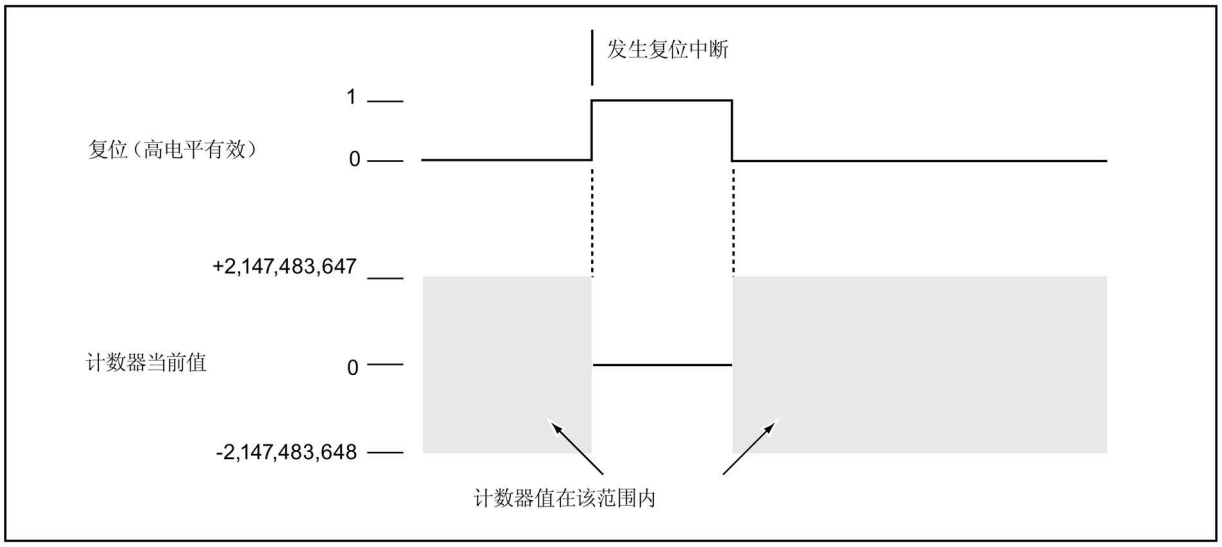
HSC 模式 9 和 10（AB 正交相位 4X）



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

HSC0 和 HSC2 计数器有两个控制位，用于组态复位的激活状态并选择 1x 或 4x 计数模式（仅限 AB 正交相计数器）。这些控制位位于各自计数器的 HSC 控制字节内，仅当执行 HDEF 指令时才会使用。下表定义了这些控制位。

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。

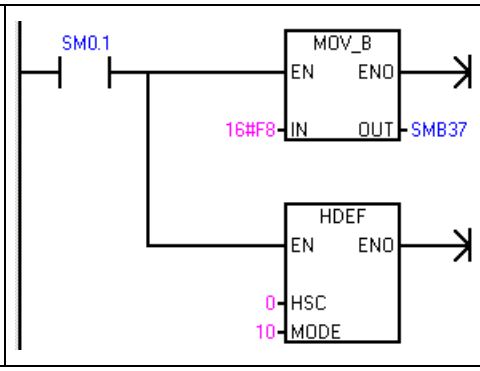
执行 HDEF 指令后，将无法再更改计数器设置，除非首先将 CPU 设为 STOP 模式。

HSC0	HSC1	HSC2	HSC3	描述（仅在执行 HDEF 时使用）
SM37.0	不受支持	SM57.0	不受支持	**复位的有效电平控制位**[1]： • 0 = 高电平激活时复位 • 1 = 低电平激活时复位
SM37.2	不受支持	SM57.2	不受支持	**AB 正交相计数器的计数速率选择**[1]： • 0 = 4X 计数速率 • 1 = 1X 计数速率

[1] 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 乘以输入时钟频率）。

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）或所有其它模式的初始计数方向
- 加载当前值
- 加载预设值

HSC 控制字节

HSC0	HSC1	HSC2	HSC3	说明
SM37.3	SM47.3	SM57.3	SM137.3	计数方向控制位： • 0 = 减计数 • 1 = 加计数
SM37.4	SM47.4	SM57.4	SM137.4	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	向 HSC 写入新预设值： • 0 = 不更新 • 1 = 更新预设值
SM37.6	SM47.6	SM57.6	SM137.6	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	启用 HSC： • 0 = 禁用 HSC • 1 = 启用 HSC

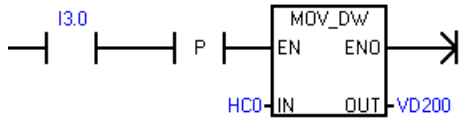
使用程序读取 HSC 当前值

只能使用后面带有计数器标识符编号（0、1、2 或 3）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 和 HSC3 的当前值

要读取的值	HSC0 地址	HSC1 地址	HSC2 地址	HSC3 地址
CV（计数器当前值）	HC0	HC1	HC2	HC3

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）：

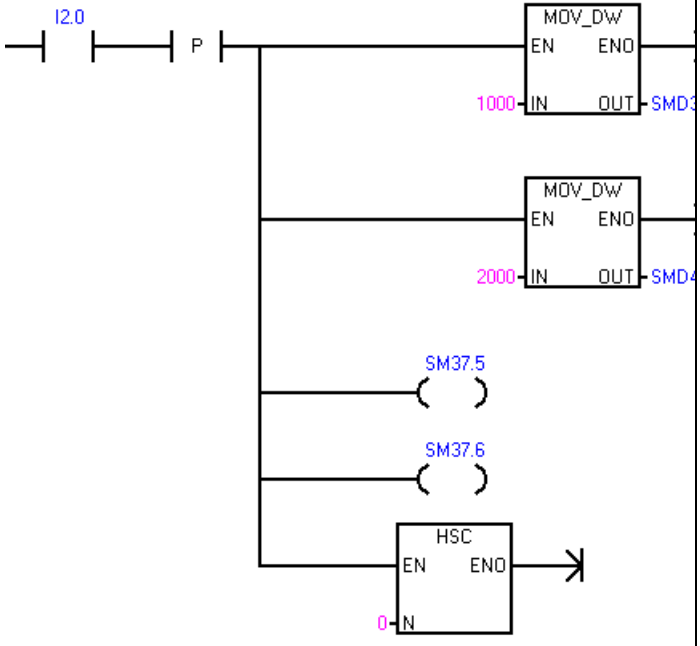
1. 加载要写入相应 SM 新当前值和/或新预设值的值（请参见下表）。加载这些新值尚不会影响高速计数器。
2. 设置或清除相应控制字节的相应位，指示是否更新当前值和/或预设值（位 x.5 代表预设值，位 x.6 代表当前值）。调节这些位尚不会影响高速计数器。
3. 执行引用相应高速计数器编号的 HSC 指令。执行该指令可检查控制字节。如果控制字节指定更新当前值、预设值或两者，则会将相应值从 SM 新当前值和/或新预设值位置复制到高速计数器内部寄存器中。

要加载的值	HSC0	HSC1	HSC2	HSC3
新当前值（新 CV）	SMD38	SMD48	SMD58	SMD138
新预设值（新 PV）	SMD42	SMD52	SMD62	SMD142

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 335)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或等于大于预设值。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。监控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 7- 14　HSC0、HSC1、HSC2 和 HSC3 的状态位

HSC0	HSC1	HSC2	HSC3	说明
SM36.5	SM46.5	SM56.5	SM136.5	**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	**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	**当前值大于预设值状态位：** • 0 = 小于或等于 • 1 = 大于

另请参见

高速计数器指令 (页 267)

高速计数器的初始化顺序示例 (页 286)

7.6.5 高速计数器的初始化顺序示例

HSC0 在以下初始化和操作顺序说明中被用作计数器。

- HSC0 和 HSC2 支持计数模式 (0, 1)、(3, 4)、(6, 7) 和 (9, 10)。
- HSC1 和 HSC3 仅支持计数模式 0。

初始化说明假设 CPU 刚刚被置于 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 HSC 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 SMB37。

   例如：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 SMB37。

   例如：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 SMB37。

   例如：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 SMB37。

   示例（1x 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）：

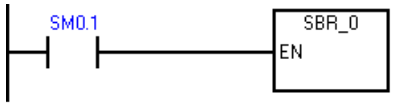
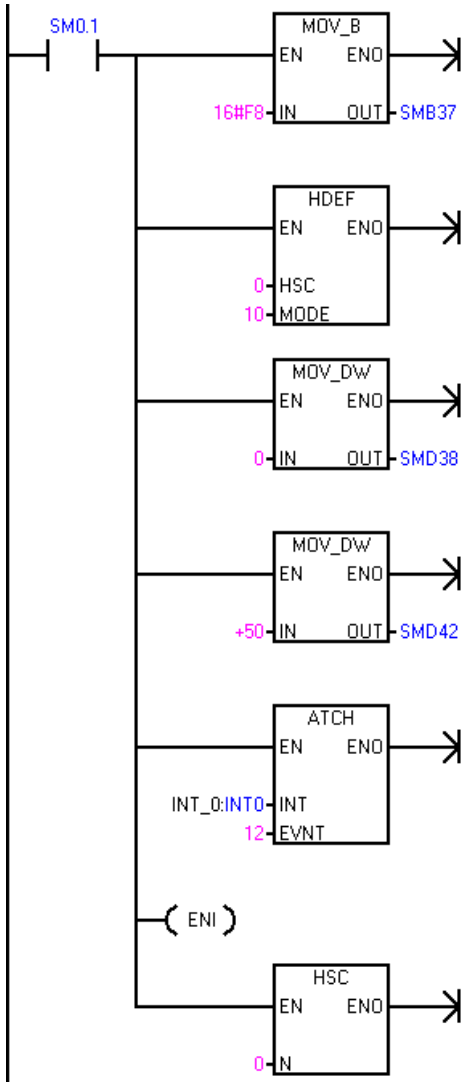
1. 加载 SMB37，以禁用计数器：

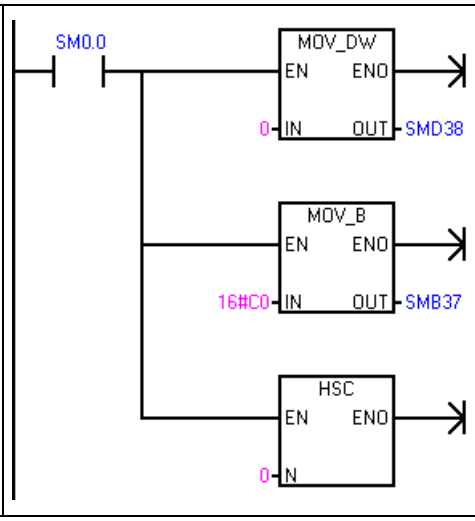
   SMB37 = 16#00

   – 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

<table>
<tr><th colspan="3">LAD</th><th>STL</th></tr>
<tr><td>MAIN</td><td> 
</td><td>首次扫描时，调用 SBR_0。</td><td>Network 1
LD SM0.1
CALL SBR_0</td></tr>
<tr><td>SBR0</td><td> 
</td><td>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。</td><td>Network 1
LD SM0.1
MOVB 16#F8, SMB37
HDEF 0, 10
MOVD +0, SMD38
MOVD +50, SMD42
ATCH INT_0, 12
ENI
HSC 0</td></tr>
</table>

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**

参见

高速计数器指令 (页 267)

高速计数器编程 (页 273)

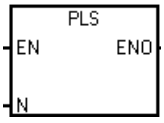
中断指令 (页 335)

7.7 脉冲输出

7.7.1 脉冲输出指令 (PLS)

脉冲输出 (PLS) 指令控制高速输出（Q0.0、Q0.1 和 Q0.3）是否提供脉冲串输出 (PTO) 和脉宽调制 (PWM) 功能。

若使用 PWM，可通过可选向导来创建 PWM 指令。

LAD/FBD	STL	说明
	**PLS N**	可使用 PLS 指令来创建最多三个 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：试图在脉冲输出有效时重新定义它 • 000EH：PTO 包络段数已设置为 0 • 0017H：试图为已分配给运动控制的 PTO/PWM 分配资源 • 001BH：试图改变已启用的 PWM 的时基 • 0090H：N 非 0、1 或 2。 • 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) 或 2 (= Q0.3)

该 CPU 具有三个 PTO/PWM 生成器（PLS0、PLS1 和 PLS2），可产生高速脉冲串或脉宽调制波。PLS0 分配给了数字输出端 Q0.0，PLS1 分配给了数字输出端 Q0.1，PLS2 分配给了数字输出端 Q0.3。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）、一个周期时间或频率（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 生成器和过程映像寄存器共同使用 Q0.0、Q0.1 和 Q0.3。若在 Q0、Q0.1 或 Q0.3 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。

在启用 PTO/PWM 操作前，请将过程映像寄存器中 Q0.0、Q0.1 和 Q0.3 的值设置为 0。所有控制位、周期时间/频率、脉冲宽度和脉冲计数值的默认值均为 0。

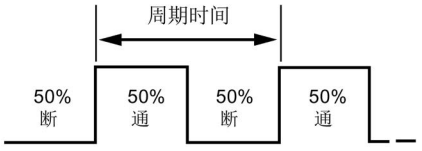
说明

脉冲输出 (PLS) 指令仅可用于以下 S7-200 SMART CPU：

- SR20/ST20（两个通道，Q0.0 和 Q0.1）
- SR30/ST30、SR40/ST40 以及 SR60/ST60（三个通道，Q0.0、Q0.1 和 Q0.3）

7.7.2 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 100,000 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 7- 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 > 100,000Hz	频率默认为 100,000 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

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 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，您必须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 或 SM566.6）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 100,000 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 或 SMW578）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 和 SM566.6）被置“1”，且 PTO 操作终止。

在 PTO 包络作用期间，在 SMB166、SMB176 或 SMB576 中提供当前有效段的编号。

表格 7- 16　多段 PTO 操作的包络表格式[1]

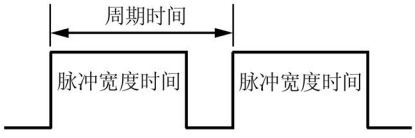
字节偏移量	段	表格条目的描述
0		段数量：1 到 255[2]
1	#1	起始频率（1 到 100,000 Hz）
5		结束频率（1 到 100,000 Hz）
9		脉冲计数（1 到 2,147,483,647）
13	#2	起始频率（1 到 100,000 Hz）
17		结束频率（1 到 100,000 Hz）
21		脉冲计数（1 到 2,147,483,647）
（依此类推）	#3	（依此类推）

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

7.7.3 脉宽调制 (PWM)

PWM
提供三条通道，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：10 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。

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。

7.7.4 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。“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。任何时候都可通过向 PTO/PWM 控制字节（SM67.7、SM77.7 或 SM567.7）使能位写入 0，然后执行 PLS 指令，来实现禁止生成 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周期时间。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 或 SM566.4）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 335)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 和 SMB566）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 功能将终止以及增量计算错误位（SM66.4、SM76.4 或 SM566.4）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 或 SM566.5）置 1。
- 如果以下任一情况发生，PTO/PWM 溢出/下溢位（SM66.6、SM76.6 或 SM566.6）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

- 确保您了解 PTO/PWM 模式选择位（SM67.6、SM77.6 和 SM567.6）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。
- 当装载周期时间/频率（SMW68、SMW78 或 SMW568）、脉冲宽度（SMW70、SMW80 或 SMW570）或脉冲计数（SMD72、SMW82 或 SMW572）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 或 SMW578）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 7- 17　PTO/PWM 控制寄存器的 SM 单元

Q0.0	Q0.1	Q0.3	状态位
SM66.4	SM76.4	SM566.4	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	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	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	PTO 空闲： • 0 = 进行中 • 1 = PTO 空闲

Q0.0	Q0.1	Q0.3	控制位
SM67.0	SM77.0	SM567.0	PTO/PWM 更新频率/周期时间： • 0 = 不更新 • 1 = 更新频率/周期时间
SM67.1	SM77.1	SM567.1	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	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	PWM 时基： • 0 = 1 μs/时标 • 1 = 1 ms/刻度
SM67.4	SM77.4	SM567.4	保留
SM67.5	SM77.5	SM567.5	PTO 单/多段操作： • 0 = 单段 • 1 = 多段
SM67.6	SM77.6	SM567.6	PTO/PWM 模式选择： • 0 = PWM • 1 = PTO
SM67.7	SM77.7	SM567.7	PWM 使能： • 0 = 禁用 • 1 = 启用

Q0.0	Q0.1	Q0.3	其它寄存器
SMW68	SMW78	SMW568	PTO 频率或 PWM 周期时间值：1 到 65,535 Hz (PTO)，2 到 65,535 (PWM)
SMW70	SMW80	SMW570	PWM 脉冲宽度值：0 到 65,535
SMD72	SMD82	SMD572	PTO 脉冲计数值：1 到 2,147,483,647
SMB166	SMB176	SMB576	进行中段的编号： 仅限多段 PTO 操作
SMW168	SMW178	SMW578	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 7- 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	时基	脉冲计数	脉冲宽度	周期时间/频率
16#80	是	PWM		1 μs/周期			
16#81	是	PWM		1 μs/周期			更新周期时间
16#82	是	PWM		1 μs/周期		更新	
16#83	是	PWM		1 μs/周期		更新	更新周期时间
16#88	是	PWM		1 ms/周期			
16#89	是	PWM		1 ms/周期			更新周期时间
16#8A	是	PWM		1 ms/周期		更新	
16#8B	是	PWM		1 ms/周期		更新	更新周期时间
16#C0	是	PTO	单段				

		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	时基	脉冲计数	脉冲宽度	周期时间/频率
16#C1	是	PTO	单段				更新频率
16#C4	是	PTO	单段		更新		
16#C5	是	PTO	单段		更新		更新频率
16#E0	是	PTO	多段				

7.7.5 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。

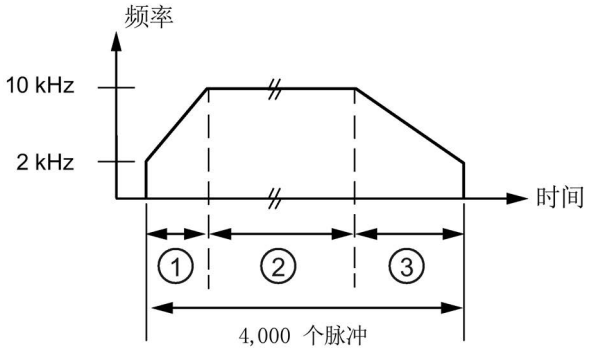
下图说明了生成输出波形所需的采样包络表值：

- 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。

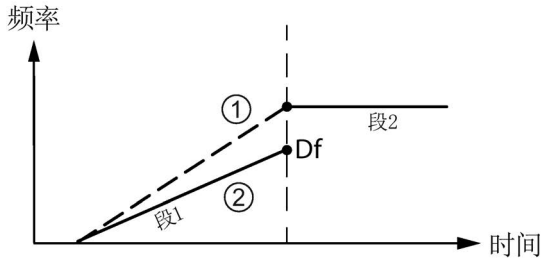
表格 7- 19　包络表值

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：

$$\Delta F = F_{Final} - F_{Initial}$$

$$T_s = PC / (F_{min} + (| \Delta F | / 2))$$

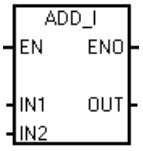
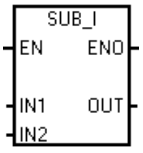
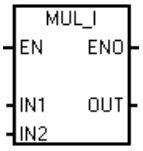
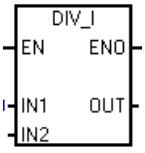
$$A_s = \Delta F / T_s$$

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)

7.8 数学

7.8.1 加法、减法、乘法和除法

LAD / FBD	STL	说明
 ADD_DI ADD_R	**+I IN1, OUT** **+D IN1, OUT** **+R IN1, 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 和 FBD：IN1 + IN2 = OUT • STL： IN1 + OUT = OUT
 SUB_DI SUB_R	**-I IN1, OUT** **-D IN1, OUT** **-R IN1, 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 和 FBD：IN1 - IN2 = OUT • STL：OUT - IN1 = OUT
 MUL_DI MUL_R	***I IN1, OUT** ***D IN1, OUT** ***R IN1, 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 和 FBD：IN1 * IN2 = OUT • STL：IN1 * OUT = OUT
 DIV_DI DIV_R	**/I IN1, OUT** **/D IN1, OUT** **/R IN1, 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 和 FBD：IN1/IN2 = OUT • STL：OUT / IN1 = OUT

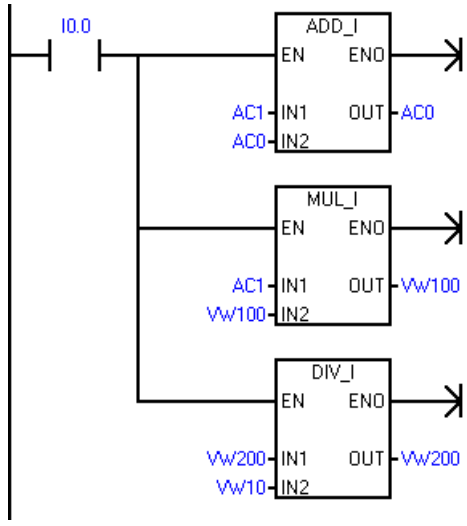
ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • SM1.1 溢出 • SM1.3 除数为零	• SM1.0 运算结果 = 零 • SM1.1 溢出、运算期间生成非法值或非法输入 • SM1.2 负数结果 • SM1.3 除数为零

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，则其它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

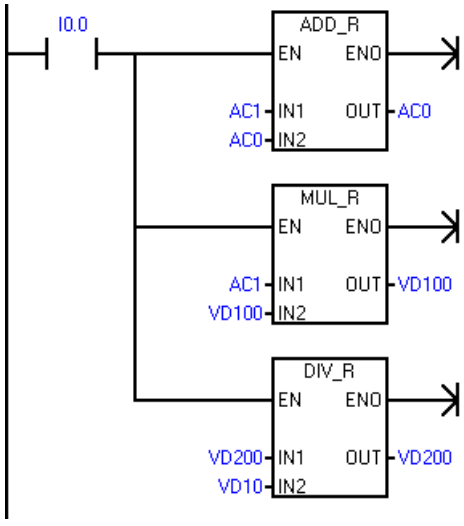
LAD	STL
	**Network** **LD I0.0** **+I AC1, AC0** ***I AC1, VW100** **/I VW10, VW200**

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

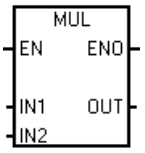
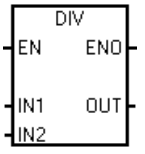
LAD	STL

	`Network 1`
	`LD I0.0`
	`+R AC1, AC0`
	`*R AC1, VD100`
	`/R 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

7.8.2 产生双整数的整数乘法和带余数的整数除法

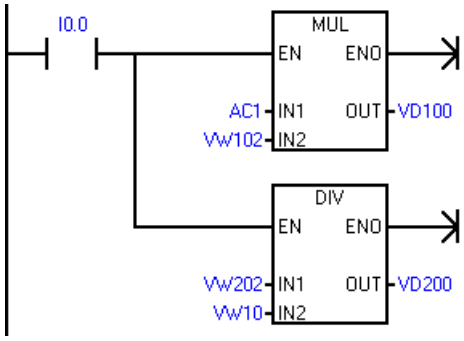
LAD/FBD	STL	说明
	**MUL IN1, OUT**	两个整数的整数乘法指令将两个 16 位整数相乘，产生一个 32 位乘积。 在 STL 中，32 位 OUT 的最低有效字（16 位）被用作其中一个乘数。 • LAD 和 FBD：IN1 * IN2 = OUT • STL：IN1 * OUT = OUT
	**DIV IN1, OUT**	带余数的整数除法指令将两个 16 位整数相除，产生一个 32 位结果，该结果包括一个 16 位的余数（最高有效字）和一个 16 位的商（最低有效字）。 在 STL 中，32 位 OUT 的最低有效字（16 位）用作被除数。 • LAD 和 FBD：IN1/IN2 = OUT • STL：OUT / 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，则其它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, VD100** **DIV VW10, VD200**

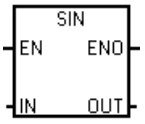
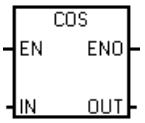
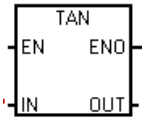
[1] VD100 包含：VW100 和 VW102，VD200 包含：VW200 和 VW202。

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	
					余数	商
数据相除	4000	/	41	=	23	97
数据地址	VW202		VW10		VW200	VW202 VD200

7.8.3 三角函数、自然对数/自然指数和平方根

正弦 (SIN)、余弦 (COS) 和正切 (TAN) 指令

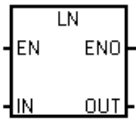
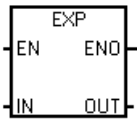
LAD/FBD	STL	说明
	**SIN IN, OUT**	正弦 (SIN)、余弦 (COS) 和正切 (TAN) 指令计算角度值 IN 的三角函数，并在 OUT 中输出结果。 输入角度值以弧度为单位。 • SIN (IN) = OUT • COS (IN) = OUT • TAN (IN) = OUT 要将角度从度转换为弧度： 使用 MUL_R (*R) 指令将以度为单位的角度乘以 1.745329E-2（约为 π/180）。 对于数学函数指令，SM1.1 用于指示溢出错误和非法值。 如果 SM1.1 置位，则 SM1.0 和 SM1.2 的状态无效，原始输入操作数不变。 如果 SM1.1 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 IN, OUT**	
	**TAN 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

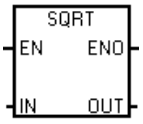
LAD/FBD	STL	说明
	**LN IN, OUT**	自然对数指令 (LN) 对 IN 中的值执行自然对数运算，并在 OUT 中输出结果。 自然指数指令 (EXP) 执行以 e 为底，以 IN 中的值为幂的指数运算，并在 OUT 中输出结果。 • LN (IN) = OUT • EXP (IN)= OUT 要从自然对数获得以 10 为底的对数： 将自然对数除以 2.302585（约为 10 的自然对数）。 若要将任意实数作为另一个实数的幂，包括分数指数：组合自然指数指令和自然对数指令。 例如，要将 X 作为 Y 的幂，请使用 EXP (Y * LN (X))。
	**EXP IN, OUT**	

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

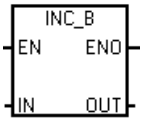
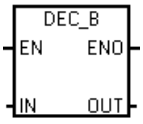
LAD/FBD	STL	说明
	**SQRT IN, OUT**	平方根指令 (SQRT) 计算实数 (IN) 的平方根，产生一个实数结果 OUT。 • SQRT (IN)= OUT 要获得其它根： • 5 的立方 = 5^3 = EXP(3*LN(5)) = 125 • 125 的立方根 = 125^(1/3) = EXP((1/3)*LN(125))= 5 • 5 的立方的平方根 = 5^(3/2) = EXP(3/2*LN(5)) = 11.18034

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。

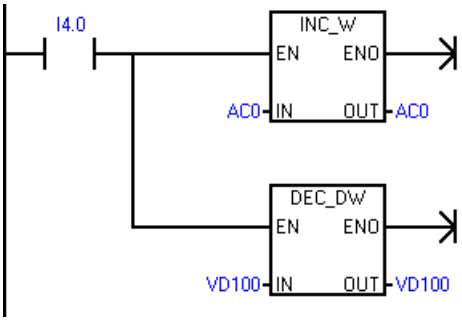
7.8.4 递增和递减

LAD/FBD	STL	说明
 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_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

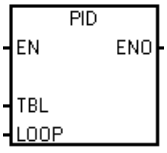
示例：递增和递减

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

7.9 PID

LAD/FBD	STL	说明
	**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 7）

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 1（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 7 的回路编号。

可以在程序中使用八条 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。

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” (页 609)。 “PID 整定控制面板” (页 619)只能用于通过 PID 向导创建的 PID 回路。

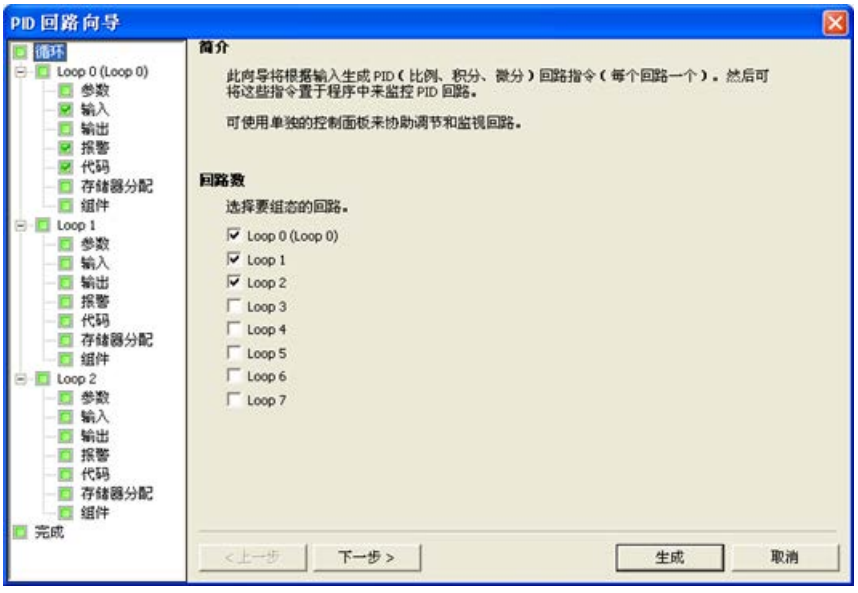
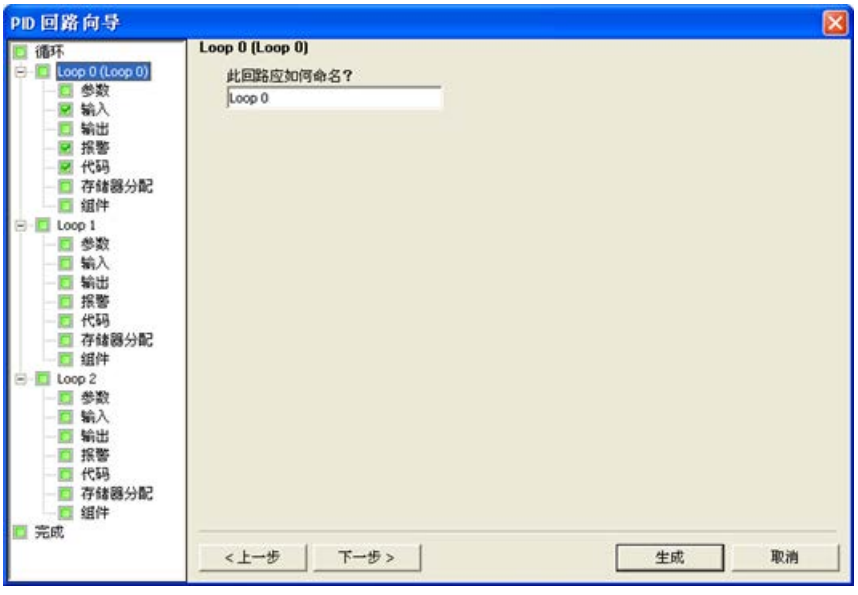
STEP 7-Micro/WIN SMART 提供 PID 向导，指导您为闭环控制过程定义 PID 算法。
从“工具”(Tools) 菜单中选择“指令向导”(Instruction Wizard)
命令，然后从“指令向导”(Instruction Wizard) 窗口中选择“PID”。

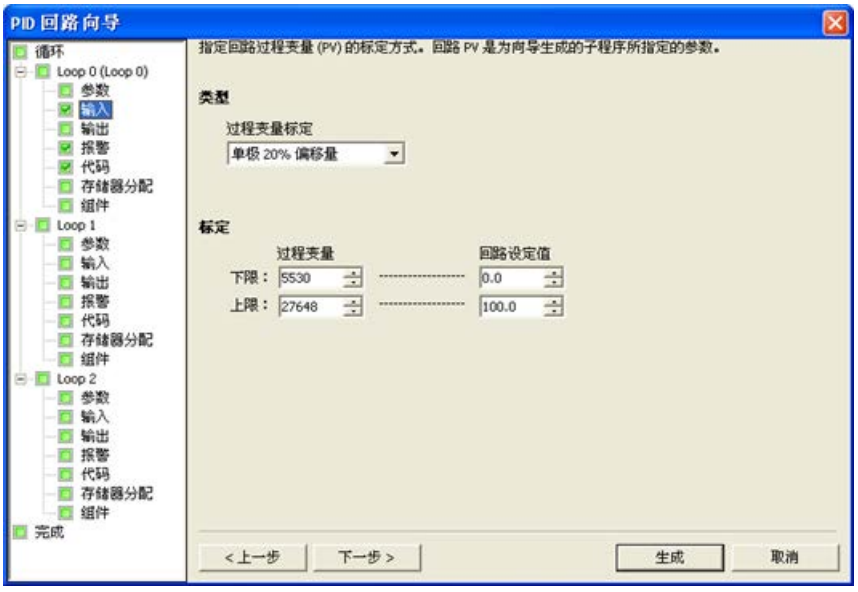
说明

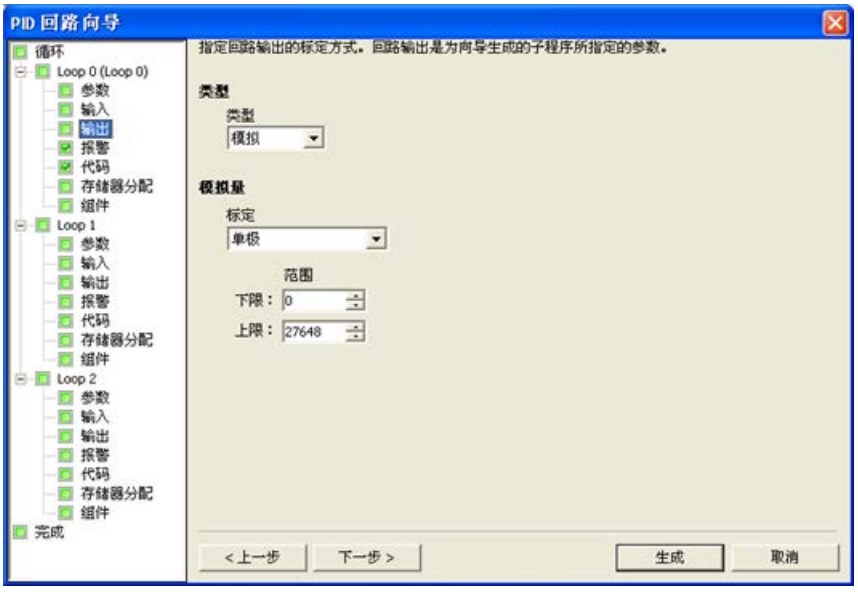
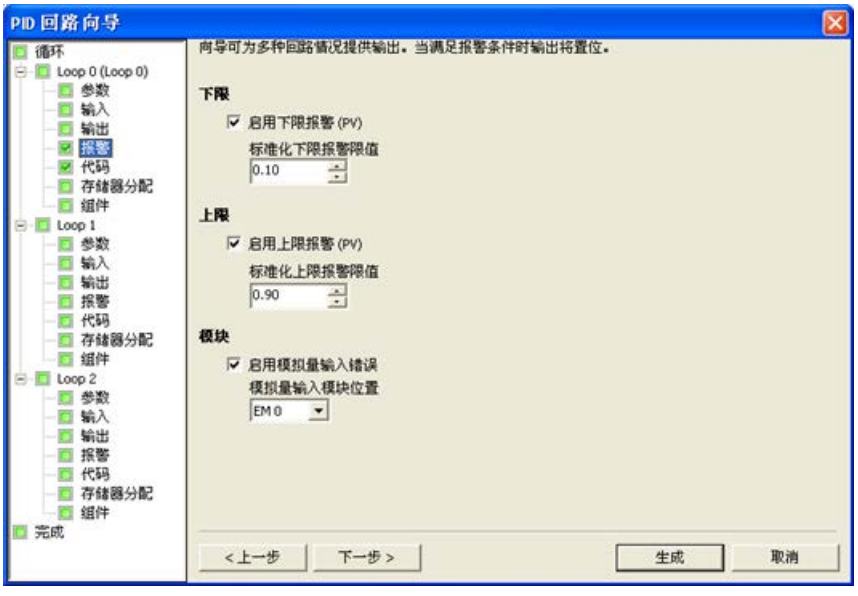
上限设定值和下限设定值应与过程变量的上下限对应。

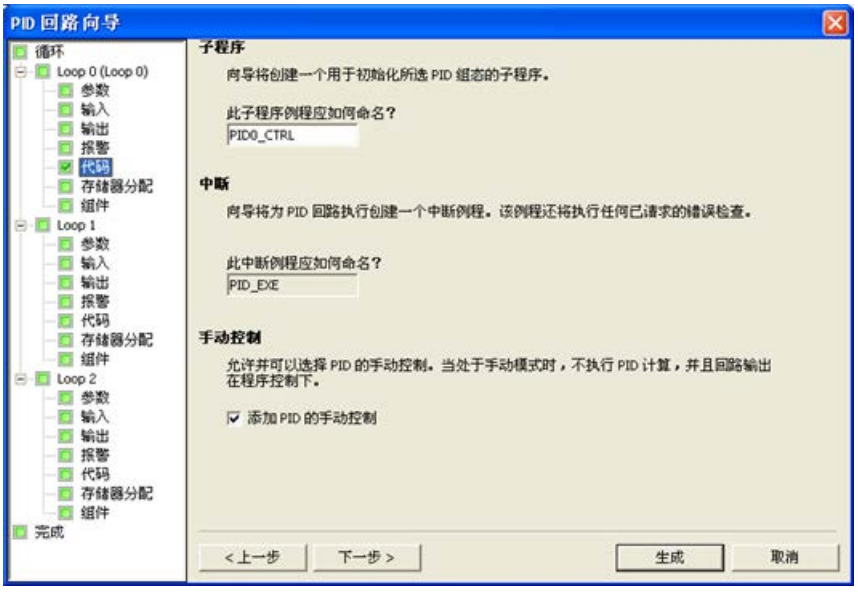
7.9.1 使用 PID 向导

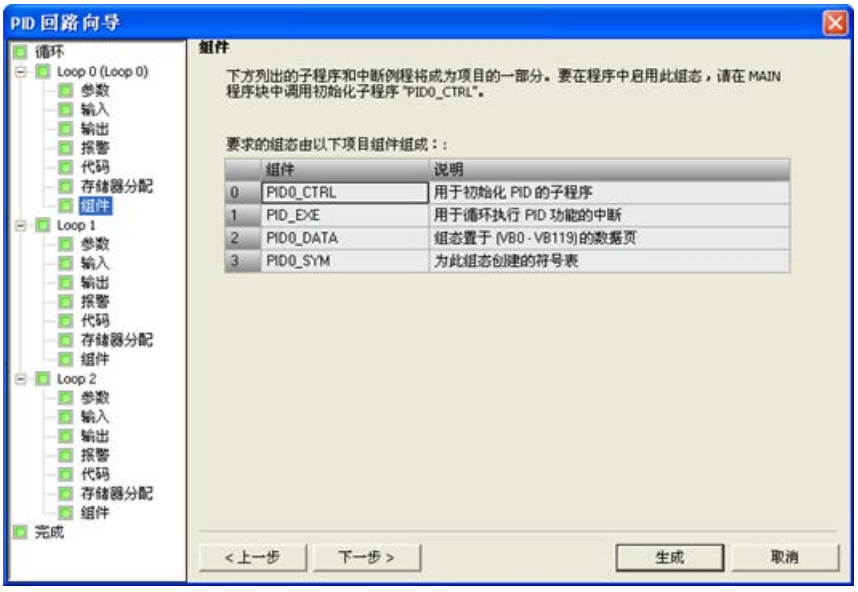
使用 PID 向导组态 PID 回路

画面	说明
	在此对话框中选择要组态的回路。 最多可组态 8 个回路。 在此对话框上选择回路时，PID 向导左侧的树视图随组态该回路所需的所有节点一起更新。
	可为回路组态自定义名称。此部分的默认名称是“回路 x”，其中“x”等于回路编号。

画面	说明
	设置下列回路参数： • 增益（默认值 = 1.00） • 采样时间（默认值 = 1.00） • 积分时间（默认值 = 10.00） • 微分时间（默认值 = 0.00）
	指定回路过程变量 (PV) 如何标定。 可以从以下选项中选择： • 单极性（默认值： 0 到 27648；可编辑） • 双极性（默认值： -27648 到 27648；可编辑） • 单极性 20% 偏移量（范围：5530 到 27648；已设定，不可变更） • 温度 x 10 °C • 温度 x 10 °F 在“标定”(Scaling) 参数中，指定回路设定值 (SP) 如何标定。 默认值是 0.0 和 100.0 之间的一个实数。

画面	说明
	输入回路输出选项： • 回路输出如何标定： – 模拟量 – 数字量 • 模拟标定参数： – 单极性（默认值： 0 到 27648；可编辑） – 双极性（默认值： -27648 到 24678；可编辑） – 单极性 20% 偏移量（范围：5530 到 27648；已设定，不可变更） • 模拟量范围参数： 指定回路输出范围。 可能的范围为 -27648 到 +27648，具体取决于标定选择。
	可指定通过报警输入识别的条件。 根据需要使用复选框启用警报： • 报警下限 (PV)： 设置 0.0 到报警上限之间的标准化报警下限；默认值是 0.10。 • 报警上限 (PV)： 设置报警下限到 1.00 之间的标准化报警上限，默认值是 0.90。 • 模拟量输入错误： 指定将输入模块连接到 PLC 的位置。

画面	说明
	可进行以下代码选择： • 子例程： PID 向导创建用于初始化所选 PID 组态的子例程。 • 中断： PID 向导创建用于 PID 回路执行的中断例程。 注： 向导为子例程和中断例程指定了默认名称；您可编辑该默认名称。 • 手动控制： 使用“添加 PID 的手动控制”(Add Manual Control of the PID) 复选框以允许手动控制 PID 回路。

画面	说明
	可指定数据块中放置组态的 V 存储器字节的起始地址。 向导可建议一个表示大小正确且未使用的 V 存储器块的地址。
	该屏幕显示 PID 向导生成的子例程和中断例程列表，并对如何将它们集成到您的程序中进行简要说明。

STEP 7-Micro/WIN SMART 中包含 PID 整定控制面板 (页 619)，允许您以图形方式监视 PID 回路。

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。

7.9.2 PID 算法

在稳态运行中，PID 控制器调节输出值，使偏差 (e) 为零。 偏差是设定值 (SP)（所需工作点）与过程变量 (PV)（实际工作点）之差。 PID 控制的原理基于以下方程，输出 M(t) 是比例项、积分项和微分项的函数：

输出 = 比例项 + 积分项 + 微分项

$$M(t) = K_C * e + K_C \int_0^t e\,dt + M_{initial} + K_C * de/dt$$

其中：

M(t)	回路输出（时间的函数）
K_C	回路增益
e	回路偏差（设定值与过程变量之差）
$M_{initial}$	回路输出的初始值

要在数字计算机中执行该控制函数，必须将连续函数量化为偏差值的周期采样，并随后计算输出。 数字计算机进行处理采用的相应方程如下：

输出 = 比例项 + 积分项 + 微分项

$$M_n = K_c * e_n + K_I * \sum_1^n e_x + M_{initial} + K_D * (e_n - e_{n-1})$$

其中：

M_n	采样时间 n 时的回路输出计算值
K_c	回路增益
e_n	采样时间 n 时的回路偏差值
e_{n-1}	前一回路偏差值（采样时间 n - 1 时）
K_I	积分项的比例常数
$M_{initial}$	回路输出的初始值
K_D	微分项的比例常数

从该公式中可以看出，积分项是从第 1 次采样到当前采样所有偏差项的函数。
微分项是当前采样和前一次采样的函数，而比例项仅是当前采样的函数。
在数字计算机中，保存偏差项的所有样本既不实际，也没有必要。

因为从第一个样本开始，每次对偏差进行采样时数字计算机都必须计算输出值，因此仅需存储前一偏差值和前一积分项值。
由于数字计算机解决方案具有重复特性，因此可以简化在任何采样时间都需计算的方程。简化后的方程如下：

输出 = 比例项 + 积分项 + 微分项

$$M_n = K_C * e_n + K_I * e_n + MX + K_D * (e_n - e_{n-1})$$

其中：

M_n	采样时间 n 时的回路输出计算值
K_c	回路增益
e_n	采样时间 n 时的回路偏差值
e_{n-1}	前一回路偏差值（采样时间 n - 1 时）
K_I	积分项的比例常数
MX	前一积分项的值（采样时间 n - 1 时）
K_D	微分项的比例常数

CPU 使用以上简化方程的改进方程计算回路输出值。 改进的方程如下：

输出 = 比例项 + 积分项 + 微分项

$$M_n = MP_n + MI_n + MD_n$$

其中：

M_n	采样时间 n 时的回路输出计算值
MP_n	采样时间 n 时回路输出的比例项值
MI_n	采样时间 n 时回路输出的积分项值
MD_n	采样时间 n 时回路输出的微分项值

理解 PID 方程的元素

PID 方程的比例项： 比例项 MP 是增益 (K_C) 与偏差 (e) 的乘积，其中，增益控制输出计算的灵敏度，偏差是给定采样时间时的设定值 (SP) 与过程变量 (PV) 之差。 CPU 求解比例项所采用的方程如下：

$$MP_n = K_C * (SP_n - PV_n)$$

其中：

MP_n	采样时间 n 时回路输出的比例项值
K_C	回路增益
SP_n	采样时间 n 时的设定值
PV_n	采样时间 n 时的过程变量值

PID 方程的积分项： 积分项 MI 与一段时间内的偏差之和 (e) 成比例。 CPU 求解积分项所采用的方程如下：

$$MI_n = K_1 e_n + MX = K_C * (T_S / T_I) * (SP_n - PV_n) + MX$$

其中：

MI_n	采样时间 n 时回路输出的积分项值
K_C	回路增益
T_S	回路采样时间
T_I	积分时间（也称为积分时间或复位）
SP_n	采样时间 n 时的设定值
PV_n	采样时间 n 时的过程变量值
MX	采样时间 n-1 时的积分项值（也称为积分和或偏置）

积分和或偏置 (MX) 是积分项的所有先前值之和。 每次计算 MI_n 之后，都会使用 MI_n 值（该值可能被调整或限定）更新偏置（有关详细信息，请参见“变量和范围”部分）。 偏置的初始值通常设为第一次计算回路输出之前的输出值 ($M_{initial}$)。

积分项还包括几个常数：增益 (K_C)、采样时间 (T_S)、积分时间或复位 (T_I)，其中，采样时间是 PID 回路重新计算输出值的周期时间，积分时间是用于控制积分项在输出计算中的影响的时间。

PID 方程的微分项： 微分项 MD 与偏差变化成比例。 CPU 使用以下方程来求解微分项：

$$MD_n = K_C * (T_D / T_S) * ((SP_n - PV_n) - (SP_{n-1} - PV_{n-1}))$$

为避免由于设定值变化而导致微分作用激活引起输出发生阶跃变化或跳变，对此方程进行了改进，假定设定值为常数 ($SP_n = SP_{n-1}$)。
这样，将计算过程变量的变化而不是偏差的变化，如下所示：

$$MD_n = K_C * (T_D / T_S) * ((SP_n - PV_n) - (SP_{n-1} - PV_{n-1}))$$

或：

$$MD_n = K_C * (T_D / T_S) * (PV_{n-1} - PV_n)$$

其中：

MD_n	采样时间 n 时回路输出的微分项值
K_C	回路增益
T_S	回路采样时间
T_D	回路的微分周期（也称为微分时间或速率）
SP_n	采样时间 n 时的设定值
SP_{n-1}	采样时间 n - 1 时的设定值
PV_n	采样时间 n - 1 时的过程变量值
PV_{n-1}	采样时间 n - 1 时的过程变量值

必须保存过程变量而不是偏差，供下次计算微分项使用。 在第一次采样时，PV_{n-1} 的值初始化为等于 PV_n。

选择回路控制的类型

在许多控制系统中，可能只需使用一种或两种回路控制方法。
例如，可能只需要使用比例控制或比例积分控制。
可以通过设置常数参数值来选择所需的回路控制类型。

如果不需要积分作用（PID
计算中没有“I”），则应为积分时间（复位）指定无穷大值“INF”。
即使没有使用积分作用，积分项的值也可能不为零，这是因为积分和 MX 有初始值。

如果不需要微分作用（PID 计算中没有“D”），则应为微分时间（速率）指定值 0.0。

如果不需要比例作用（PID 计算中没有“P”），但需要 I 或 ID 控制，则应为增益指定值 0.0。
由于回路增益是计算积分项和微分项的方程中的一个系数，如果将回路增益设置为值 0.0，计算积分项和微分项时将对回路增益使用值 1.0。

7.9.3 转换和标准化回路输入

一个回路有两个输入变量，分别是设定值和过程变量。
设定值通常是固定值，例如汽车巡航控制装置上的速度设置。
过程变量是与回路输出相关的值，因此可衡量回路输出对受控系统的影响。
在巡航控制示例中，过程变量是测量轮胎转速的测速计输入。

设定值和过程变量都是实际值，其大小、范围和工程单位可以不同。 在 PID
指令对这些实际值进行运算之前，必须将这些值转换为标准化的浮点型表示。

第一步是将实际值从 16 位整数值转换为浮点值或实数值。
下面的指令序列显示了如何将整数值转换为实数值。

```
ITD    AIW0, AC0    //将输入值转换为双字
DTR    AC0, AC0     //将 32 位整数转换为实数
```

下一步是将实际值的实数值表示转换为 0.0 到 1.0 之间的标准化值。
下面的公式用于标准化设定值或过程变量值：

$$R_{Norm} = ((R_{Raw} / Span) + Offset)$$

其中：

R_{Norm} 实际值的标准化实数值表示

R_{Raw} 实际值的非标准化或原始实数值表示

偏移 对于单极性值为 0.0
对于双极性值为 0.5

跨度 最大可能值减去最小可能值：
对于单极性值为 27,648（典型值）
对于双极性值为 55,296（典型值）

下面的指令序列显示了如何标准化 AC0 中的双极性值（其跨度为 55,296），该指令序列是前一指令序列延续：

```
/R    55296.0, AC0 //标准化累加器中的值
+R    0.5, AC0     //将值转换到 0.0 到 1.0 之间
MOVR  AC0, VD100   //将标准化值存储在回路表中
```

7.9.4 将回路输出转换为标定整数值

回路输出是控制变量，例如汽车巡航控制装置上的节气门设置。 回路输出是介于 0.0 到 1.0 之间的标准化实数值。 回路输出转换为 16
位标定整数值后，才能用于驱动模拟量输出。 此过程与将 PV 和 SP
转换为标准化值的过程相反。
第一步是使用下面给出的公式将回路输出转换为标定实数值：

$$R_{Scal} = (M_n - Offset) * Span$$

R_{Scal} 回路输出的标定实数值
M_n 回路输出的标准化实数值
偏移 对于单极性值为 0.0
对于双极性值为 0.5
跨度 最大可能值减去最小可能值：
对于单极性值为 27,648（典型值）
对于双极性值为 55,296（典型值）

以下指令序列显示了如何标定回路输出：

```
MOVR  VD108, AC0   //将回路输出移至累加器
-R    0.5, AC0     //仅当值为双极性值时才使用本语句
*R    55296.0, AC0 //标定累加器中的值
```

接下来，必须将代表回路输出的标定实数值转换为 16 位整数。
下列指令序列显示了如何进行此转换：

```
ROUND  AC0, AC0     //将实数转换为 32 位整数
DTI    AC0, LW0     //将该值转换为 16 位整数
MOVW   LW0, AQW0    //将该值写入到模拟量输出
```

7.9.5 正作用或反作用回路

如果增益为正，则回路为正作用回路；如果增益为负，则回路为反作用回路。
（对于增益值为 0.0 的 I 或 ID
控制，如果将积分时间和微分时间指定为正值，则回路将是正作用回路；如果指定负值，则回路将是反作用回路。）

变量和范围

过程变量和设定值是 PID 计算的输入值。 因此，PID
指令只能读出这些变量的回路表字段，而不能改写。

输出值通过 PID 计算得出，因此，每次 PID
计算完成之后，会更新回路表中的输出值字段。 输出值限定在 0.0 到 1.0 之间。
当输出从手动控制转换为 PID
指令（自动）控制时，用户可使用输出值字段作为输入来指定初始输出值。
（请参见下面的“模式”部分中的讨论。）

如果使用积分控制，则偏置值通过 PID 计算更新，并且更新值将用作下一次 PID
计算的输入。 如果计算出的输出值超出范围（输出小于 0.0 或大于
1.0），则将按照下列公式调整偏置：

- 如果计算出的输出 $M_n > 1.0$

  $$MX = 1.0 - (MP_n + MD_n)$$

- 如果计算出的输出 $M_n < 0$

  $$MX = -(MP_n + MD_n)$$

  MX 调整后的偏置值
  MP_n 采样时间 n 时回路输出的比例项值
  MD_n 采样时间 n 时回路输出的微分项值
  M_n 采样时间 n 时的回路输出值

如上所述调整偏置后，如果计算出的输出回到正常范围内，可提高系统响应性。
计算出的偏置也会限制在 0.0 到 1.0 之间，然后在每次 PID
计算完成时写入回路表的偏置字段。 存储在回路表中的值用于下一次 PID 计算。

用户可以在执行 PID
指令之前修改回路表中的偏置值，在某些应用情况下，这样可以解决偏置值问题。
手动调整偏置时必须格外小心，向回路表中写入的任何偏置值都必须是 0.0 到 1.0
之间的实数。

过程变量的比较值保留在回路表中，用于 PID 计算的微分作用部分。 不应修改该值。

模式

PID 回路没有内置模式控制。 仅当能流流到 PID 功能框时才会执行 PID 计算。
因此，循环执行 PID 计算时存在“自动化”或“自动”模式。 不执行 PID
计算时存在“手动”模式。

与计数器指令相似，PID 指令也具有能流历史位。 该指令使用该历史位检测 0 到 1
的能流转换。
如果检测到能流转换，该指令将执行一系列操作，从而实现从手动控制无扰动地切换到自动控制。
要无扰动地切换到自动模式，在切换到自动控制之前，必须提供手动控制设置的输出值作为 PID 指令的输入（写入 M_n 的回路表条目）。 检测到 0 到 1 的能流转换时，PID
指令将对回路表中的值执行以下操作，以确保无扰动地从手动控制切换到自动控制：

- 设置设定值 (SP_n) = 过程变量 (PV_n)
- 设置旧过程变量 (PV_{n-1}) = 过程变量 (PV_n)
- 设置偏置 (MX) = 输出值 (M_n)

PID 历史位的默认状态为“置位”，控制器启动和每次从 STOP 切换到 RUN
模式时设置此状态。 如果在进入 RUN 模式后首次执行 PID
功能框时有能流流到该功能框，则检测不到能流转换且不会执行无扰动模式切换操作。

报警检查和特殊操作

PID 指令是一种简单但功能强大的指令，可执行 PID 计算。
如果需要进行其它处理，例如报警检查或回路变量的特殊计算，则必须使用 CPU
支持的基本指令来实现。

错误条件

如果在指令中指定的回路表起始地址或 PID 回路编号操作数超出范围，那么在编译时，CPU 将生成编译错误（范围错误），编译将失败。

PID 指令不检查某些回路表输入值是否超出范围。必须确保过程变量和设定值（以及用作输入的偏置和前一过程变量）是 0.0 到 1.0 之间的实数。

如果在执行 PID 计算的数学运算时发生任何错误，将置位 SM1.1（溢出或非法值），PID 指令将终止执行。（回路表中输出值的更新可能不完全，因此，在下一次执行回路的 PID 指令之前应忽略这些值并纠正引起数学运算错误的输入值。）

回路表

回路表的长度为 80 个字节，其格式如下表所示。

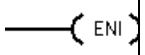
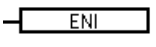
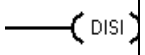
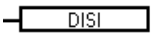
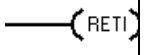
偏移	字段	格式	类型	说明
0	过程变量 (PV_n)	REAL	输入	包含过程变量，其值必须标定在 0.0 到 1.0 之间。
4	设定值 (SP_n)	REAL	输入	包含设定值，其值必须标定在 0.0 到 1.0 之间。
8	输出 (M_n)	REAL	输入/输出	包含计算出的输出，其值必须标定在 0.0 到 1.0 之间。
12	增益 (K_C)	REAL	输入	包含增益，为比例常数。可以是正数或负数。
16	采样时间 (T_S)	REAL	输入	包含采样时间，单位为秒。 必须是正数。
20	积分时间或复位 (T_I)	REAL	输入	包含积分时间或复位，单位为分。必须是正数。
24	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。必须是正数。
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 到 79	为自整定变量保留。 有关详细信息，请参见 PID 回路定义表 (页 610)。			

7.10 中断

7.10.1 中断指令

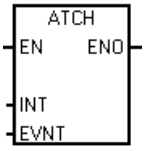
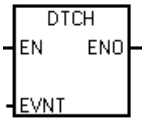
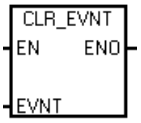
切换到 RUN 模式时，中断开始时被禁止。在 RUN 模式下，可通过执行 ENI（中断启用）指令来启用中断处理。执行 DISI（中断禁止）指令将禁止处理中断；但激活的中断事件将继续排队。

ENI、DISI 和 CRETI

LAD	FBD	STL	说明
		**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**	中断禁止指令全局性禁止对所有中断事件的处理。
		**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	说明
	**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 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。
	**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 CPU CR40、CR60：0-13、16-18、21-23、27、28 和 32 CPU SR20/ST20、SR30/ST30、SR40/ST40、SR60/ST60：0-13、16-28、32 和 34-38

7.10.2 中断例程概述和 CPU 型号事件支持

调用中断例程之前，必须在中断事件和该事件发生时希望执行的程序段之间分配关联。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，但一个事件不能同时连接到多个中断例程。

连接事件和中断例程时，仅当全局
ENI（中断启用）指令已执行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，该事件将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	说明	CR40/CR60	SR20/ST20 SR30/ST30 SR40/ST40 SR60/ST60
0	上升沿 I0.0	Y	Y
1	下降沿 I0.0	Y	Y
2	上升沿 I0.1	Y	Y
3	下降沿 I0.1	Y	Y
4	上升沿 I0.2	Y	Y
5	下降沿 I0.2	Y	Y
6	上升沿 I0.3	Y	Y
7	下降沿 I0.3	Y	Y

事件	说明	CR40/CR60	SR20/ST20 SR30/ST30 SR40/ST40 SR60/ST60
8	端口 0 接收字符	Y	Y
9	端口 0 发送完成	Y	Y
10	定时中断 0（SMB34 控制时间间隔）	Y	Y
11	定时中断 1（SMB35 控制时间间隔）	Y	Y
12	HSC0 CV=PV（当前值 = 预设值）	Y	Y
13	HSC1 CV=PV（当前值 = 预设值）	Y	Y
14-15	保留	N	N
16	HSC2 CV=PV（当前值 = 预设值）	Y	Y
17	HSC2 方向改变	Y	Y
18	HSC2 外部复位	Y	Y
19	PLS0 PTO 脉冲计数完成中断	N	Y
20	PLS1 PTO 脉冲计数完成中断	N	Y
21	定时器 T32 CT=PT（当前时间 = 预设时间）	Y	Y
22	定时器 T96 CT=PT（当前时间 = 预设时间）	Y	Y
23	端口 0 接收消息完成	Y	Y
24	端口 1 接收消息完成	N	Y
25	端口 1 接收字符	N	Y
26	端口 1 发送完成	N	Y
27	HSC0 方向改变	Y	Y
28	HSC0 外部复位	Y	Y
29-31	保留	N	N
32	HSC3 CV=PV（当前值 = 预设值）	Y	Y
33	保留	N	N
34	PLS2 PTO 脉冲计数完成中断	N	Y
35	上升沿，信号板输入 0	N	Y
36	下降沿，信号板输入 0	N	Y

事件	说明	CR40/CR60	SR20/ST20 SR30/ST30 SR40/ST40 SR60/ST60
37	上升沿，信号板输入 1	N	Y
38	下降沿，信号板输入 1	N	Y

7.10.3 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。

7.10.4 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 和
I7.1）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和定时器 T32/T96
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 和 T96 支持此类中断。否则 T32 和 T96 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）和 T96（事件 22）中断事件来启用这些中断。

7.10.5 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。其它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

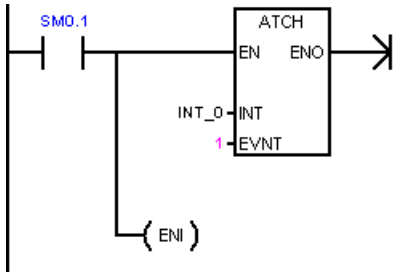
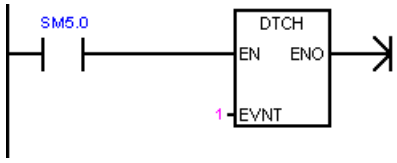
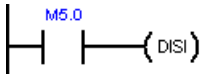
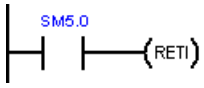
说明（0 = 无溢出，1 = 溢出）	SM 位
通信队列	SM4.0
I/O 中断队列	SM4.1
定时中断队列	SM4.2

中断事件的优先级顺序

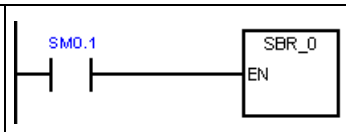
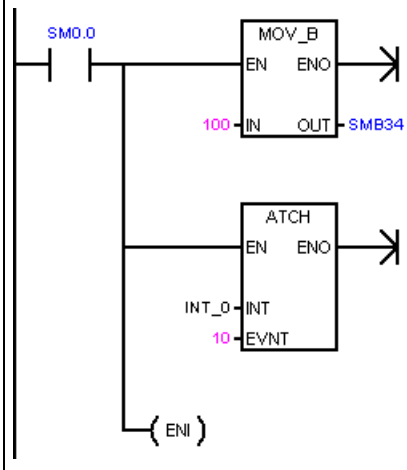
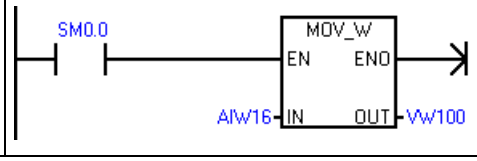
优先级组	事件	说明
通信 最高优先级	8	端口 0 接收字符
	9	端口 0 发送完成
	23	端口 0 接收消息完成
	24	端口 1 接收消息完成
	25	端口 1 接收字符
	26	端口 1 发送完成
离散 中等优先级	19	PLS0 脉冲计数完成
	20	PLS1 脉冲计数完成
	34	PLS2 脉冲计数完成
	0	I0.0 上升沿
	2	I0.1 上升沿
	4	I0.2 上升沿
	6	I0.3 上升沿
	35	I7.0 上升沿（信号板）
	37	I7.1 上升沿（信号板）
	1	I0.0 下降沿
	3	I0.1 下降沿
	5	I0.2 下降沿
	7	I0.3 下降沿
	36	I7.0 下降沿（信号板）
	38	I7.1 下降沿（信号板）

优先级组	事件	说明
	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）
	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）
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

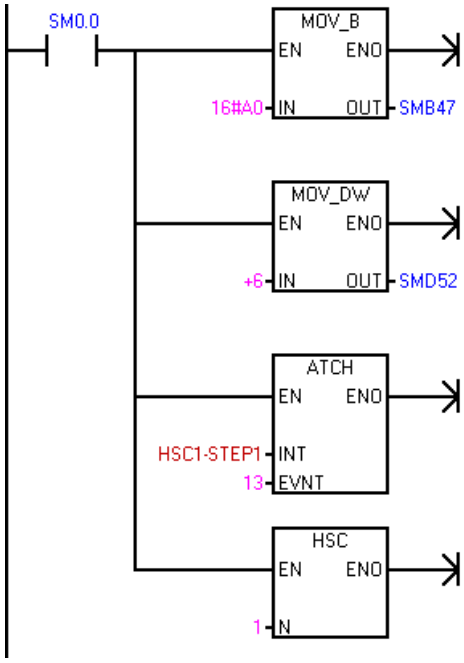
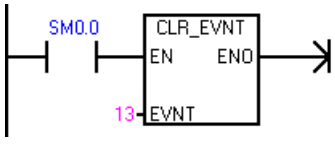
示例 1：输入信号沿检测器中断

LAD			STL
MAIN Network 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**
Network 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**
Network 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 Network 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

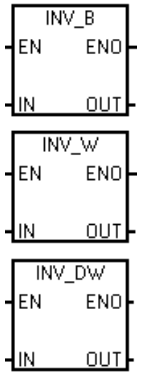
LAD			STL
MAIN Network 1		首次扫描时，调用子例程 0。	**Network 1** **LD SM0.1** **CALL SBR_0**
SBR 0 Network 1		将定时中断 0 的时间间隔设置为 100 ms。 将定时中断 0（事件 10）连接到 INT_0。 全局中断启用。	**Network 1** **LD SM0.0** **MOVB 100, SMB34** **ATCH INT_0, 10** **ENI**
INT 0 Network 1		每 100 ms 读取一次 AIW16 的值。	**Network 1** **LD SM0.0** **MOVW AIW16, VW100**

示例 3：清除中断事件指令

LAD			STL
SBR 1 Network 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 Network 2		清除机器振动引起的不必要中断。	**Network 2** **LD SM0.0** **CEVNT 13**

7.11 逻辑运算

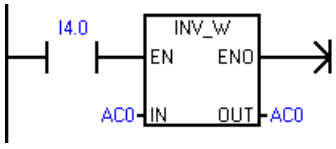
7.11.1 取反

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

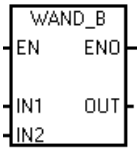
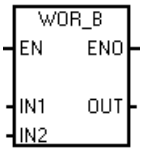
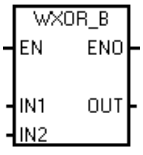
ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址	• SM1.0 运算结果 = 零

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

7.11.2 与、或和异或

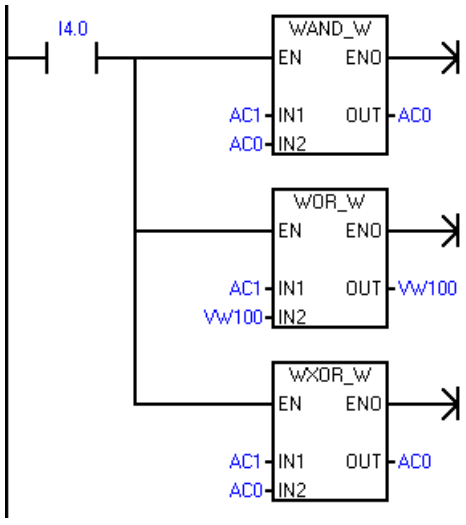
LAD/FBD	STL	说明
 WAND_W WAND_DW	**ANDB IN1, OUT** **ANDW IN1, OUT** **ANDD IN1, OUT**	字节与、字与和双字与指令对两个输入值 IN1 和 IN2 的相应位执行逻辑与运算，并将计算结果装载到分配给 OUT 的存储单元中。 • LAD 和 FBD： IN1 AND IN2 = OUT • STL： IN1 AND OUT = OUT
 WOR_W WOR_DW	**ORB IN1, OUT** **ORW IN1, OUT** **ORD IN1, OUT**	字节或、字或和双字或指令对两个输入值 IN1 和 IN2 的相应位执行逻辑或运算并，将计算结果装载到分配给 OUT 的存储单元中。 • LAD 和 FBD： IN1 OR IN2 = OUT • STL： IN1 OR OUT = OUT
 WXOR_W WXOR_DW	**XORB IN1, OUT** **XORW IN1, OUT** **XORD IN1, OUT**	字节异或、字异或和双字异或指令对两个输入值 IN1 和 IN2 的相应位执行逻辑异或运算，并将计算结果装载到存储单元 OUT 中。 • LAD 和 FBD： IN1 XOR IN2 = OUT • STL： IN1 XOR OUT = OUT

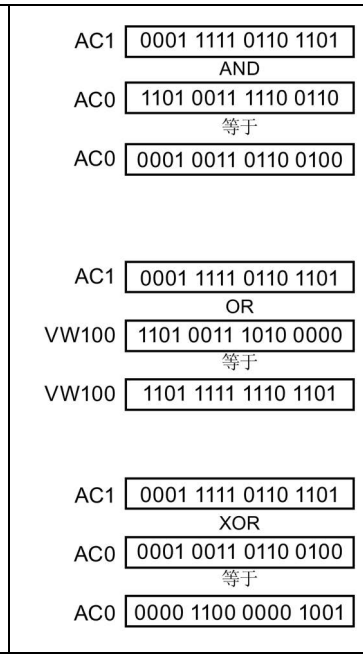
ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址	• SM1.0 运算结果 = 零

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

示例： 与、或和异或指令

LAD



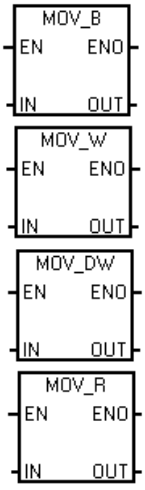


STL

```
Network 1
LD I4.0
ANDW AC1, AC0
ORW AC1, VW100
XORW AC1, AC0
```

7.12 传送

7.12.1 字节、字、双字或实数传送

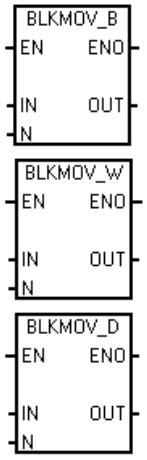
LAD/FBD	STL	说明
	**MOVB IN, OUT** **MOVW IN, OUT** **MOVD IN, OUT** **MOVR IN, OUT**	字节传送、字传送、双字传送和实数传送指令将数据值从源（常数或存储单元）IN 传送到新存储单元 OUT，而不会更改源存储单元中存储的值。 使用双字传送指令创建指针。 有关详细信息，请参见指针和间接寻址 (页 84)部分。

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

输入/输出	数据类型	操作数
	DWORD, DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

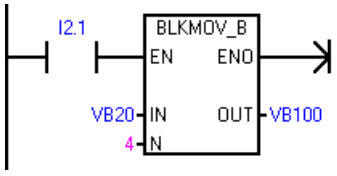
7.12.2 块传送

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。

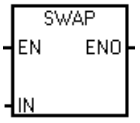
ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围	无

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

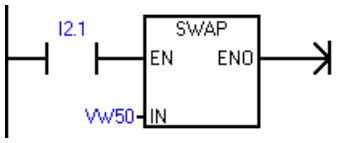
7.12.3 交换字节

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

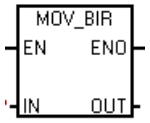
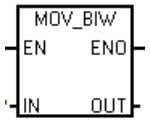
输入/输出	数据类型	操作数
IN	WORD	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、*VD、*LD、*AC

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

7.12.4 字节立即传送（读取和写入）

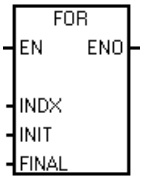
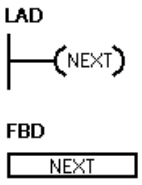
LAD/FBD	STL	说明
	**BIR IN, OUT**	移动字节立即读取指令读取物理输入 IN 的状态，并将结果写入存储器地址 OUT 中，但不更新过程映像寄存器。
	**BIW IN, OUT**	传送字节立即写入指令从存储器地址 IN 读取数据，并将其写入物理输出 OUT 以及相应的过程映像位置。

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	B、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数
OUT (BIR)	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC
OUT (BIW)	BYTE	QB、*VD、*LD、*AC

7.13 程序控制

7.13.1 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。
	**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。

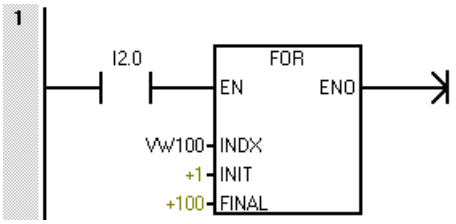
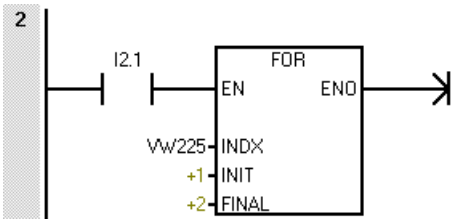
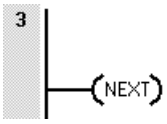
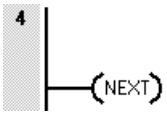
如果启用 FOR-NEXT 循环，则完成迭代操作之前会持续执行循环，除非在循环内部更改 FINAL 值。 在 FOR-NEXT 循环处于循环过程时可更改值。 再次启用循环时，会将 INIT 值复制到 INDX 值（当前循环编号）。

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

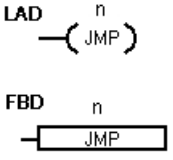
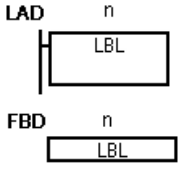
LAD		STL
1 	I2.0 接通时，执行 100 次外部循环 (Network 1 - 4)。	**Network 1** **LD I2.0** **FOR VW100, +1, +100**
2 	I2.1 接通时，每执行一次外部循环会执行两次内部循环 (Network 2 - 3)。	**Network 2** **LD I2.1** **FOR VW225, +1, +2**
3 	内部循环结束	**Network 3** **NEXT**
4 	外部循环结束	**Network 4** **NEXT**

7.13.2 JMP（跳转至标号）

可在主程序、子例程或中断例程中使用 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。

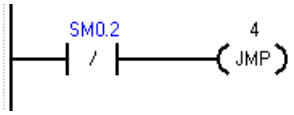
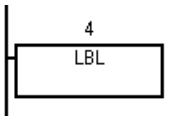
说明

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	说明
	`JMP N`	JMP（跳转）指令对程序中的标号 N 执行分支操作。
	`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
	如果保持性数据未丢失，则跳转至标号 4。	**Network 1** **LDN SM0.2** **JMP 4**
	标号 4	**Network 2** **LBL 4**

7.13.3 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。

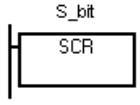
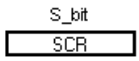
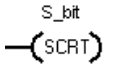
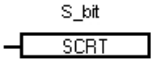
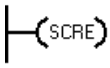
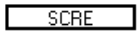
多个 POU 访问同一 S
位可能导致无法预料的过程操作，很可能会导致人员死亡或重伤。

请检查程序，确保多个 POU 不会访问的同一 S 位。

说明

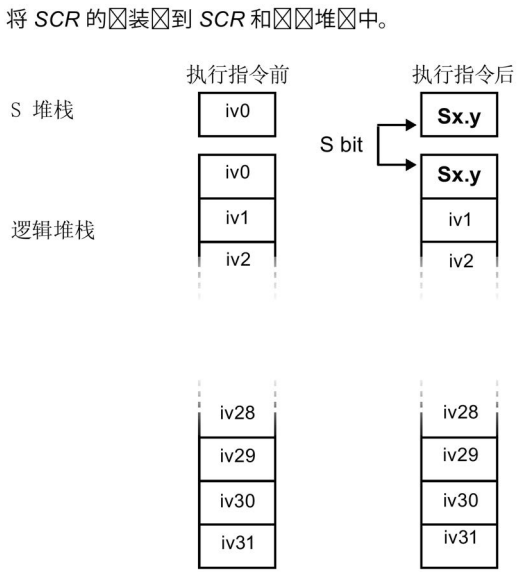
SCR 编程限制

- 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	说明
		**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。
		**SCRT S_bit**	
		**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。 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。
		**SCRE**	

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。

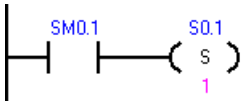
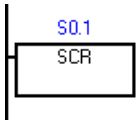
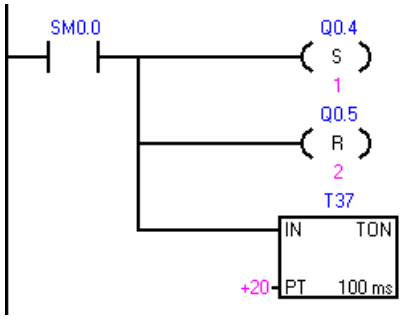
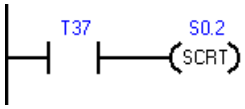
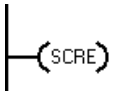
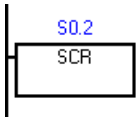
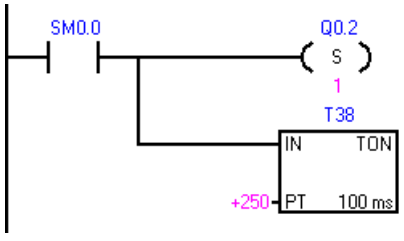
SCR 操作

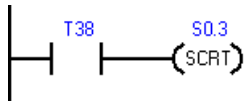
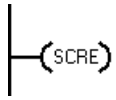
- SCR（装载 SCR）标记 SCR 段的开始， SCRE（结束 SCR）标记 SCR 程序段的结束。SCR 和 SCRE 指令之间的所有逻辑是否执行取决于 S 堆栈的值。SCRE 和下一条 SCR 指令之间的逻辑与 S 堆栈的值无关。
- SCRT（SCR 转换）将控制权从激活的 SCR 段转交给另一个 SCR 段。

  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

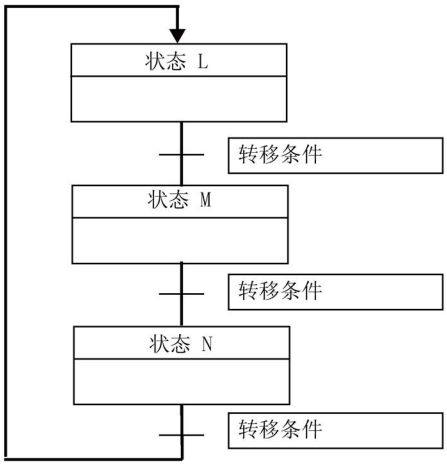
在以下示例程序中，首次扫描位 SM0.1 用于置位 S0.1，S0.1 在首次扫描时将是激活状态 1。2 秒延时后，T37 会导致转换为状态 2。该转换会禁用状态 1 SCR (S0.1) 段，并会激活状态 2 SCR (S0.2) 段。

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**

LAD		STL
	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**

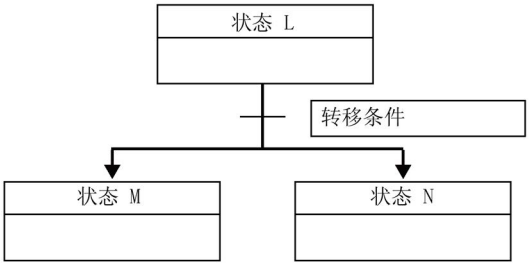
顺序控制流

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。



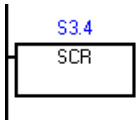
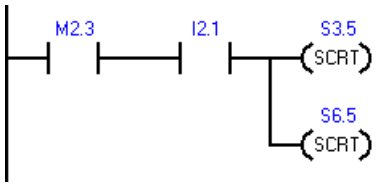
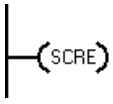
分散控制流

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。

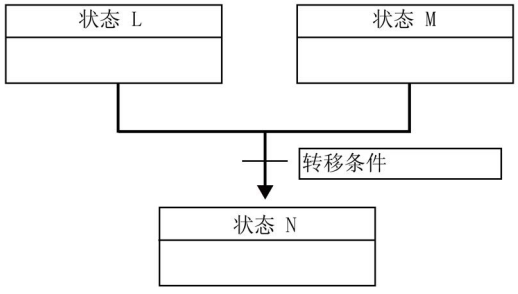
示例：SCR 分散流控制

LAD		STL
	状态 L 控制区域开始	**Network 1** **LSCR S3.4**
	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**
	状态 L 的状态区域结束	**Network 3** **SCRE**

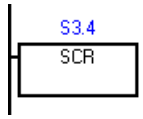
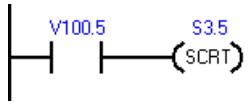
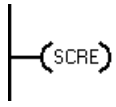
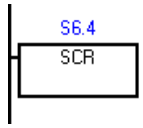
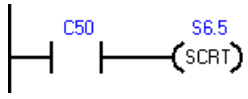
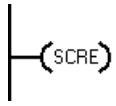
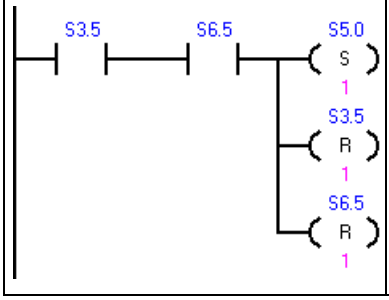
合并流控制

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。

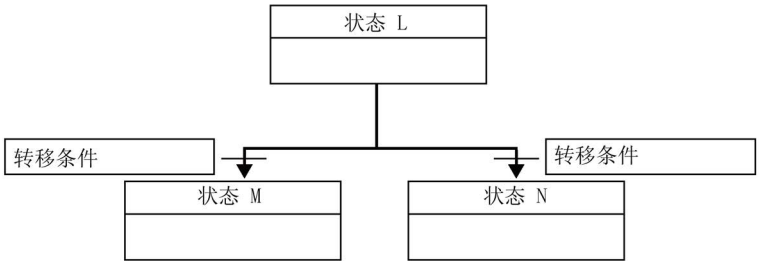


示例：SCR 合并流控制

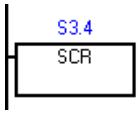
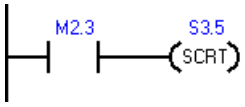
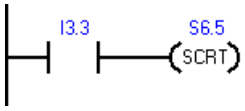
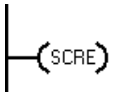
LAD		STL
	状态 L 控制区域开始	**Network 1** **LSCR S3.4**
	转换为状态 L'	**Network 2** **LD V100.5** **SCRT S3.5**
	状态 L 的 SCR 区域结束	**Network 3** **SCRE**
	状态 M 控制区域开始	**Network 4** **LSCR S6.4**
	转换为状态 M'	**Network 5** **LD C50** **SCRT S6.5**
	状态 M 的 SCR 区域结束	**Network 6** **SCRE**
	状态 L' 和状态 M' 同时激活时： • 启用状态 N (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

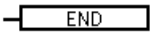
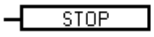
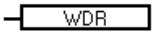
在其它情况下，控制流可能转到许多可能控制流之一，具体取决于哪个转换条件先变为真。



示例：SCR 分散流控制，具体取决于转换条件

LAD		STL
	状态 L 控制区域开始	**Network 1** **LSCR S3.4**
	转换为状态 M'	**Network 2** **LD M2.3** **SCRT S3.5**
	转换为状态 N	**Network 3** **LD I3.3** **SCRT S6.5**
	状态 L 的 SCR 区域结束	**Network 4** **SCRE**

7.13.4 END、STOP 和 WDR（看门狗定时器复位）

LAD	FBD	STL	说明
		**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，但不能在子例程或中断例程中使用。
		**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**	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。

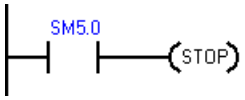
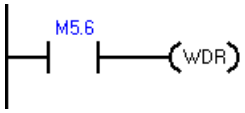
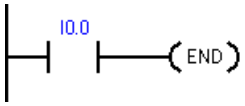
看门狗定时器操作

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
	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 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 接通时，终止当前扫描。	**Network 1** **LD I.0** **END**

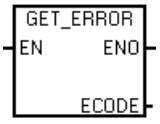
说明

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

7.13.5 GET_ERROR（获取非致命错误代码）

LAD/FBD	STL	说明
	**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

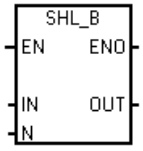
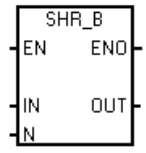
PLC 非致命错误代码 (页 844)

PLC 非致命错误 SM 标志 (页 847)

7.14 移位与循环移位

7.14.1 移位和循环移位

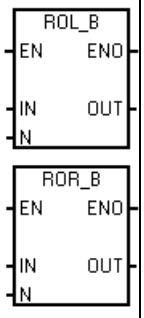
移位指令（仅说明大小为字节的 LAD 功能框，其它功能框类似）

LAD/FBD	STL	移位类型	说明
	**SLB OUT, N**	左移字节	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。
	**SRB OUT, N**	右移字节	
SHL_W	**SLW OUT, N**	左移字	
SHR_W	**SRW OUT, N**	右移字	
SHL_DW	**SLD OUT, N**	左移双字	
SHR_DW	**SRD OUT, N**	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	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*LD、*AC、常数
	DWORD	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	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
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，其它功能框类似）

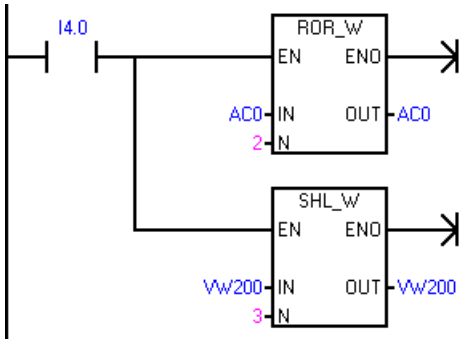
LAD/FBD	STL	循环移位类型	说明
	**RLB OUT, N**	循环左移字节	循环移位指令将输入值 IN 的位值循环右移或循环左移位位置循环移位计数 N，然后将结果装载到分配给 OUT 的存储单元中。 循环移位操作为循环操作。 如果循环移位计数大于或等于操作的最大值（字节操作为 8、字操作为 16、双字操作为 32），则 CPU 会在执行循环移位前对移位计数执行求模运算以获得有效循环移位计数。 该结果为移位计数，字节操作为 0 至 7，字操作为 0 至 15，双字操作为 0 至 31。 如果循环移位计数为 0，则不执行循环移位操作。 如果执行循环移位操作，则溢出位 SM1.1 将置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。
	RRB OUT, N	循环右移字节	
ROL_W	**RLW OUT, N**	循环左移字	
ROR_W	**RRW OUT, N**	循环右移字	
ROL_DW	**RLD OUT, N**	循环左移双字	
ROR_DW	**RRD OUT, N**	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	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*LD、*AC、常数
	DWORD	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	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
N	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数

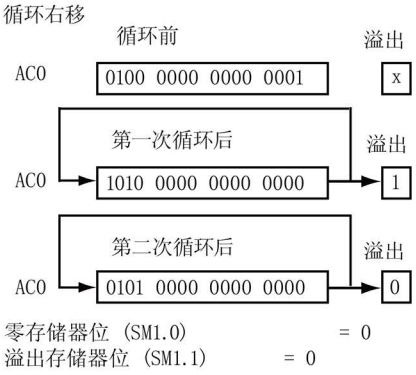
示例： 移位和循环移位指令

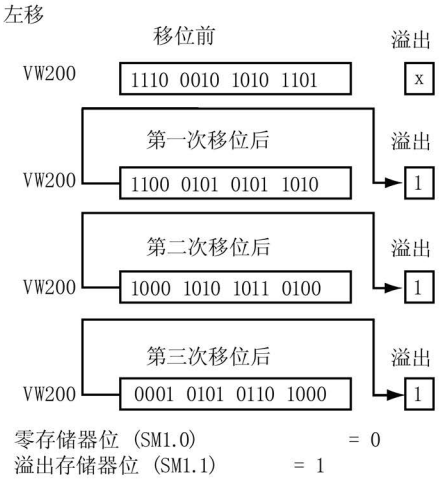
LAD



STL

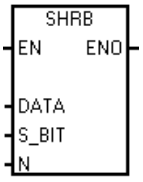
```
Network 1
LD I4.0
RRW AC0, 2
SLW VW200, 3
```





7.14.2 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 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。

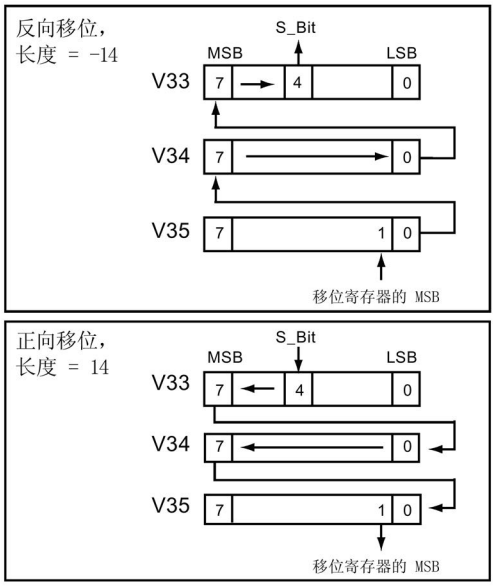
MSB.b = V33 + ([14] - 1 +4)/8

= 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。

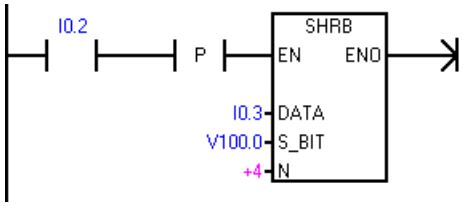


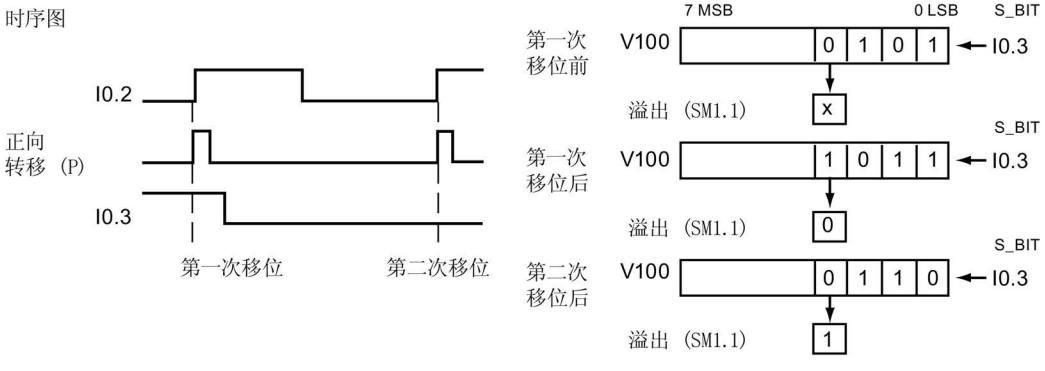
反向移位操作用长度 N 的负值表示。将 DATA 的输入值移入移位寄存器的最高有效位，然后移出由 S_BIT
指定的最低有效位位置。然后将移出的数据放在溢出存储器位 SM1.1 中。

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

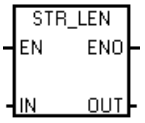
LAD	STL
	**Network 1** **LD I0.2** **EU** **SHRB I0.3, V100.0, +4**



7.15 字符串

7.15.1 字符串（获取长度、复制和连接）

SLEN（字符串长度）

LAD/FBD	STL	说明
	**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	说明
	**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。
	**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。

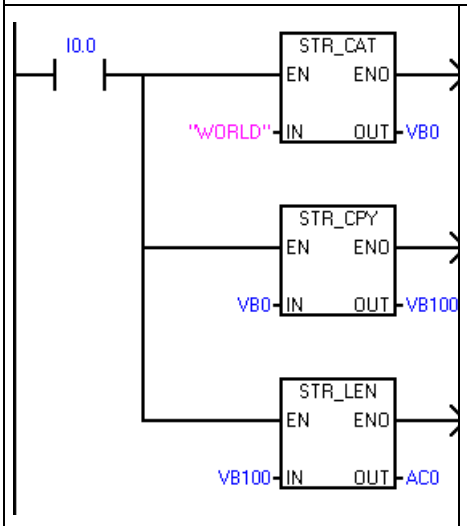
注： STR_CPY 和 STR_CAT 指令作用对象是字节而不是字符。 因为中文字符并非由单字节表示，所以 STR_CPY 和 STR_CAT 指令作用于包含中文字符的字符串时，可能出现非预期的结果。
如果您知道字符包含的字节数，则可以在使用 STR_CPY 和 STR_CAT 指令时使用正确的字节数。

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

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

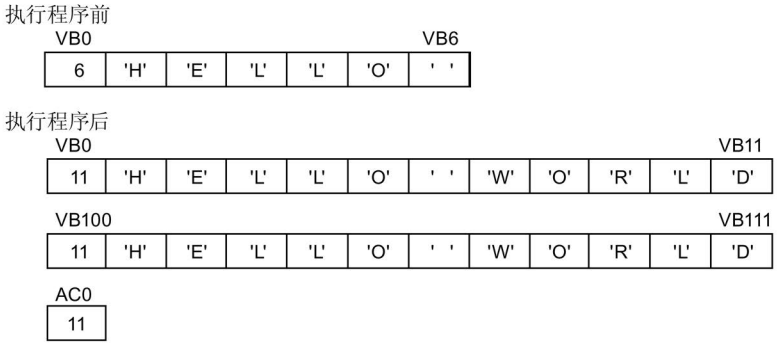
LAD



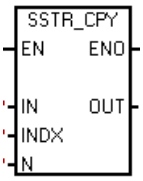
1. 将字符串“WORLD”附加到 VB0 中的字符串。
2. 将 VB0 中的字符串复制到 VB100 中的新字符串。
3. 获取从 VB100 开始的字符串的长度。

STL

```
Network 1
LD I0.0
SCAT "WORLD", VB0
SCPY VB0, VB100
SLEN VB100, AC0
```



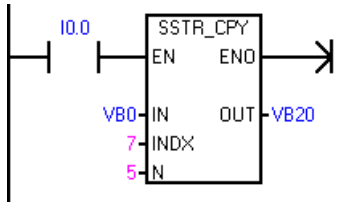
7.15.2 从字符串中复制子字符串

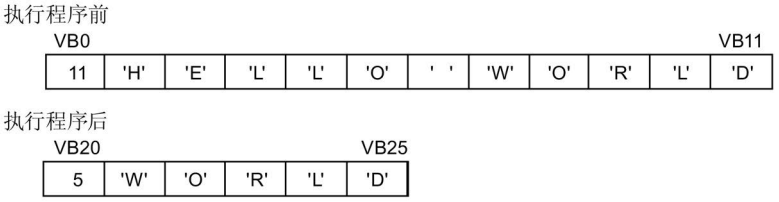
LAD/FBD	STL	说明
	**SSCPY IN, INDX, N, OUT**	从字符串中复制子字符串指令从 IN 指定的字符串中将从索引 INDX 开始的指定数目的 N 个字符复制到 OUT 指定的新字符串中。 注： SSTR_CPY 指令作用对象是字节而不是字符。 因为中文字符并非由单字节表示，所以 SSTR_CPY 指令作用于包含中文字符的字符串时，可能出现非预期的结果。 如果您知道字符包含的字节数，则可以在使用 SSTR_CPY 指令时使用正确的字节数。

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

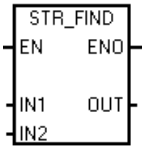
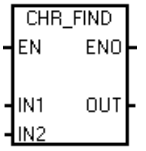
输入/输出	数据类型	操作数
IN	STRING	VB, LB, *VD, *LD, *AC, 常数字符串
OUT	STRING	VB, LB, *VD, *LD, *AC
INDX, N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC、常数

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**



7.15.3 在字符串中查找字符串和第一个字符

LAD/FBD	STL	说明
	**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。
	**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。

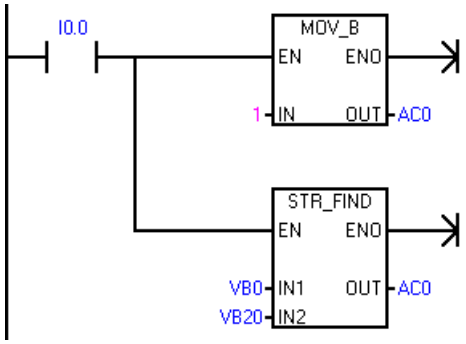
ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围 • 009BH 索引 = 0	无

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)。

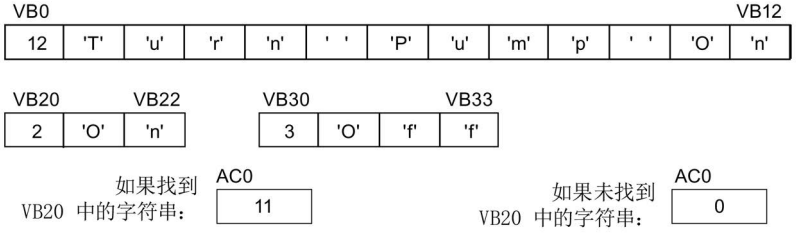
LAD



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”）。

STL

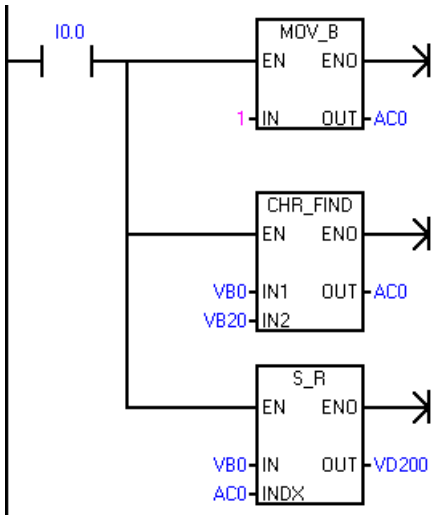
```
Network 1
LD I0.0
MOVB 1, AC0
SFND VB0, VB20, AC0
```



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。

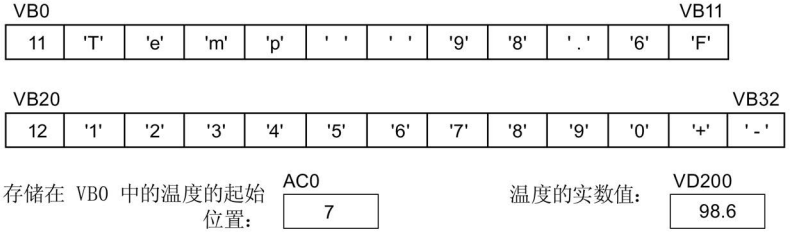
LAD



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
```

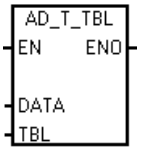


存储在 VB0 中的温度的起始位置：AC0 = 7

温度的实数值：VD200 = 98.6

7.16 表

7.16.1 添表

LAD/FBD	STL	说明
	**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

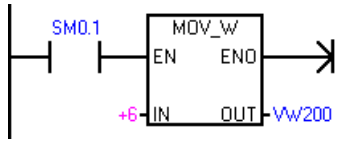
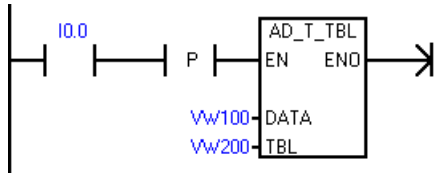
输入/输出	数据类型	操作数
DATA	INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*LD、*AC、常数
TBL	WORD	IW、QW、VW、MW、SMW、SW、T、C、LW、*VD、*LD、*AC

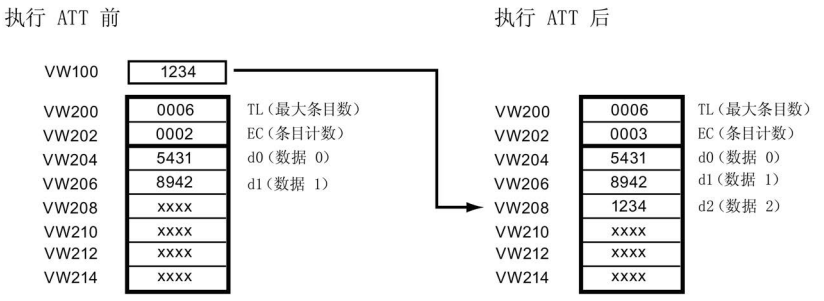
说明

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。

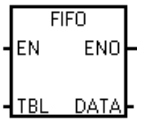
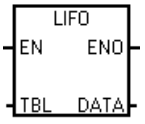
示例： 添表指令

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**



7.16.2 先进先出和后进先出

表格 7- 20 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

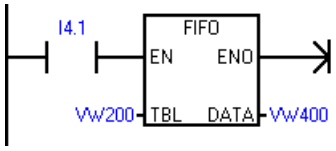
输入/输出	数据类型	操作数
TBL	WORD	IW、QW、VW、MW、SMW、SW、T、C、LW、*VD、*LD、*AC
DATA	INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AQW、*VD、*LD、*AC

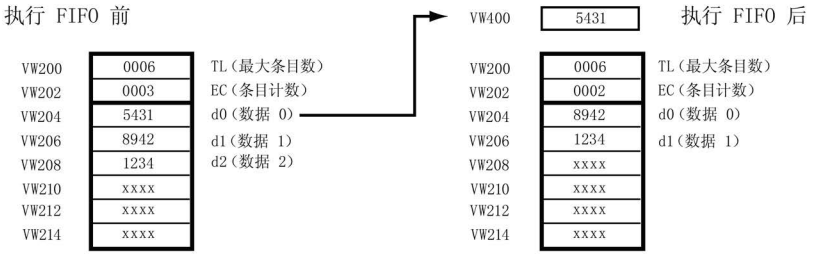
说明

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。

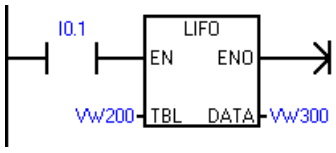
要创建表格，必须先创建用于表示最大表格条目数的条目，然后才能将其它条目放入表格中。

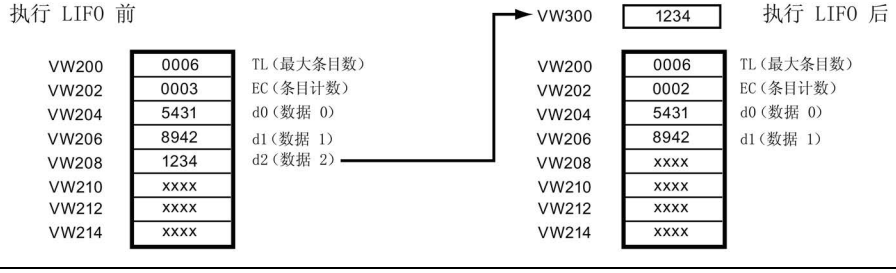
示例： FIFO 指令

LAD	STL
	**Network 1** **LD I4.1** **FIFO VW200, VW400**

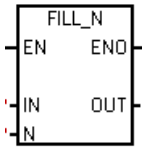


示例： LIFO 指令

	**Network 1** **LD I0.1** **LIFO VW200, VW300**



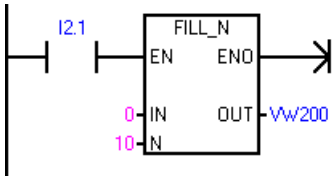
7.16.3 存储器填充

LAD/FBD	STL	说明
	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。

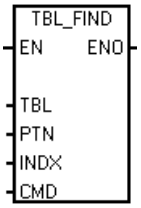
ENO = 0 时的非致命错误	受影响的 SM 位
• 0006H 间接地址 • 0091H 操作数超出范围	无

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
	

7.16.4 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 (>)

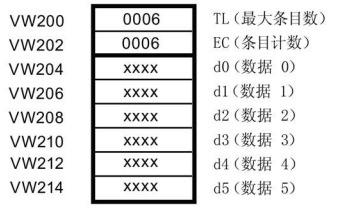
说明

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。

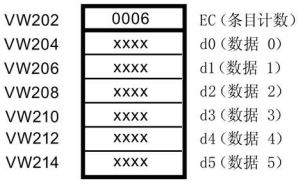
因此，应将查找指令的 TBL
操作数的地址设置得比相应的添表指令、后进先出指令或先进先出指令的 TBL
操作数高一个字（两个字节）。

ATT、LIFO、FIFO 和 TBL_FIND 指令的表格格式差异

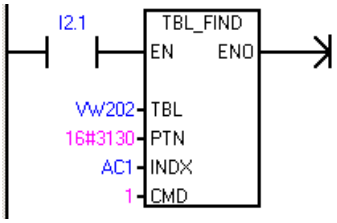
ATT、LIFO 及 FIFO 的表格式



TBL_FIND 的表格式



示例：查表指令

LAD	STL
	**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。

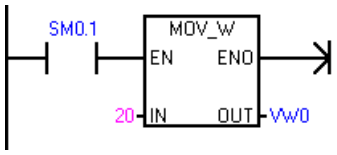
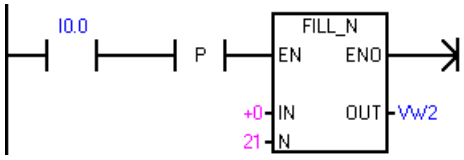
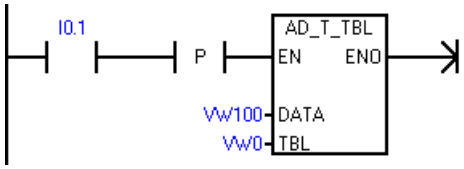
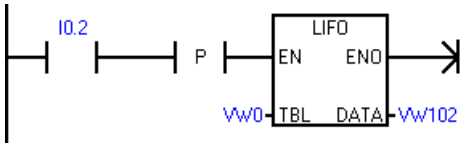
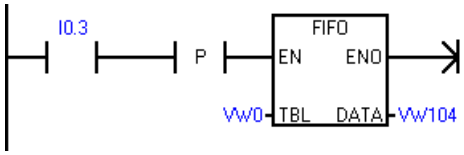
AC1	0	AC1 必须设为 0 才能从表顶部进行搜索。
执行表搜索 AC1	2	AC1 中包含的数据条目编号对应于表中的第一个匹配条目 (d2)。
AC1	3	先将 INDX 值加 1，然后再搜索表中的剩余条目。
执行表搜索 AC1	4	AC1 中包含的数据条目编号对应于表中的第二个匹配条目 (d4)。
AC1	5	先将 INDX 值加 1，然后再搜索表中的剩余条目。
执行表搜索 AC1	6	AC1 包含的值等于条目计数。 已搜索整个表，未发现其它匹配条目。
AC1	0	再次搜索表之前，必须先将 INDX 值复位为 0。

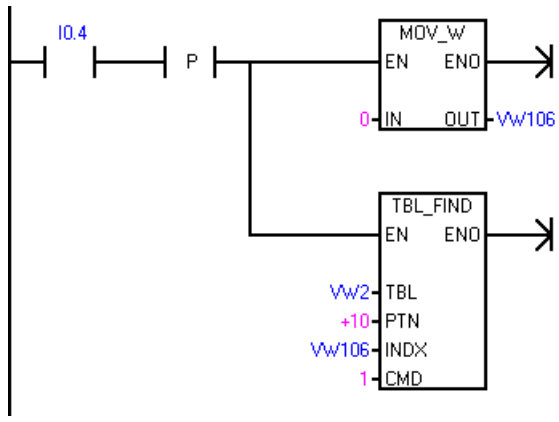
示例：表格

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。

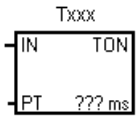
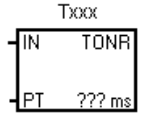
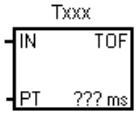
要搜索表格，在执行查找操作之前，必须将索引 (VW106) 设置为 0。如果找到匹配条目，索引存储该表格条目编号；如果未找到匹配条目，则索引为表格的当前条目计数 (VW2)。

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**
	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**

LAD		STL
	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**

7.17 定时器

7.17.1 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 T255）
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
	10 ms	327.67 s	T33 - T36, T97 - T100
	100 ms	3276.7 s	T37 - T63, T101 - T255
TONR	1 ms	32.767 s	T0、T64
	10 ms	327.67 s	T1 - T4、T65 - T68
	100 ms	3276.7 s	T5 - T31、T69 - T95

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]
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 (页 142)。

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。

7.17.2 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 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。

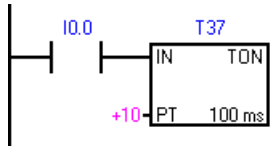
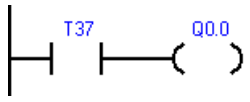
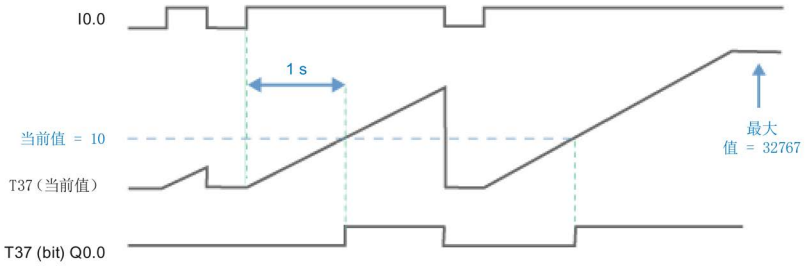
100 毫秒分辨率

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。

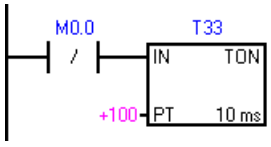
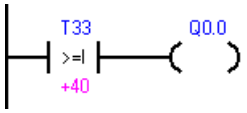
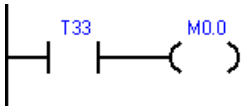
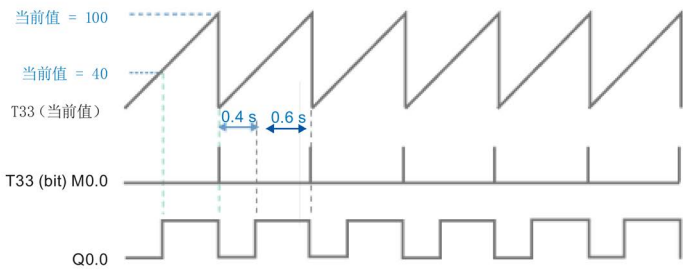
只有在执行定时器指令时，才对 100 毫秒定时器的当前值进行更新。因此，如果启用了 100
毫秒定时器但在各扫描周期内并未执行定时器指令，则不能更新该定时器的当前值并将丢失时间。同样，如果在一个扫描周期内多次执行同一条 100 毫秒定时器指令，则将 100 毫秒间隔数多次加到定时器的当前值，这延长了时间。只有在每个扫描周期仅执行一次定时器指令时，才应该使用 100 毫秒定时器。

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。

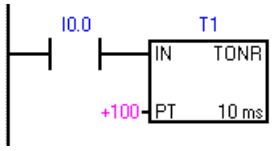
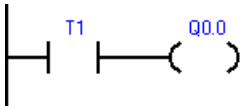
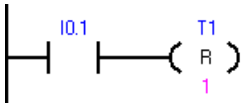
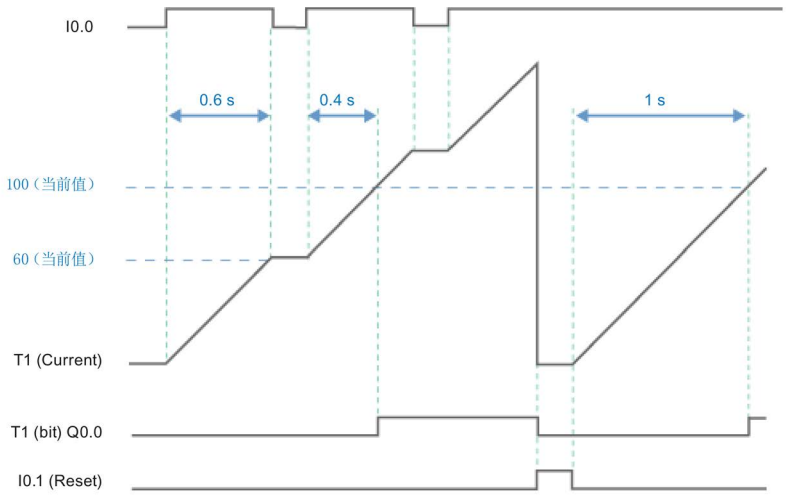
TON 接通延时定时器示例

<table>
<tr><th colspan="2">LAD</th><th>STL</th></tr>
<tr><td> 
</td><td>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</td><td>Network 1
LD I0.0
TON T37, +10</td></tr>
<tr><td> 
</td><td>T37 位由定时器 T37 控制</td><td>Network 2
LD T37
= Q0.0</td></tr>
<tr><td colspan="3">时序图
 
</td></tr>
</table>

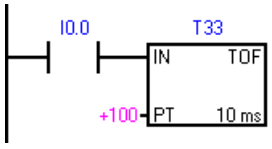
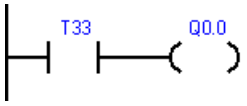
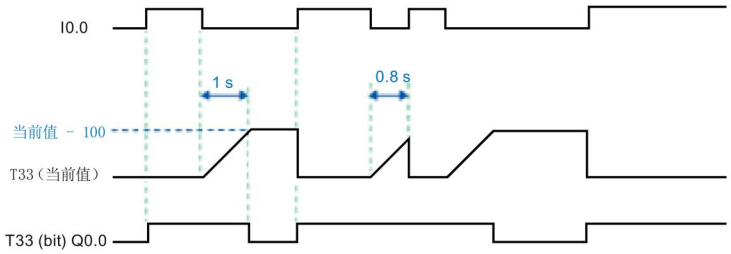
TON 自动复位接通延时定时器示例

LAD		STL
	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**
	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**
	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

<table>
<tr><th>LAD</th><th></th><th>STL</th></tr>
<tr><td> 
</td><td>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</td><td>Network 1
LD I0.0
TOF T33, +100</td></tr>
<tr><td> 
</td><td>定时器 T33 通过定时器触点 T33 控制 Q0.0。</td><td>Network 2
LD T33
= Q0.0</td></tr>
<tr><td colspan="3">时序图
 
</td></tr>
</table>

定时器分辨率对定时器位和当前时间值更新时间的影响

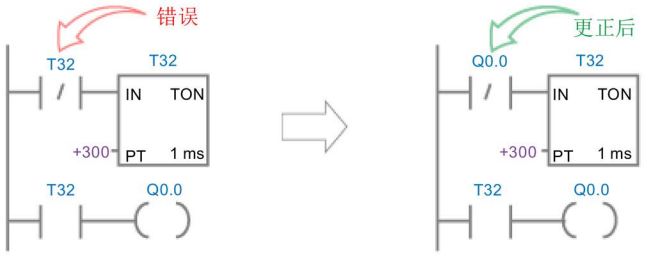
- **1 ms** 定时器：定时器位和当前值的更新与扫描周期不同步。扫描周期大于 1 ms 时，定时器位和当前值在该扫描周期内更新多次。
- **10 ms**
  定时器：定时器位和当前值在每个扫描周期开始时更新。定时器位和当前值在整个扫描期间保持不变。扫描期间累积的时间间隔会在每次扫描开始时加到当前值上。
- **100 ms** 定时器：对于分辨率为 100 ms
  的定时器，定时器位和当前值在指令执行时更新；因此，确保在每个扫描周期内程序仅执行 100 ms 定时器指令一次，这样才能保证定时器的定时正确。

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。

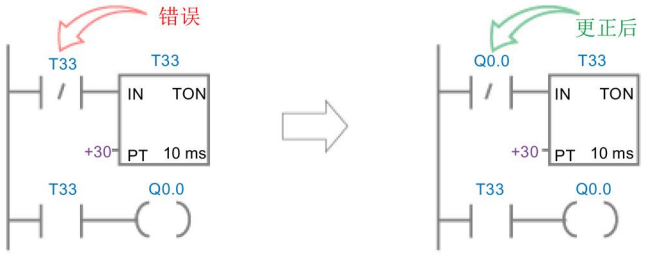
1 ms 定时器

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。



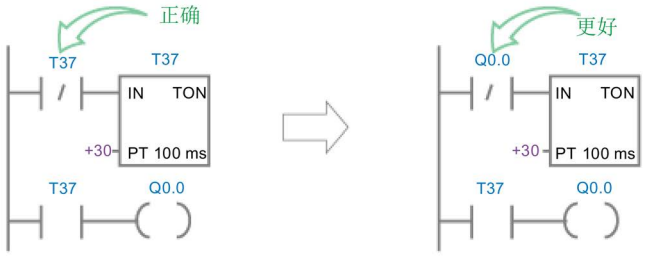
10 ms 定时器

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。



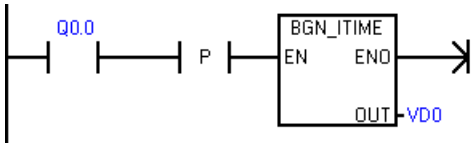
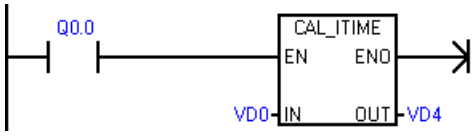
7.17.3 时间间隔定时器

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
 Ex1_Interval_time_net1	捕捉 Q0.0 接通的时刻。	**Network 1** **LD Q0.0** **EU** **BITIM VD0**
	计算 Q0.0 接通的时长。	**Network 2** **LD Q0.0** **CITIM VD0, VD4**

7.18 子例程

7.18.1 CALL（子例程）和 RET（有条件返回）

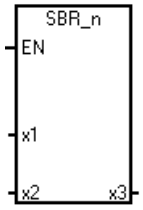
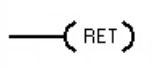
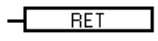
要添加新子例程，请选择**“编辑”(Edit)** 功能区，然后选择**“插入对象”(Insert Object)** 和**“子例程”(Subroutine)** 命令。 STEP 7-Micro/WIN SMART 自动在每个子例程中添加一个无条件返回。 还可以在子例程中添加有条件返回 CRET 指令。

在主程序中，可以嵌套调用子例程（在子例程中调用子例程），最大嵌套深度为八。

在中断例程中，可嵌套的子例程深度为四。

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
	**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 16 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，堆栈其它位置的值设置为零，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:  FBD: 	**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、S、T、C、L、能流 (LAD)、逻辑流 (FBD)
	BYTE	VB、IB、QB、MB、SMB、SB、LB、AC、*VD、*LD、*AC[1]、常数
	WORD, INT	VW、T、C、IW、QW、MW、SMW、SW、LW、AC、AIW、*VD、*LD、*AC[1]、常数
	DWORD, DINT	VD、ID、QD、MD、SMD、SD、LD、AC、HC、*VD、*LD、*AC[1]、&VB、&IB、&QB、&MB、&T、&C、&SB、&AI、&AQ、&SMB、常数
	STRING	*VD、*LD、*AC[1]、常数
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]
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]

[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

带调用参数调用子例程

子例程可选择使用传递参数。这些参数在子例程的变量表中定义。必须为每个参数分配局部符号名称（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	说明
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

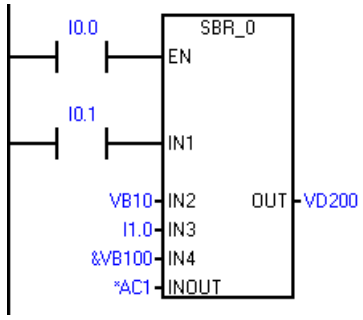
- 能流：布尔能流仅允许用于位（布尔）输入。此声明将输入参数分配给基于位逻辑指令组合的能流结果。能流输入与 EN 输入相似，都与位逻辑（例如，LAD 触点）相连接，而不连接到直接/间接地址分配。必须在变量表的最上一行（或多行）指定布尔能流输入，然后再指定任何非布尔数据类型。只有输入参数可以这样使用。下例中的使能输入 (EN) 和 IN1 输入使用能流逻辑。
- BOOL：此数据类型用于单个位输入和输出。下例中的 IN3 是分配给直接地址的布尔输入。
- BYTE、WORD、DWORD：这些数据类型分别标识 1、2 或 4 字节的无符号输入或输出参数。
- INT、DINT：这些数据类型分别标识 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

	地址	符号	变量类型	数据类型	注释
1		EN	IN	BOOL	
2	LW0	Lvl	IN	INT	罐发送器
3	LW2	Hset	IN	INT	高设定值
4	LW4	Offset	IN	INT	偏移量
5			IN		
6			IN_OUT		
7	LW6	Result	OUT	INT	结果值
8	L8.0	Valve	OUT	BOOL	控制阀
9			OUT		

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。如果参数中遗漏了对于常数的说明，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。

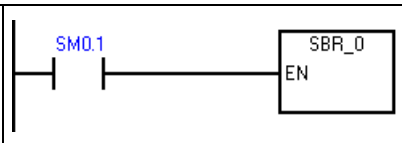
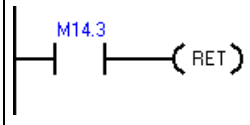
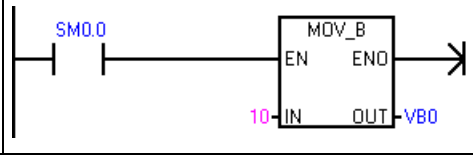
数据元素大小和类型用参数的编码表示。参数值到子例程中的局部存储器的分配如下所述：

- 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

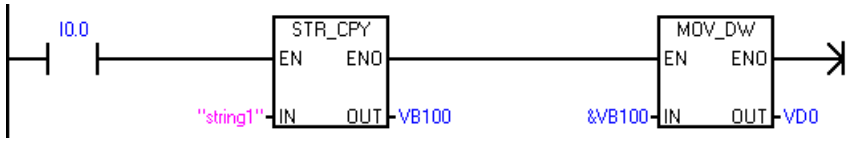
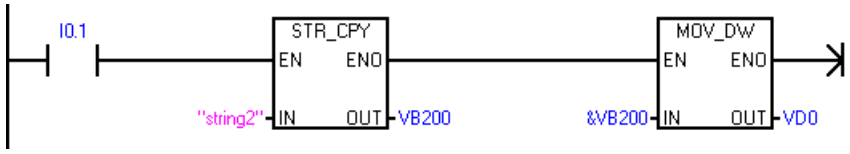
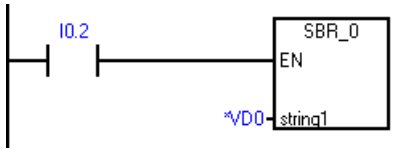
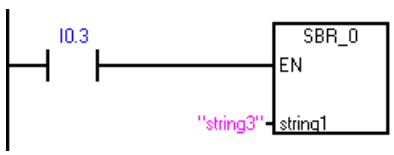
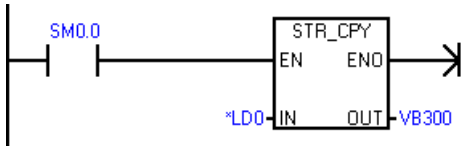
示例：子例程和子例程返回指令

LAD			STL
MAIN		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		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		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它位置。

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		Network 1 LD I0.0 SCPY "string1", VB100 AENO MOVD &VB100, VD0
MAIN		Network2 LD I0.1 SCPY "string2", VB200 AENO MOVD &VB200, VD0
MAIN		Network3 LD I0.2 CALL SBR_0, *VD0
MAIN		Network4 LD I0.3 CALL SBR_0, "string3"
SBR0		Network 1 LD SM0.0 SSCPY *LD0, VB300

通信

8

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 与其它 S7-200 SMART CPU 的对等通信
  - 与其它具有以太网功能的设备间的开放式用户通信 (OUC)
- PROFIBUS：
  - 适用于分布式 I/O 的高速通信（高达 12 Mbps）
  - 一个总线控制器连接许多 I/O 设备（支持 126 个可寻址设备）。
  - 主站和 I/O 设备间的数据交换
  - EM DP01 模块是 PROFIBUS I/O 设备。
- RS485：
  - 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）

8.1 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 CPU 或其它以太网设备。
  - 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 CPU 或网络设备。

说明

S7-200 SMART CPU 使用 GET 和 PUT 指令进行 CPU 到 CPU 的通信。

- PROFIBUS 端口：每个 EM DP01 PROFIBUS DP 模块可支持六个连接，其中一个预留给 HMI 设备。
- 集成的 RS485 端口（端口 0）：四个支持 HMI 设备的连接。
- CM01 信号板 (SB) RS232/RS485 端口（端口 1）：四个支持 HMI 设备的连接。

说明

STEP 7-Micro/WIN SMART 仅能通过以太网端口连接到 S7-200 SMART CPU。一个 PG 一次只能监视一个 CPU。

RS485 和 RS232 端口仅适用于 HMI 访问（数据读/写）和自由端口通信。

8.2 CPU 通信端口

S7-200 SMART CPU 上有四个通信接口，提供了以下通信类型：

- 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)
- PROFIBUS 端口：S7-200 SMART CPU 可支持两个 EM DP01 模块进行 PROFIBUS DP 与 HMI 通信。
- RS485 端口（端口 0）：
  - TD/HMI：RS485 类型
  - 自由端口 (XMT/RCV) 包括 Siemens 提供的 USS 和 Modbus RTU 库
- RS485/RS232 信号版 (SB)（如存在，端口 1）：
  - 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 RS432）库

8.3 HMI 和通信驱动程序

HMI

S7-200 SMART 支持以下 Siemens HMI 系列中的 HMI：

- **COMFORT HMI：**
  - SIMATIC HMI TP700 COMFORT
  - SIMATIC HMI TP900 COMFORT
  - SIMATIC HMI TP1200 COMFORT
  - SIMATIC HMI KP400 COMFORT
  - SIMATIC HMI KP700 COMFORT
  - SIMATIC HMI KP900 COMFORT
  - SIMATIC HMI KP1200 COMFORT
  - SIMATIC HMI KTP400 COMFORT
- **SMART HMI：**
  - SMART 700 IE
  - SMART 1000 IE
- **BASIC HMI：**
  - SIMATIC HMI KTP400 BASIC MONO PN
  - SIMATIC HMI KTP600 BASIC MONO PN
  - SIMATIC HMI KTP600 BASIC COLOR DP
  - SIMATIC HMI KTP600 BASIC COLOR PN
  - SIMATIC HMI KTP1000 BASIC COLOR DP
  - SIMATIC HMI KTP1000 BASIC COLOR PN
  - SIMATIC HMI TP1500 BASIC COLOR PN
  - SIMATIC HMI KP300 BASIC MONO PN
- **Micro HMI：**
  - TD 400C 文本显示，4 行

通信驱动程序

可在两个位置选择 S7-200 SMART HMI 的通信驱动程序：

- WinCC Flexible 软件
- TIA 门户

WinCC Flexible

在 WinCC Flexible 软件包中，可使用以下菜单选项来选择所需的“通信驱动程序”：

- 通信
- 连接表

在“连接表”(Connections table) 中，选择“SIMATIC S7 200 SMART”驱动程序。
如果列表中没有 SMART 驱动程序，则选择“SIMATIC S7 200”驱动程序。

TIA 门户

在 TIA 门户中，可使用以下菜单选项来选择所需的“通信驱动程序”：

- HMI 变量
- 连接

在“连接”(Connections) 中，选择“SIMATIC S7 200”驱动程序。

说明

如果 HMI 面板正在使用 DP 连接 (RS485)，则还要将“网络配置文件”(Network Profile) 设置为 PPI。

8.4 以太网

8.4.1 概述

以太网是一种差分（多点）网络，最多可有 32 个网段、1,024 个节点。以太网可实现高速（高达 100 Mbit/s）长距离（铜缆：最远约为 1.5km；光纤：最远约为 4.3km）数据传输。

可能的以太网连接包括针对以下设备的连接：

- 编程设备
- CPU 间的 GET/PUT 通信
- HMI 显示器
- 开放式用户通信 (OUC)

8.4.2 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 或其它设备）
- 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。

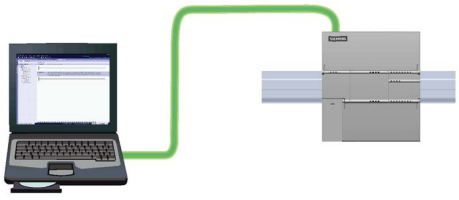
如果连接终止（例如，因断线或其中一个伙伴断开连接），主动伙伴将尝试重新建立连接。
被动设备也将注意到连接出现终止并采取行动（例如，撤销新断开连接的主动伙伴的密码权限）。

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。

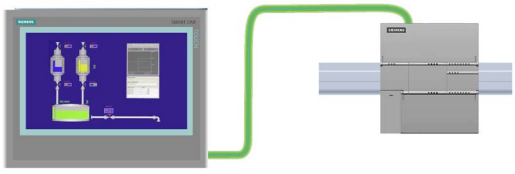
8.4.3 以太网网络组态示例

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



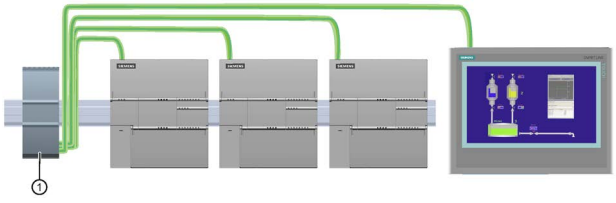
CPU 连接到编程设备



CPU 连接到 HMI

CPU 连接到另一个 CPU

CPU 上的以太网端口不包含以太网交换设备。 编程设备或 HMI 与 CPU 之间的直接连接不需要以太网交换机。 不过，含有两个以上的 CPU 或 HMI 设备的网络需要以太网交换机。



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。

8.4.4 分配 Internet 协议 (IP) 地址

8.4.4.1 为编程设备和网络设备分配 IP 地址

如果编程设备使用板载适配器卡连接到工厂 LAN（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。

使用桌面上的“网上邻居”(My Network Places) 分配或检查编程设备的 IP 地址

如果您使用的是 Windows 7
操作系统，您就可以通过以下菜单选项来分配或检查编程设备的 IP 地址：

- “启动”(Start)
- “控制面板”(Control Panel)
- “网络和共享中心”(Network and Sharing Center)
- 连接至 CPU 的网络适配器的“本地连接”(Local Area Connection)
- “属性”(Properties)
- 在“本地连接属性”(Local Area Connection Properties)
  对话框的“此连接使用下列项目：”(This connection uses the following items:) 字段中：
  - 向下滚动到“Internet 协议版本 4 (TCP/IP4)”(Internet Protocol Version 4 (TCP/IPv4))。
  - 单击“Internet 协议版本 4 (TCP/IP4)”(Internet Protocol Version 4 (TCP/IPv4))。
  - 单击“属性”(Properties) 按钮。
  - 选择“自动获得 IP 地址 (DCP)”(Obtain an IP address automatically (DCP))
    或“使用下面的 IP 地址”(Use the following IP address)（可输入静态 IP 地址）。
- 如果已选中“自动获得 IP 地址”(Obtain and IP address automatically)，则您可能需要更改为“使用下面的 IP 地址”(Use the following IP address) 选项以连接到 S7-200 SMART CPU：
  - 选择与 CPU 属于同一子网的 IP 地址 (**192.168.2**.1)。
  - 将 IP 地址设置为具有相同网络 ID 的地址（例如，**192.168.2**.200）。
  - 选择子网掩码 **255.255.255**.0。
  - 将默认网关留空。
  - 这使您能够连接到 CPU。

  说明

  网络接口卡和 CPU 必须位于同一子网，这样 STEP 7-Micro/WIN SMART 才能找到 CPU 并与之通信。

请咨询 IT 人员帮助您设置网络组态，以连接 S7-200 SMART CPU。

8.4.4.2 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

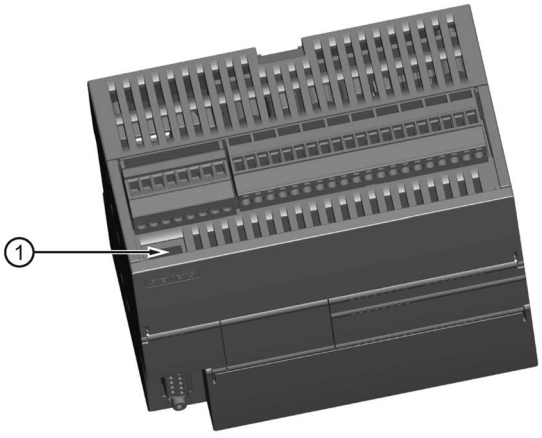
必须为网络上的每台设备设定一个唯一的 IP 地址。

- “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它网络发送消息，这些网络可能还隐含着其它 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 以太网端口

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，不能通过其它方式更改”(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，不能通过其它方式更改”(IP address data is fixed to the values below and cannot be changed by other means) 复选框，则可通过其它方式更改 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：

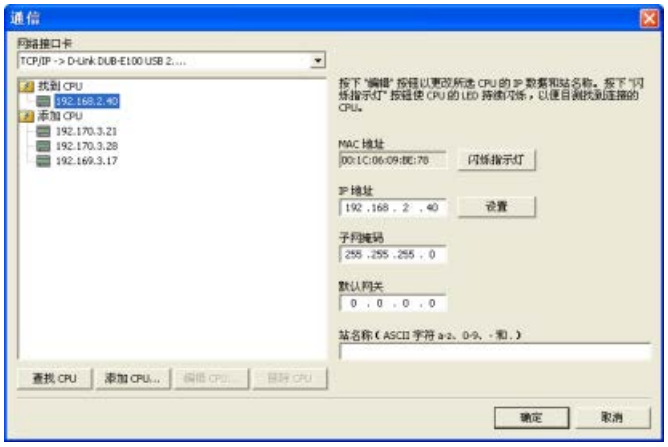
- 在导航栏中单击“通信”(Communications) 按钮。
- 在项目树中，选择“通信”(Communications) 节点，然后按下 Enter，或双击“通信”(Communications) 节点。

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“网络接口卡”(Network Interface Card) 下拉列表，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（“找到的 CPU”）。所有 CPU 都有默认 IP 地址。
- 高亮显示 CPU，然后单击“确定”(OK)。

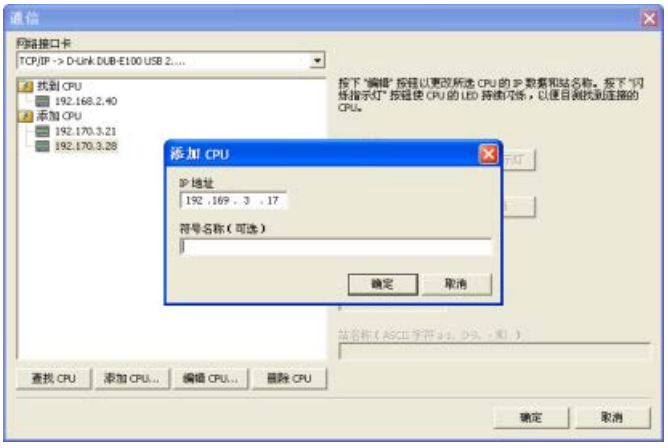


对于“已添加 CPU”（CPU
位于本地网络或远程网络），可通过“通信对话框”(Communications dialog) 与您的 CPU 建立连接：

- 单击“网络接口卡”(Network Interface Card)
  下拉列表，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“确定”(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，不能通过其它方式更改”(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。

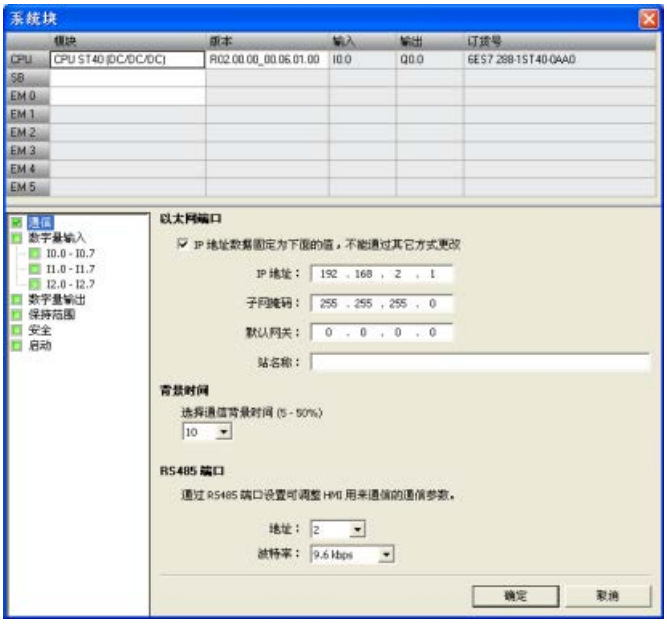
要输入或更改 IP 信息，执行以下操作：

- 如果尚未选中，单击“IP 地址数据固定为下面的值，不能通过其它方式更改”(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cia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。



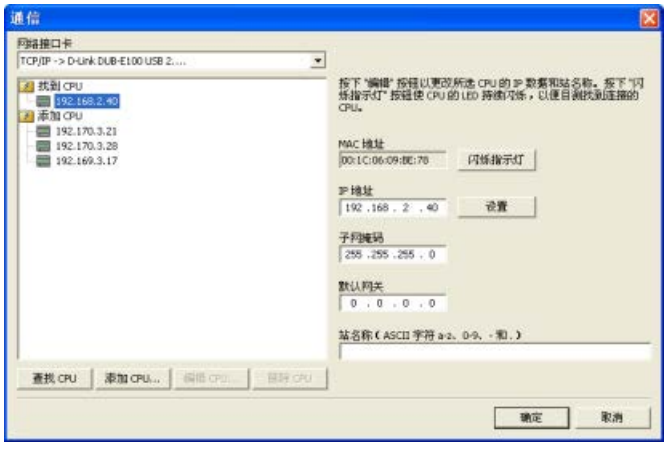
如果在“系统块”(System Block) 对话框中选中“IP 地址数据固定为下面的值，不能通过其它方式更改”(IP address data is fixed to the values below and cannot be changed by other means) 复选框，则您为板载以太网端口输入的 IP 信息为静态信息。必须将静态 IP 信息下载至 CPU，然后才能在 CPU 中激活。如果您想更改 IP 信息，则只能在“系统块”(System Block) 对话框中更改此 IP 信息并将其再次下载至 CPU。

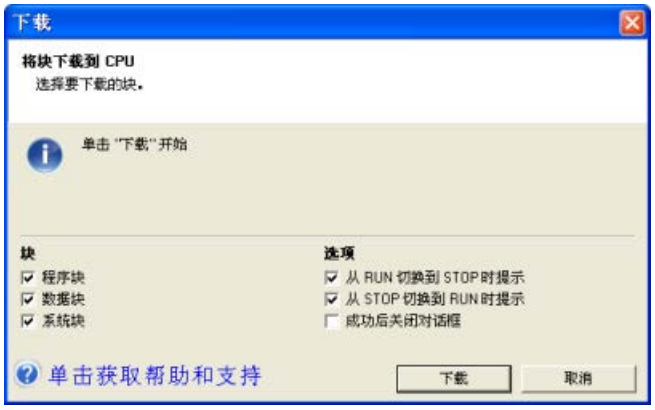
说明

如果已选中“IP 地址数据固定为下面的值，不能通过其它方式更改”(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，不能通过其它方式更改”(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）” (页 220)。

8.4.4.3 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

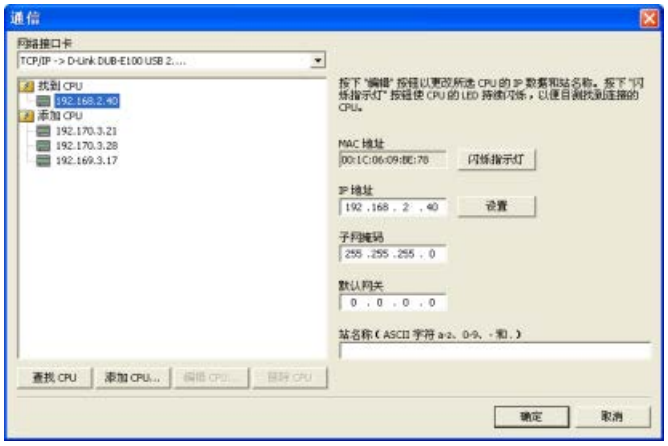
项目树中的通信

“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。



8.4.5 查找 CPU 上的以太网 (MAC) 地址

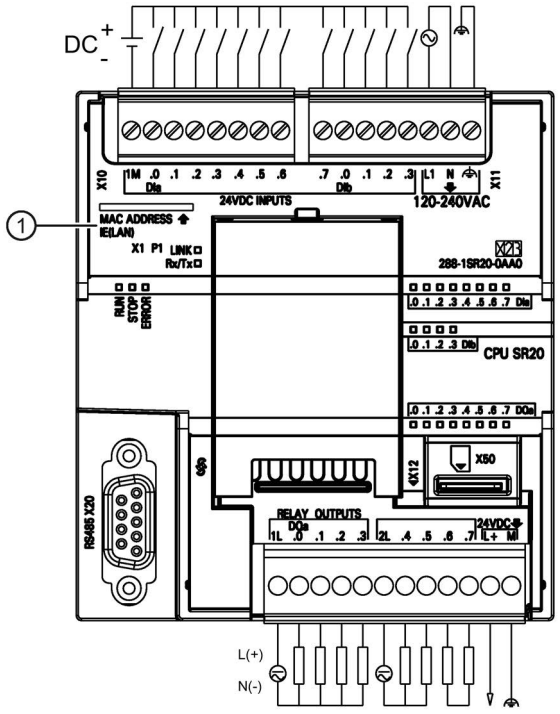
在以太网网络中，“介质访问控制”地址（MAC 地址）是制造商为了标识网络接口而分配的标识符。 MAC 地址通常用制造商的注册标识号进行编码。

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

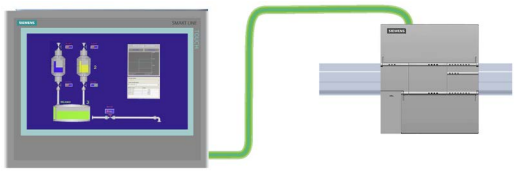
每个 CPU 在出厂时都已装载了一个永久、唯一的 MAC 地址。 您无法更改 CPU 的 MAC 地址。

MAC 地址印在 CPU 正面左上角位置。 请注意，必须打开上面的门才能看到 MAC 地址信息。



① MAC 地址

8.4.6 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。
- 一对一通信不需要以太网交换机；网络中有两个以上的设备时需要以太网交换机。

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。 CPU 上的以太网端口不包含以太网交换设备。

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 8- 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。连接一个 HMI 和一个 CPU 不需要以太网交换机。 有关详细信息，请参见“建立硬件通信连接” (页 31)。
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 422)。

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” (页 144)。

8.4.7 开放式用户通信

8.4.7.1 协议

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。

8.4.7.2 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。

8.4.7.3 端口和 TSAP

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
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。）

8.5 PROFIBUS

PROFIBUS 协议旨在实现与分布式 I/O 设备（远程 I/O）进行高速通信。PROFIBUS 系统使用一个总线控制器轮询 RS485 串行总线上以多点型分布的 DP I/O 设备。

PROFIBUS
设备种类繁多，许多制造商都能提供。这些设备从简单的输入或输出模块到复杂的电机控制器和 PLC，应有尽有。PROFIBUS
DP 设备是指任何能够处理信息并将其输出发送到主站的外围设备。DP
设备构成网络中的被动站（因其没有总线访问权），只能对接收到的消息给予确认或应主站请求发送响应信息。所有 PROFIBUS DP
设备均具有相同的优先级，而所有网络通信均源自主站。

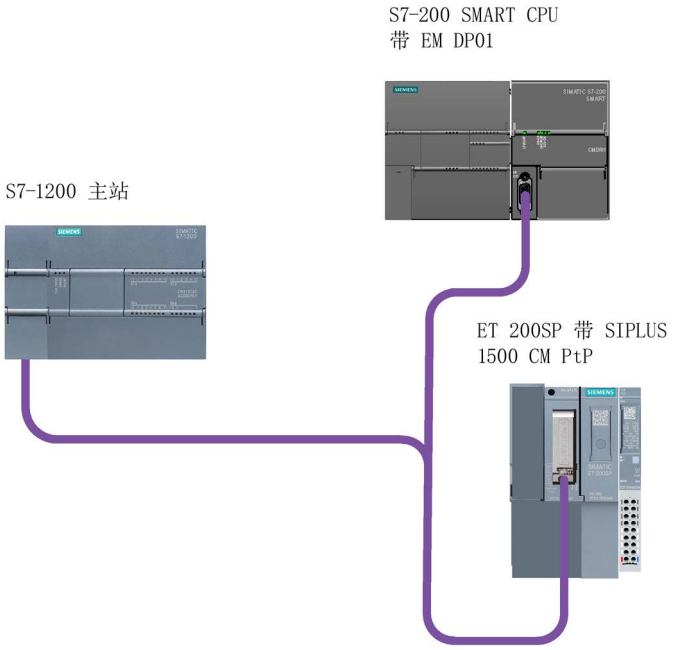
PROFIBUS 主站构成网络的“主动站”。PROFIBUS DP
定义两类主站。一类主站（通常为中央可编程控制器 (PLC) 或运行专用软件的
PC）处理常规通信，或与分配给它的 DP
设备交换数据。二类主站（通常为组态设备，如用于调试、维护或诊断的笔记本电脑或编程控制台）为专用设备，主要用于与 DP 设备通信和用于诊断目的。

ROFIBUS 网络通常有一个主站与多个 DP I/O
设备。（请参见下图。）组态后的主站设备能够识别所连 DP
设备的类型及地址。主站初始化网络并验证网络中的 DP
设备是否与组态相符。主站会不断将输出数据写入 DP 设备并从这些设备读取输入数据。

在 PROFIBUS DP 主站成功组态了 DP 设备后，才拥有该 DP
设备。若网络中存在另一个主站设备，则其访问第一个主站所拥有的 DP
设备时，将受到很大的限制。

S7-200 SMART CPU 作为带有 EM DP01 PROFIBUS DP 模块的 DP 设备连接到 PROFIBUS 网络。EM DP01 可作为 DP V0/V1 主站的通信伙伴。可从 Siemens 客户支持获取 EM DP01 GSD 文件。

下图中的 S7-200 SMART CPU 就是 S7-1200 控制器的 DP 设备:



每个 S7-200 SMART CPU（仅限 ST 与 SR 型号）可组态两个 ROFIBUS EM。

本地 CPU 存储 PROFIBUS EM 的组态数据，可通过每个模块上的开关来设置 PROFIBUS 地址。这使得必要时的通信模块更换变得非常简便。

8.5.1 EM DP01 PROFIBUS DP 模块

8.5.1.1 分布式外设 (DP) 标准通信

PROFIBUS DP（或 DP 标准）是种根据欧洲标准 EN 50170 定义的远程 I/O 通信协议。遵循这一标准的设备即使由不同的公司所制造，也能够互相兼容。DP 代表分布式外备，即远程 I/O。PROFIBUS 代表过程现场总线。

EM DP01 PROFIBUS DP 模块已实施以下通信协议标准中为 DP 设备定义的 DP 标准协议：

- EN 50 170 (PROFIBUS)
  描述了总线访问与传输协议，并规定了数据传输介质的属性。
- EN 50 170（DP 标准）描述了 DP 主站与
  DP 设备之间的周期性高速数据交换。该标准定义了组态与参数分配的过程，解释了如何使用分布式 I/O 功能实现周期性数据交换，并列出了所支持的诊断选项。

需要组态 DP 主站以识别地址、DP 设备类型以及
DP 设备所需的任何参数分配信息。DP 主站还将被告知将从
DP 设备读取的数据置于何处（输入），以及从何处获得数据以写入
DP 设备（输出）。DP 主站建立网络，然后初始化其
DP 设备。DP 主站将参数分配信息以及 I/O 组态写入 DP 设备。DP 主站随后从
DP 设备读取诊断信息以验证 DP 设备已接受参数和 I/O 组态。DP 主站随后开始与 DP 设备交换 I/O 数据。与 DP
设备发生的每个事物都是写入输出与读取输入。该数据交换模式会一直持续下去。如果出现异常，DP 设备就会通知 DP 主站，随后 DP 主站从 DP 设备读取诊断信息。

一旦 DP 主站将参数和 I/O 组态写入了 DP 设备，并且 DP 设备也接受了
DP 主站写入的参数和组态，DP 主站就拥有该 DP 设备。DP 设备只接受其所属 DP 主站的写入请求。网络中的其他 DP 主站能够读取该 DP 设备的输入和输出，但不能向该 DP 设备写入任何信息。

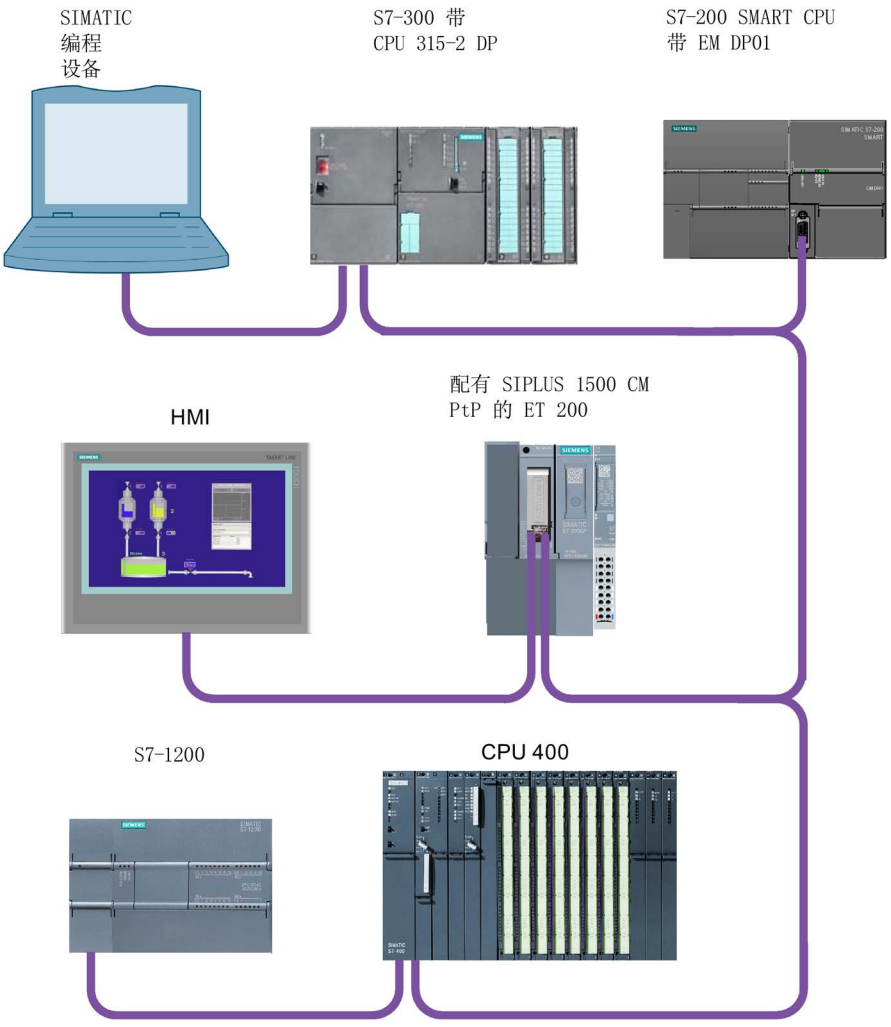
8.5.1.2 使用 EM DP01 将 S7-200 SMART 连接为 DP 设备

S7-200 SMART CPU 可通过 EM DP01 PROFIBUS DP 模块连接到 PROFIBUS DP 网络。EM DP01 作为扩展模块连接到 S7-200 SMART CPU。EM DP01 PROFIBUS DP 模块通过其 DP 通信端口连接到 PROFIBUS 网络。该端口支持 9600 波特到 12M 波特之间的任一 PROFIBUS 波特率。请参见《EM DP01 PROFIBUS DP 模块的技术规范》了解所支持的波特率。

EM DP01 作为 PROFIBUS DP 设备，可从 DP 主站接受多种不同的 I/O 组态，这有助于用户根据应用要求定制数据传输量。不同于许多 DP 设备，EM DP01 不仅限于传输 I/O 数据。EM DP01 还传送输入、计数器值、定时器值或任何其它移入 S7-200 SMART CPU 中变量存储器的值。EM DP01 也会将来自 DP 主站的数据传送到 S7-200 SMART CPU 的变量存储器。用户然后可将这些数据从变量存储器转移到其它数据存储区。

EM DP01 PROFIBUS DP 模块的 DP 端口可以连接到网络中的 DP 主站，并且依然能够作为 MPI 设备与其它主站设备（例如，同一网络中的 SIMATIC HMI 设备或 S7-300/S7-400 CPU）通信。下图所示为带有 S7-200 SMART CPU SR20 和 EM DP01 PROFIBUS DP 模块的 PROFIBUS 网络：

- 配有 CPU 315-2 的 S7-300 作为 DP 主站，已通过装有 STEP 7 编程软件的 SIMATIC 编程设备进行组态。S7-315-2 DP 能够从 EM DP01 中读取数据或将数据写入其中，支持 1 字节到 244 字节的数据。
- S7-200 SMART CPU SR20 是归 CPU 315-2 所有的 DP 设备。ET 200 I/O 模块也是归 CPU 315-2 所有的 DP 设备。
- S7-400 CPU 连接到 PROFIBUS 网络上并使用 S7-400 CPU 用户程序中的 X_GET 指令读取 CPU SR20 的数据。（其它 SIMATIC CPU 可使用 DB1 来访问 S7-200 SMART CPU 中的 V 存储器。）


SIMATIC
编程
设备
S7-300 带
CPU 315-2 DP
S7-200 SMART CPU
带 EM DP01
HMI
配有 SIPLUS 1500 CM
PtP 的 ET 200
S7-1200
CPU 400

8.5.1.3 组态 EM DP01

步骤

1. 为将 S7-200 SMART EM DP01 PROFIBUS DP 模块用作 DP 设备，必须设置 DP 端口的站地址以匹配 DP 主站组态中的地址。站地址通过 EM DP01 上的旋转开关进行设置。
2. 在完成开关更改后，必须重启 S7-200 SMART CPU 才能使新的 DP 设备地址生效。

结果

DP 主站设备通过将输出区域的信息发送到 DP 设备的输出缓冲区来与每个 DP 设备交换数据。DP 设备通过返回 DP 主站存储在输入区域中的输入缓冲区内容来响应 DP 主站发来的消息。

组态步骤

S7-200 SMART EM DP01 PROFIBUS DP 模块可通过 DP 主站进行组态，以接受 DP 主站发来的输出数据并将输入数据返回给 DP 主站。输出与输入数据缓冲区位于 S7-200 SMART CPU 的变量存储器（V 存储器）中。组态 DP 主站时，需要在 V 存储器中定义输出数据缓冲区的起始字节单元，作为 EM DP01 的部分参数分配信息。还需要将 I/O 组态定义为要写入 S7-200 SMART CPU 的输出数据量和要从 S7-200 SMART CPU 返回的输入数据量。EM DP01 决定了来自 I/O 组态的输入和输出缓冲区的大小。DP 主站将参数分配和 I/O 组态信息写入 EM DP01。EM DP01 随后将 V 存储器的地址以及输入输出的数据长度传送到 S7-200 SMART CPU。这些值存储在 S7-200 SMART CPU 的专用存储器中供用户程序使用。有关详细信息，请参见 “用户程序注意事项” (页 452)中的 SM 状态信息。

8.5.1.4 数据一致性

PROFIBUS 支持三种类型的数据一致性：

- 字节：确保字节作为整体传送。
- 字：确保字的传送过程不会被 CPU 中的其它进程所中断。
- 缓冲区：确保整个数据缓冲区作为一个单位传送，不会被 CPU 中的其它进程所中断。

EM DP01 在数据处理过程中始终利用缓冲区一致性。

S7-200 SMART CPU 和 EM DP01 的数据缓冲区一致性

EM DP01 和 S7-200 SMART CPU 可确保整个传送的缓冲区一致性：

- EM DP01 以一条消息的形式接收 DP 主站的输出。
- EM DP01 将所有输出以一条消息形式传送到 S7-200 SMART CPU，并且传送过程不可中断。
- S7-200 SMART CPU 一次性将所有输出传送到 V 存储器。传送不可受用户干扰而中断。

输入到 DP 主站时也会确保这种一致性：

- S7-200 SMART CPU 一次性将所有输入从 V 存储器传出。传送不可受用户干扰而中断。
- S7-200 SMART CPU 将所有输入以一条消息形式传送到 EM DP01。该传送不可被中断。
- EM DP01 将输入以一条消息形式发送到 DP 主站。

DP 主站的一致性

DP 主站 CPU 的一致性并非总是缓冲区一致。除非 DP 消息非常小，否则 DP 主站 CPU 不会将整个 DP 消息作为一个不可分割的对象进行处理。DP 主站 CPU 通常会以较小的单位移动 PROFIBUS 数据。既可以通过它们将数据移动到 I/O 区，也可以由用户使用 DPRD_DAT（读取 DP 设备的一致性数据）与 DPWR_DAT（写入 DP 设备的一致性数据）指令来控制移动。使用 DPRD_DAT 和 DPWR_DAT 指令，一次可获取一个组态“插槽”的信息。因为允许有两个组态插槽，这样就可以使用两条 DPRD_DAT 指令来获取所有数据。仅对每条 DPRD_DAT 指令保证一致性。

8.5.1.5 支持的组态

下表列出了 S7-200 SMART EM DP01 PROFIBUS DP 模块支持的组态：

表格 8- 2 EM DP01 PROFIBUS DP 组态选项

组态	主站的输入	主站的输出	数据一致性
1	通用模块		缓冲区一致性 [1]
2	4 个字节	4 个字节	
3	8 个字节	8 个字节	
4	16 个字节	16 个字节	
5	32 个字节	32 个字节	
6	64 个字节	64 个字节	
7	122 个字节	122 个字节	
8	128 个字节	128 个字节	

[1] 所有 EM DP01 组态均为缓冲区一致。

在 EM DP01 组态中，可以混用并匹配以上组态中的任意两种。以下是两个示例：

- 一个 32 字节输入输出的组态加上一个 8 字节输入输出的组态得到总计 40 输入字节以及 40 输出字节。
- 一个 122 字节输入输出的组态加上一个 122 字节输入输出的组态得到总计 244 输入字节以及 244 输出字节。

EM DP01 最大允许 244 输入字节和 244 输出字节。如果对 EM DP01 使用两种组态，则所有的输入数据和所有的输出数据都是连续的。更多相关信息，请参见 “V 存储器和 I/O 地址区域的示例” (页 451)。

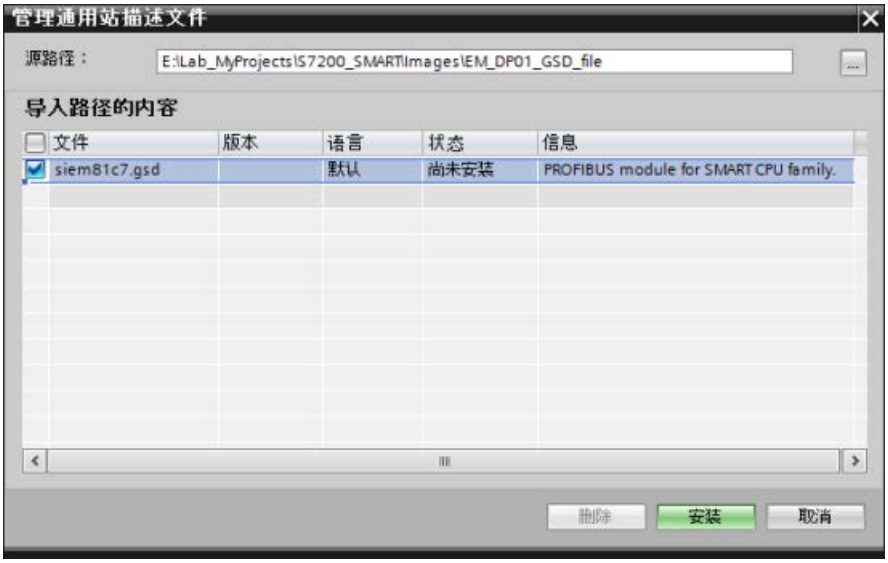
8.5.1.6 安装 EM DP01 GSD 文件

PROFIBUS GSD 文件描述 DP 设备与其功能。编程人员使用 GSD 文件组态 DP主站。

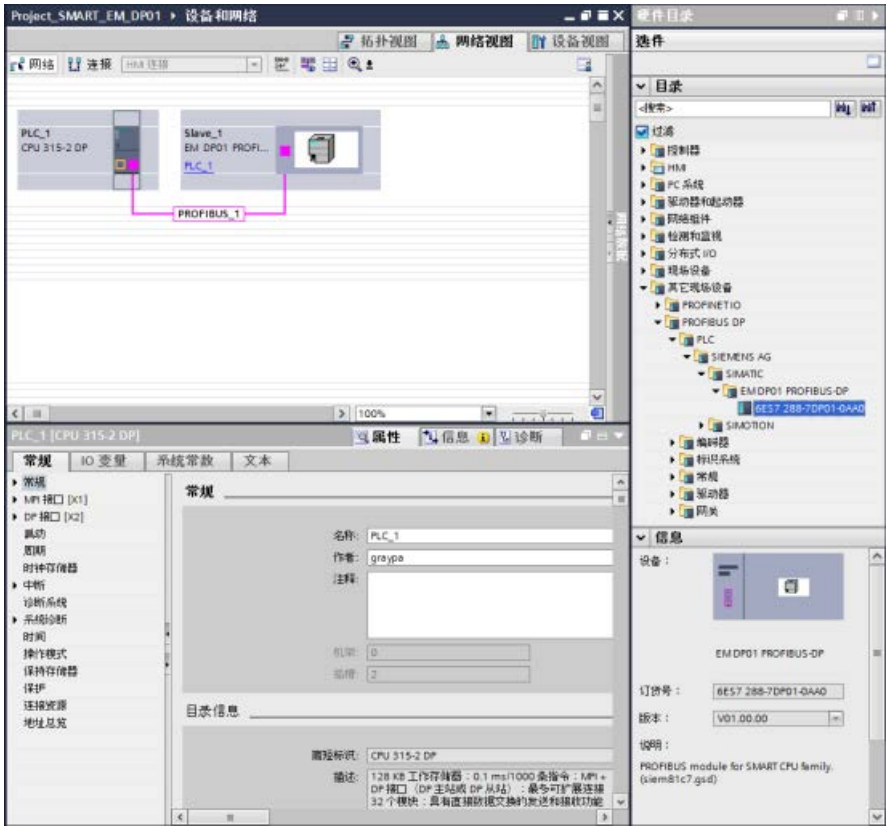
请遵循以下步骤安装 EM DP01 GSD 文件：

1. 启动 TIA Portal 软件。
2. 新建项目。
3. 在项目视图中，找到菜单栏并选择：“选项 > 管理通用站描述文件(GSD)”（Options > Manage general station description files (GSD)）

4. 在“源”(Source) 路径中，使用下拉按钮，找到之前加载到计算机中的 EM DP01 GSD 文件。
5. 选中相应 GSD 文件行的复选框。
6. 单击“安装”(Install) 按钮：



7. 执行上述操作后，将在硬件目录中安装 EM DP01 GSD 文件，如下图所示：



8. 插入 CPU 315-2 DP 作为 DP 主站。
9. 插入 EM DP01 PROFIBUS DP 模块。
10. 如上图所示，在 DP 主站和设备之间创建 PROFIBUS 网络。

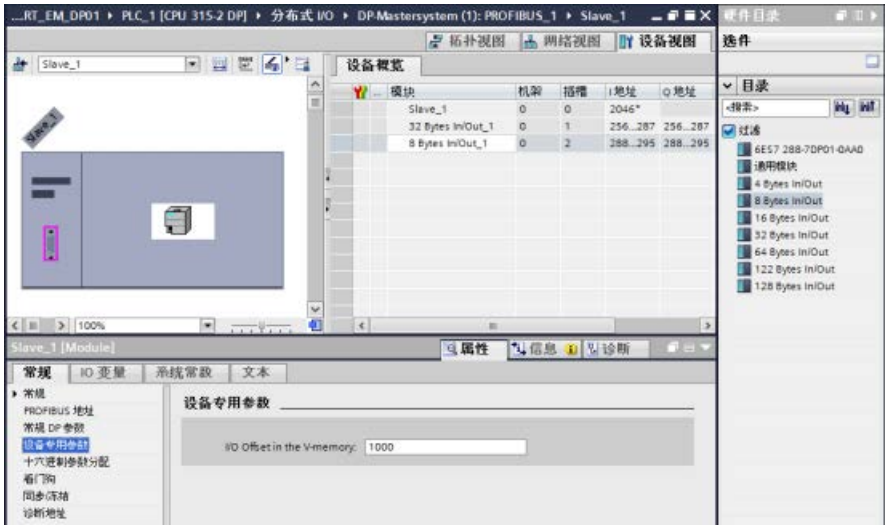
8.5.1.7 组态 EM DP01 I/O

可通过使用预组态的或通用的模块 I/O 组态选项来组态 EM DP01 I/O。EM DP01 组态支持两个插槽，因此可在 DP 主站和 S7-200 SMART CPU 之间传送超过 128 字节的数据。这也使得用户能够组态 PROFIBUS 所允许的最大 244 字节。以下两个示例中说明了两种可能的 I/O 组态组合。

32 Bytes In/Out 和 8 Bytes In/Out 组态

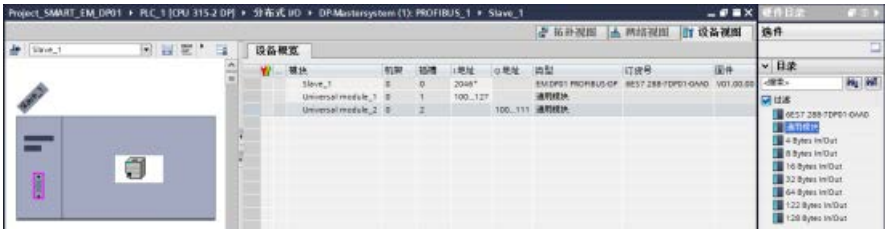
本例中的插槽一包含“32 Bytes In/Out”预组态 I/O 选项，插槽二包含“8 Bytes In/Out”预组态 I/O 选项。

在“Properties”、“General”选项卡区域，单击“Device-specific parameters”以显示“I/O Offset in the V memory”字段。在此处可分配为该操作预留的那部分 V 存储器的启动地址。



通用模块组态

本例中，插槽一和插槽二均包含“Universal module”I/O 选项，您可根据应用需要的输入输出数量（最多 244 输入字节和 244 输出字节）组态这两个插槽。



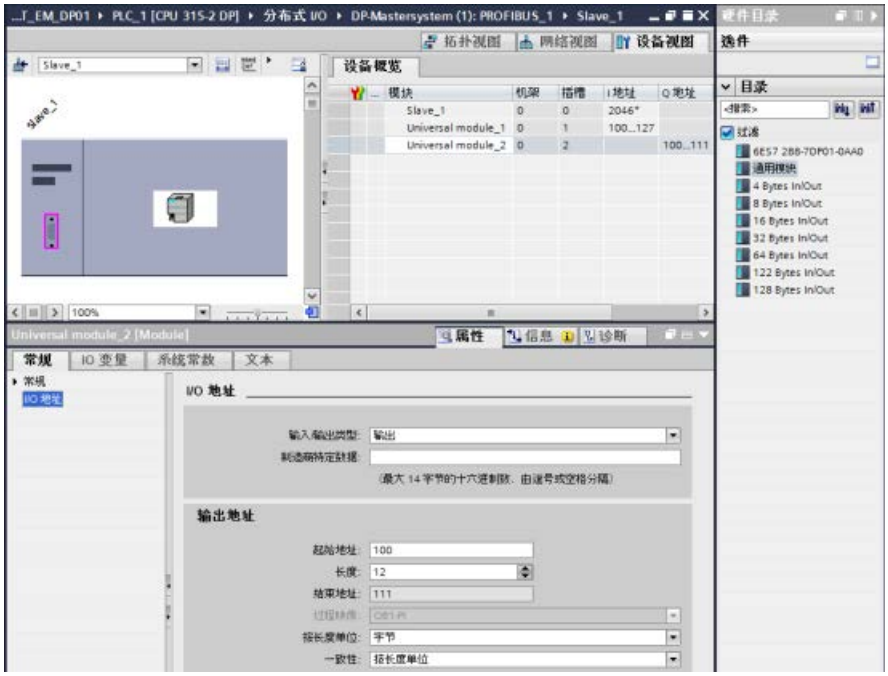
在“Properties”、“General”选项卡区域，单击“I/O addresses”以显示输入/输出地址组态字段。在“Input/output type:”字段，须为此插槽中的通用模块选择以下选项之一：

- Input
- Output
- Input/output

然后可组态应用的输入和/或输出的地址范围。

说明

“Empty slot”为“Input/output type:”字段的默认选项。须将“Empty slot”更改为“Input”、“Output”或“Input/output"以组态 I/O 地址。

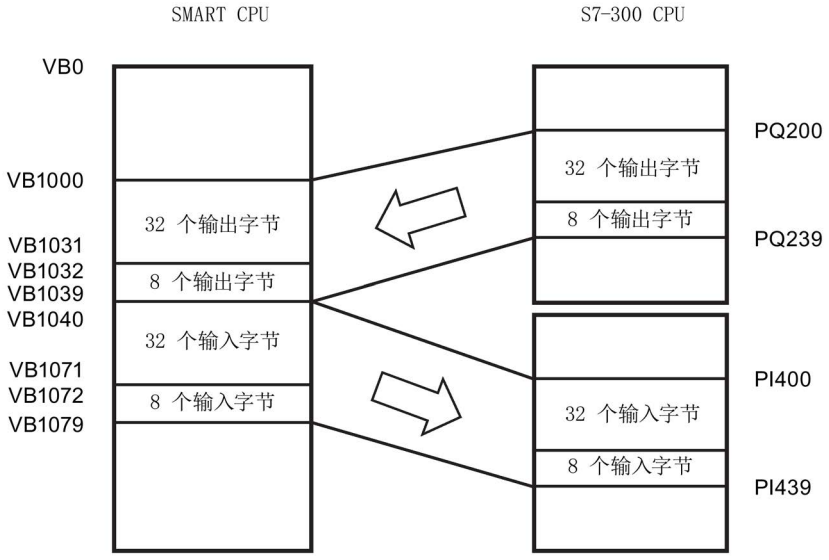


说明

在以上示例中，CPU 315-2 DP 是组态的 DP 主站。根据该主站 CPU 的类型，EM DP01 的“Properties”可能与上图的显示略有不同。

8.5.1.8 V 存储器和 I/O 地址区域的示例

下图显示了 S7-200 SMART CPU 中的 V 存储器与 S7-300 PROFIBUS DP 主站的 I/O 地址区域的示例：



本例中，DP 主站已定义一个 I/O 组态，其包含两个插槽且 V 存储器偏移量为 1000。示例将第一个插槽组态为 32 字节的输入输出，第二个插槽组态为 8 字节的输入输出。S7-200 SMART CPU 的输出与输入缓冲区均为 40 字节 (32 + 8)。输出数据（来自 DP 主站）缓冲区起始于 V1000；输入数据（送入 DP 主站）缓冲区紧随输出缓冲区并起始于 V1040。

在 EM DP01 和 SMART CPU 中，所有输出数据（全部为 40 个字节）均视为一个一致性的缓冲区数据块。S7-300 的输出数据采用不同的一致性进行处理，具体取决于用户是否利用 I 和 Q 区域或者用户是否使用 DPRD_DAT（读取 DP 设备的一致性数据）与 DPWR_DAT（写入 DP 设备的一致性数据）指令。即使使用 DPRD_DAT 和 DPWR_DAT 指令，数据也仅在 32 字节和 8 字节的块内一致。唯有用户不在用户中断块中读取或写入数据时，全部的 40 字节才一致。

说明

如果使用超出 4 字节的数据单位（一致数据），则可使用 DPRD_DAT 指令读取 DP 设备的输入，使用 DPWR_DAT 指令寻址 DP 设备的输出。有关详细信息，请参见“数据一致性”与 *S7-300 和 S7-400 的系统软件之系统及标准函数参考手册*。

可将输入和输出缓冲区的位置组态为 S7-200 SMART CPU 中 V 存储器的任意位置。输入和输出缓冲区的默认地址为 VB0。输入输出缓冲区的位置是 DP 主站写入 S7-200 SMART CPU 的参数分配信息的一部分。用户需要组态 DP 主站以识别 DP 设备并将所需参数和 I/O 组态写入主站的每个 DP 设备。

使用 STEP 7 编程软件组态 SIMATIC S7 DP 主站。有关使用组态和编程软件包的详细信息，请参见这些设备的手册。有关 PROFIBUS 网络及其组件的详细信息，请参见 *ET 200 分布式 I/O 系统手册*。

参见

数据一致性 (页 443)

8.5.1.9 用户程序注意事项

通过 DP 主站成功组态 EM DP01 PROFIBUS DP 模块后，相应 EM DP01 以及 DP 主站就进入数据交换模式。在数据交换模式中，DP 主站将输出数据写入 EM DP01，EM DP01 随后用最新的 S7-200 SMART CPU 输入数据进行响应。EM DP01 不断更新来自 S7-200 SMART CPU 的输入以为 DP 主站提供最新的输入数据。EM DP01 随后将输出数据传送到 S7-200 SMART CPU。初始化期间，来自 DP 主站的输出数据以 DP 主站提供的地址开始放入 V 存储器（输出缓冲区）中。发送给 DP 主站的输入数据从 V 存储器中紧邻输出数据的单元（输入缓冲区）获取。

须由 S7-200 SMART CPU 中的用户程序将 DP 主站发来的输出数据，从输出缓冲区移动到使用这些数据的数据区。同样，要传输给 DP 主站的输入数据须从不同的数据区移动到输入缓冲区，以便传送到主站。

DP 主站发来的输出数据在扫描相应用户程序之前立即放入 V 存储区。在扫描相应用户程序之后，将输入数据（到 DP 主站）从 V 存储器复制到 EM DP01 中以传送给 DP 主站。

发往 DP 主站的输入数据将在 EM DP01 与 DP 主站的下一次数据交换过程中传送给主站。

状态信息

每个扩展模块将基于其物理位置分配到 50 字节的专用存储器 (SM)。模块会更新 SM 单元，以反映模块相对于 CPU 的位置（相对于其他的模块）。如果它是第一个模块，则更新 SMB1400 到 SMB1449。如果它是第二个模块，则更新 SMB1450 到 SMB1499，以此类推。请参见下表：

表格 8- 3　专用存储器字节 SMB1400 到 SMB1699

专用存储器字节 SMB1400 到 SMB1699					
插槽 0 中的智能模块	**插槽 1 中的智能模块**	**插槽 2 中的智能模块**	**插槽 3 中的智能模块**	**插槽 4 中的智能模块**	**插槽 5 中的智能模块**
SMB1400 到 SMB1449	SMB1450 到 SMB1499	SMB1500 到 SMB1549	SMB1550 到 SMB1599	SMB1600 到 SMB1649	SMB1650 到 SMB1699

若尚未与 DP 主站建立 DP 通信，则这些 SM 单元会显示默认值。在 DP 主站将参数和 I/O 组态写入 EM DP01 PROFIBUS DP 模块后，这些 SM 单元就会显示由 DP 主站设置的组态。使用下表所示 SM 单元中的信息或者 V 存储器缓冲区中的数据之前，应检查协议状态字节（例如对于插槽是 SMB1424），确认 EM DP01 当前处于与 DP 主机交换数据的模式下。

说明

不能通过向 SM 存储器单元写入信息来组态 EM DP01 PROFIBUS DP I/O 缓冲区的大小或缓冲区位置。只有 DP 主站能组态 EM DP01 PROFIBUS DP 模块来实现 DP 操作。

表格 8- 4　EM DP01 PROFIBUS DP 的专用存储器字节

插槽 0 中的智能模块	...	插槽 5 中的智能模块	说明	
SMB1400	...	SMB1650	DP 设备的站地址，通过地址开关设置（0 - 99，十进制）	
SMB1401	...	SMB1651	DP 设备的主站的地址（0 到 126）（如未连接 DP 主站，则显示 255）	
SMW1402	...	SMW1652	输出缓冲区的 V 存储器地址，表现为 VB 的偏移量（例如， 1000 表示 VB1000）。	
SMB1404	...	SMB1654	输出数据的字节数	
SMB1405	...	SMB1655	输入数据的字节数	
SMB1406	...	SMB1401	DP 标准协议状态字节	
			编号	说明
			0	上电后未发起 DP 通信
			1	检测到组态/参数分配错误
			2	当前处于数据交换模式
			3	脱离数据交换模式
SMB1650 到 SMB1699	...	SMB1657 到 SMB1699	预留 - 上电时擦除	

注：每次 DP 设备接受组态/参数设定信息时，都会更新 SM 单元。甚至是检测到组态/参数分配错误时，也会更新这些单元。每次上电时会擦除这些单元。

注：STEP 7-Micro/WIN SMART中 EM DP01 的“PLC 信息”也会提供这些信息。

注：用户程序可访问这些信息并用来处理 EM DP01 数据。

8.5.1.10 EM DP01 PROFIBUS DP 的 LED 状态指示灯

EM DP01 PROFIBUS DP 模块的前面板上有四个状态 LED 用于指示 DP 端口的工作状态：

- DIAG LED：
  - 双色（绿色/红色）LED 指示 EM DP01 的工作状态和故障状态
  - 红色闪烁：自启动时开始闪烁，直到 CPU 完成 EM DP01 登录后停止闪烁，或在 EM DP01 出现故障时闪烁
  - 绿色闪烁：EM DP01 等待 S7-200 SMART CPU 传输组态和参数（登录后绿灯立即闪烁）期间或固件升级期间
  - 绿色点亮：无任何故障且 EM DP01 已组态
- POWER LED：
  - 绿色点亮：有用户 24 V DC
  - 灭：无用户 24 V DC
- DP ERROR LED：
  - 红色闪烁：DP 主站写入 EM DP01 的 I/O 组态或参数信息存在错误
  - 红色点亮：DP 通信被中断
  - 灭：无错误或从未建立数据交换
- DX MODE LED：
  - 灭：S7-200 SMART CPU 通电后，未尝试进行 DP 通信或 DP 通信被中断
  - 绿色点亮：成功发起 DP 通信后（EM DP01 已进入与 DP 主站交换数据的模式），该指示灯保持点亮，直至 EM DP01 退出数据交换模式

说明

如失去 DP 通信，将强制 EM DP01 退出数据交换模式，此时 DX MODE LED 熄灭并且 DP ERROR LED 红色亮起。此情况一直持续到 S7-200 SMART CPU 关闭或数据交换模式恢复。

下表总结了 EM DP01 状态 LED 指示的状态：

表格 8- 5　EM DP01 PROFIBUS DP 模块的状态 LED

LED	灭	红色	红色闪烁	绿色闪烁	绿色
DIAG	-	模块内部故障	自启动时开始闪烁，直到 CPU 完成 EM DP01 登录后停止闪烁，或在 EM DP01 出现故障时闪烁	EM DP01 等待 S7-200 SMART CPU 传输组态和参数期间或固件升级期间	无任何故障；EM DP01 已组态
POWER	无 24 V DC 用户电源	-	-		24 V DC 用户电源正常
DP ERROR	无错误	DP 通信中断；数据交换模式停止	参数设置/组态错误（来自 DP 主站）		-
DX MODE	数据交换模式未激活或数据通信中断	-	-		数据交换模式激活

8.5.1.11 使用 HMI 和配有 EM DP01 的 S7-CPU

无论是否用作 PROFIBUS DP 设备，EM DP01 PROFIBUS DP 模块都可用作 MPI 主站的通信接口。EM DP01 可使用 S7-300/400 的 X_GET/X_PUT 函数将 S7-300/400 连接到 S7-200 SMART。HMI 设备（例如 SMART HMI 或 TD 400）可用来通过 EM DP01 与 S7-200 SMART 通信。

一些设备允许用户选择 V 存储器作为 S7-200 SMART CPU 中的存储区。如果不能选择 V 存储器，则应该组态客户端（CPU 或 HMI 设备）读取并写入 DB1 以访问 S7-200 SMART CPU 的 V 存储器。例如，X_GET 需要远程地址设置为 P#DB1.DBX100.0 BYTE 20 才能读取 V 存储器中从 VB100 开始的 20 字节。

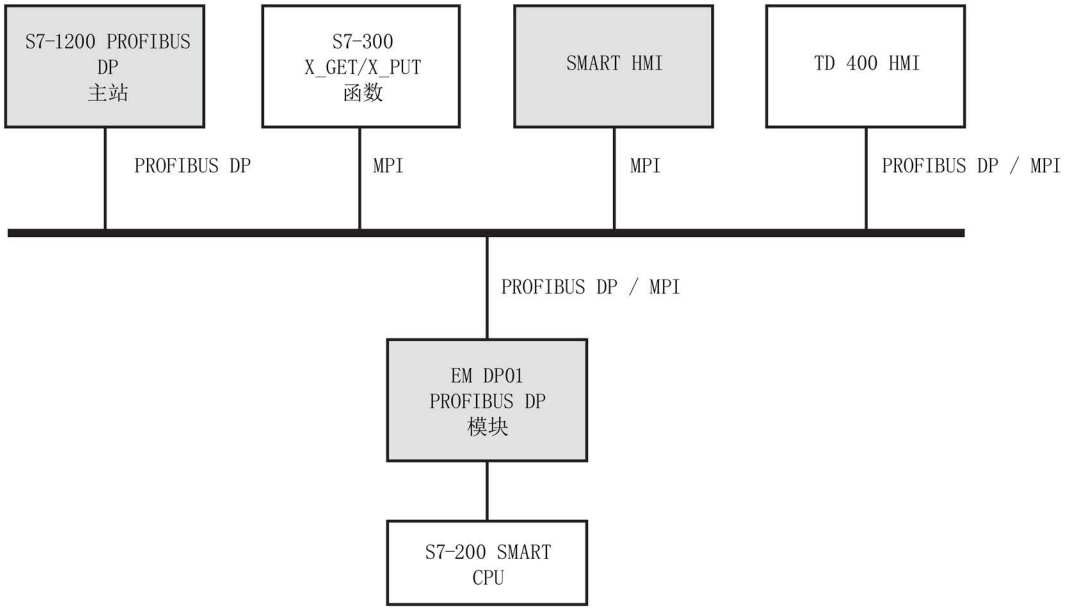
说明

S7-1200 PROFIBUS DP 主站不能使用 GET/PUT 函数访问 S7-200 SMART CPU。S7-1200 DP 主站仍能通过 PROFIBUS 数据交换模式访问 S7-200 SMART CPU。

将 EM DP01 PROFIBUS DP 模块用于 MPI 通信时，XGET/XPUT 函数的地址参数须设置为 EM DP01 的地址（地址开关）。发送到 EM DP01 的 MPI 消息将传递到 S7-200 SMART CPU。

MPI 和 HMI 设备发来的消息将占用 S7-200 SMART CPU 的后台通信时间。可增加后台通信时间以更快地响应 MPI 和 HMI 请求。

除 DP 主机以外，EM DP01 还支持最多六个连接（六个设备）。EM DP01 将这六个连接中的一个预留给了 HMI 设备。为使 EM DP01 能够与多个主站通信，所有主站均须以相同的波特率工作。请参见下图了解一种可能的网络组态：



8.5.1.12 设备数据库文件：GSD

不同的 PROFIBUS 设备具有不同的性能特征。这些特点会因功能（例如 I/O 信号和诊断消息的数量）或总线参数（例如传送速度与时间监视）而不同。这些参数因设备类型和供应商不同而不同，通常记录在技术手册中。为帮助用户简化 PROFIBUS 的组态，可在一个称为设备数据库文件或 GSD 文件的电子数据表中指定具体设备的性能特征。基于 GSD 文件的组态工具可以将不同供应商的设备方便地集成在同一个网络中。

GSD 设备数据库文件以精确定义的格式全面地描述设备的各项特征。供应商负责为每种设备都准备 GSD 文件并提供给 PROFIBUS 用户使用。组态系统使用 GSD 文件可读取 PROFIBUS 设备的特征并在组态网络时使用这些信息。

如果您的软件版本不包含 EM DP01 的组态文件，则您可从 Siemens 客户支持获取最新的 GSD 文件 (SIEM81C7.GSD)。

如果您使用的并非 Siemens 主站设备，请参见制造商提供的有关如何使用 GSD 文件组态主站设备的文件。

EM DP01 PROFIBUS-DP 6ES7288-7DP01-0AA0 的 GSD 文件

表格 8- 6　常规参数

参数	值
#Profibus_DP	
GSD_Revision	= 5
Vendor_Name	= "Siemens"
Model_Name	= "EM DP01 PROFIBUS-DP"
Revision	= "V01.00.00"
Ident_Number	= 0x81C7
Protocol_Ident	= 0
Station_Type	= 0
FMS_supp	= 0
Hardware_Release	= 1
Software_Release	= "V01.00.00"
;	
9.6_supp	= 1
19.2_supp	= 1
45.45_supp	= 1
93.75_supp	= 1
187.5_supp	= 1
500_supp	= 1
1.5M_supp	= 1
3M_supp	= 1
6M_supp	= 1
12M_supp	= 1
;	
MaxTsdr_9.6	= 40
MaxTsdr_19.2	= 40
MaxTsdr_45.45	= 40

参数	值
MaxTsdr_93.75	= 40
MaxTsdr_187.5	= 40
MaxTsdr_500	= 40
MaxTsdr_1.5M	= 40
MaxTsdr_3M	= 50
MaxTsdr_6M	= 100
MaxTsdr_12M	= 200
;	
Redundancy	= 0
Repeater_Ctrl_Sig	= 2
24V_Pins	= 2
Implementation_Type	= "DPC31"
Bitmap_Device	= "EM_DP01N"

表格 8- 7　从站规范

参数	值
OrderNumber	= "6ES7 288-7DP01-0AA0"
Periphery	= "SIMATIC S7"
Info_Text	= "PROFIBUS module for SMART CPU family."
Slave_Family	= 10@TdF@SIMATIC
Freeze_Mode_supp	= 1
Sync_Mode_supp	= 1
Set_Slave_Add_Supp	= 0
Auto_Baud_supp	= 1
Min_Slave_Intervall	= 1
Fail_Safe	= 0
;	

参数	值
Modular_Station	= 1
Max_Module	= 2
Modul_Offset	= 0
;	
Max_Input_len	= 244
Max_Output_len	= 244
Max_Data_len	= 488
Max_Diag_Data_Len	= 6

表格 8- 8　DPV1 支持

参数	值
DPV1_Slave	= 1
C1_Read_Write_supp	= 1
C2_Read_Write_supp	= 1
C1_Max_Data_Len	= 240
C2_Max_Data_Len	= 240
C1_Response_Timeout	= 100
C2_Response_Timeout	= 100
C1_Read_Write_required	= 0
C2_Read_Write_required	= 0
C2_Max_Count_Channels	= 6
Max_Initiate_PDU_Length	= 64
Ident_Maintenance_supp	= 1
DPV1_Data_Types	= 0
WD_Base_1ms_supp	= 0
Check_Cfg_Mode	= 0
Publisher_supp	= 0

表格 8- 9　UserPrmData-Definition

参数	值
ExtUserPrmData	= 1 "I/O Offset in the V-memory"
Unsigned16 0 0-20479	
EndExtUserPrmData	

表格 8- 10　UserPrmData:长度和预设置

参数	值
Max_User_Prm_Data_Len	= 5
Ext_User_Prm_Data_Const (0)	= 0x00,0x00,0x00,0x00,0x00
Ext_User_Prm_Data_Ref (3)	= 1

表格 8- 11　模块定义列表

参数	值
Module 20 EndModule	= " 4 Bytes In/Out" 0xF1
Module 21 EndModule	= " 8 Bytes In/Out" 0xF3
Module 22 EndModule	= " 16 Bytes In/Out" 0xF7
Module 23 EndModule	= " 32 Bytes In/Out" 0xFF
Module 24 EndModule	= " 64 Bytes In/Out" 0xC0, 0xDF, 0xDF

参数	值
Module 25 EndModule	= "122 Bytes In/Out" 0xC0, 0xFC, 0xFC
Module 26 EndModule	= "128 Bytes In/Out" 0xC0, 0xFF, 0xFF

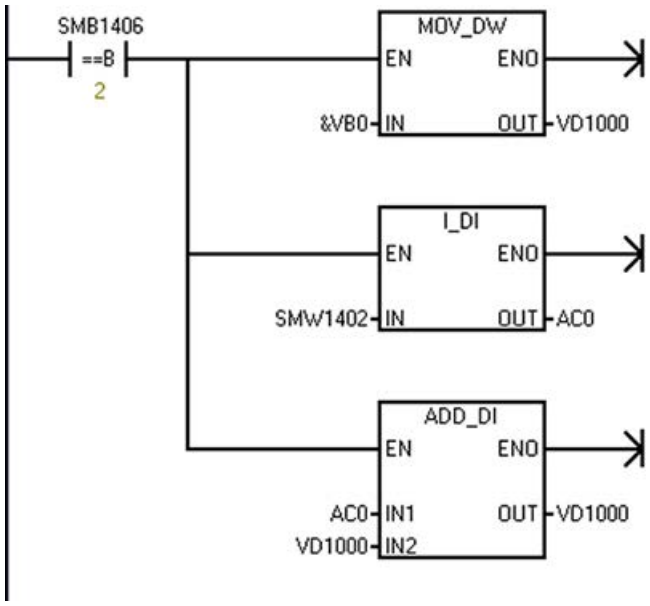
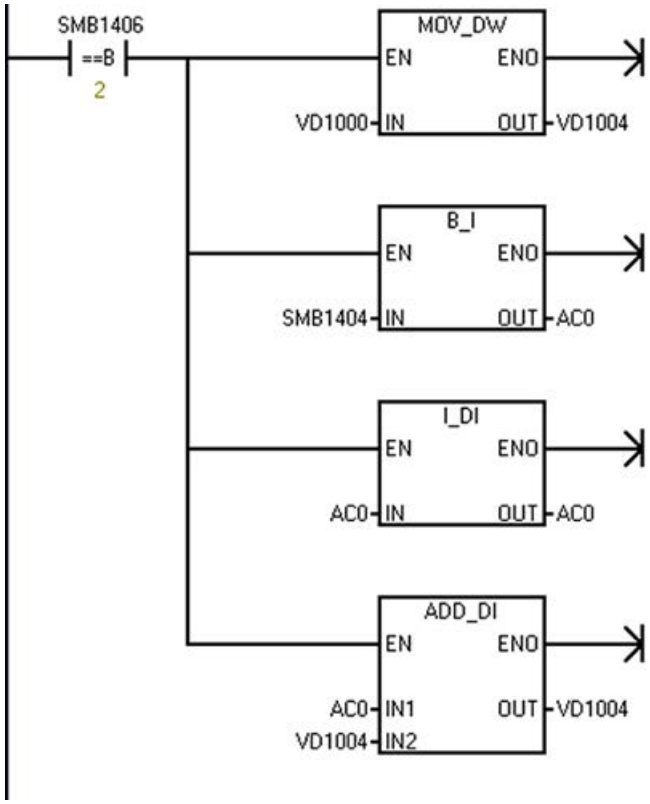
8.5.1.13 与 CPU 进行 PROFIBUS DP 通信的示例程序

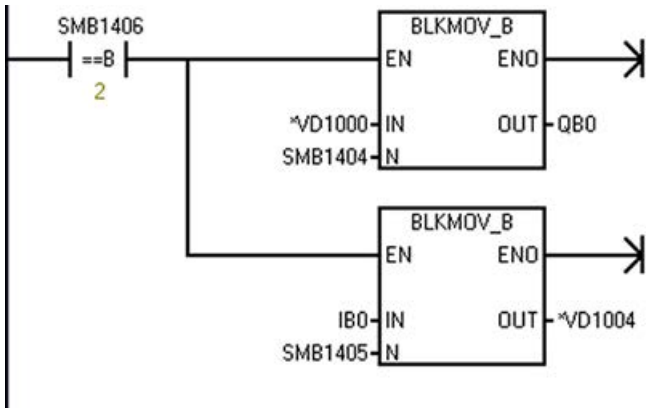
下面显示的是适用于 CPU 上插槽 0 中的 PROFIBUS DP 模块的示例程序，该程序使用 SM 存储器中 DP 端口信息。该程序通过 SMW1402 确定 DP 缓冲区位置，通过 SMB1404 和 SMB1405 确定缓冲区大小。这些信息用于将 DP 输出缓冲区的数据复制到 CPU 的过程映像输出寄存器。同样，CPU 的过程映像输入寄存器中的数据则被复制到 V 存储器输入缓冲区。

在适用于插槽 0 中 DP 模块的如下示例程序中，SM 存储区的 DP 组态数据提供了 DP 设备的组态。该程序使用以下数据：

SMB1406	DP 状态
SMB1401	主站地址
SMW1402	输出的 V 存储器偏移量
SMB1404	输出数据的字节数
SMB1405	输入数据的字节数
VD1000	输出数据指针
VD1004	输入数据指针

表格 8- 12　示例：组态与 S7-200 SMART CPU 的 DP 通信

LAD/FBD	说明	STL
程序段 1：  	计算输出数据指针。若处于数据交换模式： 1. 输出缓冲区是相对于 VB0 的偏移量。 2. 将 V 存储器偏移量转换为长整数。 3. 与 VB0 地址相加获得输出数据指针。	LDB= SMB224, 2 MOVD &VB0, VD1000 ITD SMW226, AC0 +D AC0, VD1000
程序段 2：  	计算输入数据指针。若处于数据交换模式： 1. 复制输出数据指针。 2. 获取输出字节数。 3. 与输出数据指针相加获得起始输入数据指针。	LDB= SMB224, 2 MOVD VD1000, VD1004 BTI SMB228, AC0 ITD AC0, AC0 +D AC0, VD1004

LAD/FBD	说明	STL
程序段 3：  	将主站输出传送到 CPU 输出。将 CPU 输入复制到主站输入。若处于数据交换模式： 1. 将主站输出复制到 CPU 输出。 2. 将 CPU 输入复制到主站输入。	LDB= SMB224, 2 BMB *VD1000, QB0, VB1008 BMB IB0, *VD1004, VB1009

8.5.1.14 EM DP01 PROFIBUS DP 模块技术规范参考

有关 EM DP01 PROFIBUS DP 模块的更多信息，请参见“EM DP01 PROFIBUS DP 模块” (页 832)的技术规范。

8.6 RS485

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）。

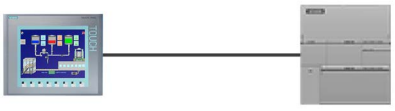
RS485 可使用 PPI 协议和自由端口：

- 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

8.6.1 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。



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 CPU 支持的连接数。

所有 S7-200 SMART CPU 都支持 PPI 和 PPI 高级协议。

表格 8- 13 S7-200 SMART CPU 的连接数

模块	波特率	连接
RS485 端口	9.6 千波特、19.2 千波特或 187.5 千波特	4
RS485/RS232 信号板	9.6 千波特、19.2 千波特或 187.5 千波特	4

8.6.2 波特率和网络地址

8.6.2.1 波特率和网络地址定义

波特率

数据在网络中的传输速度称为波特率，通常以千波特 (kbaud) 或兆波特 (Mbaud) 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 千波特说明传输率为每秒 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 8- 14 S7-200 SMART CPU 支持的波特率

网络	波特率
PPI 协议	仅限 9.6 千波特、19.2 千波特和 187.5 千波特
自由端口模式	1200 波特到 115.2 千波特

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 8- 15 S7-200 SMART 设备的默认地址

S7-200 SMART 设备	默认地址
HMI	1
S7-200 SMART CPU	2

8.6.2.2 为 S7-200 SMART CPU 设置波特率和网络地址

简介

必须为 S7-200 SMART CPU 组态 RS485 端口网络地址和波特率，以便 CPU 可通过 RS485 网络（例如 TD400C）与 SIMATIC HMI 通信。

RS485 端口网络地址必须不同于 RS485 网络中其它设备的网络地址，RS485 端口波特率必须与 RS485 网络中其它设备的波特率相同。 默认的 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 千波特。

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，必须将系统块下载到 S7-200 SMART CPU。

步骤

要访问“系统块”(System Block) 对话框，请单击以下某项：

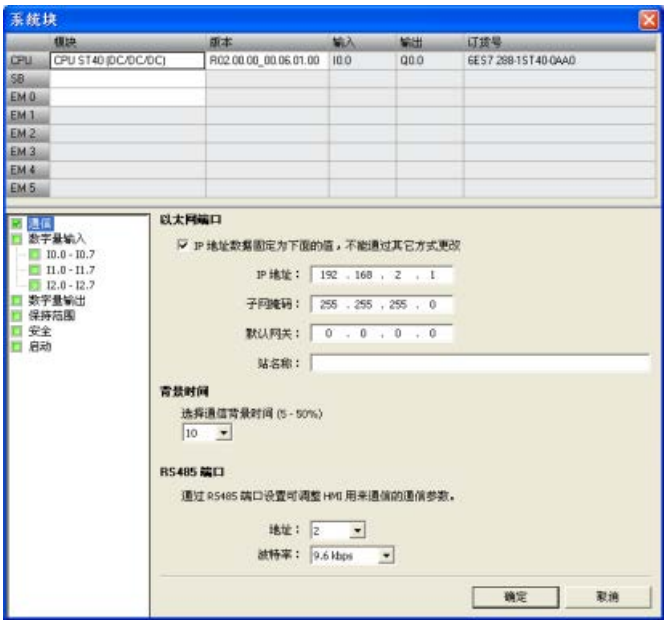
导航栏中的“系统块”(System Block) 按钮

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。

8.6.3 RS485 网络组态示例

8.6.3.1 单主站 PPI 网络

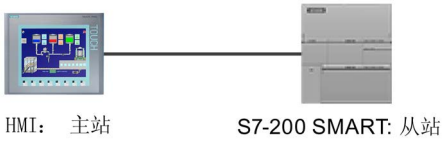
简介

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

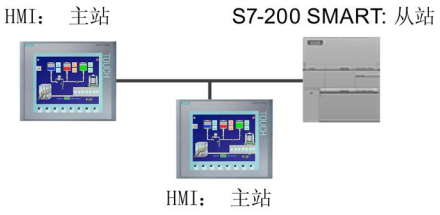
在下图的网络示例中，人机界面 (HMI) 设备（例如 TD400C、TP 或 KP）为网络主站：



在该网络示例中，CPU 是响应主站请求的从站。

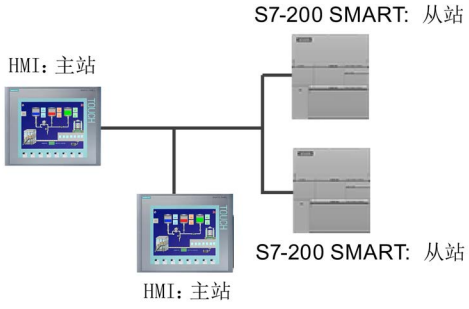
8.6.3.2 多主站和多从站 PPI 网络

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。

8.6.4 构建网络

8.6.4.1 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。

8.6.4.2 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 8- 16　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 千波特到 187.5 千波特	50 m	1,000 m
500 千波特	不受支持	400 m
1 兆波特到 1.5 兆波特	不受支持	200 m
3 兆波特到 12 兆波特	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。该距离为段中第一个节点到最后一个节点的距离。		

8.6.4.3 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**

  向网络添加中继器允许您将网络再扩展 50 米。如果将两台中继器连接在一起，中间无其它节点（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。

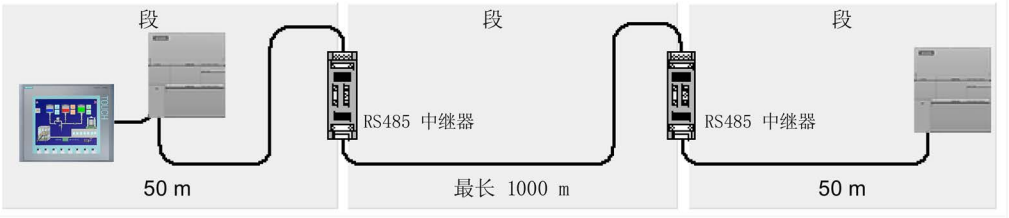
- **向网络添加设备**

  在 9600 波特时，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。



8.6.4.4 选择网络电缆

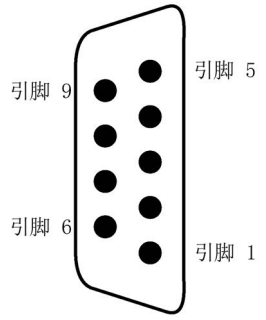
S7-200 SMART CPU 网络对双绞线电缆采用 RS485 标准。
下表列出了网络电缆的规格。 一个网段中最多可以连接 32 台设备。

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

8.6.4.5 连接器引脚分配

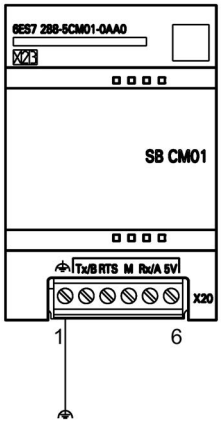
S7-200 SMART CPU 上的 RS485 通信端口是 RS485 兼容的九针超小 D 型连接器，符合欧洲标准 EN 50170 中定义的 PROFIBUS 标准。下表列出了为通信端口提供物理连接的连接器，并介绍了通信端口的引脚分配。

表格 8- 17　S7-200 SMART CPU 集成 RS485 端口（端口 0）的引脚分配

引脚编号	连接器	信号	集成 RS485 端口（端口 0）
1		屏蔽	机壳接地
2		24 V 回流	逻辑公共端
3		RS485 信号 B	RS485 信号 B
4		请求发送	RTS (TTL)
5		5 V 回流	逻辑公共端
6		+5 V	+5 V，100 Ω 串联电阻
7		+24 V	+24 V
8		RS485 信号 A	RS485 信号 A
9		不适用	10 位协议选择（输入）
连接器外壳		屏蔽	机壳接地

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 8- 18　S7-200 SMART CM01 信号板 (SB) 端口（端口 1）的引脚分配

引脚编号	连接器	信号	CM01 信号板 (SB) 端口（端口 1）：
1		接地	机壳接地
2		Tx/B	RS232-Tx/RS485-B
3		请求发送	RTS (TTL)
4		M 接地	逻辑公共端
5		Rx/A	RS232-Rx/RS485-A
6		+5 V DC	+5 V，100 Ω 串联电阻

8.6.4.6　偏置和端接网络电缆

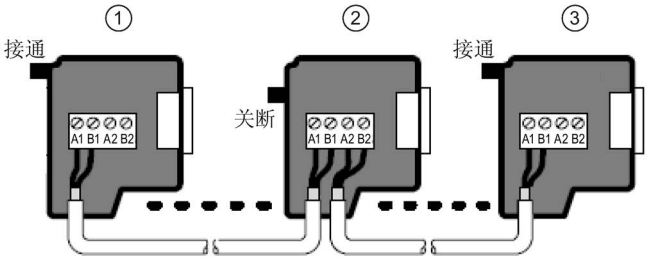
Siemens
提供了两种类型的网络连接器，可使用它们轻松地将多台设备连接到一个网络：

- 标准网络连接器
- 包括一个允许您将 HMI 设备连接到网络而不会干扰任何现有网络连接的端口的连接器

编程端口连接器负责将所有信号（包括电源引脚信号）从 S7-200 SMART CPU 传送到编程端口，这一点对于连接通过 S7-200 SMART CPU 供电的设备（如 TD 400C）尤为有用。

两个连接器都带有两组端子螺丝，分别用于连接输入和输出网络电缆。两个连接器也都带有开关，用于有选择地偏置和端接网络。下图显示了电缆连接器的典型偏置和端接。

表格 8- 19　电缆连接器的偏置和端接

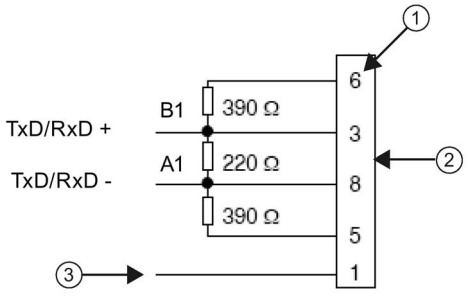
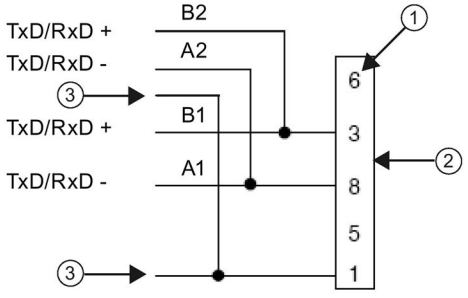
电缆两端必须端接且偏置。无屏蔽：所有位置的金属导线必须接触大约 12 mm (1/2 in)。	
	

① 开关位置 = 开 (On)：端接且偏置

② 开关位置 = 关 (Off)：无端接或偏置

③ 开关位置 = 开 (On)：端接且偏置

表格 8- 20　端接和偏置开关位置

开关位置 = 开 (On)：端接和偏置	开关位置 = 关 (Off)：无端接或偏置
	

① 引脚编号

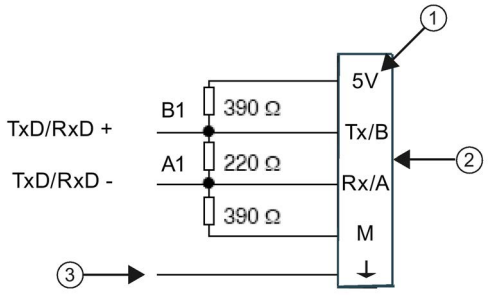
② 网络连接器

③ 电缆屏蔽

8.6.4.7 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。

信号板负责将所有信号（包括电源引脚信号）从 S7-200 SMART CPU 传送到编程端口，这一点对于连接通过 S7-200 SMART CPU 供电的设备（如 TD 400C）尤为有用。



① 端子名称
② 端子排
③ 电缆屏蔽

8.6.4.8 在 RS485 网络中使用 HMI 设备

简介

S7-200 SMART CPU 支持 Siemens 以及其他制造商生产的多种类型的 RS485 HMI 设备。 虽然其中一些 HMI 设备（如 TD400C）不允许您选择设备使用的通信协议，但其它设备（如 KP 和 TP 产品系列）允许您选择该设备的通信协议。

准则

如果 HMI 设备允许您选择通信协议，请考虑以下准则：

- 对于连接到 CPU 通信端口的 HMI 设备，如果网络中没有其它设备，请为该 HMI 设备选择 PPI 协议。
- 对于连接到已组态为主站的 CPU 通信端口的 HMI 设备，请为该 HMI 设备选择 PPI 协议。 PPI 高级协议是最佳选择。

有关如何组态 HMI 设备的详细信息，请参见设备的特定手册（见下表）。
这些手册包含在 STEP 7-Micro/WIN SMART 文档 CD 中。

表格 8- 21　S7-200 SMART CPU 支持的 RS485 HMI 设备

HMI	组态软件
TD400C	文本显示向导（STEP 7-Micro/WIN SMART 的组成部分）
KTP600 DP	WinCC flexible
KTP1000 DP	WinCC flexible

8.6.5 自由端口模式

8.6.5.1 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

要启用自由端口模式，请使用特殊存储器字节 SMB30（用于集成的 RS485 端口（端口 0））和 SMB130（用于 CM01 信号板 (SB) 端口（端口 1））。
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：**

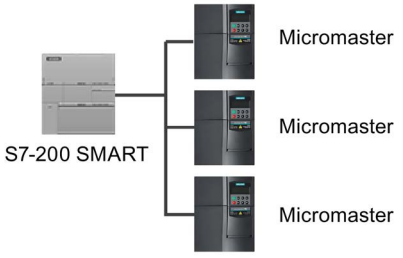
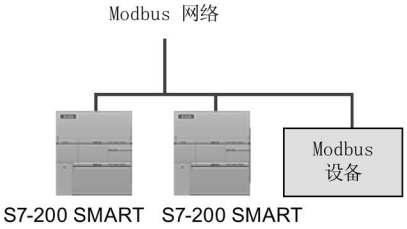
  接收字符中断会通知用户程序已在 COM 端口上接收到字符。
  随后，程序将根据所执行的协议对该字符进行处理。

- **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。

表格 8- 22　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)（仅 S CPU），您可将 RS232 设备直接连接到 CPU SB RS232，无需 PC/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 1200 波特到 115.2 千波特。 • 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。

8.6.5.2 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)（仅 S CPU）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 PC/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 1200 波特到 115.2 千波特。 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。
下表列出了波特率和对应的开关位置。

表格 8- 23　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。

8.7 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 千波特）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 9

9.1 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 492)、开放式用户通信 (页 517)和 USS 协议 (页 569)）
- 用户自定义 (页 592)（从项目 POU 创建或从其它来源获取的库）

说明

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。

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。

Modbus RTU

对于通过 CPU 串口进行的 Modbus 通信，STEP 7-Micro/WIN SMART 通过包含预组态子例程和中断例程，使得与 Modbus 设备的通信更为便捷。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。

可在项目树中“指令”(Instructions) 文件夹的“库”(Libraries) 文件夹中找到这些指令。向程序中加入 Modbus RTU 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和中断例程。

开放式用户通信

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

可在项目树中“指令”(Instructions) 文件夹的“库”(Libraries) 文件夹中找到这些指令。向程序中加入 OUC 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。

USS 协议

USS 协议库支持 Siemens 变频器。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。

可在项目树中“指令”(Instructions) 文件夹的“库”(Libraries) 文件夹中找到这些指令。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您的网络，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 (http://www.industry.siemens.com/topics/global/en/industrial-security/Documents/operational_guidelines_industrial_security_en.pdf)

9.2 Modbus 通信概述

对于通过 CPU 串口进行的 Modbus RTU 通信，STEP 7-Micro/WIN SMART 和 S7-200 SMART CPU 通过包含预组态的子例程和中断例程，使得与 Modbus 设备的通信更为便捷。

9.2.1 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 主站寻址

Modbus RTU 主站指令将地址映射至正确函数，从而发送到从站设备。（地址对应于 MBUS_MSG/MB_MSG2 (页 499) 指令的 Addr 输入参数。）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 从站寻址

Modbus RTU 从站指令支持以下地址：

- 00001 至 00256 是映射到 Q0.0 - Q31.7 的离散量输出。
- 10001 至 10256 是映射到 I0.0 - I31.7 的离散量输入。
- 30001 至 30056 是映射到 AIW0 - AIW110 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 9- 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
...		...
00255		Q31.6
00256		Q31.7
10001		I0.0
10002		I0.1
10003		I0.2
...		...
10255		I31.6
10256		I31.7
30001		AIW0
30002		AIW2
30003		AIW4
...		...
30056		AIW110
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)

MBUS_INIT 参数，限制从站可访问性

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** 指定允许 Modbus 主站访问的离散量输入或输出（I 或 Q）的最大数目。
- **MaxAI** 指定允许 Modbus 主站访问的输入寄存器 (AIW) 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。

有关为 Modbus 从站设置存储器限制的详细信息，请参见 MBUS_INIT (页 507) 指令说明。

9.2.2 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 9- 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。有些从站设备支持的数据可能小于 240 个字节。

9.3 Modbus RTU 库

9.3.1 Modbus 通信概述

9.3.1.1 Modbus RTU 库功能

STEP 7-Micro/WIN SMART 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 支持主站和从站设备均通过 RS-485（集成端口 0 和可选信号板端口 1）和 RS-232（仅限可选信号板端口 1）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。您最多可以配置 2 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 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。

9.3.1.2 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 (页 497) 会初始化 Modbus 主站协议，并使分配的 CPU 端口（0 或 1）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它用途，包括与 HMI 的通信。

- 对于由 MBUS_CTRL/MB_CTRL2 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
  - 1942 个字节的程序空间用于存储两个主站指令和支持例程
  - Modbus 主站指令的变量需要 286 个字节的 V 存储器块。您必须使用 STEP 7-Micro/WIN SMART 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 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 507)会初始化 Modbus 从站协议，并使分配的 CPU 端口（0 或 1）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它用途，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，用于两个从站指令和支持例程。
  - Modbus 从站指令的变量需要 786 个字节的 V 存储器块。您必须使用 STEP 7-Micro/WIN SMART 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)。

9.3.1.3 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 和 MB_CTRL2 指令（如果有）。MBUS_CTRL/MB_CTRL2 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 指令的执行延长了扫描进间，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 指令和任何其它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。

9.3.2 Modbus RTU 主站

9.3.2.1 使用 Modbus RTU 主站指令

STEP 7-Micro/WIN SMART 和 S7-200 SMART CPU 支持两种 Modbus RTU 主站。对于单个 Modbus RTU 主站，使用指令 MBUS_CTRL (页 497) 和 MBUS_MSG (页 499)。对于第二个 Modbus RTU 主站，使用指令 MBUS_CTRL2 (页 497) 和 MBUS_MSG2 (页 499)。

如果您在项目中使用两个 Modbus 主站，则要确保 MBUS_CTRL 和 MB_CTRL2 使用不同的端口号。

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：

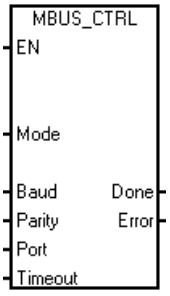
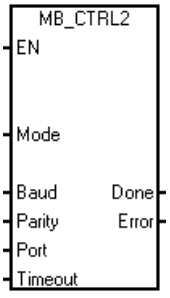
1. 在程序中插入 MBUS_CTRL/MB_CTRL2 指令并在每次扫描时执行。您可以使用 MBUS_CTRL/MB_CTRL2 指令启动或更改 Modbus 通信参数。当您插入 MBUS_CTRL/MB_CTRL2 指令时，STEP 7-Micro/WIN SMART 会在程序中添加几个受保护的子例程和中断例程。
2. 在“文件”(File) 菜单功能区的“库”(Libraries) 区域中，单击“存储器”(Memory) 按钮 存储器，指定 Modbus 库所需的 V 存储器的起始地址。或者，也可在项目树中右键单击“程序块”(Program Block) 节点，并从上下文菜单中选择“库存储器”(Library Memory)。
3. 在程序中放置一条或多条 MBUS_MSG/ MB_MSG2 指令。可以根据需要在程序中添加任意数量的 MBUS_MSG/MB_MSG2 指令，但某一时间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 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。

9.3.2.2 MBUS_CTRL/MB_CTRL2 指令（初始化主站）

MBUS_CTRL 和 MB_CTRL2 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。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。

表格 9- 3 MBUS_CTRL 和 MB_CTRL2 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**	程序调用 MBUS_CTRL/MB_CTRL2 指令来初始化、监视或禁用 Modbus 通信。 在执行 MBUS_MSG/MB_MSG2 指令前，程序必须先执行 MBUS_CTRL/MB_CTRL2 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。

必须在每次扫描时（包括首次扫描）调用 MBUS_CTRL/MB_CTRL2 指令，以便其监视 MBUS_MSG/MB_MSG2 指令启动的任何待处理消息的进程。除非每次扫描时都执行 MBUS_CTRL/MB_CTRL2，否则 Modbus 主站协议将不能正确工作。

表格 9- 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**“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**“端口”(Port)**设置物理通信端口（0 = CPU 中集成的 RS-485，1 = 可选 CM01 信号板上的 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 指令完成时，指令将“真”(TURE) 返回给“完成”(Done) 输出。

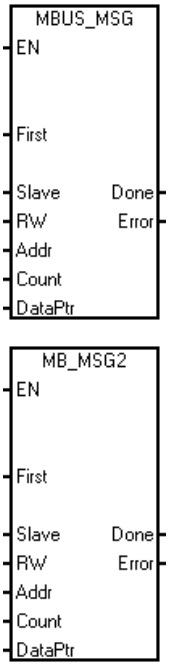
“错误”(Error) 输出包含指令执行的结果。

另请参见 Modbus RTU 主站执行错误代码 (页 503)

9.3.2.3 MBUS_MSG/MB_MSG2 指令

MBUS_MSG 和 MB_MSG2 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。

表格 9- 5 MBUS_MSG/MB_MSG2 指令

LAD/FBD	STL	说明
	**CALL MBUS_MSG, First, Slave, RW, Addr, Count, DataPtr, Done, Error** **CALL MB_MSG2, First, Slave, RW, Addr, Count, DataPtr, Done, Error**	程序调用 MBUS_MSG/MB_MSG2 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 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。

某一时间只能有一条 MBUS_MSG 或 MB_MSG2 指令处于激活状态。如果程序启用多条 MBUS_MSG 指令或多条 MB_MSG2 指令，则 CPU 将处理第一条 MBUS_MSG 指令或 MB_MSG2 指令，所有后续 MBUS_MSG 或 MB_MSG2 指令将中止并生成错误代码 6。

表格 9- 6　MBUS_MSG/MB_MSG2 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 512)。

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 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。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。

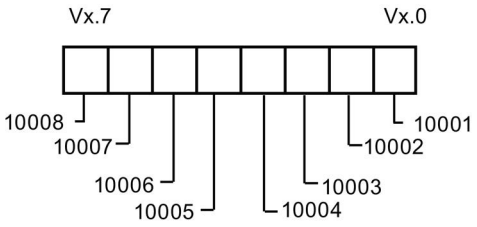
存储器布局

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 9- 7　Modbus 保持寄存器

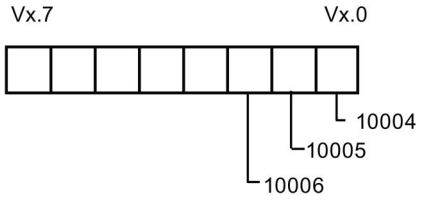
CPU 存储器字节地址		CPU 存储器字地址		Modbus 保持寄存器地址	
地址	十六进制数据	地址	十六进制数据	地址	十六进制数据
VB200	12	VW200	12 34	40001	12 34
VB201	34				
VB202	56	VW202	56 78	40002	56 78
VB203	78				
VB204	9A	VW204	9A BC	40003	9A BC
VB205	BC				

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)）。如果您打算仅写入单个位，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 指令。

输出

程序已发送请求并接收响应后，Done 输出为 FALSE。响应完成或 MBUS_MSG/MB_MSG2 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 503)才有效。

9.3.2.4 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	说明
0	无错误
1	奇偶校验类型无效
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

MBUS_MSG 错误代码	说明
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
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。

MBUS_MSG 错误代码	说明
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。

9.3.3 Modbus RTU 从站

9.3.3.1 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“存储器”(Memory) 按钮 存储器，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。

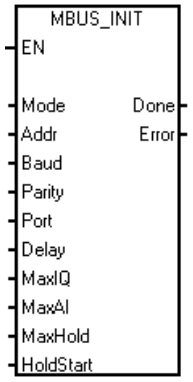
累加器（AC0、AC1、AC2、AC3）由 Modbus
从站指令使用，并显示在“交叉引用”列表中。在执行 Modbus 从站指令前，系统会先保存 Modbus 从站指令累加器中的值，在 Modbus
从站指令完成前恢复到累加器中，从而确保在执行 Modbus
从站指令时保留累加器中的所有用户数据。

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。

9.3.3.2 MBUS_INIT 指令（初始化从站）

表格 9- 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 9- 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**“端口”(Port)** 设置物理通信端口（0 = CPU 中集成的 RS-485，1 = 可选信号板上的 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 256。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 256。

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 56。值为 0 时，将禁止读取模拟量输入。建议将 MaxAI 设置为以下值，以允许访问所有 CPU 模拟量输入：

- 0（针对 CPU CR40 和 CR60）
- 56（所有其它 CPU 型号）

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 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）。

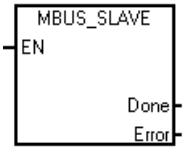
参数 **HoldStart** 是 V 存储器中保持寄存器的起始地址。该值通常设置为 VB0，因此参数 HoldStart 设置为 &VB0（地址 VB0）。也可将其它 V 存储器地址指定为保持寄存器的起始地址，以便在项目中的其它位置使用 VB0。Modbus 主站可访问起始地址为 HoldStart，字数为 MaxHold 的 V 存储器。

MBUS_INIT 指令完成时，“完成”(Done) 输出接通。

Error 输出 (页 511)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。

9.3.3.3 MBUS_SLAVE 指令

表格 9- 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 9- 11　MBUS_SLAVE 指令的参数

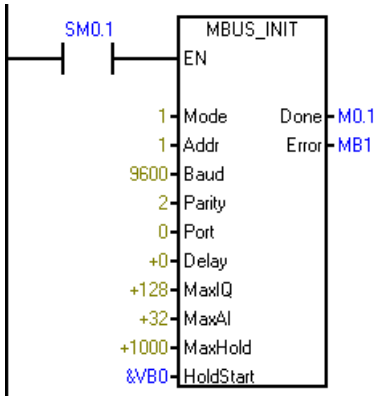
参数	数据类型	操作数
Done	BOOL	I、Q、M、S、SM、T、C、V、L
Error	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD

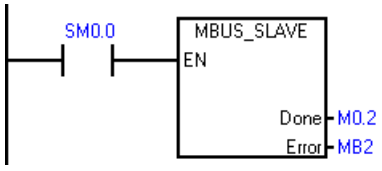
当 MBUS_SLAVE 指令响应 Modbus 请求时，“完成”(Done) 输出接通。如果未处理任何请求，“完成”(Done) 输出关闭。

Error 输出 (页 511)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 9- 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`

9.3.3.4 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
11	端口号无效
12	信号板端口 1 缺失或未组态

9.3.4 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 指令的示例同样适用。

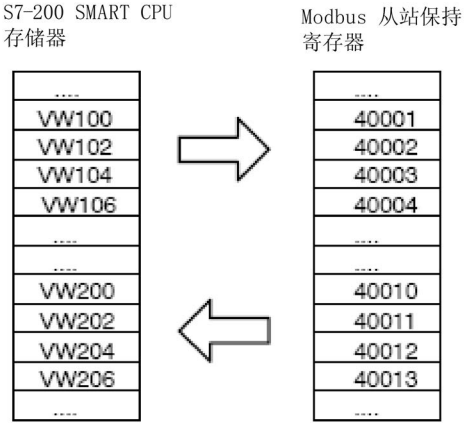
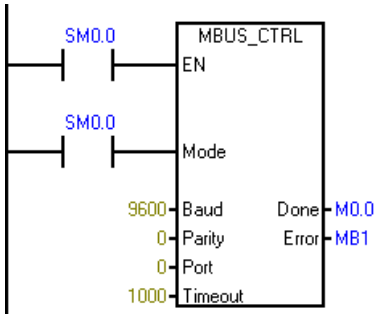
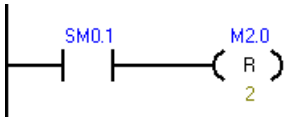
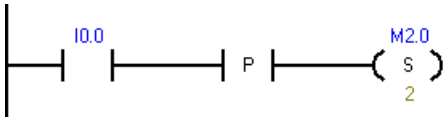
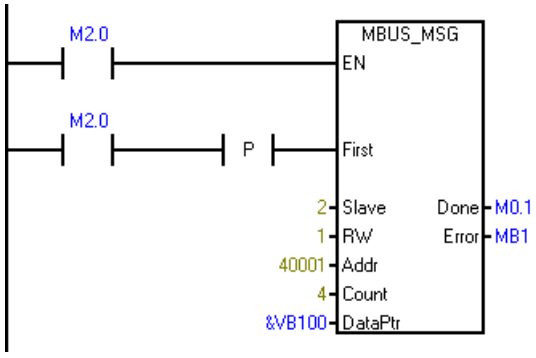
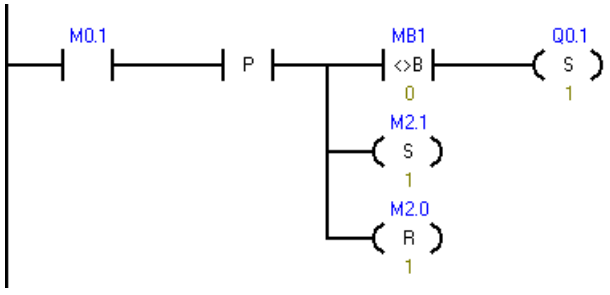
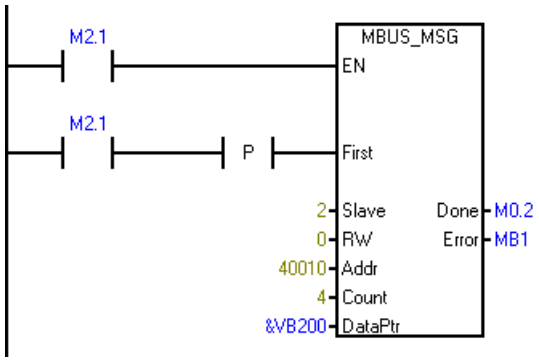
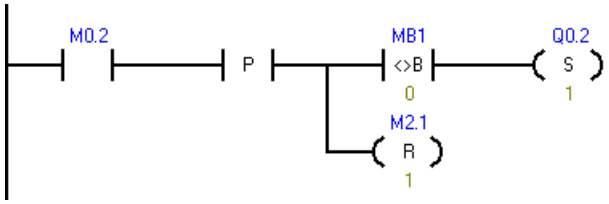


图 9-1　　程序数据传送示例

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 9- 13　Modbus 主站示例程序

LAD	说明
	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 9600 波特，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	说明
	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。

9.3.5 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 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 之后，可在 Modbus 主站符号表中找到符号 mModbusRetries 并更改该值，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 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 之后，可在 Modbus 主站符号表中找到符号 *mModbusCharTimeout*，然后更改该值，以此延长该超时时间。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 指令返回错误代码 101。

Modbus 主站协议允许用户强制 MBUS_MSG/MB_MSG2 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 之后，可在 Modbus 主站符号表中找到符号 *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 指令允许参数 **Addr** 采用其它范围，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 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。

9.4 开放式用户通信库

STEP 7-Micro/WIN SMART 开放式用户通信 (OUC) 库指令创建 OUC 指令 (页 222)（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。

9.4.1 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。

9.4.2 开放式用户通信库指令

9.4.2.1 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。

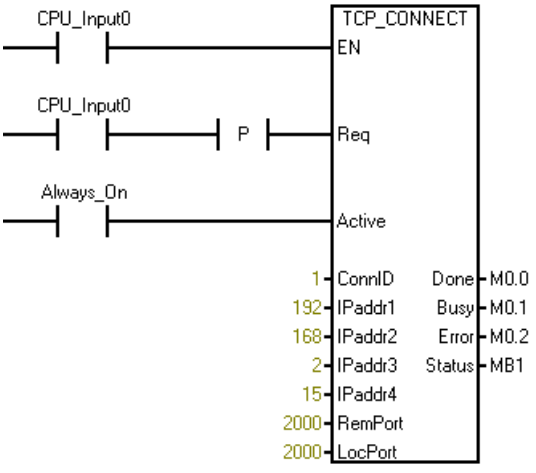
表格 9- 14　TCP_CONNECT 指令的参数

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。

参数	声明	数据类型	描述
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 518)。
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 545)。
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：



9.4.2.2 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。

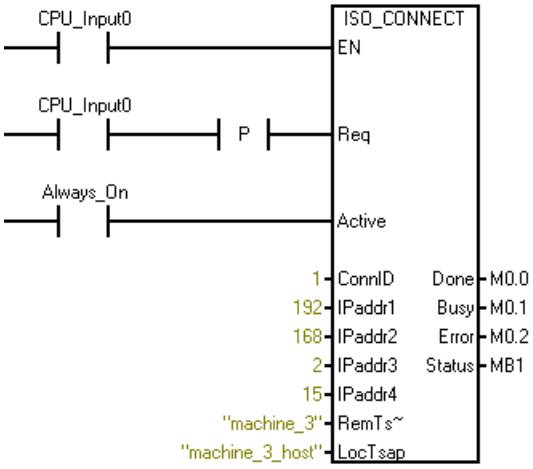
表格 9- 15　ISO_CONNECT 指令的参数

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。

参数	声明	数据类型	描述
RemTsap	IN	DWORD	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	DWORD	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。
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 545)。
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：

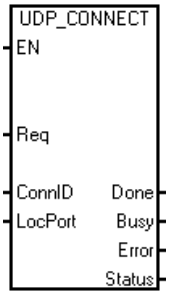


对于 RemTsap 和 LocTsap，STEP 7 Micro/WIN SMART 始终使用指针将字符串传递给 ISO_CONNECT 指令。如果您使用常量字符串（如以上示例所示），STEP7-Micro/WIN SMART
会自动创建字符串和指针。如果您希望在数据块中创建字符串，然后将指针传递给这些字符串之一，请执行以下步骤：

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。

9.4.2.3 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：

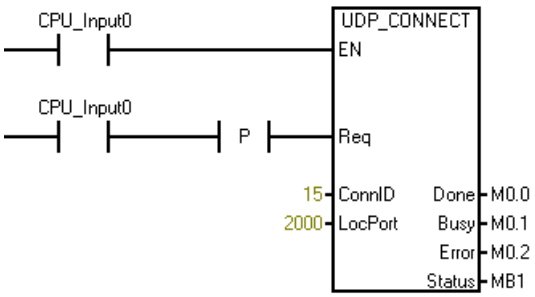
- 如果连接处于激活状态并准备发送或接收，指令置位 Done 输出。这仅表示您可以使用该连接，并不表示存在任何远程设备。
- 如果连接仍在进行，指令置位 Busy 输出。
- 如果连接不可用，指令置位 Error 输出。Status 输出字节显示其中一种错误代码，用于指示存在的问题。

表格 9- 16　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 518)。
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 545)。
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）。

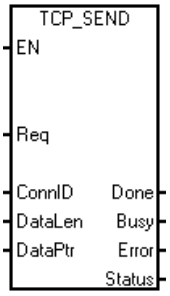
示例

这是 UDP_CONNECT 指令的应用示例：



9.4.2.4 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。

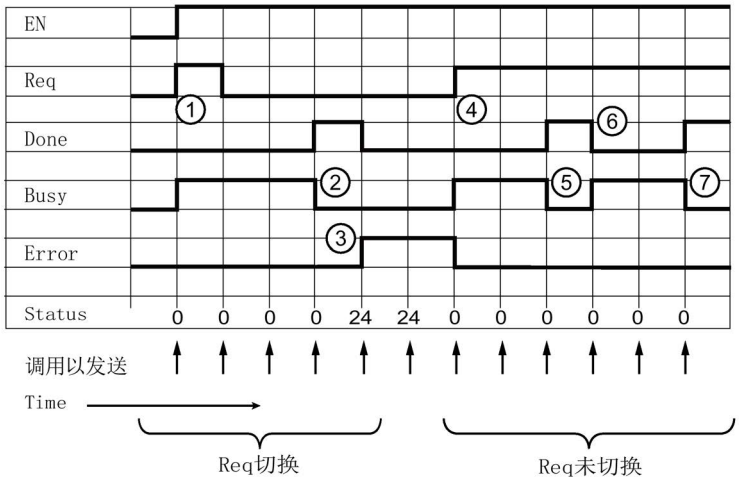
LAD/FBD	STL	描述
	**TCP_SEND Req, ConnID, DataLen, DataPtr, Done, Busy, Error, Status**	TCP_SEND 通过现有连接传输来自请求的缓冲区位置的请求的字节数。

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。

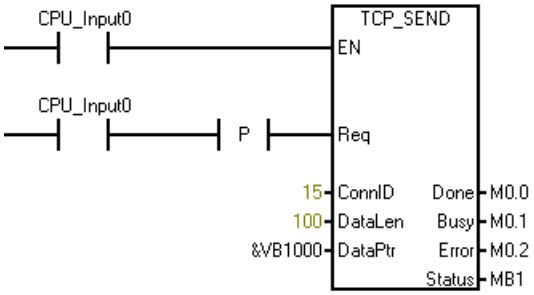
在一个发送操作中最多可以发送 1024 字节的数据。若在 Req 输入设置为 TRUE 时执行 TCP_SEND，程序会将用户存储器中发送缓冲区的数据复制到内部缓冲区。TCP_SEND 执行且指令置位 Busy 输出后，您可以更改程序发送缓冲区。

表格 9- 17　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）。
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 545)。
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）。

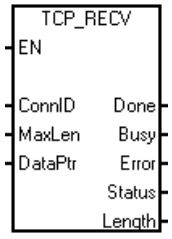
示例

这是 TCP_SEND 指令的应用示例：



9.4.2.5 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。

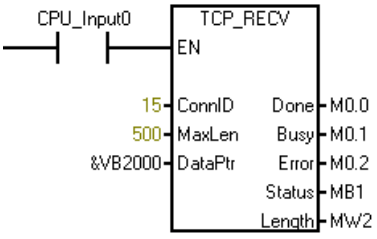
表格 9- 18　TCP_RECV 指令的参数

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 545)。

参数	声明	数据类型	描述
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。

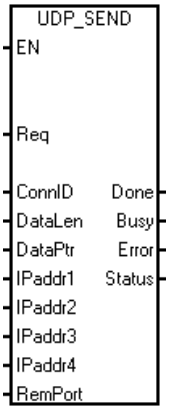
示例

这是 TCP_RECV 指令的应用示例：



9.4.2.6 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。

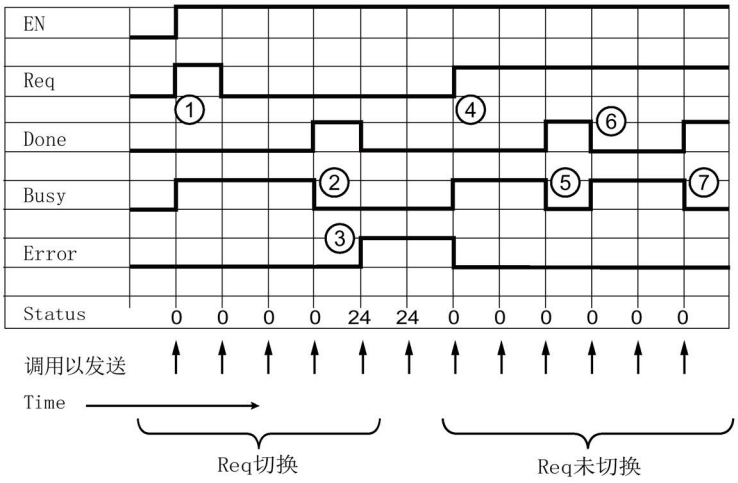
LAD/FBD	STL	描述
	**UDP_SEND Req, ConnID, DataLen, DataPtr, IPaddr1, IPaddr2, IPaddr3, IPaddr4, RemPort, Done, Busy, Error, Status**	UDP_SEND 指令将来自请求的缓冲区位置的请求的字节数传输到通过 IP 地址和端口指定的设备。

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 518)。）

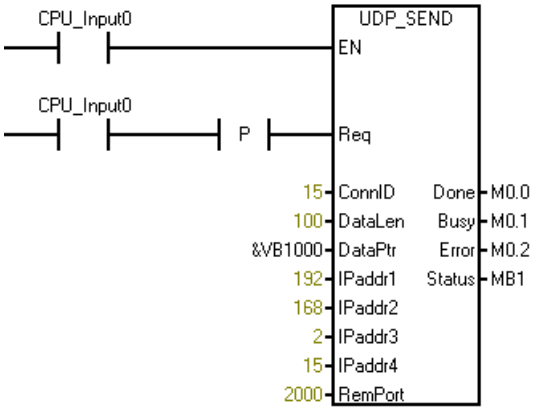
请注意，不能保证会传递 UDP 消息。如果远程设备不存在，不会返回错误。

表格 9- 19　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 545)。
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）。

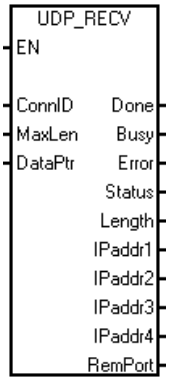
示例

这是 UDP_SEND 指令的应用示例：



9.4.2.7 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。

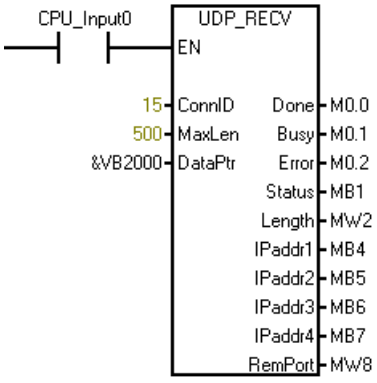
表格 9- 20　UDP_RECV 指令的参数

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 545)。
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。

参数	声明	数据类型	描述
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：



9.4.2.8 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。

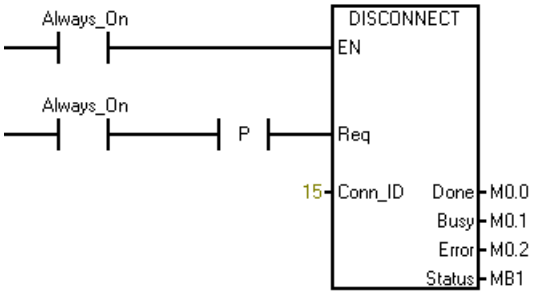
表格 9- 21　DISCONNECT 指令的参数

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。

参数	声明	数据类型	描述
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 545)。
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：



9.4.3 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 517)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
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	

错误代码	描述	CONNECT	SEND	RECV	DISCONNECT
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			
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

9.4.4 开放式用户通信库示例

9.4.4.1 主动伙伴（客户端）

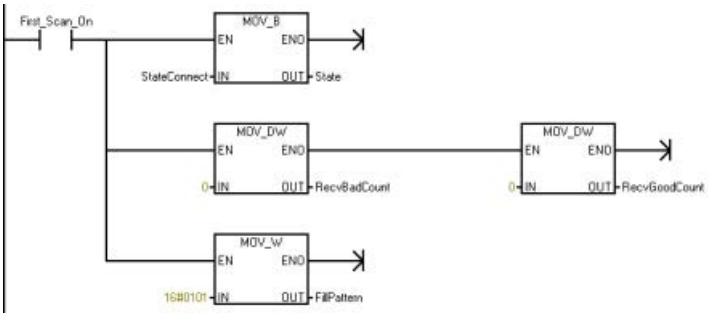
该程序用于实现简单状态机管理连接创建，周期性发送/接收消息，以及处理错误。

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 558)。

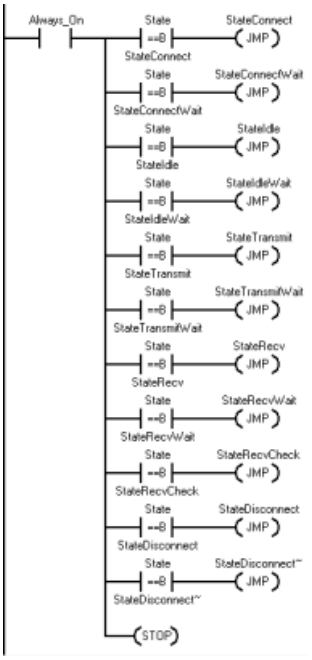
程序段 1：第一次扫描时....

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。

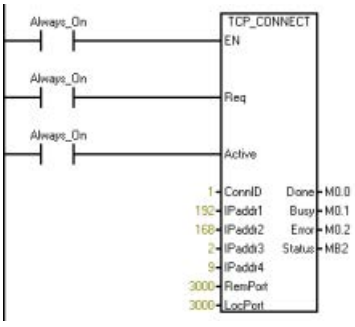


程序段 3：状态“连接”...

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。

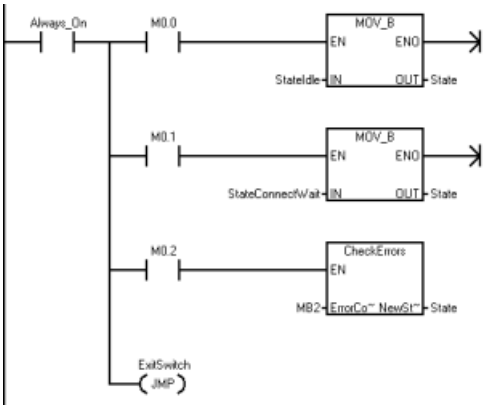


程序段 5：如果 Done 为 TRUE，则 CPU 建立该连接以进入“空闲”状态。

如果 Busy 为 TRUE，则 CPU 进入“连接等待”状态以等待连接的建立。

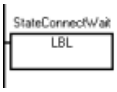
如果 Error 为 TRUE，则可能输入参数有误，检查输入参数以决定 CPU 接下来要进入哪个状态。

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。

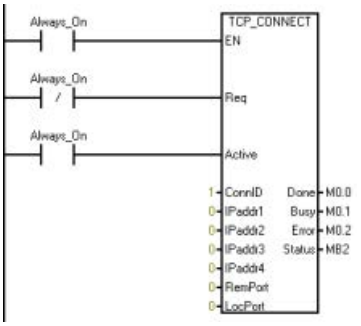


程序段 6：状态“连接等待”...

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。

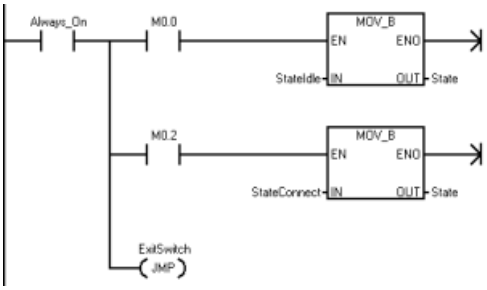


程序段 8：如果 Done 为 TRUE，则表示 CPU 已建立该连接，所以继续进入“空闲”状态。

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。

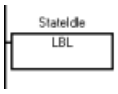
如果 Error 为
TRUE，则与被动设备的连接有误。在这种情况下，返回到“连接”状态并尝试再次建立连接。注意如果已搜索到被动设备，但其拒绝连接请求，则由于 CPU
不断尝试与被动设备建立连接，会快速返回连接错误并占用大量带宽。

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。

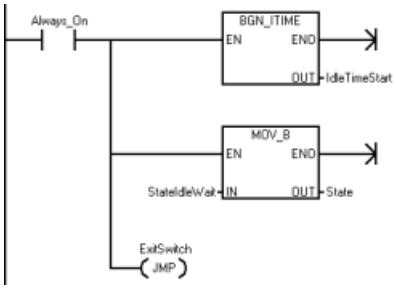


程序段 9：状态“空闲”...

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。



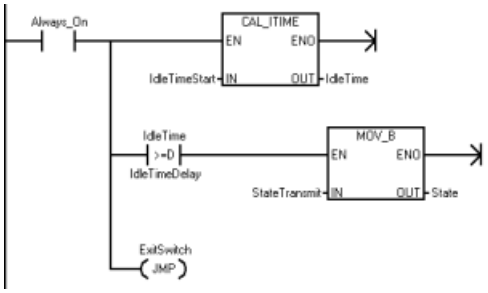
程序段 10：捕获间隔计时器并在下次传送前都处于“空闲等待”状态以延迟进程。



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)。

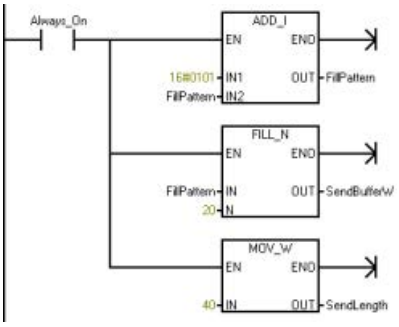
程序段 12：计算从进入“空闲”状态到现在的时间，如果其超过“IdleTimeDelay”值，则将状态更改为“传送”状态。



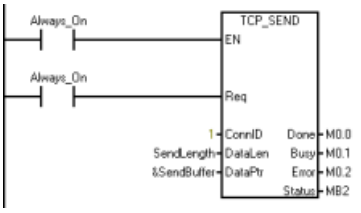
程序段 13：状态“传送”...

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
。



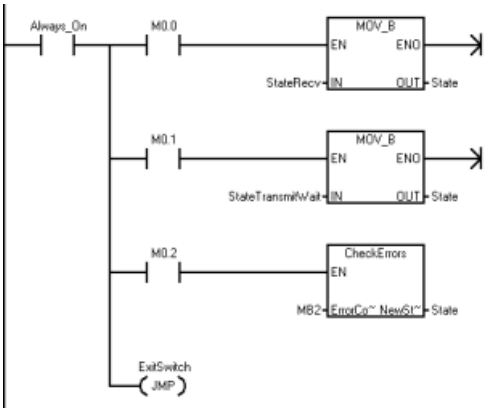
程序段 15：向被动设备发送新消息。设置 Req 为 TRUE 以初始化新的发送操作。



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。

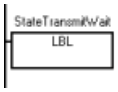
如果 Busy 为 TRUE（通常是这种情况），则进入“传送等待”状态以等待传送完成。

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。



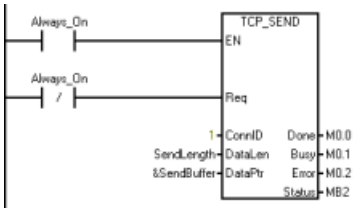
程序段 17：状态“传送等待”...

在该状态等待直到传送完成。



程序段 18：设置 Req = FALSE，调用 TCP_SEND

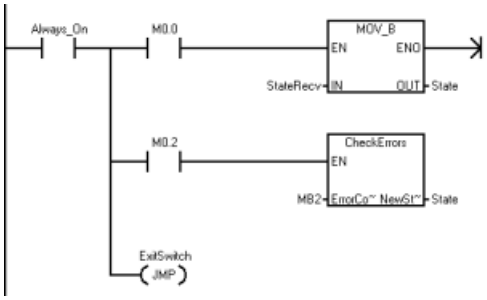
指令以确定发送是否已完成。确保使用初始化发送请求所用的发送长度和缓冲区指针。



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。



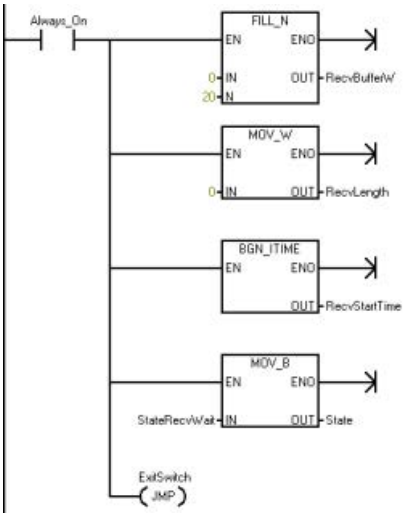
程序段 20：状态“接收”...

清空接收数据区，准备接收响应。

程序段

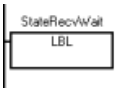
21：清理接收缓冲区和“RecvLength”，由此该区域内不会残留有上一个接收到的消息的数据。

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。

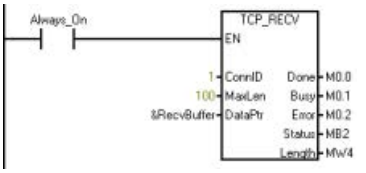


程序段 22：状态“接收等待”...

停留在该状态直到接收到某些数据或超时。



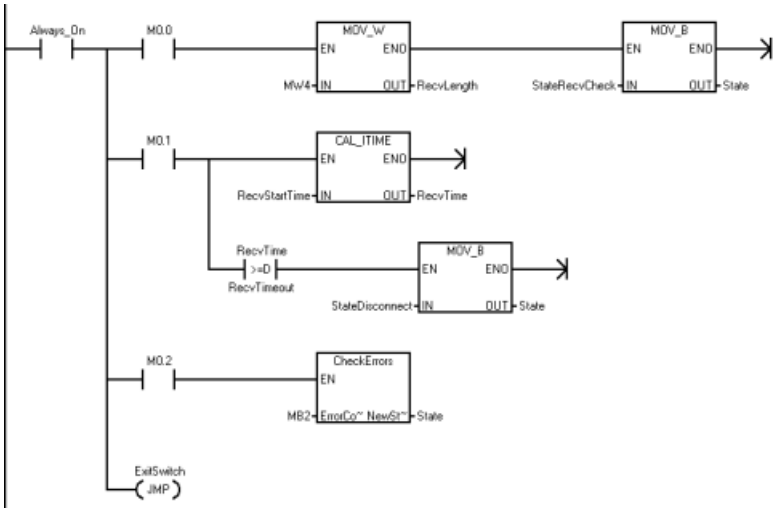
程序段 23：调用 TCP_RECV 指令以获取 CPU 接收到的任意消息。



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。

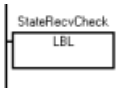
如果 Busy 为
TRUE，则停留在“接收”状态，直到停留时间达到接收超时值。如果停留在该状态的时间已超时，则断开与设备的连接，之后再重新与其建立连接。

如果 Error 为 TRUE，则检查错误代码以确定接下来的操作。

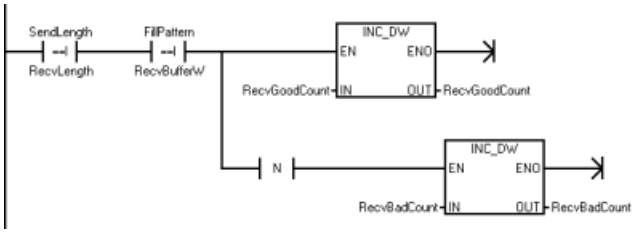


程序段 25：状态“RecvCheck”...

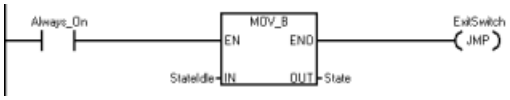
处理被动设备返回的数据。



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。



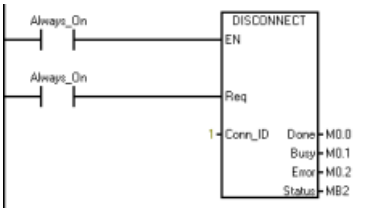
程序段 27：在所有情况下都进入“空闲”状态。



程序段 28：状态“断开连接”...

启动断开连接。

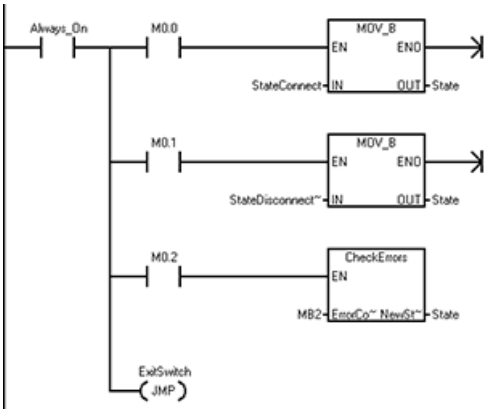
程序段 29：通过设置 Req = TRUE，调用 DISCONNECT 指令来启动与 ConnID 断开连接。



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。

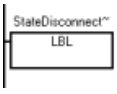
如果 Busy 为 TRUE（通常是这种情况），则进入“断开连接等待”状态。

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。

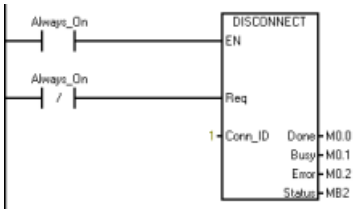


程序段 31：状态“断开连接等待”...

在该状态等待直到断开连接操作完成。



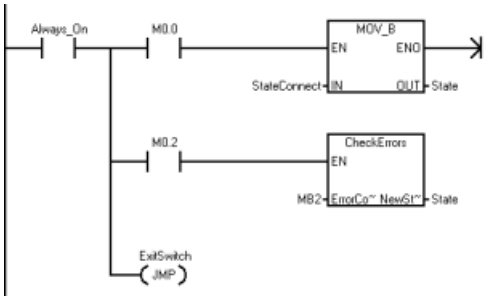
程序段 32：设置 Req = FALSE，调用 DISCONNECT 指令，以检查断开连接操作的状态。



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。

9.4.4.2 CheckErrors 子例程

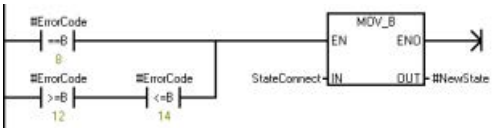
CheckErrors

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”。

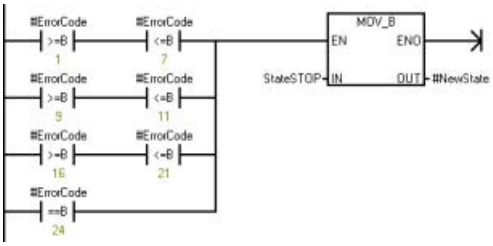


程序段 3：如果出现参数错误（错误代码 1 - 7），则停止该程序，因为该功能的某项输入存在组态错误。更改程序中的错误。

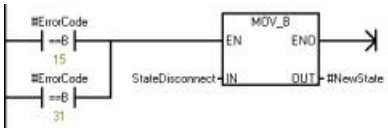
如果错误代码为 9（正在连接）、10（正在断开连接）、或 11（正在发送），则停止该程序因为状态机已损坏。设置该状态机停留在相关等待状态直到完成修复操作，且这些错误不会再次发生。

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。

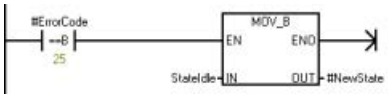
如果 SEND 和 DISCONNECT 指令返回错误 24，则表示当前没有待决操作。这可能意味着操作已完成，但是，不可能由于状态机的原因导致出现错误 24。考虑到该错误的存在，停止程序。



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。



9.4.4.3 主动伙伴符号表

下表列出了符号名称、地址和主动伙伴（客户端）程序的注释。

符号	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	
FillPattern	VW12	
RecvLength	VW16	
SendLength	VW18	
SendBufferW	VW1000	
RecvBufferW	VW2000	
StateConnect	1	
StateConnectWait	2	

符号	地址	注释
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

9.4.4.4 被动伙伴（服务器）

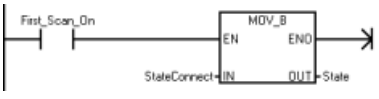
该程序用于实现简单状态机管理连接打开，接收消息，发送响应，以及处理错误。

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 568)。

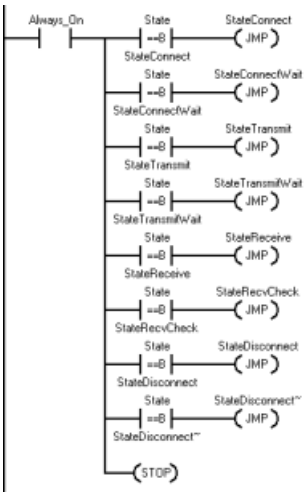
程序段 1：第一次扫描时....

初始化状态变量以初始化连接。



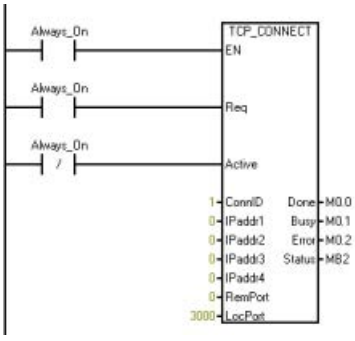
程序段 2：处理状态机...

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。

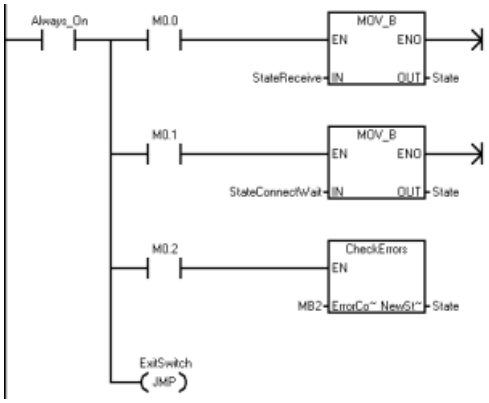


程序段 5：如果 Done 为 TRUE，则 CPU 已建立该连接，之后进入“空闲”状态。

如果 Busy 为 TRUE，则 CPU 进入“连接等待”状态以等待连接的建立。

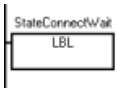
如果 Error 为 TRUE，则可能输入参数有误，检查输入参数以决定 CPU 接下来要进入哪个状态。

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。

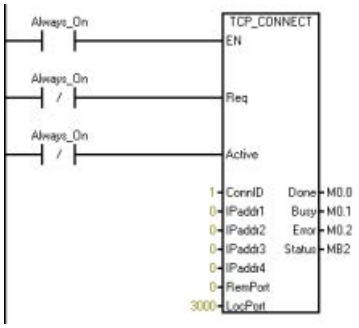


程序段 6：状态“连接等待”...

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。

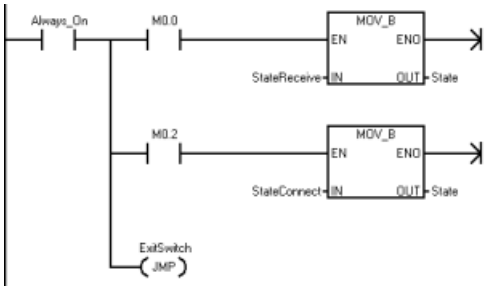


程序段 8：如果 Done 为 TRUE，则表示 CPU 已建立该连接，所以继续进入“空闲”状态。

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。

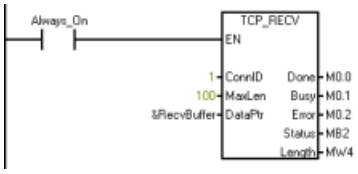
在所有情况下，都要先退出状态机才能进行该项扫描。该程序会继续进入下一项扫描的下一个状态。



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。

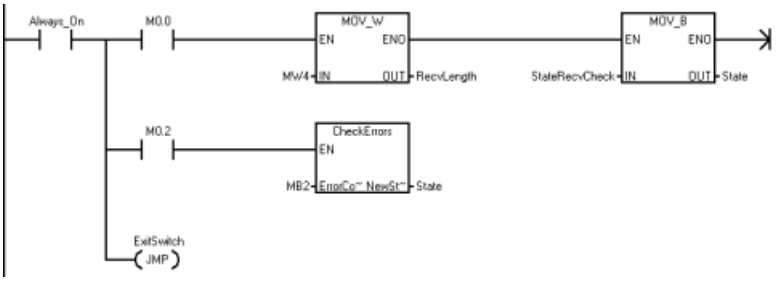


程序段 11：如果 Done 为 TRUE，则表示 CPU 已接收到新数据，之后进入“接收检查”状态。

如果 Busy 为 TRUE，则停留在“接收”状态直到接收到某些数据。

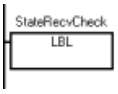
如果 Error 为 TRUE，则检查错误代码以确定接下来的操作。

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。

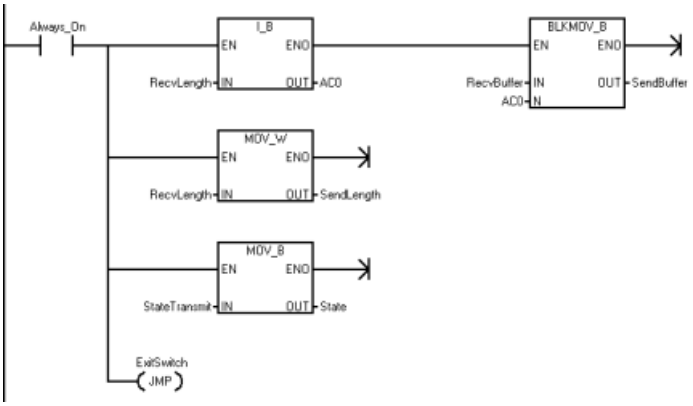


程序段 12：状态“接收检查”...

检查并处理接收到的数据。

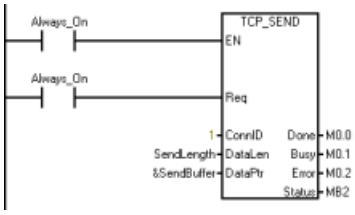


程序段 13：在该程序中，将数据传回给伙伴。将所有接收到的字节复制到发送缓冲区。更改为“传送”状态，之后退出程序，直到下一次扫描时再开启。



程序段 14：状态“传送”...

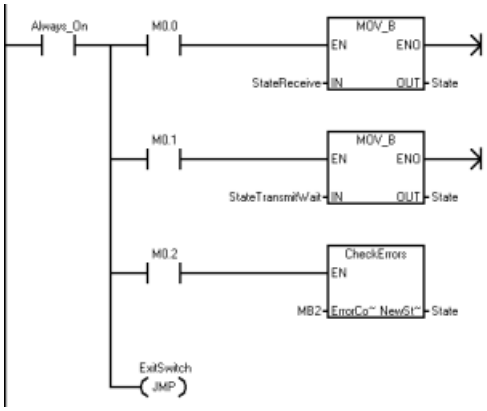
程序段 15：将数据发送回给伙伴。



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。

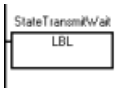
如果 Busy 为 TRUE（通常是这种情况），则进入“传送等待”状态以等待传送完成。

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。

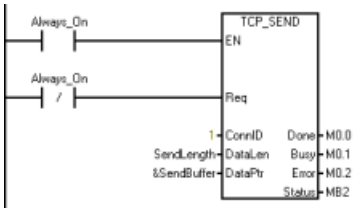


程序段 17：状态“传送等待”...

在该状态等待直到传送完成。



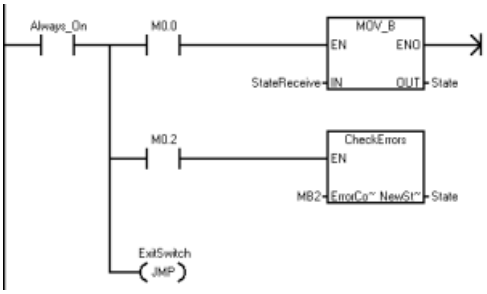
程序段 18：设置 Req = FALSE，调用 TCP_SEND 指令以确定发送完成的时间。确保使用初始化发送请求所用的发送长度和缓冲区指针。



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。

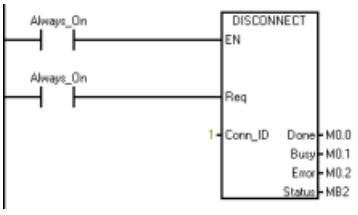
如果 Error 为 TRUE，则检查出现错误的原因，如果是连接出现问题，需要更改状态。



程序段 20：状态“断开连接”...

启动断开连接。

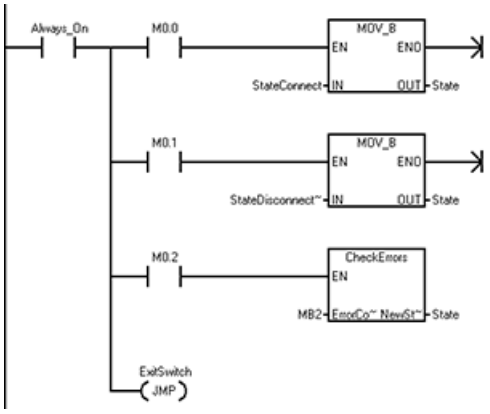
程序段 21：通过设置 Req = TRUE，调用 DISCONNECT 指令来启动与 ConnID 断开连接。



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。

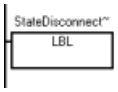
如果 Busy 为 TRUE（通常是这种情况），则进入“断开连接等待”状态。

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。

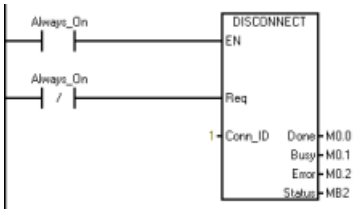


程序段 23：状态“断开连接等待”...

在该状态等待直到断开连接操作完成。



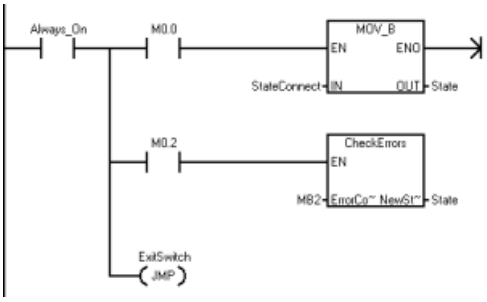
程序段 24：设置 Req = FALSE，调用 DISCONNECT 指令，以检查断开连接操作的状态。



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。

9.4.4.5 CheckErrors 子例程

CheckErrors 子例程 (页 557)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。

9.4.4.6 被动伙伴符号表

下表列出了符号名称、地址和被动伙伴（服务器）程序的注释。

符号	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	
StateConnect	1	
StateConnectWait	2	
StateTransmit	3	
StateTransmitWait	4	
StateReceive	5	
StateRecvCheck	6	
StateDisconnect	7	
StateDisconnectWait	8	
ExitSwitch	10	

9.5 USS 库

9.5.1 USS 通信概述

9.5.1.1 USS 协议概述

STEP 7-Micro/WIN SMART 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例程和中断例程，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 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 570)
- 计算与驱动器通信所需的时间 (页 571)

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 572)”。

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。

9.5.1.2 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，所有其它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它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 或端口 1 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它用途，包括与 HMI 进行通信。第二个通信端口允许 STEP 7-Micro/WIN SMART 在 USS 协议运行期间监视控制程序。
- 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 400 字节的 V 存储区。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。

9.5.1.3 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（或 USS_INIT_P1 指令分配端口 1 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 9- 22　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

9.5.2 USS 程序指令

9.5.2.1 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 存储器，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。

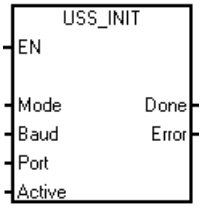
USS 协议指令由以下部分组成：

- USS_INIT (页 573)
- USS_CTRL (页 576)
- USS_RPM_X (页 581)
- USS_WPM_x (页 584)

本部分还将讨论 USS 协议程序示例 (页 589)和 USS 协议错误代码 (页 588)列表。

9.5.2.2 USS_INIT 指令

表格 9- 23 USS_INIT 指令

LAD/FBD	STL	说明
	**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。 在使用任何其它 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。

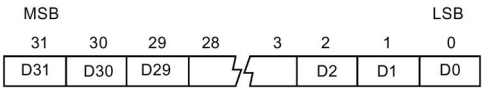
表格 9- 24 USS_INIT 指令的参数

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 9- 25　USS_INIT 参数说明

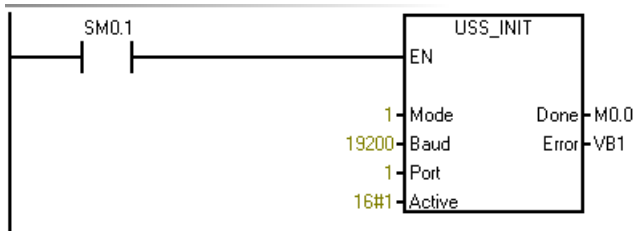
参数	说明
Mode	此值用于选择通信协议： • 输入值为 1 时，将端口分配给 USS 协议并启用该协议。 • 输入值为 0 时，将端口分配给 PPI 协议并禁用 USS 协议。
Baud	将波特率设置为 1200、2400、4800、9600、19200、38400、57600 或 115200
端口	设置物理通信端口（0 = CPU 中集成的 RS485，1 = 可选 CM01 信号板上的 RS485 或 RS232）
激活	指示激活的变频器。 有些变频器仅支持地址 0 至 30。
Done	当 USS_INIT 指令完成后接通
Error	该输出字节包含指令执行的结果。 USS 协议执行错误代码 (页 588)定义了执行该指令产生的错误状况。

表格 9- 26　激活变频器参数的格式

	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 588)。

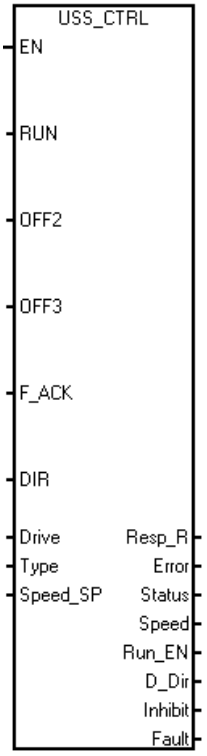
表格 9- 27　USS_INIT 示例程序

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 572)。

9.5.2.3 USS_CTRL 指令

表格 9- 28 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 9- 29　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
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。

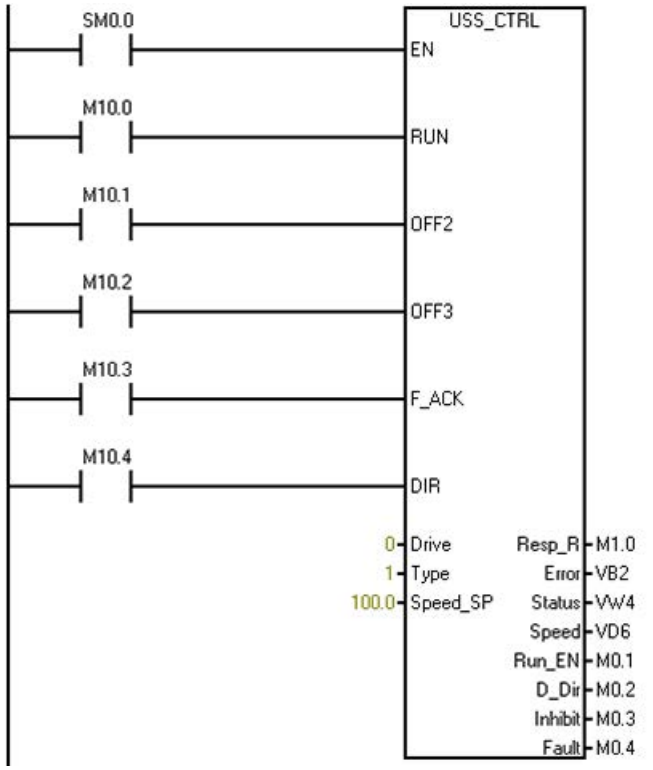
Resp_R 参数

“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
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 588)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

参数	说明
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 9- 30　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
```

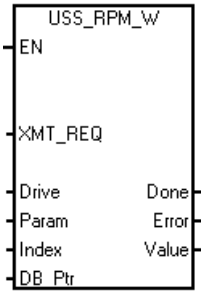
仅在 STL 中显示：

```
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 572)。

9.5.2.4 USS_RPM_x 指令

表格 9- 31 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 9- 32 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。

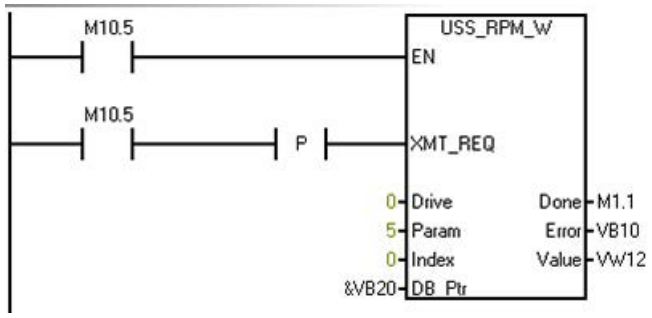
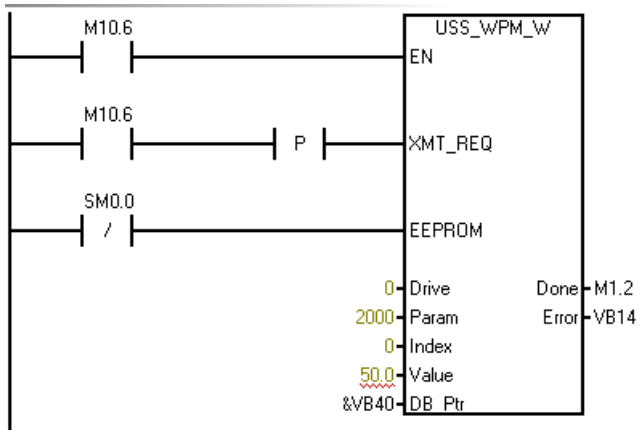
表格 9- 33　USS_RPM_x 参数说明

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 588)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

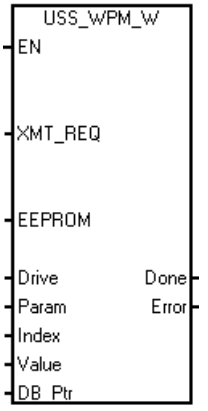
表格 9- 34　USS_RPM_x 和 USS_WPM_x 示例程序

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 572)。

9.5.2.5 USS_WPM_x 指令

表格 9- 35 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 9- 36 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 9- 37　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 588)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。

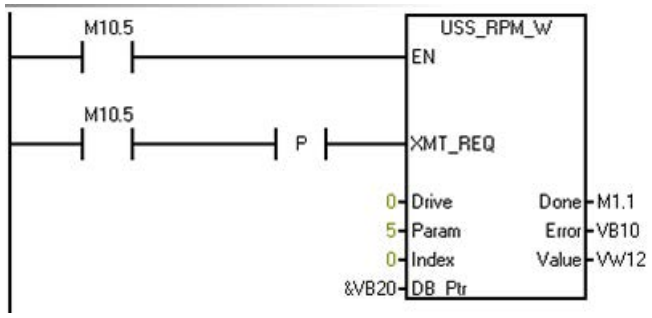
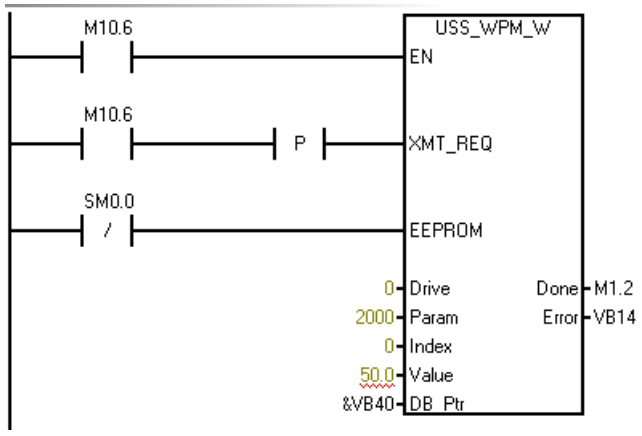
EEPROM

“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 9- 38　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 572)。

9.5.2.6 USS 协议执行错误代码

表格 9- 39 USS 协议执行错误代码

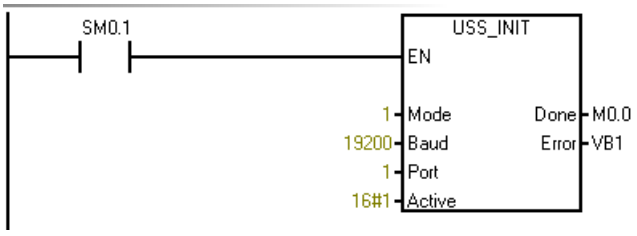
错误代码	说明
0	无错误
1	变频器无响应。
2	检测到来自变频器的响应存在检验和错误。
3	检测到来自变频器的响应存在奇偶校验错误。
4	用户程序的干扰导致错误。
5	尝试非法命令。
6	提供的变频器地址非法。
7	通信端口没有设置为用于 USS 协议通信。
8	通信端口正在忙于处理指令。
9	变频器速度输入超出范围。
10	变频器响应长度不正确。
11	变频器响应的第一个字符不正确。
12	变频器响应中的长度字符不受 USS 指令支持。
13	响应了错误的变频器。
14	提供的 DB_Ptr 地址不正确。
15	提供的参数编号不正确。
16	选择的协议无效。
17	USS 激活；不允许更改。
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。

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 572)。

9.5.2.7 USS 协议示例程序

表格 9- 40 USS 示例程序

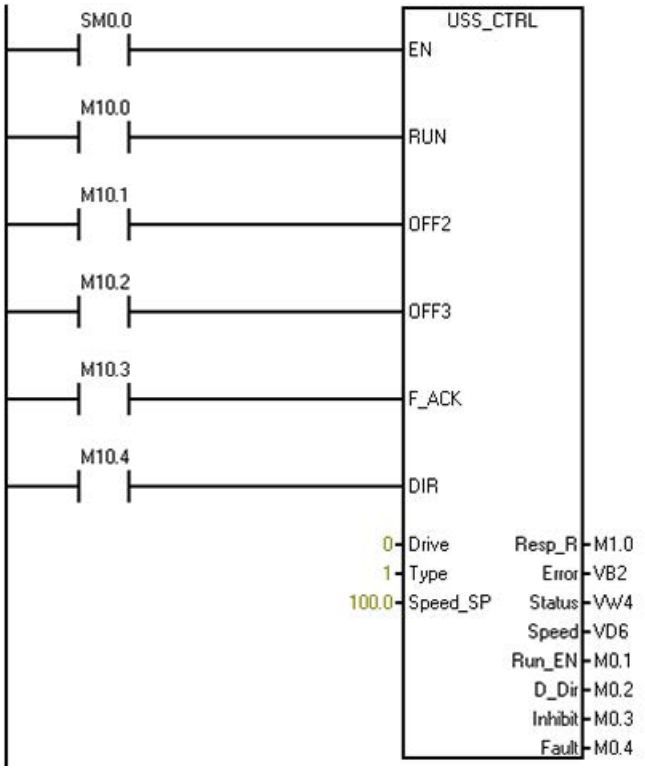
Network 1



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

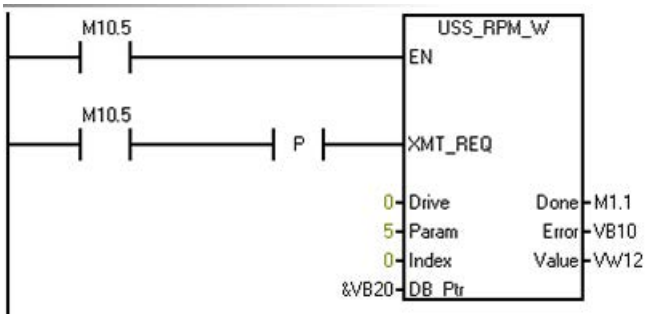


程序段 2：

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：

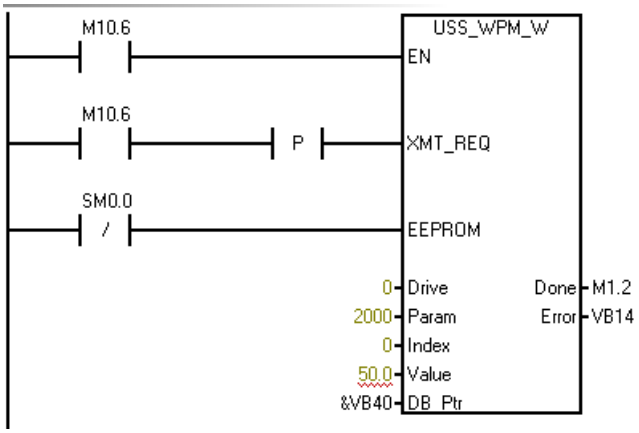
从变频器 0 读取字参数。

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 572)。

9.6 创建用户定义的指令库

STEP 7-Micro/WIN SMART 允许您创建自定义指令库或使用其他人创建的库。

创建库

要创建用户定义的指令库，需要创建标准的 STEP 7-Micro/WIN SMART 子例程，并将它们组合起来。通过将这些子例程分组到库，可以隐藏代码，以防止意外更改并对作者的技术（专有技术）提供更高程度的保护。

要创建用户定义的库，请执行以下任务：

1. 编写程序作为标准项目，并将库中包括的功能写入子例程中。
2. 确保子例程或中断例程中的所有 V 存储单元均已分配符号名称。要最大程度地减少库需要的 V 存储器的数量，请使用连续的 V 存储单元。
3. 将子例程重命名为希望在指令库中显示的名称。
4. 在“文件”(File) 菜单功能区的“库”(Libraries) 区域中，单击“创建”(Create) 按钮 

，然后通过您选择的子例程编译和创建新库。如果该子例程引用中断，STEP 7-Micro/WIN SMART 还包括库中的中断例程。

创建的库可用于新项目或现有项目，但不能用于创建这些库的项目。

更多信息

有关库编程提示和用户定义的库示例，请参见 STEP 7-Micro/WIN SMART 在线帮助库主题。

说明

使用自定义库

您必须负责测试您添加到项目中的任何库。西门子对于自定义库不承担任何责任。

调试和故障排除

10

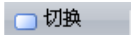
STEP 7-Micro/WIN SMART 提供软件工具来帮助您调试和测试程序。
这些功能包括：查看 CPU 正在执行的程序的状态、为 CPU
指定运行程序的扫描次数以及强制值。

请使用硬件故障排除指南 (页 607)作为排除硬件故障时确定原因和可能解决方案的指导。

10.1 调试程序

10.1.1 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：

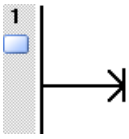


切换书签：
单击该按钮可在当前光标位置指定的程序段处设置或删除书签。

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。



10.1.2 交叉引用表

说明

您必须编译程序才能查看交叉引用表。

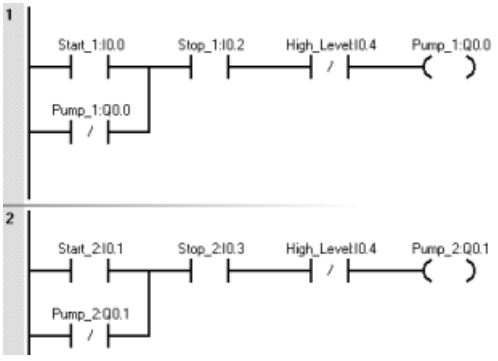
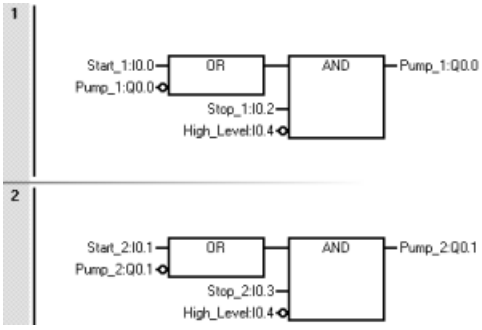
若要了解程序中是否已经使用以及在何处使用某一符号名称或存储器分配，使用交叉引用表。交叉引用表标识在程序中使用的所有操作数，并标识
POU、程序段或行位置以及每次使用操作数时的指令上下文。
在交叉引用表中双击某一元素可显示 POU 的对应部分。

“元素”(Element) 指程序中使用的操作数。 可使用切换按钮 
在符号寻址和绝对寻址之间切换，以更改所有操作数的表示。

- **“块”(Block)** 指使用操作数的 POU。
- **“位置”(Location)** 指使用操作数的行或程序段。
- **“上下文”(Context)** 指使用操作数的程序指令。

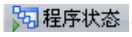
示例

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
LAD		 元素 | 块 | 位置 | 上下文 1 Start_1:I0.0 | MAIN (OB1) | 程序段 1 | -||- 2 Start_2:I0.1 | MAIN (OB1) | 程序段 2 | -||- 3 Stop_1:I0.2 | MAIN (OB1) | 程序段 1 | -||- 4 Stop_2:I0.3 | MAIN (OB1) | 程序段 2 | -||- 5 High_Level:I0.4 | MAIN (OB1) | 程序段 1 | -|/|- 6 High_Level:I0.4 | MAIN (OB1) | 程序段 2 | -|/|- 7 Pump_1:Q0.0 | MAIN (OB1) | 程序段 1 | -( ) 8 Pump_1:Q0.0 | MAIN (OB1) | 程序段 1 | -|/|- 9 Pump_2:Q0.1 | MAIN (OB1) | 程序段 2 | -( ) 10 Pump_2:Q0.1 | MAIN (OB1) | 程序段 2 | -|/|-
FBD		 元素 | 块 | 位置 | 上下文 1 Start_1:I0.0 | MAIN (OB1) | 程序段 1 | OR 2 Start_2:I0.1 | MAIN (OB1) | 程序段 2 | OR 3 Stop_1:I0.2 | MAIN (OB1) | 程序段 1 | AND 4 Stop_2:I0.3 | MAIN (OB1) | 程序段 2 | AND 5 High_Level:I0.4 | MAIN (OB1) | 程序段 1 | AND 6 High_Level:I0.4 | MAIN (OB1) | 程序段 2 | AND 7 Pump_1:Q0.0 | MAIN (OB1) | 程序段 1 | AND 8 Pump_1:Q0.0 | MAIN (OB1) | 程序段 1 | OR 9 Pump_2:Q0.1 | MAIN (OB1) | 程序段 2 | AND 10 Pump_2:Q0.1 | MAIN (OB1) | 程序段 2 | OR
STL		 元素 | 块 | 位置 | 上下文 1 Start_1:I0.0 | MAIN (OB1) | 程序段 1, 行 1 | LD 2 Start_2:I0.1 | MAIN (OB1) | 程序段 2, 行 1 | LD 3 Stop_1:I0.2 | MAIN (OB1) | 程序段 1, 行 3 | A 4 Stop_2:I0.3 | MAIN (OB1) | 程序段 2, 行 3 | A 5 High_Level:I0.4 | MAIN (OB1) | 程序段 1, 行 4 | AN 6 High_Level:I0.4 | MAIN (OB1) | 程序段 2, 行 4 | AN 7 Pump_1:Q0.0 | MAIN (OB1) | 程序段 1, 行 2 | ON 8 Pump_1:Q0.0 | MAIN (OB1) | 程序段 1, 行 5 | = 9 Pump_2:Q0.1 | MAIN (OB1) | 程序段 2, 行 2 | ON 10 Pump_2:Q0.1 | MAIN (OB1) | 程序段 2, 行 5 | =

10.2 显示程序状态

10.2.1 显示程序编辑器中的状态

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  。

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  ，暂停和恢复程序状态采集。

状态图表 (页 600)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 17 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。

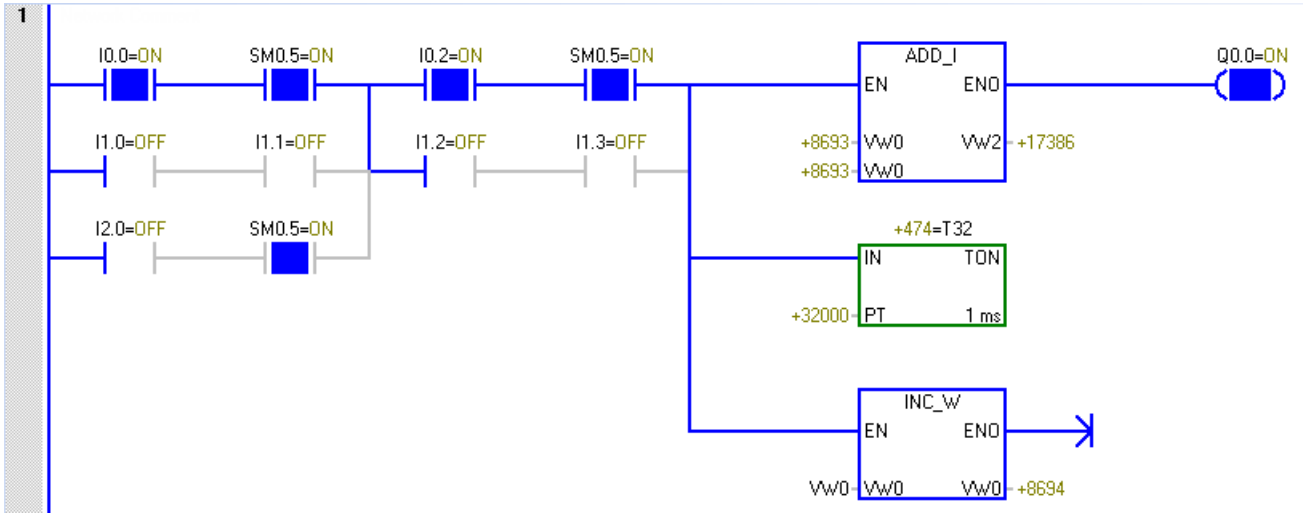
在“工具”(Tools) 菜单功能区的“选项”(Options) 设置中，通过“程序编辑器”(Program Editor) 选项的“STL 状态”(STL Status) 选项卡 (页 599)可以选择或取消选择这些值类别。
如果取消选择某一项，它就不会出现在状态显示中。

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

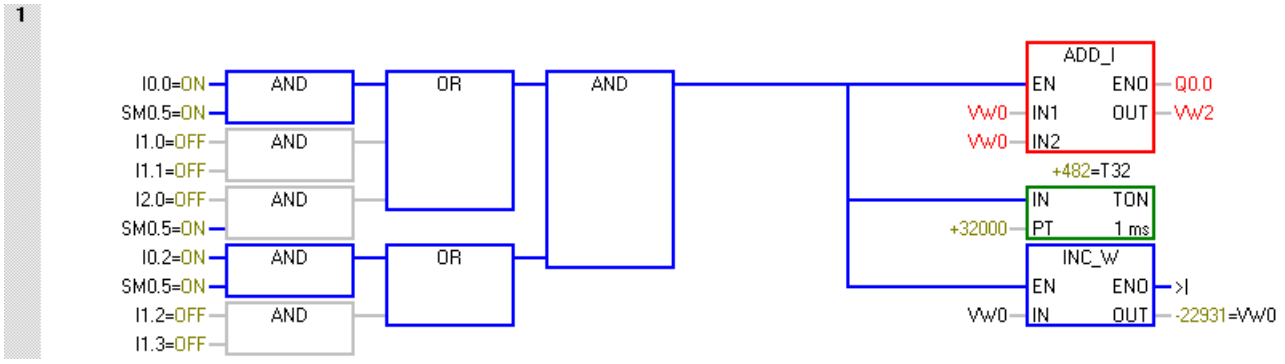
LAD 程序编辑器中的程序状态示例

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。



10.2.2 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。 允许使用的最大数目为 17。
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 596)

10.3 使用状态图以监视程序

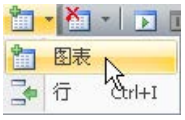
在状态图表中，可以输入地址或已定义的符号名称，通过显示当前值来监视或修改程序输入、输出或变量的状态。 通过状态图表还可强制或更改过程变量的值。
可以创建多个状态图表，以查看程序不同部分中的元素。

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) 和“程序状态”(Program Status) 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，然后选择“图表”(Chart)。



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) 字段中为每个需要的值输入地址（或符号名）。
   符号名必须是已在符号表中定义的名称。
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：
   - 单击状态图表工具栏上的“插入”(Insert) 按钮，然后选择“行”(Row)。

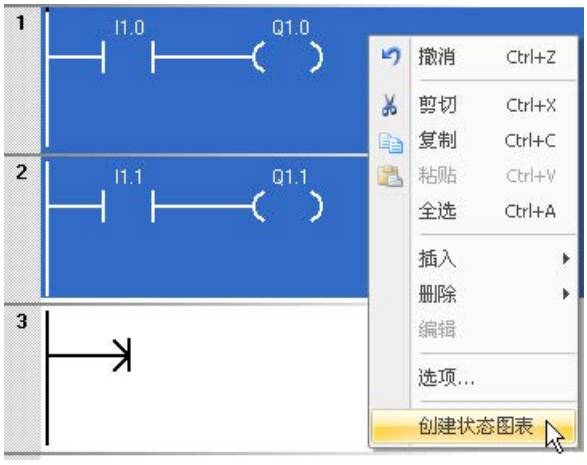
     

   - 在“编辑”(Edit) 菜单功能区的“插入”(Insert) 区域中，单击“行”(Row) 按钮。
   - 右键单击状态图表中的单元格以弹出上下文菜单，然后选择菜单命令**“插入 > 行”(Insert > Row)**。

     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)**
只能添加前 150 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。

此外，还可以从 Microsoft Excel 电子表格复制和粘贴数据。

说明

一个项目最多可存储 32 个状态图表。

将符号从符号表复制到状态图表

您可以从符号表复制地址或符号名称，然后将其粘贴到状态图表，更快地构建图表。

10.4 强制特定值

CPU 允许您强制任意或全部 I/O 点（I 和 Q 位）。 此外，您还可以强制最多 16 个存储器值（V 或 M）或者模拟量 I/O 值（AI 或 AQ）。 V 存储器或 M 存储器值可以按字节、字或双字来强制。
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。

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)。

10.5 在 STOP 模式下写入和强制输出

要在 STOP 模式下启用“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 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 S7-200 SMART PLC 已连接到设备，这些更改内容可能传送到该设备。
这将导致设备内出现异常，进而导致严重人身伤害，甚至死亡和/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。

每次打开 STEP 7-Micro/WIN SMART 时，默认不选择“STOP 模式下强制”(Force in Stop) 菜单选项，防止用户在 PLC 处于 STOP 模式时写入或强制输出。
选择该菜单选项会为当前项目的当前编辑会话启用写入和强制功能。
打开不同的项目时，“STOP 模式下强制”(Force in Stop) 返回默认状态，防止用户在 PLC 处于 STOP 模式时写入和强制输出地址。

10.6 如何执行有限次数的扫描

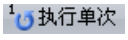
您可以指定 PLC 对程序执行有限次数扫描（从 1 次扫描到 65,535 次扫描）。 通过选择 PLC 运行的扫描次数，您可以在程序改变过程变量时对其进行监控。

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 43)。

执行单次扫描

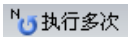
要执行单次扫描，在“调试”(Debug) 菜单功能区的“扫描”(Scan) 区域单击“执行单次扫描”(Execute Single) 按钮。



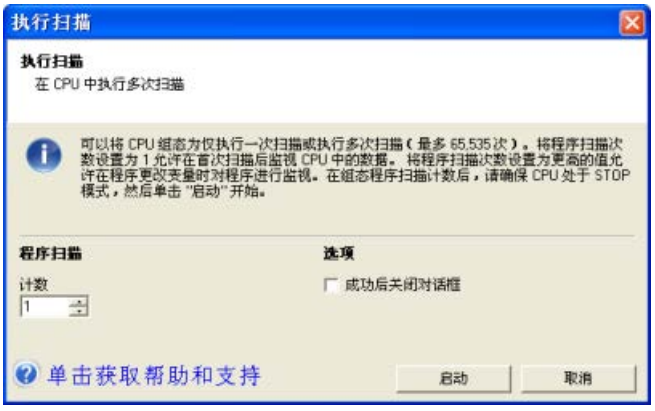
执行多次扫描

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 43)。

另请参见

调试和监视功能概述 (页 593)

如何在编辑器窗口中显示状态 (页 596)

如何在状态图表中显示状态 (页 600)

如何下载程序 (页 88)

时间戳不匹配错误 (页 843)（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 594)（确保程序编辑不引起重复赋值）

强制值 (页 603)

在 STOP 模式下强制输出 (页 604)

10.7 硬件故障排除指南

表格 10- 1 S7-200 SMART 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。请参见第 3 章中的接线指南。
	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 章的接线准则。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
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，请参见第 3 章中的接线指南。
	电压不正确	
与高能量设备相关的间歇操作	接地不正确	请参见第 3 章中的接线指南。
	在控制柜内布线	控制面板良好接地以及高压与低压不并行引线是非常重要的。 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。

问题	可能的原因	可能的解决方案
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。	• 请参见第 3 章中的接线指南和第 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)		有关网络通信的信息，请参见第 8 章。
错误处理		有关错误代码的信息，请参见附录 C。

PID 回路和整定

11

CPU 现已支持 PID 自整定功能，STEP 7-Micro/WIN SMART 中也添加了 PID 整定控制面板。这两项功能相结合，大大增强了 PID 的功能，并且使这一功能的使用变得更加简便。

可通过操作面板或 PID 调节控制面板在用户程序中触发自整定。PID 自整定器会计算建议（接近最佳）的增益值、积分时间（复位）和微分时间（速率）整定值。可以为回路选择快速响应、中速响应、慢速响应或极慢速响应等整定类型。

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。

PID 自整定器的目的在于确定一组整定参数，从而可为回路最优数值提供合理近似。使用这些推荐的整定值可以使您进行极佳的整定调节，真正优化您的控制过程。 CPU 中使用的自整定算法基于 K. J. Åström 和 T. Hägglund 在 1984 年提出的继电器反馈技术。经过二十年的发展，继电器反馈已被广泛用于工业控制的各个领域。

继电器反馈的概念是指在一个稳定的控制过程中产生一个微小但持续的振荡。过程变量中的振荡周期和振幅变化将最终决定控制过程的频率和增益。然后，利用这些最终增益和频率值，PID 自整定器会向您推荐增益、复位和速率整定值。

推荐的值取决于您为控制过程选择的回路响应速度。
您可以选取快速、中速、慢速或极慢速响应。
快速响应可能产生过调，并符合欠阻尼整定条件，具体取决于控制过程。
中速响应可能频临过调，并符合临界阻尼整定条件。
慢速响应不会导致过调，符合强衰减整定条件。
极慢速响应不会导致过调，符合强过阻尼整定条件。

除推荐整定值之外，自整定器还能够自动确定滞后值和 PV 峰值偏差。可使用这些参数减少当限制由 PID 自整定器设置的持续振荡的振幅时过程噪声所产生的影响。

PID 自整定器可以为直接作用和反作用 P、PI、PD 和 PID 回路确定建议整定值。

11.1 PID 回路定义表

通过您在 PID 指令框中针对表 (TBL) 输入的起始地址，为回路表分配八十 (80) 个字节。S7-200 SMART CPU 的 PID 指令引用包含回路参数的此回路表。

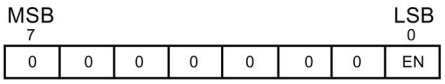
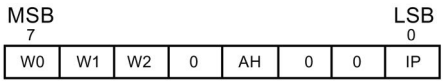
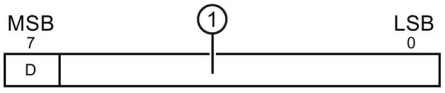
如果使用 PID 整定控制面板，则可通过控制面板处理所有与 PID 回路表的交互。如果需要通过操作员面板提供自整定功能，您的程序必须提供操作员和 PID 回路表之间的交互，这样才能启动和监视自整定过程，然后应用推荐的整定值。

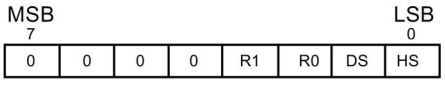
表格 11- 1　回路表

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'（PID 扩展表，版本 A）： ASCII 常数
40	AT 控制 (ACNTL)	BYTE	输入	参见下表
41	AT 状态 (ASTAT)	BYTE	输出	参见下表
42	AT 结果 (ARES)	BYTE	输入/输出	参见下表
43	AT 配置 (ACNFG)	BYTE	输入	参见下表
44	偏差 (DEV)	REAL	输入	最大 PV 振荡幅度的标准化值（范围： 0.025 到 0.25）。
48	滞后 (HYS)	REAL	输入	用于确定过零的 PV 滞后标准化值（范围：0.005 到 0.1）。 如果 DEV 与 HYS 的比值小于 4，自整定期间会发出警告。

偏移	字段	格式	类型	说明
52	初始输出阶跃 (STEP)	REAL	输入	输出值中阶跃变化的标准化大小，用于使 PV 产生振荡（范围：0.05 到 0.4）。
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	实际阶跃大小 (ASTEP)	REAL	输出	自整定过程确定的标准化输出阶跃大小值。
76	实际滞后 (AHYS)	REAL	输出	自整定过程确定的标准化 PV 滞后值。

表格 11- 2　控制和状态字段的具体描述

字段	说明	
AT 控制 (ACNTL) 输入 - 字节		
	EN	设为 1 可启动自整定；设为 0 可中止自整定
AT 状态 (ASTAT) 输出 - 字节		
	W0	警告： 偏差设置没有超过滞后设置的四倍。
	W1	警告： 过程偏差不一致可能导致对输出阶跃值的调整不正确。
	W2	警告： 实际平均偏差没有超过滞后设置的四倍。
	AH	正在进行自动滞后计算： 0 - 没有进行 1 - 正在进行
	IP	正在进行自整定： 0 - 没有进行 1 - 正在进行
	每次自整定序列启动时，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。	
AT 结果 (ARES) 输入/输出 - 字节	 ① 结果代码	
	D	“完成”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

字段	说明	
	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 至 7F - 保留
AT 配置 (ACNFG) 输入 - 字节		
	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	偏差设置： 0 - 使用回路表中的偏差值 1 - 自动确定偏差值
	HS	滞后设置： 0 - 使用回路表中的滞后值 1 - 自动确定滞后值

说明

具有 PID 向导组态的项目不直接使用标准 PID 指令 (页 319)。 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 (页 321)。

11.2 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。

启动自整定操作之前，控制过程必须达到稳定态，也就是说，PV
已经达到设定值（或者对 P 型回路来说，PV
与设定值之间的差值恒定），并且输出不会不规律地变化。

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。

11.3 自滞后和自偏差

滞后参数

滞后参数指定了相对于设定值的偏移（正或负），PV（过程变量）在此偏移范围内时，不会导致继电器控制器改变输出值。 该值用于减小 PV
信号中噪声的影响，从而更精确地计算出过程的固有振动频率。

如果选择自动确定滞后值，PID 自整定器将进入滞后确定序列。
该序列包含一段时间内的过程变量采样值，然后根据采样结果计算出标准偏移。

为了得到具有统计意义的采样数据，至少要有 100 个采样值。 如果回路的采样时间为 200 毫秒，则采集 100 个样本需要 20 秒。 采样时间较长的回路需要更多时间。
即使回路的采样时间小于 200 毫秒，从而采样 100 次的时间不需要 20
秒，滞后确定序列仍然需要至少 20 秒的采样时间。

得到所有采样之后，就可以算出采样集合的标准偏差。 滞后值定义为标准偏差的两倍。
计算出的滞后值将写入回路表的实际滞后字段 (AHYS)。

说明

计算自滞后序列时，不能执行正常的 PID 计算。
因此，在启动自整定序列之前，控制过程应处于稳定状态。
这样可以得到更好的滞后值结果，同时也可以保证自滞后序列期间控制过程不会失控。

偏差参数

偏差参数是指希望得到的 PV 对于设定值的峰峰值幅度。
如果选择自动确定该值，它将是滞后值的 4.5 倍。
在自整定过程中，会适当地调节输出，使控制过程中的振荡幅度在这一范围内。

11.4 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。

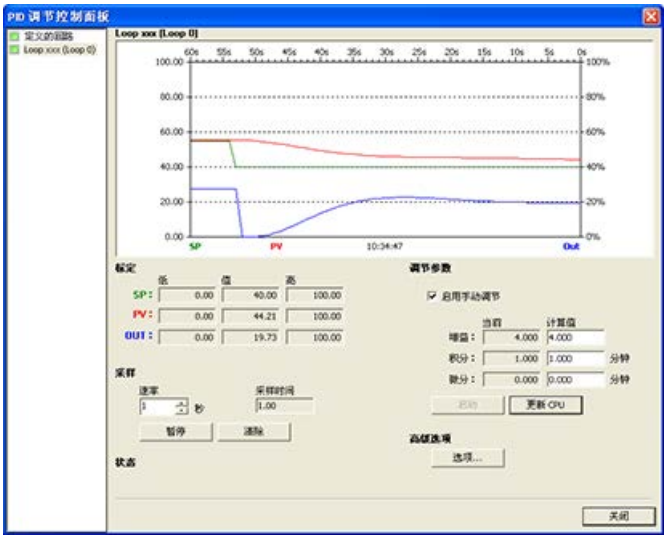
输出值的这一变化会导致过程变量值产生相应的变化。 当输出的变化使 PV 远离设定值以至于超出滞后区范围时，自整定器就会检测到过零事件。
每次发生过零事件时，自整定器将反方向改变输出。

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。

11.5 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。

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 11- 3 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

结果代码（在 ARES 中）		条件
06	因数字错误而中止	遇到非法浮点数或除数为零
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

11.6 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。

11.7 PID 整定控制面板

STEP 7-Micro/WIN SMART 中包含 PID 整定控制面板，允许您以图形方式监视 PID 回路。
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。

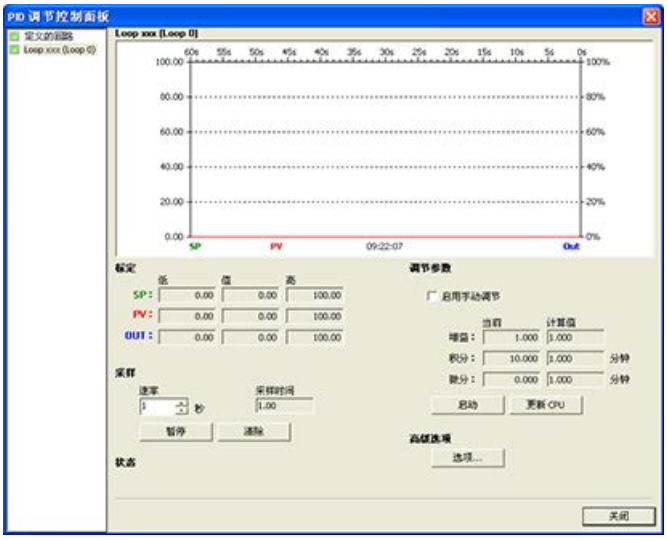
要使用控制面板，必须与 CPU 通信，并且该 CPU 中必须存在一个用于 PID 回路的向导生成的组态。要使控制面板显示对 PID 回路的操作，CPU 必须处于 RUN 模式。

采用以下任意一种方式打开 PID 控制面板：

- 单价“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。

- 在项目树中打开“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 将打开 PID 控制面板：



PID 控制面板包含以下字段：

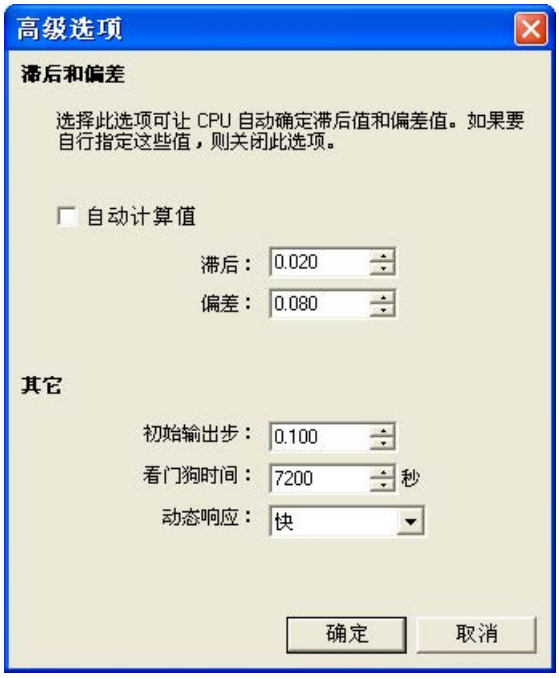
- 当前值： 显示
  SP（设定值）、PV（过程变量）、OUT（输出）、“采样时间”、“增益”、积分时间和微分时间的值。 SP、PV 和 OUT
  分别以绿色、红色和蓝色显示；使用相同颜色的图例来标明 PV、SP 和 OUT 的值。
- 图形显示区： 图形显示区中用不同的颜色显示了 PV、SP
  和输出值相对于时间的函数。 PV 和 SP
  共用图形左侧纵轴，输出使用图形右侧的纵轴。
- 整定参数： 画面左下角是“整定参数”（分钟）区域。
  在此处显示“增益”、“积分时间”和“微分时间”的值。 在“计算值”(Calculated)
  列中单击，可对这些值的三个源中任意一个源进行修改。
- “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”(Start) 按钮启动自整定序列。 一旦自整定序列启动，“启动”(Start)
  按钮将变为“停止”(Stop) 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 480 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) 按钮对自整定过程的参数进行进一步组态。 （请参见下图。）

  在高级画面中，您可以选中复选框，让自整定器自动计算滞后值和偏差值（默认设置），为了最大程度地减小自整定过程中对控制过程的干扰，您也可以自己输入这些值。



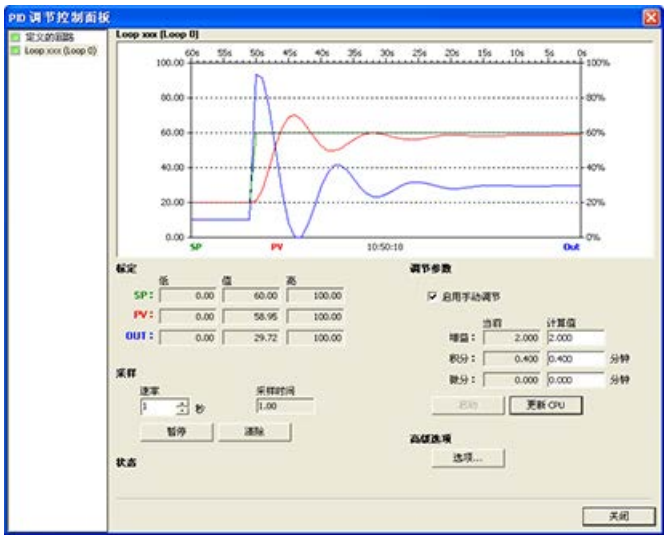
  在“动态响应”(Dynamic Response) 字段中，可使用下拉按钮选择希望在控制过程中使用的回路响应类型（“快速”(Fast)、“中速”(Medium)、“慢速”(Slow) 或“极慢速”(Very Slow)）。
  快速响应可能产生过调，并符合欠阻尼整定条件，具体取决于控制过程。
  中速响应可能频临过调，并符合临界阻尼整定条件。
  慢速响应不会导致过调，符合强衰减整定条件。
  极慢速响应不会导致过调，符合强过阻尼整定条件。

  一旦您完成了选择，可以单击“确定”(OK) 按钮返回 PID 整定控制面板的主画面。

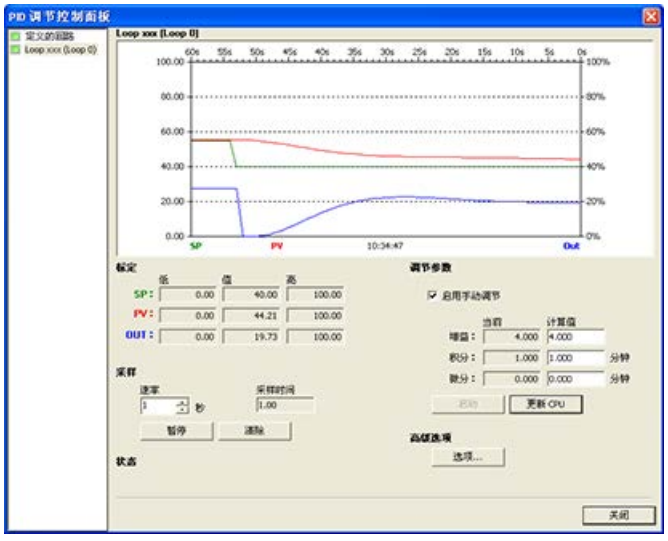
回路监视

完成自整定序列并将建议的整定参数传送到 CPU
之后，可以使用控制面板监视回路对设定值阶跃变化的响应。

下图显示了原始整定参数的设定值变化时回路的响应情况（运行自整定之前）。
请注意在使用原始整定参数时控制过程的过调和长时间振荡的现象。



如果选择快速响应，应用自整定过程确定的值后，回路设定值也会发生相同的变化。
请注意此过程没有过调现象，但仍有轻微的振荡。



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 (页 321)。

开环运动控制

12

S7-200 SMART CPU 提供了三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。（注意：无 PTO 向导可用。请用运动控制向导代替。）
- 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：内置于 CPU 中，用于速度和位置控制

CPU 提供了三个数字输出（Q0.0、Q0.1、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。

若组态 PTO 输出，无论是步进电机还是伺服电机，CPU 均会生成 50% 负载循环的脉冲串用于转速和位置的开环控制。内置 PTO 功能仅提供脉冲串输出。方向和限值控制必须通过应用程序使用 PLC 中集成的或由扩展模块提供的 I/O 来提供。

若组态 PWM 输出，CPU 将固定输出的周期时间，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。

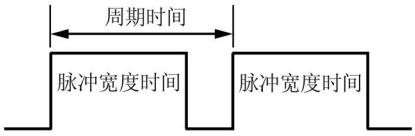
运动轴提供了带有集成方向控制和禁用输出的单脉冲串输出。运动轴还包括可编程输入，允许将 CPU 组态为包括自动参考点搜索在内的多种操作模式。运动轴为步进电机或伺服电机的速度和位置开环控制提供了统一的解决方案。

为简化在应用中使用运动控制功能，STEP 7-Micro/WIN SMART 提供了用以组态运动轴的运动控制向导以及用以组态 PWM 的 PWM 向导。这些向导会生成运动指令，可用以动态控制应用的速度和运动。对于运动轴，STEP 7-Micro/WIN SMART 还提供了控制面板，您可以通过该控制面板控制、监视和测试运动操作。

12.1 使用 PWM 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。

12.1.1 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

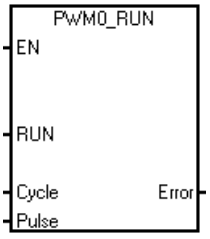
PWM 通道硬编码为具体输出：

- PWM0 已分配到 Q0.0。
- PWM1 已分配到 Q0.1。
- PWM2 已分配到 Q0.3。

12.1.2 PWMx_RUN 子例程

为简化在应用中使用脉宽调制 (PWM) 控制功能，STEP 7-Micro/WIN SMART 提供了 PWM 向导 (页 626)用以组态板载 PWM 生成器和控制 PWM 输出的负载周期。

PWMx_RUN 子程序用于在程序控制下执行 PWM。

LAD/FBD	STL	说明
	**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 12- 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	Byte	IB、QB、VB、MBV、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 12- 2　PWMx_RUN 指令错误代码

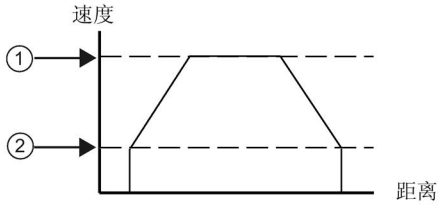
错误代码	说明
0	无错误，正常完成
131	脉冲生成器已被另一个 PWM 或运动轴占用，或者时基更改无效

12.2 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。

12.2.1 最大速度和启动/停止速度



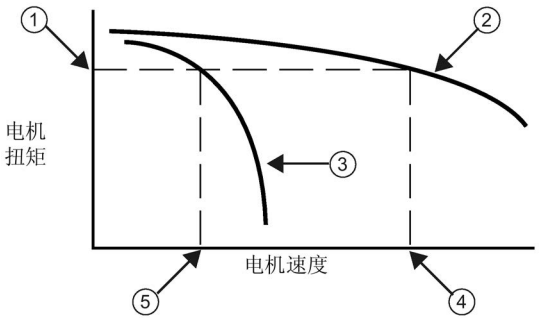
运动控制向导会提示您应用的最大速度 (MAX_SPEED) 和启动/停止速度 (SS_SPEED)。

① MAX_SPEED

② SS_SPEED

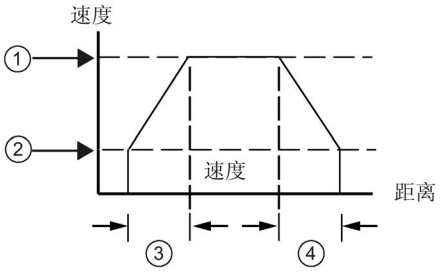
- MAX_SPEED： 在电机力扭矩能力范围内，输入应用中最佳操作速度的数值。
  驱动负载所需的扭矩由摩擦力、惯性以及加速/减速时间决定。
- 运动向导基于指定的 MAX_SPEED 计算和显示可由运动轴控制的最低速度。
- SS_SPEED： 在电机能力范围内输入一个数值，以便以较低的速度驱动负载。 如果 SS_SPEED 数值过低，电机和负载在运动的开始和结束时可能会摇摆或颤动。 如果 SS_SPEED
  数值过高，电机会在启动时丢失脉冲，并且负载在试图停止时会使电机超速。

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)

12.2.2 输入加速和减速时间



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。

① MAX_SPEED

② SS_SPEED

③ ACCEL_TIME

④ DECEL_TIME

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

12.2.3 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。

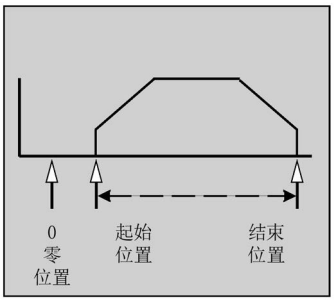
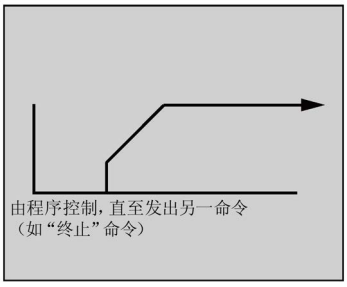
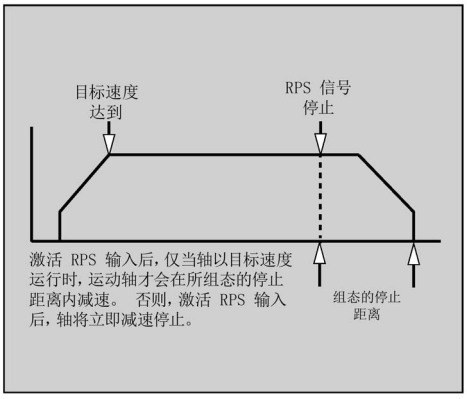
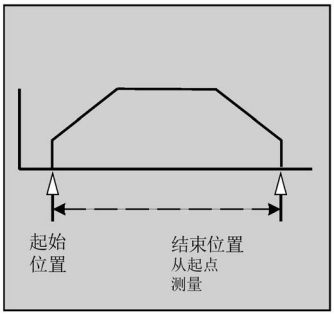
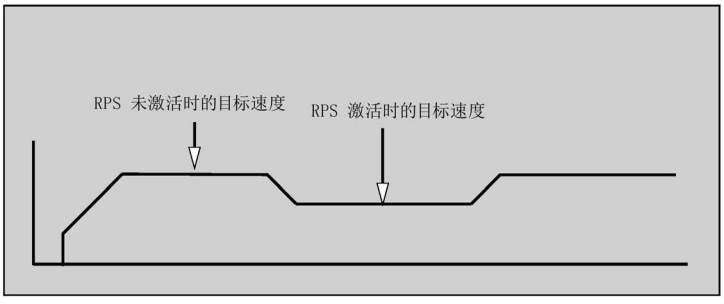
运动轴最多支持 32 个曲线。

定义运动曲线

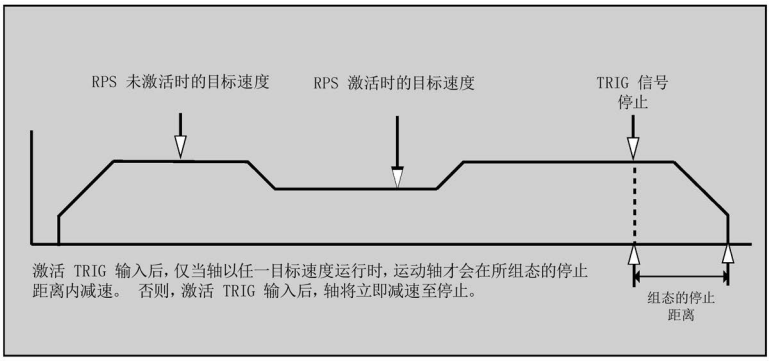
运动向导会指导您完成运动曲线定义，您可以在其中为应用定义每一个运动曲线。
对于每一个曲线，您可以选择操作模式并为曲线的每一步定义指标。
运动向导允许您为每个曲线定义符号名称，您只需在定义曲线时输入符号名称即可。

选择曲线的操作模式

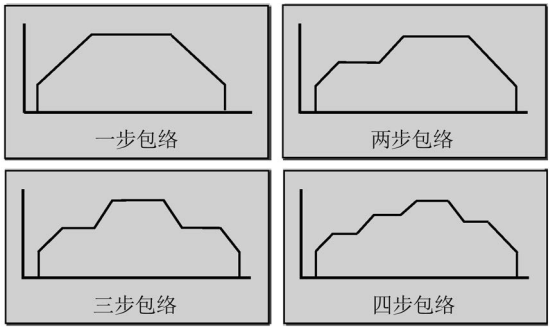
根据所需操作模式组态曲线。
运动轴支持绝对位置、相对位置、单速连续转动以及双速连续转动。
下图显示了不同的操作模式。

<table>
<tr><th colspan="2">运动轴的模式选择</th></tr>
<tr><th>绝对位置</th><th>单速连续旋转</th></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><th>单速连续旋转，并带有触发停止功能</th><th>相对位置</th></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><th colspan="2">双速连续旋转</th></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><th colspan="2">双速连续旋转，并带有触发停止功能</th></tr>
</table>

运动轴的模式选择



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。

12.3 运动控制的特点

运动控制可在最多三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
- 提供可组态的反冲补偿
- 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 可以创建运动轴所使用的全部组态和曲线信息。
这些信息和您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 (页 724)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 12- 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。 运动向导允许您组态 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。

⚠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。
在此期间，可能不会检测到持续时间少于 12.8 ms 的短“0”脉冲事件或对其计数。

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 12- 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。

12.4 编程运动轴

STEP 7-Micro/WIN SMART 提供用于组态和编程运动轴的易用工具。
只需遵循以下步骤即可：

1. 组态运动轴： STEP 7 Micro/WIN SMART
   提供一个运动向导，用于创建组态/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
   提供一个运动控制面板，用于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。

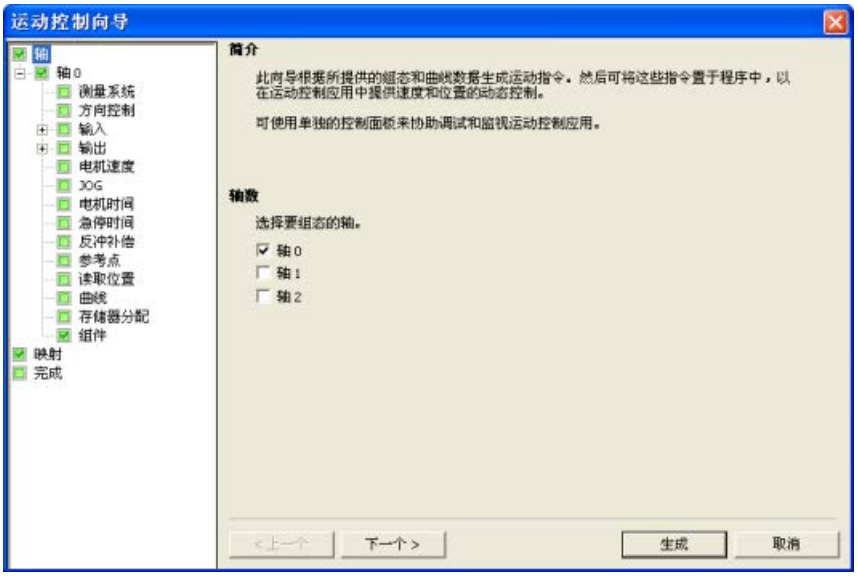
说明

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。

12.5 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。运动向导可引导您逐步完成组态过程，非常便捷。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。



运动向导允许您离线创建组态/曲线表。您可以在不连接 CPU 的情况下创建组态。

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。

如果您在以后改变了测量系统，则必须删除整个组态，包括运动向导生成的全部指令。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 12- 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)。

输入的引脚定义	说明
ZP HSC	CPU 的 HSC0 充当 ZP 输入 (I0.0)。
	CPU 的 HSC1 充当 ZP 输入 (I0.1)。
	CPU 的 HSC2 充当 ZP 输入 (I0.2)。
	CPU 的 HSC3 充当 ZP 输入 (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 为 PWM 和运动轴实施固定输出分配。

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：

轴 0	• 轴 0 的 P0 始终组态为 Q0.0。 • 如果轴的“相”(Phase) 未组态为“单相（1 个输出）”，则轴 0 的 P1 组态为 Q0.2。
轴 1	• 轴 1 的 P0 始终组态为 Q0.1。 • 轴 1 的 P1 根据轴组态在两个可能的位置映射，如下所示： – 如果轴 1 的“相”(Phase) 组态为“单相（1 路输出）”，则不会分配 P1 输出。 – 如果轴 1 的“相”(Phase) 组态为“两相（2 个输出）”(Two-phase (2 output)) 或“AB 正交相（2 个输出）”(AB quadrature phase(2 output))，则 P1 组态为 Q0.3。 – 其他情况下，轴 1 的 P1 始终组态为 Q0.7。
轴 2	• 轴 2 的 P0 始终组态为 Q0.3。 • 如果轴的“相”(Phase) 未组态为“单相（1 个输出）”(1 output)，则轴 2 的 P1 组态为 Q1.0。 • 如果轴 1 的“相”(Phase) 组态为“两相（2 个输出）”(Two-phase (2 output)) 或“AB 正交相（2 个输出）”(AB quadrature phase(2 output))，则轴 2 不可用。

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：

- 轴 0 的 DIS 始终组态为 Q0.4。
- 轴 1 的 DIS 始终组态为 Q0.5。
- 轴 2 的 DIS 始终组态为 Q0.6。

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。在此期间，可能不会检测到持续时间少于 12.8 ms 的短“0”脉冲事件或对其计数。

  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。

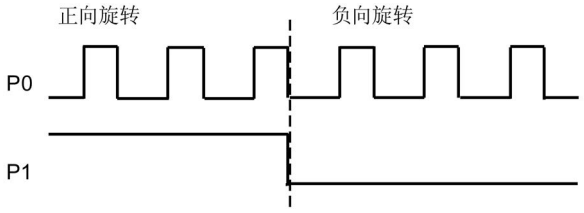
- 在“方向控制”(Directional Control) 节点，可选择下列“相位”(Phasing) 模式：
  - 单相（2 个输出）
  - 双相（2 个输出）
  - AB 正交相（2 个输出）
  - 单相（1 个输出）

  还可以选择输出的“极性”(Polarity)（正或负）。

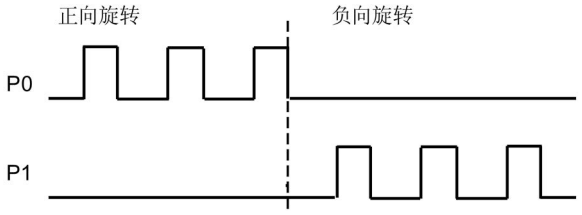
相位

步进/伺服驱动器的“相位”(Phasing) 界面有 4 个选项。这些选项如下：

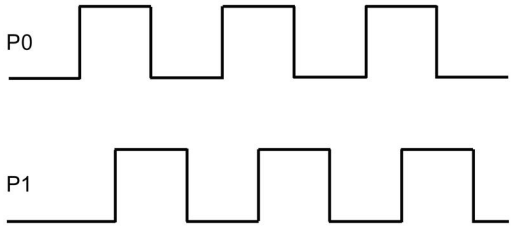
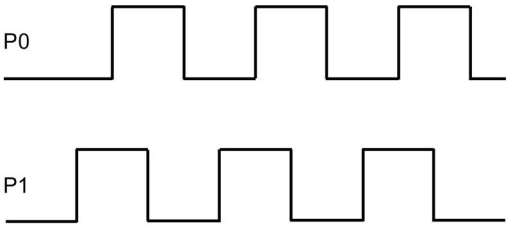
- 单相（2 个输出）：如果选择单相（2 个输出）选项，则一个输出 (P0) 控制脉冲，另一输出 (P1) 控制方向。如果脉动处于正向，则 P1 为高（有效）。如果脉动处于负向，则 P1 为低（无效）。单相（2 个输出）如下图所示（假设极性为正）：



- 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）：

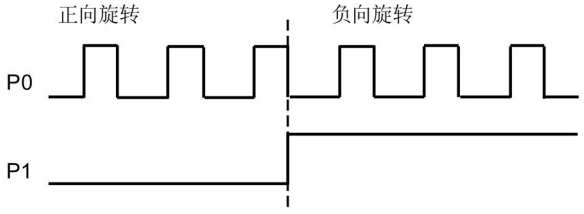
AB 正交相（2 个输出）	
（正极性）：正向旋转	（正极性）：反向旋转
	
P0 超前 P1	P1 超前 P0

- 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。当您选择此模式时，运动向导会限制您进行非法的负运动组态。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）：

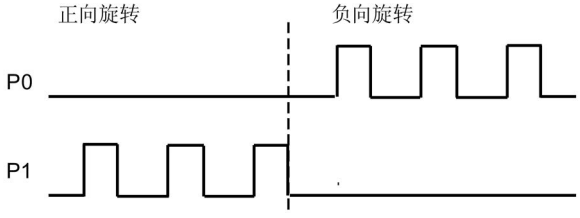
极性

您可以通过“极性”(Polarity) 参数切换正负向。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：

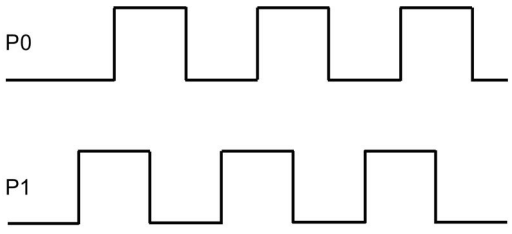
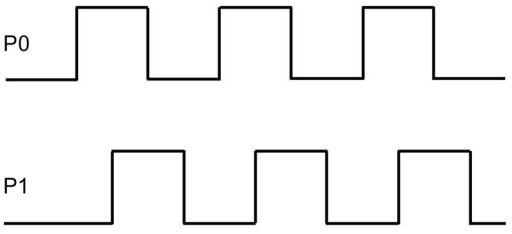
- 单相（2 个输出）：如果脉动处于正向，则 P1 为低（无效）。如果脉动处于负向，则 P1 为高（有效）。如下图所示：



- 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：

<table>
<tr><td colspan="2">AB 正交相（2 个输出）</td></tr>
<tr><td>（负极性）：正向旋转</td><td>（负极性）：反向旋转</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>P1 超前 P0</td><td>P0 超前 P1</td></tr>
</table>

- 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，请参考映射 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。

输入最大的启动和停止速度

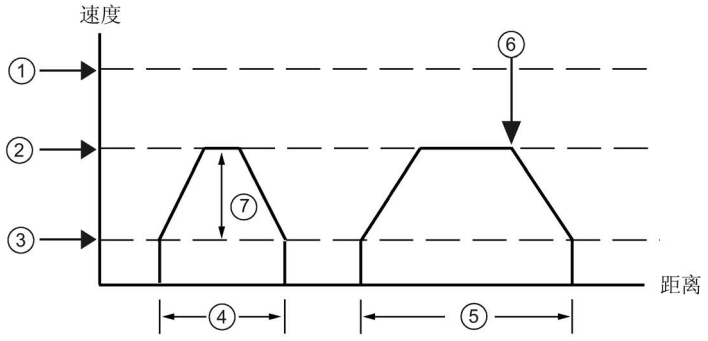
输入应用的最大速度 (MAX_SPEED) 和启动/停止速度 (SS_SPEED)。

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。如果在不到 0.5 秒的时间便终止 Jog 命令，则运动轴会以 JOG_SPEED 定义的速度将刀具移动 JOG_INCREMENT 中指定的数量。如果 0.5 秒过后 Jog 命令仍有效，则运动轴加速到 JOG_SPEED。运动会持续到 Jog 命令被终止。运动轴随后减速至停止。您可以在运动控制面板中启用点动命令，或通过运动指令启动点动命令。下图显示了 JOG 操作。



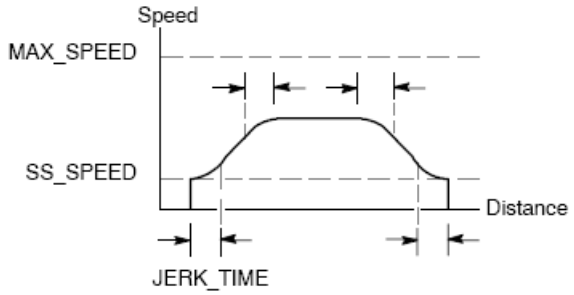
① MAX_SPEED
② JOG_SPEED
③ SS_SPEED
④ JOG_INCREMENT：JOG 命令处于激活状态不超过 0.5 秒。
⑤ JOG 命令处于激活状态超过 0.5 秒。
⑥ JOG 命令终止（开始从 JOG_SPEED 降至 SS_SPEED）。
⑦ 达到的速度可以是 SS_SPEED 至 JOG_SPEED 之间的任何速度，具体取决于 JOG_INCREMENT 的长度。

输入加速时间

在编辑框中输入加速和减速时间。

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。请参见下图：



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。

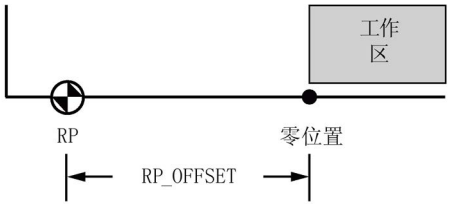
组态反冲补偿

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。

组态参考点和搜索参数

1. 为您的应用选择使用参考点或不使用参考点：
   - 若应用需要从一个绝对位置处开始运动或以绝对位置作为参考，则必须建立一个参考点 (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，则需要定义自动重新定位参考点的方法。。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. 您可以为运动轴组态搜索参考点的顺序。下图显示了一个简化了的默认 RP 搜索顺序图。您可以为 RP 搜索顺序选择以下模式：
   - **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

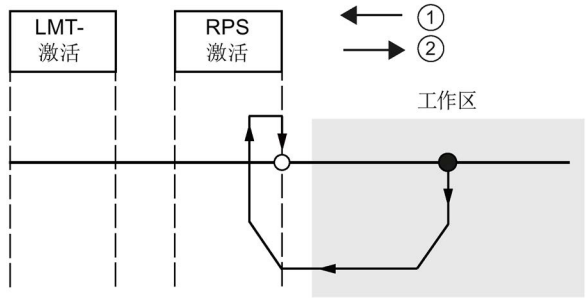


图 12-1 ①:RP 搜索方向
②：RP 接近方向

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. 为了完成需要的运动，您可以定义任意多个曲线和步。

完成组态

1. 当您完成对运动轴的组态后，只需单击“生成”(Generate)。

   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，您可在应用中使用这些指令

2. 要修改任何组态或曲线信息，您可以再次运行运动向导。

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。

12.6 运动向导为运动轴创建的子例程

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 653)	提供轴的初始化和全面控制
AXISx_MAN (页 654)	用于轴的手动模式操作
AXISx_GOTO (页 656)	命令轴转到指定位置
AXISx_RUN (页 658)	命令轴执行已组态的运动曲线
AXISx_RSEEK (页 659)	启动参考点查找操作
AXISx_LDOFF (页 660)	建立一个偏移参考点位置的新零点位置
AXISx_LDPOS (页 662)	将轴位置更改为新值
AXISx_SRATE (页 663)	修改已组态的加速、减速和急停补偿时间
AXISx_DIS (页 664)	控制 DIS 输出
AXISx_CFG (页 665)	根据需要读取组态块并更新轴设置
AXISx_CACHE (页 666)	预先缓冲已组态的运动曲线
AXISx_RDPOS (页 667)	返回当前轴位置
AXISx_ABSPOS (页 668)	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 (页 625)

12.6.1 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。

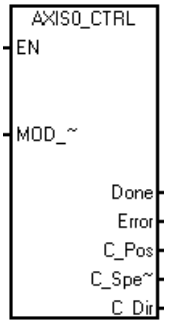
运动控制向导根据所选的度量系统自动组态速度参数（Speed 和 C_Speed）和位置参数（Pos 或 C_Pos）的值。对于脉冲，这些参数为 DINT 值。对于工程单位，这些参数是所选单位类型对应的 REAL 值。例如：如果选择厘米 (cm)，则以厘米为单位将位置参数存储为 REAL 值并以厘米/秒 (cm/sec) 为单位将速度参数存储为 REAL 值。

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。

12.6.2 AXISx_CTRL 子例程

表格 12- 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。 在您的项目中只对每条运动轴使用此子例程一次，并确保程序会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。

表格 12- 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。

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 696)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。如果您针对脉冲组态运动轴的测量系统，C_Speed 是一个 DINT 数值，其中包含脉冲数/每秒。如果您针对工程单位组态测量系统，C_Speed 是一个 REAL 数值，其中包含选择的工程单位数/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

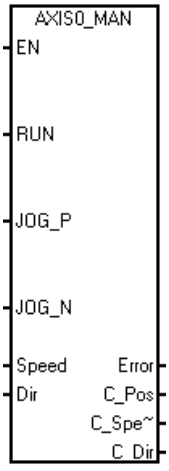
说明

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。

- 如果您使用运动控制向导修改组态，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。
- 如果您使用运动控制面板修改组态，单击“更新组态”(Update Configuration) 按钮，命令运动轴加载新组态/曲线表。
- 如果您使用另一种方法修改组态，您还必须向运动轴发出一个 AXISx_CFG 命令，以加载组态/曲线表。否则，运动轴会继续使用旧组态/曲线表。

12.6.3 AXISx_MAN 子例程

表格 12- 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 12- 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 696)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

12.6.4 AXISx_GOTO 子例程

表格 12- 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 12- 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。

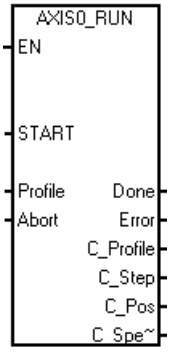
Error 参数 (页 696)包含该子例程的结果。

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)。

12.6.5 AXISx_RUN 子例程

表格 12- 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 12- 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 696)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)。

12.6.6 AXISx_RSEEK 子例程

表格 12- 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。

表格 12- 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。可使用运动控制向导、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 696)包含该子例程的结果。

12.6.7 AXISx_LDOFF 子例程

表格 12- 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 12- 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 696)包含该子例程的结果。

12.6.8 AXISx_LDPOS 子例程

表格 12- 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 12- 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)。

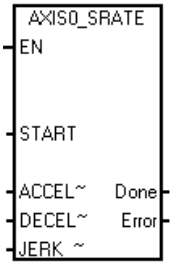
当运动轴完成此子例程时，Done 参数会开启。

Error 参数 (页 696)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)。

12.6.9 AXISx_SRATE 子例程

表格 12- 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 12- 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。

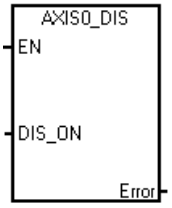
ACCEL_Time、DECEL_Time 和 JERK_Time 参数用于确定新的加速时间、减速时间以及急停时间，单位为毫秒 (ms)。

当运动轴完成此子例程时，Done 参数会开启。

Error 参数 (页 696)包含该子例程的结果。

12.6.10 AXISx_DIS 子例程

表格 12- 22 AXISx_DIS

LAD/FBD	STL	说明
	**CALL AXISx_DIS, DIS_ON, Error**	AXISx_DIS 子例程将运动轴的 DIS 输出打开或关闭。 这允许您将 DIS 输出用于禁用或启用电机控制器。 如果您在运动轴中使用 DIS 输出，可以在每次扫描时调用该子例程，或者仅在您需要更改 DIS 输出值时进行调用。

表格 12- 23 AXISx_DIS 子例程的参数

输入/输出	数据类型	操作数
DIS_ON	BOOL	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD、常数
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。

Error 参数 (页 696)包含该子例程的结果。

12.6.11 AXISx_CFG 子例程

表格 12- 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 12- 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 696)包含该子例程的结果。

12.6.12 AXISx_CACHE 子例程

表格 12- 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 12- 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。

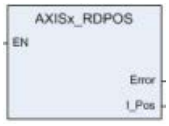
Profile 参数包含运动曲线的编号或符号名称。 “Profile”输入必须介于 0 - 31。否则子例程将返回错误。

当运动轴完成此子例程时，Done 参数会开启。

Error 参数 (页 696)包含该子例程的结果。

12.6.13 AXISx_RDPOS 子例程

表格 12- 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 12- 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 696)包含该子例程的结果。

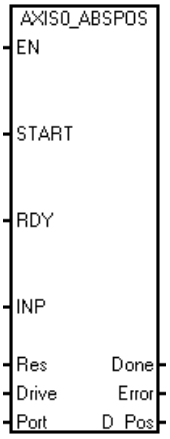
I_Pos 参数包含当前运动轴位置。

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。

12.6.14 AXISx_ABSPOS 子例程

表格 12- 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 12- 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 696)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。

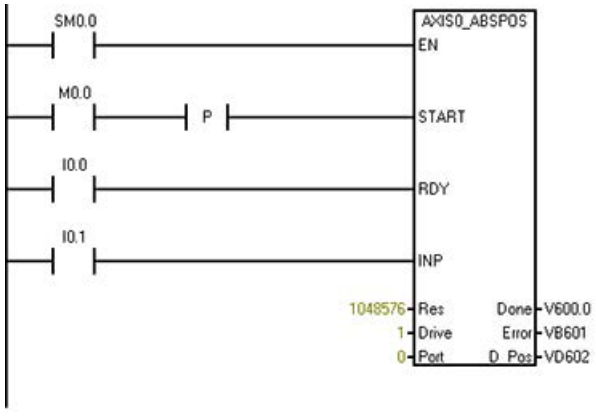
12.7 使用 AXISx_ABSPOS 子程序从 SINAMICS 伺服驱动读取绝对位置

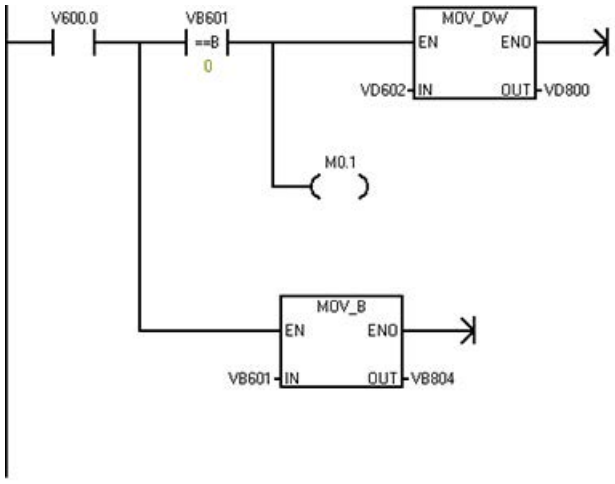
以下各节介绍如何在项目中使用 AXISx_ABSPOS 子程序从 SINAMICS V90 伺服驱动读取绝对位置。

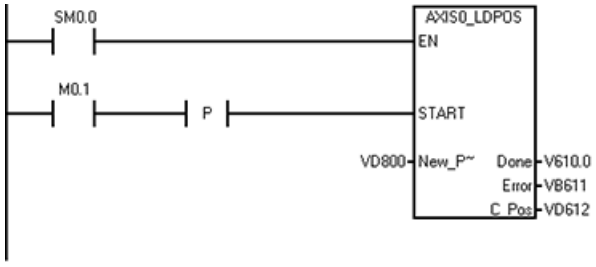
12.7.1 AXISx_ABSPOS 和 AXISx_LDPOS 子程序应用示例

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 12- 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

12.7.2 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，请在 S7-200 SMART CPU（或 S7-200 SMART CM01 信号板）和 V90 伺服驱动的 RS485 端口之间连接一根电缆。

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。

12.7.3 调试

12.7.3.1 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。

12.7.3.2 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）。

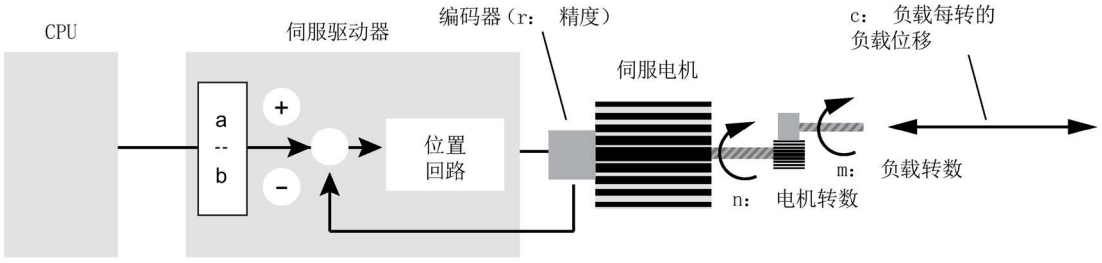
12.7.3.3 设定值脉冲串输入格式

确保 CPU 的运动轴输出相位和极性设置（在 STEP 7-Micro/WIN SMART 运动向导的“方向控制”(Directional Contro) 对话框中设置）与 V90 伺服驱动的设定值脉冲串输入格式设置（参数“p29010”）一致。

12.7.3.4 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 运动向导为 CPU 运动轴组态测量系统时，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)

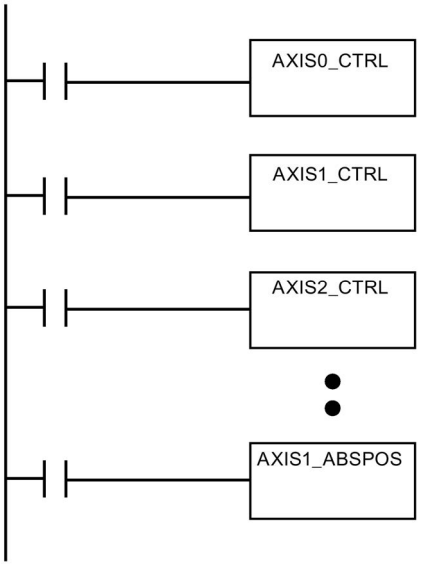
其中，“c”=负载运动（以指定的工程单位表示）/负载转，
“m/n”= 外部齿轮传动比（以负载转数/电机转表示）

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**

12.7.4 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。



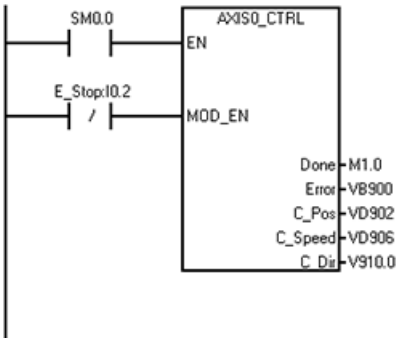
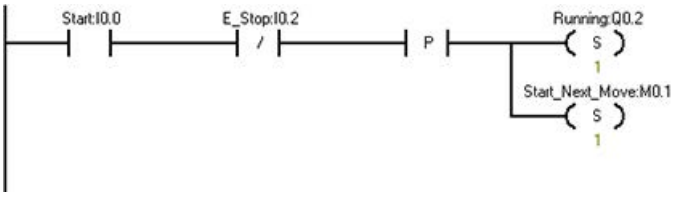
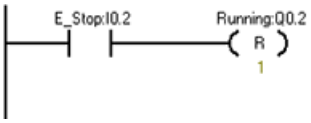
- 如果您根据“相对脉冲”设置而不是“工程单位”设置组态运动轴测量系统，则仍可以使用 AXISx_ABSPOS 子程序从 V90 伺服驱动返回位置信息。不过，需注意子程序“D_pos”参数中返回的位置值将是 DINT 类型的值，并且是伺服电机报告的实际位置值（没有对该值执行工程单位换算）。

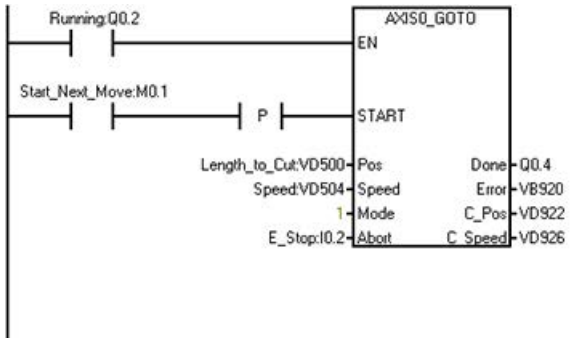
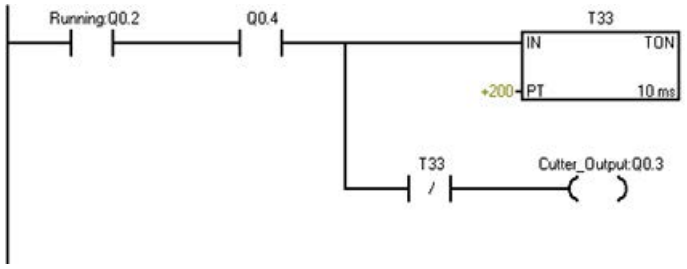
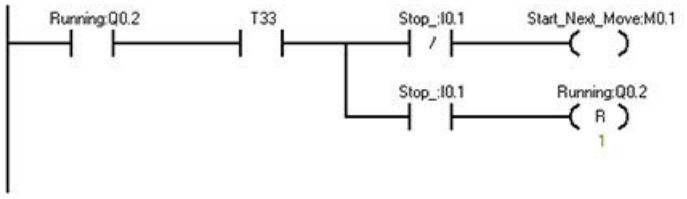
12.8 运动轴示例程序

12.8.1 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 12- 33 示例：运动轴简单相对移动（定长截断应用）

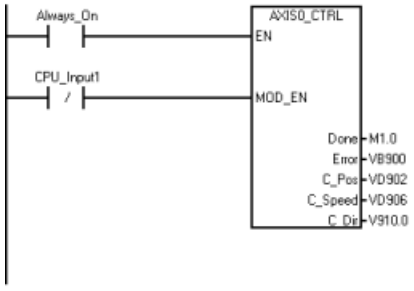
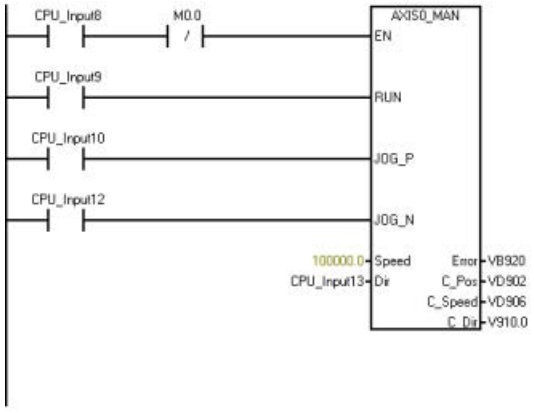
LAD/FBD	说明	STL
程序段 1： 	控制指令	LD SM0.0 = L60.0 LDN I0.2 = L63.7 LD L60.0 CALL AXIS0_CTRL, L63.7, M1.0, VB900, VD902, VD906, V910.0
程序段 2： 	开始使机器处于自动模式。	LD I0.0 AN I0.2 EU S Q0.2, 1 S M0.1, 1
程序段 3： 	E_Stop：立即停止并关闭自动模式。	LD I0.2 R Q0.2, 1

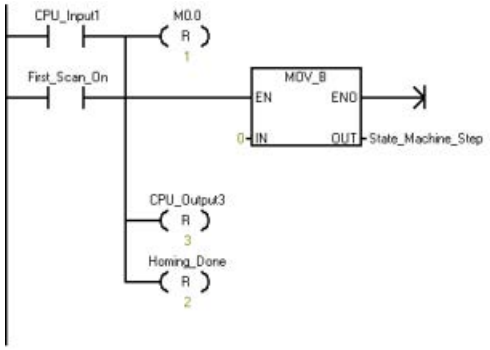
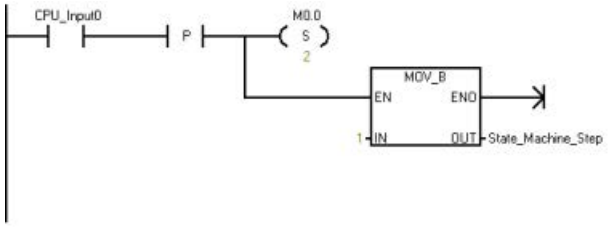
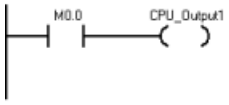
LAD/FBD	说明	STL
程序段 4：  	1. 移动到某个点： 2. 输入要剪切的长度。 3. 为目标速度输入“速度”。 4. 将模式设置为 1（相对模式）。	LD Q0.2 = L60.0 LD M0.1 EU = L63.7 LD L60.0 CALL AXIS0_GOTO, L63.7, VD500, VD504, 1, I0.2, Q0.4, VB920, VD922, VD926
程序段 5：  	就位时，开启刀具 2 秒钟以完成剪切。	LD Q0.2 A Q0.4 TON T33, +200 AN T33 = Q0.3
程序段 6：  	剪切完成时，重新启动，直至“停止”处于激活状态。	LD Q0.2 A T33 LPS AN I0.1 = M0.1 LPP A I0.1 R Q0.2, 1

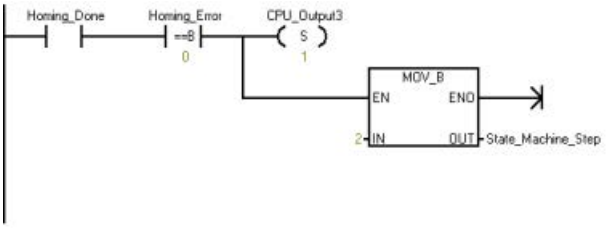
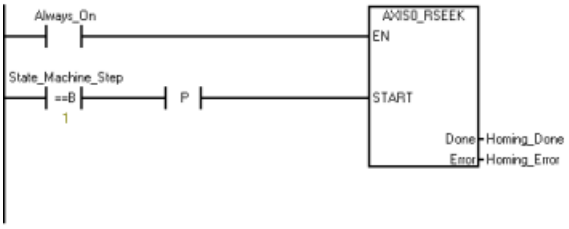
12.8.2 运动轴 AXISx_CTRL、AXISx_RUN、AXISx_SEEK 和 AXISx_MAN 示例

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。

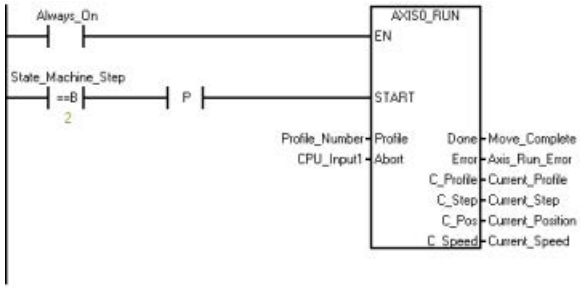
表格 12- 34 示例：运动轴 AXISx_CTRL、AXISx_RUN、AXISx_SEEK 和 AXISx_MAN 子程序应用

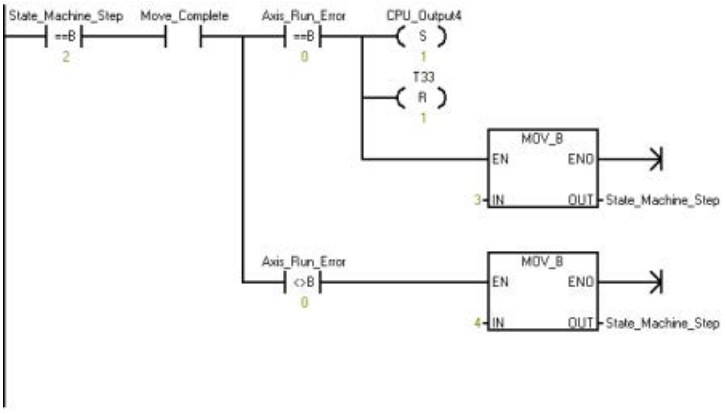
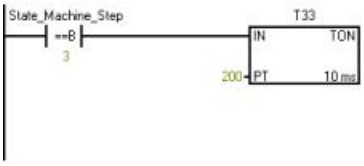
LAD/FBD	说明	STL
程序段 1 	通过关闭 CPU_Input1 启用轴。 符号和地址：[1] • Always_On = SM0.0 • AXIS0_CTRL = SBR1 • CPU_Input1 = I0.1	LD Always_On = L60.0 LDN CPU_Input1 = L63.7 LD L60.0 CALL AXIS0_CTRL, L63.7, M1.0, VB900, VD902, VD906, V910.0
程序段 2 	使用点动命令将轴移至已知位置。现在可手动移动轴。 符号和地址：[1] • AXIS0_MAN = SBR2 • CPU_Input10 = 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

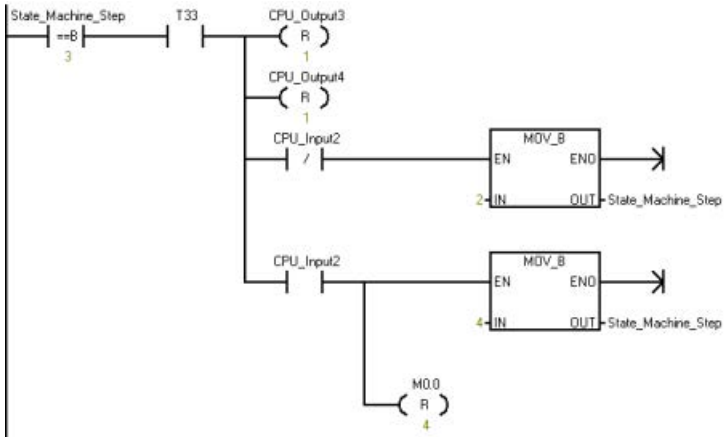
LAD/FBD	说明	STL
程序段 6 	完成归位后，移动至下一步。 符号和地址：[1] • CPU_Output3 = Q0.3 • Homing_Done = M1.1 • Homing_Error = VB930 • State_Machine_Step = VB1500	**LD Homing_Done** **AB= Homing_Error** **S CPU_Output3** **MOVB 2, State_Machine_ Step**
程序段 7 	当状态机处于步 1 时，系统自动将轴归位。如果存在归位错误，Homing_Error 输出将显示错误代码。 符号和地址：[1] • 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

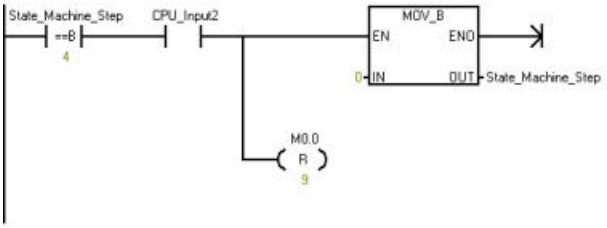
程序段 8
 
 	
	当状态机进入步 2 时，将移动所选的曲线
	
	符号和地址：[1]
	• Always_On = SM0.0
	• AXIS0_RUN = SBR4
	• Axis_Run_Error = VB940
	• CPU_Input1 = I0.1
	• Current_Position = VD914
	• Current_Profile = VB941
	• Current_Speed = VD948
	• Current_Step = VB942
	• Move_Complete = M1.2
	• Profile_Number = VB228
	• State_Machine_Step = 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**

LAD/FBD	说明	STL

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`

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`

LAD/FBD	说明	STL
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。

12.9 监视运动轴

为了帮助您开发运动控制解决方案，STEP 7-Micro/WIN SMART 提供了运动控制面板。

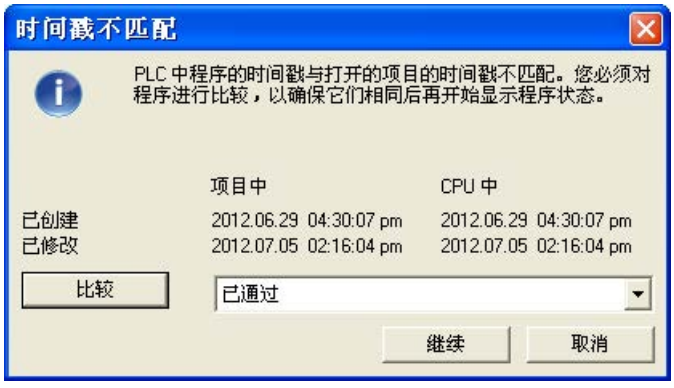
打开运动控制面板

使用以下方法之一打开运动控制面板：

- 在“工具”(Tools) 菜单功能区的“工具”(Tools) 区域单击“运动控制面板”(Motion Control Panel) 按钮。

- 在项目树中打开“工具”(Tools) 文件夹，选择“运动控制面板”(Motion Control Panel) 节点，然后按 Enter 键；或双击“运动控制面板”(Motion Control Panel) 节点。

此时执行 CPU 与 STEP 7-Micro/WIN SMART 的比较以确保组态相同。
（请参见下图。）



运动轴操作 (页 687)、组态 (页 692)和曲线组态 (页 693)设置使您能够轻松地在开发过程的启动和测试阶段监控运动轴操作。

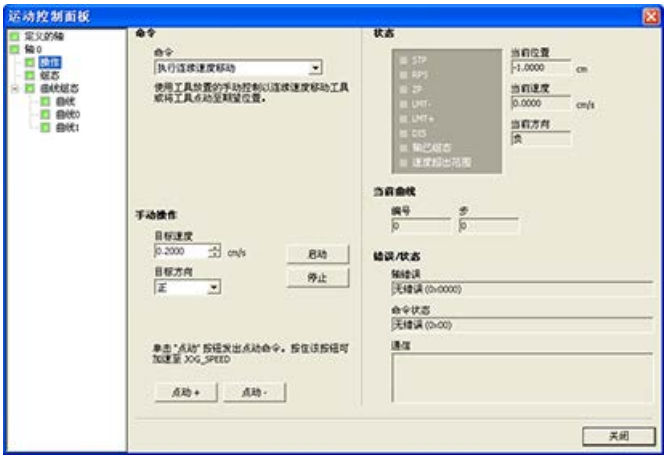
使用运动控制面板可检查运动轴接线是否正确、调整组态数据和测试各条运动曲线。

如果需要在运动轴中进行其它更改，请参见运动向导 (页 637)。

有关错误代码列表，请参见运动轴错误代码 (页 694)和运动指令错误代码 (页 696)。

12.9.1 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。

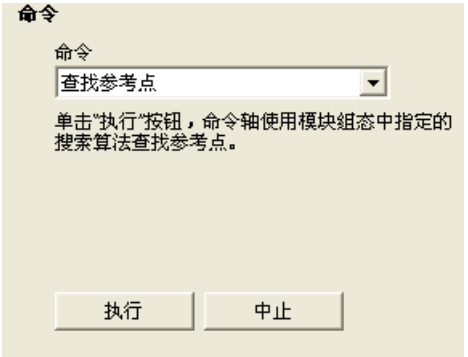
说明

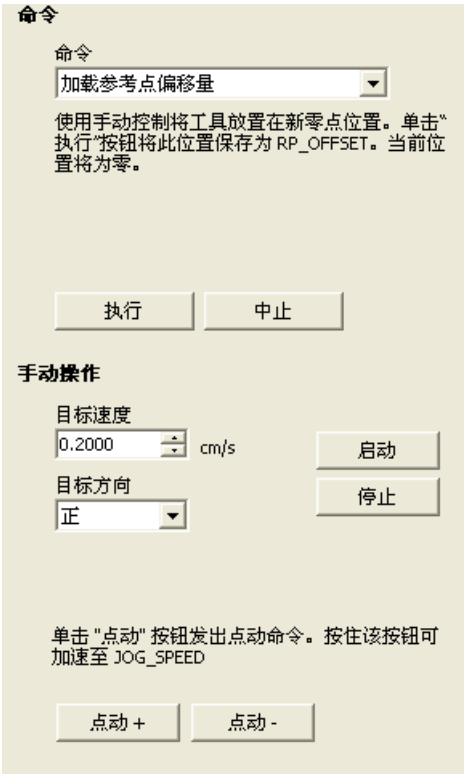
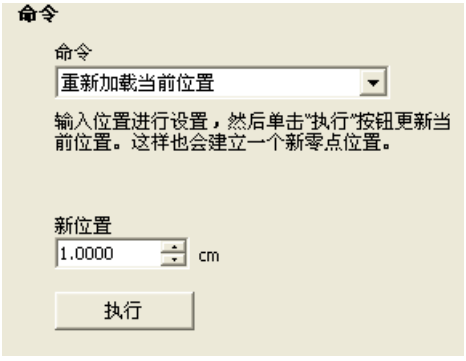
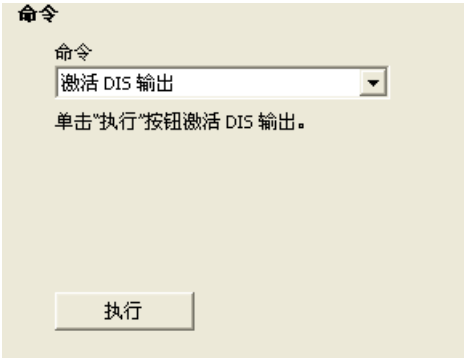
运动命令激活时，退出运动控制面板或失去通信会使该轴在 5 秒超时过后停止运动。

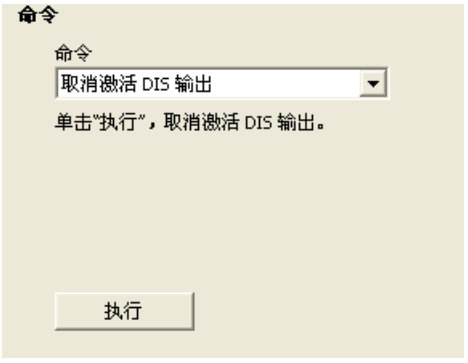
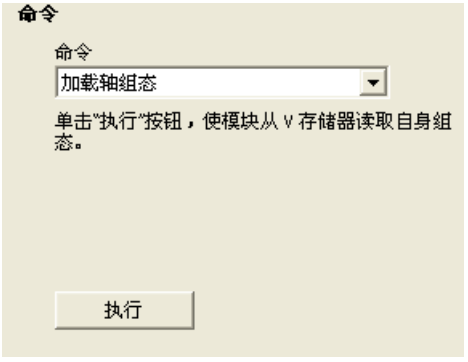
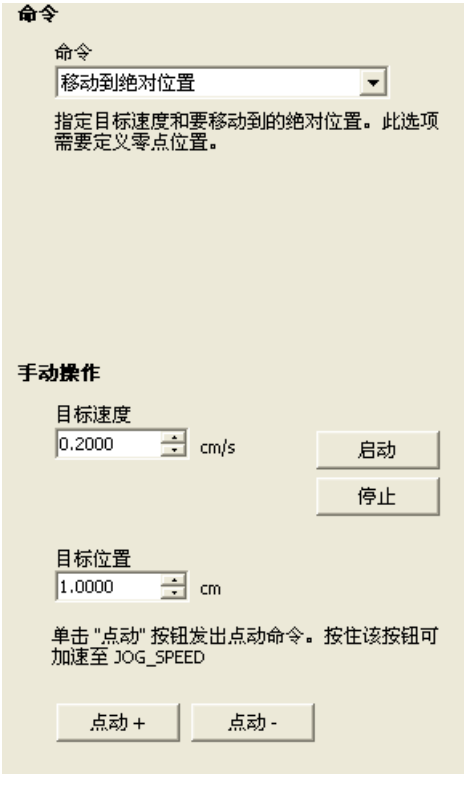
运动命令

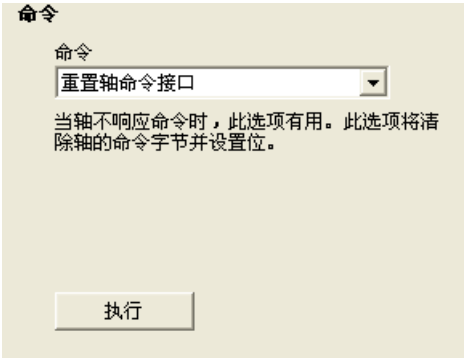
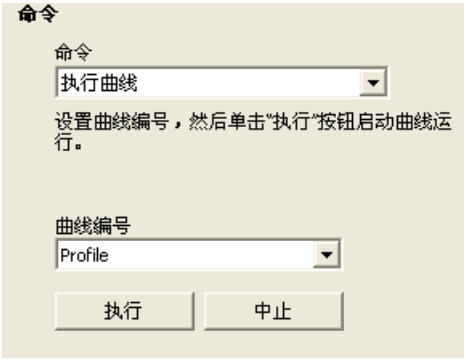
还可生成下列运动命令：

表格 12- 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。

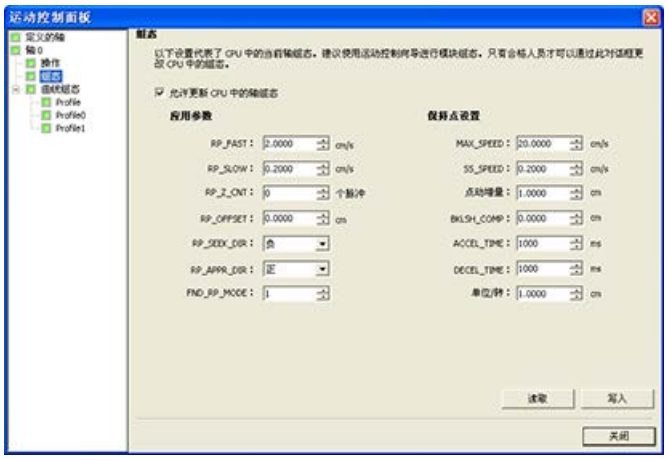
命令	说明
	加载参考点偏移量： 采用手动控制方式使工具点动运行到新位置后，加载“参考点偏移量”(Reference Point Offset)。 使用手动控制将工具置于新位置。 单击“执行”(Execute) 将此位置另存为“RP_OFFSET”。 当前位置将设置为零点。
	重新加载当前位置： 此命令更新当前位置值并建立新的零位置。 输入要设置的位置并单击“执行”(Execute) 更新当前位置。 这还将建立新的零位置。
	激活 DIS 输出： 此命令开启运动轴的 DIS 输出。 单击“执行”(Execute) 激活 DIS 输出。

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
	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)。
	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。
	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。

12.9.2 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。

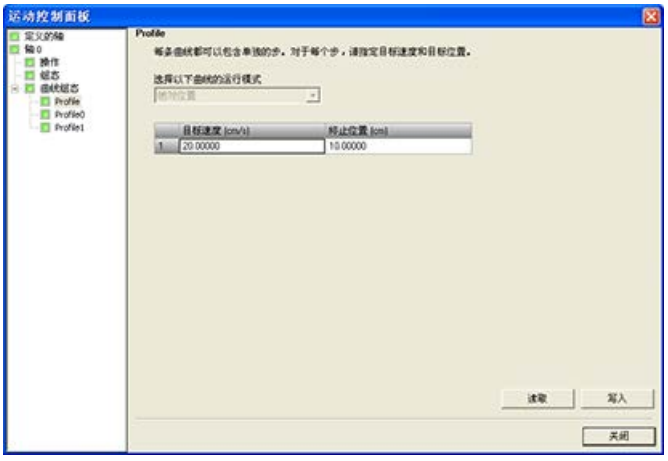
必须对项目进行可反映出这些字段最终值的手动更改。

12.9.3 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。



12.9.4 运动轴错误代码（SMW620、SMW670 或 SMW720 的 WORD）

表格 12- 36 运动轴错误代码

错误代码	说明
0	无错误
1	保留
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
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

错误代码	说明
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	保留
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	保留
41	SDB0 记录大小无效
42	SDB0 格式非法
43 到 127	保留

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。

12.9.5 运动指令的错误代码（SMB634、SMB684 或 SMB734 的七个 LS 位）

在每个轴的 SM 表中都会保留一个字节，用于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 12- 37 运动指令的错误代码

错误代码	说明
0	无错误
1	被用户中止
2	组态错误 （如果 SDB0 组态出错，则发生此错误。）
3	命令非法
4	因组态无效而中止 （如果组态表出错，则发生此错误。）
5	保留
6	由于无定义的参考点而中止
7	由于 STP 输入激活而中止
8	由于 LMT- 输入激活而中止
9	由于 LMT+ 输入激活而中止
10	由于执行运动时出现问题而中止
11	没有为指定曲线组态曲线块
12	操作模式非法
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

错误代码	说明
23	未定义 DIS 输出
24	保留
25	因 CPU 进入停止模式而中止
26	因运动控制面板活动结束而中止
27 到 127	保留
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	保留
130	运动轴未启用
131	保留
132	保留
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。

12.10 高级主题

12.10.1 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：

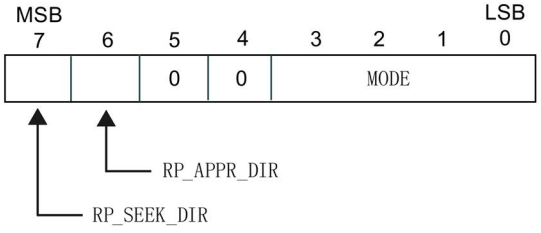
- **组态块：**包含在准备执行位置命令时用于设置运动轴的信息
- **交互块：**支持由用户程序直接设置位置参数
- **曲线块：**描述运动轴要执行的预定义移动操作。最多可组态 32 个包络块。

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 12- 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）	--
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	--

<table>
<tr><td colspan="4">组态/曲线表</td></tr>
<tr><td>字节偏移量</td><td>名称</td><td>功能说明</td><td>类型</td></tr>
<tr><td colspan="4">组态块</td></tr>
<tr><td>18</td><td>MEAS_SYS</td><td>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）</td><td>--</td></tr>
<tr><td>19</td><td>--</td><td>保留：设置为 0</td><td>--</td></tr>
<tr><td>20</td><td>PLS/REV</td><td>指定电机每转的脉冲数（仅当 MEAS_SYS 设置为 1 时可用）（4 字节）</td><td>DInt</td></tr>
<tr><td>24</td><td>UNITS/REV</td><td>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）</td><td>Real</td></tr>
<tr><td>28</td><td>UNITS</td><td>为 STEP 7-Micro/WIN SMART 保留，用于存储自定义单位字符串（4 字节）</td><td>--</td></tr>
<tr><td>32</td><td>RP_CFG</td><td>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逼近方向（0 - 正向，1 - 负向）。
<table>
<tr><td>MODE</td><td>指定参考点搜索方法</td></tr>
<tr><td>'0000'</td><td>参考点搜索被禁用。</td></tr>
<tr><td>'0001'</td><td>参考点在 RPS 输入变为激活状态的位置。</td></tr>
<tr><td>'0010'</td><td>参考点在 RPS 输入激活区域的中央。</td></tr>
</table>
</td><td>--</td></tr>
</table>

组态/曲线表				
字节偏移量	名称	功能说明		类型
组态块				
		'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
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	反冲补偿：用于对方向改变时的系统反冲进行补偿的距离（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			
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 12- 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

组态/曲线表			
字节偏移量	**名称**	**功能说明**	**类型**
交互块			
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 12- 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）	--

组态/曲线表			
字节偏移量	**名称**	**功能说明**	**类型**
曲线块 0			
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
214(+122)	第 15 步：POS	移动步 3 要到达的位置（4 字节）	DInt/Real
218(+126)	第 15 步：SPEED	移动步 3 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 12- 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 12- 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
+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 12- 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 12- 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 12- 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 12- 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 12- 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

12.10.2 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。

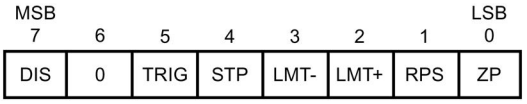
表格 12- 48 特殊存储器字节 SMB600 至 SMB749

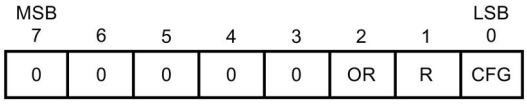
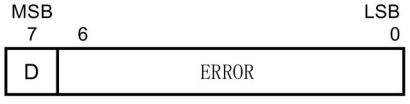
运动轴的 SM 字节：		
运动轴 0	运动轴 1	运动轴 2
SMB600 至 SMB649	SMB650 至 SMB699	SMB700 至 SMB749

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 12- 49 运动轴 0 的特殊存储区定义

SM 地址	说明
SMB600 至 SMB615	轴名称（16 个 ASCII 字符）。 SMB600 是第一个字符： “轴 0”
SMB616 至 SMB619	保留
SMW620	轴 0： 错误代码（请参见“运动轴错误代码” (页 694)列表。）

SM 地址	说明
SMB622	轴 0： 输入/输出状态： 反映输入和输出的状态  • 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-（反向限位输入）： – 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	说明
SMB623	轴 0 瞬时状态： 反映组态状态和转向状态  • 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。  • 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 696)列表。）
SMB635 至 SMB645	保留
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。

12.11 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：

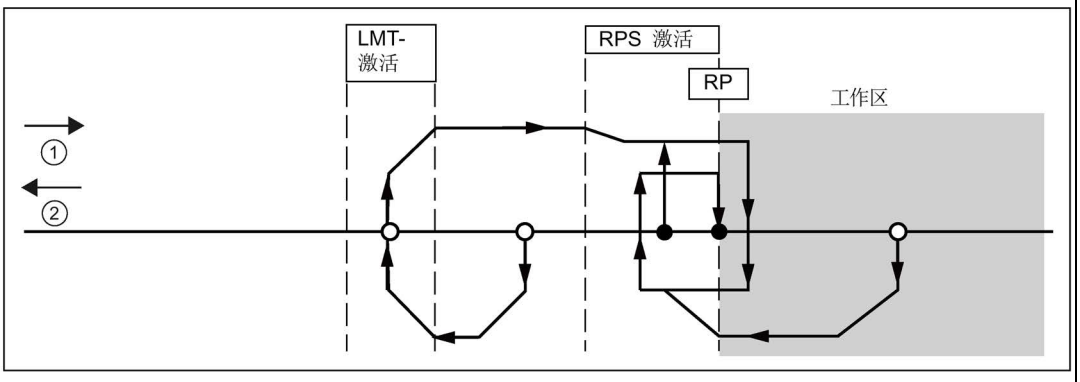
已定位每个图的工作区，以便从参考点移动到工作区需要以 RP 接近方向相同的方向移动。
通过在此方向选择工作区的位置，搜索参考点后移除了所有机械齿轮系统的反冲，以便第一次移动到工作区。

说明

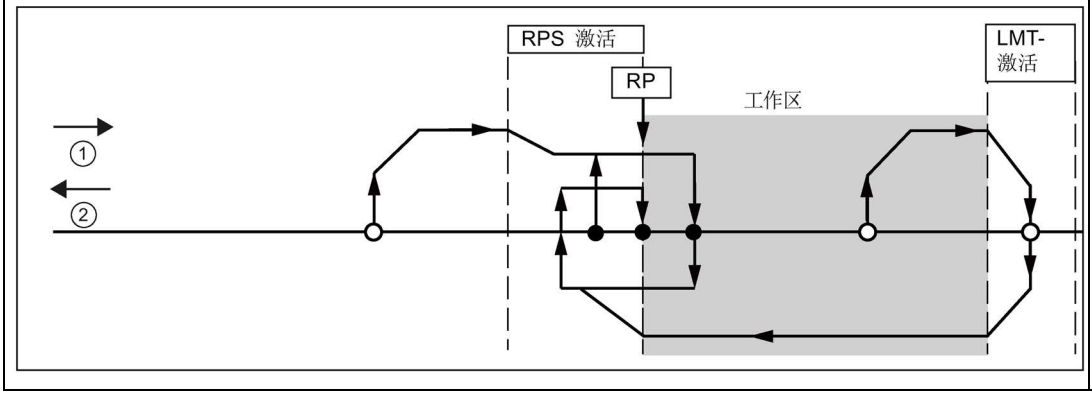
必须使能 RPS 输入才能使用 RP 搜索功能。 若要使用 RP 搜索模式 3 或模式 4，则还必须使能 ZP 输入，除非在进入 RPS 有效区后将要接收的 ZP 脉冲的数量组态为“0”。

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

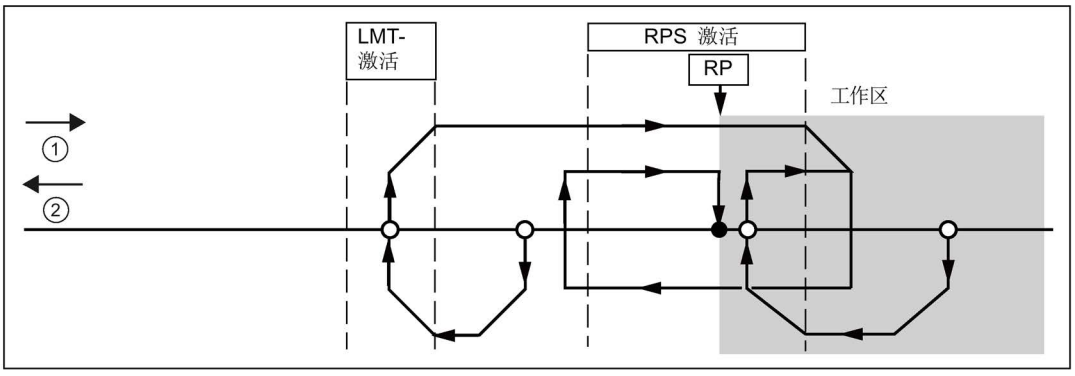


①: 正方向运动

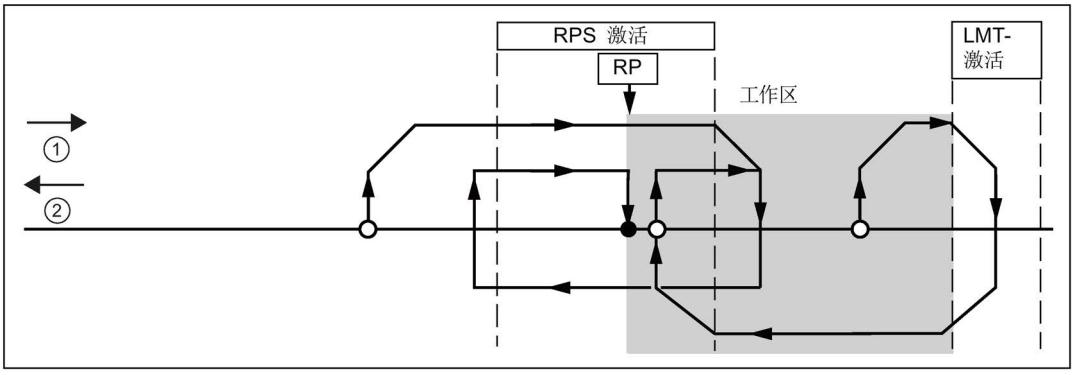
②: 负方向运动

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

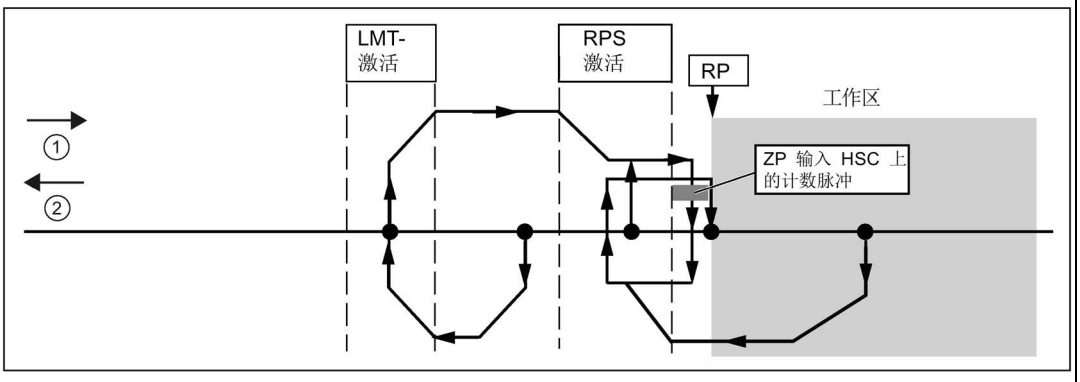


①: 正方向运动

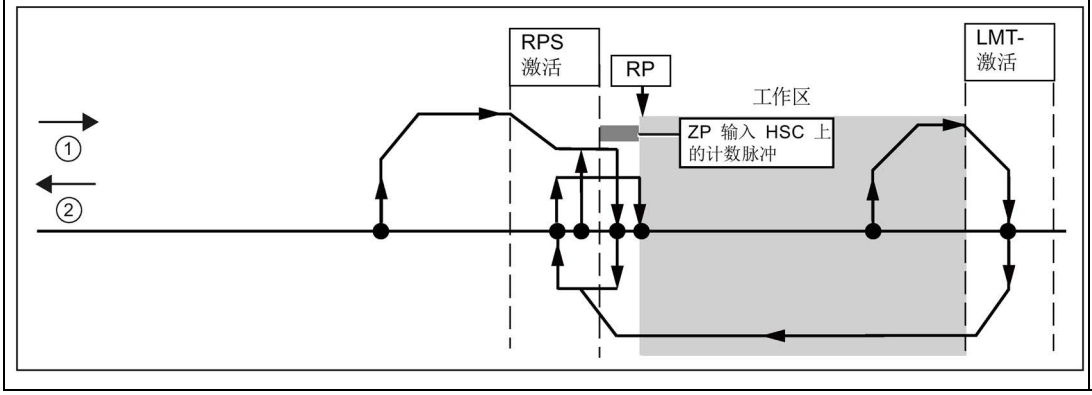
②: 负方向运动

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

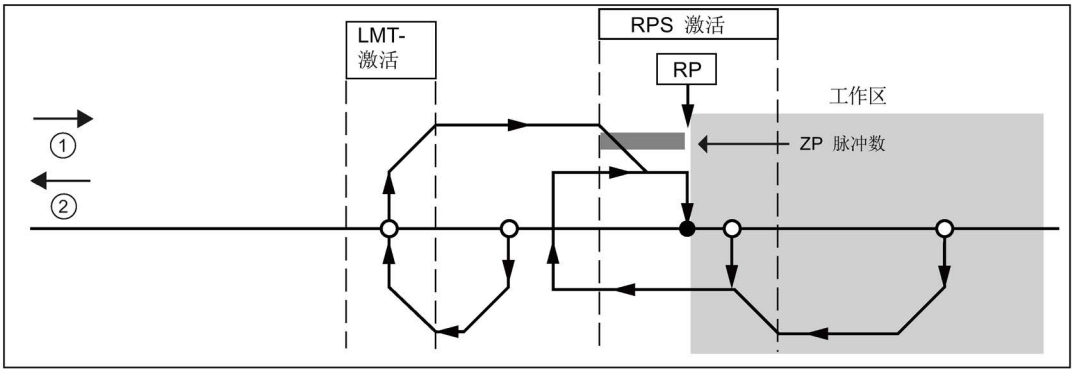
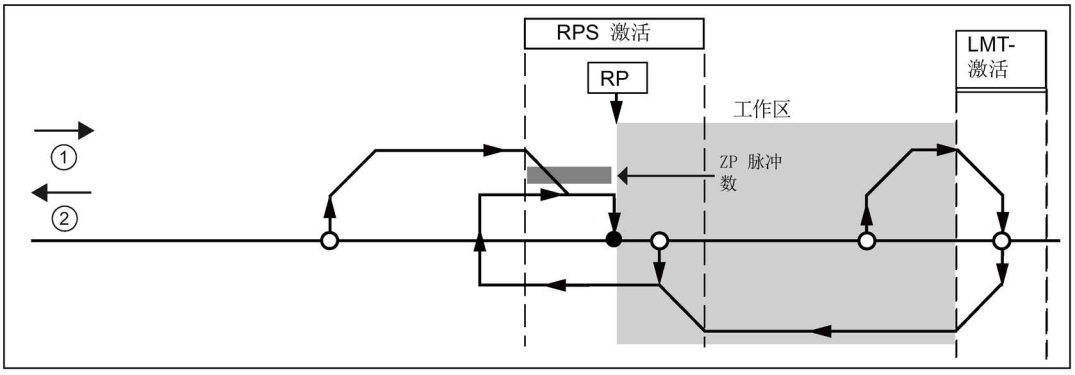


默认组态：RP 搜索方向：正方向，RP 接近方向：正方向



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

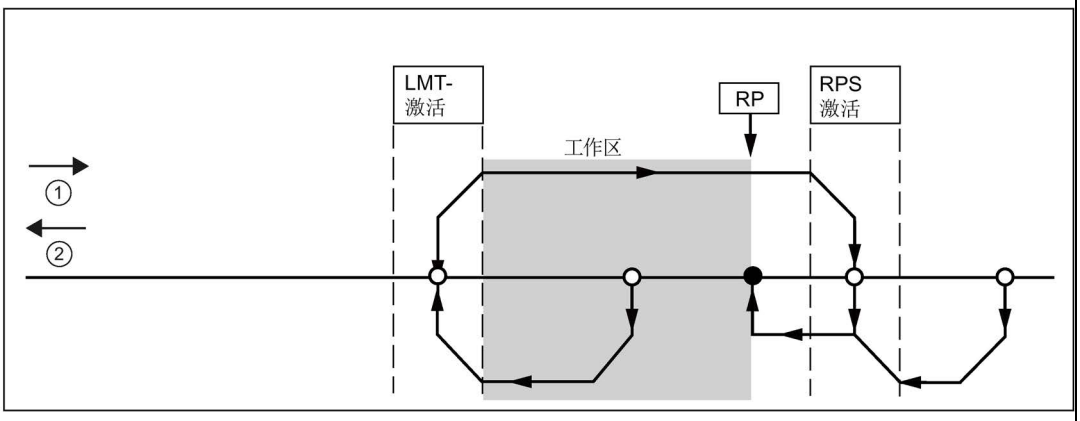
②: 负方向运动

12.11.1 选择工作区位置以消除反冲

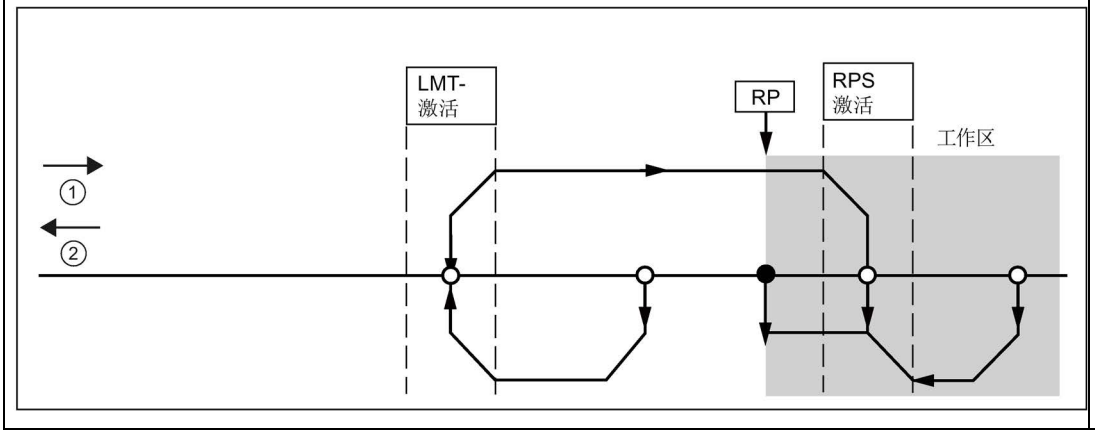
下图显示与消除反冲的接近方向的参考点 (RP)、RPS 激活区域和限位开关（LMT+ 和 LMT-）关联的工作区。 示图第二部分放置工作区，以便不消除反冲。 下图显示 RP 搜索模式 3。对于每个其他 RP
搜索模式的每个搜索序列，类似的工作区布置是可能的（尽管不建议）。

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



反冲未消除： 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

技术规范

A

A.1 常规规范

A.1.1 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

CE

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)。

- EC 指令 2006/95/EC（低压指令）“设计用于特定电压限值内的电气设备”
  - EN 61131-2：可编程控制器 — 设备要求和测试
- EC 指令 2004/108/EC（EMC 指令）“电磁兼容性”
  - 辐射标准
    EN 61000-6-4：AI：工业环境
  - 抗扰度标准
    EN 61000-6-2：工业环境

可向主管部门出具的所持 CE 一致性声明文件位于以下地址：

Siemens AG
Sector Industry
DF FA AS DH AMB
Postfach 1963
D-92209 Amberg
Germany

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	±8 kV，对所有表面的空中放电 ±4 kV，对暴露导电表面的接触放电
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 2.0 GHz，3 V/m，1 kHz 时 80% AM 2.0 到 2.7 GHz，1 V/m，1 kHz 时 80% AM
EN 61000-4-4 快速瞬变脉冲	2 kV，5 kHz，到交流和直流系统电源的耦合网络 2 kV，5 kHz，到 I/O 的耦合夹
EN 6100-4-5 浪涌抗扰度	AC 系统 - 2 kV 共模，1kV 差模 DC 系统 - 2 kV 共模，1kV 差模 对于 DC 系统（I/O 信号、DC 电源系统)，需要外部保护。
EN 61000-4-6 抗传导干扰	150 kHz 到 80 MHz，10 V RMS，1 kHz 时 80% AM
EN 61000-4-11 电压骤降	交流系统 60 Hz 时，0% 持续 1 个周期、40% 持续 12 个周期和 70% 持续 30 个周期

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	<40dB (μV/m) 准峰值；测量距离为 10 m
	230 MHz 到 1 GHz	<47dB (μV/m) 准峰值；测量距离为 10 m

环境条件

表格 A- 4　运输与存储

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，2 个周期
EN 60068-2-32，自由落体	0.3 m，5 次，产品包装
大气压	1080 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）	0 °C 到 55 °C 水平安装 0 °C 到 45 °C 垂直安装 湿度 95%，不结露
大气压	1080 至 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	5 °C 到 55 °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	2300 V AC 或 3250 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

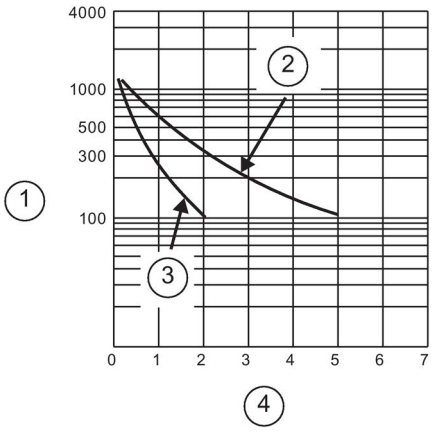
额定电压	容错
24 V DC	20.4 V DC 到 28.8 V DC
120/240 V AC	85 V AC 到 264 V AC，47 到 63 Hz

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。

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 ST20 和 CPU SR20

A.2.1.1 常规规范和特性

CPU ST20 和 CPU SR20 的常规规范和特性

表格 A- 8 常规规范

技术数据	CPU ST20 DC/DC/DC	CPU SR20 AC/DC/继电器
产品编号	6ES7288-1ST20-0AA0	6ES7288-1SR20-0AA0
尺寸 W x H x D (mm)	90 x 100 x 81	90 x 100 x 81
重量	320 g	367.3 g
功耗	20 W	14 W
可用电流（EM 总线）	最大 1400 mA(5 V DC)	最大 1400 mA(5 V DC)
可用电流 (24 V DC)	最大 300 mA（传感器电源）	最大 300 mA（传感器电源）
数字量输入电流消耗 (24 V DC)	所用的每点输入 4 mA	所用的每点输入 4 mA

表格 A- 9　CPU 特征

技术数据		说明
		CPU ST20 DC/DC/DC，CPU SR20 AC/DC/继电器
用户存储器 [1]	程序	12 KB
	用户数据 (V)	8 KB
	保持性	最大 10 KB [1]
板载数字量 I/O		12 点输入/8 点输出
过程映像		256 位输入 (I)/256 位输出 (Q)
模拟图像		56 个字的输入 (AI)/56 个字的输出 (AQ)
位存储器 (M)		256 位
临时（局部）存储器 (L)		主程序中 64 字节和每个子例程和中断例程中 64 字节 采用 LAD 或 FBD 进行编程时为 60 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		6
信号板扩展		最多 1 个
高速计数器		共 4 个 • 4 个以 200 K Hz 单相形式工作 • 2 个以 100 KHz A/B 相形式工作
脉冲输出 [2]		100 KHz 时 2 个[2]
脉冲捕捉输入		12
循环中断		2 个，分辨率为 1 ms
沿中断		4 个上升沿和 4 个下降沿（使用可选信号板时，各为 6 个）
存储卡		microSDHC 卡（可选）
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（免维护超级电容）

1　可组态 V 存储器、M 存储器、C 存储器的存储区（当前值），以及 T 存储器要保持的部分（保持性定时器上的当前值），最大可为最大指定量。

2　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　性能

指令类型	**执行速度**
布尔运算	150 ns/指令
移动字	1.2 μs/指令
实数数学运算	3.6 μs/指令

表格 A- 11　所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） 中断例程：128（0 到 127）
	嵌套深度	从主程序：8 个子例程级别 从中断例程：4 个子例程级别
累加器	数量	4
定时器	类型/数量	非保持性（TON、TOF）：192 保持性 (TONR)：64
计数器	数量	256

表格 A- 12　通信

技术数据	**说明**
端口数	以太网：1 串行端口：1 (RS485) 附加串行端口：1（带有可选 RS232/485 信号板）
HMI 设备	以太网：8 个连接 串行端口：每个端口 4 个连接
编程设备 (PG)	以太网：1 个连接
CPU (PUT/GET)	以太网：8 个客户端和 8 个服务器连接
开放式用户通信	以太网：8 个主动和 8 个被动连接
数据传输率	以太网：10/100 Mb/s RS485 系统协议：9600、19200 和 187500 b/s RS485 自由端口：1200 到 115200 b/s

技术数据	说明
隔离（外部信号与 PLC 逻辑侧）	以太网：变压器隔离，1500 V DC RS485：无
电缆类型	以太网：CAT5e 屏蔽电缆 RS485：PROFIBUS 网络电缆

表格 A- 13　电源

技术数据		CPU ST20 DC/DC/DC	CPU SR20 AC/DC/继电器
电压范围		20.4 到 28.8 V DC	85 到 264 V AC
电源频率		-	47 到 63 Hz
输入电流	最大负载时仅包括 CPU	24 V DC 时 160 mA（无 300 mA 传感器驱动功率） 24 V DC 时 430 mA（带 300 mA 传感器驱动功率）	120 V AC 时 210 mA（带 300 mA 功率传感器输出） 120 V AC 时 90 mA（无 300 mA 功率传感器输出） 240 V AC 时 120 mA（带 300 mA 功率传感器输出） 240 V AC 时 60 mA（无 300 mA 功率传感器输出）
	最大负载时包括 CPU 和所有扩展附件	24 V DC 时 720 mA	120 V AC 时 290 mA 240 V AC 时 170 mA
浪涌电流（最大）		28.8 V DC 时 11.7 A	264 V AC 时 9.3 A
隔离（输入电源与逻辑侧）		-	1500 V AC
漏地电流，交流线路对功能地		-	最大 0.5 mA
保持时间（掉电）		24 V DC 时 20 ms	120 V AC 时 30 ms 240 V AC 时 200 ms
内部保险丝，用户不可更换		3 A，250 V，慢速熔断	3 A，250 V，慢速熔断

表格 A- 14　传感器电源

技术数据	CPU ST20 DC/DC/DC	CPU SR20 AC/DC/继电器
电压范围	20.4 到 28.8 V DC	20.4 到 28.8 V DC
额定输出电流（最大）	300 mA（短路保护）	300 mA（短路保护）
最大波纹噪声 (<10 MHz)	< 1 V 峰峰值	< 1 V 峰峰值
隔离（CPU 逻辑侧与传感器电源）	未隔离	未隔离

A.2.1.2　数字量输入和输出

表格 A- 15　数字量输入

技术数据	CPU ST20 DC/DC/DC	CPU SR20 AC/DC/继电器
输入点数	12	12
类型	漏型/源型（IEC 1 类漏型，除 I0.0 到 I0.3，I0.6 到 I0.7）	漏型/源型（IEC 1 类漏型）
额定电压	4 mA 时 24 V DC，额定值	4 mA 时 24 V DC，额定值
允许的连续电压	30 V DC，最大值	30 V DC，最大值
浪涌电压	35 V DC，持续 0.5 s	35 V DC，持续 0.5 s
逻辑 1 信号（最小）	I0.0 到 I0.3，I0.6 到 I0.7：8 mA 时 4 V DC 其它输入：2.5 mA 时 15 V DC	2.5 mA 时 15 V DC
逻辑 0 信号（最大）	I0.0 到 I0.3，I0.6 到 I0.7：1 mA 时 1 V DC 其它输入：1 mA 时 5 V DC	1 mA 时 5 V DC
隔离（现场侧与逻辑侧）	500 V AC，持续 1分钟	500 V AC，持续 1分钟
隔离组	1	1
滤波时间	每个通道上可单独选择（点 I0.0 到 I1.3）： 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	每个通道上可单独选择（点 I0.0 到 I1.3）： 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

技术数据	CPU ST20 DC/DC/DC	CPU SR20 AC/DC/继电器
HSC 时钟输入频率（最大）（逻辑 1 电平 = 15 到 26 V DC）	4 个 HSC，每个 200 KHz，单相 2 个 HSC，每个 100 KHz，A/B 相	4 个 HSC，每个 200 KHz，单相 2 个 HSC，每个 100 KHz，A/B 相
同时接通的输入数	12	12
电缆长度（最大值），以米为单位	I0.0 到 I0.3： 屏蔽（仅限此类）： • 500 m 正常（低速）输入 • 50 m HSC（高速）输入	所有输入： • 屏蔽： 500 m 正常输入，50 m HSC 输入 • 非屏蔽：300 m 正常输入
	I0.6 到 I0.7： • 屏蔽（仅限此类）：500 m 正常输入	
	所有其它输入： • 屏蔽：500 m 正常输入 • 非屏蔽：300 m 正常输入	

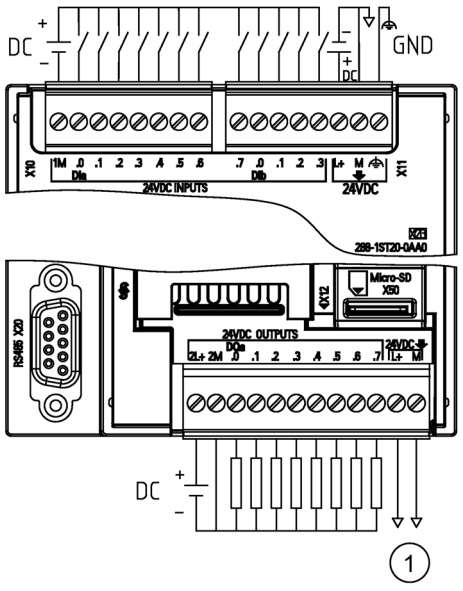
表格 A- 16　数字量输出

技术数据	CPU ST20 DC/DC/DC	CPU SR20 AC/DC/继电器
输出点数	8	8
类型	固态 - MOSFET（源型）	继电器，干触点
电压范围	20.4 到 28.8 V DC	5 到 30 V DC 或 5 到 250 V AC
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	2.0 A
每个公共端的额定电流（最大）	6 A	10.0 A
灯负载	5 W	30 W DC/200 W AC
通态电阻	最大 0.6 Ω	新设备最大为 0.2 Ω
每点的漏电流	最大 10 μA	-
浪涌电流	8 A，最长持续 100 ms	触点闭合时为 7 A

技术数据	CPU ST20 DC/DC/DC	CPU SR20 AC/DC/继电器
过载保护	无	无
隔离（现场侧与逻辑侧）	500 V AC，持续 1分钟	1500 V AC，持续 1分钟（线圈与触点） 无（线圈与逻辑侧）
隔离电阻	-	新设备最小为 100 MΩ
断开触点间的绝缘	-	750 V AC，持续 1 分钟
隔离组	2	1
电感钳位电压	L+ - 48 V DC，1 W 损耗	不推荐
开关延迟（Qa.0 到 Qa.3）	断开到接通最长为 1.0 µs 接通到断开最长为 3.0 µs	最长 10 ms
开关延迟（Qa.4 到 Qa.7）	断开到接通最长为 50 µs 接通到断开最长为 200 µs	最长 10 ms
机械寿命（无负载）	-	10,000,000 个断开/闭合周期
额定负载下的触点寿命	-	100,000 个断开/闭合周期
STOP 模式下的输出状态	上一个值或替换值（默认值为 0）	上一个值或替换值（默认值为 0）
同时接通的输出数	8	8
电缆长度（最大值），以米为单位	屏蔽：500 m 非屏蔽：300 m	屏蔽：500 m 非屏蔽：300 m

A.2.1.3 CPU ST20 和 CPU SR20 接线图

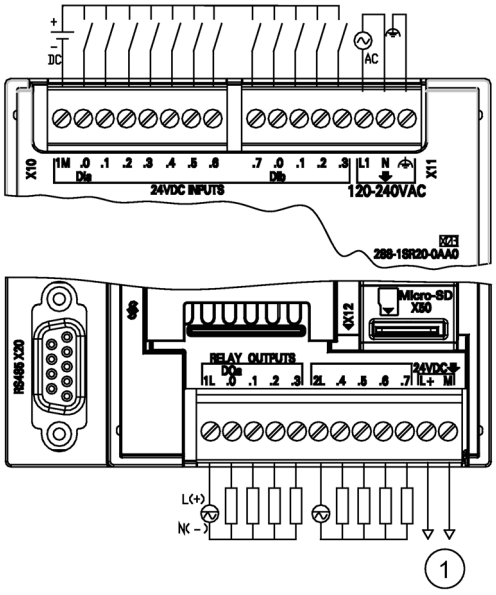
表格 A- 17 CPU ST20 DC/DC/DC (6ES7288-1ST20-0AA0) 的接线图

CPU ST20 DC/DC/DC (6ES7288-1ST20-0AA0)		
	①	24 V DC 传感器电源输出

表格 A- 18 CPU ST20 DC/DC/DC (6ES7288-1ST20-0AA0) 的连接器引脚位置

引脚	X10	X11	X12
1	1M	DI a.7	2L+
2	DI a.0	DI b.0	2M
3	DI a.1	DI b.1	DQ a.0
4	DI a.2	DI b.2	DQ a.1
5	DI a.3	DI b.3	DQ a.2
6	DI a.4	L+ / 24 V DC	DQ a.3
7	DI a.5	M / 24 V DC	DQ a.4
8	DI a.6	功能性接地	DQ a.5
9	-	-	DQ a.6
10	-	-	DQ a.7
11	-	-	L+ / 24 V DC
12	-	-	M/ 24 V DC

表格 A- 19　CPU SR20 AC/DC/继电器 (6ES7288-1SR20-0AA0) 的接线图

CPU SR20 AC/DC/继电器 (6ES7288-1SR20-0AA0)		
	①	24 V DC 传感器电源输出

表格 A- 20　CPU SR20 AC/DC/继电器 (6ES7288-1SR20-0AA0) 的连接器引脚位置

引脚	X10	X11	X12
1	1M	DI a.7	1L
2	DI a.0	DI b.0	DQ a.0
3	DI a.1	DI b.1	DQ a.1
4	DI a.2	DI b.2	DQ a.2
5	DI a.3	DI b.3	DQ a.3
6	DI a.4	L1 / 120 - 240 V AC	2L
7	DI a.5	N / 120 - 240 V AC	DQ a.4
8	DI a.6	功能性接地	DQ a.5
9	-	-	DQ a.6
10	-	-	DQ a.7
11	-	-	L+ / 24 V DC 输出
12	-	-	M / 24 V DC 输出

A.2.2 CPU ST30 和 CPU SR30

A.2.2.1 常规规范和特性

CPU ST30 和 CPU SR30 的常规规范和特性

表格 A- 21 常规规范

技术数据	CPU ST30 DC/DC/DC	CPU SR30 AC/DC/继电器
产品编号	6ES7288-1ST30-0AA0	6ES7288-1SR30-0AA0
尺寸 W x H x D (mm)	110 x 100 x 81	110 x 100 x 81
重量	375 g	435 g
功耗	12 W	14 W
可用电流（EM 总线）	最大 1400 mA(5 V DC)	最大 1400 mA(5 V DC)
可用电流 (24 V DC)	最大 300 mA（传感器电源）	最大 300 mA（传感器电源）
数字量输入电流消耗 (24 V DC)	所用的每点输入 4 mA	所用的每点输入 4 mA

表格 A- 22　CPU 特征

技术数据		**说明**	
		CPU ST30 DC/DC/DC	**CPU SR30 AC/DC/继电器**
用户存储器 [1]	程序	18 KB	18 KB
	用户数据 (V)	12 KB	12 KB
	保持性	最大 10 KB [1]	最大 10 KB [1]
板载数字量 I/O		18 点输入/12 点输出	18 点输入/12 点输出
过程映像		256 位输入 (I)/256 位输出 (Q)	256 位输入 (I)/256 位输出 (Q)
模拟图像		56 个字的输入 (AI)/56 个字的输出 (AQ)	56 个字的输入 (AI)/56 个字的输出 (AQ)
位存储器 (M)		256 位	256 位
临时（局部）存储器 (L)		主程序中 64 字节和每个子例程和中断例程中 64 字节 采用 LAD 或 FBD 进行编程时为 60 字节（STEP 7-Micro/WIN 保留 4 字节）	主程序中 64 字节和每个子例程和中断例程中 64 字节 采用 LAD 或 FBD 进行编程时为 60 字节（STEP 7-Micro/WIN 保留 4 字节）
顺序控制继电器 (S)		256 位	256 位
扩展模块扩展		6	6
信号板扩展		最多 1 个	最多 1 个
高速计数器		共 4 个 • 200 KHz 时 4 个，针对单相 • 100 KHz 时 2 个，针对 A/B 相	共 4 个 • 200 KHz 时 4 个，针对单相 • 100 KHz 时 2 个，针对 A/B 相
脉冲输出 [2]		3 个，100 KHz	-
脉冲捕捉输入		12	12
循环中断		2 个，分辨率为 1 ms	2 个，分辨率为 1 ms
沿中断		4 个上升沿和 4 个下降沿（使用可选信号板时，各为 6 个）	4 个上升沿和 4 个下降沿（使用可选信号板时，各为 6 个）
存储卡		microSDHC 卡（可选）	microSDHC 卡（可选）

技术数据	说明	
	CPU ST30 DC/DC/DC	CPU SR30 AC/DC/继电器
实时时钟精度	+/- 120 秒/月	+/- 120 秒/月
实时时钟保持时间	通常为 7 天，25 °C 时最少为 6 天（免维护超级电容）	通常为 7 天，25 °C 时最少为 6 天（免维护超级电容）

1 可组态 V 存储器、M 存储器、C 存储器的存储区（当前值），以及 T 存储器要保持的部分（保持性定时器上的当前值），最大可为最大指定量。

2 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- 23 性能

指令类型	执行速度
布尔运算	150 ns/指令
移动字	1.2 μs/指令
实数数学运算	3.6 μs/指令

表格 A- 24 所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） 中断例程：128（0 到 127）
	嵌套深度	从主程序：8 个子例程级别 从中断例程：4 个子例程级别
累加器	数量	4
定时器	类型/数量	非保持性（TON、TOF）：192 保持性 (TONR)：64
计数器	数量	256

表格 A- 25　通信

技术数据	说明
端口数	以太网：1 串行端口：1 (RS485) 附加串行端口：1（带有可选 RS232/485 信号板）
HMI 设备	以太网：8 个连接 串行端口：每个端口 4 个连接
编程设备 (PG)	以太网：1 个连接
CPU (PUT/GET)	以太网：8 个客户端和 8 个服务器连接
开放式用户通信	以太网：8 个主动和 8 个被动连接
数据传输率	以太网：10/100 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）	以太网：变压器隔离，1500 V DC RS485：无
电缆类型	以太网：CAT5e 屏蔽电缆 RS485：PROFIBUS 网络电缆

表格 A- 26　电源

技术数据		CPU ST30 DC/DC/DC	CPU SR30 AC/DC/继电器
电压范围		20.4 到 28.8 V DC	85 到 264 V AC
电源频率		-	47 到 63 Hz
输入电流	最大负载时仅包括 CPU	24 V DC 时 64 mA（无 300 mA 传感器驱动功率） 24 V DC 时 365 mA（带 300 mA 传感器驱动功率）	120 V AC 时 92 mA（带功率传感器） 120 V AC 时 40 mA（无功率传感器） 240 V AC 时 52 mA（带功率传感器） 240 V AC 时 27 mA（无功率传感器）

技术数据		CPU ST30 DC/DC/DC	CPU SR30 AC/DC/继电器
	最大负载时包括 CPU 和所有扩展附件	24 V DC 时 624 mA	120 V AC 时 136 mA 240 V AC 时 72 mA
浪涌电流（最大）		28.8 V DC 时 6 A	264 V AC 时 8.9 A
隔离（输入电源与逻辑侧）		-	1500 V AC
漏地电流，交流线路对功能地		-	最大 0.5 mA
保持时间（掉电）		24 V DC 时 20 ms	120 V AC 时 30 ms 240 V AC 时 200 ms
内部保险丝，用户不可更换		3 A，250 V，慢速熔断	3 A，250 V，慢速熔断

表格 A- 27　传感器电源

技术数据	CPU ST30 DC/DC/DC	CPU SR30 AC/DC/继电器
电压范围	20.4 到 28.8 V DC	20.4 到 28.8 V DC
额定输出电流（最大）	300 mA（短路保护）	300 mA（短路保护）
最大波纹噪声 (<10 MHz)	< 1 V 峰峰值	< 1 V 峰峰值
隔离（CPU 逻辑侧与传感器电源）	未隔离	未隔离

A.2.2.2 数字量输入和输出

表格 A- 28 数字量输入

技术数据	CPU ST30 DC/DC/DC	CPU SR30 AC/DC/继电器
输入点数	18	18
类型	漏型/源型（IEC 1 类漏型，除 I0.0 到 I0.3，I0.6 到 I0.7）	漏型/源型（IEC 1 类漏型）
额定电压	4 mA 时 24 V DC，额定值	4 mA 时 24 V DC，额定值
允许的连续电压	30 V DC，最大值	30 V DC，最大值
浪涌电压	35 V DC，持续 0.5 s	35 V DC，持续 0.5 s
逻辑 1 信号（最小）	I0.0 到 I0.3，I0.6 到 I0.7：8 mA 时 4 V DC 其它输入：2.5 mA 时 15 V DC	2.5 mA 时 15 V DC
逻辑 0 信号（最大）	I0.0 到 I0.3，I0.6 到 I0.7：1 mA 时 1 V DC 其它输入：1 mA 时 5 V DC	1 mA 时 5 V DC
隔离（现场侧与逻辑侧）	500 V AC，持续 1分钟	500 V AC，持续 1分钟
隔离组	1	1
滤波时间	每个通道上可单独选择（点 I0.0 到 I1.5）： 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	每个通道上可单独选择（点 I0.0 到 I1.5）： 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
	每个通道上可单独选择（点 I1.6 和更大）： ms：0、6.4、12.8	每个通道上可单独选择（点 I1.6 和更大）： ms：0、6.4、12.8
HSC 时钟输入频率（最大）（逻辑 1 电平 = 15 到 26 V DC）	4 个 HSC，每个 200 KHz，单相 2 个 HSC，每个 100 KHz，A/B 相	4 个 HSC，每个 200 KHz，单相 2 个 HSC，每个 100 KHz，A/B 相

技术数据	CPU ST30 DC/DC/DC	CPU SR30 AC/DC/继电器
同时接通的输入数	18	18
电缆长度（最大值），以米为单位	I0.0 到 I0.3： 屏蔽（仅限此类）： • 500 m 正常（低速）输入 • 50 m HSC（高速）输入	所有输入： • 屏蔽：500 m 正常输入，50 m HSC 输入 • 非屏蔽：300 m 正常输入
	I0.6 到 I0.7： • 屏蔽（仅限此类）：500 m 正常输入	
	所有其它输入： • 屏蔽：500 m 正常输入 • 非屏蔽：300 m 额定输入[1]	

[1] 当 I0.0 到 I0.3 用于高速计数器输入，其它所有输入必须使用屏蔽电缆。

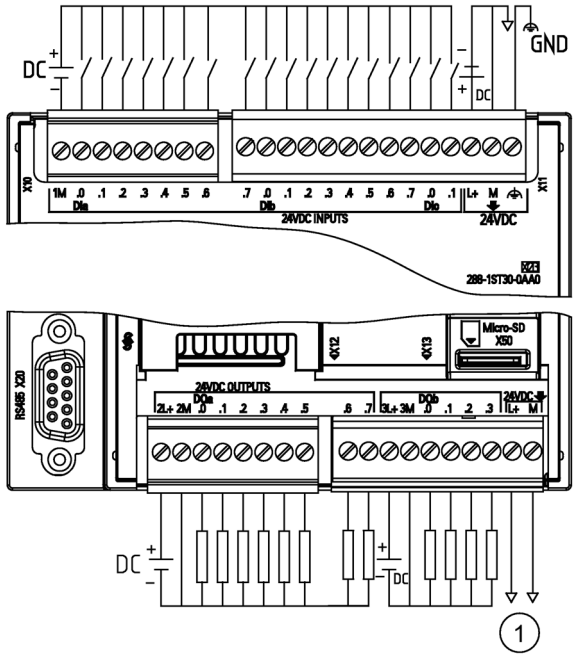
表格 A- 29　数字量输出

技术数据	CPU ST30 DC/DC/DC	CPU SR30 AC/DC/继电器
输出点数	12	12
类型	固态 - MOSFET（源型）	继电器，干触点
电压范围	20.4 到 28.8 V DC	5 到 30 V DC 或 5 到 250 V AC
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	2.0 A
每个公共端的额定电流（最大）	6 A	10.0 A
灯负载	5 W	30 W DC/200 W AC
通态电阻	最大 0.6 Ω	新设备最大为 0.2 Ω
每点的漏电流	最大 10 μA	-
浪涌电流	8 A，最长持续 100 ms	触点闭合时为 7 A
过载保护	无	无

技术数据	CPU ST30 DC/DC/DC	CPU SR30 AC/DC/继电器
隔离（现场侧与逻辑侧）	500 V AC，持续 1分钟	1500 V AC，持续 1分钟（线圈与触点） 无（线圈与逻辑侧）
隔离电阻	-	新设备最小为 100 MΩ
断开触点间的绝缘	-	750 V AC，持续 1 分钟
隔离组	1	1
电感钳位电压	L+ - 48 V DC，1 W 损耗	不推荐
开关延迟（Qa.0 到 Qa.3）	断开到接通最长为 1.0 µs 接通到断开最长为 3.0 µs	最长 10 ms
开关延迟（Qa.4 到 Qb.7）	断开到接通最长为 50 µs 接通到断开最长为 200 µs	最长 10 ms
机械寿命（无负载）	-	10,000,000 个断开/闭合周期
额定负载下的触点寿命	-	100,000 个断开/闭合周期
STOP 模式下的输出状态	上一个值或替换值（默认值为 0）	上一个值或替换值（默认值为 0）
同时接通的输出数	12	12
电缆长度（最大值），以米为单位	屏蔽：500 m 非屏蔽：150 m	屏蔽：500 m 非屏蔽：150 m

A.2.2.3 CPU ST30 和 CPU SR30 接线图

表格 A- 30 CPU ST30 DC/DC/DC (6ES7288-1ST30-0AA0) 的接线图

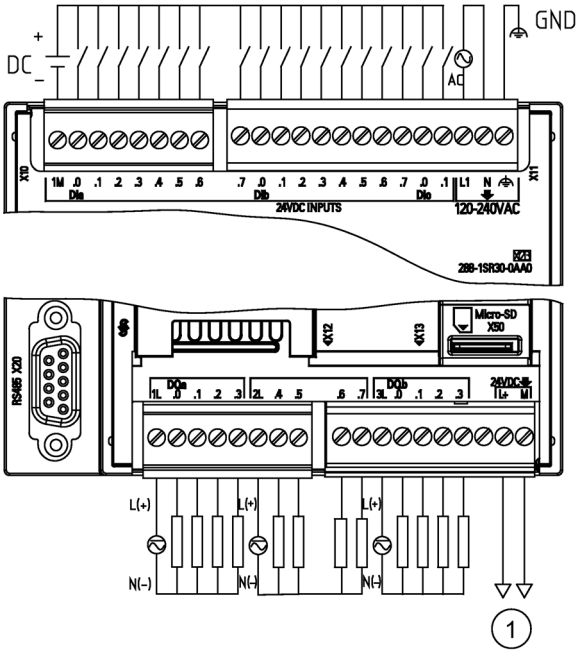
CPU ST30 DC/DC/DC (6ES7288-1ST30-0AA0)		
	①	24 V DC 传感器电源输出

表格 A- 31 CPU ST30 DC/DC/DC (6ES7288-1ST30-0AA0) 的连接器引脚位置

引脚	X10	X11	X12	X13
1	1M	DI a.7	2L+	DQ a.6
2	DI a.0	DI b.0	2M	DQ a.7
3	DI a.1	DI b.1	DQ a.0	3L+
4	DI a.2	DI b.2	DQ a.1	3M
5	DI a.3	DI b.3	DQ a.2	DQb.0
6	DI a.4	DIb.4	DQa.3	DQb.1
7	DI a.5	DIb.5	DQ a.4	DQb.2
8	DI a.6	DIb.6	DQ a.5	DQb.3
9	--	DIb.7	--	L+ / 24 V DC
10	--	DIc.0	--	M / 24 V DC

引脚	X10	X11	X12	X13
11	--	DIc.1	--	--
12	--	L+24 V DC	--	--
13	--	M / 24 V DC	--	--
14	--	功能性接地	--	--

表格 A- 32　CPU SR30 AC/DC/继电器 (6ES7288-1SR30-0AA0) 的接线图

CPU SR30 AC/DC/继电器 (6ES7288-1SR30-0AA0)		
	①	24 V DC 传感器电源输出

表格 A- 33　CPU SR30 AC/DC/继电器 (6ES7288-1SR30-0AA0) 的连接器引脚位置

引脚	X10	X11	X12	X13
1	1M	DI a.7	1L	DQ a.6
2	DI a.0	DI b.0	DQ a.0	DQ a.7
3	DI a.1	DI b.1	DQ a.1	3L
4	DI a.2	DI b.2	DQ a.2	DQ b.0
5	DI a.3	DI b.3	DQ a.3	DQ b.1
6	DI a.4	DIb.4	2L	DQ b.2
7	DI a.5	DIb.5	DQ a.4	DQ b.3
8	DI a.6	DIb.6	DQ a.5	--
9	--	DIb.7	--	L+ / 24 V DC 输出
10	--	DIc.0	--	M/24 V DC 输出
11	--	DIc.1	--	--
12	--	L1 / 120 - 240 V AC	--	--
13	--	N / 120 - 240 V AC	--	--
14	--	功能性接地	--	--

A.2.3 CPU ST40、CPU SR40 和 CPU CR40

A.2.3.1 常规规范和特性

CPU ST40、CPU SR40 和 CPU CR40 的常规规范和特性

表格 A- 34 常规规范

技术数据	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/继电器	CPU CR40 AC/DC/继电器
产品编号	6ES7288-1ST40-0AA0	6ES7288-1SR40-0AA0	6ES7288-1CR40-0AA0
尺寸 W x H x D (mm)	125 x 100 x 81	125 x 100 x 81	125 x 100 x 81
重量	410.3 g	441.3 g	440 g
功耗	18 W	23 W	18 W
可用电流（EM 总线）	最大 1400 mA(5 V DC)	最大 1400 mA(5 V DC)	--
可用电流 (24 V DC)	最大 300 mA（传感器电源）	最大 300 mA（传感器电源）	最大 300 mA（传感器电源）
数字量输入电流消耗 (24 V DC)	所用的每点输入 4 mA	所用的每点输入 4 mA	所用的每点输入 4 mA

表格 A- 35 CPU 特征

技术数据		说明	
		CPU ST40 DC/DC/DC， CPU SR40 AC/DC/继电器	CPU CR40 AC/DC/继电器
用户存储器	程序	24 KB	12 KB
	用户数据 (V)	16 KB	8 KB
	保持性	最大 10 KB [1]	最大 10 KB [1]
板载数字量 I/O		24 点输入/16 点输出	24 点输入/16 点输出
过程映像		256 位输入 (I)/256 位输出 (Q)	256 位输入 (I)/256 位输出 (Q)
模拟图像		56 个字的输入 (AI)/56 个字的输出 (AQ)	--
位存储器 (M)		256 位	256 位

技术数据	说明	
	CPU ST40 DC/DC/DC， CPU SR40 AC/DC/继电器	CPU CR40 AC/DC/继电器
临时（局部）存储器 (L)	主程序中 64 字节， 每个子例程和中断例程中 64 字节 采用 LAD 或 FBD 进行编程时为 60 字节（STEP 7-Micro/WIN 保留 4 字节）	主程序中 64 字节和每个子例程和中断例程中 64 字节 采用 LAD 或 FBD 进行编程时为 60 字节（STEP 7-Micro/WIN 保留 4 字节）
顺序控制继电器 (S)	256 位	256 位
扩展模块扩展	最多 6 个	--
信号板扩展	最多 1 个	--
高速计数器	共 4 个 • 4 个，每个 200 KHz，单相 • 2 个，每个 100 KHz，A/B 相	共 4 个 • 4 个，每个 100 KHz，单相 • 2 个，每个 50 KHz，A/B 相
脉冲输出 [2]	3 个，100 KHz	--
脉冲捕捉输入	14	14
循环中断	2 个，分辨率为 1 ms	2 个，分辨率为 1 ms
沿中断	4 个上升沿和 4 个下降沿（使用可选信号板时，各为 6 个）	4 个上升沿和 4 个下降沿
存储卡	microSDHC 卡（可选）	microSDHC 卡（可选）
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] 可组态 V 存储器、M 存储器、C 存储器的组态区（当前值），以及 T 存储器要保持的部分（保持性定时器上的当前值），最大可为最大指定量。

[2] 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- 36　性能

指令类型	**执行速度**
布尔运算	150 ns/指令
移动字	1.2 μs/指令
实数数学运算	3.6 μs/指令

表格 A- 37　所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） 中断例程：128（0 到 127）
	嵌套深度	从主程序：8 个子例程级别 从中断例程：4 个子例程级别
累加器	数量	4
定时器	类型/数量	非保持性（TON、TOF）：192 保持性 (TONR)：64
计数器	数量	256

表格 A- 38　通信

技术数据	**说明**
端口数	以太网：1 串行端口：1 (RS485) 附加串行端口：仅在 SR40/ST40 上 1 个（带有可选 RS232/485 信号板）
HMI 设备[1]	以太网：8 个连接 串行端口：每个端口 4 个连接
编程设备 (PG)	以太网：1 个连接
CPU (PUT/GET)	以太网：8 个客户端和 8 个服务器连接
开放式用户通信	以太网：8 个主动和 8 个被动连接
数据传输率	以太网：10/100 Mb/s RS485 系统协议：9600、19200 和 187500 b/s RS485 自由端口：1200 到 115200 b/s

技术数据	说明
隔离（外部信号与 PLC 逻辑侧）	以太网：变压器隔离，1500 V DC RS485：无
电缆类型	以太网：CAT5e 屏蔽电缆 RS485：PROFIBUS 网络电缆

表格 A- 39　电源

技术数据		CPU ST40 DC/DC/DC	CPU SR40 AC/DC/继电器	CPU CR40 AC/DC/继电器
电压范围		20.4 到 28.8 V DC	85 到 264 V AC	85 到 264 V AC
电源频率		--	47 到 63 Hz	47 到 63 Hz
输入电流（最大负载时）	仅 CPU	24 V DC 时 190 mA（无 300 mA 传感器驱动功率） 24 V DC 时 470 mA（带 300 mA 传感器驱动功率）	120 V AC 时 130 mA（无 300 mA 传感器驱动功率） 120 V 时 250 mA（带 300 mA 传感器驱动功率） 240 V AC 时 80 mA（无 300 mA 传感器驱动功率） 240 V AC 时 150 mA（带 300 mA 传感器驱动功率）	120 V AC 时 130 mA（无 300 mA 传感器驱动功率） 120 V 时 250 mA（带 300 mA 传感器驱动功率） 240 V AC 时 80 mA（无 300 mA 传感器驱动功率） 240 V AC 时 150 mA（带 300 mA 传感器驱动功率）
	具有所有扩展附件的 CPU	24 VDC 时 680 mA	120 V AC 时 300 mA 240 V AC 时 190 mA	--
浪涌电流（最大）		28.8 V DC 时 11.7 A	264 V AC 时 16.3 A	264 V AC 时 7.3 A
隔离（输入电源与逻辑侧）		--	1500 V AC	1500 V AC
漏地电流，交流线路对功能地		--	0.5 mA	0.5 mA
保持时间（掉电）		24 V DC 时 20 ms	120 V AC 时 30 ms 240 V AC 时 200 ms	120 V AC 时 50 ms 240 V AC 时 400 ms
内部保险丝，用户不可更换		3 A，250 V，慢速熔断	3 A，250 V，慢速熔断	3 A，250 V，慢速熔断

表格 A- 40　传感器电源

技术数据	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/继电器	CPU CR40 AC/DC/继电器
电压范围	20.4 到 28.8 V DC	20.4 到 28.8 V DC	20.4 到 28.8 V DC
额定输出电流（最大）	300 mA	300 mA	300 mA
最大波纹噪声 (<10 MHz)	< 1 V 峰峰值	< 1 V 峰峰值	< 1 V 峰峰值
隔离（CPU 逻辑侧与传感器电源）	未隔离	未隔离	未隔离

A.2.3.2　数字量输入和输出

表格 A- 41　数字量输入

技术数据	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/继电器	CPU CR40 AC/DC/继电器
输入点数	24	24	24
类型	漏型/源型（IEC 1 类漏型，I0.0 到 I0.3 除外）	漏型/源型（IEC 1 类漏型）	漏型/源型（IEC 1 类漏型）
额定电压	4 mA 时 24 V DC，额定值	4 mA 时 24 V DC，额定值	4 mA 时 24 V DC，额定值
允许的连续电压	30 V DC，最大值	30 V DC，最大值	30 V DC，最大值
浪涌电压	35 V DC，持续 0.5 s	35 V DC，持续 0.5 s	35 V DC，持续 0.5 s
逻辑 1 信号（最小）	I0.0 到 I0.3：8 mA 时 4 V DC 其它输入点：2.5 mA 时 15 V DC	2.5 mA 时 15 V DC	2.5 mA 时 15 V DC
逻辑 0 信号（最大）	I0.0 到 I0.3：1 mA 时 1 V DC 其它输入点：1 mA 时 5 V DC	1 mA 时 5 V DC	1 mA 时 5 V DC
隔离（现场侧与逻辑侧）	500 V AC，持续1分钟	500 V AC，持续1分钟	500 V AC，持续1分钟
隔离组	1	1	1

技术数据	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/继电器	CPU CR40 AC/DC/继电器
滤波时间	每个通道上可单独选择（点 I0.0 到 I1.5）： 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	每个通道上可单独选择（点 I0.0 到 I1.5）： 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	每个通道上可单独选择（点 I0.0 到 I1.5）： 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
	每个通道上可单独选择（点 I1.6 和更大）： ms：0、6.4、12.8	每个通道上可单独选择（点 I1.6 和更大）： ms：0、6.4、12.8	每个通道上可单独选择（点 I1.6 和更大）： ms：0、6.4、12.8
HSC 时钟输入频率（最大）（逻辑 1 电平 = 15 到 26 V DC）	4 个 HSC，每个 200 KHz，单相 2 个 HSC，每个 100 KHz，A/B 相	4 个 HSC，每个 200 KHz，单相 2 个 HSC，每个 100 KHz，A/B 相	4 个 HSC，每个 100 KHz，单相 2 个 HSC，每个 50 KHz，A/B 相
同时接通的输入数	24	24	24
电缆长度（最大值），以米为单位	I0.0 到 I0.3： 屏蔽（仅限此类）： • 500 m 正常（低速）输入 • 50 m HSC（高速）输入	所有输入： 屏蔽： • 500 m 正常输入 • 50 m HSC 输入 非屏蔽： • 300 m 正常输入	所有输入： 屏蔽： • 500 m 正常输入 • 50 m HSC 输入 非屏蔽： • 300 m 正常输入
	所有其它输入： 屏蔽： • 500 m 正常输入 非屏蔽： • 300 m 正常输入		

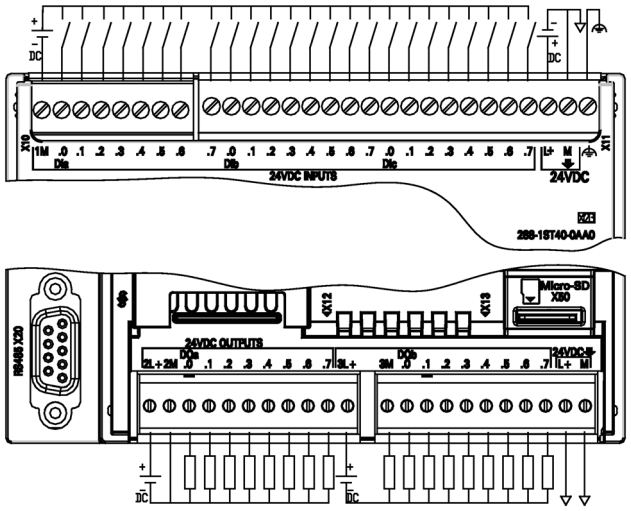
表格 A- 42　数字量输出

技术数据	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/继电器	CPU CR40 AC/DC/继电器
输出点数	16	16	16
类型	固态 - MOSFET（源型）	继电器，干触点	继电器，干触点
电压范围	20.4 到 28.8 V DC	5 到 30 V DC 或 5 到 250 V AC	5 到 30 V DC 或 5 到 250 V AC
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	2 A	2 A
每个公共端的额定电流（最大）	6 A	10 A	10 A
灯负载	5 W	30 W DC/200 W AC	30 W DC/200 W AC
通态电阻	最大 0.6 Ω	新设备最大为 0.2 Ω	新设备最大为 0.2 Ω
每点的漏电流	最大 10 μA	--	--
浪涌电流	8 A，最长持续 100 ms	触点闭合时为 7 A	触点闭合时为 7 A
过载保护	无	无	无
隔离（现场侧与逻辑侧）	500 V AC，持续1分钟	1500 V AC，持续1分钟（线圈与触点） 无（线圈与逻辑侧）	1500 V AC，持续1分钟（线圈与触点） 无（线圈与逻辑侧）
隔离电阻	--	新设备最小为 100 MΩ	新设备最小为 100 MΩ
断开触点间的绝缘	--	750 V AC，持续1分钟	750 V AC，持续1分钟
隔离组	2	4	4
电感钳位电压	L+ - 48 V DC，1 W 损耗	--	--
开关延迟（Qa.0 到 Qa.3）	断开到接通最长为 1.0 μs 接通到断开最长为 3.0 μs	最长 10 ms	最长 10 ms

技术数据	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/继电器	CPU CR40 AC/DC/继电器
开关延迟（Qa.4 到 Qb.7）	断开到接通最长为 50 µs 接通到断开最长为 200 µs	最长 10 ms	最长 10 ms
机械寿命（无负载）	--	10,000,000 个断开/闭合周期	10,000,000 个断开/闭合周期
额定负载下的触点寿命	--	100,000 个断开/闭合周期	100,000 个断开/闭合周期
STOP 模式下的输出状态	上一个值或替换值（默认值为 0）	上一个值或替换值（默认值为 0）	上一个值或替换值（默认值为 0）
同时接通的输出数	16	16	16
电缆长度（最大值），以米为单位	屏蔽：500 m 非屏蔽：150 m	屏蔽：500 m 非屏蔽：150 m	屏蔽：500 m 非屏蔽：150 m

A.2.3.3 CPU ST40、SR40 和 CR40 接线图

表格 A- 43　CPU ST40 DC/DC/DC (6ES7288-1ST40-0AA0) 的接线图

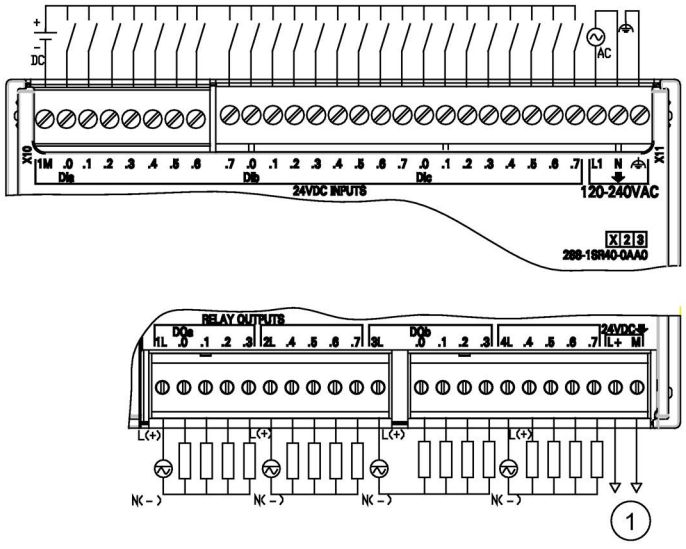
	①	24 V DC 传感器电源输出

表格 A- 44　CPU ST40 DC/DC/DC (6ES7288-1ST40-0AA0) 的连接器引脚位置

引脚	X10	X11	X12	X13
1	1M	DI a.7	2L+	3M
2	DI a.0	DI b.0	2M	DQ b.0
3	DI a.1	DI b.1	DQ a.0	DQ b.1
4	DI a.2	DI b.2	DQ a.1	DQ b.2
5	DI a.3	DI b.3	DQ a.2	DQ b.3
6	DI a.4	DI b.4	DQ a.3	DQ b.4
7	DI a.5	DI b.5	DQ a.4	DQ b.5
8	DI a.6	DI b.6	DQ a.5	DQ b.6
9	--	DI b.7	DQ a.6	DQ b.7
10	--	DI c.0	DQ a.7	L+ / 24 V DC 输出
11	--	DI c.1	3L+	M/24 V DC 输出
12	--	DI c.2	--	--
13	--	DI c.3	--	--

引脚	X10	X11	X12	X13
14	--	DI c.4	--	--
15	--	DI c.5	--	--
16	--	DI c.6	--	--
17	--	DI c.7	--	--
18	--	L+ / 24 V DC	--	--
19	--	M / 24 V DC	--	--
20	--	功能性接地	--	--

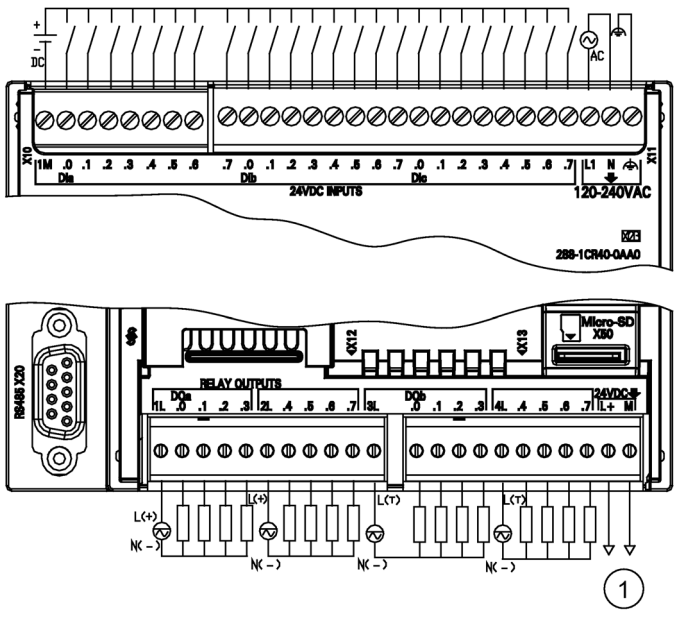
表格 A- 45　CPU SR40 AC/DC/继电器 (6ES7288-1SR40-0AA0) 的接线图

	①	24 V DC 传感器电源输出

表格 A- 46　CPU SR40 AC/DC/继电器 (6ES7288-1SR40-0AA0) 的连接器引脚位置

引脚	X10	X11	X12	X13
1	1M	DI a.7	1L	DQ b.0
2	DI a.0	DI b.0	DQ a.0	DQ b.1
3	DI a.1	DI b.1	DQ a.1	DQ b.2
4	DI a.2	DI b.2	DQ a.2	DQ b.3
5	DI a.3	DI b.3	DQ a.3	4L
6	DI a.4	DI b.4	2L	DQ b.4
7	DI a.5	DI b.5	DQ a.4	DQ b.5
8	DI a.6	DI b.6	DQ a.5	DQ b.6
9	--	DI b.7	DQ a.6	DQ b.7
10	--	DI c.0	DQ a.7	L+ / 24 V DC 输出
11	--	DI c.1	3L	M/24 V DC 输出
12	--	DI c.2	--	--
13	--	DI c.3	--	--
14	--	DI c.4	--	--
15	--	DI c.5	--	--
16	--	DI c.6	--	--
17	--	DI c.7	--	--
18	--	L1 / 120 - 240 V AC	--	--
19	--	N / 120 - 240 V AC	--	--
20	--	功能性接地	--	--

表格 A- 47　CPU CR40 AC/DC/继电器 (6ES7288-1CR40-0AA0) 的接线图

		① 24 V DC 传感器电源输出

表格 A- 48　CPU CR40 AC/DC/继电器 (6ES7288-1CR40-0AA0) 的连接器引脚位置

引脚	X10	X11	X12	X13
1	1M	DI a.7	1L	DQ b.0
2	DI a.0	DI b.0	DQ a.0	DQ b.1
3	DI a.1	DI b.1	DQ a.1	DQ b.2
4	DI a.2	DI b.2	DQ a.2	DQ b.3
5	DI a.3	DI b.3	DQ a.3	4L
6	DI a.4	DI b.4	2L	DQ b.4
7	DI a.5	DI b.5	DQ a.4	DQ b.5
8	DI a.6	DI b.6	DQ a.5	DQ b.6
9	--	DI b.7	DQ a.6	DQ b.7
10	--	DI c.0	DQ a.7	L+ / 24 V DC 输出
11	--	DI c.1	3L	M/24 V DC 输出
12	--	DI c.2	--	--

引脚	X10	X11	X12	X13
13	--	DI c.3	--	--
14	--	DI c.4	--	--
15	--	DI c.5	--	--
16	--	DI c.6	--	--
17	--	DI c.7	--	--
18	--	L1 / 120 - 240 V AC	--	--
19	--	N / 120 - 240 V AC	--	--
20	--	功能性接地	--	--

A.2.4 CPU ST60、CPU SR60 和 CPU CR60

A.2.4.1 常规规范和特性

表格 A- 49 常规规范

技术数据	CPU ST60 DC/DC/DC	CPU SR60 AC/DC/继电器	CPU CR60 AC/DC/继电器
产品编号	6ES7288-1ST60-0AA0	6ES7288-1SR60-0AA0	6ES7288-1CR60-0AA0
尺寸 W x H x D (mm)	175 x 100 x 81	175 x 100 x 81	175 x 100 x 81
重量	528.2 g	611.5 g	620 g
功耗	20 W	25 W	20 W
可用电流（EM 总线）	最大 1400 mA(5 V DC)	最大 1400 mA(5 V DC)	--
可用电流 (24 V DC)	最大 300 mA（传感器电源）	最大 300 mA（传感器电源）	最大 300 mA（传感器电源）
数字量输入电流消耗 (24 V DC)	所用的每点输入 4 mA	所用的每点输入 4 mA	所用的每点输入 4 mA

表格 A- 50 CPU 特征

技术数据		说明	
		CPU ST60、CPU SR60	CPU CR60 AC/DC/继电器
用户存储器	程序	30 KB	12 KB
	用户数据 (V)	20 KB	8 KB
	保持性	最大 10 KB [1]	最大 10 KB [1]
板载数字量 I/O		36 点输入/24 点输出	36 点输入/24 点输出
过程映像		256 位输入 (I)/256 位输出 (Q)	256 位输入 (I)/256 位输出 (Q)
模拟图像		56 个字的输入 (AI)/56 个字的输出 (AQ)	--
位存储器 (M)		256 位	256 位

技术数据	说明	
	CPU ST60、CPU SR60	CPU CR60 AC/DC/继电器
临时（局部）存储器 (L)	主程序中 64 字节和每个子例程和中断例程中 64 字节 采用 LAD 或 FBD 进行编程时为 60 字节（STEP 7-Micro/WIN 保留 4 字节）	主程序中 64 字节和每个子例程和中断例程中 64 字节 采用 LAD 或 FBD 进行编程时为 60 字节（STEP 7-Micro/WIN 保留 4 字节）
顺序控制继电器 (S)	256 位	256 位
扩展模块扩展	最多 6 个	--
信号板扩展	最多 1 个	--
高速计数器	共 4 个 • 4 个，每个 200 KHz，单相 • 2 个，每个 100 KHz，A/B 相	共 4 个 • 4 个，每个 100 KHz，单相 • 2 个，每个 50 KHz，A/B 相
脉冲输出 [2]	3 个，100 KHz	--
脉冲捕捉输入	14	14
循环中断	2 个，分辨率为 1 ms	2 个，分辨率为 1 ms
沿中断	4 个上升沿和 4 个下降沿（使用可选信号板时，各为 6 个）	4 个上升沿和 4 个下降沿
存储卡	microSDHC 卡（可选）	microSDHC 卡（可选）
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 可组态 V 存储器、M 存储器、C 存储器的存储区（当前值），以及 T 存储器要保持的部分（保持性定时器上的当前值），最大可为最大指定量。

2 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- 51　性能

指令类型	执行速度
布尔运算	150 ns/指令
移动字	1.2 µs/指令
实数数学运算	3.6 µs/指令

表格 A- 52　所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） 中断例程：128（0 到 127）
	嵌套深度	从主程序：8 个子例程级别 从中断例程：4 个子例程级别
累加器	数量	4
定时器	类型/数量	非保持性（TON、TOF）：192 保持性 (TONR)：64
计数器	数量	256

表格 A- 53　通信

技术数据	说明
端口数	以太网：1 串行端口：1 (RS485) 附加串行端口：1（带有可选 RS232/485 信号板）
HMI 设备	以太网：8 个连接 串行端口：每个端口 4 个连接
编程设备 (PG)	以太网：1 个连接
CPU (PUT/GET)	以太网：8 个客户端和 8 个服务器连接
开放式用户通信	以太网：8 个主动和 8 个被动连接
数据传输率	以太网：10/100 Mb/s RS485 系统协议：9600、19200 和 187500 b/s RS485 自由端口：1200 到 115200 b/s

技术数据	说明
隔离（外部信号与 PLC 逻辑侧）	以太网：变压器隔离，1500 V DC RS485：无
电缆类型	以太网：CAT5e 屏蔽电缆 RS485：PROFIBUS 网络电缆

表格 A- 54　电源

技术数据		CPU ST60 DC/DC/DC	CPU SR60 AC/DC/继电器	CPU CR60 AC/DC/继电器
电压范围		20.4 到 28.8 V DC	85 到 264 V AC	85 到 264 V AC
电源频率		--	47 到 63 Hz	47 到 63 Hz
输入电流（最大负载时）	仅 CPU	24 V DC 时 220 mA（无 300 mA 传感器驱动功率） 24 V DC 时 500 mA（带 300 mA 传感器驱动功率）	120 V AC 时 160 mA（无 300 mA 传感器驱动功率） 120 V AC 时 280 mA（带 300 mA 传感器驱动功率） 240 V AC 时 90 mA（无 300 mA 传感器驱动功率） 240 V AC 时 160 mA（带 300 mA 传感器驱动功率）	120 V AC 时 160 mA（无 300 mA 传感器驱动功率） 120 V AC 时 280 mA（带 300 mA 传感器驱动功率） 240 V AC 时 90 mA（无 300 mA 传感器驱动功率） 240 V AC 时 160 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 710 mA	120 V AC 时 370 mA 240 V AC 时 220 mA	--
浪涌电流（最大）		28.8 V DC 时 11.5 A	264 V DC 时 16.3 A	264 V AC 时 7.3 A
隔离（输入电源与逻辑侧）		无	1500 V AC	1500 V AC
漏地电流，交流线路对功能地		无	无	无
保持时间（掉电）		24 V DC 时 20 ms	120 V AC 时 30 ms 240 V AC 时 200 ms	120 V AC 时 50 ms 240 V AC 时 400 ms
内部保险丝，用户不可更换		3 A，250 V，慢速熔断	3 A，250 V，慢速熔断	3 A，250 V，慢速熔断

表格 A- 55　传感器电源

技术数据	CPU ST60 DC/DC/DC	CPU SR60 AC/DC/继电器	CPU CR 60 AC/DC/继电器
电压范围	20.4 到 28.8 V DC	20.4 到 28.8 V DC	20.4 到 28.8 V DC
额定输出电流（最大）	300 mA	300 mA	300 mA
最大波纹噪声 (<10 MHz)	< 1 V 峰峰值	< 1 V 峰峰值	< 1 V 峰峰值
隔离（CPU 逻辑侧与传感器电源）	未隔离	未隔离	未隔离

A.2.4.2　数字量输入和输出

表格 A- 56　数字量输入

技术数据	CPU ST60 DC/DC/DC	CPU SR60 AC/DC/继电器	CPU CR60 AC/DC/继电器
输入点数	36	36	36
类型	漏型/源型（IEC 1 类漏型，I0.0 到 I0.3 除外）	漏型/源型（IEC 1 类漏型）	漏型/源型（IEC 1 类漏型）
额定电压	4 mA 时 24 V DC，额定值	4 mA 时 24 V DC，额定值	4 mA 时 24 V DC，额定值
允许的连续电压	30 V DC，最大值	30 V DC，最大值	30 V DC，最大值
浪涌电压	35 V DC，持续 0.5 s	35 V DC，持续 0.5 s	35 V DC，持续 0.5 s
逻辑 1 信号（最小）	I0.0 到 I0.3：8 mA 时 4 V DC 其它输入点：2.5 mA 时 15 V DC	2.5 mA 时 15 V DC	2.5 mA 时 15 V DC
逻辑 0 信号（最大）	I0.0 到 I0.3：1 mA 时 1 V DC 其它输入点：1 mA 时 5 V DC	1 mA 时 5 V DC	1 mA 时 5 V DC
隔离（现场侧与逻辑侧）	500 V AC，持续1分钟	500 V AC，持续1分钟	500 V AC，持续1分钟
隔离组	1	1	1

技术数据	CPU ST60 DC/DC/DC	CPU SR60 AC/DC/继电器	CPU CR60 AC/DC/继电器
滤波时间	每个通道上可单独选择（点 I0.0 到 I1.5）： 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	每个通道上可单独选择（点 I0.0 到 I1.5）： 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	每个通道上可单独选择（点 I0.0 到 I1.5）： 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
	每个通道上可单独选择（点 I1.6 和更大）： ms：0、6.4、12.8	每个通道上可单独选择（点 I1.6 和更大）： ms：0、6.4、12.8	每个通道上可单独选择（点 I1.6 和更大）： ms：0、6.4、12.8
HSC 时钟输入频率（最大）（逻辑 1 电平 = 15 到 26 V DC）	4 个 HSC，每个 200 KHz，单相 2 个 HSC，每个 100 KHz，A/B 相	4 个 HSC，每个 200 KHz，单相 2 个 HSC，每个 100 KHz，A/B 相	4 个 HSC，每个 100 KHz，单相 2 个 HSC，每个 50 KHz，A/B 相
同时接通的输入数	36	36	36
电缆长度（最大值），以米为单位	I0.0 到 I0.3： 屏蔽（仅限此类）： • 500 m 正常（低速）输入 • 50 m HSC（高速）输入	所有输入： 屏蔽： • 500 m 正常输入 • 50 m HSC 输入 非屏蔽： • 300 m 正常输入	所有输入： 屏蔽： • 500 m 正常输入 • 50 m HSC 输入 非屏蔽： • 300 m 正常输入
	所有其它输入： 屏蔽： • 500 m 正常输入 非屏蔽： • 300 m 正常输入		

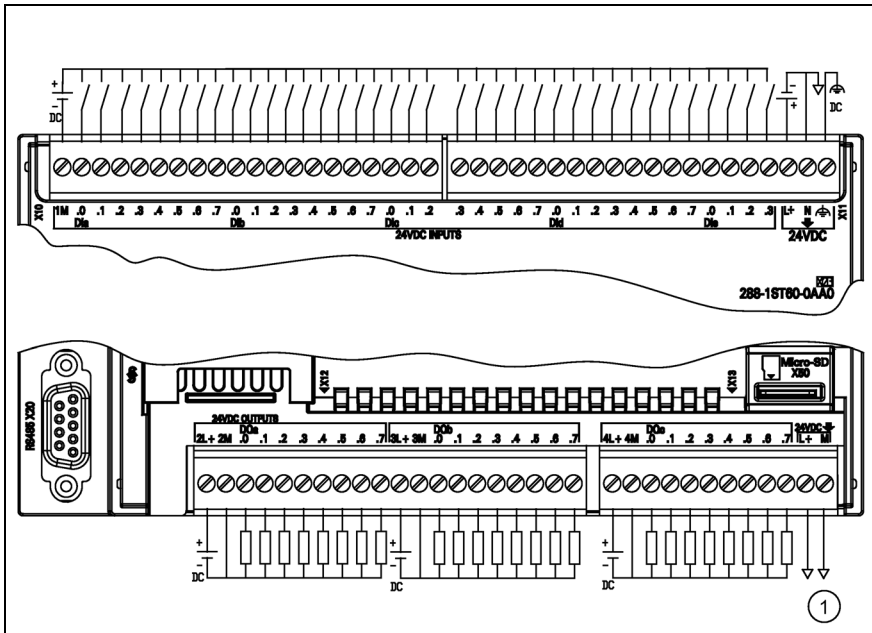
表格 A- 57　数字量输出

技术数据	CPU ST60 DC/DC/DC	CPU SR60 AC/DC/继电器	CPU CR60 AC/DC/继电器
输出点数	24	24	24
类型	固态 - MOSFET（源型）	继电器，干触点	继电器，干触点
电压范围	20.4 到 28.8 V DC	5 到 30 V DC 或 5 到 250 V AC	5 到 30 V DC 或 5 到 250 V AC
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	2 A	2 A
每个公共端的额定电流（最大）	6 A	10 A	10 A
灯负载	5 W	30 W DC/200 W AC	30 W DC/200 W AC
通态电阻	最大 0.6 Ω	新设备最大为 0.2 Ω	新设备最大为 0.2 Ω
每点的漏电流	最大 10 μA	--	--
浪涌电流	8 A，最长持续 100 ms	触点闭合时为 7 A	触点闭合时为 7 A
过载保护	无	无	无
隔离（现场侧与逻辑侧）	500 V AC，持续1分钟	1500 V AC，持续1分钟（线圈与触点）无（线圈与逻辑侧）	1500 V AC，持续1分钟（线圈与触点）无（线圈与逻辑侧）
隔离电阻	--	新设备最小为 100 MΩ	新设备最小为 100 MΩ
断开触点间的绝缘	--	750 V AC，持续1分钟	750 V AC，持续1分钟
隔离组	3	6	6
电感钳位电压	L+ - 48 V DC，1 W 损耗	--	-
开关延迟（Qa.0 到 Qa.3）	断开到接通最长为 1.0 μs 接通到断开最长为 3.0 μs	最长 10 ms	最长 10 ms

技术数据	CPU ST60 DC/DC/DC	CPU SR60 AC/DC/继电器	CPU CR60 AC/DC/继电器
开关延迟（Qa.4 到 Qc.7）	断开到接通最长为 50 µs 接通到断开最长为 200 µs	最长 10 ms	最长 10 ms
机械寿命（无负载）	--	10,000,000 个断开/闭合周期	10,000,000 个断开/闭合周期
额定负载下的触点寿命	--	100,000 个断开/闭合周期	100,000 个断开/闭合周期
STOP 模式下的输出行为	上一个值或替换值（默认值为 0）	上一个值或替换值（默认值为 0）	上一个值或替换值（默认值为 0）
同时接通的输出数	24	24	24
电缆长度（最大值），以米为单位	屏蔽：500 m 非屏蔽：150 m	屏蔽：500 m 非屏蔽：150 m	屏蔽：500 m 非屏蔽：150 m

A.2.4.3 CPU ST60、SR60 和 CR60 接线图

表格 A- 58 CPU ST60 DC/DC/DC (6ES7288-1ST60-0AA0) 的接线图

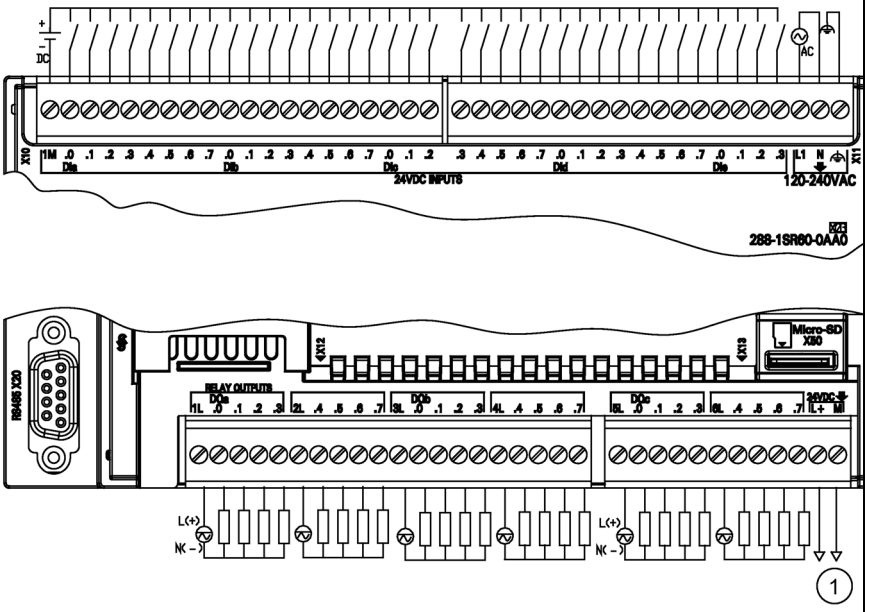
	①	24 V DC 传感器电源输出

表格 A- 59 CPU ST60 DC/DC/DC (6ES7288-1ST60-0AA0) 的连接器引脚位置

引脚	X10	X11	X12	X13
1	1M	DI c.3	2L+	4L+
2	DI a.0	DI c.4	2M	4M
3	DI a.1	DI c.5	DQ a.0	DQ c.0
4	DI a.2	DI c.6	DQ a.1	DQ c.1
5	DI a.3	DI c.7	DQ a.2	DQ c.2
6	DI a.4	DI d.0	DQ a.3	DQ c.3
7	DI a.5	DI d.1	DQ a.4	DQ c.4
8	DI a.6	DI d.2	DQ a.5	DQ c.5
9	DI a.7	DI d.3	DQ a.6	DQ c.6
10	DI b.0	DI d.4	DQ a.7	DQ c.7
11	DI b.1	DI d.5	3L+	L+ / 24 V DC 输出

引脚	X10	X11	X12	X13
12	DI b.2	DI d.6	3M	M/24 V DC 输出
13	DI b.3	DI d.7	DQ b.0	--
14	DI b.4	DI e.0	DQ b.1	--
15	DI b.5	DI e.1	DQ b.2	--
16	DI b.6	DI e.2	DQ b.3	--
17	DI b.7	DI e.3	DQ b.4	--
18	DI c.0	L+ / 24 V DC	DQ b.5	--
19	DI c.1	M / 24 V DC	DQ b.6	--
20	DI c.2	功能性接地	DQ b.7	--

表格 A- 60　CPU SR60 AC/DC/继电器 (6ES7288-1SR60-0AA0) 的接线图

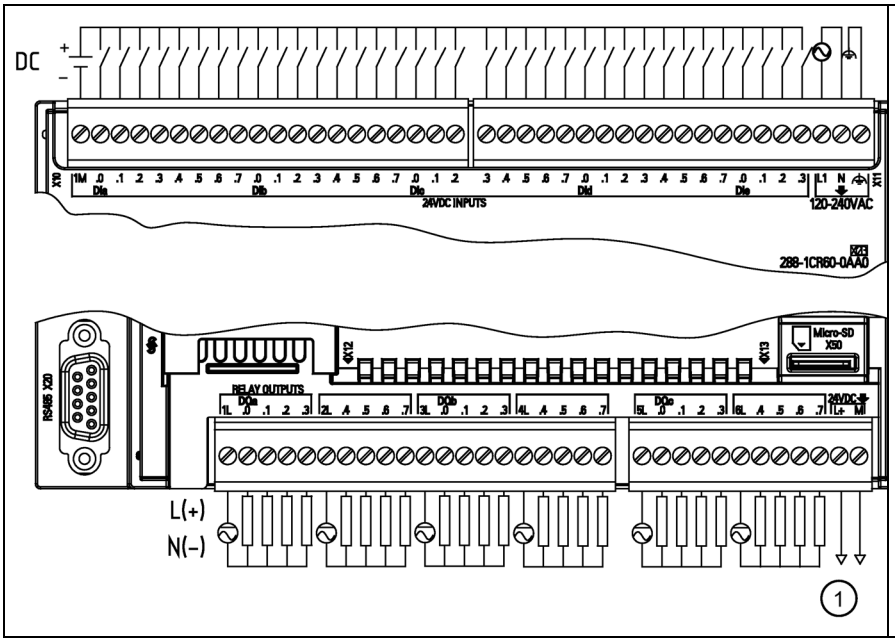


① 24 V DC 传感器电源输出

表格 A- 61　CPU SR60 AC/DC/继电器 (6ES7288-1SR60-0AA0) 的连接器引脚位置

引脚	X10	X11	X12	X13
1	1M	DI c.3	1L	5L
2	DI a.0	DI c.4	DQ a.0	DQ c.0
3	DI a.1	DI c.5	DQ a.1	DQ c.1
4	DI a.2	DI c.6	DQ a.2	DQ c.2
5	DI a.3	DI c.7	DQ a.3	DQ c.3
6	DI a.4	DI d.0	2L	6L
7	DI a.5	DI d.1	DQ a.4	DQ c.4
8	DI a.6	DI d.2	DQ a.5	DQ c.5
9	DI a.7	DI d.3	DQ a.6	DQ c.6
10	DI b.0	DI d.4	DQ a.7	DQ c.7
11	DI b.1	DI d.5	3L	L+ / 24 V DC 输出
12	DI b.2	DI d.6	DQ b.0	M/24 V DC 输出
13	DI b.3	DI d.7	DQ b.1	--
14	DI b.4	DI e.0	DQ b.2	--
15	DI b.5	DI e.1	DQ b.3	--
16	DI b.6	DI e.2	4L	--
17	DI b.7	DI e.3	DQ b.4	--
18	DI c.0	L1 / 120 - 240 V AC	DQ b.5	--
19	DI c.1	N / 120 - 240 V AC	DQ b.6	--
20	DI c.2	功能性接地	DQ b.7	--

表格 A- 62　CPU CR60 AC/DC/继电器 (6ES7288-1CR60-0AA0) 的接线图



① 24 V DC 传感器电源输出

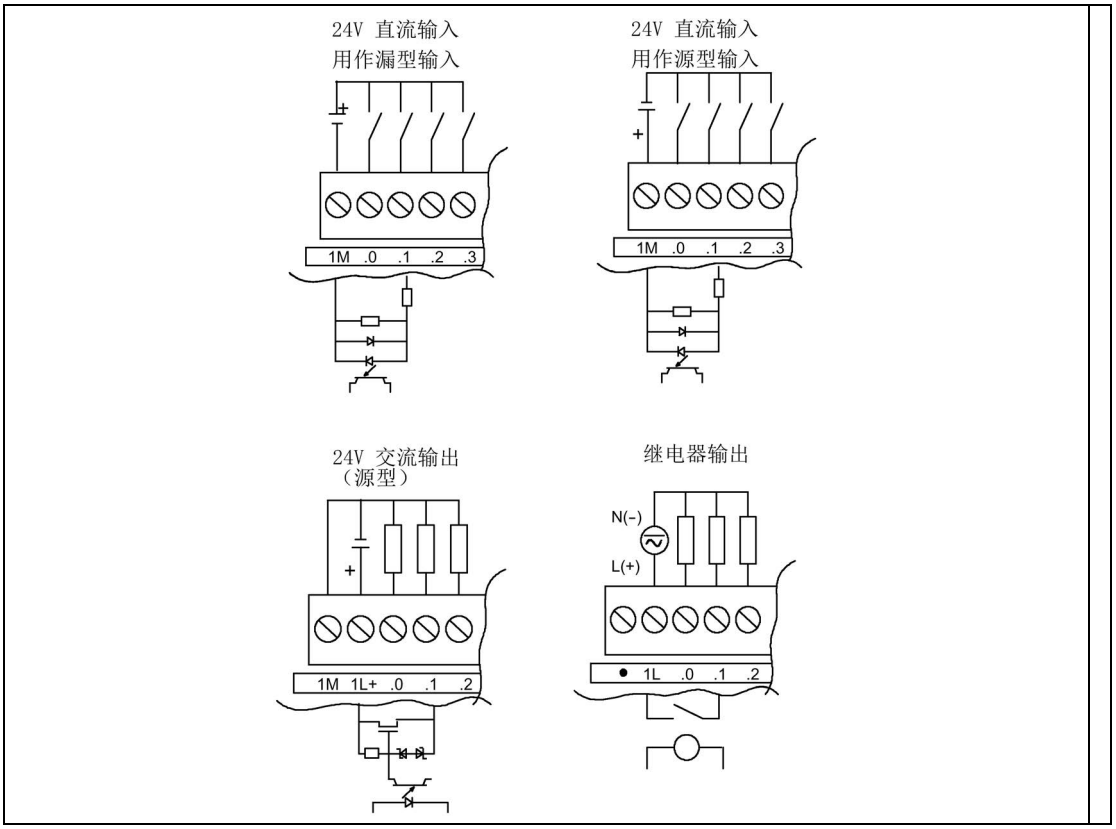
表格 A- 63　CPU CR60 AC/DC/继电器 (6ES7288-1CR60-0AA0) 的连接器引脚位置

引脚	X10	X11	X12	X13
1	1M	DI c.3	1L	5L
2	DI a.0	DI c.4	DQ1.0	DQc.0
3	DI a.1	DI c.5	DQ a.1	DQ c.1
4	DI a.2	DI c.6	DQ a.2	DQ c.2
5	DI a.3	DI c.7	DQ a.3	DQ c.3
6	DI a.4	DI d.0	2L	6L
7	DI a.5	DI d.1	DQ a.4	DQ c.4
8	DI a.6	DI d.2	DQ a.5	DQ c.5
9	DI a.7	DI d.3	DQ a.6	DQ c.6
10	DI b.0	DI d.4	DQ a.7	DQ c.7
11	DI b.1	DI d.5	3L	L+ / 24 V DC 输出
12	DI b.2	DI d.6	DQb.0	M/24 V DC 输出
13	DI b.3	DI d.7	DQ b.1	--
14	DI b.4	DI e.0	DQ b.2	--

引脚	X10	X11	X12	X13
15	DI b.5	DI e.1	DQ b.3	--
16	DI b.6	DI e.2	4L	--
17	DI b.7	DI e.3	DQ b.4	--
18	DI c.0	L1/120-240 V AC	DQ b.5	--
19	DI c.1	N/120-240 V AC	DQ b.6	--
20	DI c.2	功能性接地	DQ b.7	--

A.2.5 漏型、源型输入和继电器输出的接线图

表格 A- 64 漏型输入、源型输入和继电器输出的接线图



A.3 数字量输入和输出扩展模块 (EM)

A.3.1 EM DE08 和 EM DE16 数字量输入规范

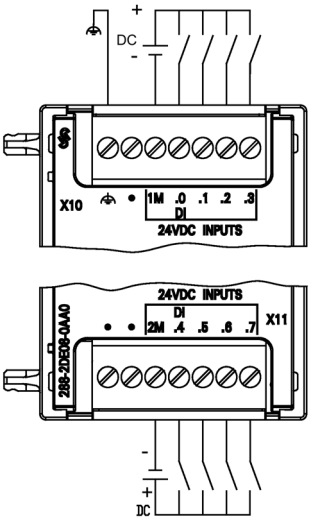
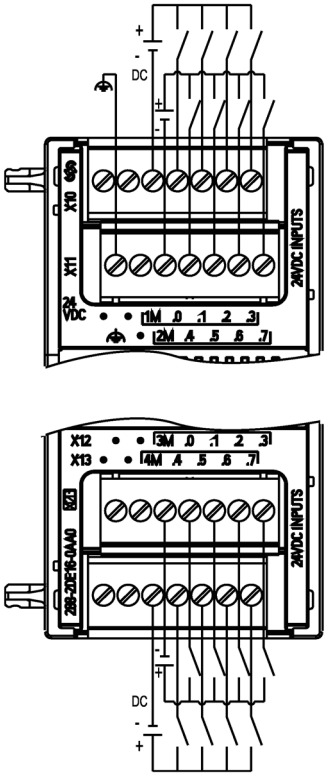
表格 A- 65 常规规范

型号	EM 8 点数字量输入 (EM DT08)	EM 16 点数字量输入 (EM DE16)
产品编号	6ES7288-2DE08-0AA0	6ES7288-2DE16-0AA0
尺寸 W x H x D (mm)	45 x 100 x 81	45 x 100 x 81
重量	141.4 g	176 g
功耗	1.5 W	2.3 W
电流消耗（SM 总线）	105 mA	105 mA
电流消耗 (24 V DC)	所用的每点输入 4 mA	所用的每点输入 4 mA

表格 A- 66 数字量输入

型号	EM 8 点数字量输入 (EM DT08)	EM 16 点数字量输入 (EM DE16)
输入点数	8	16
类型	漏型/源型（IEC 1 类漏型）	漏型/源型（IEC 1 类漏型）
额定电压	4 mA 时 24 V DC，额定值	4 mA 时 24 V DC，额定值
允许的连续电压	30 V DC，最大值	30 V DC，最大值
浪涌电压	35 V DC，持续 0.5 s	35 V DC，持续 0.5 s
逻辑 1 信号（最小）	2.5 mA 时 15 V DC	2.5 mA 时 15 V DC
逻辑 0 信号（最大）	1 mA 时 5 V DC	1 mA 时 5 V DC
隔离（现场侧与逻辑侧）	500 V AC，持续1分钟	500 V AC，持续 1分钟
隔离组	2	4
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）	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	8	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	屏蔽：500 m 正常输入 非屏蔽：300 m 正常输入

表格 A- 67　EM DE08 8 点数字量输入 (6ES7288-2DE08-0AA0) 和 EM DE16 16 点数字量输入 (6ES7288-2DE16-0AA0) 的接线图

EM DE08 8 点数字量输入 (6ES7288-2DE08-0AA0)	EM DE16 16 点数字量输入 (6ES7288-2DE16-0AA0)
	

表格 A- 68　EM DE08 8 点数字量输入 (6ES7288-2DE08-0AA0) 的连接器引脚位置

引脚	X10	X11
1	功能性接地	无连接
2	无连接	无连接
3	1M	2M
4	DI a.0	DI a.4
5	DI a.1	DI a.5
6	DI a.2	DI a.6
7	DI a.3	DI a.7

表格 A- 69　EM DE16 16 点数字量输入 (6ES7288-2DE16-0AA0) 的连接器引脚位置

引脚	X10	X11	X12	X13
1	无连接	功能性接地	无连接	无连接
2	无连接	无连接	无连接	无连接
3	1M	2M	3M	4M
4	DI a.0	DI a.4	DI b.0	DI b.4
5	DI a.1	DI a.5	DI b.1	DI b.5
6	DI a.2	DI a.6	DI b.2	DI b.6
7	DI a.3	DI a.7	DI b.3	DI b.7

A.3.2 EM DT08、EM DR08、EM QR16 和 EM QT16 数字量输出规范

表格 A- 70 常规规范

型号	EM 8 点 数字量输出 (EM DT08)	EM 8 点继电器型 数字量输出 (EM DR08)	EM 16 点继电器型 数字量输出 (EM QR16)	EM 16 点晶体管型 数字量输出 (EM QT16)
产品编号	6ES7288-2DT08-0AA0	6ES7288-2DR08-0AA0	6ES7288-2QR16-0AA0	6ES7288-2QT16-0AA0
尺寸 W x H x D (mm)	45 x 100 x 81	45 x 100 x 81	45 x 100 x 81	45 x 100 x 81
重量	147 g	166.3 g	221 g	186 g
功耗	1.5 W	4.5 W	4.5 W	1.7 W
电流消耗（SM 总线）	120 mA	120 mA	110 mA，SM 总线	120 mA，SM 总线
电流消耗 (24 V DC)	--	11 mA/使用的继电器线圈	150 mA，所有继电器开启	50 mA

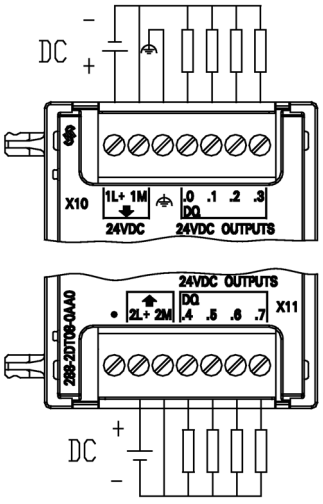
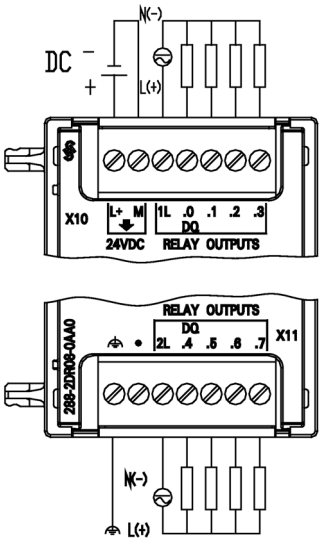
表格 A- 71 数字量输出

型号	EM 8 点 数字量输出 (EM DT08)	EM 8 点继电器型 数字量输出 (EM DR08)	EM 16 点继电器型 数字量输出 (EM QR16)	EM 16 点晶体管型 数字量输出 (EM QT16)
输出点数	8	8	16	16
类型	固态 - MOSFET（源型）	继电器，干触点	继电器，干触点	固态 - MOSFET（源型）
电压范围	20.4 到 28.8 V DC	5 到 30 V DC 或 5 到 250 V AC	5 到 30 V DC 或 5 到 250 V AC	20.4 到 28.8 V DC
最大电流时的逻辑 1 信号	20 V DC	-	-	20 V DC
具有 10 KΩ 负载时的逻辑 0 信号	0.1 V DC	-	-	0.1 V DC
每点的额定电流（最大）	0.75 A	2.0 A	2.0 A	0.75 A

型号	EM 8 点 数字量输出 (EM DT08)	EM 8 点继电器型 数字量输出 (EM DR08)	EM 16 点继电器型 数字量输出 (EM QR16)	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）	3 A	8 A	8 A	3 A
灯负载	5 W DC	30 W DC/200 W AC	30 W DC/200 W AC	5 W
通态触点电阻	0.6 Ω	新设备最大为 0.2 Ω	新设备最大为 0.2 Ω	最大 0.6 Ω
每点的漏电流	10 µA	--	--	10 µA
浪涌电流	8 A，最长持续 100 ms	触点闭合时为 7 A	触点闭合时为 7 A	8 A，最长持续 100 ms
过载保护	无	无	否	无
隔离（现场侧与逻辑侧）	光隔离，500 V AC，持续 1分钟	1500 V AC，持续 1 分钟（线圈与触点） 无（线圈与逻辑侧）	1500 V AC，持续 1 分钟（线圈与触点） 无（线圈与逻辑侧）	500 V AC，持续 1分钟
隔离电阻	-	新设备最小为 100 MΩ	新设备最小为 100 MΩ	-
断开触点间的绝缘	-	750 V AC，持续1分钟	750 V AC，持续 1 分钟	-
隔离组	2	2	4	4
电感钳位电压	- 48 V DC，1 W 损耗	-	-	L+ - 48 V，1 W 损耗
开关延时	接通延时小于 50 µs，断开延时小于 200 µS	最长 10 ms	最长 10 ms	接通延时小于 50 µs，断开延时小于 200 µS
机械寿命（无负载）	--	10,000,000 个断开/闭合周期	10,000,000 个断开/闭合周期	-
额定负载下的触点寿命	--	100,000 个断开/闭合周期	100,000 个断开/闭合周期	-

型号	EM 8 点 数字量输出 (EM DT08)	EM 8 点继电器型 数字量输出 (EM DR08)	EM 16 点继电器型 数字量输出 (EM QR16)	EM 16 点晶体管型 数字量输出 (EM QT16)
STOP 模式下的输出行为	上一个值或替换值（默认值为 0）	上一个值或替换值（默认值为 0）	上一个值或替换值（默认值为 0）	上一个值或替换值（默认值为 0）
同时接通的输出数	8	8	• 8（无相邻点），60 ℃（水平）或 50 ℃（垂直）时 • 16，55 ℃（水平）或 45 ℃（垂直）时	16
电缆长度（最大值），以米为单位	屏蔽：500 m 非屏蔽：150 m	屏蔽：500 m 非屏蔽：150 m	屏蔽：500 m 非屏蔽：150 m	屏蔽：500 m 非屏蔽：150 m

表格 A- 72　EM DT08 8 点数字量输出 (6ES7288-2DT08-0AA0) 和 EM DR08 8 点继电器型数字量输出 (6ES7288-2DR08-0AA0) 的接线图

EM DT08 8 点数字量输出 (6ES7288-2DT08-0AA0)	EM DR08 8 点继电器型数字量输出 (6ES7288-2DR08-0AA0)
	

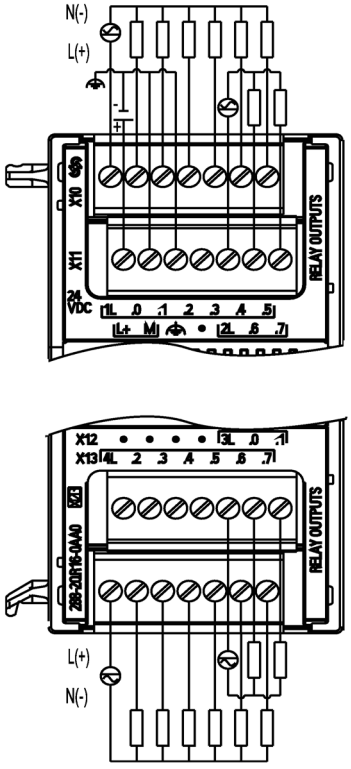
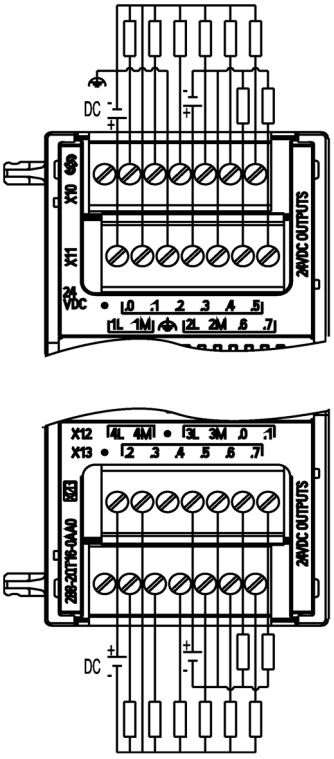
表格 A- 73　EM DT08 8 点数字量输出 (6ES7288-2DT08-0AA0) 的连接器引脚位置

引脚	X10	X11
1	1L+ / 24 V DC	无连接
2	1M / 24 V DC	2L+ / 24 V DC
3	功能性接地	2M / 24 V DC
4	DQ a.0	DQ a.4
5	DQ a.1	DQ a.5
6	DQ a.2	DQ a.6
7	DQ a.3	DQ a.7

表格 A- 74　EM DR08 8 点继电器型数字量输出 (6ES7288-2DR08-0AA0) 的连接器引脚位置

引脚	X10	X11
1	L+ / 24 V DC	功能性接地
2	M / 24 V DC	无连接
3	1L	2L
4	DQ a.0	DQ a.4
5	DQ a.1	DQ a.5
6	DQ a.2	DQ a.6
7	DQ a.3	DQ a.7

表格 A- 75　EM QR16 16 点继电器型数字量输出 (6ES7288-2QR16-0AA0) 和 EM QT16 16 点晶体管型数字量输出 (6ES7288-2QT16-0AA0) 的接线图

EM QR16 16 点继电器型数字量输出 (6ES7288-2QR16-0AA0)	EM QT16 16 点晶体管型数字量输出 (6ES7288-2QT16-0AA0)
	

表格 A- 76　EM QR16 16 点继电器型数字量输出 (6ES7288-2QR16-0AA0) 的连接器引脚位置

引脚	X10	X11	X12	X13
1	1L	L+/24 V DC	无连接	4L
2	DQ a.0	M/24 V DC	无连接	DQ b.2
3	DQ a.1	功能性接地	无连接	DQ b.3
4	DQ a.2	无连接	无连接	DQ b.4
5	DQ a.3	2L	3L	DQ b.5
6	DQ a.4	DQ a.6	DQ b.0	DQ b.6
7	DQ a.5	DQ a.7	DQ b.1	DQ b.7

表格 A- 77　EM QT16 16 点晶体管型数字量输出 (6ES7288-2QT16-0AA0) 的连接器引脚位置

引脚	X10	X11	X12	X13
1	无连接	1L/24 V DC	4L/24 V DC	无连接
2	DQ a.0	1M/24 V DC	4M/24 V DC	DQ b.2
3	DQ a.1	功能性接地	无连接	DQ b.3
4	DQ a.2	2L/24 V DC	3L/24 V DC	DQ b.4
5	DQ a.3	2M/24 V DC	3M/24 V DC	DQ b.5
6	DQ a.4	DQ a.6	DQ b.0	DQ b.6
7	DQ a.5	DQ a.7	DQ b.1	DQ b.7

A.3.3 EM DT16、EM DR16、EM DT32 和 EM DR32 数字量输入/输出规范

表格 A- 78 常规规范

型号	EM 8 点数字量输入/ 8 点数字量输出 (EM DT16)	EM 8 点数字量输入/ 8 点继电器输出 (EM DR16)	EM 16 点数字量输入/ 16 点数字量输出 (EM DT32)	EM 16 点数字量输入/ 16 点继电器输出 (EM DR32)
产品编号	6ES7288- 2DT16-0AA0	6ES7288- 2DR16-0AA0	6ES7288- 2DT32-0AA0	6ES7288-2DR32- 0AA0
尺寸 W x H x D (mm)	45 x 100 x 81	45 x 100 x 81	70 x 100 x 81	70 x 100 x 81
重量	179.7 g	201.9 g	257.3 g	295.4 g
功耗	2.5 W	5.5 W	4.5 W	10 W
电流消耗（SM 总线）	145 mA	145 mA	185 mA	180 mA
电流消耗 (24 V DC)	所用的每点输入 4 mA 所用的每个继电器线圈 11 mA	所用的每点输入 4 mA 所用的每个继电器线圈 11 mA	所用的每点输入 4 mA	所用的每点输入 4 mA

表格 A- 79　数字量输入

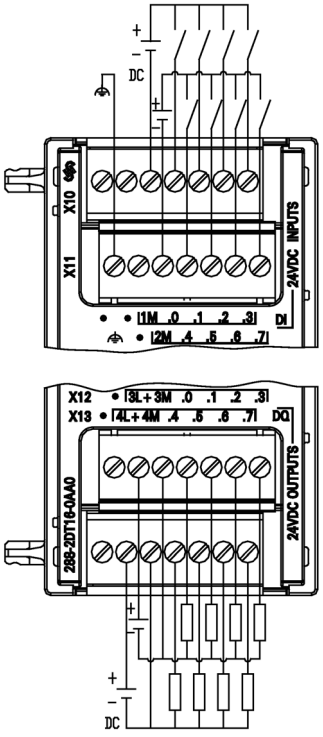
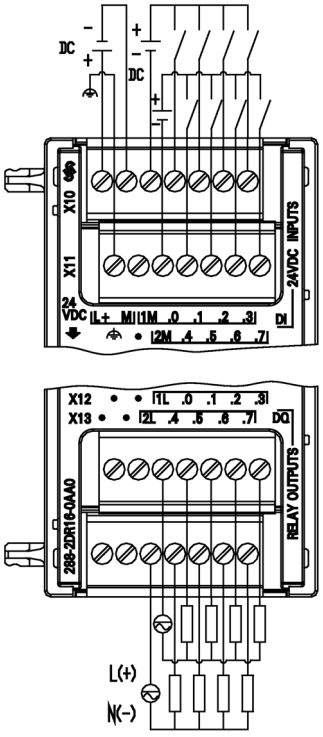
型号	EM 8 点数字量输入/8 点数字量输出 (EM DT16)	EM 8 点数字量输入/8 点继电器输出 (EM DR16)	EM 16 点数字量输入/16 点数字量输出 (EM DT32)	EM 16 点数字量输入/16 点继电器输出 (EM DR32)
输入点数	8	8	16	16
类型	漏型/源型（IEC 1 类漏型）	漏型/源型（IEC 1 类漏型）	漏型/源型（IEC 1 类漏型）	漏型/源型（IEC 1 类漏型）
额定电压	4 mA 时 24 V DC，额定值	4 mA 时 24 V DC，额定值	4 mA 时 24 V DC，额定值	4 mA 时 24 V DC，额定值
允许的连续电压	30 V DC，最大值	30 V DC，最大值	30 V DC，最大值	30 V DC，最大值
浪涌电压	35 V DC，持续 0.5 s	35 V DC，持续 0.5 s	35 V DC，持续 0.5 s	35 V DC，持续 0.5 s
逻辑 1 信号（最小）	15 V DC	15 V DC	15 V DC	15 V DC
逻辑 0 信号（最大）	5 V DC	5 V DC	5 V DC	5 V DC
隔离（现场侧与逻辑侧）	500 V AC，持续 1分钟	500 V AC，持续 1分钟	500 V AC，持续 1分钟	500 V AC，持续 1分钟
隔离组	2	2	2	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）	0.2、0.4、0.8、1.6、3.2、6.4 和 12.8 ms（可选择，4 个为一组）	0.2、0.4、0.8、1.6、3.2、6.4 和 12.8 ms（可选择，4 个为一组）	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	8	8	16	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	屏蔽： 500 m 正常输入 非屏蔽： 300 m 正常输入	屏蔽： 500 m 正常输入 非屏蔽： 300 m 正常输入	屏蔽： 500 m 正常输入 非屏蔽： 300 m 正常输入

表格 A- 80　数字量输出

型号	**EM 8 点数字量输入/ 8 点数字量输出 (EM DT16)**	**EM 8 点数字量输入/ 8 点继电器输出 (EM DR16)**	**EM 16 点数字量输入/16 点数字量输出 (EM DT32)**	**EM 16 点数字量输入/ 16 点继电器输出 (EM DR32)**
输出点数	8	8	16	16
类型	固态 - MOSFET（源型）	继电器，干触点	固态 - MOSFET（源型）	继电器，干触点
电压范围	20.4 到 28.8 V DC	5 到 30 V DC 或 5 到 250 V AC	20.4 到 28.8 V DC	5 到 30 V DC 或 5 到 250 V AC
最大电流时的逻辑 1 信号	20 V DC，最小值	--	20 V DC，最小值	--
具有 10 KΩ 负载时的逻辑 0 信号	0.1 V DC，最大值	--	0.1 V DC，最大值	--
每点的额定电流（最大）	0.75 A	2 A	0.75 A	2 A
每个公共端的额定电流（最大）	3 A	8 A	6 A	8 A
灯负载	5 W	30 W DC/200 W AC	5 W	30 W DC/200 W AC
通态触点电阻	最大 0.6 Ω	新设备最大为 0.2 Ω	最大 0.6 Ω	新设备最大为 0.2 Ω
每点的漏电流	最大 10 μA	--	最大 10 μA	--
浪涌电流	8 A，最长持续 100 ms	触点闭合时为 7 A	8 A，最长持续 100 ms	触点闭合时为 7 A
过载保护	无	无	无	无
隔离（现场侧与逻辑侧）	500 V AC，持续 1分钟	1500 V AC，持续 1 分钟（线圈与触点） 无（线圈与逻辑侧）	500 V AC，持续 1分钟	1500 V AC，持续 1 分钟（线圈与触点） 无（线圈与逻辑侧）
隔离电阻	--	新设备最小为 100 MΩ	--	新设备最小为 100 MΩ

型号	EM 8 点数字量输入/8 点数字量输出 (EM DT16)	EM 8 点数字量输入/8 点继电器输出 (EM DR16)	EM 16 点数字量输入/16 点数字量输出 (EM DT32)	EM 16 点数字量输入/16 点继电器输出 (EM DR32)
断开触点间的绝缘	--	750 V AC，持续 1 分钟	--	750 V AC，持续 1 分钟
隔离组	2	2	3	4
电感钳位电压	- 48 V	--	- 48 V	--
开关延时	断开到接通最长为 50 µs 接通到断开最长为 200 µs	最长 10 ms	断开到接通最长为 50 µs 接通到断开最长为 200 µs	最长 10 ms
机械寿命（无负载）	--	10,000,000 个断开/闭合周期	--	10,000,000 个断开/闭合周期
额定负载下的触点寿命	--	100,000 个断开/闭合周期	--	100,000 个断开/闭合周期
STOP 模式下的输出行为	上一个值或替换值（默认值为 0）	上一个值或替换值（默认值为 0）	上一个值或替换值（默认值为 0）	上一个值或替换值（默认值为 0）
同时接通的输出数	8	8	16	16
电缆长度（最大值），以米为单位	屏蔽：500 m 非屏蔽：150 m	屏蔽：500 m 非屏蔽：150 m	屏蔽：500 m 非屏蔽：150 m	屏蔽：500 m 非屏蔽：150 m

表格 A- 81　EM DT16 8 点数字量输入/8 点数字量输出 (6ES7288-2DT16-0AA0) 和 EM DR16 8 点数字量输入/8 点继电器输出 (6ES7288-2DR16-0AA0) 的接线图

EM DT16 8 点数字量输入/8 点数字量输出 (6ES7288-2DT16-0AA0)	EM DR16 8 点数字量输入/8 点继电器输出 (6ES7288-2DR16-0AA0)
	

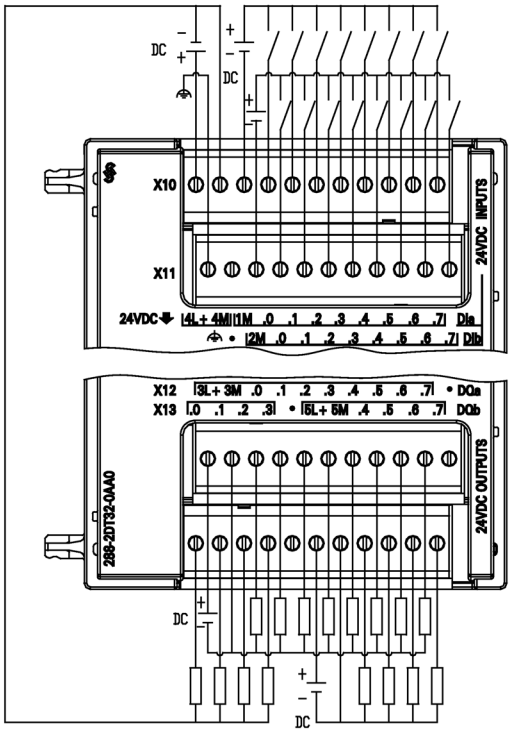
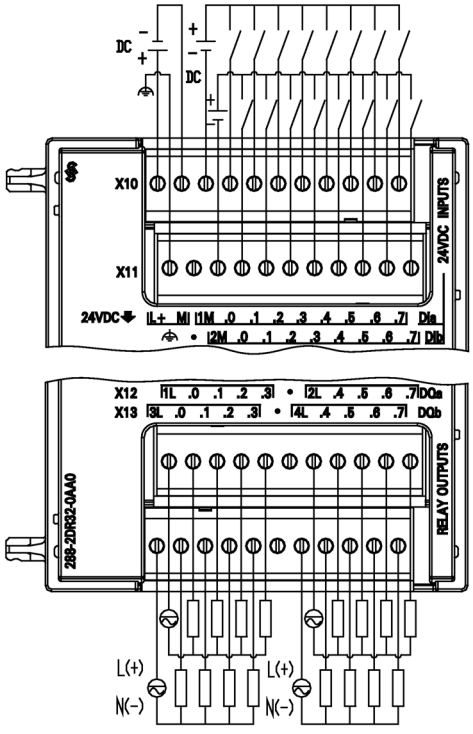
表格 A- 82　EM DT16 8 点数字量输入/8 点数字量输出 (6ES7288-2DT16-0AA0) 的连接器引脚位置

引脚	X10	X11	X12	X13
1	无连接	功能性接地	无连接	无连接
2	无连接	无连接	3L+ / 24 V DC	4L+ / 24 V DC
3	1M	2M	3M / 24 V DC	4M / 24 V DC
4	DI a.0	DI a.4	DQ a.0	DQ a.4
5	DI a.1	DI a.5	DQ a.1	DQ a.5
6	DI a.2	DI a.6	DQ a.2	DQ a.6
7	DI a.3	DI a.7	DQ a.3	DQ a.7

表格 A- 83　EM DR16 8 点数字量输入/8 点继电器输出 (6ES7288-2DR16-0AA0) 的连接器引脚位置

引脚	X10	X11	X12	X13
1	L+ / 24 V DC	功能性接地	无连接	无连接
2	M / 24 V DC	无连接	无连接	无连接
3	1M	2M	1L	2L
4	DI a.0	DI a.4	DQ a.0	DQ a.4
5	DI a.1	DI a.5	DQ a.1	DQ a.5
6	DI a.2	DI a.6	DQ a.2	DQ a.6
7	DI a.3	DI a.7	DQ a.3	DQ a.7

表格 A- 84　EM DT32 16 点数字量输入/16 点数字量输出·(6ES7288-2DT32-0AA0) 和 EM DR32 16 点数字量输入/16 点继电器输出 (6ES7288-2DR32-0AA0) 的接线图

EM DT32 16 点数字量输入/16 点数字量输出 (6ES7288-2DT32-0AA0)	EM DR32 16 点数字量输入/16 点继电器输出 (6ES7288-2DR32-0AA0)
	

表格 A- 85　EM DT32 16 点数字量输入/16 点数字量输出 (6ES7288-2DT32-0AA0) 的连接器引脚位置

引脚	X10	X11	X12	X13
1	4L+ / 24 V DC[1]	功能性接地	3L+ / 24 V DC	DQ b.0[1]
2	4M / 24 V DC[1]	无连接	3M / 24 V DC	DQ b.1[1]
3	1M	2M	DQ a.0	DQ b.2[1]
4	DI a.0	DI b.0	DQ a.1	DQ b.3[1]
5	DI a.1	DI b.1	DQ a.2	无连接
6	DI a.2	DI b.2	DQ a.3	5L+ / 24 V DC
7	DI a.3	DI b.3	DQ a.4	5M / 24 V DC
8	DI a.4	DI b.4	DQ a.5	DQ b.4
9	DI a.5	DI b.5	DQ a.6	DQ b.5
10	DI a.6	DI b.6	DQ a.7	DQ b.6
11	DI a.7	DI b.7	无连接	DQ b.7

[1] 在同一隔离组。

表格 A- 86　EM DR32 16 点数字量输入/16 点继电器输出 (6ES7288-2DR32-0AA0) 的连接器引脚位置

引脚	X10	X11	X12	X13
1	L+ / 24 V DC	功能性接地	1L	3L
2	M / 24 V DC	无连接	DQ a.0	DQ b.0
3	1M	2M	DQ a.1	DQ b.1
4	DI a.0	DI b.0	DQ a.2	DQ b.2
5	DI a.1	DI b.1	DQ a.3	DQ b.3
6	DI a.2	DI b.2	无连接	无连接
7	DI a.3	DI b.3	2L	4L
8	DI a.4	DI b.4	DQ a.4	DQ b.4
8	DI a.5	DI b.5	DQ a.5	DQ b.5
10	DI a.6	DI b.6	DQ a.6	DQ b.6
11	DI a.7	DI b.7	DQ a.7	DQ b.7

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- 87 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)
产品编号	6ES7288-3AE04-0AA0	6ES7288-3AE08-0AA0
尺寸 W x H x D (mm)	45 x 100 x 81	45 x 100 x 81
重量	147 g	186 g
功耗	1.5 W（无负载）	2.0 W（无负载）
电流消耗（SM 总线）	80 mA	80 mA
电流消耗 (24 V DC)	40 mA（无负载）	70 mA（无负载）

表格 A- 88 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（差动），可选择，2 个为一组	电压或电流（差动），可选择，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	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 801)。）
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	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 801)。）

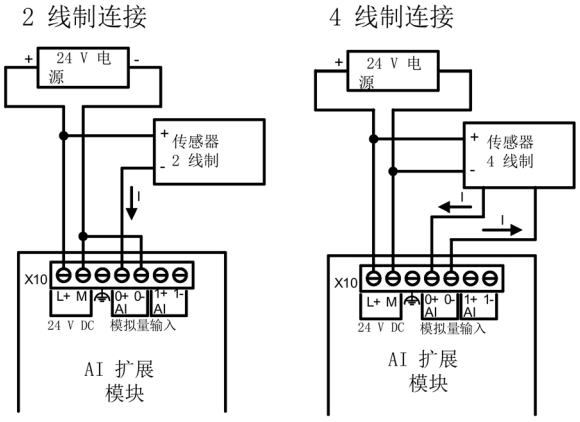
型号	EM 4 点模拟量输入 (EM AE04)	EM 8 点模拟量输入 (EM AE08)
分辨率	电压模式：12 位 + 符号位 电流模式：12 位	电压模式：12 位 + 符号位 电流模式：12 位
最大耐压/耐流	±35 V/±40 mA	±35 V/±40 mA
平滑化	无、弱、中或强	无、弱、中或强（参见阶跃响应下的模拟量输入响应时间） (页 800)
噪声抑制	400、60、50 或 10 Hz	400、60、50 或 10 Hz
输入阻抗	≥9 M Ω（电压）/250 Ω（电流）	≥9 M Ω（电压）/250 Ω（电流）
隔离（现场侧与逻辑侧）	无	无
精度（25 °C/0 到 55 °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 屏蔽双绞线	100 m 屏蔽双绞线

表格 A- 89　诊断

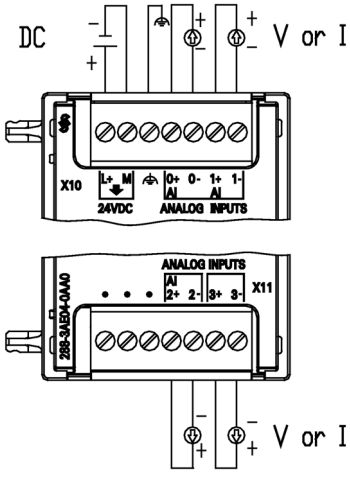
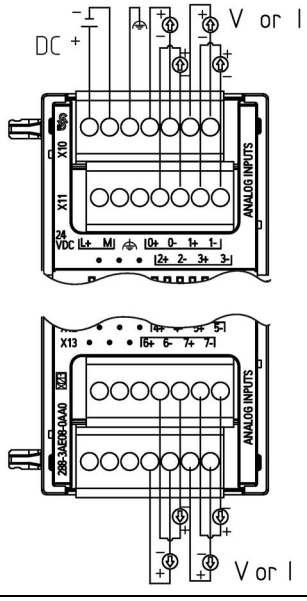
型号	EM 4 点模拟量输入 (EM AE04)	EM 8 点模拟量输入 (EM AE08)
上溢/下溢	有	有
24 V DC 低压	有	有

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。



表格 A- 90　EM AE04 4 点模拟量输入 (6ES7288-3AE04-0AA0) 和 EM AE08 8 点模拟量输入 (6ES7288-3AE08-0AA0) 的接线图

EM AE04 4 点模拟量输入 (6ES7288-3AE04-0AA0)	EM AE08 8 点模拟量输入 (6ES7288-3AE08-0AA0)
	
注：连接器必须镀金。有关产品编号，请参见附录 F“备件和其它硬件”。	

表格 A- 91　EM AE04 4 点模拟量输入 (6ES7288-3AE04-0AA0) 的连接器引脚位置

引脚	X10（镀金）	X11（镀金）
1	L+ / 24 V DC	无连接
2	M / 24 V DC	无连接
3	功能性接地	无连接
4	AI 0+	AI 2+
5	AI 0-	AI 2-
6	AI 1+	AI 3+
7	AI 1-	AI 3-

表格 A- 92　EM AE08 8 点模拟量输入 (6ES7288-3AE08-0AA0) 的连接器引脚位置

引脚	X10（镀金）	X11（镀金）	X12（镀金）	X13（镀金）
1	L+ / 24 V DC	无连接	无连接	无连接
2	M / 24 V DC	无连接	无连接	无连接
3	功能性接地	无连接	无连接	无连接
4	AI 0+	AI 2+	AI 4+	AI 6+
5	AI 0-	AI 2-	AI 4-	AI 6-
6	AI 1+	AI 3+	AI 5+	AI 7+
7	AI 1-	AI 3-	AI 5-	AI 7-

A.4.2 EM AQ02 和 EM AQ04 模拟量输出模块规范

表格 A- 93 常规规范

技术数据	EM 2 点模拟量输出 (EM AQ02)	EM 4 点模拟量输出 (EM AQ04)
产品编号	6ES7288-3AQ02-0AA0	6ES7288-3AQ04-0AA0
尺寸 W x H x D (mm)	45 x 100 x 81	45 x 100 x 81
重量	147.1 g	170.5 g
功耗	1.5 W（无负载）	2.1 W（无负载）
电流消耗（SM 总线）	60 mA	60 mA
电流消耗 (24 V DC)	50 mA（无负载）	75 mA（无负载）

表格 A- 94 模拟量输出

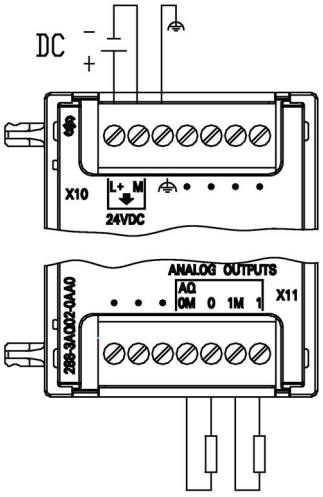
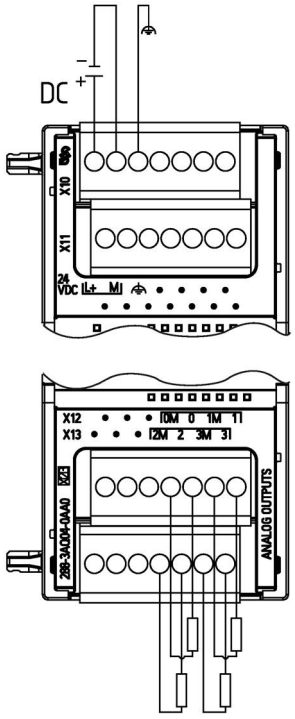
技术数据	EM 2 点模拟量输出 (EM AQ02)	EM 4 点模拟量输出 (EM AQ04)
输出点数	2	4
类型	电压或电流	电压或电流
范围	±10 V 或 0 到 20 mA	±10 V 或 0 到 20 mA
分辨率	电压模式：11 位 + 符号位 电流模式：11 位	电压模式：11 位 + 符号位 电流模式：11 位
满量程范围（数据字）	电压：-27,648 到 27,648 电流：0 到 27,648	电压：-27,648 到 27,648 电流：0 到 27,648 （请参见电压和电流的输出范围 (页 803)。）
精度（25 °C/0 到 55 °C）	满量程的 ±0.5%/±1.0%	满量程的 ±0.5%/±1.0%
稳定时间（新值的 95%）	电压：300 µs (R)，750 µs (R)，750 µs (1 µF) 电流：600 µs (1 mH)，2 ms (10 mH)	电压：300 µs (R)，750 µs (R)，750 µs (1 µF) 电流：600 µs (1 mH)，2 ms (10 mH)
负载阻抗	电压 ：≥ 1000 Ω 电流：≤ 500 Ω	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）	上一个值或替换值（默认值为 0）

技术数据	EM 2 点模拟量输出 (EM AQ02)	EM 4 点模拟量输出 (EM AQ04)
隔离（现场侧与逻辑侧）	无	无
电缆长度（最大值），以米为单位	100 m 屏蔽双绞线	100 m 屏蔽双绞线

表格 A- 95　诊断

技术数据	EM 2 点模拟量输出 (EM AQ02)	EM 4 点模拟量输出 (EM AQ04)
上溢/下溢	有	有
对地短路（仅限电压模式）	有	有
断路（仅限电流模式）	有	有
24 V DC 低压	有	有

表格 A- 96　EM AQ02 2 点模拟量输出 (6ES7288-3AQ02-0AA0) 和 EM AQ04 4 点模拟量输出 (6ES7288-3AQ04-0AA0) 的接线图

EM AQ02 2 点模拟量输出 (6ES7288-3AQ02-0AA0)	EM AQ04 4 点模拟量输出 (6ES7288-3AQ04-0AA0)
	
注：连接器必须镀金。有关产品编号，请参见附录 F“备件和其它硬件”。	

表格 A- 97　EM AQ02 2 点模拟量输出 (6ES7288-3AQ02-0AA0) 的连接器引脚位置

引脚	X10（镀金）	X11（镀金）
1	L+ / 24 V DC	无连接
2	M / 24 V DC	无连接
3	功能性接地	无连接
4	无连接	AQ 0M
5	无连接	AQ 0
6	无连接	AQ 1M
7	无连接	AQ 1

表格 A- 98　EM AQ04 4 点模拟量输出 (6ES7288-3AQ04-0AA0) 的连接器引脚位置

引脚	X10（镀金）	X11（镀金）	X12（镀金）	X13（镀金）
1	L+ / 24 V DC	无连接	无连接	无连接
2	M / 24 V DC	无连接	无连接	无连接
3	功能性接地	无连接	无连接	无连接
4	无连接	无连接	AQ 0M	AQ 2M
5	无连接	无连接	AQ 0	AQ 2
6	无连接	无连接	AQ 1M	AQ 3M
7	无连接	无连接	AQ 1	AQ 3

A.4.3 EM AM03 和 EM AM06 模拟量输入/输出模块规范

表格 A- 99 常规规范

技术数据	EM 2 点模拟量输入/1 点模拟量输出 (AM03)	EM 4 点模拟量输入/2 点模拟量输出 (AM06)
产品编号	6ES7288-3AM03-0AA0	6ES7288-3AM06-0AA0
尺寸 W x H x D (mm)	45 x 100 x 81	45 x 100 x 81
重量	172 g	173.4 g
功耗	1.1 W（无负载）	2.0 W（无负载）
电流消耗（SM 总线）	60 mA	80 mA
电流消耗 (24 V DC)	30 mA（无负载）	60 mA（无负载）

表格 A- 100 模拟量输入

型号	EM 2 点模拟量输入/1 点模拟量输出 (AM03)	EM 4 点模拟量输入/2 点模拟量输出 (AM06)
输入点数	2	4
类型	电压或电流（差动）：可 2 个选为一组	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	±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/-4,864 到 0	电压：27,649 到 32,511/-27,649 到 -32,512 电流：27,649 到 32,511/-4,864 到 0
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	电压：32,512 到 32,767/-32,513 到 -32,768 电流：32,512 到 32,767/-4,865 到 -32,768
分辨率	电压模式：12 位 + 符号 电流模式：12 位	电压模式：12 位 + 符号 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	无、弱、中或强

型号	EM 2 点模拟量输入/1 点模拟量输出 (AM03)	EM 4 点模拟量输入/2 点模拟量输出 (AM06)
噪声抑制	400、60、50 或 10 Hz	400、60、50 或 10 Hz
输入阻抗	≥9 M Ω	≥9 M Ω
隔离（现场侧与逻辑侧）	无	无
精度（25 °C/0 到 55 °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%
模数转换时间	625 µs（400 Hz 抑制）	625 µs（400 Hz 抑制）
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 屏蔽双绞线	100 m 屏蔽双绞线

表格 A- 101 模拟量输出

技术数据	EM 2 点模拟量输入/1 点模拟量输出 (AM03)	EM 4 点模拟量输入/2 点模拟量输出 (AM06)
输出点数	1	2
类型	电压或电流	电压或电流
范围	±10 V 或 0 到 20 mA	±10 V 或 0 到 20 mA
分辨率	电压模式：11 位 + 符号 电流模式：11 位	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	电压：-27,648 到 27,648 电流：0 到 27,648
精度（25 °C/0 到 55 °C）	满量程的 ±0.5%/±1.0%	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)	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：≤ 500 Ω	电压 ：≥ 1000 Ω 电流：≤ 500 Ω
STOP 模式下的输出行为	上一个值或替换值（默认值为 0）	上一个值或替换值（默认值为 0）

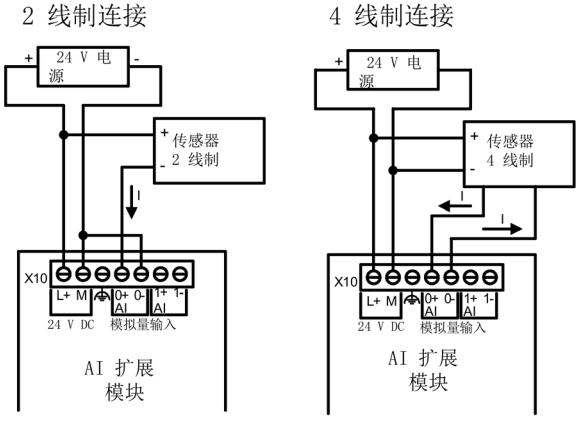
技术数据	EM 2 点模拟量输入/1 点模拟量输出 (AM03)	EM 4 点模拟量输入/2 点模拟量输出 (AM06)
隔离（现场侧与逻辑侧）	无	无
电缆长度（最大值），以米为单位	100 m 屏蔽双绞线	100 m 屏蔽双绞线

表格 A- 102 诊断

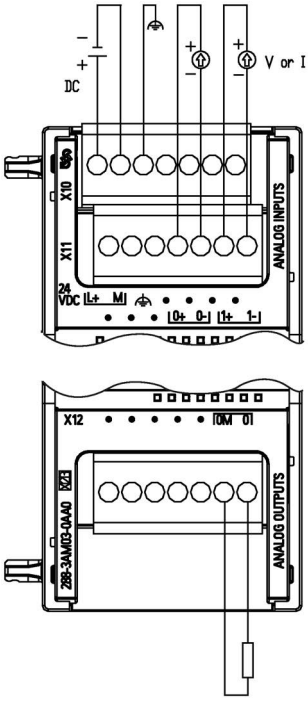
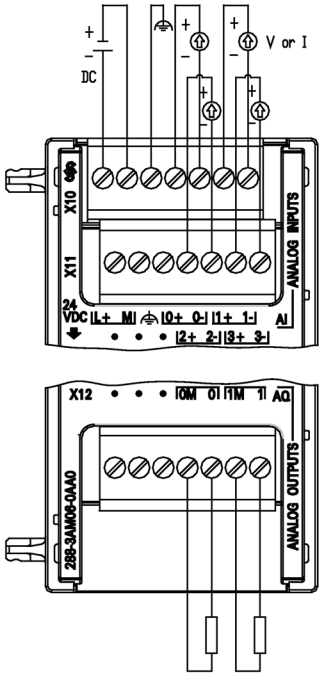
型号	EM 2 点模拟量输入/1 点模拟量输出 (AM03)	EM 4 点模拟量输入/2 点模拟量输出 (AM06)
上溢/下溢	有	有
对地短路（仅限电压模式）	有	有
断路（仅限电流模式）	有	有
24 V DC 低压	有	有

EM AM03 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。



表格 A- 103 EM AM03 2 点模拟量输入/1 点模拟量输出 (6ES7288-3AM03-0AA0) 和 AM06 4 点模拟量输入/2 点模拟量输出 (6ES7288-3AM06-0AA0) 的接线图

EM AM03 2 点模拟量输入/1 点模拟量输出 (6ES7288-3AM03-0AA0)	EM AM06 4 点模拟量输入/2 点模拟量输出 (6ES7288-3AM06-0AA0)
	
注：连接器必须镀金。有关产品编号，请参见附录 F“备件和其它硬件”。	

表格 A- 104 AM03 2 点模拟量输入/1 点模拟量输出 (6ES7288-3AM03-0AA0) 的连接器引脚位置

引脚	X10（镀金）	X11（镀金）	X12（镀金）
1	L+ / 24 V DC	无连接	无连接
2	M / 24 V DC	无连接	无连接
3	功能性接地	无连接	无连接
4	无连接	AI 0+	无连接
5	无连接	AI 0-	无连接
6	无连接	AI 1+	AQ 0M
7	无连接	AI 1-	AQ 0

表格 A- 105 AM06 4 点模拟量输入/2 点模拟量输出 (6ES7288-3AM06-0AA0) 的连接器引脚位置

引脚	X10（镀金）	X11（镀金）	X12（镀金）
1	L+ / 24 V DC	无连接	无连接
2	M / 24 V DC	无连接	无连接
3	功能性接地	无连接	无连接
4	AI 0+	AI 2+	AQ 0M
5	AI 0-	AI 2-	AQ 0
6	AI 1+	A1 3+	AQ 1M
7	AI 1-	A1 3-	AQ 1

A.4.4 模拟量输入的阶跃响应

表格 A- 106 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- 107 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- 108 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	

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	
-27649	93FF					下冲范围
-32512	8100	-11.759 V	-5.879 V	-2.940 V	-1.470 V	
-32512	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- 109 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	4 mA 到 20 mA	
32767	7FFF	23.70 mA	22.96 mA	上溢
32512	7F00			
32511	7EFF	23.52 mA	22.81 mA	过冲范围
27649	6C01			
27648	6C00	20 mA	20 mA	额定范围
20736	5100	15 mA	16 mA	
1	1	723.4 nA	4 mA + 578.7 nA	
0	0	0 mA	4 mA	
-1	FFFF			下冲范围
-4864	ED00	-3.52 mA	1.185 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- 110 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	
-27649	93FF		下冲范围
-32512	8100	-11.76 V	
-32512	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- 111 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	4 mA 到 20 mA	
32767	7FFF	请参见注 1	请参见注 1	上溢
32512	7F00	请参见注 1	请参见注 1	
32511	7EFF	23.52 mA	22.81 mA	过冲范围
27649	6C01			
27648	6C00	20 mA	20 mA	额定范围

系统		当前输出范围		
十进制	十六进制	0 mA 到 20 mA	4 mA 到 20 mA	
20736	5100	15 mA	16 mA	
1	1	723.4 nA	4 mA + 578.7 nA	
0	0	0 mA	4mA	
-1	FFFF		4 mA 到 578.7 nA	下冲范围
-6912	E500		0 mA	
-6913	E4FF			不可能。输出值限制在 0 mA。
-32512	8100			
-32512	80FF	请参见注 1	请参见注 1	下溢
-32768	8000	请参见注 1	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 热电偶模块和 RTD 扩展模块 (EM)

A.5.1 热电偶扩展模块 (EM)

A.5.1.1 EM AT04 热电偶规范

表格 A- 112 常规规范

型号	EM AT04 AI 4 x 16 位 TC
产品编号	6ES7288-3AT04-0AA0
尺寸 W x H x D (mm)	45 x 100 x 81
重量	125 g
功耗	1.5 W
电流消耗（SM 总线）	80 mA
电流消耗 (24 V DC) [1]	40 mA

[1] 20.4 到 28.8 V DC（2 类受限制电源，或 PLC 提供的传感器电源）

表格 A- 113 模拟量输入

型号		EM AT04 AI 4 x 16 位 TC
输入点数		4
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
噪声抑制		对于所选滤波器设置 （10 Hz、50 Hz、60 Hz 或 400 Hz）为 85 dB
共模抑制		120 VAC 时大于 120 dB
阻抗		≥ 10 MΩ

型号		EM AT04 AI 4 x 16 位 TC
隔离	现场侧与逻辑侧	500 V AC
	现场侧与 24 V DC	500 V AC
	24 V DC 与逻辑侧	500 V AC
通道间隔离		--
精度		请参见热电偶选型表。
可重复性		±0.05% FS
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		±1.5 °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- 114 诊断

型号	EM AT04 AI 4 x 16 位 TC
上溢/下溢 [1]	√
断线（仅限电流模式）[2]	√
24 V DC 低压[1]	√

[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。

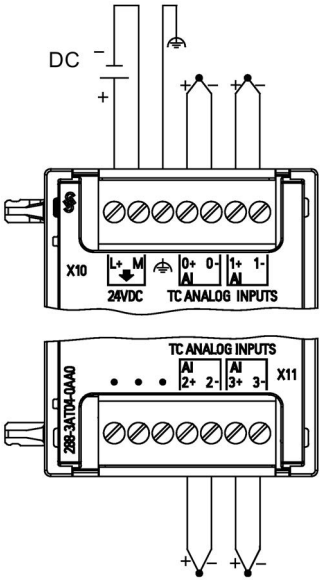
[2] 如果断线报警已禁用，但传感器接线存在开路情况，则模块可能会报告随机值。

EM AT04 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。

表格 A- 115 EM AT04 热电偶 4 点 16 位 (6ES7288-3AT04-0AA0) 接线图

EM AT04 4 x 16 位 (6ES7288-3AT04-0AA0)	
	
注：连接器必须镀金。有关产品编号，请参见附录 F“备件和其它硬件”。	

① 为清晰起见，未显示 TC 2、3、4 和 5 的连接。

表格 A- 116 EM AT04 4 点 16 位 (6ES7288-3AT04-0AA0) 的连接器引脚位置

引脚	X10（镀金）	X11（镀金）
1	L+ / 24 V DC	无连接
2	M / 24 V DC	无连接
3	功能性接地	无连接
4	AI 0+/TC	AI 2+/TC
5	AI 0-/TC	AI 2-/TC
6	AI 1+/TC	AI 3+/TC
7	AI 1-/TC	AI 3-/TC

说明

应将未使用的模拟量输入短路。
可以取消激活热电偶的未使用通道。如果取消激活未使用的通道，不会出现任何错误。

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 AT04
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 AT04 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- 117 EM AT04 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 55 °C 时的额定范围 [1, 2] 精度
J	-210.0 °C	-150.0 °C	1200.0 °C	1450.0 °C	±0.3 °C	±0.6 °C
K	-270.0 °C	-200.0 °C	1372.0 °C	1622.0 °C	±0.4 °C	±1.0 °C
T	-270.0 °C	-200.0 °C	400.0 °C	540.0 °C	±0.5 °C	±1.0 °C
E	-270.0 °C	-200.0 °C	1000.0 °C	1200.0 °C	±0.3 °C	±0.6 °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	±2.5 °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
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 ±1.5 °C。该误差已包括到本表的误差中。模块需要至少 30 分钟的预热时间才能满足该规范。

4 若是暴露在 970 MHz 到 990 MHz 的无线电辐射频率下，EM AT04 AI 4 x 16 位 TC 的精度可能会有所下降。

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- 118 EM AT04 热电偶的噪声消减和更新时间

抑制频率选择	积分时间	4 通道模块更新时间（秒）
400 Hz (2.5 ms)	10 ms [1]	0.143
60 Hz (16.6 ms)	16.67 ms	0.223
50 Hz (20 ms)	20 ms	0.263
10 Hz (100 ms)	100 ms	1.225

[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- 119 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

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	
-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.5.2 RTD 扩展模块 (EM)

EM RTD 规范

表格 A- 120 常规规范

技术数据	EM 2 点 16 位 RTD (EM AR02)	EM RTD 4 x 16 位 (EM AR04)
产品编号	6ES7288-3AR02-0AA0	6ES7288-3AR04-0AA0
尺寸 W x H x D (mm)	45 x 100 x 81	45 x 100 x 81
重量	148.7 g	150 g
功耗	1.5 W	1.5 W
电流消耗（SM 总线）	80 mA	80 mA
电流消耗 (24 V DC) [1]	40 mA	40 mA

表格 A- 121 模拟量输入

技术数据		**EM 2 点 16 位 RTD (EM AR02)**	**EM RTD 4 x 16 位 (EM AR04)**
输入点数		2	4
类型		模块参考 RTD 和 Ω	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	请参见 RTD 传感器选型表。
分辨率	温度	0.1 °C/0.1 °F	0.1 °C/0.1 °F
	电阻	15 位 + 符号位	15 位 + 符号位
最大耐压		±35 V	±35 V
噪声抑制		85 dB (10 Hz/50 Hz/60 Hz/400 Hz)	85 dB (10 Hz/50 Hz/60 Hz/400 Hz)
共模抑制		> 120 dB	>120 dB
阻抗		≥10 M Ω	≥10 M Ω
隔离	现场侧与逻辑侧	500 V AC	500 V AC
	现场侧与 24 V DC	500 V AC	500 V AC
	24 V DC 与逻辑侧	500 V AC	500 V AC
通道间隔离		0	0
精度		请参见 RTD 传感器选型表	请参见 RTD 传感器选型表。
可重复性		±0.05% FS	±0.05% FS
最大传感器功耗		0.5 m W	0.5 m W
测量原理		Sigma-delta	Sigma-delta
模块更新时间		请参见降噪选项表	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	至传感器最长 100 m
导线电阻（最大）	10 Ω RTD 除外	20 Ω	20 Ω
	10 Ω RTD	2.7 Ω	2.7 Ω

表格 A- 122 诊断

技术数据	EM 2 点 16 位 RTD (EM AR02)	EM RTD 4 x 16 位 (EM AR04)
上溢/下溢 1,2	有	有
断线3	有	有
24 V DC 低压1	有	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- 123 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	Pt 10	-243.0 °C	-200.0 °C	850.0 °C	1000.0 °C	±1.0 °C	±2.0 °C
ITS90 DIN EN 60751	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 100	-243.0 °C	-200.0 °C	850.0 °C	1000.0 °C	±0.5 °C	±1.0 °C
Pt 0.003916 Pt 0.003920	Pt 200	-243.0 °C	-200.0 °C	850.0°C	1000.0°C	± 0.5 °C	±1.0 °C
	Pt 500						
	Pt 1000						

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 0.003910	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						
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- 124 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%

[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 500 Ω 和 1000 Ω RTD 范围时，误差可能增加到指定误差的两倍。

若使用 4 线制连接，对于 10 Ω RTD 范围，将得到最高精度。

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- 125 RTD 模块的噪声消减和更新时间

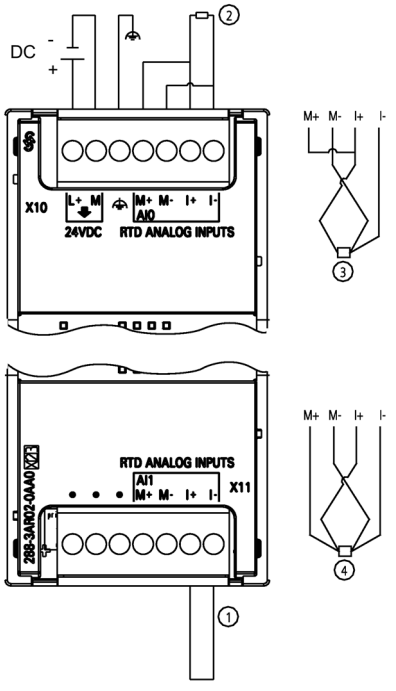
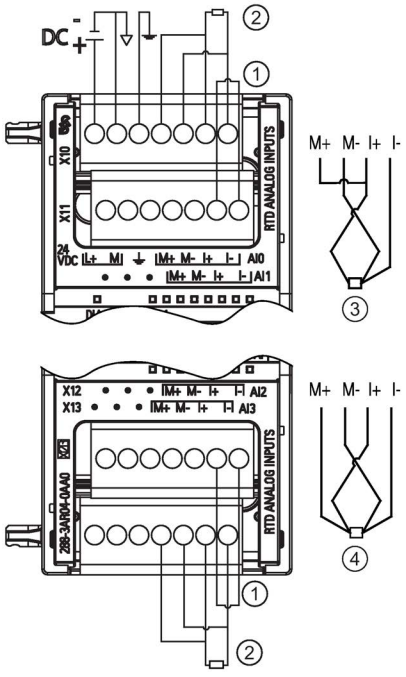
抑制频率选择	积分时间	更新时间（秒）
400 Hz (2.5 ms)	10 ms [1]	4/2 线制：0.142 3 线制：0.285
60 Hz (16.6 ms)	16.67 ms	4/2 线制：0.222 3 线制：0.445
50 Hz (20 ms)	20 ms	4/2 线制：0.262 3 线制：.505
10 Hz (100 ms)	100 ms	4/2 线制：1.222 3 线制：2.445

[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。该选择还可抑制 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。

表格 A- 126 EM AR02 RTD 2 点 16 位 (6ES7288-3AR02-0AA0) 和 EM AR04 RTD 4 点 16 位 (6ES7288-3AR04-0AA0) 的接线图

EM AR02 RTD 2 点 16 位 (6ES7288-3AR02-0AA0)	EM AR04 RTD 4 点 16 位 (6ES7288-3AR04-0AA0)
	

注：连接器必须镀金。有关产品编号，请参见附录 F“备件和其它硬件”。

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：连接器必须镀金。有关产品编号，请参见附录 F“备件和其它硬件”。

表格 A- 127 EM AR02 RTD 2 点 16 位 (6ES7288-3AR02-0AA0) 的连接器引脚位置

引脚	X10（镀金）	X11（镀金）
1	L+ / 24 V DC	无连接
2	M / 24 V DC	无连接
3	功能性接地	无连接
4	AI 0 M+/RTD	AI 1 M+/RTD
5	AI 0 M-/RTD	AI 1 M-/RTD
6	AI 0 I+/RTD	AI 1 I+/RTD
7	AI 0 I-/RTD	AI 1 I-/RTD

表格 A- 128 EM AR04 RTD 4 点 16 位 (6ES7288-3AR04-0AA0) 的连接器引脚位置

引脚	X10（镀金）	X11（镀金）	X12（镀金）	X13（镀金）
1	L+ / 24 V DC	无连接	无连接	无连接
2	M / 24 V DC	无连接	无连接	无连接
3	功能性接地	无连接	无连接	无连接
4	AI 0 M+/RTD	AI 1 M+/RTD	AI 2 M+/RTD	AI 3 M+/RTD
5	AI 0 M-/RTD	AI 1 M-/RTD	AI 2 M-/RTD	AI 3 M-/RTD
6	AI 0 I+/RTD	AI 1 I+/RTD	AI 2 I+.RTD	AI 3 I+/RTD
7	AI 0 I-/RTD	AI 1 I-/RTD	AI 2 I-/RTD	AI 3 I/-/RTD

A.6 数字信号板

A.6.1 SB DT04 数字量输入/输出规范

表格 A- 129 常规规范

技术数据	SB 2 点数字量输入/2 点数字量输出 (DT04)
产品编号	6ES7288-5DT04-0AA0
尺寸 W x H x D (mm)	35 x 52.2 x 16
重量	18.1 g
功耗	1.0 W
电流消耗 (5 V DC)	50 mA
电流消耗 (24 V DC)	所用的每点输入 4 mA

表格 A- 130 数字量输入

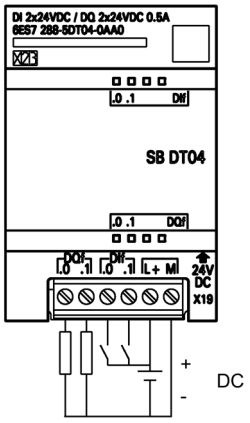
技术数据	SB 2 点数字量输入/2 点数字量输出 (DT04)
输入点数	2
类型	漏型（IEC 1 类漏型）
额定电压	4 mA 时 24 V DC，额定值
允许的连续电压	30 V DC，最大值
浪涌电压	35 V DC，持续 0.5 s
逻辑 1 信号（最小）	2.5 mA 时 15 V DC
逻辑 0 信号（最大）	1 mA 时 5 V DC
隔离（现场侧与逻辑侧）	500 V AC，持续 1 分钟
隔离组	1
滤波时间	每个通道上可单独选择： μs：0.2、0.4、0.8、1.6、3.2、6.4、12.8 ms：0.2、0.4、0.8、1.6、3.2、6.4、12.8

技术数据	SB 2 点数字量输入/2 点数字量输出 (DT04)
同时接通的输入数	2
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

表格 A- 131 数字量输出

技术数据	SB 2 点数字量输入/2 点数字量输出 (DT04)
输出点数	2
输出类型	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
最大电流时的逻辑 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）	1 A
灯负载	5 W
通态触点电阻	最大 0.6 Ω
每点的漏电流	最大 10 μA
浪涌电流	5 A，最长持续 100 ms
过载保护	无
隔离（现场侧与逻辑侧）	500 V AC，持续 1 分钟
隔离组	1
电感钳位电压	L+ - 48 V，1 W 损耗
开关延时	断开到接通最长为 2 μs 接通到断开最长为 10 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	2
电缆长度（最大值），以米为单位	屏蔽：500 m 正常输入 非屏蔽：150 m 正常输入

表格 A- 132 SB DT04 2 点数字量输入/2 点数字量输出 (6ES7288-5DT04-0AA0) 的接线图

SB DT04 2 点数字量输入/2 点数字量输出 (6ES7288-5DT04-0AA0)	
	

表格 A- 133 SB DT04 2 点数字量输入/2 点数字量输出 (6ES7288-5DT04-0AA0) 的连接器引脚位置

引脚	X19
1	DQ f.0
2	DQ f.1
3	DI f.0
4	DI f.1
5	L+ / 24 V DC
6	M / 24 V DC

A.7 模拟信号板

A.7.1 SB AE01 模拟量输入规范

表格 A- 134 常规规范

技术数据	SB 1 点模拟量输入 (SB AE01)
产品编号	6ES7288-5AE01-0AA0
尺寸 W x H x D (mm)	35 x 52.2 x 16
重量	20 g
功耗	0.4 W
电流消耗 (5 V DC)	50 mA（5 V 和 3.3 V 组合）
电流消耗 (24 V DC)	无

表格 A- 135 模拟量输入

技术数据	SB 1 点模拟量输入 (SB AE01)
输入点数	1
类型	电压或电流（差动）
范围	±10 V、±5 V、±2.5 V 或 0 到 20 mA
分辨率	11 位 + 符号位（电压模式） 11 位（电流模式）
满量程范围（数据字）	-27,648 到 27,648
精度（25 °C/0 到 55 °C）	电压模式：满量程的 ±0.3%/±0.6% 电压模式：满量程的 ±0.3%/±0.6%
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 801)。）
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压表示法 (页 801)。）

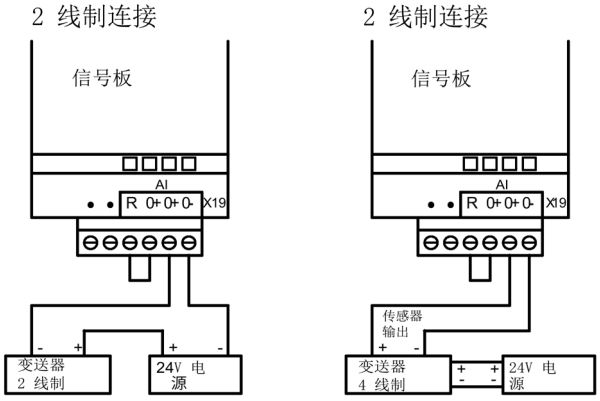
技术数据		**SB 1 点模拟量输入 (SB AE01)**
最大耐压/耐流		±35 V / ±40 mA
滤波		无、弱、中或强（参见阶跃响应下的模拟量输入响应时间 (页 800)。）
噪声抑制		400、60、50 或 10 Hz
共模抑制测量原理		40 dB，DC 到 60 Hz 抑制
工作信号范围（信号电压加共模电压）		信号加共模电压必须小于 +35 V 且大于 -35 V
输入阻抗		
	差模	220 KΩ（电压）/250 Ω（电流）
	共模	55 KΩ（电压）/55 KΩ（电流）
隔离（现场侧与逻辑侧）		无
电缆长度（米）		100 m 屏蔽双绞线

表格 A- 136 诊断

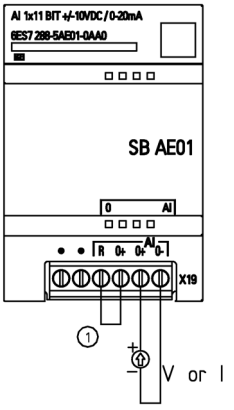
型号	**SB 1 点模拟量输入 (SB AE01)**
上溢/下溢	有
24 V DC 低压	无

SB AE01 接线电流变送器

接线电流变送器可用作 2 线制传感器和 4 线制变送器。



表格 A- 137 SB AE01 1 点模拟量输入 (6ES7288-5AE01-0AA0) 接线图

SB AE01 - SB 1 点模拟量输入 (6ES7288-5AE01-0AA0)	
	① 为电流应用连接“R”和“0+” 注：连接器必须镀金。有关产品编号，请参见附录 F“备件和其它硬件”。

表格 A- 138 SB AE01 1 点模拟量输入 (6ES7288-5AE01-0AA0) 的连接器引脚位置

引脚	X19
1	无连接
2	无连接
3	AI R
4	AI 0+
5	AI 0+
6	AI 0-

A.7.2 SB AQ01 模拟量输出规范

表格 A- 139 常规规范

技术数据	SB 1 点模拟量输出 (SB AQ01)
产品编号	6ES7288-5AQ01-0AA0
尺寸 W x H x D (mm)	35 x 52.2 x 16
重量	17.4 g
功耗	1.5 W
电流消耗 (5 V DC)	15 mA
电流消耗 (24 V DC)	40 mA（无负载）

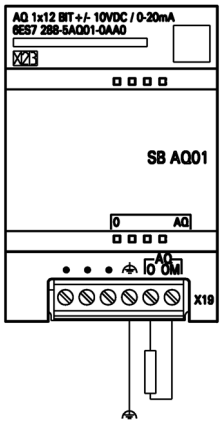
表格 A- 140 模拟量输出

技术数据	SB 1 点模拟量输出 (SB AQ01)
输出点数	1
类型	电压或电流
范围	±10 V，0 到 20 mA
分辨率	电压：11 位 + 符号 电流：11 位
满量程范围（数据字） 请参见电压和电流的输出范围。	-27,648 到 27,648（-10 V 到 10 V） 0 到 27,648（0 到 20 mA）
精度（25 °C/0 到 55 °C）	±0.5%/±1%
稳定时间（新值的 95%）	电压：300 µs (R)，750 µs (1 uF) 电流：600 µs (1mH)，2 ms (10 mH)
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	10 m 屏蔽双绞线

表格 A- 141 诊断

技术数据	SB 1 点模拟量输出 (SB AQ01)
上溢/下溢	有
对地短路（仅限电压模式）	有
断路（仅限电流模式）	有

表格 A- 142 SB AQ01 1 点模拟量输出 (6ES7288-5AQ01-0AA0) 的接线图

SB AQ01 1 点模拟量输出 (6ES7288-5AQ01-0AA0)	
	

表格 A- 143 SB AQ01 1 点模拟量输出 (6ES7288-5AQ01-0AA0) 的连接器引脚位置

引脚	X19
1	无连接
2	无连接
3	无连接
4	功能性接地
5	AQ 0
6	AQ 0M

A.8 RS485/RS232 信号板

A.8.1 SB RS485/RS232 规范

表格 A- 144 常规规范

技术数据	SB RS485/RS232
产品编号	6ES7288-5CM01-0AA0
尺寸 W x H x D (mm)	35 x 52.2 x 16
重量	18.2 g
功耗	0.5 W
电流消耗 (5 V DC)	50 mA
电流消耗 (24 V DC)	不适用

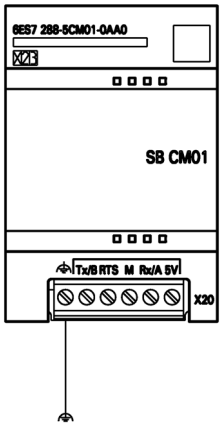
表格 A- 145 RS485 发送器和接收器

技术数据	SB RS485/RS232
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 12 K Ω
接收器阈值/灵敏度	最小 +/- 0.2 V，典型滞后 60 mV
隔离 RS 485 信号与外壳接地 RS485 信号与 CPU 逻辑公共端	无
电缆长度，屏蔽	带有隔离中继器：1000 m，最多 187.5 千波特 不带有隔离中继器：50 m

表格 A- 146 RS232 发送器和接收器

技术数据	**SB RS485/RS232**
发送器输出电压	RL = 3K Ω 时，最小 +/-5 V
传送输出电压	+/- 15 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	+/- 30 V DC ，最大值
隔离 RS232 信号与外壳接地 RS232 信号与 CPU 逻辑公共端	无
电缆长度，屏蔽	最长 10 m

表格 A- 147 SB CM01 RS485/RS232 (6ES7288-5CM01-0AA0) 的接线图

SB CM01 RS485/RS232 (6ES7288-5CM01-0AA0)	
	

表格 A- 148 SB CM01 RS485/RS232 (6ES7288-5CM01-0AA0) 的连接器引脚位置

引脚	X20
1	功能性接地
2	Tx/B
3	RTS
4	M
5	Rx/A
6	5 V 输出（偏置电压）

A.9 电池板信号板 (SB)

A.9.1 SB BA01 电池板

SB BA01 电池板

S7-200 SMART SB BA01 电池板适用于实时时钟的长期备份。其可插入 S7-200 SMART CPU（固件版本 V2.0 及更高版本）的信号板插槽中。必须将 SB BA01 添加到设备组态并将硬件配置下载到 CPU 中，SB BA01 才可以使用附加电池健康状况报告选项。

电池（型号 CR1025）未随 SB BA01 一起提供，用户必须另行购买。

说明

SB BA01 在机械设计上适合固件版本 V2.0 及以上版本的 CPU。

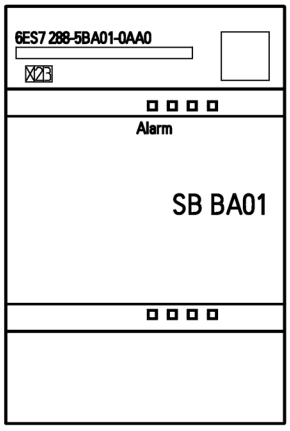
表格 A- 149 常规规范

技术数据	SB BA01 电池板
产品编号	6ES7288-5BA01-0AA0
尺寸 W x H x D (mm)	35 x 52.2 x 16
重量	20 g
功耗	0.6 W
电流消耗 (5 V DC)	18 mA
电流消耗 (24 V DC)	无

电池（未包含）	SB BA01 电池板
保持时间	大约 1 年
电池类型	CR1025 请参见安装或更换 SB BA01 电池板中的电池 (页 53)
额定电压	3 V
额定容量	30 mAH

诊断	SB BA01 电池板
临界电池电压	< 2.5 V
电池诊断	低压指示灯： • 电池电压低会使 BA01 面板上的 LED 呈红色常亮状态。 • 诊断报警和/或电量不足时数字量输出状态可用
电池状态	提供的电池状态位 0 = 电池正常 1 = 电池电量低
电池状态更新	电池状态会在开机时更新，之后在 CPU 处于 RUN 模式时，每天更新一次。

表格 A- 150 SB BA01 电池板 (6ES7288-5BA01-0AA0) 的接线图

SB BA01 电池板 (6ES7288-5BA01-0AA0)	
	

A.10 EM DP01 PROFIBUS DP 模块

表格 A- 151 常规规范

技术数据		**EM DP01 PROFIBUS DP**
产品编号		6ES7288-7DP01-0AA0
尺寸 W x H x D (mm)		70 x 100 x 81
重量		176.2 g
功耗		1.5 W（无负载）
V DC 要求		
	+5 V DC （SM 总线）	150 mA（无负载）
	+24 V DC	见下文

表格 A- 152 EM 特征参数

技术数据	**EM DP01 PROFIBUS DP 模块**
端口数量（有限功率）	1
电气接口	RS485
PROFIBUS DP/MPI 波特率（自动设置）	9.6、19.2、45.45、93.75、187.5 和 500 kbaud；1.5、3、6 和 12 Mbaud
协议	PROFIBUS DP 从站和 MPI 从站
电缆长度	
最大 93.7 kbaud	1200 m
187.5 kbaud	1000 m
500 kbaud	400 m
1 到 1.5 Mbaud	200 m
3 到 12 Mbaud	100 m
网络功能	
站地址设置	0 到 99（通过旋转开关设置）
每个网段最多站数	32

技术数据	EM DP01 PROFIBUS DP 模块
每个网络最多站数	126，最多 99 个 EM DP01 站
MPI 连接	总计 6 个（为 OP 保留 1 个）

表格 A- 153 电源

技术数据	EM DP01 PROFIBUS DP
24 V DC 输入电源要求	
电压范围	20.4 到 28.8 V DC （2 类受限制电源，或 PLC 提供的传感器电源）
最大电流： 仅端口激活的模块 加 5 V 端口负载的 90 mA 加 24 V 端口负载的 120 mA	 30 mA 60 mA 180 mA
波纹噪声 (< 10 MHz)	< 1 V 峰峰值（最大值）
隔离（现场侧与逻辑侧）[1]	500 V AC，持续 1 分钟
通信端口上 5 V DC 电源	
各端口最大电流：	额定值 5 V 时为 900 mA
电流限值	2.7 A @5 V
隔离（5 V DC 与逻辑侧）	500 V AC，持续 1 分钟
通信端口上 24 V DC 电源	
电压范围	20.4 到 28.8 V DC
各端口最大电流：	额定值 24 V 时为 120 mA
电流限值	0.7 到 2.4 A
隔离	无隔离，与 24 V DC 输入电路相同

[1] 24 V DC 电源不会给逻辑模块供电。24 V DC 电源给通信端口供电。

A.10.1 支持 EM DP01 PROFIBUS DP 模块的 S7-200 SMART CPU

S7-200 SMART EM DP01 PROFIBUS DP 模块是可与下表中 S7-200 SMART CPU 共用的智能扩展模块。

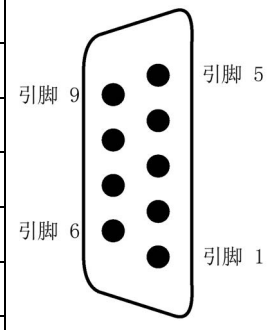
表格 A- 154 EM DP01 PROFIBUS DP 模块与 S7-200 SMART CPU 的兼容性

CPU	说明
ST20	CPU ST20 (DC/DC/DC)
SR20	CPU SR20 (AC/DC/Relay)
ST30	CPU ST30 (DC/DC/DC)
SR30	CPU SR30 (AC/DC/Relay)
ST40	CPU ST40 (DC/DC/DC)
SR40	CPU SR40 (AC/DC/Relay)
ST60	CPU ST60 (DC/DC/DC)
SR60	CPU SR60 (AC/DC/Relay)

A.10.2 EM DP01 连接器引脚分配

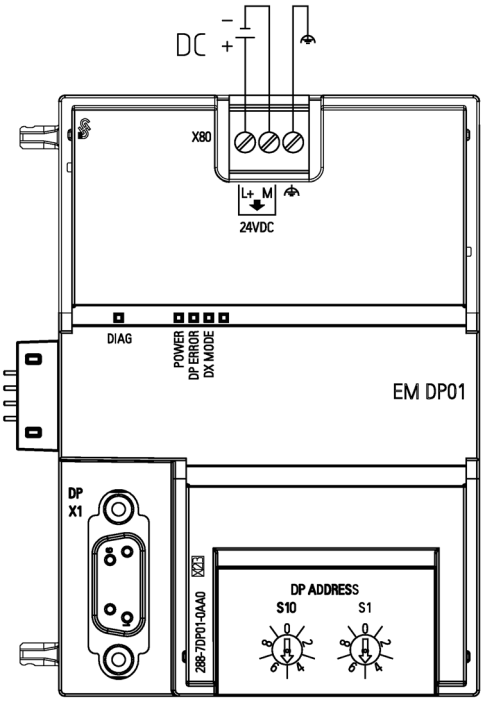
EM DP01 上的 RS485 串行通信接口是一个 RS485 兼容的九针迷你 D 型插口，与欧洲标准 EN 50170 规定的 PROFIBUS 标准一致。下表显示了为通信端口提供物理连接的连接器并介绍了通信端口的引脚分配。

表格 A- 155 S7-200 SMART EM DP01 的引脚分配

引脚编号	连接器	PROFIBUS
1		屏蔽
2		返回 24 V
3		RS485 信号 B
4		请求发送
5		返回 5 V
6		+5 V（隔离）
7		+24 V
8		RS485 信号 A
9		NC

A.10.3 EM DP01 PROFIBUS DP 模块接线图

表格 A- 156 EM DP01 PROFIBUS DP 模块 (6ES7288-7DP01-0AA0) 的接线图

EM DP01 PROFIBUS DP 模块 (6ES7288-7DP01-0AA0)	
	

表格 A- 157 EM DP01 PROFIBUS DP 模块 (6ES7288-7DP01-0AA0) 的连接器引脚位置

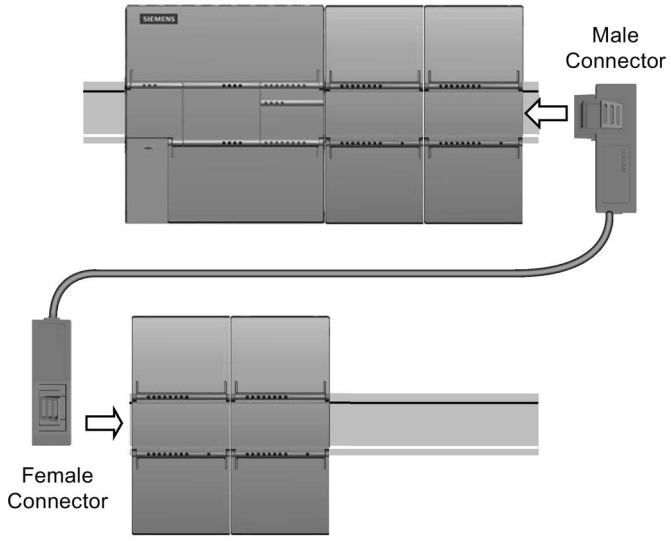
引脚	X80
1	L+ / 24 V DC
2	M / 24 V DC
3	功能性接地

A.11 S7-200 SMART I/O 扩展电缆

表格 A- 158 S7-200 SMART 扩展电缆

技术数据	
订货号	6ES7288-6EC01-0AA0
电缆长度	1 m
重量	80 g

有关安装和拆卸 S7-200 SMART 扩展电缆的信息，请参见安装部分。



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 的技术规范确定 24 V DC 传感器供电预算、CPU 所提供的 5 V DC 逻辑预算以及扩展模块和信号板的 5 V DC 电源要求。请参考计算功率预算，确定 CPU 能为您的组态提供多少电能（或电流）。

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。要格外注意系统组态以确保 CPU 可以提供所选扩展模块所需的 5 V DC 电源。如果组态要求的电源超出 CPU 提供的电源范围，则必须拆下一些模块。

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 也提供 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 24 V DC 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。

请参见具体 CPU (页 724) 的技术规范确定 24 V DC 传感器供电预算、CPU 所提供的 5 V DC 逻辑预算以及扩展模块和信号板的 5 V DC 电源要求。

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 SR40 AC/DC/继电器
- 3 个 EM 8 点继电器型数字量输出 (EM DR08)
- 1 个 EM 8 点数字量输入 (EM DE08)

该安装共有 32 点输入和 40 点输出。

说明

该 CPU 已分配驱动 CPU
内部继电器线圈所需的功率。功率预算计算中无需包括内部继电器线圈功率要求。

本例中的 CPU 提供了足够的 5 V DC
电流，但没有通过传感器电源为所有输入和扩展继电器线圈提供足够的 24 V DC
电流。I/O 需要 392 mA，但 CPU 提供了 300 mA。该安装额外需要一个至少为 92 mA
的 24 V DC 电源以运行所有包括的 24 V DC 输入和输出。

表格 B- 1　示例组态功率预算计算

CPU 功率预算	**5 V DC**	**24 V DC**
CPU SR40 AC/DC/继电器	1400 mA	300 mA
减去		
系统要求	**5 V DC**	**24 V DC**
CPU SR40，24 点输入		24 * 4 mA = 96 mA
插槽 0：EM DR08	120 mA	8 * 11 mA = 88 mA
插槽 1：EM DR08	120 mA	8 * 11 mA = 88 mA
插槽 2：EM DR08	120 mA	8 * 11 mA = 88 mA
插槽 3：EM DE08	105 mA	8 * 4 mA = 32 mA
总要求	465 mA	392 mA
等于		

CPU 功率预算	**5 V DC**	**24 V DC**
电流差额	**5 V DC**	**24 V DC**
总电流差额	275 mA	(92 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 719)。

表格 B- 2　功率预算

功率预算	**5 V DC**	**24 V DC**
减去		
系统要求	**5 V DC**	**24 V DC**
总要求		
等于		
电流差额	**5 V DC**	**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、删除或修改变量或全局符号。
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 SMAR 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。

- **运行时间编程错误**是在程序执行过程中由用户或用户程序造成的非致命错误条件。例如，编译程序期间间接地址指针有效，程序执行时被修改为指向超出范围的地址。访问 PLC 菜单功能区的“PLC 信息”(PLC Information) 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，运行时间编程错误会清除。

- **PLC 编译器错误**（或程序编译错误）将阻止您将程序下载到 PLC 中。当您编译或下载 (页 42)程序时，STEP 7-Micro/WIN SMART 将检测编译错误并在输出窗口显示检测到的错误。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

十六进制错误代码	非致命 PLC 程序编译器错误
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
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
009B	字符串操作的索引非法
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

十六进制错误代码	非致命运行时间编程问题
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	试图在脉冲输出有效时重新定义它
000E	PTO 包络段数已设置为 0
000F	在比较触点指令中遇到非法数字值
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
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）

参见

特殊存储器 (SM) 和系统符号名称 (页 851)

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 851)。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	SMB9	模块 0 I/O 错误字节
SM0.7	RTC_Lost	SMB11	模块 1 I/O 错误字节
SM1.3	除数为零错误	SMB13	模块 2 I/O 错误字节
SM3.0	奇偶校验错误	SMB15	模块 3 I/O 错误字节
SM4.0	通信中断队列溢出	SMB17	模块 4 I/O 错误字节
SM4.1	输入中断队列溢出	SMB19	模块 5 I/O 错误字节
SM4.2	定时中断队列溢出	SMB29	信号板 I/O 错误字节
SM4.3	运行时间编程问题		
SM5.0	I/O 错误（任何 I/O 错误位置位）		

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。

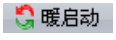
检测到致命错误时，PLC 执行下列任务。

- 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 

。

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 

。

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

特殊存储器 (SM) 和系统符号名称

D

D.1 SM（特殊存储器）概述

SMB0 至 SMB29、SMB480 至 SMB515 以及 SMB1000 至 SMB1699（S7-200 SMART 只读特殊存储器）

CPU
操作系统会将新的更改内容写入存储在特殊存储器中的系统数据。

从 CPU 中读取系统状态

程序中的 SMB0 至 SMB29、SMB480 至 SMB515 以及 SMB1000 至 SMB1699 为只读。如果程序尝试对只读 SM 地址执行写入操作，STEP 7-Micro/WIN SMART 在编译程序时将出现错误。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 854) 系统状态位
- SMB1 (页 855) 指令执行状态位
- SMB2 (页 856) 自由端口接收字符
- SMB3 (页 857) 自由端口奇偶校验错误
- SMB4 (页 858)
  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 859) I/O 错误状态位
- SMB6-SMB7 (页 859) CPU ID、错误状态和数字量 I/O 点
- SMB8-SMB19 (页 860)I/O 模块 ID 和错误
- SMW22-SMW26 (页 861) 扫描时间
- SMB28-SMB29 (页 862) 信号板 ID 和错误
- SMB480-SMB515 (页 878)数据日志状态（只读）

- SMB1000-SMB1049 (页 882) CPU 硬件/固件 ID
- SMB1050-SMB1099 (页 883) SB（信号板）硬件/固件 ID
- SMB1100-SMB1399 (页 884)EM（扩展模块）硬件/固件 ID
- SMB1400-SMB1699 (页 887) EM（扩展模块）模块特定的数据

SMB30 至 SMB194 以及 SMB566 至 SMB749（S7-200 SMART 读/写特殊存储器）

根据要求，S7-200 CPU
操作系统从特殊存储器读取组态/控制数据，并将新的更改内容写入存储在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 数据的正常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）和 SMB130（端口 1） (页 863)集成 RS485 端口（端口 0）和 CM01 信号板 (SB) RS232/RS485 端口（端口 1）的端口组态
- SMB34-SMB35 (页 864) 定时中断的时间间隔
- SMB36-45 (HSC0)、SMB46-55 (HSC1)、SMB56-65 (HSC2)、SMB136-145 (HSC3) (页 865) 高速计数器组态和操作
- SMB66-SMB85 (页 870) PLS0 和 PLS1 高速输出
- SMB86-SMB94 和 SMB186-SMB194 (页 874) 接收消息控制
- SMW98 (页 876) I/O 扩展总线通信错误
- SMW100-SMW114 (页 877) 系统报警
- CM01 信号板 (SB) RS232/RS485 端口（端口 1）的 SMB130 (页 863) 端口组态（请参见 SMB30）
- SMB136-SMB145 (页 878) HSC3 高速计数器（请参见 SMB36）
- SMB166-SMB169 (页 870) PTO0 包络定义表

- SMB176-SMB179 (页 870) PTO1 包络定义表
- SMB186-SMB194 (页 874) 接收消息控制（请参见 SMB86-SMB94）
- SMB566-SMB575 (页 870) PLS2 高速输出
- SMB576-SMB579 (页 870) PTO2 包络定义表
- SMB600-SMB649 (页 879) 轴 0 开环运动控制
- SMB650-SMB699 (页 881) 轴 1 开环运动控制
- SMB700-SMB749 (页 881) 轴 2 开环运动控制

⚠警告

使用绝对特殊存储器 (SM) 寻址的 STEP 7-Micro/WIN 版本 4.0 或更高版本（.mwp 文件）存在风险

如果较早版本的 STEP 7-Micro/WIN（.mwp 文件）使用 OB 中的符号 SM 寻址，且已生成系统符号表，则符号将正确映射到新地址。但是，如果 .mwp 文件使用 OB 中的绝对 SM 寻址，则那些绝对 SM 地址将不会映射到新 SM 地址。

如果 SM 地址的映射错误，则会导致意外的机械或过程操作，从而可能导致人员死亡、重伤和/或设备损坏。

删除“S7-200 符号”表并生成 SMART“系统符号”表。OB 中的符号将映射到 SMART 系统符号表中的新 SM 地址方案。

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 更新这些位。 程序可以读取这些位的状态，然后根据位值做出决定。

表格 D- 1 SMB0 系统状态位

S7-200 SMART 符号名	SM 地址	说明
Always_On	SM0.0	该位始终接通。（设置为 1）
First_Scan_On	SM0.1	该位在第一个扫描周期接通，然后断开。 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，该位会接通一个扫描周期： • 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）。 • NAND 闪存上保留的记录出现问题 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	从上电或暖启动条件进入 RUN 模式时，该位接通一个扫描周期。 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	该位提供时钟脉冲，该脉冲的周期时间为 1 分钟，OFF（断开）30 秒，ON（接通）30 秒。 该位可简单轻松地实现延时或 1 分钟时钟脉冲。
Clock_1s	SM0.5	该位提供时钟脉冲，该脉冲的周期时间为 1 秒，OFF（断开）0.5 秒，然后 ON（接通）0.5 秒。 该位可简单轻松地实现延时或 1 秒钟时钟脉冲。
Clock_Scan	SM0.6	该位是扫描周期时钟，接通一个扫描周期，然后断开一个扫描周期，在后续扫描中交替接通和断开。 该位可用作扫描计数器输入。
RTC_Lost	SM0.7	如果实时时钟设备的时间被重置或在上电时丢失（导致系统时间丢失），则该位将接通一个扫描周期。 该位可用作错误存储器位或用来调用特殊启动顺序。

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。 程序可以读取位值，然后根据值做出决定。

表格 D- 2 SMB1 指令执行状态位

S7-200 SMART 符号名	SM 地址	说明
Result_0	SM1.0	执行某些指令时，如果运算结果为零，该位将接通。
Overflow_Illegal	SM1.1	执行某些指令时，如果结果溢出或检测到非法数字值，该位将接通。
Neg_Result	SM1.2	数学运算得到负结果时，该位接通。
Divide_By_0	SM1.3	尝试除以零时，该位接通。
Table_Overflow	SM1.4	执行添表 (ATT) 指令时，如果参考数据表已满，该位将接通。
Table_Empty	SM1.5	LIFO 或 FIFO 指令尝试从空表读取时，该位接通。
Not_BCD	SM1.6	将 BCD 值转换为二进制值期间，如果值非法（非 BCD），该位将接通。
Not_Hex	SM1.7	将 ASCII 码转换十六进制 (ATH) 值期间，如果值非法（非十六进制 ASCII 数），该位将接通。

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。
在自由端口模式下接收的每个字符均放置到该位置，以便于程序访问。

表格 D- 3 SMB2 自由端口接收字符

S7-200 SMART 符号名	SM 地址	说明
Receive_Char	SMB2	该字节包含在自由端口通信过程中从端口 0 或端口 1 接收的各字符。

说明

SMB2 和 SMB3 供端口 0 与端口 1 之间共享

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 则包含该字符的错误状态。

D.5 SMB3： 自由端口字符错误

SMB3

用于自由端口模式，并且包含在接收字符中检测到奇偶校验、帧、中断或超限错误时所置位的错误位。 检测到奇偶校验、帧、中断或超限错误时，SM3.0 接通。

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 符号名	SM 地址	说明
Parity_Err	SM3.0	该位指示端口 0 或端口 1 上收到奇偶校验、帧、中断或超限错误。（0 = 无错误；1 = 有错误）

D.6 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

特殊存储器字节 4 (SM4.0 - SM4.7)
包含中断队列溢出位和一个指示中断是启用还是禁止的位 (SM 4.4)。
这些位指示中断发生速度比可处理速度快，或使用全局中断禁用指令禁用了中断。

其它位指示：

- 运行时程序错误
- 自由端口发送器状态
- 任何 PLC 存储器值当前是否被强制。

表格 D- 5 SMB4 系统状态

S7-200 SMART 符号名	SM 地址	说明
Comm_Int_Ovr	** SM4.0	1 = 通信中断队列已溢出。
Input_Int_Ovr	** SM4.1	1 = 输入中断队列已溢出。
Timed_Int_Ovr	** SM4.2	1 = 定时中断队列已溢出。
RUN_Err	SM4.3	1 = 检测到运行时间编程非致命错误。
Int_Enable	SM4.4	1 = 中断已启用。
Xmit0_Idle	SM4.5	1 = 端口 0 发送器空闲（0 = 正在传输）。
Xmit1_Idle	SM4.6	1 = 端口 1 发送器空闲（0 = 正在传输）。
Force_On	SM4.7	1 = 存储器位置被强制。

** 只能在中断例程中使用状态位 4.0、4.1 和 4.2。 队列变空时这些状态位复位，控制权返回到主程序。

D.7 SMB5： I/O 错误状态

特殊存储器字节 5 (SM5.0 - SM5.7) 包含用于指示在 I/O 系统中检测到的错误条件的状态位。 这些位概述了检测到的 I/O 错误。

表格 D- 6 SMB5 I/O 错误状态

S7-200 SMART 符号名	SM 地址	说明
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 符号名称	SM 地址	只读 SMB6 和 SMB7（CPU ID、错误状态和数字 I/O 点）								
CPU_ID	SMB6	MSB							LSB	
		7							0	
		1	x	x	x	c	d	0	0	
	SM6.4 – SM6.6		0	0	0	= reserved				
			0	0	1	= CPU CR40				
			0	1	0	= CPU CR60				
			0	1	1	= CPU SR20 / ST20				
			1	0	0	= CPU SR40 / ST40				
			1	0	1	= CPU SR60 / ST60				
			1	1	0	= reserved				
			1	1	1	= CPU SR30 / ST30				
	SM6.2 – SM6.3					c				组态/参数分配错误 0 = 无错误，1 = 错误
							d			报警诊断（请参见 SMW100 查看报警代码） 0 = 无错误，1 = 错误

CPU_IO	SMB7	MSB							LSB	
		7							0	
		i	i	i	i	q	q	q	q	
	SM7.0 – SM7.7	i	i	i	i					0 - 15 个字节的数字输入点
						q	q	q	q	0 - 15 个字节的数字输出点

另请参见 SMW100-SMW114 系统报警代码 (页 877)

D.9 SMB8-SMB19： I/O 模块 ID 和错误

SMB8 至 SMB19 以字节对的形式组织，用于扩展模块 0 至 5。

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 符号名	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（诊断报警代码参见 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（诊断报警代码参见 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（诊断报警代码参见 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（诊断报警代码参见 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（诊断报警代码参见 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（诊断报警代码参见 SMW114）

偶数字节 - I/O 模块 ID 寄存器

MSB							LSB
7							0
m	0	0	a	i	i	q	q

m： 模块是否存在
- 0 = 存在
- 1 = 不存在

a：I/O 类型
- 0 = Digital
- 1 = Analog

ii：输入
- 0 0 = 无输入
- 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
- 0 1 = 2 AQ 或 8 DQ
- 1 0 = 4 AQ 或16 DQ
- 1 1 = 8 AQ 或 32 DQ

奇数字节 - I/O 模块错误寄存器

MSB							LSB
7							0
c	d	0	b	0	0	0	m

c：
- 0 无错误
- 1 组态/参数化错误

d：
- 0 无错误
- 1 诊断报警

b：
- 0 无错误
- 1 总线访问错误

m：
- 0 OK
- 1 缺失已组态模块

另请参见 SMW100-SMW114 系统报警代码 (页 877)

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- 7 SMW22-SMW26 PLC 扫描时间

S7-200 SMART 符号名	SM 地址	说明
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。
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。

D.11 SMB28-SMB29： 信号板 ID 和错误

SMB28-SMB29 字节地址存储信号板类型和错误状态。

S7-200 SMART 符号名称	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（参见 SMW102 查看诊断报警代码）

	SMB28 信号板 ID 寄存器								SMB28 信号板错误寄存器								
	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：模块存在	0 = 存在 1 = 不存在								c:	0 无错误 1 组态/参数分配错误							
									d:	0 无错误 1 诊断报警							
a：I/O 类型	0 = 数字 1 = 模拟								b:	0 无错误 1 总线访问错误							
ii：输入	0 0 = 无输入 0 1 = 2 AI 或 8 DI 1 0 = 4 AI 或 16 DI 1 1 = 8 AI 或 32 DI								m:	0 正常 1 组态的信号板缺失							
qq：输出	0 0 = 无输出 0 1 = 2 AQ 或 8 DQ 1 0 = 4 AQ 或 16 DQ 1 1 = 8 AQ 或 32 DQ																

另请参见 SMW100-SMW110 系统报警代码 (页 877)

D.12 SMB30： （端口 0）和 SMB130： （端口 1）

SMB30 组态端口 0（板载 RS485 端口）。

SMB130 组态端口 1（可选 CM01 信号板）。

可对 SMB30 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 符号名	SM 地址		位格式	位格式 MSB LSB								
	Port 0	Port 1		7							0	
P0_Config	SMB30			p	p	d	b	b	b	m	m	
P1_Config		SMB130										
	SM30.6 - SM30.7	SM130.6 - SM130.7	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	d:	0	= 每个字符 8 个数据位							
				1	= 每个字符 7 个数据位							
	SM30.2 - SM30.4	SM130.2 - SM130.4	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					
P0_Config_0	SM30.0		mm:	0	0	= PPI 从站模式						
P1_Config_0		SM130.0		0	1	= 自由端口协议						
	SM30.1	SM130.1		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，将忽略位 2 至位 7。								

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 的时间间隔。可以指定从 1 ms 至 255 ms 的时间间隔（以 1 ms 为增量）。当相应的定时中断事件连接到中断程序时，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- 8 SMB34-SMB35 定时中断时间间隔

S7-200 SMART 符号名	SM 地址	说明
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 SMB36-45 (HSC0)、SMB46-55 (HSC1)、SMB56-65 (HSC2)、SMB136-145 (HSC3)：高速计数器

这些地址可为 HSC0、HSC1、HSC2 和 HSC3 提供高速计数器组态和操作。

表格 D- 9　高速计数器 0 组态和操作

S7-200 SMART 符号名	SM 地址	说明
HSC0_Status	**SMB36**	**HSC0 计数器状态** **注：** 仅当正在执行高速计数器事件触发的中断例程时，计数器状态位才有效。
	SM36.0–SM36.4	保留
HSC0_Status_5	SM36.5	HSC0 当前计数方向状态位： 1 = 加计数
HSC0_Status_6	SM36.6	HSC0 当前值等于预设值状态位： 1 = 相等
HSC0_Status_7	SM36.7	HSC0 当前值大于预置值状态位： 1 = 大于
HSC0_Ctrl	**SMB37**	**HSC0 计数器控制**
HSC0_Reset_Level	SM37.0	复位的有效电平控制位： 0 = 复位为高电平有效，1 = 复位为低电平有效
	SM37.1	保留
HSC0_Rate	SM37.2	HSC0 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
HSC0_Dir	SM37.3	HSC0 方向控制位： 1 = 加计数
HSC0_Dir_Update	SM37.4	HSC0 更新方向： 1 = 更新方向
HSC0_PV_Update	SM37.5	HSC0 更新预设值： 1 = 将新预设值写入 HSC0 预设值
HSC0_CV_Update	SM37.6	HSC0 更新当前值： 1 = 将新当前值写入 HSC0 当前值
HSC0_Enable	SM37.7	HSC0 使能位： 1 = 使能

S7-200 SMART 符号名	SM 地址	说明
HSC0_CV	**SMD38**	**HSC0 新当前值** SMD38 用于将 HSC0 当前值设置为您所选择的任何值。 要更新当前值，可将所需的新当前值写入 SMD38，将 SM37.6 设置为 1 并执行该 HSC 指令。 然后，新的当前值将写入 HSC0 的当前计数寄存器。
HSC0_PV	**SMD42**	**HSC0 新预设值** SMD42 用于将 HSC0 预设值设置为您所选择的任何值。 要更新当前值，可将所需的新当前值写入 SMD42，将 SM37.5 设置为 1 并执行该 HSC 指令。 然后，新的预设值将写入 HSC0 的预设寄存器。

表格 D- 10　高速计数器 1 组态和操作

S7-200 SMART 符号名	SM 地址	说明
HSC1_Status	**SMB46**	**HSC1 计数器状态** **注意：** 仅当正在执行高速计数器事件触发的中断例程时，计数器状态位才有效。
	SM46.0– SM46.4	保留
HSC1_Status_5	SM46.5	HSC1 当前计数方向状态位： 1 = 加计数
HSC1_Status_6	SM46.6	HSC1 当前值等于预设值状态位： 1 = 相等
HSC1_Status_7	SM46.7	HSC1 当前值大于预设值状态位： 1 = 大于
HSC1_Ctrl	**SMB47**	**HSC1 控制**
	SM47.0-SM47.2	保留
HSC1_Dir	SM47.3	HSC1 方向控制位： 1 = 加计数；0 = 减计数
HSC1_Dir_Update	SM47.4	HSC1 更新方向： 1 = 更新方向
HSC1_PV_Update	SM47.5	HSC1 更新预设值： 1 = 将新预设值写入 HSC1 预设值
HSC1_CV_Update	SM47.6	HSC1 更新当前值： 1 = 将新当前值写入 HSC1 当前值
HSC1_Enable	SM47.7	HSC1 使能位： 1 = 启用 HSC；0 = 禁用 HSC

S7-200 SMART 符号名	SM 地址	说明
HSC1_CV	**SMD48**	**HSC1 新当前值** SMD48 用于将 HSC1 当前值设置为您所选择的任何值。 要更新当前值，可将所需的新当前值写入 SMD48，将 SM47.6 设置为 1 并执行该 HSC 指令。 然后，新的当前值将写入 HSC1 的当前计数寄存器。
HSC1_PV	**SMD52**	**HSC1 新预设值** SMD52 用于将 HSC1 预设值设置为您所选择的任何值。 要更新当前值，可将所需的新当前值写入 SMD52，将 SM47.5 设置为 1 并执行该 HSC 指令。 然后，新的预设值将写入 HSC1 的预设寄存器。

表格 D- 11　高速计数器 2 组态和操作

S7-200 SMART 符号名	SM 地址	说明
HSC2_Status	**SMB56**	**HSC2 计数器状态** 注意： 仅当正在执行高速计数器事件触发的中断例程时，计数器状态位才有效。
	SM56.0– SM56.4	保留
HSC2_Status_5	SM56.5	HSC2 当前计数方向状态位： 1 = 加计数
HSC2_Status_6	SM56.6	HSC2 当前值等于预设值状态位： 1 = 相等
HSC2_Status_7	SM56.7	HSC2 当前值大于预设值状态位： 1 = 大于
HSC2_Ctrl	**SMB57**	**HSC2 控制**
HSC2_Reset_Level	SM57.0	HSC2 复位的有效电平控制： 0 = 复位为高电平有效；1 = 复位为低电平有效
	SM57.1	保留
HSC2_Rate	SM57.2	HSC2 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
HSC2_Dir	SM57.3	HSC2 方向控制位： 1 = 加计数

S7-200 SMART 符号名	SM 地址	说明
HSC2_Dir_Update	SM57.4	HSC2 更新方向： 1 = 更新方向
HSC2_PV_Update	SM57.5	HSC2 更新预设值： 1 = 将新预设值写入 HSC2 预设值
HSC2_CV_Update	SM57.6	HSC2 更新当前值： 1 = 将新当前值写入 HSC2 当前值
HSC2_Enable	SM57.7	HSC2 使能位： 1 = 使能
HSC2_CV	**SMD58**	**HSC2 新当前值** SMD58 用于将 HSC2 当前值设置为您所选择的任何值。 要更新当前值，可将所需的新当前值写入 SMD58，将 SM57.6 设置为 1 并执行该 HSC 指令。 然后，新的当前值将写入 HSC2 的当前计数寄存器。
HSC2_PV	**SMD62**	**HSC2 新预设值** SMD62 用于将 HSC2 预设值设置为您所选择的任何值。 要更新当前值，可将所需的新当前值写入 SMD62，将 SM57.5 设置为 1 并执行该 HSC 指令。 然后，新的预设值将写入 HSC2 的预设寄存器。

表格 D- 12　高速计数器 3 组态和操作

S7-200 SMART 符号名	SM 地址	说明
HSC3_Status	**SMB136**	**HSC3 计数器状态** **注意：** 仅当正在执行高速计数器事件触发的中断例程时，计数器状态位才有效。
	SM136.0– SM136.4	保留
HSC3_Status_5	SM136.5	HSC3 当前计数方向状态位： 1 = 加计数
HSC3_Status_6	SM136.6	HSC3 当前值等于预设值状态位： 1 = 相等
HSC3_Status_7	SM136.7	HSC3 当前值大于预设值状态位： 1 = 大于
HSC3_Ctrl	**SMB137**	**HSC3 计数器控制**
	SM137.0– SM137.2	保留

S7-200 SMART 符号名	SM 地址	说明
HSC3Dir	SM137.3	HSC3 方向控制位： 1 = 加计数
HSC3_Dir_Update	SM137.4	HSC3 更新方向： 1 = 更新方向
HSC3_PV_Update	SM137.5	HSC3 更新预设值： 1 = 将新预设值写入 HSC3 预设值
HSC3_CV_Update	SM137.6	HSC3 更新当前值： 1 = 将新当前值写入 HSC3 当前值
HSC3_Enable	SM137.7	HSC3 使能位： 1 = 使能
HSC3_CV	**SMD138**	**HSC3 新当前值** SMD138 用于将 HSC3 当前值设置为您所选择的任何值。 要更新当前值，可将所需的新当前值写入 SMD138，将 SM137.6 设置为 1 并执行该 HSC 指令。 然后，新的当前值将写入 HSC3 的当前计数寄存器。
HSC3_PV	**SMD142**	**HSC3 新预设值** SMD142 用于将 HSC3 预设值设置为您所选择的任何值。 要更新当前值，可将所需的新当前值写入 SMD142，将 SM137.5 设置为 1 并执行该 HSC 指令。 然后，新的预设值将写入 HSC3 的预设寄存器。

D.15 SMB66-SMB85、SMB166-SMB169、SMB176-SMB179 和 SMB566-SMB579：PTO0、PWM0、PTO1、PWM1、PTO2 和 PWM2 高速输出

对于 PLS（脉冲）指令来说，S7-200 SMART CPU 使用 SMB66-SMB85、SMB166-SMB169和 SMB176-SMB179 来监视与控制脉冲串输出（PTO0 和 PTO1）和脉宽调制输出（PWM0 和 PWM1）。

SMB566-SMB579 用来监视与控制脉冲串输出 PTO2 及脉宽调制输出 PWM2。

表格 D- 13　高速输出 0 组态和控制

S7-200 SMART 符号名	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：0= 未中止；1= 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：0= 未禁止；1= 手动禁止
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：0= 未上溢；1= 管道上溢/下溢
PLS0_Idle	SM66.7	PTO0 空闲：0=PTO 进行中；1=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：0= 未更新；1= 写入新周期时间/频率
PWM0_PW_Update	SM67.1	PWM0 更新脉宽值：0= 未更新；1= 写入新脉宽
PTO0_PC_Update	SM67.2	PTO0 更新脉冲计数值：0= 未更新；1= 写入新脉冲计数
PWM0_TimeBase	SM67.3	PWM0 时基：0 = 1 μs/刻度；1 = 1 ms/刻度
	SM67.4	保留
PTO0_Operation	SM67.5	PTO0 选择单/多段操作：0 = 单段；1= 多段
PLS0_Select	SM67.6	PTO0/PWM0 模式选择：0=PWM；1=PTO
PLS0_Enable	SM67.7	PTO0/PWM0 使能：0 = 禁用；1 = 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 65,535Hz）**
PWM0_PW	**SMW70**	**字数据类型：PWM0 脉宽值（0 到 65,535 单位的时基）**

S7-200 SMART 符号名	SM 地址	功能
PTO0_PC	SMD72	双字数据类型：PTO0 脉冲计数值（1 到 2^31 -1）
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- 14　高速输出 1 组态和控制

S7-200 SMART 符号名	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：0= 未中止；1= 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：0= 未禁止；1= 手动禁止
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：0= 未上溢；1= 管道上溢/下溢
PLS1_Idle	SM76.7	PTO1 空闲：0=PTO 进行中；1=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）
PLS1_Cycle_Update	SM77.0	PTO1/PWM1 更新周期时间或频率值：0= 未更新；1= 写入新周期时间/频率
PWM1_PW_Update	SM77.1	PWM1 更新脉宽值：0= 未更新；1= 写入新脉宽
PTO1_PC_Update	SM77.2	PTO1 更新脉冲计数值：0= 未更新；1= 写入新脉冲计数
PWM1_TimeBase	SM77.3	PWM1 时基：0 = 1 μs/刻度；1 = 1 ms/刻度
	SM77.4	保留
PTO1_Operation	SM77.5	PTO1 选择单/多段操作：0 = 单段；1= 多段
PLS1_Select	SM77.6	PTO1/PWM1 模式选择：0=PWM；1=PTO
PLS1_Enable	SM77.7	PTO1/PWM1 使能：0 = 禁用；1 = 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 65,535Hz）
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 2^31 -1）

S7-200 SMART 符号名	SM 地址	功能
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- 15　高速输出 2 组态和控制

S7-200 SMART 符号名	SM 地址	说明
PTO2_Status	SMB566	PTO2 状态
PLS2_Abort_AE	SM566.4	PTO2 包络因相加错误而中止：0= 未中止；1= 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：0= 未禁止；1= 手动禁止
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：0= 未上溢；1= 管道上溢/下溢
PLS2_Idle	SM566.7	PTO2 空闲：0=PTO 进行中；1=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：0= 未更新；1= 写入新周期时间/频率
PWM2_PW_Update	SM567.1	PWM2 更新脉宽值：0= 未更新；1= 写入新脉宽
PTO2_PC_Update	SM567.2	PTO2 更新脉冲计数值：0= 未更新；1= 写入新脉冲计数
PLS2_TimeBase	SM567.3	PWM2 时基：0 = 1 μs/刻度；1 = 1 ms/刻度
	SM567.4	保留
PTO2_Operation	SM567.5	PTO2 选择单/多段操作：0 = 单段；1= 多段
PLS2_Select	SM567.6	PTO2/PWM2 模块选择：0=PWM；1=PTO
PLS2_Enable	SM567.7	PTO2/PWM2 使能：0 = 禁用；1 = 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 65,535Hz）
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 2^31 -1）

S7-200 SMART 符号名	SM 地址	说明
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）

D.16 SMB86-SMB94 和 SMB186-SMB194： 接收信息控制

SMB86-SMB94 和 SMB186-SMB194 用于控制和读取 RCV（接收消息）指令的状态。

S7-200 SMART 符号名称	SM 地址		位格式	接收消息状态字节 MSB LSB								
	端口 0	端口 1		7							0	
P0_Stat_Rcv	**SMB86**			n	r	e	0	0	t	c	p	
P1_Stat_Rcv		**SMB186**										
P0_Stat_Rcv_7	SM86.7		n:	1	= 接收消息被用户禁用命令终止，否则 n = 0							
P1_Stat_Rcv_7		SM186.7										
P0_Stat_Rcv_6	SM86.6		r:	1	= 接收消息终止：（未定义开始条件，字符计数为 0，或在传送激活情况下执行消息接收）否则，r = 0							
P1_Stat_Rcv_6		SM186.6										
P0_Stat_Rcv_5	SM86.5		e:	1	= 接收到结束字符，否则 e = 0							
P1_Stat_Rcv_5		SM186.5										
P0_Stat_Rcv_2	SM86.2		t:	1	= 接收消息终止：定时器终止，否则 t =0							
P1_Stat_Rcv_2		SM186.2										
P0_Stat_Rcv_1	SM86.1		c:	1	= 接收消息终止：达到最大字符数计数，否则 c =0							
P1_Stat_Rcv_1		SM186.1										
P0_Stat_Rcv_0	SM86.0		p:	1	= 终止接收消息，原因可能为奇偶校验、组帧、中断或超限错误，否则 p = 0							
P1_Stat_Rcv_0		SM186.0										

			位格式	接收消息控制字节 MSB LSB								
	端口 0	端口 1		7							0	
P0_Ctrl_Rcv	SMB87			en	sc	ec	il	c/m	Tm	r	bk	0
P1_Ctrl_Rcv		SMB187										
P0_Ctrl_Rcv_7	SM87.7		en:	0	= 禁用接收消息功能	= 启用接收消息功						
P1_Ctrl_Rcv_7		SM187.7		1	能							
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 的值检测消息的开始							
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							
P0_Ctrl_Rcv_4	SM87.4		il:	0	= 忽略 SMW90 或 SMW190							
P1_Ctrl_Rcv_4		SM187.4		1	= 使用 SMW90 或 SMW190 的值检测空闲线路 条件							
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							
P0_Ctrl_Rcv_2	SM87.2		tmr:	0	= 忽略 SMW92 或 SMW192							
P1_Ctrl_Rcv_2		SM187.2		1	= 在超出 SMW92 或 SMW192 的时长时终止接收							
P0_Ctrl_Rcv_1	SM87.1		bk:	0	= 忽略中断条件							
P1_Ctrl_Rcv_1		SM187.1		1	= 使用中断条件作为消息检测的开始							

消息控制字节的位用于定义识别信息的条件。 定义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（连续是指线空闲后出现起始字符或中断后出现起始字符），则可确定消息开始。
要确定消息结束，对启用的消息结束条件执行逻辑 OR（或）操作。
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 符号名	SM 地址		Bit 格式	设定消息起始条件 MSB							LSB	
	端口 0	端口 1		7							0	
P0_Ctrl_Rcv	SMB87			en	sc	ec	il	c/m	Tm r	bk	0	
P1_Ctrl_Rcv		SMB187										
空闲线检测					0		1			0		SMW90 > 0
起始字符检测					1		0			0		SMW90 = 任意'值
终端检测					0		0			1		SMW90 = 任意'值
任何针对请求的响应					0		1			0		SMW90 = 0 （如果没有响应，则可通过消息定时器终止接收）
中断及起始字符					1		0			1		SMW90 = 任意'值
空闲线和起始字符					1		1			0		SMW90 > 0
空闲线和起始字符（非法）					1		1			0		SMW90 = 0
注： 接收会因校验错误、帧错误、超限错误或中断错误而自动终止。												

S7-200 SMART 符号名	SM 地址		
	端口 0	端口 1	
P0_Start_Char	SMB88		消息字符开始。
P1_Start_Char		SMB188	
P0_End_Char	SMB89		消息字符结束。
P1_End_Char		SMB189	
P0_Idle_Time	SMW90		字类型数据：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首个字符为新消息的起始字符。
P1_Idle_Time		SMW190	
P0_Timeout	SMW92		字类型数据： 以毫秒为单位指定字符间/消息定时器的超时值。 如果超出该时间段，则终止接收消息。
P1_Timeout		SMW192	
P0_Max_Char	SMB94		要接收的最大字符数（1 至 255 字节）。 注： 即使未使用字符计数消息终止，此范围也必须设置为所需的最大缓冲区大小。
P1_Max_Char		SMB194	

D.17 SMW98： I/O 扩展总线通信错误

SMW98 提供有关扩展 I/O 总线错误的信息。

表格 D- 16 SMW98 I/O 扩展总线通信错误计数器

S7-200 SMART 符号名	SM 地址（读/写）	说明
EM_Parity_Err	SMW98	每次检测到扩展 I/O 总线发生奇偶校验错误时，该字的值将加 1。上电和用户写入零时，该字将清零。

D.18 SMW100-SMW114 系统报警

特殊存储器字 SMW100-SMW114 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 符号名称	SM 地址	当前诊断报警源
CPU_Alarm	SMW100	CPU 诊断报警代码
SB_Alarm	SMW102	信号板诊断报警代码
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

报警代码格式	MSB															LSB	
	15	14	13					8	7							0	
	d	s	c	c	c	c	c	c	a	a	a	a	a	a	a	a	
d：报警方向	0	输入通道或不适用的输出通道															
	1																
s：报警范围	0	在单个通道上															
	1	在整个模块上															
c：通道编号	c	c	c	c	c	c	如果报警范围 = “在单个通道上”，为受影响的通道编号，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：保留								
	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：保留								
	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：保留								
	0	0	1	0	0	0	0	0	20H：内部错误（MID 问题）21 H：内								
	0	0	1	0	0	0	0	1	部错误（IID 问题）								
	0	0	1	0	0	0	1	0	22H：保留								
	0	0	1	0	0	0	1	1	23H：组态错误								
	0	0	1	0	0	1	0	0	24H：保留								
	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：保留								
	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：保留								

D.19 SMB130： 端口 1 的自由端口控制（请参见 SMB30）

有关详细信息，请参见“SMB30：（端口 0）和 SMB130：（端口 1）” (页 863)。

D.20 SMB136-SMB145： HSC3 高速计数器

有关详细信息，请参见“SMB36-45 (HSC0)、SMB46-55 (HSC1)、SMB56-65 (HSC2)、SMB136-145 (HSC3)：高速计数器” (页 865)。

D.21 SMB186-SMB194： 接收消息控制（请参见 SMB86-SMB94）

有关详细信息，请参见“SMB86-SMB94 和 SMB186-SMB194” (页 874)。

D.22 SMB480-SMB515： 数据日志状态

SMB480 至 SMB515 为只读特殊存储器地址，用于监视数据日志的操作状态。

S7-200 SMART 符号名	SM 地址	功能
DL0_InitResult	SMB480	数据日志 0 的初始化结果代码： 数据日志分析在上电及下载系统块之后执行。 • 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：保留 • 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

S7-200 SMART 符号名	SM 地址	功能
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-SMB749： 轴（0、1 和 2）开环运动控制

轴组态和控制 SM 地址

此轴特殊存储器数据通常由向导生成的程序代码进行读写。

轴数据 SM 地址			轴功能
轴 0	轴 1	轴 2	
SMB600 - SMB615	SMB650 - SMB665	SMB600 - SMB615	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	SMB616 - SMB619	SMB616 - SMB619	保留
SMW620	SMW670	SMW720	错误代码 - 请参见运动轴错误代码 (页 694)

轴 0 **SMB622**	轴 1 **SMB672**	轴 2 **SMB722**		MSB 7	6	5	4	3	2	1	LSB 0
				DIS	0	**TRIG**	**STP**	**LMT-**	**LM +**	**RPS**	**ZP**
SM622.7	SM672.7	SM722.7	**DIS**: 禁用输出	0	无电流						
				1	电流						
SM622.5	SM672.5	SM722.5	**TRIG**：触发 输入			0	未激活				
						1	激活				
SM622.4	SM672.4	SM722.4	**STP** ： 停止输入					0	未激活		
								1	激活		
SM622.3	SM672.3	SM722.3	**LMT -** ： 负向移动限制输入							0	未激活
										1	激活
SM622.2	SM672.2	SM722.2	**LMT +** ： 正向移动限制输入							0	未激活
										1	激活
SM622.1	SM672.1	SM722.1	**RPS** ： 参考点开关输入								0 未激活
											1 激活
SM622.0	SM672.0	SM722.0	**ZP** ： 零脉冲输入								0 未激活
											1 激活

轴 0 **SMB623**	轴 1 **SMB673**	轴 2 **SMB723**		MSB 7	6	5	4	3	2	1	LSB 0
				0	0	0	0	0	**OR**	**R**	**CFG**
SM622.2	SM672.2	SM722.2	OR：目标速度是否超出范围						0	在范围内	
									1	超出范围	
SM622.1	SM672.1	SM722.1	R：旋转方向							0	正向旋转
										1	反向旋转
SM622.0	SM672.0	SM722.0	CFG： 轴是否已组态								0 未组态
											1 已组态

轴数据 SM 地址			轴功能
轴 0	**轴 1**	**轴 2**	
SMB624	SMB674	SMB724	**CUR_PF** 是一个指示当前正在执行的包络的字节。
SMB625	SMB675	SMB725	**CUR_STP** 是一个指示当前正在包络中执行的步的字节。
SMD626	SMD676	SMD726	**CUR_POS** 是指示轴当前位置的双字值。
SMD630	SMD680	SMD730	**CUR_SPD** 是指示轴当前速度的双字值。

轴 0 **SMB634**	轴 1 **SMB684**	轴 2 **SMB734**		MSB 7	6	5	4	3	2	1	LSB 0
				D	错误：						
SM634.7	SM684.7	SM734.7	**D**： “完成”位	0							
				1	操作完成（初始化过程中通过轴置位）						
SM634.0-SM634.6	SM684.0-SM684.6	SM734.0-SM734.6	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	
SMB635-SMB645	SMB685-SMB695	SMB735-SMB745	保留
SMD646	SMD646	SMD746	指向轴的组态/包络表的 V 存储器指针。 不接受指向 V 存储器以外其它区域的指针值。

D.24 SMB650-SMB699： 轴 1 开环运动控制（请参见 SMB600-SMB740）

有关详细信息，请参见“SMB600-SMB749：轴（0、1 和 2）开环运动控制 (页 879)”。

D.25 SMB700-SMB749： 轴 2 开环运动控制（请参见 SMB600-SMB740）

有关详细信息，请参见“SMB600-SMB749：轴（0、1 和 2）开环运动控制 (页 879)”。

D.26 SMB1000-SMB1049： CPU 硬件/固件 ID

在上电或暖启动切换后，此 CPU 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： （始终为 0x002A）
SMB1002 至 SMB1021	CPU 订货号 (MLFB)： ASCII 字符，在字段中左对齐，用空格补位
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 是 ASCII“V”；字节 1 = 功能版本；字节 2 = 次要更改版本；字节 3 = 错误修补版本（字节 1-3 的范围 = 0x00 至 0xFF）
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.27 SMB1050-SMB1099： SB（信号板）硬件/固件 ID

在上电或热重启切换后，此信号板信息写入特殊存储器。 特殊存储器的 SMB1050-SMB1099 部分为只读。

SM 地址	说明
SMW1050	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	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	信号板序列号： ASCII 字符，在字段中左对齐，用空格补位
SMW1088	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	信号板固件版本；字节 0 是 ASCII“V”；字节 1 = 功能版本；字节 2 = 次要更改版本；字节 3 = 错误修补版本（字节 1-3 的范围 = 0x00 至 0xFF）
SMW1094	信号板固件版本计数器（范围 0x0000 至 0x00FF）
SMW1096	保留，始终为 0x0000
SMW1098	信号板设备类型： I/O = 0x0003，通信 = 0x0004，所有其它值保留

D.28 SMB1100-SMB1399： EM（扩展模块）硬件/固件 ID

在上电或暖重启切换后，此扩展模块信息写入特殊存储器。 特殊存储器的 SMB1100-SMB1399 部分为只读。

插槽 0 的 SM 地址	说明
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 是 ASCII“V”；字节 1 = 功能版本；字节 2 = 次要更改版本；字节 3 = 错误修补版本（字节 1-3 的范围 = 0x00 至 0xFF）
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，所有其它值保留

插槽 1 的 SM 地址	说明
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
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 是 ASCII“V”；字节 1 = 功能版本；字节 2 = 次要更改版本；字节 3 = 错误修补版本（字节 1-3 的范围 = 0x00 至 0xFF）
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，所有其它值保留

插槽 2 的 SM 地址	说明
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 是 ASCII“V”；字节 1 = 功能版本；字节 2 = 次要更改版本；字节 3 = 错误修补版本（字节 1-3 的范围 = 0x00 至 0xFF）
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，所有其它值保留

插槽 3 的 SM 地址	说明
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
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 是 ASCII“V”；字节 1 = 功能版本；字节 2 = 次要更改版本；字节 3 = 错误修补版本（字节 1 - 3 的范围 = 0x00 至 0xFF）
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，所有其它值保留

插槽 4 的 SM 地址	说明
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 是 ASCII“V”；字节 1 = 功能版本；字节 2 = 次要更改版本；字节 3 = 错误修补版本（字节 1 - 3 的范围 = 0x00 至 0xFF）
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，所有其它值保留

插槽 5 的 SM 地址	说明
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
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 是 ASCII“V”；字节 1 = 功能版本；字节 2 = 次要更改版本；字节 3 = 错误修补版本（字节 1 - 3 的范围 = 0x00 至 0xFF）
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，所有其它值保留

D.29 SMB1400-SMB1699：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	说明
SMB1400 至 SMB1449	EM 总线槽 0：模块特定的信息
SMB1450 至 SMB1499	EM 总线槽 1：模块特定的信息
SMB1500 至 SMB1549	EM 总线槽 2：模块特定的信息
SMB1550 至 SMB1599	EM 总线槽 3：模块特定的信息
SMB1600 至 SMB1649	EM 总线槽 4：模块特定的信息
SMB1650 至 SMB1699	EM 总线槽 5：模块特定的信息

参考

E

E.1 常用特殊存储器位

在系统符号表中，完整列出了可用于您的项目的预定义特殊存储器程序符号。

表格 E- 1 常用特殊存储器位

SM 地址	系统符号名称	说明
SM0.0	Always_On	始终接通
SM0.1	First_Scan_On	仅在首次扫描周期接通
SM0.2	Retentive_Lost	如果保持数据丢失，接通一个扫描周期
SM0.3	RUN_Power_Up	从上电进入 RUN 模式时，接通 1 个扫描周期
SM0.4	Clock_60s	针对 1 分钟的周期时间，时钟脉冲接通 30 s，断开 30 s。
SM0.5	Clock_1s	针对 1 s 的周期时间，时钟脉冲接通 0.5 s，断开 0.5 s。
SM0.6	Clock_Scan	扫描周期时钟，一个扫描周期接通，下一个扫描周期关断
SM0.7	RTC_Lost	如果实时时钟设备的时间被重置或在上电时丢失（导致系统时间丢失），则该位将接通一个扫描周期。 该位可用作错误存储器位或用来调用特殊启动顺序。
SM1.0	Result_0	特定指令的操作结果 = 0 时，置位为 1
SM1.1	Overflow_Illegal	特定指令执行结果溢出或数值非法时，置位为 1
SM1.2	Neg_Result	当数学运算产生负数结果时，置位为 1
SM1.3	Divide_By_0	尝试除以零时，置位为 1
SM1.4	Table_Overflow	当填表指令尝试过度填充表格时，置位为 1
SM1.5	Table_Empty	当 LIFO 或 FIFO 指令尝试从空表读取时，置位为 1
SM1.6	Not_BCD	尝试将非 BCD 数值转换为二进制数值时，置位为 1
SM1.7	Not_Hex	当 ASCII 数值无法被转换为有效十六进制数值时，置位为 1

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	说明
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
离散 中等优先级	19	PTO0 完成中断
	20	PTO1 完成中断
	34	PTO2 完成中断
	0	I0.0 上升沿
	2	I0.1 上升沿
	4	I0.2 上升沿
	6	I0.3 上升沿
	35	I7.0 上升沿（信号板）
	37	I7.1 上升沿（信号板）
	1	I0.0 下降沿
	3	I0.1 下降沿
	5	I0.2 下降沿
	7	I0.3 下降沿
	36	I7.0 下降沿（信号板）
	38	I7.1 下降沿（信号板）
	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）
	16	HSC2 CV=PV（当前值 = 预设值）

优先级组	事件	说明
	17	HSC2 方向改变
	18	HSC2 外部复位
	32	HSC3 CV=PV（当前值 = 预设值）
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

E.3 高速计数器汇总

表格 E- 3 S7-200 SMART HSC 输入分配和功能

	时钟 A	Dir/时钟 B	复位	单相最大时钟/输入速率	双相/AB 正交相最大时钟/输入速率
HSC0	I0.0	I0.1	I0.4	200 kHz（S 型号 CPU）[1] 100 kHz（C 型号 CPU）[2]	100 kHz（S 型号 CPU） （最大 1 倍计数速率 = 100 kHz） （最大 4 倍计数速率 = 400 kHz） 50 kHz（C 型号 CPU） （最大 1 倍计数速率 = 50 kHz） （最大 4 倍计数速率 = 200 kHz）
HSC1	I0.1			200 kHz（S 型号 CPU） 100 kHz（C 型号 CPU）	
HSC2	I0.2	I0.3	I0.5	200 kHz（S 型号 CPU） 100 kHz（C 型号 CPU）	100 kHz（S 型号 CPU） （最大 1 倍计数速率 = 100 kHz） （最大 4 倍计数速率 = 400 kHz） 50 kHz（C 型号 CPU） （最大 1 倍计数速率 = 50 kHz） （最大 4 倍计数速率 = 200 kHz）
HSC3	I0.3			200 kHz（S 型号 CPU） 100 kHz（C 型号 CPU）	

[1] S 型号 CPU： SR20/ST20、SR30/ST30、SR40/ST40、SR60/ST60

[2] C 型号 CPU： CR40、CR60

E.4 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 (页 171)一章。

布尔指令	
STL	**说明**
LD 位	加载
LDI 位	立即加载
LDN 位	取反后加载
LDNI 位	取反后立即加载
A 位	AND
AI 位	立即与
AN 位	与非
ANI 位	立即与非
O 位	或
OI 位	立即或
ON 位	或非
ONI 位	立即或非
LDBx IN1, IN2	加载字节比较结果 IN1 (x:<, <=,=, >=, >, <>I) IN2
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

布尔指令	
STL	**说明**
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 ED	上升沿检测 下降沿检测
= 位 =I bit	赋值 立即赋值
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

布尔指令	
STL	**说明**
ALD	与装载
OLD	或装载
LPS	逻辑进栈（堆栈控制）
LRD	逻辑读取（堆栈控制）
LPP	逻辑出栈（堆栈控制）
LDS n	载入堆栈（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	**说明**
+I IN1, OUT +D IN1, OUT +R IN1, OUT	整数、双整数或实数加法 IN1+OUT=OUT
-I IN1, OUT -D IN1, OUT -R IN1, OUT	整数、双整数或实数减法 OUT-IN1=OUT
MUL IN1, OUT	整数乘法（16*16->32）
*I IN1, OUT *D IN1, OUT *R IN1, IN2	整数、双整数或实数乘法 IN1 * OUT = OUT
DIV IN1, OUT	整数除法（16/16->32）
/I IN1, OUT /D,IN1, OUT /R IN1, OUT	整数、双整数或实数除法 OUT / 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

算术运算、递增和递减指令	
STL	**说明**
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	**说明**
TON Txxx, PT	接通延时定时器
TOF Txxx, PT	断开延时定时器
TONR Txxx, PT	保持型接通延时定时器
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

高速指令	
STL	**说明**
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	**说明**
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	**说明**
END	程序有条件结束
STOP	转入 STOP（停止）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	**说明**

MOVB IN, OUT
MOVW IN, OUT
MOVD IN, OUT
MOVR IN, OUT
	传送字节、字、双字和实数

BIR IN, OUT BIW IN, OUT	传送字节立即读取 传送字节立即写入
BMB IN, OUT, N BMW IN, OUT, N BMD IN, OUT, N	字节、字和双字块传送
SWAP IN	字节交换
SHRB DATA, S_BIT, N	移位寄存器位
SRB OUT, N SRW OUT, N SRD OUT, N	右移字节、字和双字
SLB OUT, N SLW OUT, N SLD OUT, N	左移字节、字和双字
RRB OUT, N RRW OUT, N RRD OUT, N	循环右移字节、字和双字
RLB OUT, N RLW OUT, N RLD OUT, N	循环左移字节、字和双字

逻辑指令	
STL	**说明**
ANDB IN1, OUT ANDW IN1, OUT ANDD IN1, OUT	对字节、字和双字执行逻辑与运算
ORB IN1, OUT ORW IN1, OUT ORD IN1, OUT	对字节、字和双字执行逻辑或运算
XORB IN1, OUT XORW IN1, OUT XORD IN1, OUT	对字节、字和双字执行逻辑异或运算
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	**说明**
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	**说明**
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	**说明**
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 将整数转换为双整数 将双整数转换为整数
DTR IN, OUT TRUNC IN, OUT ROUND IN, OUT	将双字转换为实数 将实数转换为双整数 将实数转换为双整数
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 将子字符串转换为双整数 将子字符串转换为实数

中断指令	
STL	**说明**
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 存储器范围和特性

说明		CPU SR20、CPU ST20	CPU SR30、CPU ST30	CPU SR40、CPU ST40	CPU CR40、CPU CR60	CPU SR60、CPU ST60
用户程序大小		12 KB	18 KB	24 KB	12 KB	30 KB
用户数据大小		8 KB	12 KB	16 KB	8 KB	20 KB
过程映像输入寄存器		I0.0 至 I31.7	I0.0 到 I31.7	I0.0 到 I31.7	I0.0 到 I31.7	I0.0 到 I31.7
过程映像输出寄存器		Q0.0 到 Q31.7	Q0.0 到 Q31.7	Q0.0 到 Q31.7	Q0.0 到 Q31.7	Q0.0 到 Q31.7
模拟量输入（只读）		AIW0 到 AIW110	AIQ0 到 AIW110	AIW0 到 AIW110	--	AIW0 到 AIW110
模拟量输出（只写）		AQW0 到 AQW110	AQW0 到 AQW110	AQW0 到 AQW110	---	AQW0 到 AQW110
变量存储器 (V)		VB0 到 VB8191	VB0 到 VB12287	VB0 到 VB16383	VB0 到 VB8191	VB0 到 VB20479
局部存储器 (L)[1]		LB0 到 LB63	LB0 到 LB63	LB0 到 LB63	LB0 到 LB63	LB0 到 LB63
位存储器 (M)		M0.0 到 M31.7	M0 到 M31.7	M0.0 到 M31.7	M0.0 到 M31.7	M0.0 到 M31.7
特殊存储器 (SM)	总计	SM0.0 到 SM2047.7	SM0.0 到 SM2047.7	SM0.0 到 SM2047.7	SM0.0 到 SM2047.7	SM0.0 到 SM2047.7
	只读	SM0.0 至 SM29.7 SMB480.0 至 SM515.7 SM1000.0 至 SM1699.7	SM0.0 至 SM29.7 SMB480.0 至 SM515.7 SM1000.0 至 SM1699.7	SM0.0 至 SM29.7 SMB480.0 至 SM515.7 SM1000.0 至 SM1699.7	SM0.0 至 SM29.7 SMB480.0 至 SM515.7 SM1000.0 至 SM1699.7	SM0.0 至 SM29.7 SMB480.0 至 SM515.7 SM1000.0 至 SM1699.7
定时器		256（T0 到 T255）	256（T0 到 T255）	256（T0 到 T255）	256（T0 到 T255）	256（T0 到 T255）
保持型接通延时	1 ms	T0、T64	T0、T64	T0、T64	T0、T64	T0、T64
	10 ms	T1 到 T4，以及 T65 到 T68	T1 至 T4 和 T65 至 T68	T1 到 T4，以及 T65 到 T68	T1 到 T4，以及 T65 到 T68	T1 到 T4，以及 T65 到 T68

说明		CPU SR20、CPU ST20	CPU SR30、CPU ST30	CPU SR40、CPU ST40	CPU CR40、CPU CR60	CPU SR60、CPU ST60
	100 ms	T5 到 T31，以及 T69 到 T95	T5 到 T31 和 T69 到 T95	T5 到 T31，以及 T69 到 T95	T5 到 T31，以及 T69 到 T95	T5 到 T31，以及 T69 到 T95
接通/断开延时	1 ms	T32、T96	T32、T96	T32、T96	T32、T96	T32、T96
	10 ms	T33 到 T36，以及 T97 到 T100	T33 到 T36 和 T97 到 T100	T33 到 T36，以及 T97 到 T100	T33 到 T36，以及 T97 到 T100	T33 到 T36，以及 T97 到 T100
	100 ms	T37 到 T63，以及 T101 到 T255	T37 到 T63 和 T101 到 T255	T37 到 T63，以及 T101 到 T255	T37 到 T63，以及 T101 到 T255	T37 到 T63，以及 T101 到 T255
计数器		256（C0 到 C255）	256（C0 到 C255）	256（C0 到 C255）	256（C0 到 C255）	256（C0 到 C255）
高速计数器		HC0 到 HC3	HC0 到 HC3	HC0 到 HC3	HC0 到 HC3	HC0 到 HC3
顺序控制继电器 (S)		S0.0 到 S31.7	S0.0 到 S31.7	S0.0 到 S31.7	S0.0 到 S31.7	S0.0 到 S31.7
累加器寄存器		AC0 到 AC3	AC0 到 AC3	AC0 到 AC3	AC0 到 AC3	AC0 到 AC3
跳转/标号		0 到 255	0 到 255	0 到 255	0 到 255	0 到 255
调用/子例程		0 到 127	0 到 127	0 到 127	0 到 127	0 到 127
中断例程		0 到 127	0 到 127	0 到 127	0 到 127	0 到 127
正/负跳变		1024	1024	1024	1024	1024

说明	CPU SR20、CPU ST20	CPU SR30、CPU ST30	CPU SR40、CPU ST40	CPU CR40、CPU CR60	CPU SR60、CPU ST60
PID 回路	0 到 7	0 到 7	0 到 7	0 到 7	0 到 7
端口	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）	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）	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）	以太网编程端口、集成的 RS485 端口（端口 0）	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）

[1] 在 LAD 或 FBD 中进行编程时，STEP 7-Micro/WIN SMART 会保留 LB60 到 LB63。

订购信息

F

F.1 CPU 模块

CPU 型号	产品编号
CPU SR20 AC/DC/继电器	6ES7288-1SR20-0AA0
CPU ST20 DC/DC/DC	6ES7288-1ST20-0AA0
CPU SR30 AC/DC/继电器	6ES7288-1SR30-0AA0
CPU ST30 DC/DC/DC	6ES7288-1ST30-0AA0
CPU ST40 DC/DC/DC	6ES7288-1ST40-0AA0
CPU SR40 AC/DC/继电器	6ES7288-1SR40-0AA0
CPU CR40 AC/DC/继电器	6ES7288-1CR40-0AA0
CPU SR60 AC/DC/继电器	6ES7288-1SR60-0AA0
CPU ST60 DC/DC/DC	6ES7288-1ST60-0AA0
CPU CR60 AC/DC/继电器	6ES7288-1CR60-0AA0

F.2 扩展模块 (EMs) 和信号板 (SBs)

扩展模块和信号板	产品编号
EM 8 点数字量输入 (EM DT08)	6ES7288-2DE08-0AA0
EM 16 点数字量输入 (EM DE16)	6ES7288-2DE16-0AA0
EM 8 点数字量输出 (EM DT08)	6ES7288-2DT08-0AA0
EM 8 点继电器型数字量输出 (EM DR08)	6ES7288-2DR08-0AA0
EM 16 点继电器型数字量输出 (EM QR16)	6ES7288-2QR16-0AA0
EM 16 点晶体管型数字量输出 (EM QT16)	6ES7288-2QT16-0AA0
EM 8 点数字量输入/8 点数字量输出 (EM DT16)	6ES7288-2DT16-0AA0
EM 8 点数字量输入/8 点继电器输出 (EM DR16)	6ES7288-2DR16-0AA0
EM 16 点数字量输入/16 点数字量输出 (EM DT32)	6ES7288-2DT32-0AA0
EM 16 点数字量输入/16 点继电器输出 (EM DR32)	6ES7288-2DR32-0AA0
EM 4 点模拟量输入 (EM AE04)	6ES7288-3AE04-0AA0
EM 2 点模拟量输出 (EM AQ02)	6ES7288-3AQ02-0AA0
EM 4 点模拟量输出 (EM AQ04)	6ES7288-3AQ04-0AA0
EM 8 点模拟量输入 (EM AE08)	6ES7288-3AE08-0AA0
EM 2 点模拟量输入/1 点模拟量输出 (EM AQ03)	6ES7288-3AM03-0AA0
EM 4 点模拟量输入/2 点模拟量输出 (EM AM06)	6ES7288-3AM06-0AA0
EM 2 点 16 位 RTD (EM AR02)	6ES 288-3AR02-0AA0
EM RTD 4 x 16 位 (EM AR04)	6ES7288-3AR04-0AA0
EM TC 4 x 16 位 (EM AT04)	6ES7288-3AT04-0AA0
EM DP01 PROFIBUS DP SMART (EM DP01)	6ES7288-7PD01-0AA0
SB 2 点数字量输入/2 点数字量输出 (SB DT04)	6ES7288-5DT04-0AA0
SB 1 点模拟量输出 (SB AQ01)	6ES7288-5AQ01-0AA0
SB 1 点模拟量输入 (SB AE01)	6ES7288-5AE01-0AA0
SB RS485/RS232 (SB CM01)	6ES7288-5CM01-0AA0
SB 电池 (SB BA01)	6ES7288-5BA01-0AA0

F.3 编程软件

编程软件	产品编号
STEP 7-Micro/WIN SMART 独立许可证 (CD-ROM)	6ES7288-8SW01-0AA0
Drives V90（PC 工具）软件	可从 Siemens 服务和支持网站下载

F.4 通信

通信卡	产品编号
CP 5411：短 AT ISA	6GK1541-1AA00
CP 5512：PCMCIA 类型 II	6GK1551-2AA00
CP 5611：PCI 卡（版本 3.0 或更高版本）	6GK1561-1AA00

F.5 备件和其它硬件

电缆、网络连接器、中继器和末端保持器	产品编号
I/O 扩展电缆，1 m	6ES7288-6EC01-0AA0
MPI 电缆	6ES7901-0BF00-0AA0
PROFIBUS 网络电缆	6XV1830-0AH10
网络总线连接器（带编程端口连接器），垂直电缆出口	6ES7972-0BB11-0XA0
网络总线连接器（不带编程端口连接器），垂直电缆出口	6ES7972-0BA11-0XA0
RS485 总线连接器（带 35° 电缆出口，不带编程端口连接器）	6ES7972-0BA40-0XA0
RS485 总线连接器（带 35° 电缆出口，带编程端口连接器）	6ES7972-0BB40-0XA0
RS485 IP 20 中继器，隔离	6ES7972-0AA00-0XA0
TD/CPU 连接电缆	6ES7901-3EB10-0XA0
末端保持器，热塑性塑料材质，10 MM	8WA1808
末端保持器，钢制	8WA1805

表格 F- 1　端子排备件套件

如果您拥有 S7-200 SMART 模块（产品编号）	使用此端子排备用套件（每包 4 件）	
	端子排产品编号	端子排描述
CPU SR20，交流/直流/继电器 (6ES7288-1SR20-0AA0)	6ES7292-1AH30-0XA0	8 针，镀锡
	6ES7292-1AH40-0XA0	8 针，镀锡，带键
	6ES7292-1AM30-0XA0	12 针，镀锡
CPU ST20，直流/直流/直流 (6ES7288-1ST20-0AA0)	6ES7292-1AH30-0XA0	8 针，镀锡
	6ES7292-1AM30-0XA0	12 针，镀锡
CPU SR30，交流/直流/继电器 (6ES7288-1SR30-0AA0)	6ES7292-1AH30-0XA0	8 针，镀锡
	6ES7292-1AH40-0XA0	8 针，镀锡，带键
	6ES7292-1AP30-0XA0	14 针，镀锡
	6ES7292-1AK30-0XA0	10 针，镀锡
CPU ST30，直流/直流/直流 (6ES7288-1ST30-0AA0)	6ES7292-1AH30-0XA0	8 针，镀锡
	6ES7292-1AP30-0XA0	14 针，镀锡
	6ES7292-1AK30-0XA0	10 针，镀锡

如果您拥有 S7-200 SMART 模块（产品编号）	使用此端子排备用套件（每包 4 件）	
	端子排产品编号	端子排描述
CPU ST40，直流/直流/直流 (6ES7288-1ST40-0AA0)	6ES7292-1AH30-0XA0	8 针，镀锡
	6ES7292-1AV30-0XA0	20 针，镀锡
	6ES7 292-1AL30-0XA0	11 针，镀锡
CPU SR40，交流/直流/继电器 (6ES7288-1SR40-0AA0)	6ES7292-1AV30-0XA0	20 针，镀锡
	6ES7292-1AV40-0XA0	20 针，镀锡，带键
	6ES7292-1AM30-0XA0	12 针，镀锡
CPU SR40，交流/直流/继电器 (6ES7288-1CR40-0AA0)	6ES7292-1AH30-0XA0	8 针，镀锡
	6ES7292-1AV30-0XA0	20 针，镀锡
	6ES7292-1AL30-0XA0	11 针，镀锡
	6ES7292-1AL40-0XA0	11 针，镀锡，带键
CPU ST60，直流/直流/直流 (6ES7288-1ST60-0AA0)	6ES7292-1AV30-0XA0	20 针，镀锡
	6ES7292-1AM30-0XA0	12 针，镀锡
CPU SR60，交流/直流/继电器 (6ES7288-1SR60-0AA0)	6ES7292-1AV30-0XA0	20 针，镀锡
	6ES7292-1AV40-0XA0	20 针，镀锡，带键
	6ES7292-1AM30-0XA0	12 针，镀锡
CPU CR60，AC/DC/继电器 (6ES7288-1CR60-0AA0)	6ES7292-1AV30-0XA0	20 针，镀锡
	6ES7292-1AV40-0XA0	20 针，镀锡，带键
	6ES7292-1AM30-0XA	12 针，镀锡 0
EM 8 点数字量输入 (EM DE08) (6ES7288-2DE08-0AA0)	6ES7292-1AG30-0XA0	7 针，镀锡
EM 8 点模拟量输出 (EM DT08) (6ES7288-2DT08-0AA0)	6ES7292-1AG30-0XA0	7 针，镀锡
EM 8 点数字量输出继电器 (EM DR08) (6ES7288-2DR08-0AA0)	6ES7292-1AG30-0XA0	7 针，镀锡
	6ES7292-1AG40-0XA0	7 针，镀锡，带键（右侧）
EM 8 点数字量输入/8 点数字量输出 (EM DT16) (6ES7288-2DT16-0AA0)	6ES7292-1AG30-0XA0	7 针，镀锡
EM 8 点数字量输入/8 点继电器输出 (EM DR16)	6ES7292-1AG30-0XA0	7 针，镀锡

如果您拥有 S7-200 SMART 模块（产品编号）	使用此端子排备用套件（每包 4 件）	
	端子排产品编号	端子排描述
(6ES7288-2DR16-0AA0)	6ES7292-1AG40-0XA0	7 针，镀锡，带键（右侧）
EM 16 点数字量输入/16 点数字量输出 (EM DT32) (6ES7288-2DT32-0AA0)	6ES7292-1AL30-0XA0	11 针，镀锡
EM 16 点数字量输入/16 点继电器输出 (EM DR32) (6ES7288-2DR32-0AA0)	6ES7292-1AL30-0XA0	11 针，镀锡
	6ES7292-1AL40-0XA0	11 针，镀锡，带键
EM 4 点模拟量输入 (EM AE04) (6ES7288-3AE04-0AA0)	6ES7292-1BG30-0XA0	7 针，镀金
EM 8 点模拟量输入 (EM AE08) (6ES7288-3AE08-0AA0)	6ES7292-1BG30-0XA0	7 针，镀金
EM 2 点模拟量输出 (EM AQ02) (6ES7288-3AQ02-0AA0)	6ES7292-1BG30-0XA0	7 针，镀金
EM 4 点模拟量输出 (EM AQ04) (6ES7288-3AQ04-0AA0)	6ES7292-1BG30-0XA0	7 针，镀金
EM 2 点模拟量输入/1 点模拟量输出 (EM AM03) (6ES7288-3AM03-0AA0)	6ES7292-1BG30-0XA0	7 针，镀金
EM 4 点模拟量输入/2 点模拟量输出 (EM AM06) (6ES7288-3AM06-0AA0)	6ES7292-1BG30-0XA0	7 针，镀金
EM RTD 2 x 16 位 (EM AR02) (6ES7288-3AR02-0AA0)	6ES7292-1BG30-0XA0	7 针，镀金
EM RTD 4 x 16 位 (EM AR04) (6ES7288-3AR04-0AA0)	6ES7292-1BG30-0XA0	7 针，镀金
EM TC 4 x 16 位 (EM AT04) (6ES7288-3AT04-0AA0)	6ES7292-1BG30-0XA0	7 针，镀金
EM PROFIBUS DP SMART (EM DP01) (6ES7288-7DP01-0AA0)	6ES7292-1BG30-0XA0	7 针，镀金

F.6 人机界面设备

人机界面设备	产品编号
SMART LINE HMI	
SMART LINE 700 IE	6AV6648 -0BC11-3AX0
SMART LINE 1000 IE	6AV6648 -0BE11-3AX0
Micro HMI	
TD 400C 文本显示，4 行[1]	6AV6640-0AA00-0AX1
TD400C 空白面板材料，A4 大小（10 张/包）	6AV6671-0AP00-0AX0

[1] ：包括定制的空白面板覆层如需要更多空白面板覆层，请订购适合 TD 设备的空白面板材料。

索引

符号

*D（STL - 双整数乘法）, 308
*I（STL - 整数乘法）, 308
*R（STL - 实数乘法）, 308
.mwp 文件, 106
.smart 文件, 106
/D（STL - 双整数除法）, 308
/I（STL - 整数除法）, 308
/R（STL - 实数除法）, 308
+D（STL - 双整数加法）, 308
+I（STL - 整数加法）, 308
+R（STL - 实数加法）, 308
<=B, 236
<=R, 236
<=W, 236
<>B, 236
<>R, 236
<>S, 240
<>W, 236
<B, 236
<R, 236
<W, 236
=（STL - 输出）, 180
==B, 236
==R, 236
==S, 240
==W, 236
=I（STL - 立即输出）, 180
>=B, 236
>=R, 236
>=W, 236
>B, 236
>R, 236
>W, 236

A

A（STL - 与）, 171
AB<（STL - 与运算比较字节小于）, 236
AB<=（STL - 与运算比较字节小于或等于）, 236
AB<>（STL - 与运算比较字节不等于）, 236
AB=（STL - 与运算比较字节等于）, 236
AB>（STL - 与运算比较字节大于）, 236
AB>=（STL - 与运算比较字节大于或等于）, 236
AC
    绝缘准则, 60
    接线准则, 61
AD<（STL - 与运算比较双字小于）, 236
AD<=（STL - 与运算比较双字小于或等于）, 236
AD<>（STL - 与运算比较双字不等于）, 236
AD=（STL - 与运算比较双字等于）, 236
AD>（STL - 与运算比较双字大于）, 236
AD>=（STL - 与运算比较双字大于或等于）, 236
ADD_DI, 308
ADD_I, 308
ADD_R, 308
AENO（STL 堆栈 - 与 ENO 位）
    指令, 176
    逻辑堆栈概述, 175
AI（STL - 立即与）, 173
ALD（STL 堆栈 - 与装载）, 176
AN（STL - 与非）, 171
ANDB（STL-字节与）, 349
ANDD（STL-双字与）, 349

ANDW（STL-字与）, 349
ANI（STL - 取反后立即与）, 173
AR<（STL - 与运算比较实数小于）, 236
AR<=（STL - 与运算比较实数小于或等于）, 236
AR<>（STL - 与运算比较实数不等于）, 236
AR=（STL - 与运算比较实数等于）, 236
AR>（STL - 与运算比较实数大于）, 236
AR>=（STL - 与运算比较实数大于或等于）, 236
AS<>（STL - 与运算字符串比较不等于）, 240
AS=（STL - 与运算字符串比较等于）, 240
ASCII 数组转换指令, 246
ATCH, 335
ATH, 246
ATT, 383
AW<（STL - 与运算比较字小于）, 236
AW<=（STL - 与运算比较字小于或等于）, 236
AW<>（STL - 与运算比较字不等于）, 236
AW=（STL - 与运算比较字等于）, 236
AW>（STL - 与运算比较字大于）, 236
AW>=（STL - 与运算比较字大于或等于）, 236

B

B_I, 242
BA01 电池信号板, 165
BCD_I, 242
BCDI（STL - BCD 码转换为整数）, 242
BGN_ITIME, 403
BIR（STL-字节立即读取）, 355
BITIM（STL-开始间隔时间）, 403
BIW（STL-字节立即写入）, 355
BLKMOV_B, 352
BLKMOV_D, 352
BLKMOV_W, 352
BMB（STL-字节块传送）, 352
BMD（STL-双字块传送）, 352
BMW（STL-字块传送）, 352
BTI（STL - 字节转换为整数）, 242

C

CAL_ITIME, 403
CALL, 405
CE 认证, 719
CFND（STL-查找字符）, 380
CHR_FIND, 380
CITIM（STL-计算间隔时间）, 403
CLR_EVNT, 335
CM01 信号板
  连接器引脚分配, 477
  偏置且端接, 479
COS（余弦）, 314
CPU
  CR40 规范, 744
  CR40 接线图, 752
  CR60, 757
  CR60 接线图, 765
  DIN 导轨, 52
  IP 地址, 423
  LED, 20
  LED 指示灯, 99
  MAC 地址, 431
  PPI 连接数, 467
  SR20 规范, 724
  SR20 接线图, 732
  SR30 规范, 733
  SR40 规范, 744
  SR40 接线图, 752
  SR60 规范, 757
  SR60 接线图, 765
  ST20 接线图, 732
  ST30 规范, 733

ST30 接线图, 741
ST40 规范, 744
ST40 接线图, 752
ST60 规范, 757
ST60 接线图, 765
支持的扩展模块, 23
尺寸, 20, 49
以太网通信, 419
以太网端口, 423
扩展电缆, 57
过程映像寄存器, 70
在面板上的安装, 51
存储卡, 96
安装, 50
设置类型, 40
访问数据, 72
连接电源, 30
系统块, 135
非致命错误存储单元, 847
组态与 HMI 的通信, 432
绝缘准则, 60
致命错误, 848
特性, 20
通信连接数, 414
通信类型, 25, 413
接线准则, 61
清除存储区, 166
CPU
SR30 接线图, 741
CPU ID 寄存器 (SMB6-SMB7), 859
CPU ST20
规范, 724
CPU 和模块的设备组态, 135
CPU 硬件/固件 ID (SMB1000-SMB1049), 882
CRET（STL-从子例程有条件返回）, 405
CTD（向下计数）, 264
CTU（向上计数）, 264
CTUD（向上计数/向下计数）, 264

D

-D（STL - 双整数减法）, 308
DC
绝缘准则, 60
接线准则, 61
DEC_B, 317
DEC_DW, 317
DEC_W, 317
DECB（STL - 字节递减）, 317
DECD（STL - 双字递减）, 317
DECO, 260
DECW（STL - 字递减）, 317
DI_I, 242
DI_R, 242
DI_S, 252
DIN 导轨, 50, 52
DISI, 335
DIV, 312
DIV_DI, 308
DIV_I, 308
DIV_R, 308
DP 设备
EM DP01 PROFIBUS DP, 440
DTA, 246
DTCH, 335
DTI（STL - 双整数转换为整数）, 242
DTR（STL - 双整数转换为实数）, 242
DTS（STL-双整数转换为字符串）, 252

E

ED（STL - 下降沿）, 179

EM DE16 规范, 771
EM DP01 PROFIBUS DP 模块
DP 协议, 440
GSD 设备数据库文件, 458
PROFIBUS 网络中, 441
状态 LED, 835
其它组态功能, 457
组态选项, 445
接线图, 836
数据交换模式, 452
EM（扩展模块）硬件/固件 ID (SMB1100-SMB1299), 884
ENCO, 260
END, 368
ENI, 335
EU（STL - 上升沿）, 179
EXP（自然指数）, 314

F

FBD 编辑器, 113
FIFO, 385
FILL（STL-表格填充）, 387
FILL_N, 387
FND=、<>、<、>（STL-表查找）, 388
FOR, 356

G

GET, 196
GET_ADDR, 219
GET_ERROR, 370
GET_ERROR（STL-获取非致命错误代码）, 370
GIP_ADDR, 220
GSD 文件
EM DP01 PROFIBUS DP 模块, 458

H

HDEF（高速计数器定义）, 267
HMI
支持的设备, 24, 416
多主站和多从站 PPI 网络, 472
设备, 479
单主站 PPI 网络, 471
组态以太网通信, 432
常见网络指南, 479
HSC（高速计数器）, 169, 267
HSC0、HSC1 和 HSC2 高速计数器寄存器（SMB36-65，SMB136-145）, 865
HTA, 246

I

-I（STL - 整数减法）, 308
I/O
阶跃响应时间 (SM), 800
模拟量输入的电压表示法, 801
模拟量输入的电流表示法, 802
模拟量输出的电压表示法, 803
模拟量输出的电流表示法, 803
I/O 扩展总线 - 通信错误 (SMW98), 876
I/O 寻址, 83
I/O 错误状态
PLC 非致命错误 SM 标志, 847
PLC 非致命错误代码, 844
SMB5, 859
I/O 模块 ID 和错误寄存器 (SMB8-SMB19), 860
I_B, 242
I_BCD, 242
I_DI, 242
I_S, 252
IBCD（STL-整数转换为 BCD 码）, 242
INC_B, 317

INC_DW, 317
INC_W, 317
INCB（STL - 字节递增）, 317
INCD（STL - 双字递增）, 317
INCW（STL - 字递增）, 317
INV_B, 348
INV_DW, 348
INV_W, 348
INVB（STL-字节取反）, 348
INVD（STL-双字取反), 348
INVW（STL-字取反）, 348
IP 地址, 422
  MAC 地址, 431
  分配, 420, 428
  组态, 423
IP 路由器, 423
ITA, 246
ITB（STL - 整数转换为字节）, 242
ITD（STL - 整数转换为双整数）, 242
ITS（STL-整数转换为字符串）, 252

J

JMP, 358

L

L 存储器, 121
LAD 编辑器, 112
LBL, 358
LD（STL - 装载）, 171
LD（STL 堆栈 - 取反后立即装载）, 175
LD（STL 堆栈 - 装载）, 175
LDB<（STL - 装载比较字节小于）, 236
LDB<=（STL - 装载比较字节小于或等于）, 236
LDB<>（STL - 装载比较字节不等于）, 236
LDB=（STL - 装载比较字节等于）, 236
LDB>（STL - 装载比较字节大于）, 236
LDB>=（STL - 装载比较字节大于或等于）, 236
LDD<（STL - 装载比较双字小于）, 236
LDD<=（STL - 装载比较双字小于或等于）, 236
LDD<>（STL - 装载比较双字不等于）, 236
LDD=（STL - 装载比较双字等于）, 236
LDD>（STL - 装载比较双字大于）, 236
LDD>=（STL - 装载比较双字大于或等于）, 236
LDI（STL - 立即装载）, 173
LDI（STL 堆栈 - 立即装载）, 175
LDN（STL - 取反后装载）, 171
LDN（STL 堆栈 - 取反后装载）, 175
LDNI（STL - 取反后立即装载）, 173
LDR<（STL - 装载比较实数小于）, 236
LDR<=（STL - 装载比较实数小于或等于）, 236
LDR<>（STL - 装载比较实数不等于）, 236
LDR=（STL - 装载比较实数等于）, 236
LDR>（STL - 装载比较实数大于）, 236
LDR>=（STL - 装载比较实数大于或等于）, 236
LDS（STL 堆栈 - 装载）, 176
LDS<>（STL - 装载字符串比较不等于）, 240
LDS=（STL - 装载字符串比较等于）, 240
LDW<（STL - 装载比较字小于）, 236
LDW<=（STL - 装载比较字小于或等于）, 236
LDW<>（STL - 装载比较字不等于）, 236
LDW=（STL - 装载比较字等于）, 236
LDW>（STL - 装载比较字大于）, 236
LDW>=（STL - 装载比较字大于或等于）, 236
LED 指示灯
  CPU 状态, 99
LIFO, 385
LN（自然对数）, 314
LPP（STL 堆栈 - 逻辑出栈）, 176
LPS（STL 堆栈 - 逻辑进栈）, 176
LRD（STL 堆栈 - 逻辑读栈）, 176

LSCR（STL-装载 SCR）, 359

M

MAC 地址, 431
MBUS_CTRL（初始化 Modbus 主站通信）, 497
MBUS_INIT（初始化从站通信）, 507
MBUS_MSG/MB_MSG2（从 Modbus 主站发送消息）, 499
MBUS_SLAVE（从站对主站消息的响应）, 510
Modbus RTU 从站
    MBUS_INIT（初始化从站通信）, 507
    MBUS_SLAVE（从站对主站消息的响应）, 510
    执行错误代码, 511
    使用指令, 505
Modbus RTU 主站
    MBUS_CTRL（初始化主站通信）, 497
    MBUS_MSG/MB_MSG2（从主站发送消息）, 499
    示例程序, 512
    执行错误代码, 503
    使用指令, 496
Modbus 库概述, 487
Modbus 常规
    Modbus 协议的初始化和执行时间, 495
    寻址, 489
    库功能, 492
    使用 Modbus 指令的要求, 493
    高级用户信息, 515
MOV_B, 351
MOV_BIR, 355
MOV_BIW, 355
MOV_DW, 351
MOV_R, 351
MOV_W, 351
MOVB（STL-字节传送）, 351
MOVD（STL-双字传送）, 351
MOVR（STL-实数传送）, 351
MOVW（STL-字传送）, 351
MUL, 312
MUL_DI, 308
MUL_I, 308
MUL_R, 308

N

NEXT, 356
NOP, 184
NOT (STL), 178

O

O（STL - 或）, 171
OB<（STL - 或运算比较字节小于）, 236
OB<=（STL - 或运算比较字节小于或等于）, 236
OB<>（STL - 或运算比较字节不等于）, 236
OB=（STL - 或运算比较字节等于）, 236
OB>（STL - 或运算比较字节大于）, 236
OB>=（STL - 或运算比较字节大于或等于）, 236
OB1, 103
OD<（STL - 或运算比较双字小于）, 236
OD<=（STL - 或运算比较双字小于或等于）, 236
OD<>（STL - 或运算比较双字不等于）, 236
OD=（STL - 或运算比较双字等于）, 236
OD>（STL - 或运算比较双字大于）, 236
OD>=（STL - 或运算比较双字大于或等于）, 236
OI（STL - 立即或）, 173
OLD（STL 堆栈 - 或装载）, 176
ON（STL - 或非）, 171
ONI（STL - 取反后立即或）, 173
OR<（STL - 或运算比较实数小于）, 236
OR<=（STL - 或运算比较实数小于或等于）, 236
OR<>（STL - 或运算比较实数不等于）, 236

OR=（STL - 或运算比较实数等于）, 236
OR>（STL - 或运算比较实数大于）, 236
OR>=（STL - 或运算比较实数大于或等于）, 236
ORB（STL-字节或）, 349
ORD（STL-双字或）, 349
ORW（STL-字或）, 349
OS<>（STL - 或运算字符串比较不等于）, 240
OS=（STL - 或运算字符串比较等于）, 240
OUC 库示例
  CheckErrors 子例程程序, 557
  主动伙伴（客户端）程序, 547
  被动伙伴（服务器）程序, 559
OUC 库示例主动伙伴（客户端）程序
  符号表, 558
OUC 库示例被动伙伴（服务器）程序
  符号表, 568
OW<（STL - 或运算比较字小于）, 236
OW<=（STL - 或运算比较字小于或等于）, 236
OW<>（STL - 或运算比较字不等于）, 236
OW=（STL - 或运算比较字等于）, 236
OW>（STL - 或运算比较字大于）, 236
OW>=（STL - 或运算比较字大于或等于）, 236

P

PC/PPI 电缆, 475
PID 回路指令
  了解, 326
  回路控制类型, 330
  报警检查, 333
PID 回路控制
  PID 整定控制面板, 619
  回路定义表, 610
PID 自整定
  PV 超限, 618
  先决条件, 614
  自偏差, 614
  自滞后, 614
  序列, 615
  例外情况, 617
PID 整定控制面板, 619
PLC
  扩展电缆, 57
  在面板上的安装, 51
  存储卡, 96
  安装, 50
  系统块, 135
  非致命错误存储单元, 847
  信息（硬件/固件、错误状态、运行/停止事件日志）, 127
  致命错误, 848
  清除存储区, 166
  编译及运行时间错误, 844
PLC 菜单
  下载, 88
  上传, 91
PLS
  用于监视和控制 PTO 和 PWM 输出的专用存储器, 870
  指令, 294
PPI 通信
  切换到自由端口模式, 205
  多主站和多从站 PPI 网络, 472
  系统块中的端口组态, 137
  单主站网络, 471
PROFIBUS
  DP 设备, 438
  S7-200 SMART EM DP01 PROFIBUS DP 模块, 439

PTO0、PWM0、PTO1、PWM1、PTO2 和 PWM2 高速输出（SMB66-SMB85、SMB166-SMB169、SMB176-SMB179 和 SMB566-SMB579）, 870
PUT, 196
PWM 向导
  PWMx_RUN 子例程, 628
PWMx_RUN, 628

R

-R（STL - 实数减法）, 308
R（STL - 复位）, 181
R_S, 252
RCV（接收消息控制 SMB86-SMB94 和 SMB186-SMB194）, 874
READ_RTC, 188
READ_RTCX, 191
RET, 405
RETI, 335
RI（STL - 立即复位）, 181
ROUND, 242
RS（LAD/FBD 复位优先双稳态触发器）, 182
RS232, 485
  自由端口模式, 483
  通信连接数, 414
  通信类型, 25, 413
RS485
  网络组态示例, 471
  设置波特率和端口网络地址, 469
  通信连接数, 414
  通信类型, 25, 413
  通信概述, 466
  通信端口组态, 137
RTA, 246
RTD 模拟量输入
  RTD 类型, 154
  平滑化, 154
  电阻, 154
  抑制, 154
  系统块组态, 154
  系数, 154
  标定, 154
RTS（STL-实数转换为字符串）, 252
RUN 模式, 43, 98

S

S（STL - 置位）, 181
S_DI, 257
S_I, 257
S_R, 257
S7-200 SMART
  作为 PROFIBUS 从站设备, 441
S7-200 SMART 硬件故障排除, 607
SB（信号板）硬件/固件 ID (SMB1050-SMB1099), 883
SCR, 359
SCRE, 359
SCRT, 359
SEG, 242
SET_ADDR, 219
SET_RTC, 188
SET_RTCX, 191
SFND（STL-查找字符串）, 380
SHL_B, 371
SHL_DW, 371
SHL_W, 371
SHR_B, 371
SHR_DW, 371
SHR_W, 371
SHRB, 375

SI（STL - 立即置位）, 181
Siemens 提供的库, 487
SIN（正弦）, 314
SIP_ADDR, 220
SLB（STL-左移字节）, 371
SLD（STL-左移双字）, 371
SLW（STL-左移字）, 371
SM 存储器，PTO/PWM 操作, 299
SM 位置（运动轴）, 709
SM（特殊存储器）分配及功能, 851
SMB0 系统状态位, 854
SMB1 指令执行状态位, 855
SMB1000-SMB1049 CPU 硬件/固件 ID, 882
SMB1050-SMB1099 SB（信号板）硬件/固件 ID, 883
SMB1100-SMB1299 EM（扩展模块）硬件/固件 ID, 884
SMB130 端口 1 组态, 863
SMB136-145 (HSC3) 高速计数器 3, 865
SMB186-SMB194 接收消息控制, 874
SMB2 自由端口接收字符, 856
SMB28-SMB29 信号板 ID 和错误寄存器, 862
SMB3 自由端口字符错误, 857
SMB30（端口 0）和 SMB130（端口 1）组态, 863
SMB36-45 (HSC0) 高速计数器 0, 865
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, 858
SMB46-55 (HSC1) 高速计数器 1, 865
SMB480-SMB515 数据日志状态, 878
SMB5 I/O 错误状态位, 859
SMB56-65 (HSC2) 高速计数器 2, 865
SMB566-SMB579：PTO2 和 PWM2 高速输出, 870
SMB600-SMB649 轴 0 运动控制, 879
SMB66-SMB85、SMB166-SMB169 和 SMB176-SMB179：PTO0、PWM0、PTO1 和 PWM1 高速输出, 870
SMB66-SMB85、SMB166-SMB169、SMB176-SMB179 和 SMB566-SMB579：PTO0、PWM0、PTO1、PWM1、PTO2 和 PWM2 高速输出, 870
SMB6-SMB7 CPU ID 寄存器, 859
SMB86-SMB94 和 SMB186-SMB194 接收消息控制, 874
SMB8-SMB19 I/O 模块 ID 和错误寄存器, 860
SMW100-SMW114 系统报警, 877
SMW22-SMW26 扫描时间, 861
SMW98 I/O 扩展总线 - 通信错误, 876
SQRT（平方根）, 314
SR（LAD/FBD 置位优先双稳态触发器）, 182
SRB（STL-右移字节）, 371
SRD（STL-右移双字）, 371
SRW（STL-右移字）, 371
SSCPY（STL-复制子字符串）, 379
SSTR_CPY, 379
STD（STL-子字符串转换为双整数）, 257
STEP 7-Micro/WIN SMART
  RUN 和 STOP 模式, 43, 98
  与 CPU 连接, 33, 424
  以太网端口组态, 423
  设备要求, 26
STEP 7-Micro/WIN（早期版）, 106
STL
  状态选项, 599
  逻辑堆栈操作, 176
STL 编辑器, 114
STOP 模式, 43, 98
  写入和强制输出, 604
  数字量输出状态, 141
  模拟量输出状态, 152
STOP（指令）, 368
STR（STL-子字符串转换为实数）, 257
STR（STL-子字符串转换为整数）, 257

STR_FIND, 380
SUB_DI, 308
SUB_I, 308
SUB_R, 308
SWAP, 354

T

TAN（正切）, 314
TBL_FIND, 388
TC 模拟量输入
  TC 类型, 159
  平滑化, 159
  抑制, 159
  系统块组态, 159
  标定, 159
TC 模拟量输入模块, 159
TODR（STL - 读取日时钟）, 188
TODRX（STL - 读取扩展日时钟）, 191
TODW（STL - 写入日时钟）, 188
TODWX（STL - 写入扩展日时钟）, 191
TOF（断开延时定时器）, 392
TON（接通延时定时器）, 392
TONR（保持型接通延时定时器）, 392
TRUNC, 242

U

USS 协议库
  计算通信时间, 571
  执行错误, 588
  使用 USS 协议指令, 572
  要求, 570
  概述, 487, 569
USS 协议指令
  USS_CTRL, 576
  USS_INIT, 573
  USS_RPM_x, 581
  USS_WPM_x, 584
  示例程序, 589
  使用, 572

V

V90 驱动器, 634, 668, 907

W

WAND_B, 349
WAND_DW, 349
WAND_W, 349
WDR（看门狗定时器复位）, 368
WOR_B, 349
WOR_DW, 349
WOR_W, 349
WXOR_B, 349
WXOR_DW, 349
WXOR_W, 349

X

XORB（STL-字节异或）, 349
XORD（STL-双字异或）, 349
XORW（STL-字异或）, 349

G

工作模式
  切换为 RUN, 43, 98
  切换为 STOP, 43, 98
  启动选项, 148
工具（选项）
  STL 状态, 599
  执行状态颜色指示, 596

工具菜单
PID 整定控制面板, 619
运动控制面板, 686

X

下载
示例程序, 42
程序, 88

S H

上电后恢复数据, 98
上传程序, 91

Z

子例程
PWMx_RUN, 628
用户程序元素, 103
运动轴, 651
准则, 652
子例程指令
CALL、RET, 405
调用参数和返回示例, 408

K

开环控制, 625
开放式用户通信 (OUC)
连接指令, 435
连接类型, 435
开放式用户通信 (OUC) 库
DISCONNECT 指令, 543
ISO_CONNECT 指令, 524
TCP_CONNECT 指令, 521
TCP_RECV 指令, 532
TCP_SEND 指令, 529
UDP_CONNECT 指令, 527
UDP_RECV 指令, 539
UDP_SEND 指令, 536
共用库指令参数, 518
指令错误代码, 545
概述, 487

Y

元素使用, 594

Z H

专用存储器字节
EM DP01 PROFIBUS DP, 453
支持, 3

B

比例项，PID 算法, 328
比较指令
比较字符串, 240
比较数值, 236

R

日时钟
扩展时钟指令, 191
时钟指令, 188
读写保护, 144

Z H

中继器, 474
中断
CPU 型号的事件支持, 337
中断队列溢出 (SMB4), 858
中断事件类型, 341

示例程序, 342
优先级和排队, 342
全局中断启用状态 (SMB4), 858
连接/分离、启用/禁用、有条件返回和清除事件指令, 335
定时中断的时间间隔值 (SMB34-SMB35), 864
编程准则, 339
概述, 337
中断例程, 70
用户程序的元素, 104

Q

气流, 47
气流和冷却间隙, 47

C

从 RUN 切换到 STOP
数字量输出状态, 141
模拟量输出状态, 152
从自由端口收到的字符错误 (SMB3), 857

F

分配
全局符号, 117
局部变量, 124
变量（局部）, 121

W

文件菜单
下载, 88
上传, 91

J

计数器指令
标准计数器, 264
高速计数器, 267
高速计数器（初始化示例）, 286
高速计数器（编程示例）, 273

D

订购信息, 905

C H

尺寸
CPU, 20
安装, 49

D

队列中断溢出 (SMB4), 858

Y

以太网
GET, 196
IP 地址, 422
ISO-on-TCP 协议, 434
MAC 地址, 431
TCP 协议, 434
TSAP, 437
UDP 协议, 434
组态 CPU 与 HMI 设备之间的通信, 432
通信连接数, 414
通信类型, 25, 413
程序段, 418
端口, 435

以太网网络
组态 CPU 的 IP 地址, 423
搜索 CPU, 429

S H

示例
DISCONNECT 指令, 544
GET 和 PUT 指令, 201
ISO_CONNECT 指令, 526
Modbus RTU 从站协议，编程, 510
TCP_CONNECT 指令, 523
TCP_RECV 指令, 535
UDP_CONNECT 指令, 529
UDP_RECV 指令, 542
UDP_SEND 指令, 539
与 CPU 进行 PROFIBUS DP 通信, 463
开放式用户通信 (OUC) 库, 547, 557, 559
用于对模拟量输入值进行采样的子例程, 105
向上/向下计数计数器指令, 266
安装 PROFIBUS DP EM DP01 GSD 文件, 446
运动轴 AXISx_CTRL、AXISx_RUN、AXISx_SEEK 和 AXISx_MAN 子程序应用, 679
运动轴简单相对移动（定长截断应用）, 677
位逻辑输入, 184
位逻辑输出, 186
表格, 390
表格查找 (TBL_FIND) 指令, 390
转换指令, 244
使用 AXISx_ABSPOS 子程序从 SINAMICS V90 伺服驱动读取绝对位置。, 670
组态 PROFIBUS DP EM DP01 I/O, 448
移位寄存器位 (SHRB) 指令, 376
示例控制程序, 35

D

打开早期版 STEP 7-Micro/WIN 项目, 106

G

功率要求
CPU, 47, 839
计算, 842
示例, 841

B

本地 I/O 和扩展 I/O 寻址, 83
本地/伙伴连接, 418

K

卡（存储）, 96

D

电缆
扩展设备, 837
电源, 47, 839
电磁兼容性 (EMC), 720

Y

用于定时中断的 SMB34-SMB35 时间间隔值, 864
用户定义的库, 592

B

包络表值
PTO 生成器, 304

Z H

主动/被动通信伙伴, 418

主条目, 771
主程序, 103

L

立即 I/O 读取/写入, 70
立即指令
  LAD、FBD 和 STL, 173

X

写入值
  在 STOP 模式下写入和强制输出, 604
  输出, 69

C H

出厂默认存储卡, 168

B

边缘检测器, 179

F

发送指令
  示例, 217
  发送数据, 206
发热区, 47

D

动态 IP 信息, 423

Z H

执行
  单次或多次扫描, 605
  程序, 70

K

扩展 I/O 和本地 I/O 寻址, 83
扩展电缆, 837
  安装, 57
  卸下, 57
扩展总线 - 通信错误 (SMW98), 876
扩展模块, 23, 859
  EM DE08, 771
  EM QR16, 773
  尺寸, 49
  模块 ID 和错误寄存器 (SMB8-SMB19), 860
  模块错误状态 (SMB5), 859
扩展模块 (EM)
  EM AE04 规范, 788
  EM AE04 接线图, 791
  EM AM06 规范, 796
  EM AM06 接线图, 799
  EM AQ02 规范, 792
  EM AQ02，接线图, 794
  EM AQ04，接线图, 794
  EM AR02 (RTD) 规范, 811
  EM AR02 (RTD) 接线图, 817
  EM AT04 规范, 805
  EM DE08 规范, 771
  EM DE08 接线图, 771
  EM DR08 规范, 773
  EM DR08 接线图, 776
  EM DR16 规范, 780
  EM DR16 接线图, 784
  EM DR32 规范, 780
  EM DR32 接线图, 786
  EM DT08 规范, 773
  EM DT08 接线图, 776
  EM DT16 规范, 780
  EM DT16 接线图, 784

EM DT32 规范, 780
EM DT32 接线图, 786
安装和拆除, 56
接线图, 805
模拟量输入的电压表示法, 801
模拟量输入的电流表示法, 802
模拟量输出的电压表示法, 803
模拟量输出的电流表示法, 803
扩展模块 (SB)
SB AE01 规范, 821

S

扫描周期
执行多次扫描, 605
执行单次扫描, 605
扫描时间 (SMW22-SMW26), 861

G

过压, 722

X

协议
PROFIBUS DP, 440
西门子技术支持, 3

C

存储卡
使用, 93
复位为出厂默认设置, 168
类型, 92
程序传送卡, 96
存储器, 847
非致命错误指示器的地址, 847
保持范围组态, 142
清除 PLC, 166

T

同步更新（PWM 指令）, 299

H

回路表, 334
回路控制 (PID)
正作用/反作用, 332
回路定义表, 610
转换输入, 330
转换输出, 331
调整偏置, 332
错误条件, 334
模式, 333

W

网络（通信）
RS485 网络组态示例, 471
计算网络距离, 473
地址, 468
网络组态, 466
安全注意事项, 473
单主站 PPI, 471
选择网络电缆, 475
通用构建指南, 473
通信类型, 25, 413
偏置电缆, 478
偏置和端接网络电缆, 477
网络连接器引脚分配, 476
网络组态示例，RS485 设备, 471
网络通信, 30

X

先前版 STEP 7-Micro/WIN 项目, 106

D

丢失密码, 168

C H

传送指令
SWAP（交换字中的字节数据）, 354
传送（字节、字、双字、实数）, 351
字节立即传送（读取和写入）, 355
块传送（字节、字和双字）, 352
传输率, 473

Z

自由端口模式
SMB86-SMB94 和 SMB186-SMB194
接收消息控制, 874
中断, 341
示例, 482
自由端口发送器空闲 (SMB4), 858
自由端口字符错误 (SMB3), 857
自由端口组态（SMB30 - 端口 0 和 SMB130 - 端口 1）, 863
自由端口接收字符 (SMB2), 856
字符中断控制, 216
启用, 204
自整定 PID, 609

X

向导
文本显示, 24
高速计数器 (HSC), 273

Q

全局符号, 117

D

多次扫描, 605

J

交叉引用, 594
交流感性负载, 63

C H

产品编号, 905, 906, 907, 911

D

灯负载, 62

W

污染等级, 722

Z

字一致性
PROFIBUS, 443
字节一致性
PROFIBUS, 443
字符串
表示方式, 82
格式, 81
字符串指令
查找字符串/字符, 380
复制子字符串, 379

A

安全, 144
安全电路, 101
安装
DIN 导轨, 52, 52
气流和冷却间隙, 47
尺寸, 49
扩展电缆, 57, 57
扩展模块 (EM), 56
灯负载, 62
面板, 51, 51
信号板 (SB), 53
将设备与热源、高压和电气噪声隔离开, 45
绝缘准则, 60
准则, 45
剧烈振动环境, 52
接地, 60
接地和电路, 59
接线准则, 61
隔离, 60
概述, 45, 50, 50
感性负载, 63
端子块连接器, 54

S H

设置 CPU 类型, 40

F

访问权限
CPU 安全, 144
密码权限级别, 144

X

寻址
计数器存储器, 76
本地 I/O 和扩展 I/O, 83
过程映像输出寄存器：, 74
存储区, 73
创建指针和使用间接地址, 84
局部存储器, 79
使用指针访问表中数据的示例, 85
使用指针偏移量访问数据的示例, 87
变量存储器, 74
定时器存储器, 75
标志存储器, 74
顺序控制继电器 (SCR) 存储器, 80
特殊存储器 (SM) 位, 78
高速计数器, 76
累加器, 77
符号表, 117
模拟量输入, 80
模拟量输出, 80

Y

运动曲线
创建步, 633
定义, 631
组态, 631
操作模式, 632
运动向导
SM 位置, 709
组态/曲线表, 698
最大速度和启动/停止速度, 629
运动指令，错误代码, 696
运动轴
ACCEL_TIME, 630
AXISx_ABSPOS, 668
AXISx_CACHE, 666

AXISx_CFG, 665
AXISx_CTRL, 653
AXISx_DIS, 664
AXISx_GOTO, 656
AXISx_LDOFF, 660
AXISx_LDPOS, 662
AXISx_MAN, 654
AXISx_RDPOS, 667
AXISx_RSEEK, 659
AXISx_RUN, 658
AXISx_SRATE, 663
RP 搜索模式, 712
SM 位置, 709
子例程, 651
子例程准则, 652
运动控制面板, 686
极性, 643
定义运动曲线, 649
组态, 637
组态反冲补偿, 646
组态参考点和搜索参数, 647
组态输入引脚位置, 638
相位, 642
显示运动轴的曲线组态, 693
显示和修改轴组态, 692
显示和控制轴操作, 687
映射 I/O, 639
消除反冲, 717
编程, 636
输入加速时间, 645
输入点动参数, 645
输入急停时间, 646
输入最大启动和停止速度, 645
错误代码, 694
运动控制
运动特性, 634
极性, 643
定义运动曲线, 649
组态反冲补偿, 646
组态参考点和搜索参数, 647
组态输入引脚位置, 638
相位, 642
映射 I/O, 639
输入加速时间, 645
输入点动参数, 645
输入急停时间, 646
输入最大启动和停止速度, 645
运动控制中的步进电机, 629
运动控制向导
开环运动控制, 625
运动控制面板, 686
显示运动轴的曲线组态, 693
显示和修改轴组态, 692
显示和控制轴操作, 687
运动输入和输出
CPU, 634
运行时间和 PLC 编译错误, 844

J

技术支持, 3
技术规范, 719

Y

抑制电路, 63

B

报警
系统 (SMW100-SMW114), 877
模拟量输入组态, 151

L

连接
- 连接数 (RS232), 414
- 连接数 (RS485), 414
- 连接数（以太网）, 414
- 通信类型, 25, 413

连接器, 54

S H

时间戳不匹配（PC/PLC 程序不同）, 843
时钟指令
- READ_RTC, 188
- READ_RTCX, 191
- SET_RTC, 188
- SET_RTCX, 191

W

位逻辑指令
- AENO（STL - 与 ENO）, 175
- NOP（空操作）, 184
- NOT, 178
- STL 逻辑堆栈操作, 176
- 边缘检测器, 179
- 输入示例, 184
- 输入的 STL 逻辑堆栈指令, 175
- 输出示例, 186
- 输出线圈, 180
- 置位和复位优先双稳态触发器, 182
- 置位和复位位, 181
- 触点, 171
- 触点（立即）, 173

X

系统块, 104
- BA01 电池信号板, 165
- CPU 的 IP 地址, 426
- CPU 组态, 135
- RS485/RS232 CM01 通信信号板, 163
- RTD 模拟量输入模块, 154
- TC 模拟量输入模块, 159
- 启动选项, 148
- 密码和安全, 144
- 数字量输入滤波器, 140

系统报警 (SMW100-SMW114), 877
系统状态位 (SMB0), 854

D

冻结输出
- 数字量输出组态, 141
- 模拟量输出组态, 152

Z H

状态
- LED 指示灯, 99
- 执行有限次数的扫描, 605
- 在程序编辑器中显示, 596
- 构建状态图表, 600

状态 LED
- EM DP01 PROFIBUS DP, 455

状态错误（时间戳不匹配）, 843

K

库
- 创建, 592
- 类型, 487

L

冷却, 47
冷端补偿
热电偶, 163, 808

W

忘记密码, 144, 168

J

间接寻址
创建指针和使用间接地址, 84
使用指针访问表中数据的示例, 85
使用指针偏移量访问数据的示例, 87
符号表, 119

K

快速访问工具栏, 26

Q

启动
STEP 7-Micro/WIN SMART, 32
启动选项, 148

Z H

诊断
LED 指示灯, 99

J

局部变量, 121

Q

驱动器, 634, 668, 907
驱动器通信
计算时间需求, 571

H

环境
工业环境, 720
运行条件, 721
运输和存储条件, 721

B

表格指令
ATT, 383
FIFO/LIFO, 385
FILL_N, 387
TBL_FIND, 388

G

规范, (SB BA01)
CE 认证, 719
CPU CR40, 744
CPU CR60, 757
CPU SR20, 724
CPU SR30, 733
CPU SR40, 744
CPU SR60, 757
CPU ST20, 724
CPU ST30, 733
CPU ST40, 744
CPU ST60, 757
EM AE04, 788
EM AM06, 796
EM AQ02, 792
EM AR02 (RTD), 811
EM AT04, 805

EM DP01 PROFIBUS DP, 832
EM DR08, 773
EM DR16, 780
EM DR32, 780
EM DT08, 773
EM DT16, 780
EM DT32, 780
EM QR16, 773
EM QT16, 773
SB AE01, 821
SB AQ01, 825, 826
SB CM01, 827
SB DT04, 818, 820
工业环境, 720
电磁兼容性 (EMC), 720
过压, 722
污染等级, 722
阶跃响应时间 (SM), 800
环境条件, 721
保护, 722
绝缘, 722
继电器电气使用寿命, 723
常规技术规范, 719
模拟量输入的电压表示法, 801
模拟量输入的电流表示法, 802
模拟量输出的电压表示法, 803
模拟量输出的电流表示法, 803
额定电压, 723

Z H

直流感性负载, 63

G

构建状态图表, 600
构建通信网络, 473

S H

事件，中断, 337

Z H

转换指令
ASCII 子字符串转换为数值, 257
ASCII 数组转换, 246
标准转换, 242
编码和解码, 260
数值转换为字符串, 252

R

软件调试, 593

F

非易失性存储器, 92
非法语法
符号表, 117
非致命 PLC 错误
特殊存储器位置, 847
编译及运行时间, 844
非致命错误对运行时执行的影响, 128

T

图表
打开, 600
创建, 600
构建, 600

F

服务与支持, 3

B

变量表, 121
波特率
开关选择：RS232/PPI 多主站电缆, 483
设置, 468
通信, 137

D

定义
局部变量, 124
定时中断组态 (SMB34-SMB35), 864
定时器指令
BITIM、CITIM, 403
TON、TONR、TOF, 392
中断, 341
编程提示和示例, 396

S H

实数值, 81

X

线圈
立即复位位, 181
立即置位位, 181
复位位, 181
输出, 180
输出（立即）, 180
置位位, 181

Z

组态
CPU，系统块, 135
EM DP01 PROFIBUS DP, 443
IP 地址, 423
MAC 地址, 431
以太网, 423
动态 IP 信息, 424
曲线表（运动轴）, 698
静态 IP 信息, 426
组态图, 101

X

项目
打开先前版 STEP 7-Micro/WIN 项目, 106

Z H

指令
GET, 196
GET_ADDR, 219
GIP_ADDR, 220
PUT, 196
SET_ADDR, 219
SIP_ADDR, 220
回路控制 (PID), 319
快速参考指南, 893
指令执行状态位 (SMB1), 855
指令库, 592
指针
创建指针和使用间接地址, 84
使用指针访问表中数据的示例, 85
使用指针偏移量访问数据的示例, 87

A

按字访问, 73

G

故障排除
LED 指示灯, 99

B

标志存储器, 74

Z H

轴 0 运动控制 (SMB600-SMB649), 879

X

显示状态, 596
选择网络电缆, 475
选项
  STL 状态, 599

F

复位为出厂默认存储卡, 168

B

保存项目, 41
保护等级, 722
保持范围，系统块组态, 142
保持性存储器, 92

X

信号板 (SB)
  SB AQ01 规范, 825
  SB AQ01 接线图, 826
  SB BA01 规范, 830
  SB BA01 接线图, 830
  SB CM01 规范, 827
  SB CM01 接线图, 829
  SB DT04 规范, 818
  SB DT04 接线图, 820
  安装和拆除, 53
  输入的电压表示法, 801
  输入的电流表示法, 802
  模拟量输出的电压表示法, 803
  模拟量输出的电流表示法, 803
信号板 ID 和错误寄存器 (SMB28-SMB29), 862
信号模块 (SM)
  阶跃响应时间, 800

M

脉冲串输出 (PTO)
  周期时间, 296
  指令, 169
  脉冲输出指令 (PLS), 294
脉冲捕捉位，系统块中的数字量输入组态, 140
脉宽调制 (PWM)
  周期时间, 298
  指令, 169
  脉冲输出, 628
  脉冲输出指令 (PLS), 294
  输出, 626, 626

S H

首次扫描，执行, 605
首次扫描标志 (SMB0), 854

H

恢复为出厂默认设置的存储卡, 92

K

客户支持, 3

J

绝缘, 722
绝缘准则, 60

Z H

振动, 52

R

热电偶
  EM AT04 热电偶选型表, 809
  EM AT04 热电偶滤波器选型表, 809
  冷端补偿, 163, 808
  基本操作, 163, 808
热线, 3
热源、高压和电噪声, 45

Z H

致命错误 (PLC), 848
致命错误对运行时执行的影响, 130

T

特性
  CPU, 20
  支持的扩展模块, 23
特殊存储器分配和功能, 851

J

积分项，PID 算法, 328

G

高速计数器, 273
高速计数器寄存器, 865

Z H

准则
  在面板上的安装, 51
  安装步骤, 50
  接地和电路, 59
  接线准则, 61
  隔离, 60

J

兼容性
  EM DP01 PROFIBUS DP 模块, 834

F

浮点值, 330

D

调试和监视
  程序编辑器状态, 596
  强制值, 603

J

剧烈振动环境, 52

T

通信
  IP 地址, 423
  RS485 组态, 466
  STEP 7-Micro/WIN SMART 设置, 32
  以太网、RS485 和 RS232, 25, 413
  对话框, 424
  在 CPU 上查找 MAC 地址, 431
  网络, 30
  连接数 (RS232), 414

连接数 (RS485), 414
连接数（以太网）, 414
限制写入, 144
点对点接口 (PPI) 协议, 467
选择, 466
硬件连接, 31
模块产品编号, 907
通信协议
PROFIBUS, 438
通信驱动程序, 417
通信指令
传送, 203
接收, 203
通信端口, 415
自由端口模式, 480
连接器引脚分配, 476
系统块组态, 137

J

继电器电气使用寿命, 723
接地, 60
接收指令
用户终止, 215
字符间定时器, 214
奇偶校验错误, 215
空闲线检测, 210
结束字符检测, 214
结束条件, 210
起始字符检测, 212
消息定时器, 215
断开检测, 213
最大字符计数, 215
接线图
CPU CR40, 752
CPU CR60, 765
CPU SR20, 732
CPU SR30, 741
CPU SR40, 752
CPU SR60, 765
CPU ST20, 732
CPU ST30, 741
CPU ST40, 752
CPU ST60, 765
EM AE04, 791
EM AM06, 799
EM AQ02, 794
EM AQ04, 794
EM AR02 (RTD), 817
EM AT04, 805
EM DE08, 771
EM DE16, 771
EM DP01 PROFIBUS DP 模块, 836
EM DR08, 776
EM DR16, 784
EM DR32, 786
EM DT08, 776
EM DT16, 784
EM DT32, 786
SB AQ01, 826
SB CM01, 829
接线图
SB BA01, 830
接线准则, 61
DIN 导轨, 52
气流和冷却间隙, 47
灯负载, 62
安装, 45
将设备与热源、高压和电气噪声隔离开, 45
接地, 60
接地和电路, 59
隔离, 60

感性负载, 63
端子块连接器, 54

C H

常开触点
立即, 173
标准, 171
常闭触点
立即, 173
标准, 171
常规技术规范, 719

L

逻辑、控制, 67
逻辑运算指令
与、或、异或（字节、字和双字）, 349
取反, 348
逻辑堆栈
STL 堆栈操作, 176
STL 输入, 175

Y

移位和循环移位指令
字节、字、双字, 371
位 (SHRB), 375

F

符号（符号寻址）
间接寻址, 119
定义全局符号, 117
符号表, 117

P

偏置 PID 回路, 320
偏置且端接
CM01 信号板, 479
网络电缆, 477

D

断电 (PLC), 142

Q

清除 PLC 存储区, 166
清除存储卡, 168

M

密码
权限级别, 144
丢失或忘记, 168
密码保护, 144

L

联系信息, 3

Y

硬件故障排除, 607

C H

程序
子例程, 103
元素, 103
中断例程, 104
执行有限扫描, 605
存储卡, 96
状态图表, 600
程序中的书签, 593
程序块, 103

程序状态
执行有限次数的扫描, 605
在程序编辑器中显示, 596
构建状态图表, 600
程序指令
子例程, 405, 408
比较, 236, 240
中断, 335
计数器, 264, 267, 273, 286
传送, 351, 352, 354, 355
字符串, 379, 380
时钟, 188, 191
位逻辑, 171, 173, 175, 176, 178, 179, 180, 181, 182, 184, 184, 186
表, 383, 385, 387
表格查找, 388
转换, 242, 246, 252, 257, 260
定时器, 392, 403
逻辑运算, 348, 349
移位与循环移位, 371, 375
程序控制, 356, 358, 359, 368, 370
数学, 308, 312, 314, 317
程序控制指令
END、STOP 和 WDR, 368
FOR-NEXT 循环, 356
GET_ERROR, 370
JMP-LBL, 358
SCR（顺控继电器）, 359
程序编辑器
STL 状态选项, 599
书签, 593
使用, 36
类型, 111
调试和监视, 596

Z H

装配
接线准则, 61
隔离, 60

Q

强制
在 STOP 模式下写入和强制输出, 604
强制值指示器 (SMB4), 858

G

隔离, 60

H

缓冲区一致性
PROFIBUS, 443

B

编译程序
PLC 非致命程序代码, 844
下载, 88

G

感性负载, 63

S H

输入
NOT 触点, 178
STL 逻辑堆栈, 175
边缘检测器, 179
位逻辑示例, 184
实际情况和在程序中, 67
标准触点, 171, 173

脉冲捕捉位（系统块）, 140
读取, 69
输入过程映像寄存器, 69
输入滤波时间, 139
输出
写入, 69
位逻辑示例, 186
实际情况和在程序中, 67
线圈, 180
置位和复位优先双稳态触发器, 182
置位和复位位, 181
输出映像寄存器, 68

Z H

置位和立即复位指令, 181
置位和复位优先双稳态触发器指令, 182

C

错误
GET_TABLE 和 PUT_TABLE 指令, 197
I/O 错误状态, 859
I/O 模块 ID 和错误寄存器 (SMB8-SMB19), 860
Modbus RTU 从站执行, 511
Modbus RTU 主站执行, 503
PID 自整定, 617
PLC 错误处理, 127
PWMx_RUN 子例程, 628
存储单元（PLC 非致命错误）, 847
自由端口通信期间收到的字符错误 (SMB3), 857
运动指令, 696
运动轴, 694
时间戳不匹配（PC/PLC 程序不同）, 843
非致命错误对运行时执行的影响, 128
信号板 ID 和错误寄存器 (SMB28-SMB29), 862
致命 (PLC), 848
致命错误对运行时执行的影响, 130
编译和运行时间错误（PLC 程序）, 844
数据保持, 142

W

微分项，PID 算法, 329

C H

触点
NOT, 178
下降沿检测器, 179
上升沿检测器, 179
常开, 171
常开（立即）, 173
常闭, 171
常闭（立即）, 173

S H

数字量输入, 139
数字量输入滤波器, 140
数字量输出, 141
数字输入滤波时间, 139
数学运算指令
加法、减法、乘法和除法, 308
产生双整数的整数乘法和带余数的整数除法, 312
带余数的整数除法, 312
递增和递减, 317
数学函数, 314
数值转换为字符串的指令, 252
数据
保持, 142
接收, 206
数据一致性
PROFIBUS, 443

数据日志
状态 (SMB480-SMB515), 878
数据块 (DB), 104

L

滤波时间, 139
滤波器，数字量输入组态, 140

J

静态 IP 信息, 423

M

模块
CPU CR40, 744
CPU CR60, 757
CPU SR20, 724
CPU SR30, 733
CPU SR40, 744
CPU SR60, 757
CPU ST20, 724
CPU ST40, 744
CPU ST60, 757
EM AE04, 788
EM AM06, 796
EM AQ02, 792
EM AR02 (RTD), 811
EM DE08, 771
EM DE16 规范, 771
EM DP01 PROFIBUS DP, 832
EM DR08, 773
EM DR16, 780
EM DR32, 780
EM DT08, 773
EM DT16, 780
EM DT32, 780
EM QR16, 773
EM QT16, 773
SB AE01, 821
SB AQ01, 825, 826
SB CM01, 827
SB DT04, 818, 820
ST30, 733
尺寸, 49
模拟量 I/O
阶跃响应时间 (SM), 800
输入的电压表示法, 801
输入的电流表示法, 802
输出的电压表示法, 803
输出的电流表示法, 803
模拟量输入
平滑化, 149
抑制, 149
系统块组态, 149
模拟量类型组态, 149
模拟量输出
RUN/STOP 转换时的状态, 152
模拟量类型组态, 152

G

管道化
PTO 脉冲, 297

D

端子块连接器, 54

E

额定电压, 723

C

操作站, 101

M

默认网关 IP 地址, 422