SIEMENS

SIMATIC
S7-200 可编程序控制器

系统手册

E20001-H5540-C400-V2-5D00

版本 04/2002

安全指南

本手册包括了保证人身安全与保护本产品及连接的设备应遵守的注意事项。这些注意事项在手册中以警告三角形加以突出，并按照危险等级标明如下：

危险
表示如果不采取适当的预防措施，将导致死亡或者严重的人身伤害。

警告
表示如果不采取适当的预防措施，将有导致死亡或者严重人身伤害的可能。

小心
表示如果不采取适当的预防措施，将有导致较轻微的人身伤害的可能。

小心
表示如果不采取适当的预防措施，将有导致财产损失的可能。

注意
表示如果不采取适当的预防措施，有可能导致不希望的结果或状态。

合格人员

只有**合格人员**才允许安装和操作该设备。合格人员是指被授权按照既定安全惯例和标准，对线路、设备和系统进行调试、接地和加标识的人员。

正确应用

注意如下：

警告

该设备及其部件只能用于产品目录或者技术说明中所描述的范畴，并且只能与 Siemens 公司认可或者推荐的第三方厂家出产的设备或部件一起使用。

只有正确地运输、保管、配置和安装，并且按照推荐的方式操作和维护，产品才能正常、安全地运行。

注册商标

SIMATIC®、SIMATIC HMI® 和 SIMATIC NET® 是 SIEMENS AG 的注册商标。
手册中还包括其它一些注册商标，如果它们因个人目的而被第三方厂家所使用，商标所有者的权力将受到侵害。

Siemens AG
Automation and Drives (A&D)
Industrial Automation Systems (AS)
Postfach 4848, D-90327 Nürnberg

Siemens AG
订货号：E20001-H5540-C400-V2-5D00

前言

S7-200 系列小型 PLC（Micro PLC）可应用于各种自动化系统。紧凑的结构、低廉的成本以及功能强大的指令集使得 S7-200 PLC 成为各种小型控制任务理想的解决方案。S7-200 产品的多样化以及基于 Windows 的编程工具，使您能够更加灵活地完成自动化任务。

读者

该手册为有一定 PLC 背景知识的工程师、编程人员、安装人员及电气人员编写。

适用范围

本手册包括以下产品信息：

- S7-200 CPU：CPU221，CPU222，CPU224，CPU226 和 CPU226XM
- S7-200 EM22X 扩展模块
- STEP 7-Micro/WIN，3.2 版，32 位 S7-200 编程软件包
- STEP 7-Micro/WIN Toolbox，1.0 版，32 位 S7-200 工具软件包

它为需要使用 TP070、Modbus 或者 Micro Master 系列变频器等产品的用户提供软件编程工具。

认证标准

SIMATIC S7-200 系列产品符合以下标准：

- European Community (CE) Low Directive 73/23/EEC
  EN 61131-2：可编程序控制器——设备要求
- European Community (CE) EMC Directive 89/336/EEC
  电磁辐射标准
  EN50081-1：民用、商务及轻工业
  EN50081-2：工业环境
  防电磁辐射标准：
  EN61000-6-2：工业环境
- Underwriters Laboratories，Inc。
  UL 508 Listed (工业控制设备)　注册号 E75310
- Canadian Standards Association：CSA C22.2 Number 142 Certified（过程控制设备）
- Factory Mutual Research: FM Class I，Division 2，Group A，B，C&D Hazardous Locations，T4A and Class I，Zone 2，IIC，T4。
- 相关信息参见附录 A。

船用许可

至本手册印发之日为止，S7-200 系列产品已得到了以下船用代理机构的认可。如果想得到最新的产品许可信息，请与当地的 Siemens 公司分销商或者销售办公室联系。

代理机构	认证号
Lloyds Register of Shipping (LRS)	99 / 20018 (E1)
American Bureau of Shipping (ABS)	01-HG20020-PDA
Germanischer Lloyd (GL)	12 045 – 98 HH
Det Norske Veritas (DNV)	A-8071
Bureau Veritas (BV)	09051/A2 BV
Nippon Kaiji Kyokai (NK)	A-534

如何使用本手册

如果您是初次使用 S7-200 系列产品，您需要通读本手册。如果您是一位有经验的用户，可以通过目录和索引查找相应信息。

S7-200 可编程序控制器系统手册按照以下主题组织编排：

- 第一章（产品概述）对 S7-200 系列 PLC 产品的特点作一简单的描述。
- 第二章（使用入门）教您如何创建并下载一个简单的控制程序。
- 第三章（S7-200 的安装）提供 S7-200 CPU 模块和扩展 I/O 模块的安装尺寸和基本安装指南。
- 第四章（PLC 的基本概念）提供 S7-200 的操作信息。
- 第五章（编程的概念、惯例及特点）描述 STEP 7-Micro/WIN 软件的特点、程序编辑器、指令集的种类（IEC 1131-3 或者 SIMATIC）、S7-200 的数据类型和创建程序的步骤。
- 第六章（S7-200 指令集）编程指令的描述及示例。
- 第七章（网络通讯）介绍 S7-200 支持的各种网络配置。
- 第八章（硬件故障处理方法及软件调试工具）介绍 S7-200 硬件故障的处理方法以及 STEP 7-Micro/WIN 软件中为您提供的调试工具。
- 第九章（为定位模块创建程序）介绍如何使用向导为 EM235 定位模块创建程序。
- 第十章（为 Modem 模块创建程序）介绍如何使用向导为 EM 241 Modem 模块创建程序。
- 第十一章（CP 243-1 工业以太网通讯处理器）介绍了如何使用 CP 243-1。
- 第十二章（用 USS 协议控制 MicroMaster 系列变频器）不仅介绍了如何使用 USS 指令，而且介绍了如何配置 MicroMaster 系列第三代和第四代变频器。
- 第十三章（使用 Modbus 协议）介绍 Modbus 通讯协议指令。
- 附录 A（S7-200 技术规范）提供 S7-200 的硬件技术指标。

其它附录提供了更多的相关参考信息，例如对于错误代码、特殊存储器（SM）标志位、S7-200 订货号、STL 指令执行时间以及 S7-200 的应用示例等信息的描述。

其它帮助信息

S7-200 和 STEP 7-Micro/WIN 的相关信息

除去本手册之外，STEP 7-Micro/WIN 软件为编程使用者提供在线帮助。如果您购买 STEP 7-Micro/WIN 软件，将免费得到一张资料光盘。光盘的内容包括应用示例、电子版的系统手册和其它信息。

在线帮助

帮助信息距您只“一键之遥”！只要按下 F1 键您就可以在 STEP 7-Micro/WIN 软件中得到在线帮助信息。

在线帮助信息不仅能使您在对 S7-200 的编程过程中得到帮助，而且包括其它一些主题。

电子手册

在资料光盘中有电子版的 S7-200 系统手册。您可以将它安装到您的计算机硬盘上，以便在需要的时候随时使用。

应用示例

资料光盘中包括了一系列应用示例的程序。这些例子程序在您编制自己的应用程序时是值得借鉴的。这些例子程序在 Internet 上也可以找到。

Internet：www.siemens.com/S7-200

如果想得到关于 Siemens 公司产品与服务、技术支持、常见问题、产品更新以及应用示例的更多信息，可以访问以下网址：

- www.ad.siemens.de
  该网址包括 Siemens 自动化与驱动部的 SIMATIC 系列产品和 Siemens 其它产品的相关信息。
- www.siemens.com/s7-200
  该网址包括了 S7-200 产品的常见问题、应用示例、最新产品信息、产品更新或下载。
- www.ad.siemens.com.cn/s7-200
  该网站为中文网站，以中文形式介绍 S7-200 的有关问题。

技术帮助和购买 S7-200 产品

当地 Siemens 公司销售部或分销商

有关技术咨询，产品培训及产品订货的相关事宜，请与当地 Siemens 公司销售部或分销商联系。由于我们的销售代表具备一定的工业过程知识背景，对于您所使用的 Siemens 公司产品进行过技术培训，他们能够快捷高效地回答您有可能遇到的问题。

技术服务

在 S7-200 技术支持中心，经过高级技术培训的人员同样可以为您解决您有可能遇到的问题。您可以在任何时候与他们联系。

- 美国国内
  当地时间：星期一到星期五的 8:00~19:00（东部时间）
  电话： +1 800 241-4453
  传真： +1（0）770 740-3699
  E-mail： drives.support@sea.siemens.com
- 美国以外的美洲地区
  当地时间：星期一到星期五的 8:00~19:00（东部时间）
  电话： +1（0）770 740-3505
  传真： +1（0）770 740-3699
  E-mail： drives.suppert@sea.siemens.com
- 欧洲和非洲地区
  当地时间（纽伦堡）：星期一到星期五的 7:00~17:00
  电话： +49（0）180 5050-222
  传真： +49（0）180 5050-223
  E-mail： techsupport@ad.siemens.de
- 亚洲及澳大利亚
  当地时间（新加坡）：星期一到星期五的 8:30~17:30
  电话： +65（0）740-7000
  传真： +65（0）740-7001
  E-mail： drives.support@sea.siemens.com.sg
- 中国
  当地时间：星期一到星期五的 8:30~17:15
  电话： +86（10）6471-9990
  传真： +86（10）6471-9991
  E-mail： adcs@pek1.siemens.com.cn

目录

1 产品概述

S7-200 系列是一种可编程序逻辑控制器（Micro PLC）。它能够控制各种设备以满足自动化控制需求。S7-200 的用户程序中包括了位逻辑、计数器、定时器、复杂数学运算以及与其它智能模块通讯等指令内容，从而使它能够监视输入状态，改变输出状态以达到控制目的。紧凑的结构、灵活的配置和强大的指令集使 S7-200 成为各种控制应用的理想解决方案。

本章内容：

S7-200 CPU

S7-200 CPU 将一个微处理器、一个集成电源和数字量 I/O 点集成在一个紧凑的封装中，从而形成了一个功能强大的微型 PLC，参见图 1-1。当您下载程序之后，S7-200 就可以按照逻辑关系监控 I/O 设备从而实现您的应用要求。

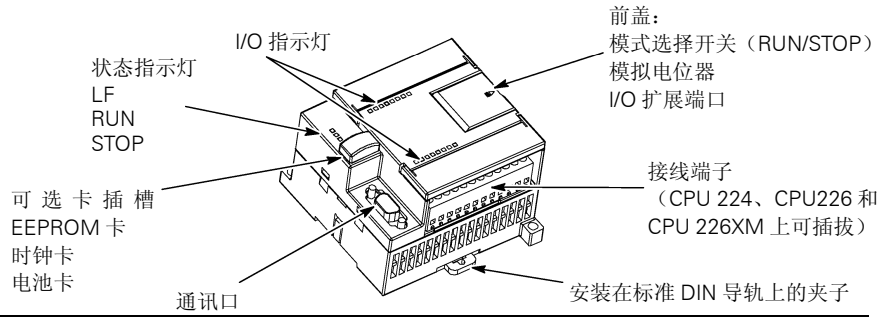


图 1-1 S7-200 PLC

Siemens 公司提供多种类型的 CPU 以适应各种应用，表 1-1 中对各种 CPU 的特性作一简单比较，详细信息参见附录 A。

表 1-1 S7-200 的技术指标

特性	CPU 221	CPU 222	CPU 224	CPU 226	CPU 226XM
外形尺寸（mm）	90×80×62	90×80×62	120.5×80×62	190×80×62	190×80×62
程序存储区	2048 字	2048 字	4096 字	4096 字	8192 字
数据存储区	1024 字	1024 字	2560 字	2560 字	5120 字
掉电保持时间	50 小时	50 小时	190 小时	190 小时	190 小时
本机 I/O	6 入/4 出	8 入/6 出	14 入/10 出	24 入/16 出	24 入/16 出
扩展模块数量	0	2	7	7	7
高速计数器 单相 双相	 4 路 30KHz 2 路 20KHz	 4 路 30KHz 2 路 20KHz	 6 路 30KHz 4 路 20KHz	 6 路 30KHz 4 路 20KHz	 6 路 30KHz 4 路 20KHz
脉冲输出（DC）	2 路 20KHz	2 路 20KHz	2 路 20KHz	2 路 20KHz	2 路 20KHz
模拟电位器	1	1	2	2	2
实时时钟	配时钟卡	配时钟卡	内置	内置	内置
通讯口	1 RS-485	1 RS-485	1 RS-485	2 RS-485	2 RS-485
浮点数运算	有				
I/O 映象区	256 （128 入/128 出）				
布尔指令执行速度	0.37 μs/指令				

S7-200 扩展模块

为了更好地满足应用要求，S7-200 系列为您提供多种类型的扩展模块。你可以利用这些扩展模块完善 CPU 的功能。表 1-2 列出了现有的扩展模块，详细信息参见附录 A。

表 1-2 S7-200 的扩展模块

扩展模块	型号		
数字量模块 输入	8×DC 输入	8×AC 输入	
输出	8×DC 输出	8×AC 输出	8×继电器输出
混合	4×DC 输入/4×DC 输出	8×DC 输入/8×DC 输出	16×DC 输入/16×DC 输出
	4×DC 输入/4×继电器输出	8×DC 输入/8×继电器输出	16×DC 输入/16×继电器输出
模拟量模块 输入	4 输入	4 热电偶输入	2 热电阻输入
输出	2 输出		
混合	4 输入/1 输出		
智能模块	定位模块	Modem 模块	PROFIBUS-DP 模块
其它模块	ASI		

STEP 7-Micro/WIN 编程软件

STEP 7-Micro/WIN 编程软件为用户开发、编辑和监控自己的应用程序提供了良好的编程环境。为了能快捷高效地开发您的应用程序，STEP 7-Micro/WIN 软件为您提供了三种程序编辑器。STEP 7-Micro/WIN 软件中提供了在线帮助系统，以便您获取所需要的信息。另外，软件附带的资料光盘上，包括了电子手册、应用示例以及其它有用的信息。

计算机配置要求

STEP 7-Micro/WIN 既可以在 PC 机上运行，也可以在 Siemens 公司的编程器上运行。PC 机或编程器的最小配置如下：

- 操作系统
- Windows 95，Windows 98，Windows 2000，Windows Me（Millennium Edition）或者 Windows NT 4.0 以上
- 至少 50M 硬盘空间
- 鼠标（推荐）
- 如果希望在 Windows Me 或者 Windows2000 系统中支持 PC/PPI 电缆多主站模式，您的计算机至少是 400 MHz PII 处理器，参见第 7 章。

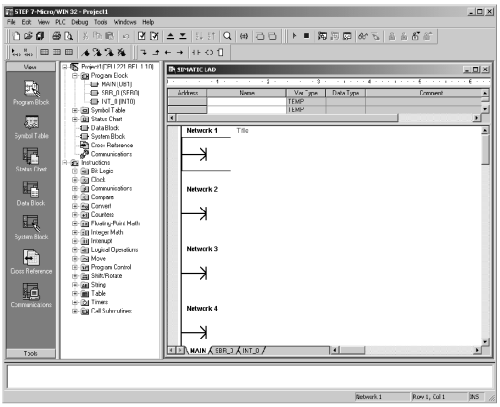


图 1-2 STEP 7–Micro/WIN

安装 STEP 7-Micro/WIN

将 STEP 7-Micro/WIN 的安装光盘插入你的 CD-ROM，安装向导程序将自动启动并引导你完成整个安装过程。如果想得到更多的安装信息，你可以查阅 Readme 文件。

提示：

如果在 Windows NT 或者 Windows 2000 操作系统中安装 STEP 7-Micro/WIN，你必须具备管理员权限。

通讯方式选择

你可以有两种方式连接 S7-200 和你的编程设备：一种是直接使用 PC/PPI 电缆；另一种是用通讯卡和 MPI 电缆。

PC/PPI 电缆比较常用而且成本较低。它将 S7-200 的编程口与计算机的 RS-232 相连。PC/PPI 电缆也可用于其它通讯设备与 S7-200 的连接。

如果使用 MPI 电缆，你必须先在计算机上安装通讯卡。使用这种方式时，你可以用较高的波特率进行通讯。

显示面板

TD200 文本显示器

TD200 是可以与 S7-200 相连的 20 字符的双行文本显示器。

使用 TD200 向导程序进行编程，能很方便地显示所需要的文本信息和数据。

TD200 为您监控数据，修改参数提供了一个价格低廉的人机界面。

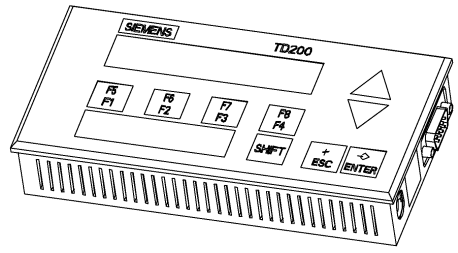


图 1-3 TD200 文本显示器

有一本单独的手册详细描述了 TD200 的功能和技术规范。

TP070 触摸屏

TP070 是一种能与 S7-200 相连的触摸屏。它能够用于制作用户操作界面。

TP070 可以显示图形、棒图、变量和按钮等用户定义的元素。

对 TP070 编程，您应使用 TP Designer 1.0 软件。

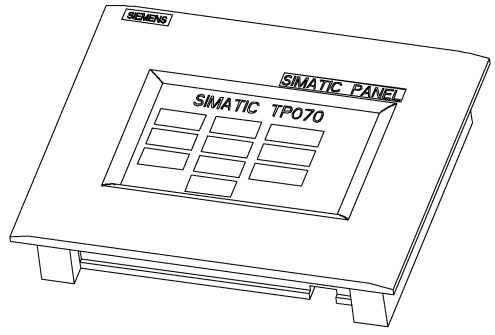


图 1-4 TP070 触摸屏

使用入门

2

STEP 7-Micro/WIN 软件使您能够很容易地对 S7-200 进行编程。通过一个简单例子程序的几个简短步骤，您将学会如何在 S7-200 中连接、编程和运行程序。

为完成这个例子程序，你需要 PC/PPI 电缆、S7-200 CPU 和能够运行 STEP 7-Micro/WIN 软件的编程设备。

本章内容：

连接 S7-200 CPU

连接 S7-200 十分容易。在本例中，你只需要给 S7-200 CPU 供电，然后在编程设备与 S7-200 CPU 之间连上通讯电缆即可。

给 S7-200 CPU 供电

第一个步骤就是要给 S7-200 的 CPU 供电。图 2-1 给出了直流供电和交流供电两种 CPU 模块的接线方式。

在安装和拆除任何电气设备之前，必须确认该设备的电源已断开。在安装和拆除 S7-200 之前，必须遵循适当的安全防护规范，并确认 S7-200 的电源已断开。

警告

在带电情况下对 S7-200 及相关设备进行安装或接线有可能造成电击或者操作设备误动作。在安装或拆卸过程中，如果没有切断 S7-200 及相关设备的供电，有可能导致死亡或者严重的人身伤害和设备损坏。

必须遵循适当的安全防护规范，并确认 S7-200 的电源已断开。

图 2-1 给 S7-200 CPU 供电

连接 PC/PPI 电缆

图 2-2 给出了用 PC/PPI 电缆连接 S7-200 和编程设备的方法。步骤如下：

1. 将 PC/PPI 电缆上 RS-232 的一端连在编程设备的串行口上（本例中为 COM1）。
2. 将 PC/PPI 电缆上，RS-485 的一端连在 S7-200 的编程口上。
3. 按照图 2-2 中所示设置 DIP 开关。

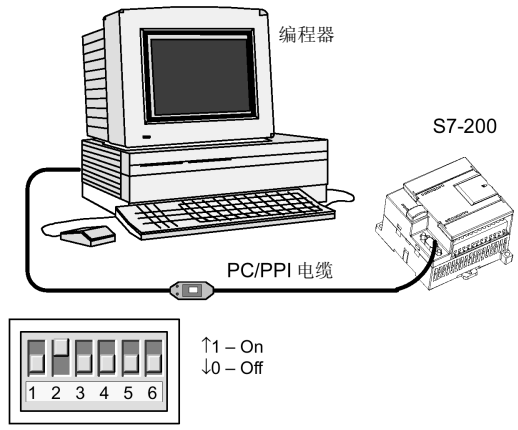


图 2-2 连接 PC/PPI 电缆

打开 STEP 7-Micro/WIN

点击 STEP 7-Micro/WIN 的图标，打开一个新的项目，如图 2-3 所示。

注意左侧的操作条。您可以用操作条中的图标，打开 STEP 7-Micro/WIN 项目中的组件。

点击操作条中的通讯图标进入通讯对话框。你可以用这个对话框为 STEP 7-Micro/WIN 设置通讯参数。

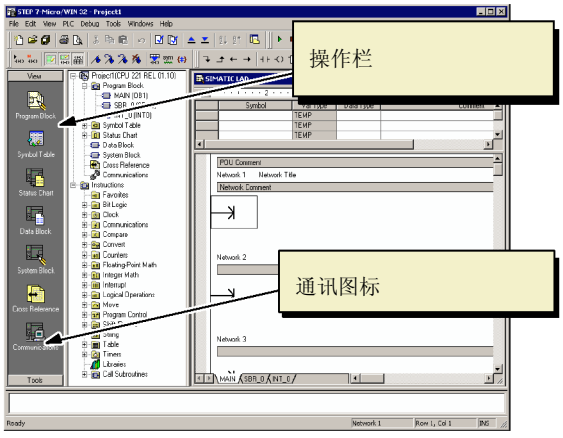


图 2-3 新的 STEP 7-Micro/WIN 项目

为 STEP 7-Micro/WIN 设置通讯参数

在本例中使用 PC/PPI 电缆的缺省设置。参数如下：

1. PC/PPI 电缆的通讯地址设为 0。
2. 接口使用 COM1。
3. 传输波特率用 9.6Kbps。

如果你需要改变通讯设置，请参考第 7 章。

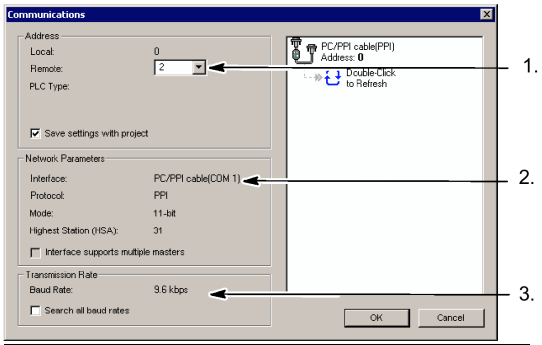


图 2-4 设置通讯参数

与 S7-200 建立通讯

用通讯对话框与 S7-200 建立通讯：

1. 在通讯对话框中双击刷新图标。STEP 7-Micro/WIN 搜寻并显示所连接的 S7-200 站的 CPU 图标。
2. 选择 S7-200 站并点击 OK。

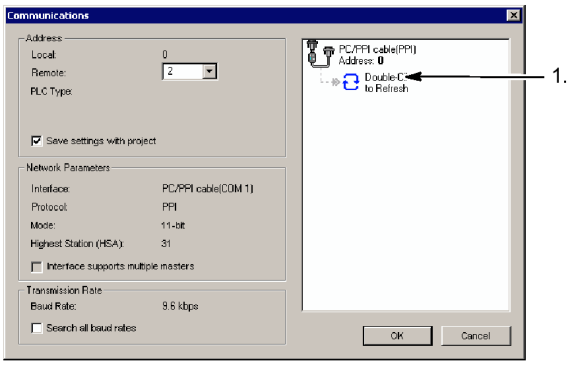


图 2-5 与 S7-200 建立通讯

如果 STEP 7-Micro/WIN 未能找到你的 S7-200 CPU，请核对你的通讯参数设置。

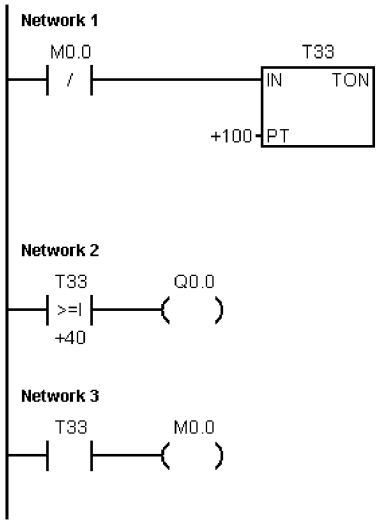
在与 S7-200 建立通讯之后，你可以准备创建和下载程序。

创建一个例子程序

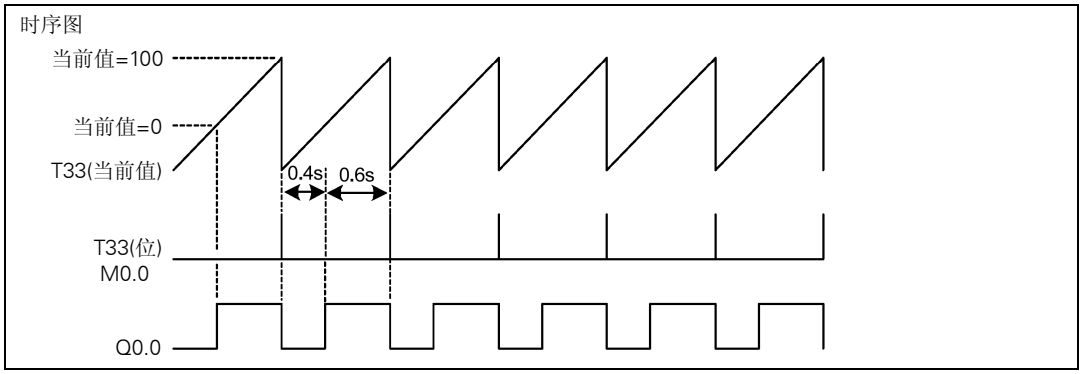
创建这个例子程序将使你体会到使用 STEP 7-Micro/WIN 编程有多简单。这个例子程序在三个程序段中用 6 条指令，完成了一个定时器自启动、自复位的简单功能。

在本例中，你用梯形图编辑器来录入程序。下面给出了完整的梯形图和语句表程序。语句表中的注释，解释了程序的逻辑关系。时序图给出了程序的操作结果。

STEP 7-Micro/WIN 使用入门例子程序




```
Network 1       // 10ms 定时器 T33 在 1 秒后输出。
                // M0.0 输出脉冲过窄无法监视。
LDN             M0.0
TON             T33，+100
Network 2       // 比较指令使 Q0.0 闭合 0.6 秒，断开
                // 0.4 秒，便于监视。
LDW>=           T33，+40
=               Q0.0
Network 3       // T33 输出脉冲过窄无法监视。
                // 每过 1 秒通过 M0.0 复位 T33。
LD              T33
=               M0.0
```



打开程序编辑器

点击程序块图标，打开程序编辑器，如图 2-6 所示。

注意指令树和程序编辑器。你可以用拖拽的方式将梯形图指令插入到程序编辑器中。在工具栏图标中有一些命令的快捷方式。

在输入和保存程序之后，你可以下载程序到 S7-200 中。

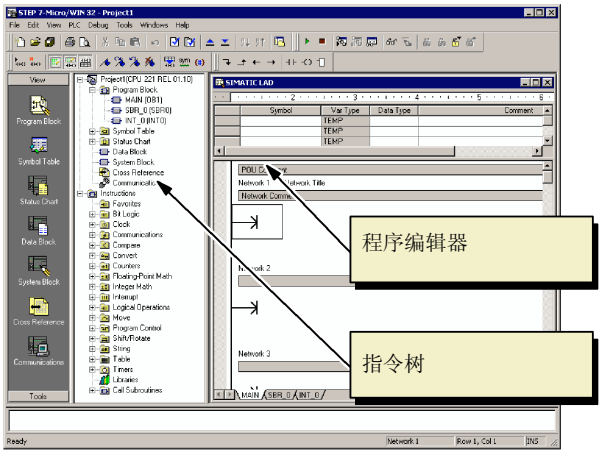


图 2-6 STEP 7-Micro/WIN 窗口

输入程序段 1：启动定时器

当 M0.0 的状态为 0 时，常闭触点接通启动定时器。常闭触点 M0.0 的输入步骤如下：

1. 双击位逻辑图标或者单击其左侧的加号可以显示出全部位逻辑指令。
2. 选择常闭触点。
3. 按住鼠标左键将触点拖到第一个程序段中。
4. 单击“???”并输入地址：M0.0。
5. 按回车键确认。

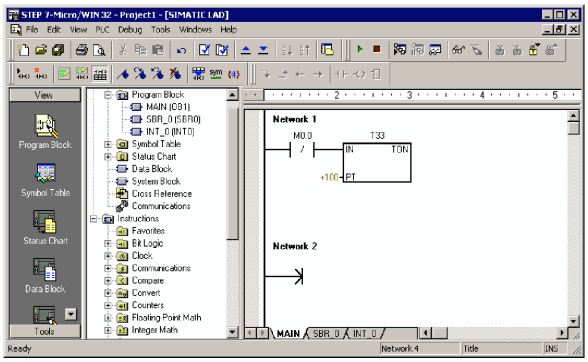


图 2-7 程序段 1

定时器指令 T33 的输入步骤如下：

1. 双击定时器图标，显示定时器指令。
2. 选择延时接通定时器 TON。
3. 按住鼠标左键将定时器拖到第一个程序段中。
4. 单击定时器上方的“???”输入定时器号：T33。
5. 按回车键确认后，光标会自动移动到预置时间值（PT）参数。
6. 输入预置时间值：100。
7. 按回车键确认。

输入程序段 2：使输出点闭合

当定时器 T33 的定时值大于等于 40 时（大于等于 0.4 秒），S7-200 的输出点 Q0.0 会闭合。

输入比较指令的步骤如下：

1. 双击比较指令图标，显示所有的比较指令。选择“>=I”指令。
2. 按住鼠标左键将比较指令拖到第二个程序段中。
3. 单击触点上方的“???”并输入定时器号：T33。

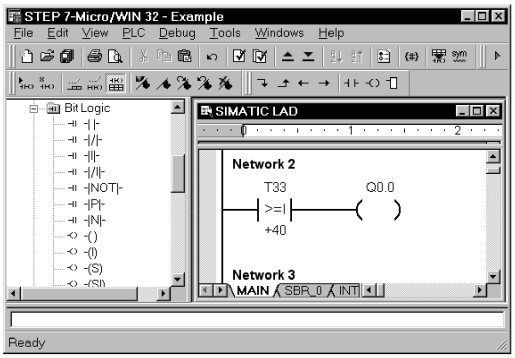


图 2-8 程序段 2

4. 按回车键确认后，光标会自动移动到比较指令下方的比较值参数。
5. 在该处输入比较值 40。
6. 按回车键确认。

输出指令的输入步骤如下：

1. 双击位逻辑图标，显示位逻辑指令并选择输出线圈。
2. 按住鼠标左键将输出线圈拖到第二个程序段中。
3. 单击线圈上方的“???”并输入地址：Q0.0。
4. 按回车键确认。

输入程序段 3：定时器复位

当计时值到达预置时间值（100）时，定时器触点会闭合。T33 闭合会使 M0.0 置位。由于定时器是靠 M0.0 的常闭触点启动的，M0.0 的状态由 0 变 1 会使定时器复位。

输入触点 T33 的步骤如下：

1. 在位逻辑指令中选择常开触点。
2. 按住鼠标左键将触点拖到第三个程序段中。
3. 单击触点上方的“???”并输入地址：T33。

4. 按回车键确认。

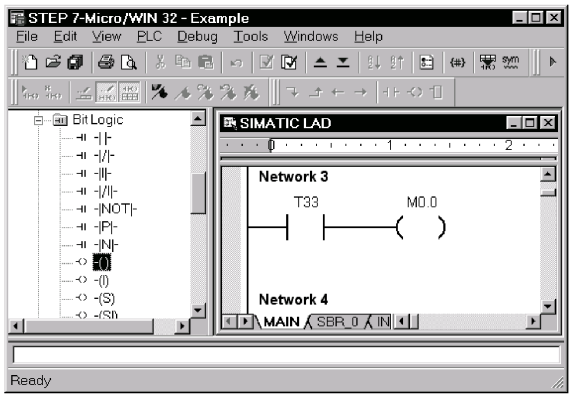


图 2-9 程序段 3

输入线圈 M0.0 的步骤如下：

1. 在位逻辑指令中选择输出线圈。
2. 按住鼠标左键将输出线圈拖到第三个程序段中。
3. 双击线圈上方的“???”并输入地址：M0.0。
4. 按回车键确认。

存储例子程序

在输入完以上三个程序段后，你就已经完成了整个例子程序。当你存储程序时，你也创建了一个包括 S7-200 CPU 类型及其它参数在内的一个项目。存储项目的步骤如下：

1. 在菜单条中选择菜单命令 File>Save As。
2. 在 Save As 对话框中输入项目名。
3. 点击 Save 存储项目。

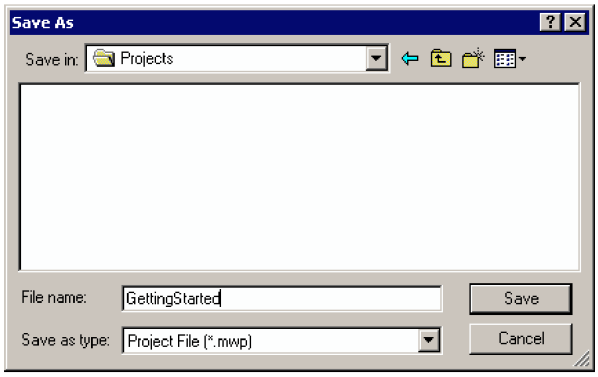


图 2-10 存储例子程序

项目存储之后，你可以下载程序到 S7-200。

下载例子程序

提示：

每一个 STEP 7-Micro/WIN 项目都会有一个 CPU 类型（CPU 221、CPU 222、CPU 224、CPU 226 或 CPU 226XM）。如果你在项目中选择的 CPU 类型，与你实际连接的 CPU 类型不匹配，STEP 7-Micro/WIN 会提示你并要你作出选择。如果你在本例中遇到这种情况，可以选择“继续下载”。

1. 你可以点击工具条中的下载图标或者在命令菜单中选择 **PLC>Download** 来下载程序。
2. 点击 OK 下载程序到 S7-200。如果你的 S7-200 处于运行模式，将有一个对话提示你 CPU 将进入停止模式。点击 Yes 将 S7-200 转入停止模式。

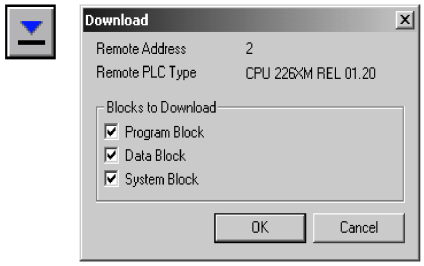


图 2-11 下载例子程序

将 S7-200 转入运行模式

如果想通过 STEP 7-Micro/WIN 软件将 S7-200 转入运行模式，S7-200 的模式开关必须设置为 TERM 或者 RUN。当 S7-200 转入运行模式后，程序开始运行：

1. 单击工具条中的运行图标或者在命令菜单中选择 **PLC>RUN**。
2. 点击 Yes 切换模式。

当 S7-200 转入运行模式后，CPU 将执行程序使 Q0.0 的 LED 指示灯时亮时灭。

图 2-12 将 S7-200 转入运行模式

恭喜你！你已经完成了你的第一个 S7-200 程序。

你可以通过选择 **Debug>Program Status** 来监控程序。

STEP 7-Micro/WIN 显示执行结果。你可以用点击 STOP 图标或者在命令菜单中选择 **PLC> STOP** 来停止程序的运行。

S7-200 的安装

3

S7-200 的设计使其便于安装。可以利用安装孔把模块固定在控制柜的背板上，或者利用设备上的 DIN 夹子，把模块固定在一个标准(DIN)的导轨上。体积小巧的 S7-200 可以使你更为有效地安排空间。

本章提供 S7-200 系统的安装和接线的指导。

本章内容：

S7-200 设备的安装指南

S7-200 既可以安装在控制柜背板上，也可以安装在标准 DIN 导轨上；既可以水平安装，也可以垂直安装。

将 S7-200 与加热装置、高电压和电子噪声隔离开

按照惯例，在安装元器件时，总是把产生高电压和高电子噪声的设备与诸如 S7-200 这样的低压、逻辑型的设备分隔开。

在控制柜背板上安排 S7-200 时，应考虑把产热器件和电子器件安排在控制柜中温度较低的区域内。电子器件在高温环境下工作会缩短其无故障时间。

要考虑控制柜背板的布线，应该避免把低压信号线和通讯电缆与交流供电线和高能量、开关频率很高的直流信号线设计在同一个线槽中。

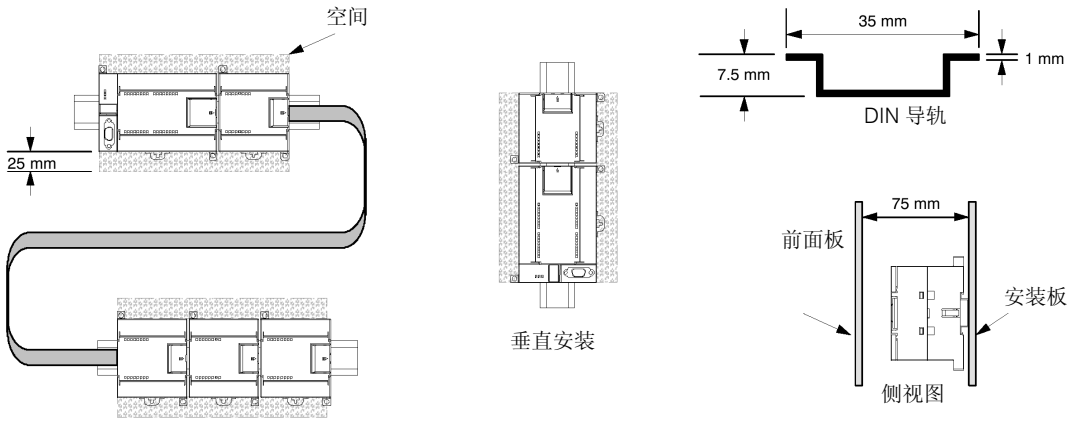
为接线和散热留出适当的空间

S7-200 设备的设计采用自然对流散热方式，在器件的上方和下方都必须留有至少 25mm 的空间，以便于正常的散热。前面板与背板的板间距离也应保持至少 75mm。

提示：

在垂直安装时，其允许的最高环境温度要比水平安装时低 10℃，而且 CPU 应安装在所有扩展模块的下方。

在安排 S7-200 设备时，应留出接线和连接通讯电缆的足够空间。当配置 S7-200 系统时，可以灵活地使用 I/O 扩展电缆。



（一套 S7-200 只允许使用一根 I/O 扩展电缆）

图 3-1 安装要求

电源预算

所有的 S7-200CPU 都有一个内部电源，为 CPU 自身、扩展模块和其它用电设备提供 24V 直流电源。

S7-200 为系统中的所有扩展模块提供 5V 直流逻辑电源。必须格外注意你的系统配置，要确保 CPU 所提供的 5V 电源，能够满足你所选择的所有扩展模块的需要。如果你的配置要求超出了 CPU 的供电能力，你只有去掉一些模块或者选择一个供电能力更强的 CPU。在附录 A 中，你会得到有关 S7-200 CPU 5V 直流逻辑电源的供电能力以及扩展模块对 5V 直流电源需求的信息。附录 B 中给出了 CPU 所能为系统提供电流大小的计算方法。

S7-200 的所有 CPU 也提供 24V 直流传感器供电，此 24VDC 可以为输入点、扩展模块上的继电器线圈或者其它设备供电。如果设备用电量超过了传感器供电预算，必须为系统另配一个外部 24VDC 供电电源。对于特定的 S7-200 CPU，可以在附录 A 中查到其 24VDC 传感器供电电源预算。

如果你使用了外部 24VDC 供电电源，要确保该电源没有与 S7-200 CPU 上的传感器电源并联使用。为了加强电子噪音保护，建议将不同电源的公共端（M）连在一起。

警告

将外部 24VDC 电源与 S7-200 的 24VDC 传感器供电电源并联，会造成两路供电之间的冲突，每一路电源都试图建立自己的电位输出。

这种冲突的结果会缩短电源寿命，或者一路或二路电源立即损坏，这样会使 PLC 系统产生一系列不确定的操作。这种不确定的操作会造成死亡或者严重的人身伤害和设备损坏。

S7-200 的 24VDC 传感器供电和任何外部供电应该分别给不同的点提供电源。

S7-200 模块的安装与拆卸

S7-200 能够安装在一个标准的 DIN 导轨上，或者安装在面板上。

先决条件

在安装和拆卸电子器件之前，要确保该设备的供电已被切断。同样，也要确保与该设备相关联的设备的供电已被切断。

警告

试图在带电情况下安装或拆卸 S7-200 及其相关设备有可能导致电击或者设备误动作。

在安装和拆卸 S7-200 及其相关设备时，如果未能切断所有电源，有可能造成死亡或严重的人身伤害和设备损坏。

在安装和拆卸 S7-200 及其相关设备时，必须预先采取适当的安全措施并且确认 S7-200 的供电被切断。

在更换或安装 S7-200 器件时，要确保使用了正确的模块或等同的模块。

警告

如果你安装了不正确的模块，S7-200 的程序可能会产生错误的功能。

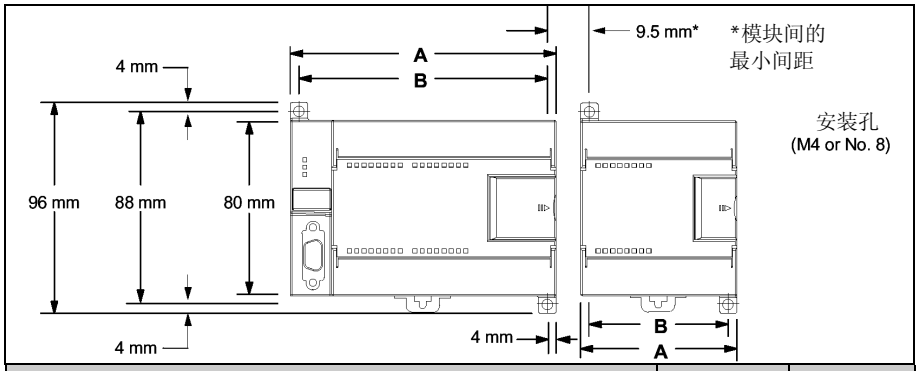
如果未能使用相同的模块按照相同的方向和顺序替换 S7-200 的器件，有可能导致死亡或者严重的人身伤害和设备损坏。

在更换 S7-200 的器件时，除了要使用相同的模块外，还要确保安装的方向和位置是正确的。

安装尺寸

S7-200 的 CPU 和扩展模块都有安装孔，可以很方便地安装在背板上。表 3-1 给出了安装尺寸。

表 3-1 安装尺寸



S7-200 Module	宽度 A	宽度 B
CPU 221 和 CPU 222	90 mm	82mm
CPU 224	120.5 mm	112.5mm
CPU226 和 CPU 226XM	196 mm	188mm
扩展模块： 8 点直流和继电器 I/O（8I，8Q 和 4I/4Q）	46 mm	38mm
扩展模块： 16 点数字量 I/O（8I/8Q），模拟量 I/O（4I，4AI/1AQ，2AQ），RTD，TC，PROFIBUS，ASI，8 点交流 I/O（8I 和 8Q），定位模块和 Modem 模块。	71.2mm	63.2mm
扩模模块： 32 点数字量 I/O（16I/16Q）	137.3mm	129.3mm

CPU 和扩展模块的安装

按照以下步骤安装 S7-200。

面板安装

1．按照表 3-1 的尺寸要求定位打孔（用 M4 或者美国标准 8 号螺钉）。

2．用合适的螺钉将模块固定在背板上。

3．如果你使用了扩展模块，将扩展模块的扁平电缆连到前盖下面的扩展口。

DIN 导轨安装

1．将导轨固定在背板上，保持间距 75mm。

2．打开模块底部的 DIN 夹子，将模块背部卡在 DIN 导轨上。

3．如果你使用了扩展模块，将扩展模块的扁平电缆连到前盖下面的扩展口。

4．旋转模块贴近 DIN 导轨，合上 DIN 夹子。仔细检查模块上 DIN 夹子与 DIN 导轨是否紧密固定好。为避免模块损坏，不要直接按压模块正面，而要按压安装孔的部分。

注意

当 S7-200 的使用环境震动比较大或者采用垂直安装方式时，应该使用 DIN 导轨挡块。

如果系统处于高震动环境中，使用背板安装方式可以得到较高的震动保护等级。

拆卸 CPU 或者扩展模块

按照以下步骤拆卸 CPU 或者扩展模块：

1．拆除 S7-200 的电源。

2．拆除模块上的所有连线和电缆。大多数的 CPU 和扩展模块有可拆卸的端子排，使这项工作变得简单。

3．如果有其它扩展模块连接在你所拆卸的模块上，请打开前盖，拔掉相邻模块的扩展扁平电缆。

4．拆掉安装螺钉或者打开 DIN 夹子。

5．拆下模块。

拆卸和安装端子排

为了安装和替换模块方便，大多数的 S7-200 模块都有可拆卸的端子排。附录 A 中给出了哪些 S7-200 模块有可拆卸的端子排。你可以为不带端子排的模块订购，订货号参见附录 E。

端子排的拆卸

1．打开端子排安装位置的上盖。

2．把螺丝刀插入端子块中央的槽口中。

3．如图 3-2 所示用力下压并撬出端子排。

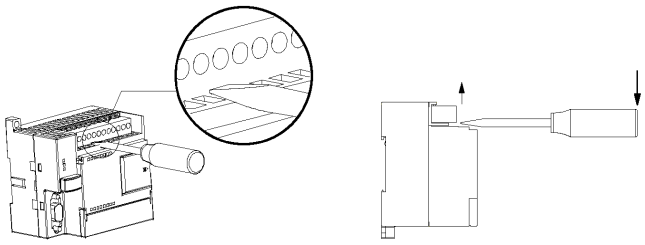


图 3-2 拆卸端子排

端子排的安装

1．打开端子排安装位置的上盖。

2．确保模块上的插针与端子排边缘的小孔对正。

3．将端子排向下压入模块。确保端子块对准了位置并锁住。

接地和接线指南

对所有的电器设备合理的接地和接线是非常重要的，它能够确保你的系统具备最优的操作特性，同时能够为你的系统提供更好的电子噪声保护。

先决条件

在接地和接线之前，必须先确保设备的电源已被切断，也要保证与该设备相关的设备电源已被切断。

在对 S7-200 及其相关设备接线时，必须确保遵从所有有效的电气编码规则。安装和操作所有设备要符合所有有效的国家或地区标准。同地区的权威保持联系，以确定哪些标准符合你的特殊需要。

警告

试图在带电情况下安装或拆卸 S7-200 及其相关设备有可能导致电击或者设备误动作。在安装和拆卸 S7-200 及其相关设备时，如果未能切断所有电源，有可能造成死亡或严重的人身伤害和设备损坏。

在安装和拆卸 S7-200 及其相关设备时，必须采取适当的安全措施并且确认 S7-200 的供电被切断。

在设计 S7-200 系统的接地和接线时必须考虑安全因素。象 S7-200 这样的控制设备有可能造成它所监控的设备的误动作。因此，你应该执行所有的安全规定以避免人员伤害和设备损坏。

警告

控制设备有可能造成它所控制的设备的误操作。这种误操作有可能导致死亡或者严重的人身伤害和设备损坏。

使用独立于 S7-200 的急停功能、机电互锁或者其它冗余的安全措施。

隔离指南

S7-200 的交流供电和 I/O 点与交流电路之间的隔离为 1500VAC。这些隔离被检验并证明可以作为交流线与低压电路之间的安全隔离。

所有与 S7-200 相连的低压电路，例如 24V 供电，必须与交流线和其它高压之间有安全隔离，符合各种安全标准。这些安全标准包括国际电子安全标准、SELV、PELV、Class II 或者其它安全标准。

警告

使用与交流电路不隔离或者单隔离的电源给低压电路供电，会在安全电路，例如通讯电路或者低压传感器电路中产生不安全电压。

这种高电压会导致死亡或者严重的人身伤害和设备损坏。只有使用高电压到低电压的变换器，才能保证电路安全。

S7-200 接地指南

最佳的接地方案是要保证 S7-200 及其相关设备的所有公共点在一点接地。这个单独的接地点必须直接连接在大地上。为了提高抗电子噪声保护特性，建议将所有直流电源的公共点连接到同一个单一接地点上。同样建议将 24VDC 传感器供电的公共点（M）接地。

所有的接地线应该尽量短并且用较粗的线径（1.50mm^2 或者 14AWG）。

当选择接地点时，应当考虑安全接地要求和对隔离器件的适当保护。

S7-200 接线指南

在设计 S7-200 的接线时，应该提供一个单独的开关，能够同时切断 S7-200 CPU、输入电路和输出电路的所有供电。提供熔断器或断路器等过流保护装置来限制供电线路中的电流。你也可以为每一路输出电路都提供熔断器或其它限流设备作为额外的保护。

在有可能遭受雷击浪涌的线路上安装浪涌抑制器件。避免将低压信号线和通讯电缆与交流线和高能量快速开并的直流线设计在同一个走线槽中。使用双绞线并且用中性线或者公共线与能量线或者信号线相配对。

导线尽量短并且保证线粗能够满足电流要求。端子排适合的线粗为 1.50mm^2 到 0.5mm^2（14AWG 到 22AWG）。使用屏蔽电缆可以得到最佳的抗电子噪声特性。通常将屏蔽层接地可以得到最佳效果。

当输入电路由一个外部电源供电时，要在电路中添加过流保护器件。如果使用 S7-200 CPU 上的 24VDC 传感器供电电源，则无需额外添加过流保护器件，因为此电源已经有限流保护。大多数的 S7-200 模块有可拆卸的端子排。（附录 A 中标明了哪些模块有端子排）。为了防止连接松动，要确保端子排插接牢固，同时也要确保导线牢固地连接在端子排上。为了避免损坏端子排，螺钉不要拧得太紧。螺钉连接的最大扭矩为 0.56N-m（5 inch-pounds）。

为了避免意想不到的电流流入系统，S7-200 在合适的部分提供电气隔离。当你设计系统走线时，应考虑这些隔离。附录 A 中给出了电路中包含哪些隔离及它们的隔离级别。级别低于 1500VAC 的隔离不能作为安全隔离。

提示

在通讯网络中，如果不使用中继器，通讯电缆的最大长度为 50m。S7-200 的通讯口是不隔离的。详细内容参见第 7 章。

抑制电路的设计指南

在使用感性负载时，要加入抑制电路来限制输出关断时电压的升高。抑制电路可以保护输出点不至于因为高感抗开关电流而过早的损坏。另外，抑制电路还可以限制感性负载开关时产生的电子噪声。

提示

抑制电路的有效性取决于应用，你应该调整其参数以适应你的特殊应用。要确保所有器件参数与实际应用相符合。

晶体管输出和控制直流负载的继电器输出

晶体管输出有内部保护，可以适应多种应用。由于继电器型输出既可以连接直流负载，又可以连接交流负载，因而没有内部保护。

图 3-3 给出了直流负载抑制电路的一个实例。在大多数的应用中，用附加的二极管 A 即可，但如果你的应用中要求更快的关断速度，则推荐你加上齐纳二极管 B。确保齐纳二极管能够满足输出电路的电流要求。

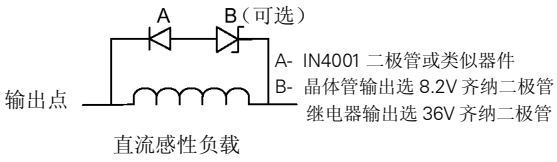


图 3-3 直流负载的抑制电路

交流输出和控制交流负载的继电器输出

交流输出有内部保护，可以适应多种应用。由于继电器型输出既可以连接直流负载，又可以连接交流负载，因而没有内部保护。

图 3-4 给出了交流负载抑制电路的一个实例。在大多数的应用中，附加的金属氧化物可变电阻（MOV）可以限制峰值电压，从而保护 S7-200 内部电路。要确保 MOV 的工作电压比正常的线电压至少高出 20%。

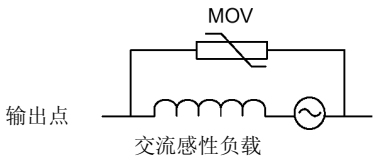


图 3-4 交流负载的抑制电路

PLC 的基本概念

4

S7-200 的基本功能是监视现场的输入，根据你的控制逻辑去控制现场输出设备的接通和关断。本章为你解释有关程序执行、变量类型以及存储区掉电保持等方面的一些概念。

本章内容：

理解 S7-200 如何执行你的控制逻辑

S7-200 周而复始的执行程序中的控制逻辑和读写数据。

S7-200 将你的程序和物理输入输出点联系起来。

S7-200 的基本操作非常简单：

- CPU 读输入状态
- CPU 中存储的程序利用输入执行控制逻辑。当程序运行时，CPU 刷新有关数据。
- CPU 将数据写到输出。

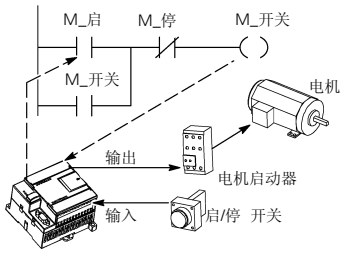


图 4-1 输入和输出的控制

图 4-1 给出了一个简图，说明一个电磁继电器如何连到 S7-200。在本例中，电机启动开关的状态和其它输入点的状态结合在一起。它们计算的结果，最终决定了控制执行机构启动电机的输出点的状态。

S7-200 在扫描循环中完成它的任务

S7-200 周而复始地执行一系列任务。任务循环执行一次称为一个扫描周期。图 4-2 中给出了在一个扫描周期中 S7-200 的全部操作。

- 读输入：S7-200 将物理输入点上的状态复制到输入过程映象寄存器中。
- 执行逻辑控制程序：S7-200 执行程序指令并将数据存储在变量存储区中。
- 处理通讯请求：S7-200 执行通讯任务。
- 执行 CPU 自诊断：S7-200 检查固件、程序存储器和扩展模块是否工作正常。
- 写输出：在输出过程映象寄存器中存储的数据被复制到物理输出点。

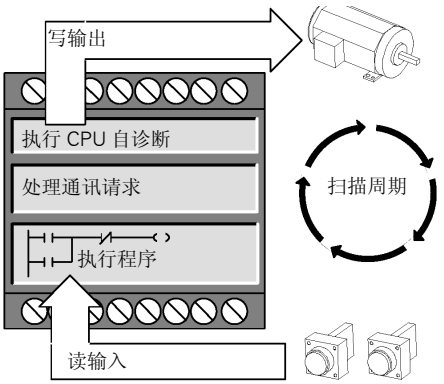


图 4-2 S7-200 扫描周期

扫描周期的执行取决于 S7-200 是处于停止模式还是处于运行模式。当 S7-200 处于运行模式时，CPU 执行程序；当 S7-200 处于停止模式时，CPU 不执行程序。

读输入

数字量输入：在每一个扫描周期的开始，CPU 会读数字量输入并把这些值写入输入过程映象寄存器。

模拟量输入：除非使能模拟量滤波，否则 CPU 在扫描周期中并不刷新模拟量输入值。模拟量滤波会使你得到较稳定的信号。你可以使能每个模拟量输入通道的滤波。

当你使能了模拟量输入滤波功能后，S7-200 会在每一个扫描周期刷新模拟量、执行滤波功能并且在内部存储滤波值。当程序中访问模拟量输入时使用滤波值。

如果你没有使能模拟量输入滤波，当程序访问模拟量输入时，CPU 每次从物理模块直接读取模拟值。

提示

模拟量滤波会使你得到较稳定的信号。在模拟量输入信号随时间变化缓慢时使用模拟量输入滤波。如果信号变化很快，不应该选用模拟量滤波。

对于在模拟量值中传递数字信息或者报警指示的模拟量不要使用模拟量滤波。对于 RTD、TC 和 ASI 模块，不能使用模拟量滤波。

执行程序

在扫描周期的执行程序阶段，CPU 从头至尾执行应用程序。在程序或中断服务程序中，直接 I/O 指令允许你对 I/O 点直接存取。

如果在程序中使用了中断，与中断事件相关的中断服务程序作为程序的一部分被存储。中断程序并不作为正常扫描周期的一部分来执行，而是当中断事件发生时才执行（可能在扫描周期的任意点）。

处理通讯请求

在扫描周期的处理阶段，CPU 处理从通讯端口或者智能 I/O 模块接收到的任何信息。

执行 CPU 自诊断测试

在扫描周期的这个阶段，S7-200 检测 CPU 的操作、存储区以及扩展模块的状态是否正常。

写数字输出

在每个扫描周期的结尾，CPU 把存储在输出映象寄存器中的数据写到数字输出点。（模拟量输出直接刷新，与扫描周期无关。）

S7-200 数据的存取

S7-200 将信息存于不同的存储器单元，每个单元都有唯一的地址。你可以明确指出要存取的存储器地址。这就允许用户程序直接存取这个信息。表 4-1 给出了不同长度的数据所能表示的数值范围。

表 4-1 不同长度的数据表示的十进制和十六进制数范围

数据长度	字节（B）	字（W）	双字（D）
无符号整数	0 到 255 0 到 FF	0 到 65,535 0 到 FFFF	0 到 4,294,967,295 0 到 FFFF FFFF
符号整数	-128 到＋127 80 到 7F	-32,768 到+32,767 8000 到 7FFF	-2,147,483,648 到+2,147,483,647 8000 0000 到 7FFF FFFF
实数 IEEE 32 位浮点数			+1.175495E-38 到+3.402823E+38(正数) -1.175495E-38 到-3.402823E+38(负数)

若要存取存储器区域的某一位，则必须指定地址，包括存储器标识符、字节地址和位号。

图 4-3 是一个位寻址的例子（也称为“字节.位”寻址）。在这个例子中，存储器区、字节地址（I 代表输入，3 代表字节 3）和位地址（第 4 位）之间用点号“.”相隔开。

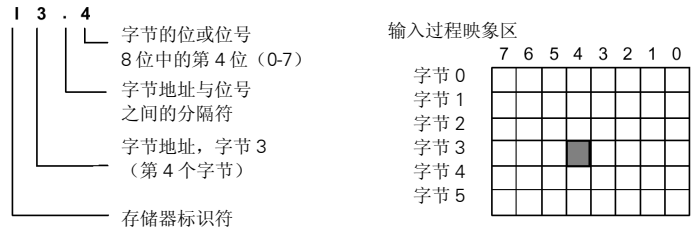


图 4-3 字节位寻址

使用这种字节寻址方式，可以按照字节、字或双字来存取许多存储器区域（V、I、Q、M、S、L 及 SM）中的数据。若要存取 CPU 中的一个字节、字或双字数据，则必须以类似位寻址的方式给出地址，包括存储器标识符、数据大小以及该字节、字或双字的起始字节地址，如图 4-4 所示。

其它 CPU 存储器区域（如 T，C，HC 和累加器）中存取数据使用的地址格式包括区域标识符和设备号。

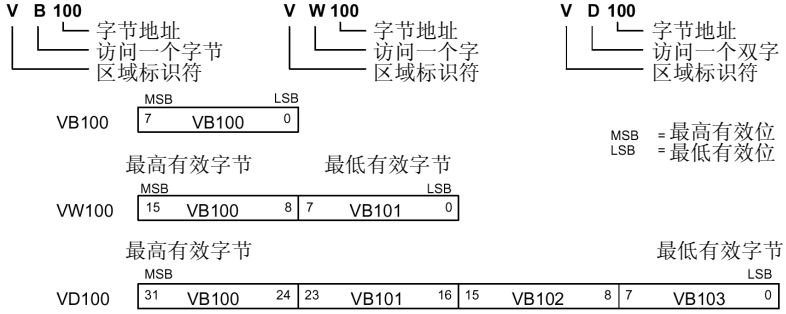


图 4-4 对同一地址进行字节，字和双字存取操作的比较。

存储区数据的存取

输入过程映象寄存器：I

在每次扫描周期的开始，CPU 对物理输入点进行采样，并将采样值写入输入过程映象寄存器中。可以按位、字节、字或双字来存取输入过程映象寄存器中的数据。

位：	I[字节地址].[位地址]	I0.1
字节、字或双字：	I[长度][起始字节地址]	IB4

输出过程映象寄存器：Q

在每次扫描周期的结尾，CPU 将输出过程映象寄存器中的数值复制到物理输出点上。可以按位、字节、字或双字来存取输出过程映象寄存器中的数据。

位：	Q[字节地址].[位地址]	Q1.1
字节、字或双字：	Q[长度][起始字节地址]	QB5

变量存储区：V

你可以用 V 存储器存储程序执行过程中控制逻辑操作的中间结果，也可以用它来保存与工序或任务相关的其它数据。可以按位、字节、字或双字来存取 V 存储器中的数据。

位：	V[字节地址].[位地址]	V10.2
字节、字或双字：	V[长度][起始字节地址]	VW100

位存储区：M

可以用位存储区作为控制继电器来存储中间操作状态和控制信息。可以按位、字节、字或双字来存取位存储区中的数据。

位：	M[字节地址].[位地址]	M26.7
字节、字或双字：	M[长度][起始字节地址]	MD20

定时器存储区 T：

S7-200 CPU 中，定时器可用于时间累计，其分辨率（时基增量）分为 1ms、10ms 和 100ms 三种。定时器有两种形式：

- 当前值：16 位有符号整数，存储定时器所累计的时间。
- 定时器位：按照当前值和预置值的比较结果置位或者复位。预置值是定时器指令的一部分。

可以用定时器地址（T＋定时器号）来存取这两种形式的定时器数据。究竟使用哪种形式取决于所使用的指令，如果使用位操作指令则是存取定时器位；如果使用字操作指令，则是存取定时器当前值。如图 4-5 中所示，常开触点指令是存取定时器位；而字移动指令则是存取定时器的当前值。

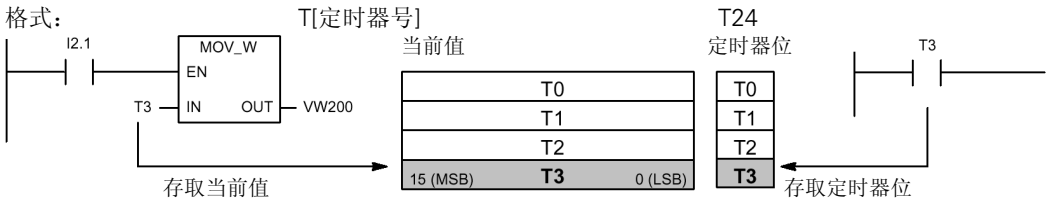


图 4-5 存取定时器位或者定时器的当前值

计数器存储区：C

在 S7-200 CPU 中，计数器可以用于累计其输入端脉冲电平由低到高的次数。CPU 提供了三种类型的计数器：一种只能增计数；一种只能减计数；另外一种既可以增计数，又可以减计数。计数器有两种形式：

- 当前值：16 位有符号整数，存储累计值。
- 计数器位：按照当前值和预置值的比较结果来置位或者复位。预置值是计数器指令的一部分。

可以用计数器地址（C＋计数器号）来存取这两种形式的计数器数据。究竟使用哪种形式取决于所使用的指令，如果使用位操作指令则是存取计数器位；如果使用字操作指令，则是存取计数器当前值。如图 4-6 中所示，常开触点指令是存取计数器位；而字移动指令则是存取计数器的当前值。

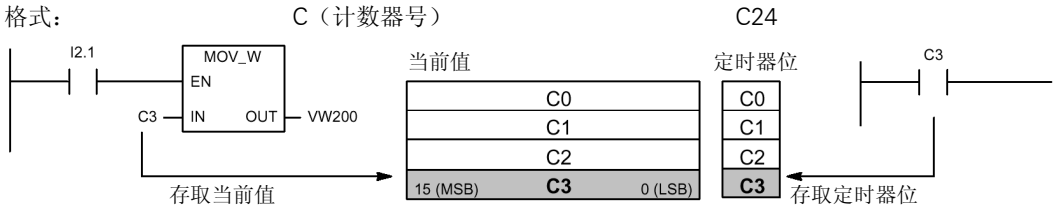


图 4-6 存取计数器位或者计数器的当前值

高速计数器：HC

高速计数器对高速事件计数，它独立于 CPU 的扫描周期。高速计数器有一个 32 位的有符号整数计数值（或当前值）。若要存取高速计数器中的值，则应给出高速计数器的地址，即存储器类型（HC）加上计数器号（如 HC0）。高速计数器的当前值是只读数据，可作为双字（32 位）来寻址。

格式：　　HC[高速计数器号]　　HC1

累加器：AC

累加器是可以象存储器一样使用的读写设备。例如，可以用它来向子程序传递参数，也可以从子程序返回参数，以及用来存储计算的中间结果。S7-200 提供 4 个 32 位累加器（AC0，AC1，AC2 和 AC3）。可以按字节、字或双字的形式来存取累加器中的数值。

被访问的数据长度取决于存取累加器时所使用的指令。如图 4-7 所示，当以字节或者字的形式存取累加器时，使用的是数值的低 8 位或者低 16 位。当以双字的形式存取累加器时，使用全部 32 位。

关于如何在中断服务程序中使用累加器的相关信息，参见第 6 章中的中断指令部分。

格式：　　AC[累加器号]　　AC0

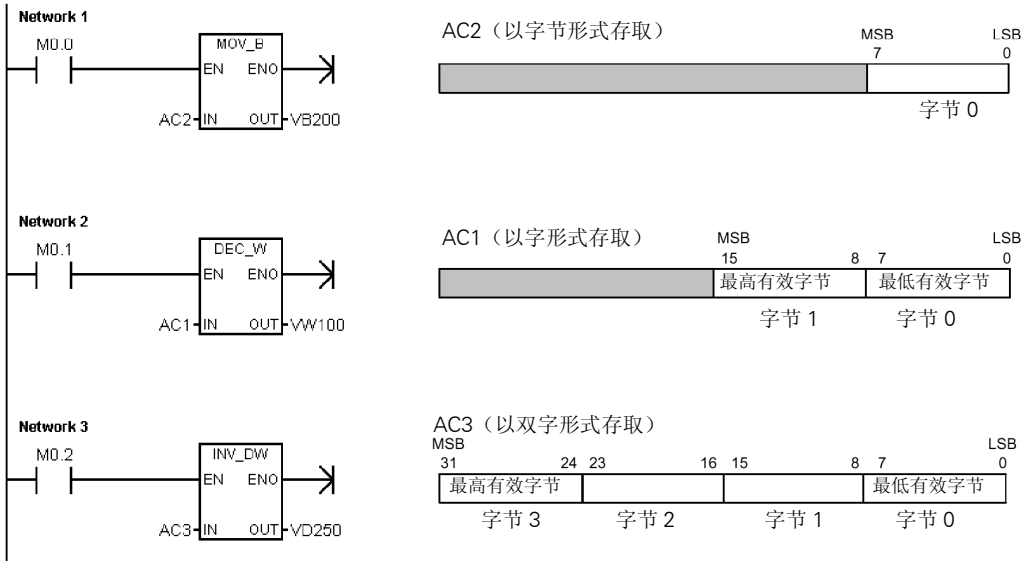


图 4-7 存取累加器

特殊存储器：SM

SM 位为 CPU 与用户程序之间传递信息提供了一种手段。可以用这些位选择和控制 S7-200 CPU 的一些特殊功能。例如：首次扫描标志位、按照固定频率开关的标志位或者显示数学运算或操作指令状态的标志位。（有关 SM 位的详细信息参见附录 D）。你可以按位、字节、字或者双字的形式来存取。

位：	SM[字节地址].[位地址]	SM0.1
字节、字或者双字：	SM[长度][起始字节地址]	SMB86

局部存储器：L

S7-200 有 64 个字节的局部存储器，其中 60 个可以用作临时存储器或者给子程序传递参数。

提示

如果用梯形图或功能块图编程，STEP7-Micro/WIN 保留这些局部存储器的最后四个字节。如果用语句表编程，可以寻址所有的 64 个字节，但是建议不要使用局部存储器的最后四个字节。

局部存储器和变量存储器很相似，主要区别是变量存储器是全局有效的，而局部存储器只在局部有效。全局是指同一个存储器可以被任何程序存取（包括主程序、子程序和中断服务程序）。局部是指存储器区和特定的程序相关联。S7-200 给主程序分配 64 个局部存储器；给每一级子程序嵌套分配 64 个字节局部存储器；同样给中断服务程序分配 64 个字节局部存储器。

子程序或者中断服务程序不能访问分配给主程序的局部存储器。子程序不能访问分配给主程序、中断服务程序或者其它子程序的局部存储器。同样的，中断服务程序也不能访问分配给主程序或子程序的局部存储器。

S7-200 PLC 根据需要分配局部存储器。也就是说，当主程序执行时，分配给子程序或中断服务程序的局部存储器是不存在的。当发生中断或者调用一个子程序时，需要分配局部存储器。新的局部存储器在分配时可以重新使用分配给另一个子程序或中断服务程序的局部存储器。

局部存储器在分配时 PLC 不进行初始化，初值可能是任意的。当在子程序调用中传递参数时，在被调用子程序的局部存储器中，由 CPU 替换其被传递的参数的值。局部存储器在参数传递过程中不传递值，在分配时不被初始化，也没有任何数值。

位：	L[字节地址].[位地址]	L0.0
字节、字或双字：	L[长度][起始字节地址]	LB33

模拟量输入：AI

S7-200 将模拟量值（如温度或电压）转换成 1 个字长（16 位）的数字量。可以用区域标识符（AI）、数据长度（W）及字节的起始地址来存取这些值。因为模拟输入量为 1 个字长，且从偶数位字节（如 0，2，4）开始，所以必须用偶数字节地址（如 AIW0，AIW2，AIW4）来存取这些值。模拟量输入值为只读数据。

格式：	AIW[起始字节地址]	AIW4

模拟量输出：AQ

S7-200 把 1 个字长（16 位）数字值按比例转换为电流或电压。可以用区域标识符（AQ）、数据长度（W）及字节的起始地址来改变这些值。因为模拟量为一个字长，且从偶数字节（如 0，2，4）开始，所以必须用偶数字节地址（如 AQW0，AQW2，AQW4）来改变这些值。模拟量输出值为只写数据。

格式：	AQW[起始字节地址]	AQW4

顺控继电器存储器：S

顺控继电器位（S）用于组织机器操作或者进入等效程序段的步骤。SCR 提供控制程序的逻辑分段。可以按位、字节、字或者双字来存取 S 位。

位：	S[字节地址].[位地址]	S3.1
字节、字或双字：	S[长度][起始字节地址]	SB4

实数的格式

实数（浮点数）由 32 位单精度数表示，其格式按照 ANSI/IEEE 754-1985 标准中所描述的形式，如图 4-8 中所示。实数按照双字长度存取。

对于 S7-200 来说，浮点数精确到小数点后第六位。因而当你使用一个浮点数常数时，你最多可以指定到小数点后第六位。

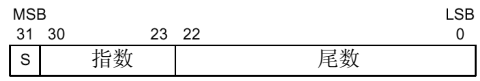


图 4-8 实数的格式

实数运算的精度

在计算中涉及到非常大和非常小的数，则有可能导致计算结果不精确。例如 10^x 中当 x ＞6 时会出现以下结果。

100 000 000+1=100 000 000

字符串的格式

字符串指的是一系列字符，每个字符以字节的形式存储。字符串的第一个字节定义了字符串的长度，也就是字符的个数。4-9 给出了一个字符串的格式。一个字符串的长度可以是 0 到 254 个字符，再加上长度字节，一个字符串的最大长度为 255 个字节。

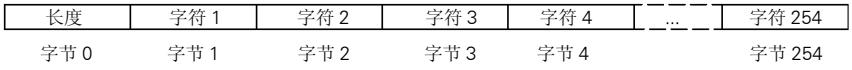


图 4-9 字符串的格式

在 S7-200 指令中输入常数值

在 S7-200 的许多指令中，都可以使用常数值。常数可以是字节、字或者双字。S7-200 以二进制数的形式存储常数，可以分别表示十进制数、十六进制数、ASCII 码或者实数（浮点数）。如表 4-2 所示。

表 4-2 常数表示法

数制	格式	举例
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 码	'[ASCII 码文本]'	'Text goes between single quotes.'
实数	ANSI/IEEE 754-1985	+1.175495E-38（正数）-1.175495E-38（负数）

提示

S7-200CPU 不支持数据类型检测（例如指定常数存储为一个整数、有符号整数或者双整数）。例如：可以在加法指令中使用 VW100 中的值作为有符号整数，同时也可以在异或指令中将 VW100 中的数据当作无符号的二进制数。

本地 I/O 和扩展 I/O 的寻址

CPU 提供的本地 I/O 具有固定的 I/O 地址。你可以将扩展模块连接到 CPU 的右侧来增加 I/O 点，形成 I/O 链。对于同种类型的输入输出模块而言，模块的 I/O 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。类似的，模拟量模块不会影响数字量模块的地址，反之亦然。

提示

数字量模块总是保留以 8 位（1 个字节）递增的过程映象寄存器空间。如果模块没有给保留字节中每一位提供相应的物理点，那些未用位不能分配给 I/O 链中的后续模块。对于输入模块，这些保留字节中的未用位会在每个输入刷新周期中被清零。

模拟量扩展模块总是以两点递增的方式来分配空间。如果模块没有给每个点分配相应的物理点，那么这些 I/O 点会消失并且不能够分配给 I/O 链中的后续模块。

图 4-10 中是一个特定的硬件配置中的 I/O 地址。地址间隙（用灰色斜体文字表示）无法在程序中使用。

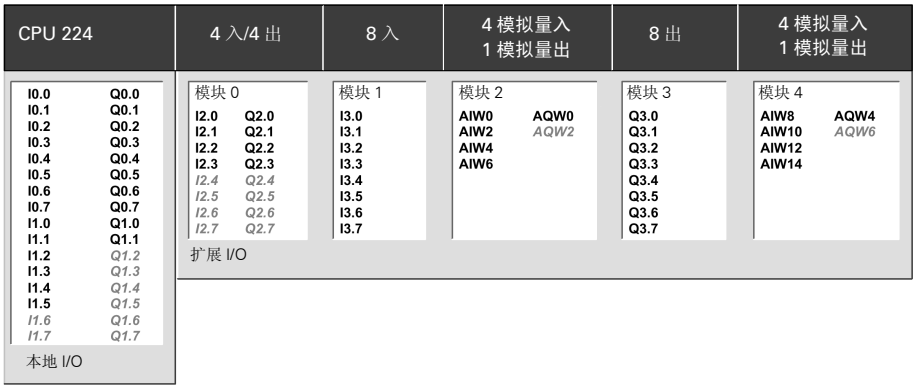


图 4-10 CPU224 的本地和扩展 I/O 地址举例

用指针对 S7-200 存储区间接寻址

间接寻址是指用指针来访问存储区数据。指针以双字的形式存储其它存储区的地址。只能用 V 存储器、L 存储器或者累加器寄存器（AC1、AC2、AC3）作为指针。要建立一个指针，必须以双字的形式，将需要间接寻址的存储器地址移动到指针中。指针也可以为子程序传递参数。

S7-200 允许指针访问以下存储区：I、Q、V、M、S、T（仅限于当前值）和 C（仅限于当前值）。你无法用间接寻址的方式访问位地址，也不能访问 AI、AQ、HC、SM 或者 L 存储区。

要使用间接寻址，你应该用“&”符号加上要访问的存储区地址来建立一个指针。指令的输入操作数应该以“&”符号开头来表明是存储区的地址而不是其内容移动到指令的输出操作数（指针）中。

当指令中的操作数是指针时，应该在操作数前面加上“＊”号。如图 4-11 所示，输入 *AC1 指定 AC1 是一个指针，MOVW 指令决定了指针指向的是一个字长的数据。在本例中，存储在 VB200 和 VB201 中的数值被移动到累加器 AC0 中。

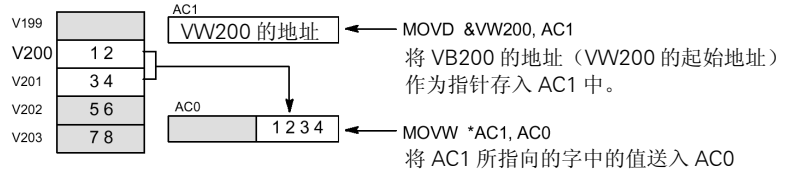


图 4-11 建立和使用指针

如图 4-12 所示，你可以改变一个指针的数值。由于指针是一个 32 位的数据，要用双字指令来改变指针的数值。简单数学运算，如加法指令或者递增指令，可用于改变指针的数值。

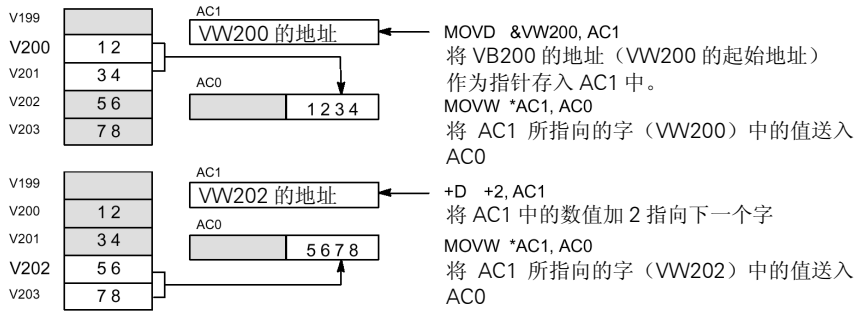


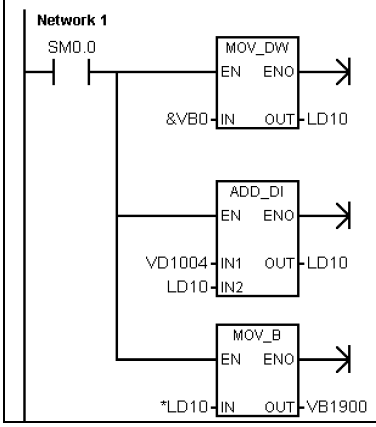
图 4-12 改变指针

提示

按照所访问的数据长度使用不同的指令：当访问字节时，使用递增指令使指针值加 1；当访问字或者计数器、定时器的当前值时，用加法或者递增指令使指针值加 2；当访问双字时，使用加法或者递增指令使指针值加 4。

用地址偏移量来访问 V 存储区数据的例子程序

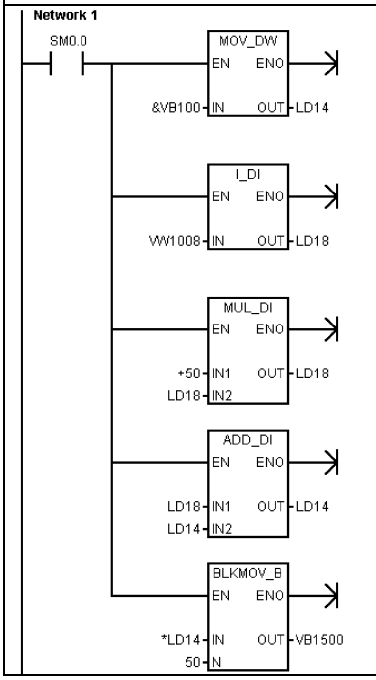
本例中用 LD10 作为 VB0 的地址指针。然后可以利用 VD1004 中存储的地址偏移量来改变指针值。经过改变后，LD10 指向 V 区中的另外一个地址（VB0＋偏移量）。然后将 LD10 指向的 V 区地址中存储的数值复制到 VB1900 中。通过改变 VD1004 中的数值，你可以访问 V 存储器中的任意单元。




```
Network1  //如何利用偏移量访问 V 存储器中的任意单元：
          //1. 将 V 区的起始地址装载到指针中。
          //2. 将偏移量加到指针上。
          //3. 将 V 存储器中的数值复制到 VB1900 中。
LD        SM0.0
MOVD      &VB0，LD10
+D        VD1004，LD10
MOVB      *LD10，VB1900
```

用指针访问数据表的例子程序

本例中用 LD14 作为指向一个配方表的指针，配方表的起始地址为 VB100。在本例中 VW1008 用来存储一个指定的配方在表中的索引号。如果每条配方的长度为 50 个字节，则用这个索引号乘以 50 就可以得到这条配方起始地址的偏移量。用指针加上偏移量，你就可以访问表中的每一条配方。在本例中，配方被复制到从 VB1500 开始的 50 个字节中。




```
Network1  //如何从配方表中传递一条配方：
          //每条配方的长度为 50 个字节。
          //索引值（VW1008）指定装载哪一条配方。
          //1. 建立一个指针指向配方表的起始地址。
          //2. 将配方的索引值转换为双字。
          //3. 乘以每条配方的长度算出偏移量。
          //4. 将偏移量加到指针上。
          //5. 将配方传递至 VB1500 到 VB1549 中。
LD        SM0.0
MOVD      &VB100，LD14
ITD       VW1008，LD18
*D        ＋50，LD18
+D        VD18，LD14
BMB       *LD14，VB1500，50
```

理解 S7-200 如何保存和存储数据

S7-200 提供了多种安全措施来确保用户程序、程序数据和组态数据不丢失。

S7-200 提供了一个超级电容在 CPU 掉电时保存整个 RAM 存储器中的数据。根据 CPU 的类型，超级电容可以保存 RAM 达几天之久。

CPU 提供了一个 EEPROM 来永久保持用户程序、 用户选择的数据区以及组态数据。

S7-200 支持可选的电池卡，当 CPU 掉电后，可以延长 RAM 保持的时间。电池卡只有在超级电容耗尽后才提供电源。

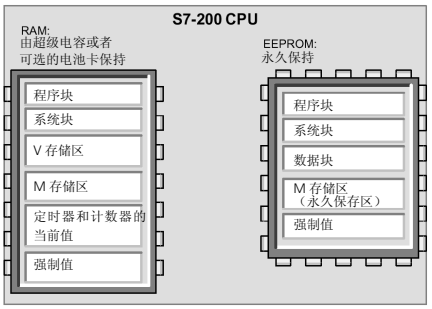


图 4-13 S7-200 CPU 的存储区域

下载和上载用户程序

用户程序包括三部分：程序块、数据块（可选）和系统块（可选）。

图 4-14 中给出了如何下载一个程序到 S7-200 的 CPU。

当你下载程序时，程序存储在 RAM 中。S7-200 会自动将程序块、 数据块和系统块复制到 EEPROM 中作永久保存。

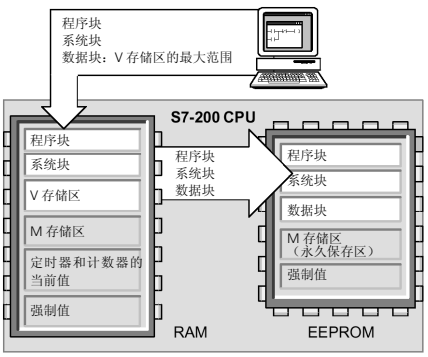


图 4-14 下载程序到 S7-200

图 4-15 中给出了如何从 S7-200 的 CPU 中上载一个程序。

当你上载程序到你的 PC 机时，S7-200 从 RAM 中上载系统块，从 EEPROM 中上载程序块和数据块。

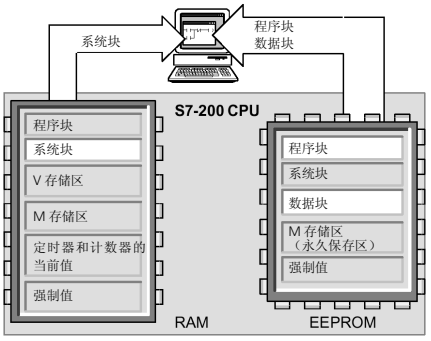


图 4-15 从 S7-200 上载程序

掉电后保存 M 存储区

如果你将 M 存储区的前 14 个字节（MB0 到 MB13）定义为掉电保持，则当 S7-200 CPU 掉电时，这些字节会被永久保存在 EEPROM 中。

如图 4-16 所示，S7-200 将这些 M 存储区移动到 EEPROM 中。

M 存储区的前 14 个字节的缺省设置为不保持。当你给 S7-200 断电时，缺省值使你不能保存这 14 个字节数据到 EEPROM。

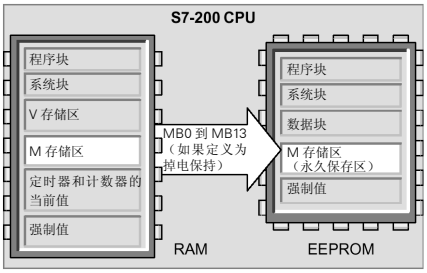


图 4-16 M 存储区的掉电保持

开机后数据的恢复

开机后，S7-200 从 EEPROM 中恢复程序块和系统块，如图 4-17 所示。同时，CPU 检查 RAM 存储器，确认超级电容器是否成功保存了 RAM 存储器中的数据。如果成功保存，则 RAM 存储器的保持区域将保持不变。

V 存储器中的保持区和非保持区，从 EEPROM 中的相应区域恢复回来。如果 RAM 存储器的内容没有保持下来（例如较长时间的断电），CPU 会清除 RAM 存储器（包括保持区和非保持区），并在上电后的第一个扫描周期置保持数据丢失标志位（SM0.2）为“1”。然后，将 EEPROM 中的数据恢复到 RAM 中。

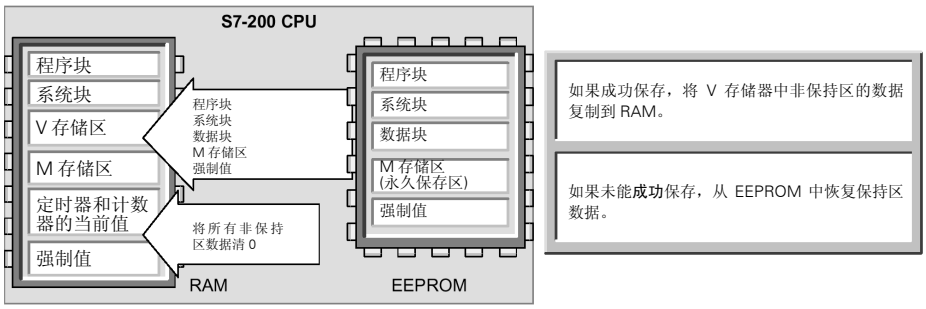


图 4-17 开机后数据的恢复

将应用程序存储到存储卡上

CPU 支持可选的存储卡，为用户程序提供了一个便携式的 EEPROM 存储器。S7-200 在存储卡上存储下列内容：程序块、数据块、系统块和强制值。

只有当 S7-200 上电，处于停止模式并且安装了存储卡时，才能将程序从 RAM 中复制到存储卡中。你可以在上电的情况下插拔存储卡。

注意

静电放电会损坏存储卡或 CPU 接口。

当你拿存储卡时，你应使用接地导电垫或者戴接地手套，还应当把存储卡存放在导电容器中。

要安装存储卡，应先从 S7-200 CPU 上取下塑料盖，然后将存储卡插入槽中。正确安装存储卡至关重要。

将应用程序复制到存储卡中

存储卡安装好之后，按照如下步骤复制程序：

1. 将 S7-200 CPU 置于停止模式。
2. 如果程序尚未下载到 S7-200 CPU 中，先下载程序。
3. 使用菜单命令 PLC＞Program Memory Cartridge 来向存储卡中复制程序。

   图 4-18 中显示了存储卡中存储的 CPU 存储器中的内容。
4. 拆下存储卡，盖上 CPU 上的塑料盖。（可选）

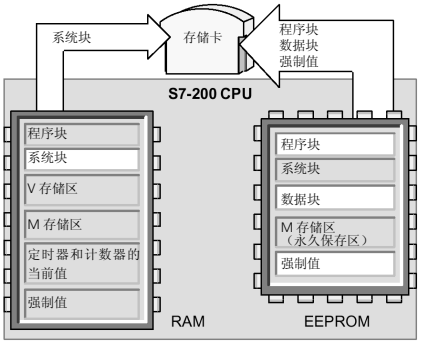


图 4-18 将程序复制到存储卡中

从存储卡中恢复程序

为了把存储卡中的程序传送到 S7-200 CPU 中，你必须先插入存储卡，然后给 CPU 上电。

注意

在 CPU 通电时，若存储卡是空白的或者存储的是不同类型 CPU 的程序，会导致错误。高型号的 CPU 可以读出用低型号 CPU 写入的存储卡，反之则不行。例如：用 CPU221 或者 CPU222 写入的存储卡可以用 CPU224 读出来，但 CPU224 写入的存储卡却不能够用 CPU221 或者 CPU222 读出来。

拆掉存储卡再次给 CPU 上电后，如果需要的话，存储卡可以再次插入和编程。

如图 4-19 所示，当你在插入存储卡之后给 S7-200 上电时，CPU 完成以下操作：

1. 如果存储卡中的内容与 EEPROM 中不同，S7-200 会清除 RAM 区。
2. S7-200 将存储卡中的内容复制到 RAM 中。
3. S7-200 将程序块、系统块和数据块复制到 EEPEOM 中。

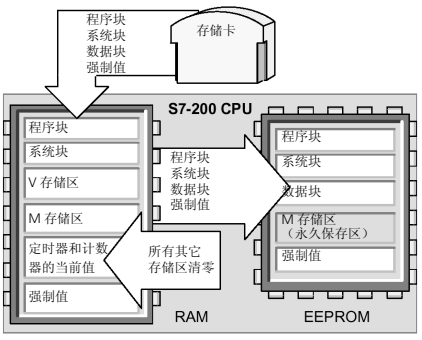


图 4-19 从存储卡中恢复程序

为 S7-200 CPU 选择操作模式

S7-200 有两种操作模式：停止模式和运行模式。CPU 前面板上的 LED 状态显示了当前的操作模式。在停止模式下，S7-200 不执行程序，你可以下载程序和 CPU 组态。在运行模式下，S7-200 运行程序。

- S7-200 提供一个模式开关来改变操作模式。你可以用模式开关（位于 S7-200 前盖下面）手动选择操作模式：可以将模式开关打在停止模式，停止程序的执行；可以将模式开关打在运行模式，启动程序的执行；也可以将模式开关打在 TERM（终端）模式，不改变当前操作模式。

  如果模式开关打在 STOP 或者 TERM 模式，且电源状态发生变化，则当电源恢复时，CPU 会自动进入 STOP 模式。如果模式开关打在 RUN 模式，且电源状态发生变化，则当电源恢复时，CPU 会进入 RUN 模式。
- STEP7-Micro/WIN 允许你改变与之相连的 S7-200 的操作模式。如果希望用软件来改变操作模式，CPU 上的模式开关必须打在 RUN 或者 TERM 上。你可以用菜单命令中的 PLC＞STOP 和 PLC＞RUN 或者工具栏中的相关按钮来改变操作模式。
- 你可以在应用程序中插入 STOP 指令来将 S7-200 置为停止模式。它可以使逻辑程序停止运行，有关 STOP 指令的更多信息参见第 6 章。

用编程的方式将 V 存储区中的数据存入 EEPROM

可以将存储在 V 存储器中的数据（字节、字或者双字）存储到 EEPROM 中。一般来说，一次写 EEPROM 的操作会使扫描周期最多增加 5ms。保存操作中所写入的数据会覆盖先前 EEPROM 中 V 存储区的数据。

存 EEPROM 操作并不更新存储卡中的数据。

提示

由于存 EEPROM 操作的次数是有限的（最少 10 万次，典型值为 100 万次），请注意只有在必要时才进行保存操作。否则，EEPROM 可能会失效，从而引起 CPU 故障。一般来说，当特定事件发生时，才执行存储操作，而特定事件是不很频繁发生的。

例如，如果 S7-200 扫描周期为 50ms，每次扫描存一个数据，那么 EEPROM 最短只能工作 5,000 秒，还不到一个半小时。另一方面，如果每小时存一个数据，则 EEPROM 至少可以工作 11 年。

复制 V 存储区到 EEPROM

特殊存储器字节 31（SMB31）命令 S7-200 复制 V 存储区中的一个数据到 EEPROM 中的 V 存储区。特殊存储器字 32（SMW32）中存储所要复制数据的地址。图 4-20 中给出了 SMB31 和 SMW32 的格式。

采用下列步骤来保存或者写入 V 存储区中的一个特定数值：

1．将要保存的 V 存储器的地址装载到 SMW32 中。

2．将数据长度装载到 SM31.0 和 SM31.1 中。如图 4-20 所示。

3．将 SM31.7 置为 1。

在每次扫描的末尾，CPU 自动检查 SM31.7，如果 SM31.7 等于 1，则将指定的数据存入 EEPROM。当 CPU 将 SM31.7 清零时，操作结束。

在保存操作完成之前，不要改变 V 存储器中的数值。

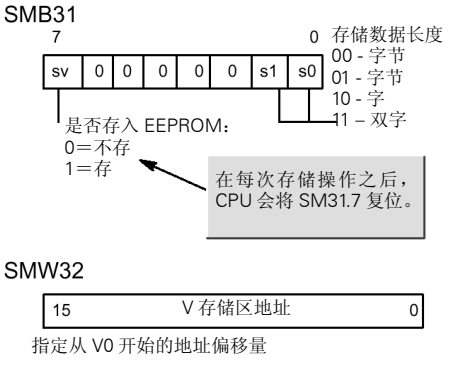
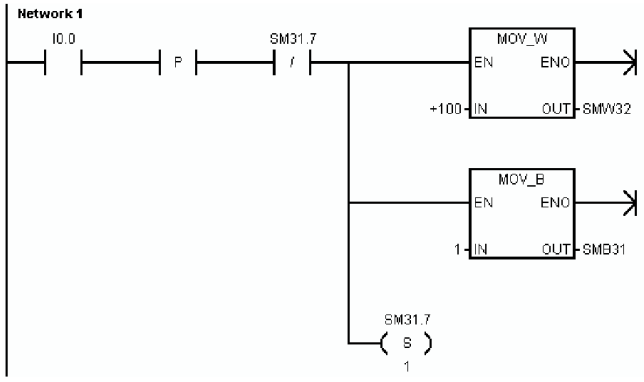


图 4-20 SMB31 和 SMW32

复制 V 存储区到 EEPROM 的例子程序

本例中将 VB100 存入 EEPROM。当 I0.0 有上升沿信号产生，并且没有其它写 EEPROM 操作发生时，将 V 存储器的地址传送到 SMW32。接着选择所要传输数据的长度（1 代表字节，2 代表字，3 代表双字或者实数）。然后将 SM31.7 置位，S7-200 会在扫描末尾传送数据。
当传送完成后，S7-200 会自动复位 SM31.7。




```
Network1  //传送 V 存储器（VB100）
          //到 EEPROM
LD        I0.0
EU
AN        SM31.7
MOVW      ＋100，SMW32
MOVB      1，SMB31
S         SM31.7，1
```

S7-200 的特性

S7-200 提供了几条特殊的性能帮助你更好地运用 S7-200 的功能，完成应用程序。

S7-200 允许你在程序中立即读写 I/O

在 S7-200 的指令集中提供了立即读写物理 I/O 点的指令。尽管通常情况下我们使用映象寄存器作为源地址和目的地址来访问 I/O，但这些立即 I/O 指令却允许我们直接访问真正的输入、输出点。

当使用立即指令访问一个输入点时，相应的过程映象输入寄存器不会发生改变。但当你用立即指令访问一个输出点时，相应的过程映象输出寄存器会同时更新。

提示

当选择模拟量滤波时，S7-200 读到的是模拟量滤波值。当不使用模拟量滤波时，S7-200 读到的是模拟量输入点上的当前值。当你将一个数值写到模拟量输出时，输出值会立即更新。

通常认为在执行应用程序时，用过程映象寄存器会比使用直接访问输入、输出具有优越性。之所以这样有以下三个原因：

- 所有输入点的采样是在扫描周期的一开始同步进行的。在整个扫描周期的程序执行过程中输入值被冻结。而输出点按照映象寄存器中的值刷新是在程序执行完成之后。这样会使系统更加稳定。
- 访问映象寄存器的速度比直接访问 I/O 点要快，有利于程序快速运行。
- I/O 点是位实体，只能按位或者字节来访问，而你可以按位、字节、字或者双字的形式来访问映象寄存器。也就是说，使用映象寄存器更为灵活。

S7-200 允许在程序扫描周期中使用中断

如果你使用了中断，与中断事件相关的中断服务程序作为程序的一部分被保存。中断服务程序不会在正常的程序扫描周期中执行，它只有在中断事件发生时才被执行（有可能在扫描周期的任意一点）。

在中断优先级相同的情况下，S7-200 遵循先来先服务的原则来执行中断服务程序。有关中断的更多信息参见第 6 章。

S7-200 允许设定通讯任务的处理时间

你可以设定一个扫描周期的百分比用来处理与 RUN 模式编译或执行状态相关的通讯请求。当你增加用于处理通讯请求时间的百分比的同时，也增加了程序的扫描周期，使你的控制过程变慢。

用于处理通讯请求的时间百分比的缺省值为 10%。这个默认设置为在对控制过程影响最小的前提下处理编译和状态操作，提供了一个合理的时间。你可以在 5%到 50%之间调节这个值。要想设置背景通讯的扫描周期时间片，按以下步骤：

1. 在命令菜单中选择 **View>Component>System Block**，单击背景时间标签。
2. 改变通讯背景时间的值并单击 OK。
3. 将改变后的系统块下载到 S7-200 中。

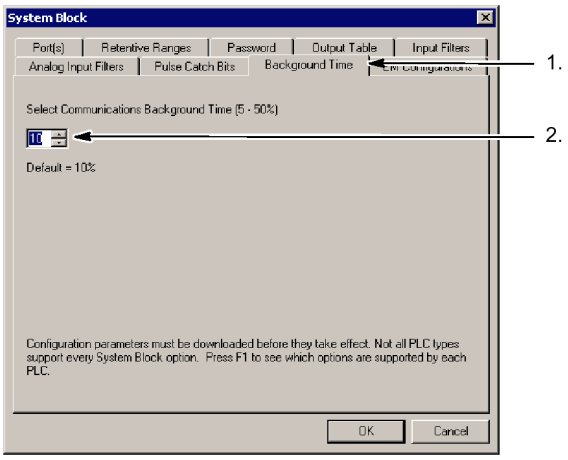


图 4-21 通讯背景时间

S7-200 允许你设置停止模式下的数字量输出状态

S7-200 的输出表允许你选择 STOP 模式下的输出状态，是将已知值传送到数字量输出点，还是使输出保持 STOP 模式之前的状态。输出表是系统块的一部分，它被下载并存储在 CPU 中，而且仅供数字量输出使用。

1．在命令菜单中选择 View>Component>System Block，单击输出表标签。

2．如果要冻结上一个状态的输出，选择 Freeze Outputs 复选框。

3．如果要将输出表中的值复制到输出点上，则要填写输出表。在你希望从运行到停止模式转换后置 1 的相应位置上点击。（输出表的缺省设置全部为 0。）

4．点击 OK 保存你的选择。

5．将改变后的系统块下载到 S7-200 中。

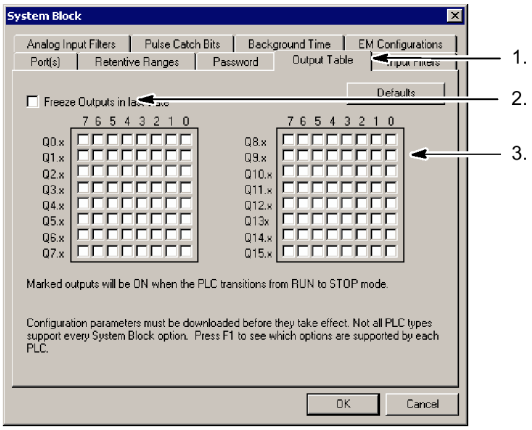


图 4-22 组态输出表

S7-200 允许你定义掉电保持存储区

如果你希望在掉电后仍然保持存储区中的数据，你可以定义最多六个掉电保持区的地址范围。在掉电保持区中你可以使用以下存储区的地址范围：V、M、C 和 T。只有保持型定时器（TONR）可以设为掉电保持的。M 存储区的前 14 个字节，缺省设置为不保持。

对于定时器和计数器来说，只有当前值可以保持，而定时器位和计数器位是不能保持的。

提示

如果将 MB0 到 MB13 设为掉电保持，则当 CPU 掉电时，它们会自动被保持到 EEPROM 中。

按照以下步骤设置掉电保持区：

1. 在命令菜单中选择 View>Component>System Block，单击掉电保持区标签。
2. 设置掉电保持区的范围并单击 OK。
3. 将改变后的系统块下载到 S7-200 中。

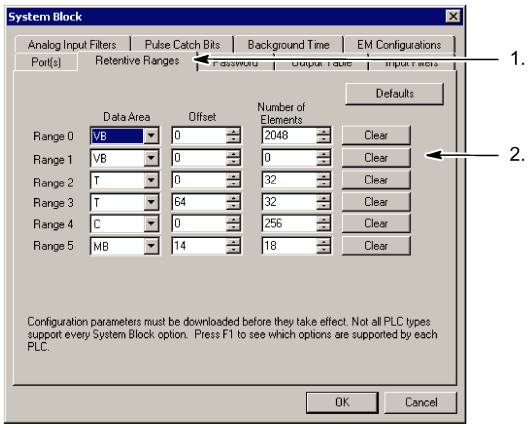


图 4-23 掉电保持存储区

S7-200 允许你对数字量输入加滤波器

S7-200 允许你为某些或者全部本机数字量输入点选择输入滤波器，并为滤波器定义延迟时间（从 0.2ms 到 12.8ms 可选）。这个延迟时间有助于滤除输入噪声，以免引入输入状态不可预测的变化。

输入滤波器是系统块的一部分，它被下载并存储在 CPU 中。滤波器延迟时间的缺省值为 6.4ms。如图 4-24 所示，一组输入点共用一个延迟时间。

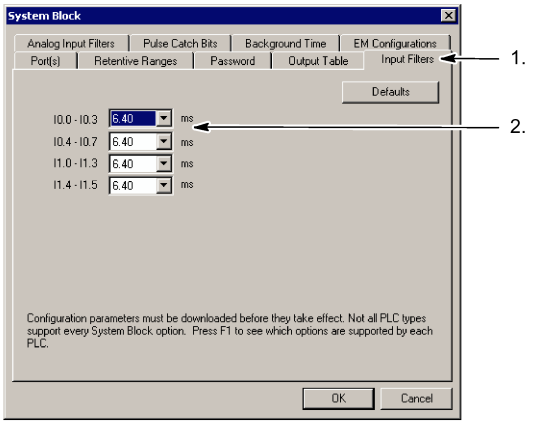


图 4-24 组态输入滤波器

按照以下步骤设置输入滤波器延迟时间：

1. 在命令菜单中选择 View>Component>System Block，单击输入滤波器标签。
2. 为每一组输入指定延迟时间。
3. 将改变后的系统块下载到 S7-200 中。

提示

数字量输入滤波器会对读输入指令、输入中断和脉冲捕捉产生影响。根据你的选择，应用程序有可能丢掉一个中断事件或者脉冲捕捉。高速计数器不受此影响。

S7-200 允许你对模拟量输入加滤波器

S7-200 允许你对每一路模拟量输入选择软件滤波器。滤波值是多个模拟量输入采样值的平均值。滤波器参数（采样次数和死区）对于允许滤波的所有模拟量输入是相同的。

滤波器具有快速响应的特点，可以反映信号的快速变化。当输入与平均值的差超过设定的变化时，滤波器对最近的模拟量输入值产生一个阶跃函数。这个差称为死区，并用模拟量输入的数字信号设定。

缺省配置是允许所有输入滤波。

1. 在命令菜单中选择 View> Component> System Block，单击模拟量输入滤波标签。
2. 选择需要滤波的模拟量输入、采样个数和死区。
3. 单击 OK。
4. 将改变后的系统块下载到 S7-200 中。

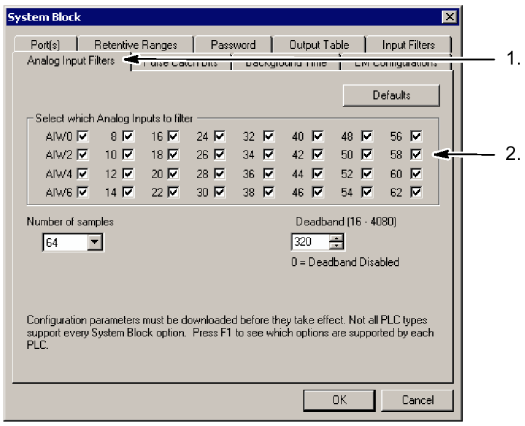


图 4-25 模拟量输入滤波

提示

不要对在模拟量字中传递数字信息或者报警指示的模块使用模拟量输入滤波。对于 RTD、TC 和 ASI 主站模块，不能使用模拟量输入滤波。

S7-200 允许你捕捉窄脉冲

S7-200 为每个本机数字量输入提供脉冲捕捉功能。脉冲捕捉功能允许 PLC 捕捉到持续时间很短的高电平脉冲或者低电平脉冲。而在扫描周期的开始，这些脉冲不是总能被 CPU 读到。当一个输入设置了脉冲捕捉功能时，输入端的状态变化被锁存并一直保持到下一个扫描循环刷新。这就确保了一个持续时间很短的脉冲被捕捉到并保持到 S7-200 读取输入点。

可以分别使能每一个本机数字量输入点的脉冲捕捉功能。按照以下步骤设置脉冲捕捉：

1. 在命令菜单中选择 View> Component>System Block，单击脉冲捕捉标签。
2. 点击相应的复选框并点击 OK。
3. 将改变后的系统块下载到 S7-200 中。

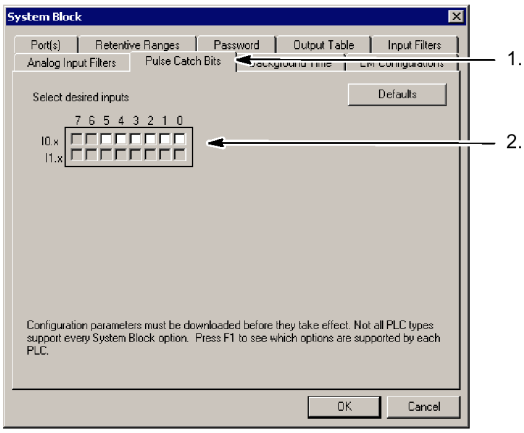


图 4-26 脉冲捕捉

带有和不带有脉冲捕捉功能的 S7-200 的基本操作如图 4-27 所示。

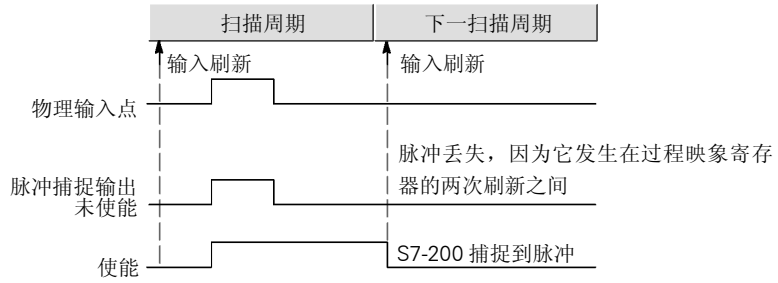


图 4-27 带有和不带有脉冲捕捉功能的 S7-200 操作

由于脉冲是在通过了输入滤波器之后，才能够被捕捉到，因而要调整输入滤波时间，确保脉冲不被滤掉。数字输入电路的方框图如图 4-28 所示。

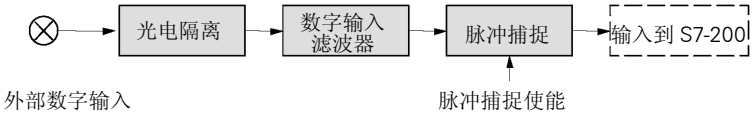


图 4-28 数字输入电路

对于不同的输入条件，脉冲捕捉功能的响应如图 4-29 所示。如果在给定的扫描周期中有不止一个脉冲，则只有第一个脉冲被读到。这种情况下，你应该使用上升/下降沿中断事件。（表 6-44 中给出了中断事件列表。）

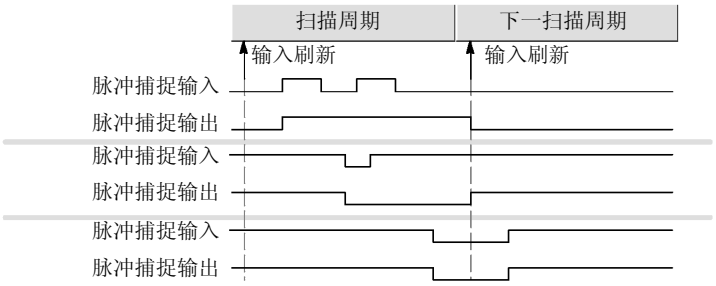


图 4-29 不同输入条件下的脉冲捕捉响应

S7-200 提供密码保护功能

所有的 S7-200 CPU 都提供了密码保护，来限制对某些特定功能的使用。

对 CPU 功能及存储器的访问权限是通过密码来实现的。不设定密码保护，对 S7-200 的访问没有限制。设置了密码保护，根据安装密码时的设置，CPU 禁止所有的受限操作。

密码不区分大小写。

如表 4-3 所示，S7-200 CPU 提供了限制 CPU 访问功能的三个等级。每个等级允许特定的无需密码的访问功能。S7-200 的缺省设置为等级 1（没有限制）。

在网络中输入密码不会对 CPU 的密码保护有所危害。

表 4-3 S7-200 的存取限制

CPU 功能	1 级	2 级	3 级
读写用户数据	不限制	不限制	不限制
启动、停止重启 CPU			
读写时钟			
上载程序、数据和配置	不限制	不限制	要密码
下载到 CPU		要密码	
监视执行状态			
强制数据式执行程序			
复制到存储卡			
在 STOP 模式下写输出			

CPU 只允许一个用户使用访问权限，禁止其它用户使用这些功能。同一时刻，只允许一个用户不受限制地访问 CPU。

提示

当输入密码后，在编程设备同 CPU 断开的一分钟之内，该授权等级仍然有效。在断开电缆之前，一定要先退出 STEP7-Micro/WIN，以避免其它用户利用编程设备访问 CPU。

为 S7-200 配置密码

如图 4-30 所示的系统块对话框允许你为 S7-200 配置密码。

1. 在命令菜单中选择 **View> Component>System Block**，单击密码标签。
2. 为 S7-200 选择合适的访问级别。
3. 输入并确认密码。
4. 单击 OK。
5. 将改变后的系统块下载到 S7-200 中。

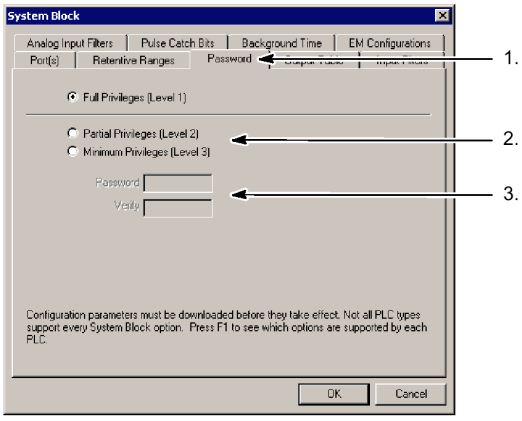


图 4-30 创建密码

密码忘记后如何恢复

如果你忘记了密码，你必须清除存储器，重新下载应用程序。清除存储器会使 S7-200 处于停止模式，并且将 S7-200 中，除了网络地址、波特率和时钟以外的其它参数恢复到出厂设置。按照以下步骤清除 S7-200 中的应用程序：

1. 在命令菜单中选择 PLC＞Clear 来显示清除对话框。
2. 选择所有的块并点击 OK 确认。
3. 如果配置了密码，STEP7-Micro/WIN 会显示密码授权对话框。要清除密码，在密码授权对话框中输入“CLEARPLC”，就可以继续执行全部清除的操作。（“CLEARPLC”不区分大小写。）全部清除操作不会去掉存储卡中的程序。由于密码和程序一同保存在存储卡中，因而必须重新写存储卡，才能从程序中去掉密码。

警告

清除 S7-200 存储器会导致输出关闭（对于模拟量输出来说，会被冻结在某个特定值）。

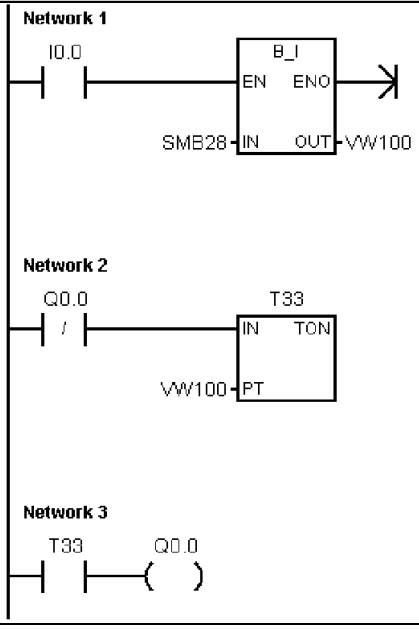
当清除 S7-200 存储器时，如果 S7-200 与其它设备相连，那么输出状态的变化会传递到该设备。如果输出的“安全状态”与厂家设定不同，输出的变化可能会引起设备产生不可预料的动作，也可能导致死亡或严重的人身伤害和设备损坏。在清除 S7-200 存储器之前，必须有合适的安全预防措施，确保你的操作是安全的。

S7-200 提供模拟电位器

模拟电位器位于模块前盖下面。你可以调节这些电位器来增加或者减小存于特殊存储器中的值（SMB）。这些只读值在程序中可用作很多功能，如更新定时器或计数器的当前值，输入或修改预置值、限定值等。可以用一个小螺丝刀来进行调节：将电位器顺时针（向右）旋转来使数值增大；逆时针（向左）旋转来使数值减小。

SMB28 中的数值代表模拟电位器 0 的位置。SMB29 中的数值代表模拟电位器 1 的位置。模拟电位器的标定范围为 0 到 255，重复度为±2。

用模拟电位器输入数值的例子程序




```
Network1  //读模拟电位器 0(SMB28)
          //将数值保存到 VW100 中。
LD        I0.0
BTI       SMB28，VW100
Network2  //用 VW100 中的整数值作为定时器的预置值。
LDN       Q0.0
TON       T33，VW100
Network3  //当 T33 达到预置值时接通 Q0.0。
LD        T33
=         Q0.0
```

S7-200 提供高速 I/O

高速计数器

S7-200 具有集成的高速计数功能，它能够对外部高速事件计数而不影响 S7-200 的性能。不同 CPU 支持的计数速率，参见附录 A。每个计数器都有专用的输入点作为时钟、方向控制、复位端、启动端等功能输入。在不同的模式下有不同的计数速率。关于高速计数器的详细信息参见第 6 章。

高速脉冲输出

S7-200 支持高速脉冲输出功能。

PTO 输出方波（占空比 50%），并可指定所输出的脉冲数量和周期。脉冲数可指定为 1 到 4,294,967,295。周期的单位可以是微秒（μs），也可以是毫秒（ms），设为 50 到 65,535 微秒或者 2 到 65,535 毫秒。脉冲序列输出（PTO）功能可以编程为产生一列脉冲或产生由多个脉冲序列组成的脉冲包络。在脉冲包络操作方式中，PTO 功能被编程为控制一个步进电机运行一个简单的斜坡上升、运行和斜坡下降操作序列或更复杂的操作序列。包络最多由 255 段组成，每一段对应一个斜坡上升或斜坡下降操作。

PWM 功能提供具有可变占空比的固定周期的输出脉冲。周期和脉宽既可以用微秒又可以用毫秒为单位。周期范围为 50 到 65,535 微秒或者 2 到 65,535 毫秒。脉宽可以从 0 到 65,535 微秒或者 0 到 65,535 毫秒。当脉宽等于周期时，占空比为 100%，输出恒定为 1；当脉宽等于 0 时，占空比为 0，输出恒定为 0。

关于高速脉冲输出的详细信息，参见第 6 章。

编程的概念、惯例及特点

5

S7-200 周而复始地执行应用程序，控制一个任务或过程。利用 STEP7-Micro/WIN 可以创建一个用户程序并将它下载到 S7-200 中。STEP7-Micro/WIN 软件中提供了多种工具和特性用于完成和调试应用程序。

本章内容：

设计一个微型 PLC 系统的指导原则

设计一个微型 PLC 系统有许多设计方法。以下这些通用的指导原则适用于许多设计项目。当然，你所在公司的规程和你在培训中接受的实践经验是必须遵循的。

分解控制过程或者机器

将你的控制过程或者机器分解成相互独立的部分。分解决定了控制器之间的界限，并将影响功能描述和资源的分配。

创建功能说明

写出过程或者机器每一部分的操作描述。它包括以下内容：I/O 点；操作的功能描述；每个执行机构（线圈、电机和驱动器等）在动作之前需要满足的状态；操作接口的描述以及过程或者机器与其它部分的接口。

安全电路的设计

识别要求设计硬件安全线路的设备。控制设备在不安全的条件下出现故障，会造成不可预料的启动或者机器操作的变化。在不可预料或者不正确的机器操作会造成人身伤害或严重的财产损失的场合，应该考虑采用独立于 S7-200 的机电冗余来防止不安全的操作。在安全电路的设计中应该考虑以下任务：

- 识别有可能不合适或者不可预料操作有可能会造成危害的执行机构。
- 识别确保操作不发生危害的条件，并决定如何独立于 CPU 来检测这些条件。
- 识别上电或断电时，CPU 和 I/O 对过程有何影响，识别错误何时被检测出来。这个信息只能用于常规的和可以预料的异常操作，不能用于保障安全的目的。
- 设计独立于 CPU 的手动或机电冗余来阻止危险的操作。
- 向 CPU 提供独立电路的状态信息，便于程序和操作员界面得到需要的信息。
- 识别其它与过程安全操作相关的安全要求。

指定操作员站

根据功能描述的要求建立操作员站的配置图。包括以下内容：

- 与过程或者机器有关的每个操作员站的位置总图。
- 操作员站的设备机械图（显示器、开关、指示灯等）
- 与 CPU 或扩展模块有关的电气图

创建配置图

根据功能描述的要求建立控制设备的配置图。包括如下内容：

- 和过程或者机器有关的每个 CPU 的位置图。
- CPU 和扩展 I/O 模块的机械布局图（包括控制柜和其它设备）。
- 每个 CPU 和扩展模块的电气图（包括设备型号、通讯地址和 I/O 地址）。

建立符号名表（可选）

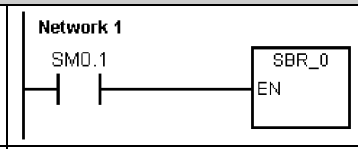
如果选择了符号名寻址，需要对绝对地址建立一个符号名表。符号名表不仅包括物理输入/输出信号，也包括程序中用到的其它元件。

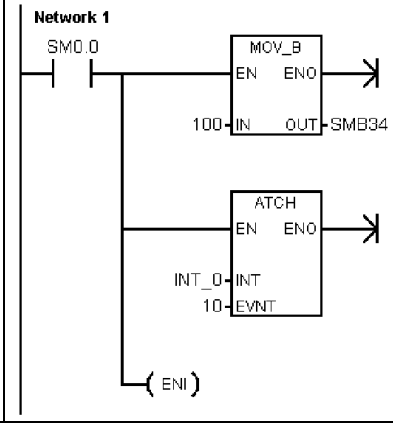
程序的基本组件

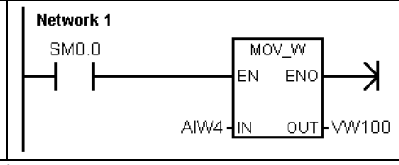
一个程序块由可执行代码和注释组成。可执行代码由主程序和若干子程序或者中断服务程序组成。可执行代码被编译并下载到 S7-200 中，而程序注释不会被下载。你可以使用这些组件（主程序、子程序和中断服务程序）来结构化你的控制程序。

以下例子程序包括一个子程序和一个中断服务程序。该例子程序使用一个定时中断，每 100ms 读一次模拟量输入值。

中断程序举例

MAIN		Netwrok 1 // 在第一个扫描周期， // 调用子程序。 LD SM0.1 CALL SBR_0

SBR0		Network 1 // 输入 100ms 时间间隔，
		//使能定时中断 0。
		LD SM0.0
		MOVB 100, SMB34
		ATCH INT_0, 10
		ENI

INT0		Network 1 // 采样 AIW 4。 LD SM0.0 MOVW AIW4,VW100

主程序

主程序中包括控制应用的指令。S7-200 在每一个扫描周期中顺序执行这些指令。主程序也被表示为 OB 1。

子程序

子程序是应用程序中的可选组件。只有被主程序、中断服务程序或者其它子程序调用时，子程序才会执行。当你希望重复执行某项功能时，子程序是非常有用的。与其在主程序中的不同位置多次使用相同的程序代码，不如将这段程序逻辑写在子程序中，然后在主程序中需要的地方调用。调用子程序有几个优点：

- 用子程序可以减小程序的长度
- 由于将代码从主程序中移出，因而用子程序可以缩短程序扫描周期。S7-200 在每个扫描周期中处理主程序中的代码，不管代码是否执行。而子程序只有在被调用时，S7-200 才会处理其代码。在不调用子程序时，S7-200 不会处理其代码。
- 用子程序创建的程序代码是可传递的。你可以在一个子程序中完成一个独立的功能，然后将它复制到另一个应用程序中而无需作重复工作。

提示

在子程序中使用 V 存储器地址会限制它的可移殖性。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其分配有冲突。相比之下，在子程序中的所有变量地址都使用局部变量（L 存储器），会使子程序有极高的可移殖性。因为当子程序使用局部变量时，子程序与程序的其它部分之间不会有地址冲突。

中断服务程序

中断服务程序是应用程序中的可选组件。当特定的中断事件发生时，中断服务程序执行。你可以为一个预先定义好的中断事件设计一个中断服务程序。当特定的事件发生时，S7-200 会执行中断服务程序。

中断服务程序不会被主程序调用。你将一个中断服务程序与一个中断事件相关联。只有每次中断事件发生时，S7-200 才会执行中断服务程序。

提示

因为无法预测何时会产生中断，所以应考虑尽量限制中断服务程序和程序中其它部分所使用的变量个数。

使用中断服务程序中的局部变量，可以保证中断服务程序只使用临时存储器，并且不会覆盖程序中其它部分使用的数据。为了保证主程序与中断服务程序正确地共享数据，你可以使用许多编程技巧。关于这些技巧的描述在第 6 章的中断指令部分有详细说明。

程序中的其它组件

其它块中也包含了 S7-200 的信息。当你下载程序时，你可以选择同时下载这些块。

系统块

系统块

系统块允许你为 S7-200 配置不同的硬件参数。

数据块

数据块

数据块存储应用程序中所使用的不同变量值。你可以用数据块输入数据的初始值。

用 STEP7-Micro/WIN 创建用户程序

要打开 STEP7-Micro/WIN，可以双击 STEP7-Micro/WIN 图标，也可以在命令菜单中选择 **Start>SIMATIC>STEP7-Micro WIN3.2**。如图 5-1 所示，STEP7-Micro/WIN 项目窗口为创建你的控制程序提供了一个便利的工作空间。

工具栏为常用菜单命令的快捷方式提供按钮。你可以显示或者隐藏任意工具栏。

操作栏为访问 STEP7-Micro/WIN 中不同的程序组件提供了一组图标。指令树显示了所有的项目对象和创建你的控制程序所需要的指令。你可以将指令从指令树中拖到你的应用程序中，也可用双击指令的方法将该指令插入到程序编辑器中的当前光标所在地。

程序编辑器中包括程序逻辑和局部变量表。你可以在局部变量表中为临时的局部变量定义符号名。在程序编辑器的底部有子程序和中断服务程序的标签。点击这些标签，你可以在主程序、子程序和中断服务程序之间切换。

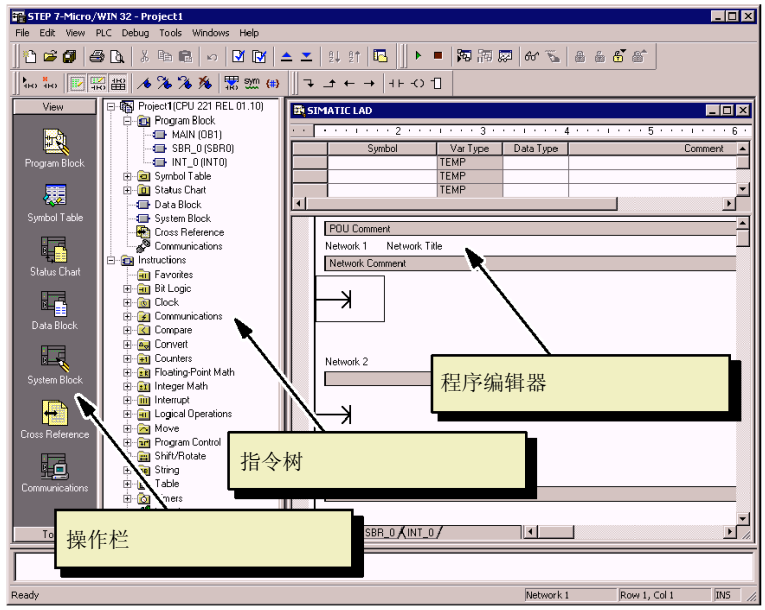


图 5-1　STEP7-Micro/WIN


程序编辑器

STEP7-Micro/WIN 提供三种编辑器来创建你的程序：梯形图（LAD）、语句表（STL）和功能块图（FBD）。用任何一种程序编辑器编写的程序，都可以用另外一种程序编辑器来浏览和编辑，但要遵循一些输入规则。

STL 编辑器的特点

STL 编辑器按照文本语言的形式显示程序。STL 编辑器允许你输入指令助记符来创建你的控制程序。语句表也允许你创建用 LAD 和 FBD 编辑器无法创建的程序。这是因为你在使用 S7-200 的本族语言进行编程，而在图形编辑器中，为了正确地画出图形，必须遵守一些规则。如图 5-2 所示，这些基于文本的概念与汇编语言编程非常相似。

S7-200 从上到下按照程序的次序执行每一条指令，然后回到程序的开始重新执行。

STL 使用一个逻辑堆栈来分析控制逻辑。你插入 STL 指令来处理堆栈操作。

```
LD      I0.0        // 读一个输入
A       I0.1        // 与另一个输入“与”
=       Q1.0        // 写一个输出值
```

图 5-2　STL 程序举例

当你选择 STL 编辑器时，考虑以下要点：

- STL 最适合于有经验的程序员。
- STL 有时让你能够解决用 LAD 或者 FBD 不容易解决的问题。
- 当你使用 STL 编辑器时，只能使用 SIMATIC 指令集。
- 虽然你可以用 STL 编辑器查看或者编辑用 LAD 或者 FBD 编辑器编写的程序，但是反之不一定成立。LAD 或者 FBD 编辑器不一定总能显示所有利用 STL 编辑器编写的程序。

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。梯形图程序允许程序仿真来自电源的电流通过一系列的逻辑输入条件，决定是否使能逻辑输出。一个 LAD 程序包括左侧提供能流的能量线。闭合的触点允许能流经过并到达下一个元素；打开的触点会阻塞能流。

逻辑控制是分段的，程序在同一时间执行一段，从左到右，从上到下。图 5-3 给出了 LAD 程序的一个例子。不同的指令用不同的图形符号表示。它包括三种基本形式。

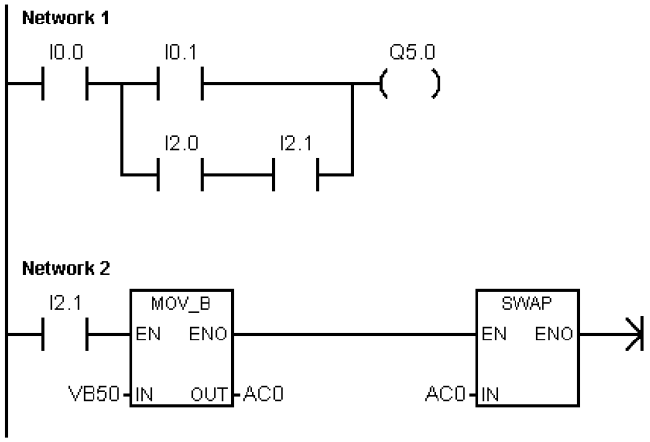


图 5-3　LAD 程序举例

触点代表逻辑输入条件，例如；开关、按钮或者内部条件等。

线圈通常表示逻辑输出结果，例如：灯负载、电机启动器、中间继电器或者内部输出条件。

盒表示其它一些指令，例如：定时器、计数器或者数学运算指令。

当你选择 LAD 编辑器时，考虑以下要点：

- 梯形图逻辑易于初学者使用。
- 图形表示法易于理解而且全世界通用。
- LAD 编辑器能够使用 SIMATIC 和 IEC 1131-3 指令集。
- 可以使用 STL 编辑器显示所有用 SIMATIC LAD 编辑器编写的程序。

FBD 编辑器的特点

FBD 编辑器以图形方式显示程序，由通用逻辑门图形组成。它没有梯形图编辑器中的触点和线圈，但有与之等价的指令，用盒指令表示。图 5-4 中给出了 FBD 程序的一个例子。

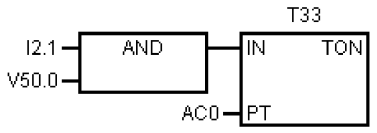


图 5-4　FBD 程序举例

FBD 不使用左右能量线，因此“能流”这个术语用于表示通过 FBD 逻辑块控制流这样一个类似的概念。

逻辑“1”通过 FBD 元素称为能流。能流的原始输入和最终的输出可以直接分配给操作数。

程序逻辑由这些盒指令之间的连接决定。也就是说，一条指令（例如 AND 盒）的输出可以用来允许另一条指令（例如定时器），这样可以建立所需要的控制逻辑。这样的连接概念使你可以解决各种各样的逻辑问题。

当你选择 FBD 编辑器时，考虑以下要点：

- 图形逻辑门的表示形式有利于程序流的跟踪。
- FBD 编辑器能够使用 SIMATIC 和 IEC 1131-3 指令集。
- 可以使用 STL 编辑器显示所有用 SIMATIC FBD 编辑器编写的程序。

SIMATIC 和 IEC 1131-3 指令集的选择

大部分 PLC 产品提供相似的基本指令，但是不同厂商的 PLC 产品在它们的表示和操作上常常有小的差别。近年来，国际电工委员会（IEC）推出了一个有关 PLC 编程各个方面的一个全球标准。这个标准鼓励不同的 PLC 厂商向用户提供与 IEC 指令集的表示和操作一致的指令。S7-200 提供两种指令集用于完成各种自动化任务。

IEC1131-3 标准编译 IEC 指令集，而 SIMATIC 指令集是 S7-200 专用的。

提示：

当在 STEP7-Micro/WIN 中选择 IEC 模式时，在指令树中不能够被 IEC1131-3 标准使用的指令旁边会显示一个红色的菱形（◆）。

在 SIMATIC 指令集和 IEC1131-3 指令集之间有一些主要区别。

- IEC 1131-3 指令集是不同 PLC 厂商的指令标准。SIMATIC 指令集中的一些指令并不是 IEC1131-3 规范中的标准指令。这些是仍在使用的非标准指令，但是如果使用它们，程序就不再严格的与 IEC1131-3 兼容。
- 一些指令可以接受多种数据格式，这个概念通常指多重功能。例如，数学指令盒中不区分 ADD_I（整数加法）和 ADD_R（实数加法），而是在加法指令中检查被加数的格式，并自动选择正确的 CPU 指令。这样可以节省宝贵的程序设计时间。
- 当使用 IEC1131-3 指令时，自动检查指令参数并选择合适的数据格式。数据格式检查不需要用户介入。例如，如果你给一个位操作指令输入一个整数值，就会出现一个错误。这样，可以有助于减少编程的语法错误。

在选择 SIMATIC 或 IEC 指令集时，应考虑以下因素：

- SIMATIC 指令通常执行时间最短。一些 IEC 指令的执行时间较长。
- 一些 IEC 指令与 SIMATIC 指令操作数不同，例如定时器指令、计数器指令、乘法指令和除法指令等。
- 你可以在全部的三种程序编辑器（LAD、STL、FBD）中使用 SIMATIC 指令集，但只能在 LAD 和 FBD 编辑器中使用 IEC 指令。
- 对于不同品牌的 PLC，IEC 指令的操作是标准的，因而创建 IEC 程序的知识与 PLC 操作平台无关。
- 因为 IEC 标准中定义的指令少于 SIMATIC 指令集，因而可以用 SIMATIC 指令完成更多功能。
- IEC1131-3 规定变量必须使用类型声明，而且支持系统数据类型检查。

理解程序编辑器中使用的惯例

STEP7-Micro/WIN 在所有程序编辑器中使用以下惯例：

- 在符号名前加#（#Var1）表示该符号为局部变量。
- 在 IEC 指令中%表示直接地址。
- 操作数符号“??”或者“????”表示需要配置操作数。

LAD 程序被分为程序段。一个程序段是按照顺序安排的以一个完整电路的形式连接在一起的触点、线圈和盒，不能短路或者开路，也不能有能流倒流的现象存在。STEP7-Micro/WIN 允许你为 LAD 程序中的每一个程序段加注释。FBD 编程同样使用程序段的概念和允许注释程序。

STL 程序不用分段，但是你可以用关键词 NETWORK 将程序分段。

LAD 编辑器中使用的惯例

在 LAD 编辑器中，你可以使用 F4、F6 和 F9 来快速输入触点、盒和线圈指令。LAD 编辑器使用以下惯例：

- 符号“——>>”表示开路或者需要能流连接。

- 符号“➔”表示指令输出能流，可以级连或串联。
- 符号“>>”表示你可以使用能流。

FBD 编辑器中使用的惯例

在 FBD 编辑器中，你可以使用 F4、F6 和 F9 来快速输入 AND、OR 和盒指令。FBD 编辑器使用以下惯例：

- 在 EN 操作数上的符号“——>>”表示能流或者操作数指示器。它也可用于表示开路或者需要能流连接。
- 符号“➔”表示指令输出能流，可以级连或串联。
- 符号“<<”和“>>”表示你可以使用数值或能流。
- 反向圈：操作数或者能流的负逻辑或者反向输入表示为在输入端加一个小圆圈。在图 5-5 中 Q0.0 是 I0.0 的非和 I0.1 与的结果。反向圈仅用于能够作为参数或者能流的布尔信号。

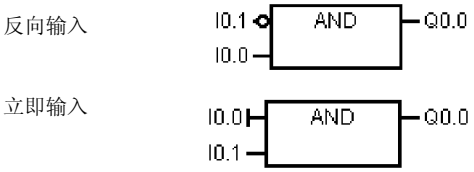


图 5-5 FBD 惯例

- 立即输入：如图 5-5 中所示，在 FBD 编辑器中，用在 FBD 指令输入端加一条垂直线的方法来表示布尔操作数的立即输入。立即输入直接从物理输入点上读取数据。立即操作数只能用物理输入点。
- 没有输入或者输出的盒：一个盒没有输入意味着这条指令与能流无关。

提示：

AND 和 OR 指令的操作数的个数可以扩展到最多 32 个。要增加或者减少操作数的个数，用键盘上的“＋”或者“－”。

S7-200 编程的通用惯例

EN/ENO 的定义

EN（使能输入）是 LAD 和 FBD 中盒的布尔输入。要使盒指令执行，必须使能流到达这个输入。在 STL 中，指令没有 EN 输入，但是要想使 STL 指令执行，堆栈顶部的逻辑值必须是“1”。

ENO（使能输出）是 LAD 和 FBD 中盒的布尔输出。如果盒的 EN 输入有能流并且指令正确执行，则 ENO 输出会将能流传递给下一元素。如果指令的执行出错，则能流在出错的盒指令处被中断。

在 STL 中没有使能输出，但是 STL 指令象相关的有 ENO 输出的 LAD 和 FBD 指令一样，置位一个特殊的 ENO 位。这个位可以由 AND ENO（AENO）指令访问，并且能与盒指令的 ENO 产生同样的影响。

提示

EN/ENO 操作数的数据类型并没有在每条指令中的操作数表中给以说明，因为这一操作数在所有 LAD 和 FBD 指令中都是一样的。表 5-1 列出了这些 LAD 和 FBD 中的操作数和数据类型。这些操作数对本手册中介绍的所有 LAD 和 FBD 指令均适用。

表 5-1 LAD 和 FBD 中 EN/ENO 操作数和数据类型

程序编辑器	输入/输出	操作数	数据类型
LAD	EN、ENO	能流	BOOL
FBD	EN、ENO	I、Q、V、M、SM、S、T、C、L	BOOL

条件输入/无条件输入

在 LAD 和 FBD 中，如果一个盒或者线圈的左侧没有任何元素，则它与能流有关。如果一个盒或者线圈的左侧直接连接到能量线上，则它与能流无关。表 5-2 中给出了条件输入和无条件输入的例子。

表 5-2 条件输入和无条件输入的表示方法

能流	LAD	FBD
与能流有关的指令（条件输入）	1 —(JMP)	JMP
与能流无关的指令（无条件输入）	—(NEXT)	NEXT

没有输出的指令

无法级连的盒指令被表示为没有布尔输出。这些包括子程序调用、跳转和条件返回指令。梯形线圈也只能放在能量线之后。这些指令包括标签、装载 SCR、SCR 条件结束和 SCR 结束指令。这在 FBD 中用无输出的盒指令作为与其它指令的区别。

比较指令

无论是否有能流，比较指令都会被执行。如果无能流则输出 0。如果有能流，输出值取决于比较结果。在 STMATIC FBD、IEC 梯形图和 IEC FBD 中，比较指令都用盒表示，尽管它实现的是触点操作。

使用向导帮你创建控制程序

STEP7-Micro/WIN 提供向导使你的编程变得更自动更容易。在第 6 章中，具有相关向导的指令会有一个指令向导图标。

指令向导

S7-200 中的出错处理

S7-200 将错误分为致命错误和非致命错误。你可以在命令菜单中选择 PLC>Information 来得到所产生错误的错误代码。图 5-6 中显示 PLC 信息对话框，其中包括错误代码和错误描述。

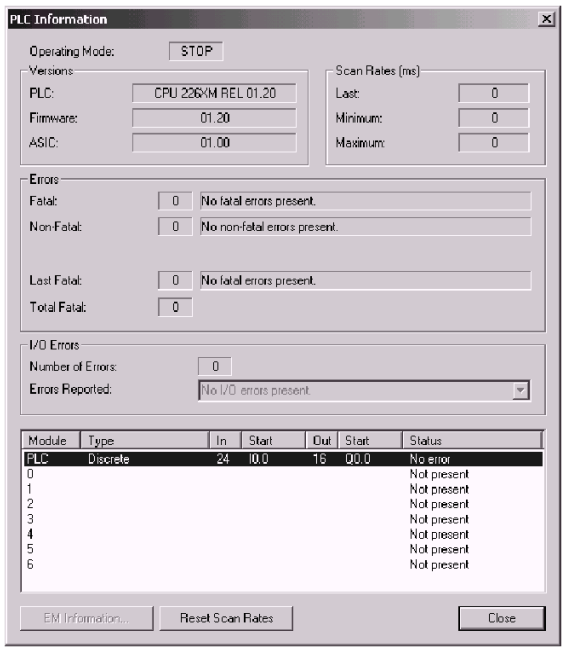


图 5-6　PLC 信息对话框

Last Fatal 区显示 S7-200 发生的前一致命错误代码。如果 RAM 区是掉电保持的，该代码也会保持。当 S7-200 全清或者 RAM 区掉电保持失败时，该区也被清除。

Total Fatal 区是前一次 CPU 清除所有存储区后产生致命错误的次数。如果 RAM 区是掉电保持的，这个次数也会保持。当 S7-200 全清或者 RAM 区掉电保持失败时，该区也被清除。

附录 C 中列出了 S7-200 的错误代码。附录 D 中描述了可用于监视错误的特殊存储器（SM）标志位。

非致命错误

非致命错误是指用户程序结构问题、用户程序指令执行问题和扩展 I/O 模块问题。你可以用 STEP7-Micro/WIN 来得到所产生错误的错误代码。

程序编译错误

当下载程序时，S7-200 会编译程序。如果 S7-200 发现程序违反了编译规则，会停止下载并产生一个错误代码。（已经下载到 S7-200 中的程序会存储在 EEPROM 中，不会丢失。）可以在修正错误后再次下载程序。编译错误列表参见附录 C。

I/O 错误

启动时，S7-200 从每个模块读取 I/O 配置。正常运行过程中，S7-200 周期性的检测每个模块的状态与启动时得到的配置相比较。如果 S7-200 检测到差别，它会将模块错误寄存器中的配置错误标志位置位。在模块配置再次匹配之前，S7-200 不再读写模块的输入输出数据。

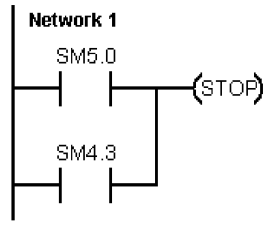
模块的启动信息存储在特殊存储器（SM）标志位中。应用程序可以监视这些标志位。有关用于 I/O 错误报告的 SM 标志的更多信息参见附录 D。

程序执行错误

在程序执行过程中有可能产生错误。这类错误有可能来自使用了不正确的指令或者在过程中产生了非法数据。例如：一个编译正确的间接寻址指针，在程序执行过程中，可能会改为指向一个非法地址。这是一个实时程序问题的例子。当实时程序问题发生时，SM4.3 会在 CPU 处于 RUN 模式期间置位。（实时程序问题的列表参见附录 C）。程序执行错误信息存储在特殊寄存器（SM）标志位中。应用程序可以监视这些标志位。有关用于程序执行错误报告的 SM 标志的更多信息参见附录 D。

当 S7-200 发生非致命错误时，S7-200 并不切换到 STOP 模式。它仅仅是把事件记录到 SM 存储器中并继续执行应用程序。但是如果用户希望在发生非致命错误时，将 CPU 切换到 STOP 模式，可以通过编程实现。下列例子程序用于监视两个非致命错误标志位。当两个标志中任意一个置位，S7-200 将切换到 STOP 模式。

非致命错误检测程序举例




```
Network1    //当有 I/O 错误或实时运行程序错误发生时，将 CPU
            //切换到 STOP 模式。
LD          SM5.0
O           SM4.3
STOP
```

致命错误

致命错误会导致 S7-200 停止程序执行。按照致命错误的严重程度，S7-200 使其部分或全部功能无法执行。处理致命错误的目的是把 CPU 引向安全状态，CPU 可以对存在的错误条件作出响应。当 CPU 检测到一个致命错误时，CPU 会切换到 STOP 模式，点亮系统错误 LED 和 STOP（停止）LED 指示灯，并忽略输出表，关闭所有输出。S7-200 会一直保持这种状态，直到消除致命错误条件。

一旦消除了致命错误条件，必须重新启动 CPU。可以用以下方法重新启动 CPU：

- 重新启动电源
- 将模式开关由 RUN 或者 TERM 变为 STOP
- 在 STEP7-Micro/WIN 命令菜单中选择 **PLC>Power-Up Reset**，可以强制 CPU 启动并清除所有致命错误。

重启 CPU 会清除致命错误，并执行上电诊断测试来确认已改正错误。如果发现其它致命错误，CPU 会重新点亮错误 LED 指示灯，表示仍存在错误。否则 CPU 会开始正常工作。

有些错误可能会使 CPU 无法进行通讯。这种情况下你无法看到来自 CPU 的错误代码。这种错误表示硬件故障，CPU 模块需要修理，而修改程序或清除 CPU 内存是无法清除这些错误的。

在数据块中指定地址和初始值


数据块

数据块编辑器只用于为 V 存储器（变量存储器）指定初始值。你可以以字节、字或者双字的形式来分配 V 存储器，也可以选择输入注释。

数据块编辑器是一个自由格式的文本编辑器，也就是说，没有特定的区域被定义用于特定类型的信息。当你完成一行的输入并按回车键确认后，数据块编辑器将该行格式化（将地址、数据和注释分别列对齐，V 存储器地址大写）并重新显示。数据块编辑器根据你所定义变量的地址和长度（字节、字或者双字）为 V 存储器分配空间。

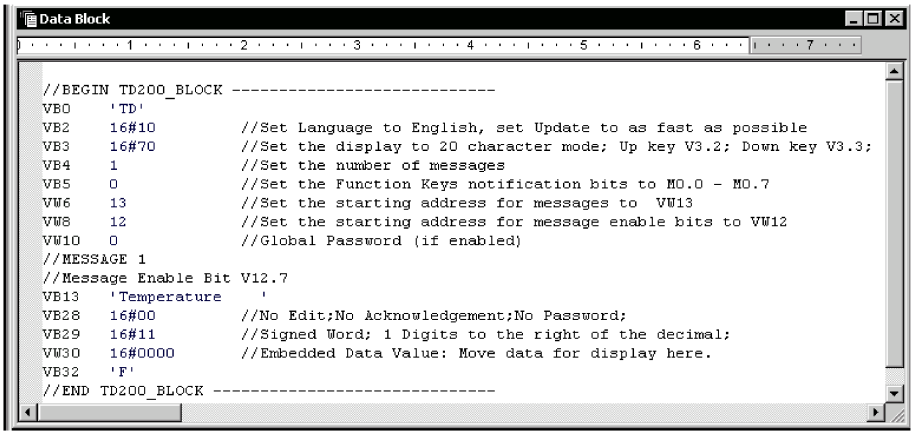


图 5-7　数据块编辑器

数据块的第一行必须有一个明确的地址分配。接下来的行中可以是明确的地址，也可以使用隐含地址。隐含地址是由编辑器分配的。当你在一个地址后面输入多个数据或者在一行中只输入数据时，你使用的是隐含地址。

数据块编辑器接受大小写字母，并且用逗号、制表符或者空格作为地址与数据之间的分隔符。

用符号表来定义变量的符号地址


符号表

符号表允许你定义和编辑符号名，使你能在程序中用符号地址访问变量。你可以创建多个符号表。你也可以在程序中使用系统定义的符号表。符号表也被称作全局变量表。

你可以使用绝对地址或者符号地址来输入指令操作数。绝对地址用存储区加上位或字节地址来标识地址。符号地址则用一串字母组合来标识地址。

			Symbol	Address	Comment
1			AlwaysOn	SM0.0	Always on contact
2			Pump1	Q2.3	Pump 1 on/off
3			Pump1Limit	I1.1	Pump 1 pressure limit switch
4			Pump1Pressure	VD100	Pump 1 current pressure (real)
5			Pump1Rpm	VW200	Pump1 PRMs (integer)
6					

图 5-8 符号表

在 SIMATIC 程序中，你可以使用符号表中定义的全局符号。在 IEC 程序中，你可以使用全局变量表中定义的全局符号。

为地址定义符号按如下步骤：

1. 在操作栏中单击符号表图标打开符号表。
2. 在 Symbol Name 列中输入一个符号名（例如：Input1），符号名的最大长度为 23 个字符。
3. 在 Address 列中输入地址（例如：I0.0）。
4. 对于 IEC 的全局变量表，在数据类型列中输入一个值或者从列表中选择一个。

你可以创建多个符号表，但无论是在同一个符号表中还是在不同的符号表中，都不能多次使用同一个字符串作为全局符号。

使用局部变量

你可以使用程序编辑器中的局部变量表来为子程序和中断服务程序分别指定变量，如图 5-9 所示。

SIMATIC LAD

	Name	Var Type	Data Type	Comment
	EN	IN	BOOL	
L0.0	FirstPass	IN	BOOL	First pass flag
LB1	Addr	IN	BYTE	Address of slave device
LW2	Data	IN	INT	Data to write to slave
LB4	Status	IN_OUT	BYTE	Status of write
L5.0	Done	OUT	BOOL	Done flag
LW6	Error	OUT	WORD	Error number (if any)

MAIN | SBR_0 | INT_0

图 5-9 局部变量表

局部变量可用于子程序传递参数，它增强了子程序的可移殖性和再利用性。

用状态图来监视用户程序


状态图

状态图允许你在控制程序运行的过程中对过程变量的值进行监视和修改。你可以跟踪程序的输入、输出或者变量，显示它们的当前值。状态图也允许你强制或者改变过程变量的值。

为了监控应用程序中不同部分的元素，你可以创建多个状态图。

在命令菜单中选择 **View>Component>Status Chart** 或者在操作过程中单击 Status Chart 图标来访问状态图。

当创建状态图时，你应该输入要监控的过程变量的地址。你无法监视常数、累加器和局部变量的状态。你可以按位或者字两种形式来显示定时器和计数器的值。以位形式显示的是定时器和计数器的状态位，而以字形式则显示定时器和计数器的当前值。

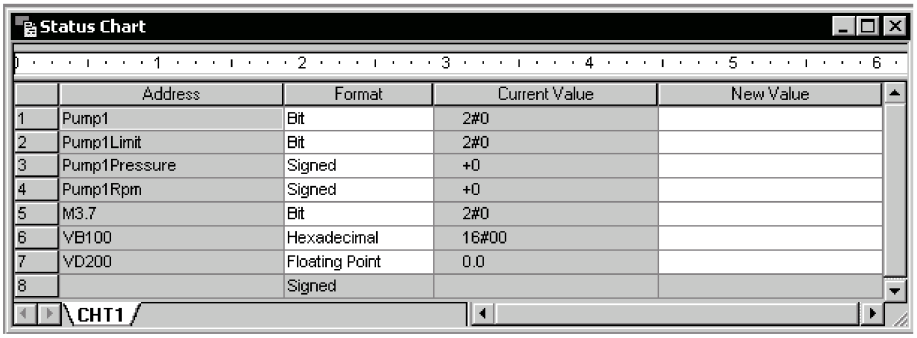


图 5-10 状态图

建立一个状态图来监视变量，按以下步骤：

1. 在地址区输入需要的地址。
2. 在格式列中选择数据类型。
3. 在命令菜单中选择 **Debug>Chart Status** 来监视 S7-200 中过程变量的状态。
4. 要连续采样数值或者单次读取状态，可以点击工具栏中相应的按钮。状态图也允许你强制或者修改过程变量的值。

在命令菜单中选择 **Edit>Insert>Row** 可以在状态图中插入一行。

提示：

你可以按逻辑分组为变量创建多个状态图，使每个状态图更短，便于分别监视。

创建一个指令库

STEP7-Micro/WIN 允许你创建自己的指令库，也允许你使用其它人已建好的库。见图 5-11。

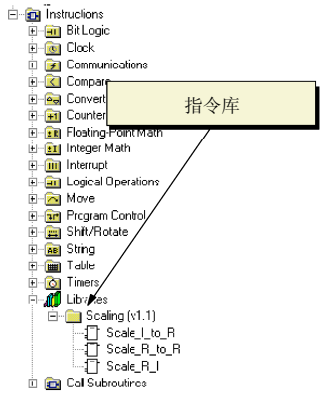


图 5-11 指令库

创建指令库的同时，你也在 STEP7-Micro/WIN 中建立了子程序和中断服务程序。为了防止意外地改动或者保护作者的技术专利，你可以隐藏这些程序代码。

要创建一个指令库，你需要完成以下任务：

1. 创建一个标准的 STEP7-Micro/WIN 项目，并且把指令库中包括的功能写入一个子程序或者中断服务程序中。
2. 确保在子程序或中断服务程序中使用的 V 存储器都定义了符号名。使用连续的 V 存储区来最小化其要求的数量。
3. 将子程序或者中断服务程序更名为你希望在指令库中显示的名称。
4. 在命令菜单中选择 File>Create Library 来编译新的指令库。

在 STEP7-Micro/WIN 的在线帮助中，你可以得到有关指令库的更多信息。

用下列步骤访问指令库中的指令：

1. 在命令菜单中选择 File>Add Libraries 来在指令树中增加一个库。
2. 选择特定的指令插入你的程序。（就象使用任何标准指令一样。）如果库程序要求 V 存储器，STEP7-Micro/WIN 会提示你分配一个存储器块。用库存储器对话框来分配存储器块。

应用程序的调试

STEP7-Micro/WIN 具备以下特点帮助你调试应用程序：

- 书签允许你在很长的程序中方便地来回移动。
- 交叉参考表允许你检查程序的使用参考信息。
- RUN 模式下编辑允许你在小规模修改程序的过程中，对过程控制产生最小的影响。当在 RUN 模式下编辑程序时，你同样可以下载程序块。

关于调试应用程序的更多信息参见第 8 章。

S7-200 指令集

6

本章描述用于对 S7-200 微型 PLC 编程的 SIMATIC 和 IEC1131 指令集。

本章内容：

用于描述指令的习惯用语

图 6-1 给出了对一条指令的典型描述，并指出了用于描述指令及其操作的不同区域。指令说明包括 LAD、FBD 和 STL 三种格式。操作数表列出了指令的操作数，并给出有效数据类型、存储器区和每个操作数的大小。

EN/ENO 操作数和数据类型没有在指令操作数表中列出，因为这些操作数对于所有 LAD 和 FBD 指令来说都是一样的。

- *对于 LAD：* EN 和 ENO 是能流，为布尔数据类型。
- *对于 FBD：* EN 和 ENO 是 I、Q、V、M、SM、S、T、C、L 或者能流，为布尔数据类型。

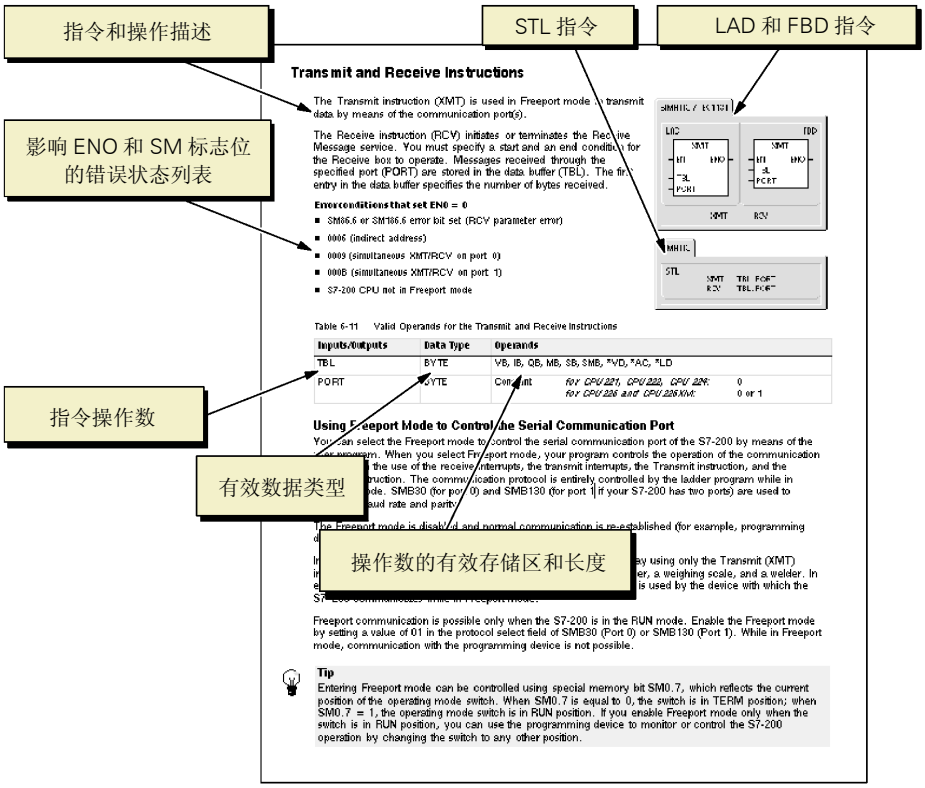


图 6-1 指令描述

S7-200 存储器范围及特性

表 6-1 S7-200 CPU 存储器范围及特性

描述	CPU221	CPU222	CPU224	CPU226	CPU226XM
用户程序大小	2K 字	2K 字	4K 字	4K 字	8K 字
用户数据大小	1K 字	1K 字	2.5K 字	2.5K 字	5K 字
输入映像寄存器	I0.0-I15.7	I0.0-I15.7	I0.0-I15.7	I0.0-I15.7	I0.0-I15.7
输出映像寄存器	Q0.0-Q15.7	Q0.0-Q15.7	Q0.0-Q15.7	Q0.0-Q15.7	Q0.0-Q15.7
模拟量输入（只读）	--	AIW0-AIW30	AIW0-AIW62	AIW0-AIW62	AIW0-AIW62
模拟量输出（只写）	--	AQW0-AQW30	AQW0-AQW62	AQW0-AQW62	AQW0-AQW62
变量存储器（V）	VB0-VB2047	VB0-VB2047	VB0-VB5119	VB0-VB5119	VB0-VB10239
局部存储器（L）[1]	LB0-LB63	LB0-LB63	LB0-LB63	LB0-LB63	LB0-LB63
位存储器（M）	M0.0-M31.7	M0.0-M31.7	M0.0-M31.7	M0.0-M31.7	M0.0-M31.7
特殊存储器（SM）	SM0.0-SM179.7	SM0.0-SM299.7	SM0.0-SM549.7	SM0.0-SM549.7	SM0.0-SM549.7
只读	SM0.0-SM29.7	SM0.0-SM29.7	SM0.0-SM29.7	SM0.0-SM29.7	SM0.0-SM29.7
定时器	256(T0-T255)	256(T0-T255)	256(T0-T255)	256(T0-T255)	256(T0-T255)
有记忆接通延迟 1ms	T0,T64	T0,T64	T0,T64	T0,T64	T0,T64
有记忆接通延迟 10ms	T1-T4,T65-T68	T1-T4,T65-T68	T1-T4,T65-T68	T1-T4,T65-T68	T1-T4,T65-T68
有记忆接通延迟 100ms	T5-T31, T69-T95	T5-T31, T69-T95	T5-T31, T69-T95	T5-T31, T69-T95	T5-T31, T69-T95
接通/关断 延迟 1ms	T32,T96	T32,T96	T32,T96	T32,T96	T32,T96
接通/关断 延迟 10ms	T33-T36, T97-T100	T33-T36, T97-T100	T33-T36, T97-T100	T33-T36, T97-T100	T33-T36, T97-T100
接通/关断 延迟 100ms	T37-T63, T101-T255	T37-T63, T101-T255	T37-T63, T101-T255	T37-T63, T101-T255	T37-T63, T101-T255
计数器	C0-C255	C0-C255	C0-C255	C0-C255	C0-C255
高速计数器	HC0,HC3,HC4, HC5	HC0,HC3,HC4, HC5	HC0-HC5	HC0-HC5	HC0-HC5
顺序控制继电器（S）	S0.0-S31.7	S0.0-S31.7	S0.0-S31.7	S0.0-S31.7	S0.0-S31.7
累加寄存器	AC0-AC3	AC0-AC3	AC0-AC3	AC0-AC3	AC0-AC3
跳转/标号	0-255	0-255	0-255	0-255	0-255
调用/子程序	0-63	0-63	0-63	0-63	0-127
中断程序	0-127	0-127	0-127	0-127	0-127
正/负跳变	256	256	256	256	256
PID 回路	0-7	0-7	0-7	0-7	0-7
端口	端口 0	端口 0	端口 0	端口 0，1	端口 0，1

[1] LB60~LB63 为 STEP 7－Micro/WIN32 的 3.0 版本或以后的版本软件保留

表 6-2 S7-200 CPU 操作数范围

存取方式		CPU 221	CPU 222	CPU 224，CPU 226	CPU 226XM
位存取（字节. 位）	I	0.0-15.7	0.0-15.7	0.0-15.7	0.0-15.7
	Q	0.0-15.7	0.0-15.7	0.0-15.7	0.0-15.7
	V	0.0-2047.7	0.0-2047.7	0.0-5119.7	0.0-10239.7
	M	0.0-31.7	0.0-31.7	0.0-31.7	0.0-31.7
	SM	0.0-179.7	0.0-299.7	0.0-549.7	0.0-549.7
	S	0.0-31.7	0.0-31.7	0.0-31.7	0.0-31.7
	T	0-255	0-255	0-255	0-255
	C	0-255	0-255	0-255	0-255
	L	0.0-59.7	0.0-59.7	0.0-59.7	0.0-59.7
字节存取	IB	0-15	0-15	0-15	0-15
	QB	0-15	0-15	0-15	0-15
	VB	0-2047	0-2047	0-5119	0-10239
	MB	0-31	0-31	0-31	0-31
	SMB	0-179	0-299	0-549	0-549
	SB	0-31	0-31	0-31	0-31
	L	0-63	0-63	0-63	0-255
	AC	0-3	0-3	0-3	0-255
字存取	IW	0-14	0-14	0-14	0-14
	QW	0-14	0-14	0-14	0-14
	VW	0-2046	0-2046	0-5118	0-10238
	MW	0-30	0-30	0-30	0-30
	SMW	0-178	0-298	0-548	0-548
	SW	0-30	0-30	0-30	0-30
	T	0-255	0-255	0-255	0-255
	C	0-255	0-255	0-255	0-255
	LW	0-58	0-58	0-58	0-58
	AC	0-3	0-3	0-3	0-3
	AIW	无	0-30	0-62	0-62
	AQW	无	0-30	0-62	0-62
双字存取	ID	0-12	0-12	0-12	0-12
	QD	0-12	0-12	0-12	0-12
	VD	0-2044	0-2044	0-5116	0-10236
	MD	0-28	0-28	0-28	0-28
	SMD	0-176	0-296	0-546	0-546
	SD	0-28	0-28	0-28	0-28
	LD	0-56	0-56	0-56	0-56
	AC	0-3	0-3	0-3	0-3
	HC	0, 3, 4, 5	0, 3, 4, 5	0-5	0-5

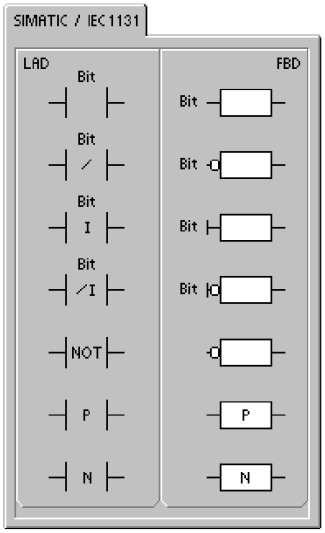
位逻辑指令

触点

标准触点

常开触点指令（LD、A 和 O）与常闭触点指令（LDN、AN 和 ON）从存储器或者过程映象寄存器中得到参考值。标准触点指令从存储器中得到参考值。（如果数据类型是 I 或 Q，则从过程映象寄存器中得到参考值。）

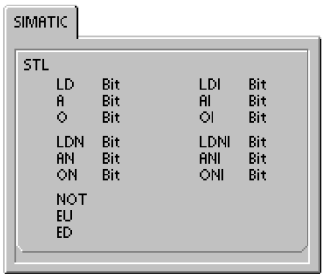
当位值为 1 时，常开触点闭合；当位值为 0 时，常闭触点闭合。在 FBD 中，与和或操作的输入可以最多扩展到 32 个。在 STL 中，常开指令 LD、AND 和 OR 将相应地址位的位值存入栈顶；而常闭指令 LDN、AN 和 ON，则将相应地址位的位值取反，再存入栈顶。



立即触点

立即触点并不依赖于 S7-200 的扫描周期刷新，它会立即刷新。在程序执行过程中，常开立即触点指令（LDI、AI 和 OI）与常闭立即触点指令（LDNI、ANI 和 ONI）得到物理输入值，但过程映象寄存器并不刷新。

当物理输入点状态为 1 时，常开立即触点闭合；当物理输入点状态为 0 时，常闭立即触点闭合。常开立即触点指令 LDI、AI 和 OI 将相应物理输入值存入栈顶；而常闭立即触点指令 LDNI、ANI 和 ONI 则将相应物理输入值取反，再存入栈顶。



取反指令

取反指令（NOT）改变能流输入的状态（也就是说，它将栈顶值由 0 变为 1，由 1 变为 0）。

正、负跳变指令

正跳变触点指令（EU）检测到每一次正跳变（由 0 到 1），让能流接通一个扫描周期。负跳变触点指令（ED）检测到每一次负跳变（由 1 到 0），让能流接通一个扫描周期。对于正跳变指令，一旦发现有正跳变发生（由 0 到 1），该栈顶值被置为 1，否则置 0。对于负跳变指令，一旦发现有负跳变发生（由 1 到 0），该栈顶值被置为 1，否则置 0。

对于实时编辑（在 RUN 模式下编辑应用程序），你必须为正、负跳变指令输入一个参数。关于在 RUN 模式下编辑程序的更多信息参见第 5 章。

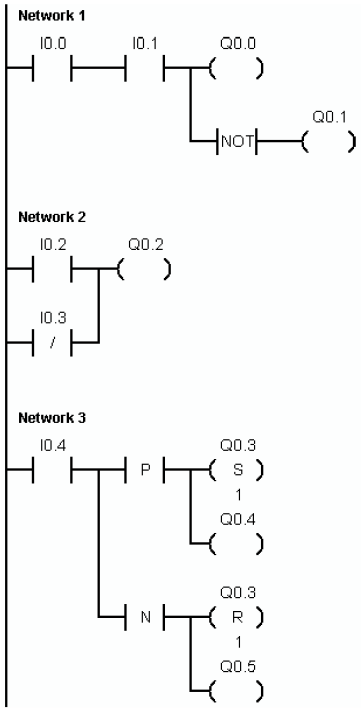
表 6-3 位逻辑输入指令的有效操作数

输入/输出	数据类型	操作数
位	BOOL	I、Q、V、M、SM、S、T、C、L、能流
位（立即）	BOOL	I

提示

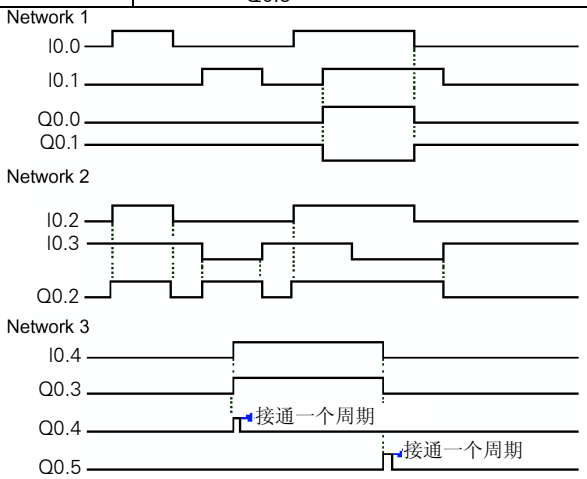
由于正跳变指令和负跳变指令要求由 1 到 0 或者由 0 到 1 的变化，你不能在第一个扫描周期中检测到上升沿或者下降沿的变化。在第一个扫描周期，S7-200 利用这些指令储存指定位的状态。在接下来的扫描周期中，这些指令能够检测到指定位的变化。

触点指令举例




```
Network1  //要想激活 Q0.0，常开触点 I0.0
          //和 I0.1 必须为 on（闭合）。
          //在 RUN 模式下，Q0.0 和 Q0.1 具有相反的逻辑状态。
LD        I0.0
A         I0.1
=         Q0.0
NOT
=         Q0.1
Network2  //要想激活 Q0.2，需要常开触点 I0.2 为 on 或者常
          //闭触点为 off。要激活输出，LAD 的并行分支中应
          //有一个或多个逻辑值为真。
LD        I0.2
ON        I0.3
=         Q0.2
Network3  //在 P 触点的一个上升沿或者在 N 触点的一个下降
          //沿出现时，一个扫描周期内输出一个脉冲。
          //在 RUN 模式，Q0.4 和 Q0.5 的脉冲状态变化太快
          //以至于在程序中无法用状态图监视。
          //Set 和 Reset 指令将 Q0.3 的状态变化锁存，
          //使程序可以监视。
LD        I0.4
LPS
EU
S         Q0.3，1
=         Q0.4
LPP
ED
R         Q0.3，1
=         Q0.5
```

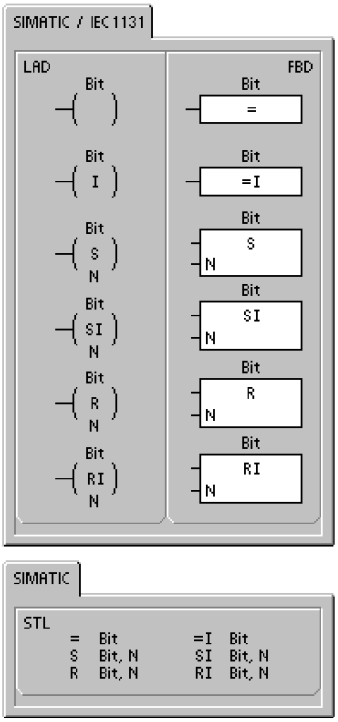
时序图



线圈

输出

输出指令（=）将新值写入输出点的过程映象寄存器。当输出指令执行时，S7-200 将输出过程映象寄存器中的位接通或者断开。在 LAD 和 FBD 中，指定点的值等于能流。在 STL 中，栈顶的值复制到指定位。



立即输出

当指令执行时，立即输出指令（=I）将新值同时写到物理输出点和相应的过程映象寄存器中。

当立即输出指令执行时，物理输出点立即被置为能流值。在 STL 中，立即指令将栈顶的值立即复制到物理输出点的指定位上。“I”表示立即，当指令执行时，新值会同时被写到物理输出和相应的过程映象寄存器。这一点不同于非立即指令，只把新值写入过程映象寄存器。

置位和复位

置位（S）和复位（R）指令将从指定地址开始的 N 个点置位或者复位。你可以一次置位或者复位 1-255 个点。

如果复位指令指定的是定时器或者计数器，指令不但复位定时器位或者计数器位，而且清除定时器或者计数器的当前值。

使 ENO＝0 的出错条件：

- 0006（间接寻址）
- 0091（操作数超出范围）

立即置位和立即复位

立即置位和立即复位指令将从指定地址开始的 N 个点立即置位或者立即复位。你可以一次立即置位或者立即复位 1-128 个点。“I”表示立即，当指令执行时，新值会同时被写到物理输出和相应的过程映象寄存器。这一点不同于非立即指令，只把新值写入过程映象寄存器。

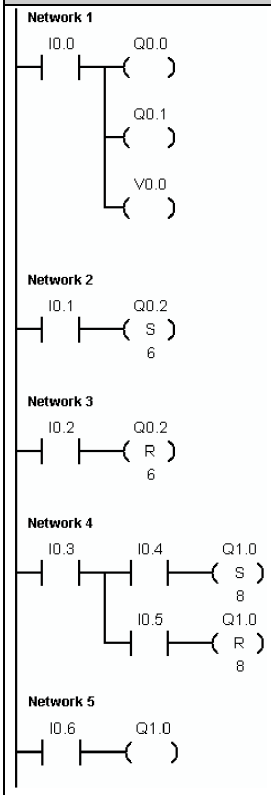
使 ENO＝0 的出错条件：

- 0006（间接寻址）
- 0091（操作数超出范围）

表 6-4　位逻辑输出指令的有效操作数

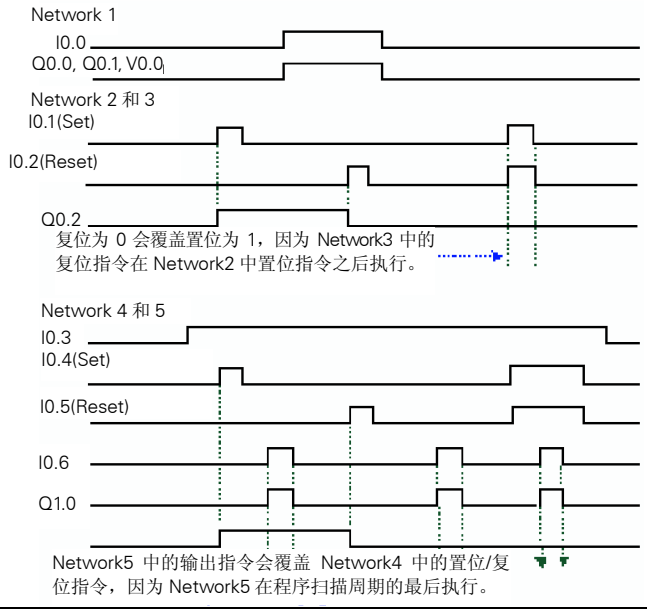
输入/输出	数据类型	操作数
位	BOOL	I、Q、V、M、SM、S、T、C、L
位（立即）	BOOL	Q
N	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数

线圈指令举例

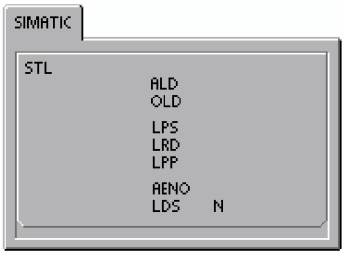



```
Network1   //输出指令为外部 I/O（I、Q）和内部存储
           //器（M、SM、T、C、V、S、L）指定位值。
LD     I0.0
=      Q0.0
=      Q0.1
=      V0.0
Network2   //连续将一组 6 位置为 1。
           //指定起始地址和置位的个数。
           //当第一位（Q0.2）的值为 1 时，置位指令
           //的程序状态指示器为 ON。
LD     I0.1
S      Q0.2，6
Network3   //连续将一组 6 位置为 0。
           //指定起始地址和复位的个数。
           //当第一位（Q0.2）的值为 0 时，复位指
           //令的程序状态指示器为 ON。
LD     I0.2
R      Q0.2，6
Network4   //置位和复位一组 8 个输出位（Q1.0~Q1.7）。
LD     I0.3
LPS
A      I0.4
S      Q1.0，8
LPP
A      I0.5
R      Q1.0，8
Network5   //置位和复位指令实现锁存器功能。
           //完成置位/复位功能，必须确保这些
           //位没有在其它指令中被改写。在本例
           //中，Network4 置位和复位一组
           //8 个输出位（Q1.0~Q1.7）。在 RUN 模式
           //下 Network5 会覆盖 Q1.0 的值，从而
           //控制 Network4 中的程序状态显示器。
LD     I0.6
=      Q1.0
```

时序图



逻辑堆栈指令



栈装载与

栈装载与指令对堆栈中第一层和第二层的值进行逻辑与操作，结果放入栈顶。执行完栈装载与指令后，堆栈深度减 1。

栈装载或

栈装载或指令对堆栈中第一层和第二层的值进行逻辑或操作，结果放入栈顶。执行完栈装载或指令后，堆栈深度减 1。

逻辑推入栈

逻辑推入栈指令复制栈顶的值，并将这个值推入栈。栈底的值被推出并消失。

逻辑读栈

逻辑读栈指令复制堆栈中的第二个值到栈顶。堆栈没有推入栈或者弹出栈操作，但旧的栈顶值被新的复制值取代。

逻辑弹出栈

逻辑弹出栈指令弹出栈顶的值，堆栈的第二个值成为新的栈顶值。

ENO 与

ENO 与指令对 ENO 位和栈顶的值进行逻辑与操作，其产生的效果与 LAD 或者 FBD 中盒指令的 ENO 位相同。与操作的结果成为新的栈顶。ENO 是 LAD 和 FBD 中盒指令的布尔输出。如果盒指令的 EN 输入有能流并且执行没有错误，则 ENO 将能流传递给下一元素。你可以把 ENO 作为指令成功完成的使能标志位。ENO 位被用作栈顶，影响能流和后续指令的执行。STL 中没有 EN 输入。条件指令要想执行，栈顶值必须为逻辑 1。在 STL 中也没有 ENO 输出。但是在 STL 中，那些与 LAD 和 FBD 中具有 ENO 输出的指令相应的指令，存在一个特殊的 ENO 位。它可以被 AENO 指令访问。

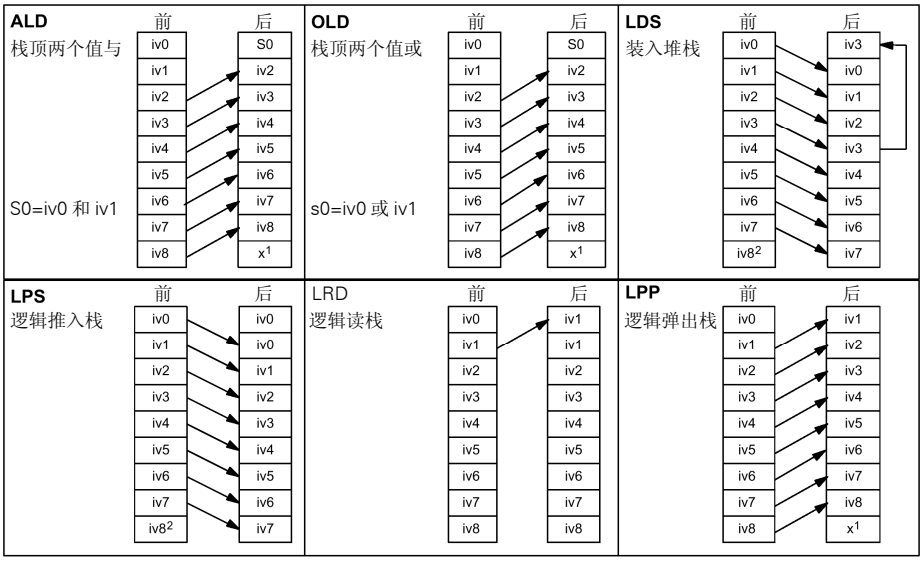
装入堆栈

装入堆栈指令复制堆栈中的第 N 个值到栈顶。栈底的值被推出并消失。

表 6-5 装入堆栈指令的有效操作数

输入/输出	数据类型	操作数
N	BYTE	常数（0 到 8）

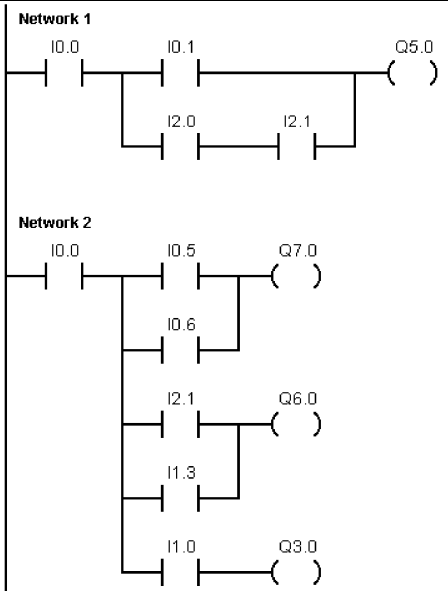
如图 6-2 中所示，S7-200 用逻辑堆栈来决定控制逻辑。在本例中，iv0 到 iv7 表示逻辑堆栈的初始值，nv 表示指令提供的一个新值，S0 表示逻辑堆栈中存储的计算值。



[1] 数值是不确定的（可以是 0，也可以是 1）
[2] 在逻辑入栈或者装入堆栈指令执行后，iv8 的值丢失。

图 6-2 逻辑堆栈指令的操作

逻辑堆栈指令程序举例




```
Network1
LD      I0.0
LD      I0.1
LD      I2.0
A       I2.1
OLD
ALD
=       Q5.0
Network2
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
```

RS 触发器指令

置位优先触发器是一个置位优先的锁存器。当置位信号（S1）和复位信号（R）都为真时，输出为真。

复位优先触发器是一个复位优先的锁存器。当置位信号（S）和复位信号（R1）都为真时，输出为假。

Bit 参数用于指定被置位或者复位的布尔参数。可选的输出反映 Bit 参数的信号状态。

表 6-7 中给出了例子程序的真值表。

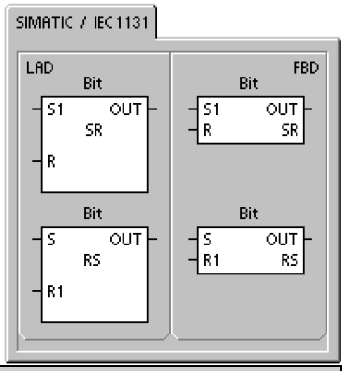
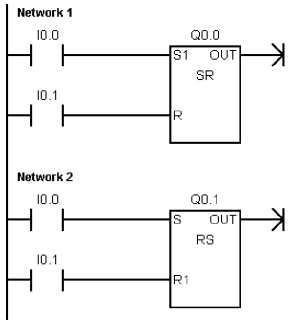


表 6-6 RS 触发器指令的有效操作数

输入/输出	数据类型	操作数
S1、R	BOOL	I、Q、V、M、SM、S、T、C、能流
S、R1、OUT	BOOL	I、Q、V、M、SM、S、T、C、L、能流
Bit	BOOL	I、Q、V、M、S

RS 触发器指令举例



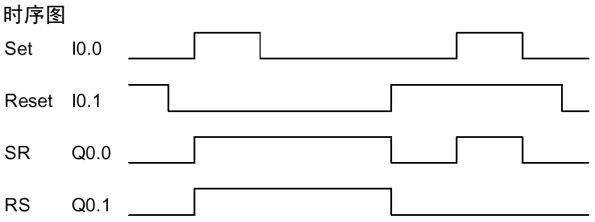
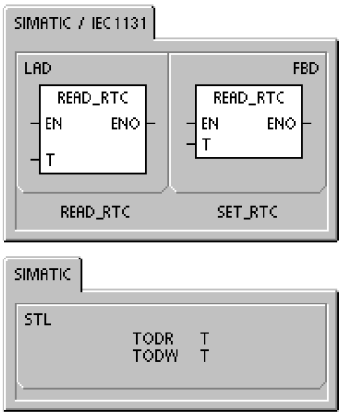


表 6-7 RS 触发器指令真值表

指令	S1	R	输出（Bit）
置位优先触发器指令（SR）	0	0	保持前一状态
	0	1	0
	1	0	1
	1	1	1
指令	**S**	**R1**	**输出（Bit）**
复位优先触发器指令（RS）	0	0	保持前一状态
	0	1	0
	1	0	1
	1	1	0

时钟指令

读实时时钟和写实时时钟



读实时时钟（TODR）指令从硬件时钟中读当前时间和日期，并把它装载到一个 8 字节，起始地址为 T 的时间缓冲区中。写实时时钟（TODW）指令将当前时间和日期写入硬件时钟，当前时钟存储在以地址 T 开始的 8 字节时间缓冲区中。

你必须按照 BCD 码的格式编码所有的日期和时间值（例如：用 16#97 表示 1997 年）。图 6-3 给出了时间缓冲区（T）的格式。

当扩展电源停电或者存储器丢失时，实时时钟会初始化为以下日期和时间：

日期： 01-Jan-90
时间： 00：00：00
星期： 星期日

使 ENO＝0 的出错条件：

- 0006（间接寻址）
- 0007（TOD 数据错误），只对写实时时钟指令有效。
- 000C（时钟模块不存在）

表 6-8 时钟指令的有效操作数

输入/输出	数据类型	操作数
T	BYTE	IB、QB、VB、MB、SMB、SB、LB、*VD、*LD、*AC

T	T+1	T+2	T+3	T+4	T+5	T+6	T+7
年 00~99	月 01~12	日 01~31	小时 00~23	分钟 00~59	秒 00~59	0	星期 0~7*

*T+7 1=星期日，7=星期六
0=禁用星期

图 6-3 8 字节时间缓冲区的格式

提示：

S7-200 CPU 不会检查和核实日期与星期是否合理。无效日期 February 30（2 月 30 日）可能被接受。故必须确保输入的数据是正确的。

不要同时在主程序和中断程序中使用 TODR/TODW 指令。如果这样做，而在执行 TOD 指令时出现了执行 TOD 指令的中断，则中断程序中的 TOD 指令不会被执行。SM4.3 指示了试图对时钟进行两个同时的访问（非致命错误 0007）。

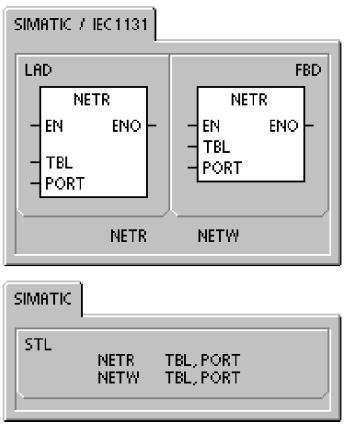
S7-200 PLC 只使用年信息的后两位，不会受到世纪跨越的影响。但是，用到年份进行计算或比较的用户程序必须考虑两位的表示方法和世纪的变化。

在 2096 年之前可以进行闰年的正确处理。

通讯指令

网络读写指令

网络读指令（NETR）初始化一个通讯操作，通过指定端口（PORT）从远程设备上采集数据并形成表（TBL）。网络写指令（NETW）初始化一个通讯操作，通过指定端口（PORT）向远程设备写表（TBL）中的数据。



使 ENO＝0 的出错条件：

- 0006（间接寻址）
- 如果功能返回出错信息，会置位表状态字节中的 E。（见图 6-4）

网络读指令可以从远程站点读取最多 16 个字节的信息，网络写指令可以向远程站点写最多 16 个字节的信息。

在程序中，你可以使用任意条网络读写指令，但是在同一时间，最多只能有 8 条网络读写指令被激活。例如，在所给的 S7-200 CPU 中，可以有 4 条网络读指令和 4 条网络写指令，或者 2 条网络读指令和 6 条网络写指令在同一时间被激活。


指令向导

你可以使用网络读写向导程序。要启动网络读写向导程序，在命令菜单中选择 Tools> Instruction Wizard，并且在指令向导窗口中选择网络读写。

表 6-9 网络读写指令的有效操作数

输入/输出	数据类型	操作数
TBL	BYTE	VB、MB、*VD、*LD、*AC
PORT	BYTE	常数　对于 CPU221、CPU222、CPU224：　0 对于 CPU226 和 CPU226XM：　0 或 1

图 6-4 中给出了 TBL 参数参照表，表 6-10 列出了错误代码。

D 完成 （操作已完成）： 0=未完成 1=完成
A 有效 （操作已被排队）： 0=无效 1=有效
E 错误 （操作返回一个错误）： 0=无错误 1=错误

远程站地址：被访问的 PLC 的地址

远程站的数据区指针：被访问数据的间接指针

数据长度：远程站上被访问数据的字节数

接收和发送数据区：如下描述的保存数据的 1 到 16 个字节

对 NETR，执行 NETR 指令后，从远程站读到的数据放在这个数据区

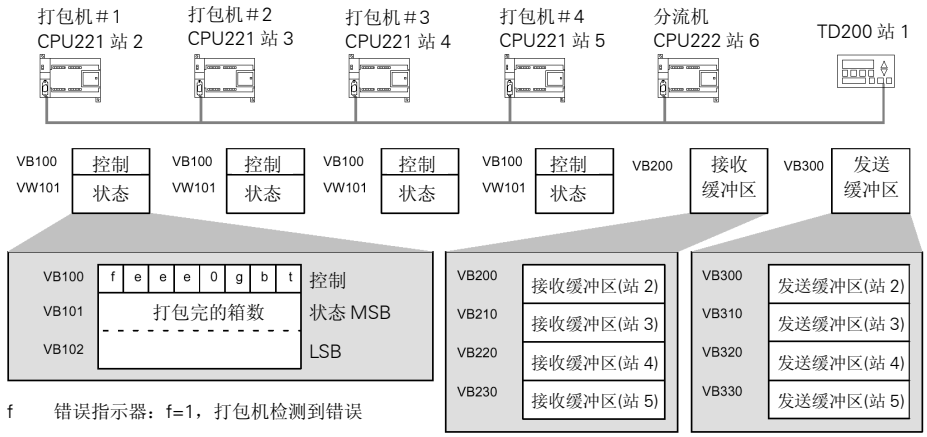
对 NETW，执行 NETW 指令前，要发送到远程站的数据放在这个数据区

图 6-4 网络读写指令的 TBL 参数

表 6-10 TBL 参数的错误代码

错误码	定义
0	无错误
1	时间溢出错，远程站点不响应
2	接收错：奇偶校验错，响应时帧或校验和出错
3	离线错：相同的站地址或无效的硬件引发冲突
4	队列溢出错：激活了超过 8 个 NETR/NETW 方框
5	违反通信协议：没有在 SMB30 中允许 PPI，就试图执行 NETR/NETW 指令
6	非法参数：NETR/NETW 表中包含非法或无效的值
7	没有资源：远程站点正在忙中（上装或下装程序在处理中）
8	第 7 层错误：违反应用协议
9	信息错误：错误的数据地址或不正确的数据长度
A-F	未用：（为将来的使用保留）

图 6-5 给出了一个实例来解释网络读写指令的使用。本例中，考虑一条生产线正在灌装黄油桶并将其送到四台包装机（打包机）中的一台上。打包机把 8 个黄油桶包装到一个纸板箱中。一个分流机控制着黄油桶流向各个打包机。4 个 CPU221 模块用于控制打包机，一个 CPU222 模块安装了 TD200 操作器接口，被用来控制分流机。图 6-5 给出了该网络配置。



f　错误指示器：f=1，打包机检测到错误

g　粘结剂供应慢：g=1，30 分钟内必须增加粘结剂

b　包装箱供应慢：b=1，30 分钟内必须增加包装箱

t　没有可包装的黄油桶：t=1，没有黄油桶

eee　识别出现的错误类型的错误码

图 6-5　网络读写指令举例

图 6-6 中给出了 2 号站中接收缓冲区（VB200）和发送缓冲区（VB300）中的数据。S7-200 使用网络读指令不断地读取每个打包机的控制和状态信息。每次某个打包机包装完 100 箱，分流机会注意到，并用网络写指令发送一条信息清除状态字。

用于读打包机＃1 的分流机接收缓冲区

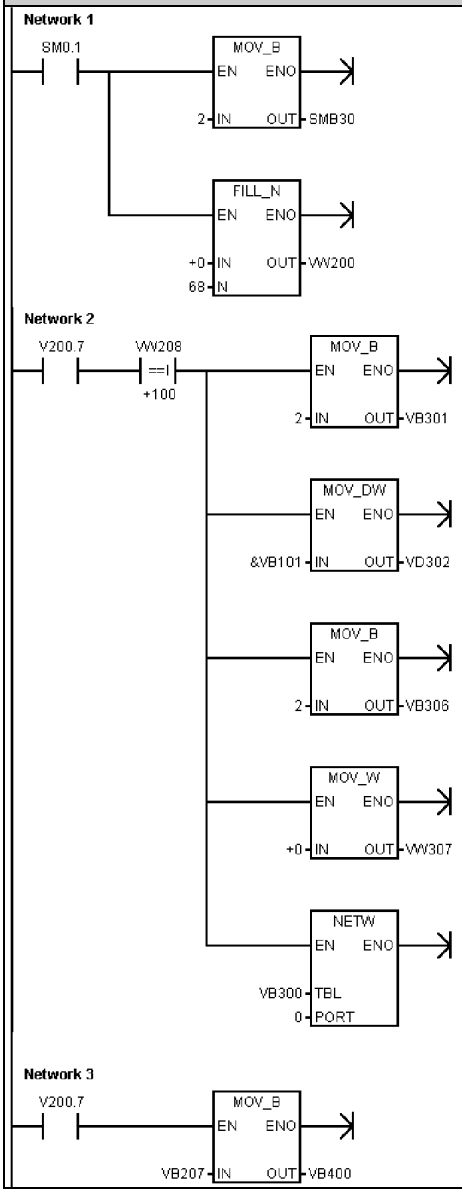
	7				0
VB200	D	A	E	0	错误码
VB201	远程站地址				
VB202	指向远程站（&VB100）				
VB203	的数据区指针				
VB204					
VB205					
VB206	数据长度=3 字节				
VB207	控制字节				
VB208	状态（最高有效字节）				
VB209	状态（最低有效字节）				

清除打包机＃1 读数的分流机发送缓冲区

	7				0
VB300	D	A	E	0	错误码
VB301	远程站地址				
VB302	指向远程站（&VB100）				
VB303	的数据区指针				
VB304					
VB305					
VB306	数据长度=2 字节				
VB307	0				
VB308	0				

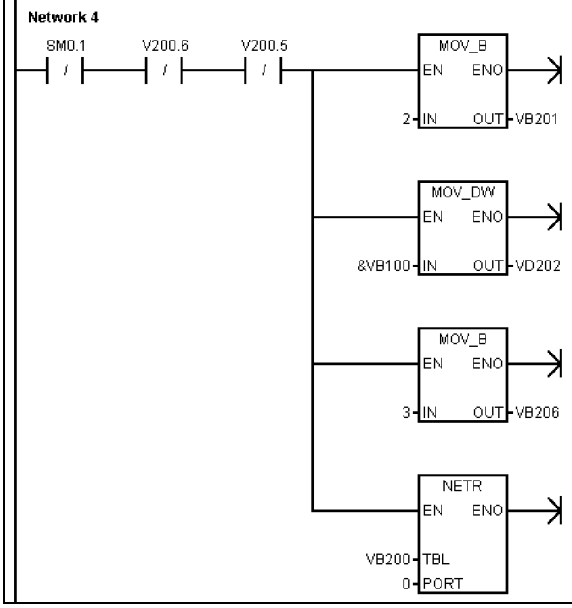
图 6-6　网络读写指令中 TBL 数据举例

网络读写指令程序举例




```
Network1  //在第一个扫描周期，使能 PPI 主站模式，并且
          //清除所有接收和发送缓冲区。
LD        SM0.1
MOVB      2，SMB30
FILL      +0，VW200，68
Network2  //当 NETR 完成标志位（V200.7）
          //置位并且包装完 100 箱时：
          //1. 装载 1＃包装机的站地址。
          //2. 装载指向远程站数据的指针。
          //3. 装载发送数据的长度。
          //4. 装载发送数据。
          //5. 复位 1＃包装机包装的箱数。
LD        V200.7
AW=       VW208，+100
MOVB      2，VB301
MOVD      &VB101，VD302
MOVB      2，VB306
MOVW      +0，VW307
NETW      VB300，0
Network3  //当 NETR 完成标志位置位时，保存来自 1＃包
          //装机的控制数据。
LD        V200.7
MOVB      VB207，VB400
```

网络读写指令程序举例



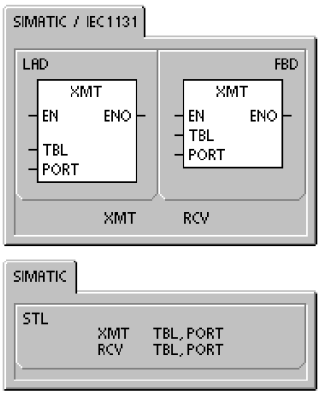

```
Network4  //如果不是第一个扫描周期并且没有错
          //误发生：
          //1. 装载 1＃包装机的站地址。
          //2. 装载指向远程站数据的指针。
          //3. 装载接收数据的长度。
          //4. 读 1＃包装机的控制和状态数据。
LDN       SM0.1
AN        V200.6
AN        V200.5
MOVB      2，VB201
MOVD      &VB100，VD202
MOVB      3，VB206
NETR      VB200，0
```

发送和接收指令

发送指令（XMT）用于在自由口模式下依靠通讯口发送数据。

接收指令（RCV）启动或者终止接收信息功能。你必须为接收操作指定开始和结束条件。

从指定的通讯口接收到的信息被存储在数据缓冲区（TBL）中。数据缓冲区的第一个数据指明了接收到的字节数。



使 ENO=0 的出错条件：

- 0006（间接寻址）
- 0009（在 Port 0 同时发送和接收）
- 000B（在 Port 1 同时发送和接收）
- RCV 参数错误，置位 SM86.6 或者 SM186.6
- S7-200 CPU 没有处于自由口模式。

表 6-11 发送和接收指令的有效操作数

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　对于 CPU221、CPU222、CPU224：　0 对于 CPU226 和 CPU226XM：　0 或 1

使用自由口模式控制串行通讯口

通过编程，你可以选择自由口模式来控制 S7-200 的串行通讯口。当选择了自由口模式，用户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讯口的操作。当处于自由口模式时，通讯协议完全由梯形图程序控制。SMB30（对于端口 0）和 SMB130（对于端口 1，如果你的 S7-200 有两个端口）被用于选择波特率和校验类型。

当 S7-200 处于 STOP 模式时，自由口模式被禁止，重新建立正常的通讯（例如：编程设备的访问）。

在最简单的情况下，可以只用发送指令（XMT）向打印机或者显示器发送信息。其它例子包括与条码阅读器、称重计和焊机的连接。在每种情况下，你都必须编写程序，来支持在自由口模式下与 S7-200 通讯的设备所使用的协议。

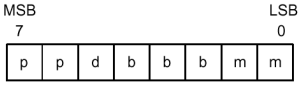
只有当 S7-200 处于 RUN 模式时，才能进行自由口通讯。要使能自由口模式，应该在 SMB30（端口 0）或者 SMB130（端口 1）的协议选择区中设置 01。处于自由口通讯模式时，不能与编程设备通讯。

提示

可以使用特殊寄存器位 SM0.7 来控制自由口模式。SM0.7 反映的是操作模式开关的当前位置。当 SM0.7=0 时，开关处于 TERM 位置；当 SM0.7=1 时，开关处于 RUN 位置。如果只有模式开关处于 RUN 位置时，才允许自由口模式，你可以将开关改变到其它位置上，使用编程设备监控 S7-200 的运行。

将 PPI 通讯转变为自由口模式

SMB30 和 SMB130 分别配置通讯口 0 和通讯口 1，并且为自由口操作提供波特率、校验和数据位数的选择。自由口的控制字节如图 6-7 所示。每个配置都产生一个停止位。



SMB30 = 端口 0
SMB130 = 端口 1

pp: 校验选择
- 00= 不校验
- 01= 偶校验
- 10= 不校验
- 11= 奇校验

d: 每个字符的数据位
- 0= 每个字符 8 位
- 1= 每个字符 7 位

bbb: 自由口波特率
- 000=38，400 波特
- 001=19，200 波特
- 010=9，600 波特
- 011=4，800 波特
- 100=2，400 波特
- 101=1，200 波特
- 110=115.2K 波特 [1]
- 111=57.6K 波特 [1]

mm: 协议选择
- 00=PPI/从站模式
- 01=自由口模式
- 10=PPI/主站模式
- 11=保留（缺省设置为 PPI/从站模式）

[1] S7-200 CPU 版本在 1.2 以上支持 57.6K 和 115.2K 波特率。

图 6-7 自由口模式 SM 控制字节

发送数据

发送指令使你能够发送一个字节或多个字节的缓冲区，最多为 255 个。

图 6-8 给出了发送缓冲区的格式。

图 6-8 发送缓冲区的格式

如果有一个中断服务程序连接到发送结束事件上，在发送完缓冲区中的最后一个字符时，则会产生一个中断（对端口 0 为中断事件 9，对端口 1 为中断事件 26）。

你可以不使用中断来执行发送指令（例如：向打印机发送信息）。通过监视 SM4.5 或者 SM4.6 信号，判断发送是否完成。

把字符数设置为 0 并执行 XMT 指令，可以产生一个 BREAK 状态。这样产生的 BREAK 状态，在线上会持续以当前波特率传输 16 位数据所需要的时间。发送 BREAK 的操作和发送其它任何信息的操作是一样的。当 BREAK 完成时，产生一个发送中断并且 SM4.5 或者 SM4.6 反应发送操作的当前状态。

接收数据

接收指令使你能够接收一个字节或多个字节的缓冲区，最多为 255 个。

如果有一个中断服务程序连接到接收信息完成事件上，在接收完缓冲区中的最后一个字符时，S7-200 会产生一个中断（对端口 0 为中断事件 23，对端口 1 为中断事件 24。）

图 6-9 接收缓冲区的格式

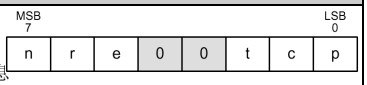
你可以不使用中断，通过监视 SMB86（端口 0）或者 SMB186（端口 1）来接收信息。当接收指令未被激活或者已经被中止时，这一字节不为 0；当接收正在进行时，这一字节为 0。

如表 6-12 中所示，接收指令允许你选择信息的启始和结束条件。使用 SMB86 至 SMB94 对端口 0 进行设置，SMB186 至 SMB194 对端口 1 进行设置。

提示

当超限或有校验错误时，接收信息功能会自动终止。必须为接收信息功能操作定义一个启始条件和一个结束条件（最大字符数）。

表 6-12 接收缓冲区字节（SMB86 到 SMB94 和 SMB186 到 SMB194）

端口 0	端口 1	描述
SMB86	SMB186	接收信息状态字节  n： 1= 用户通过禁止命令结束接收信息 r： 1= 接收信息结束：输入参数错误或缺少起始和结束条件 e： 1= 收到结束字符 t： 1= 接收信息结束：超时 c： 1= 接收信息结束：字符数超长 p： 1= 接收信息结束：奇偶校验错误
SMB87	SMB187	接收信息状态字节  en：0= 禁止接收信息功能 1= 允许接收信息功能 每次执行 RCV 指令时检查允许/禁止接收信息位。 sc：0= 忽略 SMB88 或 SMB188 1= 使用 SMB88 或 SMB188 的值检测起始信息 ec：0= 忽略 SMB89 或 SMB189 1= 使用 SMB89 或 SMB189 的值检测结束信息 il： 0= 忽略 SMW90 或 SMW190 1= 使用 SMW90 值检测空闲状态 c/m：0= 定时器是内部字符定时器 1= 定时器是信息定时器 tmr：0= 忽略 SMW92 或 SMW192 1= 当执行 SMW92 或 SMW192 时终止接收 bk：0= 忽略中断条件 1= 使用中断条件来检测起始信息。 信息的中断控制字节位用来定义识别信息的标准。信息的起始和结束均需定义。 起始信息：il*sc+bk*sc 结束信息=ec+tmr+最大字符数 起始信息编程： 1. 空闲检测： il=1,sc=0,bk=0,SMW90>0 2. 起始字符检测： il=0,sc=1,bk=0,SMW90 被忽略 3. 中断检测： il=0,sc=1,bk=1,SMW90 被忽略 4. 对一个信息的响应： il=1,sc=0,bk=0,SMW90=0 （信息定时器用来终止没有响应的接收） 5. 中断一个起始字符： il=0,sc=1,bk=1,SMW90 被忽略 6. 空闲和一个起始字符： il=1,sc=1,bk=0,SMW90>0 7. 空闲和起始字符（非法）： il=1,sc=1,bk=0,SMW90=0 注意：通过超时和奇偶校验错误（如果允许），可以自动结束接收过程。
SMB88	SMB188	信息字符的开始
SMB89	SMB189	信息字符的结束
SMB90 SMB91	SMB190 SMB191	空闲线时间段按毫秒设定。空闲线时间溢出后接收的第一个字符是新的信息的开始字符。SMB90（或 SMB190）是最高有效字节，SMB91（或 SMB191）是最低有效字节。
SMB92 SMB93	SMB192 SMB193	中间字符/信息定时器溢出值按毫秒设定。如果超过这个时间段，则终止接收信息。SMB92（或 SMB192）是最高有效字节，SMB93（或 SMB193）是最低有效字节。
SMB94	SMB194	要接收的最大字符数（1 到 255 字节）。 注：这个范围必须设置到所希望的最大缓冲区大小，即使信息的字符数始终达不到。

接收指令的启动和结束条件

接收指令使用接收信息控制字节（SMB87 或者 SMB187）中的位来定义信息的起始和结束条件。

提示

当接收指令执行时，在接收口上有来自其它器件的信号，接收信息功能有可能从一个字符的中间开始接收字符，从而导致校验错误和接收信息功能的中止。如果校验没有被使能，接收到的信息有可能包含错误字符。当起始条件被指定为一个特定的起始字符或任意字符时，这种情况有可能发生，正象下面第 2 条和第 6 条中所描述的那样。

接收指令支持几种信息起始条件。在指定的起始条件中引入一段停顿或者一个空闲检测可以避免以上问题。在将字符装入信息缓冲区之前，用起始字符可以强制接收信息功能与信息的起始同步。

接收指令支持几种起始条件：

1．*空闲线检测：*空闲线条件是指在传输线上一段安静或者空闲的时间。在 SMW90 或者 SMW190 中指定其毫秒数。当接收指令在程序中执行时，接收信息功能对空闲线条件进行检测。如果在空闲线时间到之前接收到任何字符，接收信息功能会忽略那些字符并且按照 SMW90 或者 SMW190 中给定的时间值重新启动空闲线定时器。如图 6-10 所示。在空闲线时间到之后，接收信息功能将所有接收到的字符存入信息缓冲区。

空闲线时间应该总是大于在指定波特率下传输一个字符（包括起始位、数据位、校验位和停止位）的时间。空闲线时间的典型值为在指定波特率下传输三个字符的时间。

对于二进制协议、没有特定起始字符的协议或者指定了信息之间最小时间间隔的协议，你可以使用空闲线检测作为起始条件。

设置：il=1，sc=0，bk=0，SMW90/SMW190=空闲线超时时间，单位为毫秒

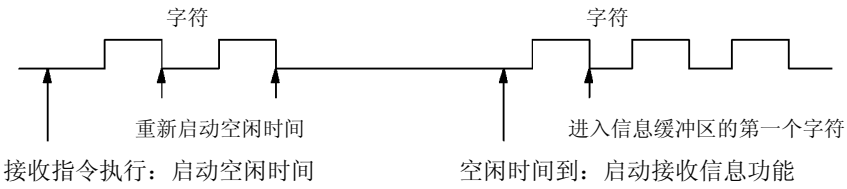


图 6-10 用空闲时间检测来启动接收指令

2．*起始字符检测：*起始字符可以是用于作为一条信息首字符的任意一个字符。当接收到 SMB88 或者 SMB188 中指定的起始字符后，一条信息开始。接收信息功能将起始字符作为信息的第一个字符存入接收缓冲区。接收信息功能忽略所有在起始字符之前接收到的字符。起始字符和起始字符之后接收到的所有字符一起存入信息缓冲区。

通常，对于所有信息都使用同一字符作为起始的 ASCII 码协议，你可以使用起使字符检测。

设置：il=0，sc=1，bk=0，SMW90/SM190 被忽略，SMB88/SMB188=起始字符

3．*空闲线和起始字符：*接收指令可以用空闲线和起始字符的组合来启动一条消息。当接收指令执行时，接收信息功能检测空闲线条件。在空闲线条件满足后，接收信息功能搜寻指定的起始字符。如果接收到的字符不是起始字符，接收信息功能重新检测空闲线条件。所有在空闲线条件满足和接收到起始字符之前接收到的字符被忽略掉。起始字符与字符串一起存入信息缓冲区。

空闲线时间应该总是大于在指定波特率下传输一个字符（包括起始位、数据位、校验位和停止位）的时间。空闲线时间的典型值为在指定波特率下传输三个字符的时间。

通常，对于指定信息之间最小时间间隔并且信息的首字符是特定设备的站号或其它信息的协议，你可以使用这种类型的起始条件。这种方式尤其适用于在通讯连接上有多个设备的情况。在这种情况下，只有当接收到的信息的起始字符为特定的站号或者设备时，接收指令才会触发一个中断。

设置：il=1，sc=1，bk=0，SMW90/SM190>0，SMB88/SMB188=起始字符

4．*断点检测：*断点是指在大于一个完整字符传输时间的一段时间内，接收数据一直为 0。一个完整字符传输时间定义为传输起始位、数据位、校验位和停止位的时间总和。如果接收指令被配置为用接收一个断点作为信息的起始，则任何在断点之后接收到的字符都会存入信息缓冲区。任何在断点之前接收到的字符被忽略。

通常，只有当通讯协议需要时，才使用断点检测作为起始条件。

设置：il=0，sc=0，bk=1，SMW90/SM190 被忽略，SMB88/SMB188 被忽略

5．*断点和起始字符：*接收指令可以被配置为接收到断点条件和一个指定的起始字符之后，启动接收。在断点条件满足之后，接收信息功能寻找特定的起始字符。如果收到了除起始字符以外的任意字符，接收信息功能重新启动寻找新的断点。所有在断点条件满足和接收到起始字符之前接收到的字符都会被忽略。起始字符与字符串一起存入信息缓冲区。

设置：il=0，sc=1，bk=1，SMW90/SM190 被忽略，SMB88/SMB188=起始字符

6．*任意字符：*接收指令可以被配置为立即接收任意字符并把全部接收到的字符存入信息缓冲区。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 或者 SMW190）被设置为 0。这使得接收指令一经执行，就立即开始接收字符。

设置：il=1，sc=0，bk=0，SMW90/SMW190=0，SMB88/SMB188 被忽略

用任意字符开始一条信息允许使用信息定时器，来监控信息接收是否超时。这对于自由口协议的主站是非常有用的，并且当在指定时间内，没有来自从站的任何响应的情况，也需要采取超时处理。由于空闲线时间被设置为 0，当接收指令执行时，信息定时器启动。如果没有其它终止条件满足，信息定时器超时会结束接收信息功能。

设置：il=1，sc=0，bk=0，SMW90/SMW190=0，SMB88/SMB188 被忽略

c/m=1，tmr=1，SMW92/SMW192=信息超时时间，单位为毫秒。

接收指令支持几种结束信息的方式。结束信息的方式可以是以下一种或者几种的组合：

1．*结束字符检测：*结束字符是用于表示信息结束的任意字符。在找到起始条件之后，接收指令检查每一个接收到的字符，并且判断它是否与结束字符匹配。如果收到了结束字符，将其存入信息缓冲区，接收结束。

通常，对于所有信息都使用同一字符作为结束的 ASCII 码协议，你可以使用结束字符检测。你可以使用结束字符检测与字符间隔定时器、信息定时器或者最大字符计数相结合来结束一条信息。

设置：ec=1，SMB89/SMB189=结束字符

2. *字符间隔定时器：*字符间隔时间是指从一个字符的结尾（停止位）到下一个字符的结尾（停止位）之间的时间。如果两个字符之间的时间间隔（包括第二个字符）超过了 SMW92 或者 SMW192 中指定的毫秒数，接收信息功能结束。接收到字符后，字符间隔定时器重新启动，见图 6-11。

当协议没有特定的信息结束字符时，你可以用字符间隔定时器来结束一条信息。由于定时器总是包含接收一个完整字符（包括起始位、数据位、校验位和停止位）的时间，因而该时间值应设置为大于在指定波特率下传输一个字符的时间。

你可以使用字符间隔定时器与结束字符检测或者最大字符计数相结合，来结束一条信息。

设置：c/m=0，tmr=1，SMW92/SMW192=信息超时时间，单位为毫秒。

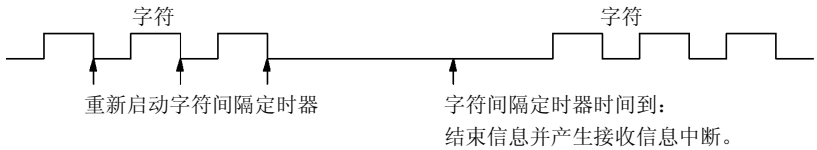


图 6-11 使用字符间隔定时器来结束接收指令

3. *信息定时器：*从信息的开始算起，在经过一段指定的时间之后，信息定时器结束一条信息。接收信息功能的启动条件一满足，信息定时器就启动。当经过的时间超出 SMW92 或者 SMW192 中指定的毫秒数时，信息定时器时间到，见图 6-12。

通常，当通讯设备不能保障字符中间没有时间间隔或者使用 Modem 通讯时，你可以使用信息定时器。对于 Modem 方式，你可以用信息定时器指定一个从信息开始算起，接收信息允许的最大时间。信息定时器的典型值是在当前波特率下，接收到最长信息所需时间值的大约 1.5 倍。

你可以使用信息定时器与结束字符检测或者最大字符计数相结合，来结束一条信息。

设置：c/m=1，tmr=1，SMW92/SMW192=信息超时时间，单位为毫秒。

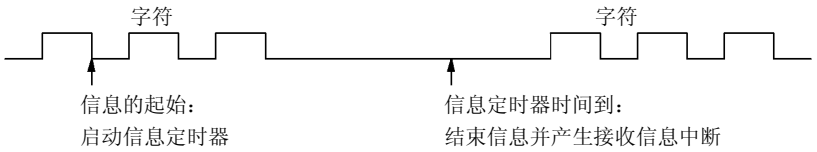


图 6-12 使用信息定时器来结束接收指令

4. *最大字符计数：*必须告诉接收指令接收字符的最大个数（SMB94 或者 SMB194）。当达到或者超出这个值，接收信息功能结束。即使不会被用作结束条件，接收指令要求用户指定一个最大字符个数。这是因为只有知道了接收到数据的最大长度，才能确保信息缓冲区之后的用户数据不会被覆盖。

   对于信息的长度已知并且恒定的协议，可以使用最大字符计数来结束信息。最大字符计数总是与结束字符检测、字符间隔定时器或者信息定时器结合在一起使用。

5. *校验错误：*当接收字符的同时出现硬件信号校验错误时，接收指令会自动结束。只有在 SMB30 或者 SMB130 中使能了校验位，才有可能出现校验错误。没有办法禁止此功能。

6. *用户结束：*用户可以通过程序来结束接收信息功能，先将 SMB87 或 SMB187 中的使能位置为 0，再次执行接收指令即可。这样可以立即结束接收信息功能。

使用字符中断控制接收数据

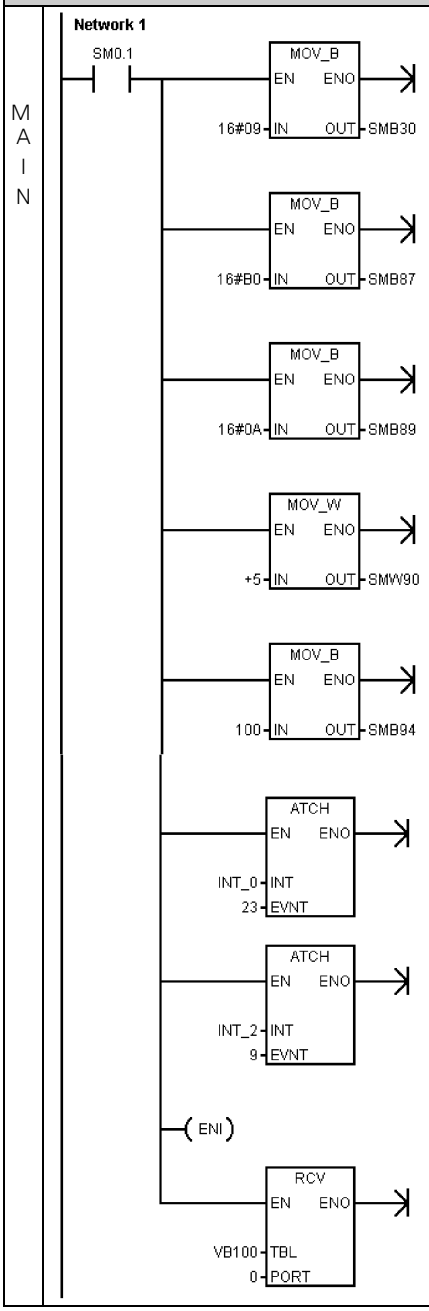
为了完全适应对各种协议的支持，你也可以使用字符中断控制的方式接收数据。接收每个字符时都会产生中断。在执行与接收字符事件相连的中断服务程序之前，接收到的字符存入 SMB2 中，校验状态（如果使能的话）存入 SM3.0。SMB2 是自由口接收字符缓冲区。在自由口模式下，每一个接收到的字符都会存放到这一位置，便于用户程序访问。SMB3 用于自由口模式。它包含一个校验错误标志位。当接收字符的同时检测到校验错误时，该位被置位。该字节的其它位被保留。利用该位丢弃信息或者向该信息发否定应答。

在较高的波特率下（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：在 38.4 时为 260 毫秒；在 57.6K 时为 173 毫秒；在 115.2K 时为 86 毫秒。确保你的中断服务程序足够短，不会丢失字符或者使用接收指令。

提示

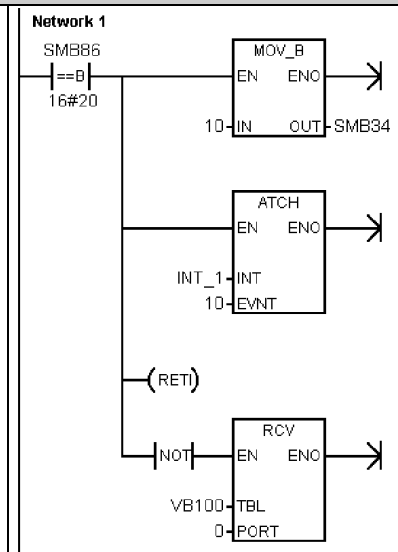
SMB2 和 SMB3 是端口 0 和端口 1 共用的。当字符的接收来自端口 0 时，执行与事件（中断事件 8）相连的中断服务程序，SMB2 中存储从端口 0 接收到的字符，SMB3 中存储该字符的校验状态。当字符的接收来自端口 1 时，执行与事件（中断事件 25）相连的中断服务程序，SMB2 中存储从端口 1 接收到的字符，SMB3 中存储该字符的校验状态。

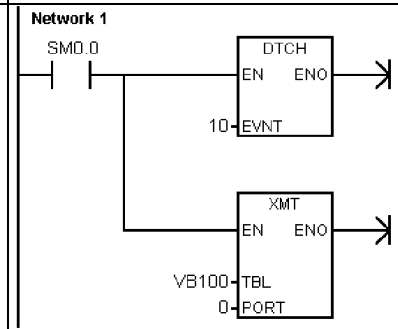
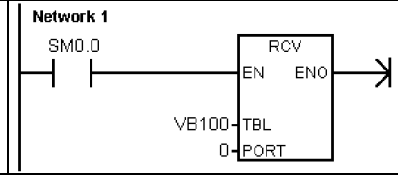
发送和接收指令程序举例




```
Network1  //本程序接收一个字符串，直到接收到换行字符。
          //接收完成后，信息会发送回到发送方。
LD     SM0.1          //在第一个扫描周期
MOVB   16#09，SMB30   //1.初始化自由口：
                      //  选择 9600 波特
                      //  选择 8 位数据位
                      //  选择无校验
MOVB   16#B0，SMB87   //2.初始化 RCV 信息控制字节：
                      //  RCV 使能
                      //  检测信息结束字符
                      //  检测空闲线信息条件
MOVB   16#0A，SMB89   //3.设定信息结束字符为
                      //  16#0A（换行字符）
MOVW   +5，SMW90      //4.设置空闲线超时为 5ms
MOVB   100，SMB94     //5.设置最大字符数为 100
ATCH   INT_0，23      //6.连接中断 0 到接收结束事件
ATCH   INT_2，9       //7.连接中断 2 到发送结束事件
ENI                   //8.允许用户中断
RCV    VB100，0       //9.执行接收指令
                      //  接收缓冲区指向 VB100
```

发送和接收指令举例		

INT 0		Network1 //接收完成中断
		//1.如果接收状态显示接收到结束字符，
		// 连接一个 10ms 定时器触发发送，然后返回。
		//2.如果由于任何其它原因接收完成，
		// 启动一个新的接收。
		LDB= SMB86，16＃20
		MOVB 10，SMB34
		ATCH INT_1，10
		CRETI
		NOT
		RCV VB100，0

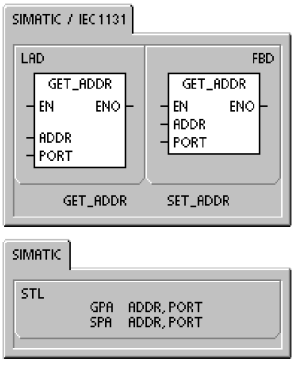
INT 1		Network1 //10ms 定时器中断 //1.断开定时器中断 //2.在端口 0 向用户回送信息 LD SM0.0 DTCH 10 XMT VB100，0
INT 2		Network1 //发送完成中断 //允许另一个接收 LD SM0.0 RCV VB100，0

获取口地址和设定口地址指令

获取口地址指令（GPA）读取 PORT 指定的 CPU 口的站地址，并将数值放入 ADDR 指定的地址中。

设定口地址指令（SPA）将口的站地址（PORT）设置为 ADDR 指定的数值。

新地址不能永久保存。重新上电后，口地址将返回到原来的地址值（用系统块下载的地址）。



使 ENO=0 的错误条件：

- 0006（间接寻址）

- 0004（试图在中断服务程序中执行设定口地址指令）

表 6-13 获取口地址和设定口地址指令的有效操作数

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 （常数值仅用于设定口地址指令。）
PORT	BYTE	常数　对于 CPU221、CPU222、CPU224：　0 对于 CPU226 和 CPU226XM：　0 或者 1

比较指令

数值比较

比较指令用于比较两个数值：

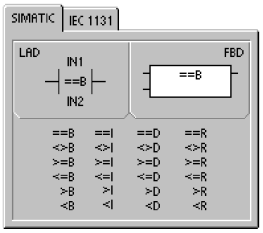
IN1=IN2　IN1>=IN2　IN1<=IN2
IN1>IN2　IN1<IN2　IN1<>IN2

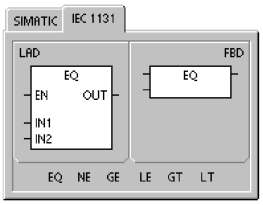
字节比较操作是无符号的。
整数比较操作是有符号的。
双字比较操作是有符号的。
实数比较操作是有符号的。

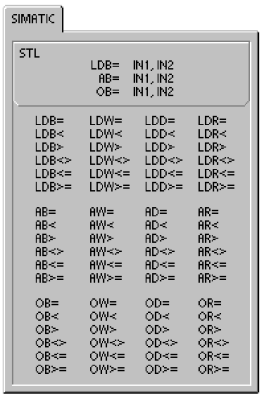
对于 LAD 和 FBD：当比较结果为真时，比较指令使能点闭合（LAD）或者输出接通（FBD）。

对于 STL：当比较结果为真时，将栈顶值置 1。

当你使用 IEC 比较指令时，你可以使用各种数据类型作为输入。但是，两个输入的数据类型必须一致。







注意：

下列情况是致命错误，并且会导致 S7-200 立即停止执行用户程序：

- 非法的间接地址（任意比较指令）
- 非法的实数（例如：NAN），（实数比较指令）

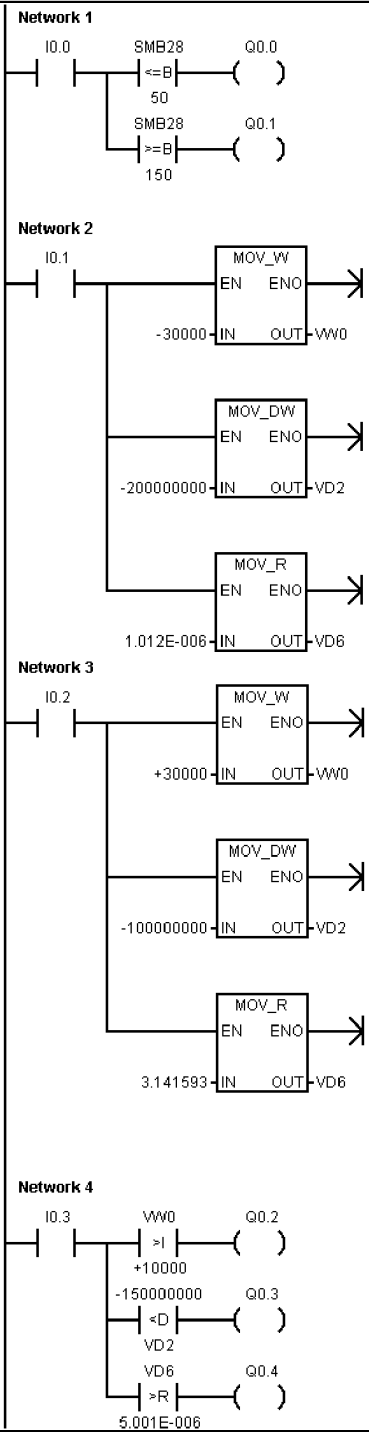
为了避免这些情况的发生，在执行比较指令之前，要确保合理使用了指针和存储实数的数值单元。

不管能流的状态如何，比较指令都会被执行。

表 6-14 比较指令的有效操作数

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、LW、T、C、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	I、Q、V、M、SM、S、T、C、L、能流

比较指令程序举例



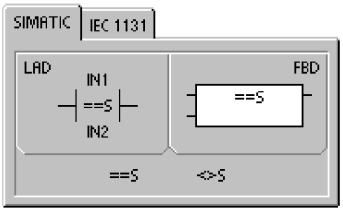

```
Network1  //调节模拟调节电位器 0 来改变 SMB28 的数值。
          //当 SMB28 中的数值小于等于 50 时，Q0.0 输出
          //当 SMB28 中的数值大于等于 150 时，Q0.1 输出
          //当比较结果为真时，状态指示器点亮。
LD        I0.0
LPS
AB<=      SMB28，50
=         Q0.0
LPP
AB>=      SMB28，150
=         Q0.1
Network2  //在 V 存储器地址中装载较小的数值，
          //使比较结果为假并且关闭状态指示器。
LD        I0.1
MOVW      -30000，VW0
MOVD      -200000000，VD2
MOVR      1.012E-006，VD6
Network3  //在 V 存储器地址中装载较大的数值，
          //使比较结果为真并且点亮状态指示器。
LD        I0.2
MOVW      +30000，VW0
MOVD      -10000000，VD2
MOVR      3.141593，VD6
Network4  //整数字比较检测 VW0>+10000 是否为真。
          //在程序中使用常数是为了显示不同的数据类型，
          //你也可以比较两个存储在程序存储区中的数值，
          //例如：VW0>VW100。
LD        I0.3
LPS
AW>       VW0，+10000
=         Q0.2
LRD
AD<       -150000000，VD2
=         Q0.3
LPP
AR>       VD6，5.001E-006
=         Q0.4
```

字符串比较

字符串比较指令比较两个字符串的 ASCII 码字符：

IN1=IN2　　IN1<>IN2

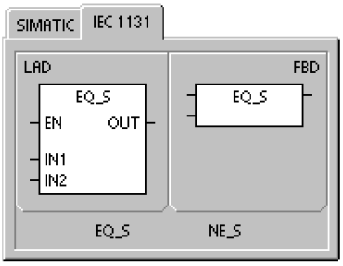
当比较结果为真时，比较指令使触点闭合（LAD）或者输出接通（FBD），或者将栈顶值置 1（STL）。



注意：

下列情况是致命错误，并且会导致 S7-200 立即停止执行用户程序：

- 非法的间接地址（任意比较指令）
- 字符串的长度超过 254 个字符（字符串比较指令）
- 一个字符串的起始地址和长度使它不适合所指定的存储区（字符串比较指令）



为了避免这些情况的发生，在执行比较指令之前，要确保合理使用了指针和保存 ASCII 码字符串的存储区。确保一个保存 ASCII 码字符串的缓冲区能够在指定的存储区完整的保留。

不管能流的状态如何，比较指令都会执行。

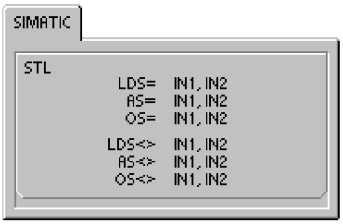


表 6-15　字符串比较指令的有效操作数

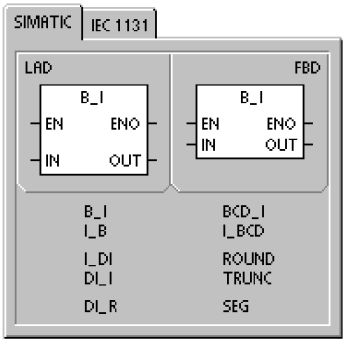
输入/输出	数据类型	操作数
IN1、IN2	BYTE(字符)	VB、LB、*VD、*LD、*AC
OUT	BOOL	I、Q、V、M、SM、S、T、C、L、能流

转换指令

标准转换指令

数字转换

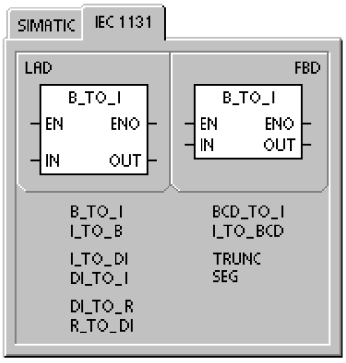
字节转为整数（BTI）、整数转为字节（ITB）、整数转为双整数（ITD）、双整数转为整数（DTI）、双整数转为实数（DTR）、BCD 码转为整数（BCDI）和整数转为 BCD 码（IBCD）。以上指令将输入值 IN 转换为指定的格式并存储到由 OUT 指定的输出值存储区中。例如：你可以将双整数值转为实数值；你也可以在整数和 BCD 码格式之间相互转换。



四舍五入和取整

四舍五入指令（ROUND）将一个实数转为一个双整数值，并将四舍五入的结果存入 OUT 指定的变量中。

取整指令（TRUNC）将一个实数转为一个双整数值，并将实数的整数部分作为结果存入 OUT 指定的变量中。



段码

段码指令（SEG）允许你产生一个点阵，用于点亮七段码显示器的各个段。

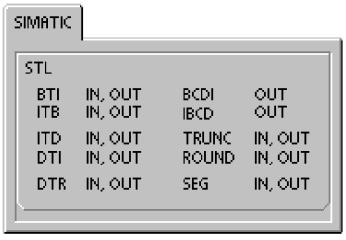


表 6-16　标准转换指令的有效操作数

输入/输出	数据类型	操作数
IN	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数
	WORD、INT	IW、QW、VW、MW、SMW、SW、T、C、LW、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、INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、*VD、*LD、*AC
	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC

BCD 码转为整数和整数转为 BCD 码的操作

BCD 码转整数指令（BCDI）将一个 BCD 码 IN 的值转换成整数值，并且将结果存入 OUT 指定的变量中。IN 的有效范围是 0 到 9999 的 BCD 码。

整数转 BCD 码指令（IBCD）将输入的整数值 IN 转换成 BCD 码，并且将结果存入 OUT 指定的变量中。IN 的有效范围是 0 到 9999 的整数。

使 ENO=0 的错误条件：
- SM1.6（无效的 BCD 码）
- 0006（间接寻址）

影响的 SM 标志位：
- SM1.6（无效的 BCD 码）

双整数转为实数指令的操作

双整数转实数指令（DTR）将一个 32 位的符号整数转换成一个 32 位的实数，并将结果存入 OUT 指定的变量中。

使 ENO=0 的错误条件：
- 0006（间接寻址）

双整数转为整数指令的操作

双整数转整数指令（DTI）将一个双整数值 IN 转换为一个整数值，并且存入 OUT 指定的变量中。

如果所转换的数值太大以致于无法在输出中表示，则溢出标志位置位并且输出不会改变。

使 ENO=0 的错误条件：
- SM1.1（溢出）
- 0006（间接寻址）

影响的 SM 标志位：
- SM1.1（溢出）

整数转为双整数指令的操作

整数转双整数指令（ITD）将整数值 IN 转换成双整数值，并且存入 OUT 指定的变量中。符号位扩展到高字节中。

使 ENO=0 的错误条件：
- 0006（间接寻址）

字节转为整数指令的操作

字节转整数指令（BTI）将字节值 IN 转换成整数值，并且存入 OUT 指定的变量中。字节是无符号的，因而没有符号位扩展。

使 ENO=0 的错误条件：
- 0006（间接寻址）

整数转为字节指令的操作

整数转字节指令（ITB）将一个字的值 IN 转换成一个字节值，并且存入 OUT 指定的变量中。只有 0 到 255 中的值被转换，其它值会产生溢出并且输出不会改变。

使 ENO=0 的错误条件：
- SM1.1（溢出）
- 0006（间接寻址）

影响的 SM 标志位：
- SM1.1（溢出）

提示：

如果想将一个整数转换成实数，先用整数转双整数指令，再用双整数转实数指令。

四舍五入取整和取整指令的操作

四舍五入取整指令（ROUND）将实数值 IN 转换成双整数值，并且存入 OUT 指定的变量中。如果小数部分大于等于 0.5，则数字向上取整。

取整指令（TRUNC）将一个实数值 IN 转换成一个双整数，并且存入 OUT 指定的变量中。只有实数的整数部分被转换，小数部分舍去。

使 ENO=0 的错误条件：

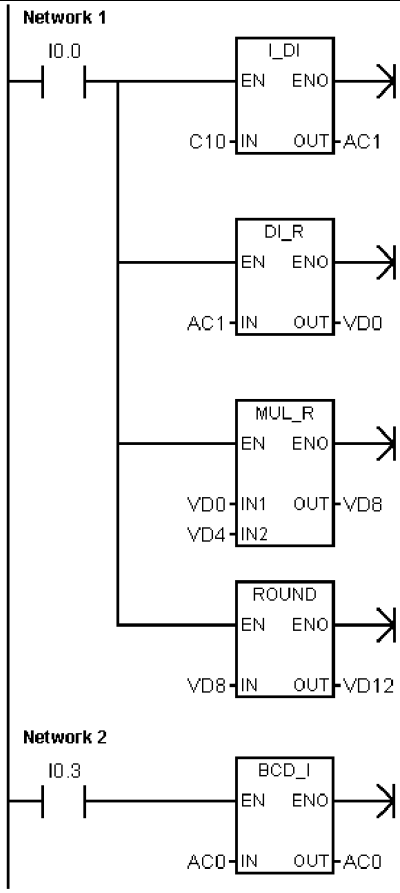
- SM1.1（溢出）
- 0006（间接寻址）

影响的 SM 标志位：

- SM1.1 （溢出）

如果所转换的不是一个有效的实数，或者其数值太大以致于无法在输出中表示，则溢出标志位置位并且输出不会改变。

标准转换指令程序举例




```
Network1    //转换英寸为厘米：
            //1.装载计数值（英寸）到 AC1。
            //2.将该值转换为实数。
            //3.乘以 2.54（转换为厘米）。
            //4.将数值转回整数。
LD          I0.0
ITD         C10，AC1
DTR         AC1，VD0
MOVR        VD0，VD8
*R          VD4，VD8
ROUND       VD8，VD12
Network2    //将 BCD 码转为整数
LD          I0.3
BCDI        AC0
```

双整数转实数和四舍五入取整指令

C10	101	计数值为 101 英寸
VD0	101.0	计数值（实数形式）
VD4	2.54	2.54 常数（英寸到厘米）
VD8	256.54	256.54 厘米数的实数形式
VD12	257	257 厘米（双整数形式）

BCD 码转为整数

AC0	1234
	BCDI
AC0	04D2

段码指令的操作

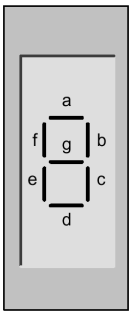
要点亮七段码显示器中的段，可以使用段码指令。段码指令将 IN 中指定的字符（字节）转换生成一个点阵并存入 OUT 指定的变量中。

点亮的段表示的是输入字节中低 4 位所代表的字符。图 6-13 给出了段码指令使用的七段码显示器的编码。

使 ENO=0 的错误条件：

- 0006（间接寻址）

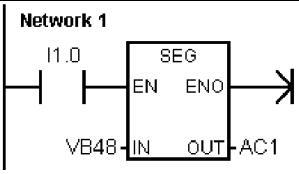
输入 LSD	七段码显示器	输出 -gfe dcba
0	0	0011 1111
1	1	0000 0110
2	2	0101 1011
3	3	0100 1111
4	4	0110 0110
5	5	0110 1101
6	6	0111 1101
7	7	0000 0111



输入 LSD	七段码显示器	输出 -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

图 6-13 七段码显示器的编码

段码指令程序举例



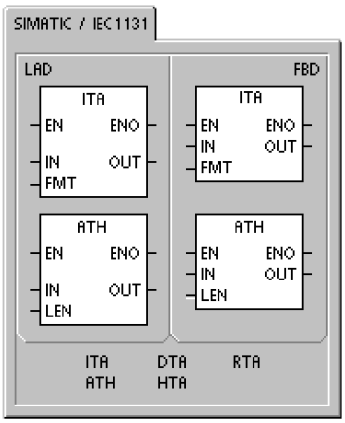

```
Network1
LD    I1.0
SEG   VB48，AC1
```

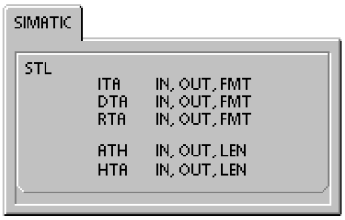
05 VB48　段码　6D AC1

5　（显示字符）

ASCII 码转换指令

有效的 ASCII 码字符为十六进制的 30 到 39 和 41 到 46。





在 ASCII 码和十六进制数之间相互转换

ASCII 码转十六进制数指令（ATH）将一个长度为 LEN 从 IN 开始的 ASCII 码字符串转换成从 OUT 开始的十六进制数。十六进制数转 ASCII 码指令（HTA）将从输入字节 IN 开始的十六进制数，转换成从 OUT 开始的 ASCII 码字符串。被转换的十六进制数的位数由长度 LEN 给出。

能够被转换的 ASCII 码字符串或者十六进制数的最大数量为 255。

使 ENO=0 的错误条件：

- SM1.7（非法的 ASCII 码）只对 ATH 有效
- 0006（间接寻址）
- 0091（操作数超出范围）

影响的 SM 标志位

- SM1.7（非法的 ASCII 码）

将数值转为 ASCII 码

整数转为 ASCII 码（ITA），双整数转为 ASCII 码（DTA）和实数转为 ASCII 码（RTA）指令，分别将整数、双整数或者实数值转换为 ASCII 码字符串。

表 6-17 ASCII 码转换指令的有效操作数

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

整数转 ASCII 码指令的操作

整数转 ASCII 码（ITA）指令将一个整数字 IN 转换成一个 ASCII 码字符串。格式 FMT 指定小数点右侧的转换精度和小数点是使用逗号还是点号。转换结果放在 OUT 指定的连续 8 个字节中。

ASCII 码字符串始终是 8 个字节。

使 ENO=0 的错误条件：

- 0006（间接寻址）
- 非法的格式
- nnn>5

整数转 ASCII 码指令的格式操作数如图 6-14 所示。输出缓冲区的大小始终是 8 个字节，nnn 用于指定输出缓冲区中小数点右侧的位数。nnn 的有效范围为 0 到 5。将小数点右侧的位数定为 0，使得所显示的数值没有小数点。对于 nnn 大于 5 的情况，输出缓冲区会被空格键的 ASCII 码填冲。c 指定是用逗号（c=1）或者点号（c=0）作为整数和小数的分隔符。高 4 位必须为 0。

图 6-14 中给出了一个数值的例子，其格式为使用点号（c=0），小数点右侧有三位小数（nnn=011）。输出缓冲区的格式符合以下规则：

- 正数值写入输出缓冲区时没有符号位。
- 负数值写入输出缓冲区时以负号（-）开头。
- 小数点左侧的开头的 0（除去靠近小数点的那个之外）被隐藏。
- 数值在输出缓冲区中是右对齐的。

FMT

MSB 7	6	5	4	3	2	1	LSB 0
0	0	0	0	c	n	n	n

c=逗号（1）或点号（0）
nnn=小数点右边的位数

	Out	Out +1	Out +2	Out +3	Out +4	Out +5	Out +6	Out +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

图 6-14 整数转 ASCII 码（ITA）指令的 FMT 操作数

双整数转 ASCII 码指令操作

双整数转 ASCII 码（DTA）指令将一个双字 IN 转换成一个 ASCII 码字符串。格式操作数 FMT 指定小数点右侧的转换精度。转换结果存储在从 OUT 开始的连续 12 个字节中。

使 ENO=0 的错误条件：

- 0006（间接寻址）
- 非法的格式
- nnn>5

输出缓冲区的大小总是 12 个字节。

图 6-15 描述了双整数转 ASCII 码指令的格式操作数。nnn 给出在输出缓冲区中，小数点右侧的数字位数。nnn 的有效范围为 0 到 5。指定小数点右侧的位数为 0，导致显示的值没有小数点。对于 nnn 大于 5 的情况，输出缓冲区会被空格键的 ASCII 码填冲。c 指定是用逗号（c=1）或者点号（c=0）作为整数和小数的分隔符。高 4 位必须为 0。

图 6-15 中给出了一个数值的例子，其格式为使用点号（c=0），小数点右侧有四位小数（nnn=100）。输出缓冲区的格式符合以下规则：

- 正数值写入输出缓冲区时没有符号位。
- 负数值写入输出缓冲区时以负号（-）开头。
- 小数点左侧的开头的 0（除去靠近小数点的那个之外）被隐藏。
- 数值在输出缓冲区中是右对齐的。

FMT

MSB 7	6	5	4	3	2	1	LSB 0
0	0	0	0	c	n	n	n

c=逗号（1）或点号（0）
nnn=小数点右边的位数

	Out	Out +1	Out +2	Out +3	Out +4	Out +5	Out +6	Out +7	Out +8	Out +9	Out +10	Out +11
in=–12						–	0	.	0	0	1	2
in=1234567					1	2	3	.	4	5	6	7

图 6-15 双整数转 ASCII 码（DTA）指令的 FMT 操作数

实数转 ASCII 码指令操作

实数转 ASCII 码指令(RTA)将一个实数值 IN 转为 ASCII 码字符串。格式操作数 FMT 指定小数点右侧的转换精度，小数点是用逗号还是用点号表示和输出缓冲区的大小。

使 ENO=0 的错误条件：

- 0006（间接寻址）
- nnn>5
- ssss<3
- ssss<OUT 中的字符个数

转换结果存储在从 OUT 开始的输出缓冲区中。

输出 ASCII 码字符串的位数（长度）就是输出缓冲区的大小，它的值可以在 3 到 15 个字节或字符之间。

S7-200 的实数格式支持最多 7 位小数。试图显示 7 位以上的小数会产生一个四舍五入错误。

图 6-16 是对 RTA 指令中格式操作数 FMT 的描述。ssss 表示输出缓冲区的大小。0、1 或者 2 个字节的大小是不合理的。nnn 表示输出缓冲区中小数点右侧的数字位数。nnn 的合理范围为 0 到 5。如果指定小数点右侧为 0 位数字，导致显示的值没有小数点。对于 nnn 大于 5 或者指定的输出缓冲区太小以致于无法存储转换值的情况，输出缓冲区会被空格键的 ASCII 码填冲。c 指定是用逗号（c=1）或者点号（c=0）作为整数和小数的分隔符。

图 6-16 中给出了一个数值的例子，其格式为：使用点号（c=0）、小数点右侧有 1 位小数（nnn=001）和 6 个字节的缓冲区大小（ssss=0110）。输出缓冲区的格式符合以下规则：

- 正数值写入输出缓冲区时没有符号位。
- 负数值写入输出缓冲区时以负号（-）开头。
- 小数点左侧开头的 0（除去靠近小数点的那个之外）被隐藏。
- 小数点右侧的数值按照指定的小数点右侧的数字位数被四舍五入。
- 输出缓冲区的大小应至少比小数点右侧的数字位数多三个字节。
- 数值在输出缓冲区中是右对齐的。

FMT

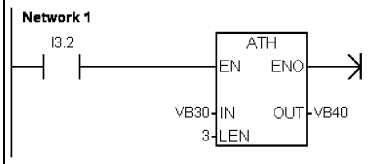
MSB 7	6	5	4	3	2	1	LSB 0
s	s	s	s	c	n	n	n

ssss=输出缓冲区的大小
c=逗号（1）或者点号（0）
nnn=小数点右侧的位数

	Out	Out +1	Out +2	Out +3	Out +4	Out +5
in = 1234.5	1	2	3	4	.	5
in = –0.0004				0	.	0
in = –3.67526			–	3	.	7
in = 1.95				2	.	0

图 6-16 实数转 ASCII 码（RTA）指令的 FMT 操作数

ASCII 码转十六进制数指令程序举例

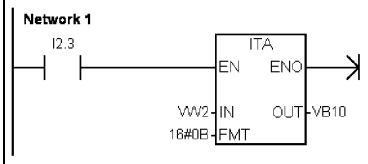



```
Network1
LD      I3.2
ATH     VB30，VB40，3
```

'3'	'4'	'A'			
33	45	41	ATH	3E	Ax
VB30				VB40	

注意：X 表示低四位（半个字节）未发生变化。

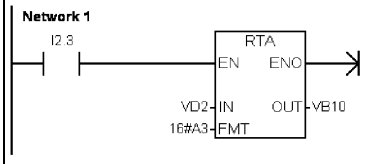
整数转 ASCII 码指令程序举例




```
Network1  //将 VW2 中的整数值转换为从 VB10
          //开始的 8 个 ASCII 码字符，
          //使用 16#0B 的格式，
          //用逗号作小数点，保留 3 位小数。
LD        I2.3
ITA       VW2，VB10，16#0B
```

		' '	' '	'1'	'2'	','	'3'	'4'	'5'
12345	ITA	20	20	31	32	2C	33	34	35
VW2		VB10	VB11	...					

实数转 ASCII 码指令程序举例



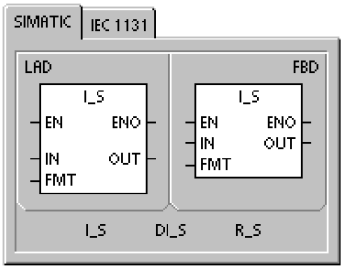

```
Network1  //将 VD2 中的实数值转换为从 VB10
          //开始的 10 个 ASCII 码字符，
          //使用 16＃A3 的格式，
          //用点号作小数点，保留 3 位小数。
LD        I2.3
RTA       VD2，VB10，16＃A3
```

		' '	' '	' '	'1'	'2'	'3'	'.'	'4'	'5'	'0'
123.45	RTA	20	20	20	31	32	33	2E	34	35	30
VD2		VB10	VB11	...							

字符串转换指令

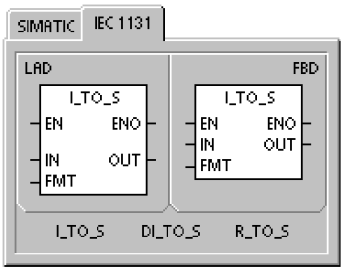
将数值转换为字符串

整数转字符串（ITS）、双整数转字符串（DTS）和实数转字符串（RTS）指令，将整数、双整数或者实数转换为 ASCII 码字符串。



整数转字符串的操作

整数转字符串指令（ITS）将一个整数字 IN 转换为 8 个字符长的 ASCII 码字符串。



格式操作数 FMT 指定小数点右侧的转换精度和使用逗号还是点号作为小数点。结果字符串被写入从 OUT 开始的 9 个连续字节中。要得到更多信息，请参见第 4 章字符串的格式一节。

使 ENO=0 的错误条件：

- 0006（间接寻址）
- 0091（操作数超出范围）
- 非法格式（nnn>5）

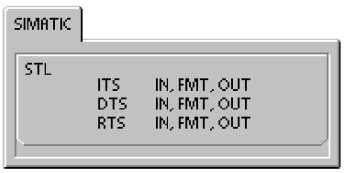


图 6-17 是对整数转字符串指令中格式操作数的描述。输出字符串的长度总是 8 个字符。nnn 表示输出缓冲区中小数点右侧的数字位数。nnn 的合理范围为 0 到 5。如果指定小数点右侧为 0 位数字，则导致显示的值没有小数点。如果 nnn 的值大于 5，输出是由 8 个空格键的 ASCII 码组成的字符串。c 指定是用逗号（c=1）或者点号（c=0）作为整数和小数的分隔符。格式操作数的高 4 位必须为 0。

图 6-17 中给出了一个数值的例子，其格式为：使用点号（c=0）并且小数点后保留 3 位小数。OUT 的值为字符串的长度。

输出缓冲区的格式符合以下规则：

- 正数值写入输出缓冲区时没有符号位。
- 负数值写入输出缓冲区时以负号（-）开头。
- 小数点左侧开头的 0（除去靠近小数点的那个之外）被隐藏。
- 在输出字符串中的数值是右对齐的。

表 6-18 数值转字符串指令的有效操作数

输入/输出	数据类型	操作数
IN	BYTE(字符)	VB、LB、*VD、*LD、*AC
	INT	IW、QW、VW、MW、SMW、SW、T、C、LW、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
LNDX、FMT	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数
OUT	BYTE(字符)	VB、LB、*VD、*LD、*AC
	INT	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

FMT

MSB							LSB
7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

c=逗号（1）或者点号（0）
nnn=小数点右侧的位数

	Out	Out +1	Out +2	Out +3	Out +4	Out +5	Out +6	Out +7	Out +8
in=12	8				0	.	0	1	2
in=–123	8				0	.	1	2	3
in=1234	8				1	.	2	3	4
in = –12345	8		–	1	2	.	3	4	5

图 6-17 整数转字符串指令的 FMT 操作数

双整数转字符串指令操作

双整数转字符串指令（DTS）将一个双整数 IN 转换为一个长度为 12 个字符的 ASCII 码字符串。格式操作数 FMT 指定小数点右侧的转换精度和使用逗号还是点号作为小数点。结果字符串被写入从 OUT 开始的连续 13 个字节。要得到更多信息，请参见第 4 章字符串的格式一节。

使 ENO=0 的错误条件：

- 0006（间接寻址）
- 0091（操作数超出范围）
- 非法格式（nnn>5）

图 6-18 是对双整数转字符串指令中格式操作数的描述。输出字符串的长度总是 13 个字符。nnn 表示输出缓冲区中小数点右侧的数字位数。nnn 的合理范围为 0 到 5。如果指定小数点右侧为 0 位数字，则导致显示的值没有小数点。如果 nnn 的值大于 5，输出是由 12 个空格键的 ASCII 码组成的字符串。c 指定是用逗号（c=1）或者点号（c=0）作为整数和小数的分隔符。格式操作数的高 4 位必须为 0。

图 6-18 中给出一个数值的例子，其格式为：使用点号（c=0）并且小数点后保留 4 位小数。OUT 的值为字符串的长度。输出缓冲区的格式符合以下规则：

- 正数值写入输出缓冲区时没有符号位。
- 负数值写入输出缓冲区时以负号（-）开头。
- 小数点左侧开头的 0（除去靠近小数点的那个之外）被隐藏。
- 在输出字符串中的数值是右对齐的。

FMT

MSB							LSB
7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

c=逗号（1）或者点号（0）
nnn=小数点右侧的位数

	Out	Out +1	Out +2	Out +3	Out +4	Out +5	Out +6	Out +7	Out +8	Out +9	Out +10	Out +11	Out +12
in=12	12					.	–	0	.	0	0	1	2
in=–1234567	12					1	2	3	.	4	5	6	7

图 6-18 双整数转字符串指令的 FMT 操作数

实数转字符串指令操作

实数转字符串指令（RTS）将一个实数值 IN 转换为一个 ASCII 码字符串。格式操作数 FMT 指定小数点右侧的转换精度和使用逗号还是点号作为小数点。转换结果放在从 OUT 开始的一个字符串中。结果字符串的长度由格式操作数给出，它可以是 3 到 15 个字符。要得到更多信息，请参见第 4 章字符串的格式一节。

使 ENO=0 的错误条件：

- 0006（间接寻址）
- 0091（操作数超出范围）
- 非法格式
  nnn>5
  ssss<3
  ssss<要求的字符数

S7-200 的实数格式支持最多 7 位小数。试图显示 7 位以上的小数会产生一个四舍五入错误。

图 6-19 是对实数转字符串指令中格式操作数的描述。ssss 表示输出字符串的长度。0、1 或者 2 个字节的大小是不合理的。nnn 表示输出缓冲区中小数点右侧的数字位数。nnn 的合理范围是 0 到 5。如果指定小数点右侧为 0 位数字，则导致显示的值没有小数点。对于 nnn 大于 5 或者指定的输出缓冲区太小以致于无法存储转换值的情况，输出缓冲区会被空格键的 ASCII 码填冲。c 指定是用逗号（c=1）或者点号（c=0）作为整数和小数的分隔符。

图 6-19 中给出了一个数值的例子，其格式为：使用点号（c=0），小数点右侧有 1 位小数（nnn=001）和 6 个字节的缓冲区大小（ssss=0110）。OUT 的值为字符串的长度。输出字符串的格式符合以下规则：

- 正数值写入输出缓冲区时没有符号位。
- 负数值写入输出缓冲区时以负号（-）开头。
- 小数点左侧开头的 0（除去靠近小数点的那个之外）被隐藏。
- 小数点右侧的数值按照指定的小数点右侧的数字位数被四舍五入。
- 输出缓冲区的大小应至少比小数点右侧的数字位数多三个字节。
- 数值在输出缓冲区中是右对齐的。

FMT

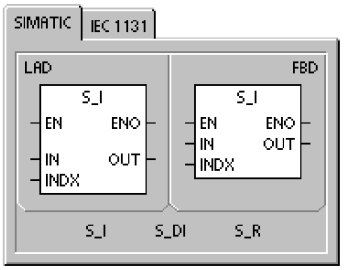
MSB 7	6	5	4	3	2	1	LSB 0
s	s	s	s	c	n	n	n

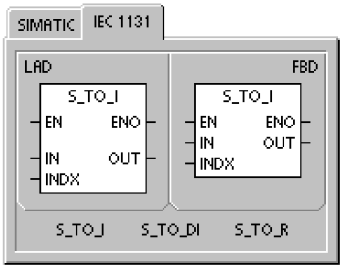
ssss=输出字符串长度
c=逗号（1）或者点号（0）
nnn=小数点右侧的位数

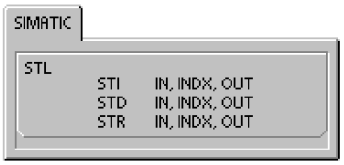
	Out	Out +1	Out +2	Out +3	Out +4	Out +5	Out +6
in=1234.5	6	1	2	3	4	.	5
in= –0.0004	6				0	.	0
in= –3.67526	6			–	3	.	7
in = 1.95	6				2	.	0

图 6-19 实数转字符串指令的 FMT 操作数

将子字符串转换为数字值







子字符串转整数（STI）、子字符串转双整数（STD）和子字符串转实数（STR）指令，将一个字符串 IN，从偏移量 INDX 开始，转换为整数、双整数或者实数值，存储在 OUT 中。

使 ENO=0 的错误条件：

- 0006（间接寻址）
- 0091（操作数超出范围）
- 009B（偏移量=0）
- SM1.1（溢出）

子字符串转整数和子字符串转双整数指令将字符串转换为以下格式：[空格][+或-][数字 0-9]

子字符串转实数指令将字符串转换为以下格式：[空格][+或-][数字 0-9][.或，][数字 0-9]

INDX 值通常设置为 1，从字符串的第一个字符开始转换。INDX 可以被设置为其它值，从字符串的不同位置进行转换。这可以被用于字符串中包含非数值字符的情况。例如：输入字符串为“Temperature：77.8”，你可以将 INDX 设为 13，这样就可以跳过字符串开头的“Temperature：”。子字符串转实数指令不能用于转换以科学计数法或者指数形式表示实数的字符串。指令不会产生溢出错误（SM1.1），但是它会将字符串转换到指数之前，然后停止转换。

例如：字符串“1.234E6”转换为实数值 1.234，并且没有错误提示。

当到达字符串的结尾或者遇到第一个非法字符时，转换指令结束。

非法字符是指任意非数字（0-9）字符。

当转换产生的整数值过大以致输出值无法表示时，溢出标志（SM1.1）会置位。例如：当输入字符串产生的数值大于 32767 或者小于-32768 时，子字符串转整数指令会置位溢出标志。

当输入字符串中并不包含可以转换的合法数值时，溢出标志（SM1.1）也会置位。例如：如果输入字符串的“A123”，转换指令会置位 SM1.1（溢出）并且输出值保持不变。

表 6-19 子字符串转换为数值指令的有效操作数

输入/输出	数据类型	操作数
IN	BYTE(字符)	IB、QB、VB、MB、SMB、SB、LB、*VD、*LD、*AC、常数
INDX	BYTE	VB、IB、QB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数
OUT	BYTE(字符)	VB、IB、QB、MB、SMB、SB、LB、*VD、*LD、*AC、常数
	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

对于整数和双整数
合法的输入字符串

输入字符串	输出整数
'123'	123
'–00456'	–456
'123.45'	123
'+2345'	2345
'000000123ABCD'	123

对于实数
合法的输入字符串

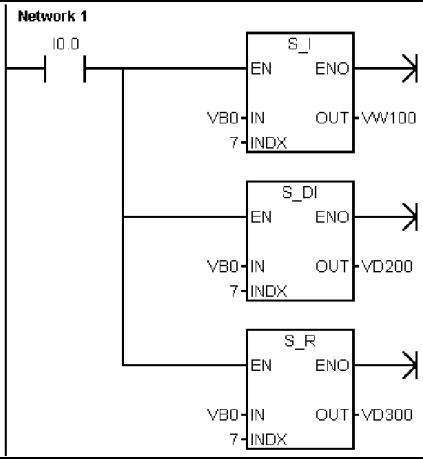
输入字符串	输出实数
'123'	123.0
'–00456'	–456.0
'123.45'	123.45
'+2345'	2345.0
'00.000000123'	0.000000123

非法的输入
字符串

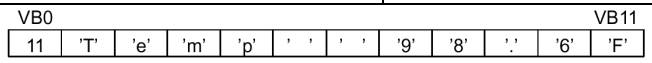
输入字符串
'A123'
' '
'++123'
'+–123
'+ 123'

图 6-20 合法和非法的输入字符串举例

子字符串转整数、双整数和实数指令程序举例




```
Network1   //将数字字符串转为整数。
           //将数字字符串转为双整数。
           //将数字字符串转为实数。
LD     I0.0
STI    VB0，7，VW100
STD    VB0，7，VD200
STR    VB0，7，VD300
```



程序执行后：
VW100（整数）=98
VD200（双整数）=98
VD300（实数）=98.6

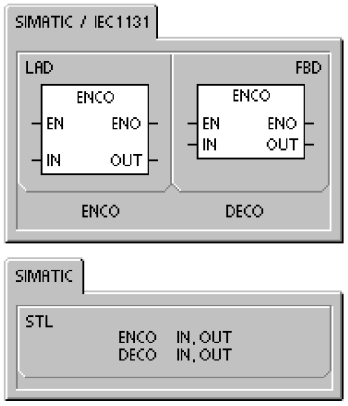
编码和解码指令

编码

编码指令（ENCO）将输入字（IN）的最低有效位的位号写入输出字节（OUT）的低四位（半个字节）。

译码

译码指令（DECO）根据输入字节（IN）的低四位所表示的位号置输出字（OUT）的相应位为 1，其它位清 0。



SM 标志位和 ENO

对于编码和译码指令，下列条件影响 ENO。

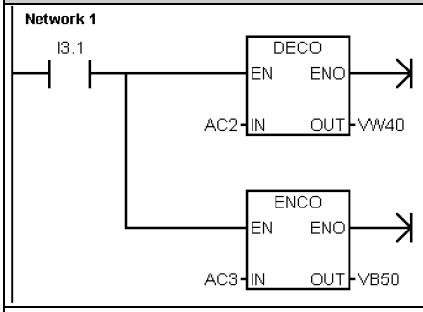
使 ENO=0 的错误条件：

- 0006（间接寻址）

表 6-20 编码和解码指令的有效操作数

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

译码和编码指令程序举例




```
Network1  //AC2 中包含错误检测位
          //1.译码指令将 VW40 中相应位置位
          //2.编码指令将最低有效位转换为
          //  错误代码，存入 VB50 中。
LD        I3.1
DECO      AC2，VW40
ENCO      AC3，VB50
```

AC2 3

DECO

VW40 15 0000 0000 0000 1000 3 0

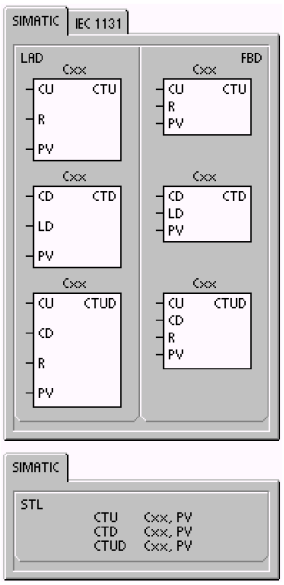
AC3 15 1000 0010 0000 0000 9 0

ENCO

VB50 9

计数器指令

SIMATIC 计数器指令



增计数器

增计数指令（CTU）从当前计数值开始，在每一个（CU）输入状态从低到高时递增计数。当 CXX 的当前值大于等于预置值 PV 时，计数器位 CXX 置位。当复位端（R）接通或者执行复位指令后，计数器被复位。当它达到最大值（32，767）后，计数器停止计数。

STL 操作：

- 复位输入：栈顶
- 计数输入：其值被装载在第二个堆栈中。

减计数器

减计数指令（CTD）从当前计数值开始，在每一个（CD）输入状态的低到高时递减计数。当 CXX 的当前值等于 0 时，计数器位 CXX 置位。当装载输入端（LD）接通时，计数器位被复位，并将计数器的当前值设为预置值 PV。当计数值到 0 时，计数器停止计数，计数器位 CXX 接通。

STL 操作：

- 装载输入：栈顶
- 计数输入：其值被装载在第二个堆栈中。

增/减计数器

增/减计数指令（CTUD），在每一个增计数输入（CU）的低到高时增计数，在每一个减计数输入（CD）的低到高时减计数。计数器的当前值 CXX 保存当前计数值。在每一次计数器执行时，预置值 PV 与当前值作比较。

当达到最大值（32767）时，在增计数输入处的下一个上升沿导致当前计数值变为最小值（-32768）。当达到最小值（-32768）时，在减计数输入端的下一个上升沿导致当前计数值变为最大值（32767）。

当当前值大于或者等于预置值 PV 时，计数器位 CXX 接通。否则，计数器位关断。当复位输入端（R）接通或者执行复位指令时，计数器被复位。当达到预置值 PV 时，CTUD 计数器停止计数。

STL 操作：

- 复位输入：栈顶
- 减计数输入：其值被装载在第二个堆栈中。
- 加计数输入：其值被装载在第二个堆栈中。

表 6-21　SIMATIC 计数器指令的有效操作数

输入/输出	数据类型	操作数
CXX	WORD	常数（C0 到 C255）
CU、CD、 LD、R	BOOL	I、Q、V、M、SM、S、T、C、L、能流
PV	INT	IW、QW、VW、MW、SMW、SW、LW、T、C、AC、AIW、*VD、*LD、*AC、常数

提示:

由于每一个计数器只有一个当前值，所以不要多次定义同一个计数器。（具有相同标号的增计数器、增/减计数器、减计数器访问相同的当前值。）

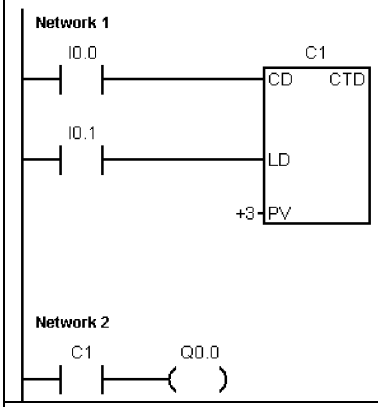
当使用复位指令复位计数器时，计数器位复位并且计数器当前值被清零。计数器标号既可以用来表示当前值，又可以用来表示计数器位。

表 6-22　计数器指令的操作

类型	操作	计数器位	上电周期/首次扫描
CTU	CU 使当前值递增，当前值持续递增直至 32767。	当当前值≥预置值时，计数器位接通。	计数器位关断。 当前值可以保留。
CTUD	CU 使当前值递增 CD 使当前值递减 当前值持续递增或递减除非计数器被复位	当当前值≥预置值时，计数器位接通。	计数器位关断。 当前值可以保留。
CTD	CD 使当前值递减直至当前值为 0。	当当前值＝0 时，计数器位接通。	计数器位关断。 当前值可以保留。

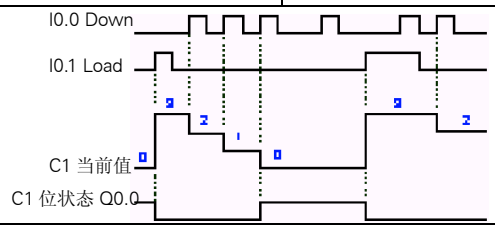
[1] 你可以选择计数器的当前值是否掉电保持。关于 S7-200 CPU 存储器掉电保持的信息，参见第 4 章。

SIMATIC 减计数器指令程序举例

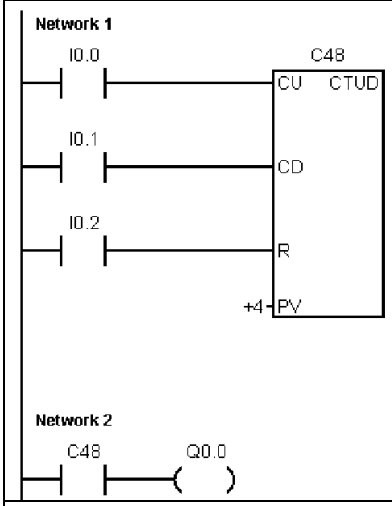



```
Network 1      // 当 I0.1 断开时，减计数器 C1 的当前
               // 值从 3 变到 0。I0.0 的上升沿使 C1 的
               // 当前值递减。I0.1 接通时装载预置值 3。
LD     I0.0
LD     I0.1
CTD    C1, +3
Network 2      // 当计数器 C1 的当前值=0 时，C1 接通。
LD     C1
=      Q0.0
```

时序图

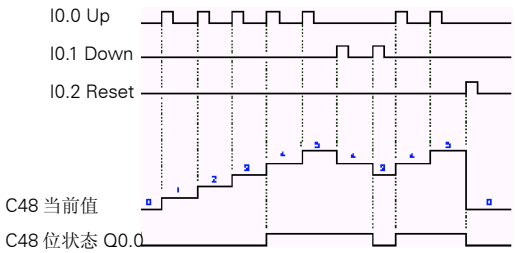


SIMATIC 增/减计数器指令程序举例

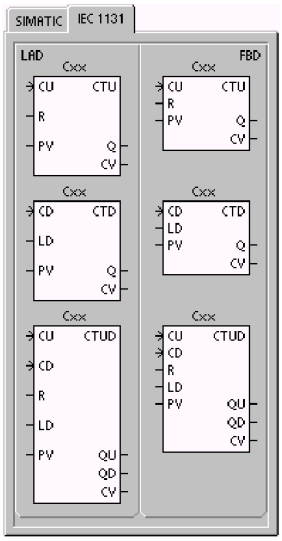



```
Network 1      // I0.0 增计数
               // I0.1 减计数
               // I0.2 将当前值复位为 0
LD     I0.0
LD     I0.1
LD     I0.2
CTUD   C48, + 4
Network 2      // 当当前值≥4 时，将增/减计数器 C48
               // 接通。
LD     C48
=      Q0.0
```

时序图



IEC 计数器指令



增计数器

增计数指令（CTU）在每一个（CU）输入的上升沿从当前值开始增计数，直至预置值（PV）。当当前值（CV）大于等于预置值时，计数器输出位（Q）接通。当复位端（R）使能时，计数器复位。当计数到达预置值时，增计数器停止。

减计数器

减计数器指令（CTD）从预置值开始，在每一个（CD）输入的上升沿减计数。当当前值（CV）等于 0 时，计数器输出位（Q）接通。当装载输入（LD）使能时，计数器复位并且将计数器的当前值设为预置值 PV。当计数值到 0 时，减计数器停止。

增/减计数器

增/减计数器指令（CTUD），在每一个增计数输入（CU）从低到高时增计数；在每一个减计数输入（CD）从低到高时减计数。当当前值等于预置值时，增计数输出（QU）接通。当当前值等于 0 时，减计数输出（QD）接通。当装载输入（LD）使能时，计数器将当前值设为预置值（PV）。类似的，当复位端（R）使能时，计数器复位并且当前值清 0。当计数值达到预置值或者 0 时，计数器停止。

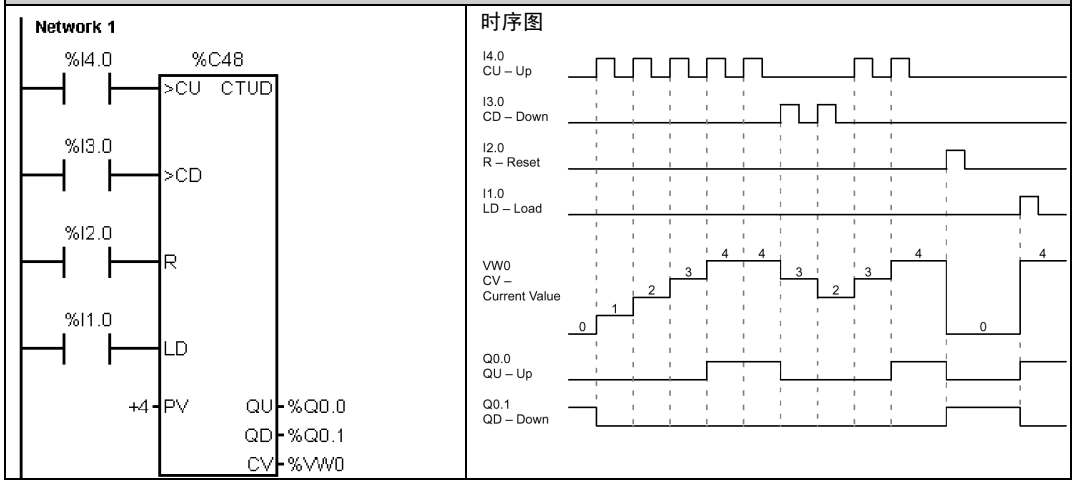
表 6-23　IEC 计数器指令的有效操作数

输入/输出	数据类型	操作数
Cxx	CTU、CTD、CTUD	常数 (C0 到 C255)
CU、CD、LD、R	BOOL	I、Q、V、M、SM、S、T、C、L,能流
PV	INT	IW、QW、VW、MW、SMW、SW、LW、AC、AIW、*VD、*LD、*AC,常数
Q、QU、QD	BOOL	I、Q、V、M、SM、S、L
CV	INT	IW、QW、VW、MW、SW、LW、AC、*VD、*LD、*AC

提示：

由于每一个计数器只有一个当前值，所以不要多次定义同一个计数器。（具有相同标号的增计数器、增/减计数器和减计数器访问相同的当前值。）

IEC 计数器指令程序举例



高速计数器指令

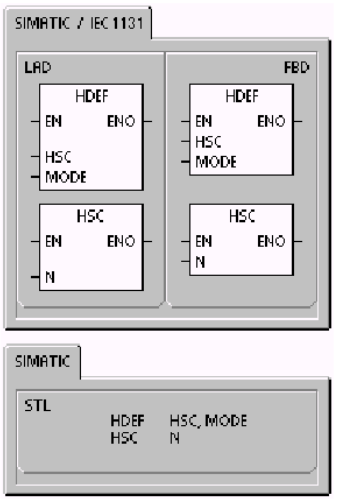
定义高速计数器

定义高速计数器指令（HDEF）为指定的高速计数器（HSCx）选择操作模式。模式的选择决定了高速计数器的时钟方向、启动和复位功能。

对于每一个高速计数器使用一条定义高速计数器指令。

使 ENO＝0 的错误条件：

- 0003（输入点冲突）
- 0004（中断中的非法指令）
- 000A（HSC 重复定义）



高速计数器

高速计数器指令（HSC）在 HSC 特殊存储器位状态的基础上，配置和控制高速计数器。参数 N 指定高速计数器的标号。

高速计数器可以被配置为 12 种模式中的任意一种，参见表 6-25。每一个计数器都有时钟、方向控制、复位、启动的特定输入。对于双相计数器，两个时钟都可以运行在最高频率。在正交模式下，你可以选择一倍速（1x）或者四倍速（4x）计数速率。所有计数器都可以运行在最高频率下而互不影响。

使 ENO＝0 的错误条件：

- 0001（在 HDEF 指令之前执行 HSC 指令）
- 0005（同时执行 HSC/PLS）

表 6-24　高速计数器指令的有效操作数

输入/输出	数据类型	操作数
HSC、MODE	BYTE	常数
N	WORD	常数

应用实例

可以参考资料光盘上应用示例中使用高速计数器的程序。参见应用示例 4 和应用示例 29。

高速计数器用于对 S7-200 扫描速率无法控制的高速事件进行计数。高速计数器的最高计数频率取决于你的 CPU 类型。更多信息请参见附录 A。

提示：

CPU221 和 CPU222 支持 HSC0、HSC3、HSC4 和 HSC5 不支持 HSC1 和 HSC2。CPU224、CPU226 和 CPU226XM 支持 HSC0 到 HSC5 全部六个高速计数器。

一般来说，高速计数器被用作驱动鼓式计时器，该设备有一个安装了增量轴式编码器的轴，以恒定的速度转动。轴式编码器每圈提供一个确定的计数值和一个复位脉冲。来自轴式编码器的时钟和复位脉冲作为高速计数器的输入。

高速计数器装入一组预置值中的第一个值，当前计数值小于当前预置值时，希望的输出有效。计数器设置成在当前值等于预置值和有复位时产生中断。

随着每次当前计数值等于预置值的中断事件的出现，一个新的预置值被装入，并重新设置下一个输出状态。当出现复位中断事件时，设置第一个预置值和第一个输出状态，这个循环又重新开始。

由于中断事件产生的速率远低于高速计数器的计数速率，用高速计数器可实现精确控制，而与 PLC 整个扫描周期的关系不大。采用中断的方法允许在简单的状态控制中用独立的中断程序装入一个新的预置值。（同样的，也可以在一个中断服务程序中，处理所有的中断事件。）

理解不同的高速计数器

对于操作模式相同的计数器，其计数功能是相同的。计数器共有四种基本类型：带有内部方向控制的单相计数器，带有外部方向控制的单相计数器，带有两个时钟输入的双相计数器和 A/B 相正交计数器。注意，并不是所有计数器都能使用每一种模式。你可以使用以下类型：无复位或启动输入，有复位无启动输入或者既有启动又有复位输入。

- 当激活复位输入端时，计数器清除当前值并一直保持到复位端失效。
- 当激活启动输入端时，它允许计数器计数。当启动端失效时，计数器的当前值保持为常数，并且忽略时钟事件。
- 如果在启动输入端无效的同时，复位信号被激活，则忽略复位信号，当前值保持不变。如果在复位信号被激活的同时，启动输入端被激活，当前值被清除。

在使用高速计数器之前，应该用 HDEF（高速计数器定义）指令为计数器选择一种计数模式。使用初次扫描存储器位 SM0.1（该位仅在第一次扫描周期接通，之后断开）来调用一个包含 HDEF 指令的子程序。

高速计数器编程

你可以使用指令向导来配置计数器。向导程序使用下列信息：

计数器的类型和模式、计数器的预置值、计数器的初始值和计数的初始方向。要执行 HSC 指令向导，可以在命令菜单中选择 Tools>Instruction Wizard，然后在向导窗口中选择 HSC 指令。

对高速计数器编程，你必须完成下列基本操作：

- 定义计数器和模式
- 设置控制字节
- 设置初始值
- 设置预置值
- 指定并使能中断服务程序
- 激活高速计数器

定义计数器的模式和输入

使用高速计数器定义指令来定义计数器的模式和输入。

表 6-25 中给出了与高速计数器相关的时钟、方向控制、复位和启动输入点。同一个输入点不能用于两个不同的功能，但是任何一个没有被高速计数器的当前模式使用的输入点，都可以被用作其它用途。例如：如果 HSC0 被定义为模式 1，I0.0 和 I0.2 被占用，I0.1 可以被用于边沿中断或者被 HSC3 占用。

提示：

注意：HSC0 的所有模式都使用 I0.0，HSC4 的所有模式都使用 I0.3，因此在使用这些计数器时，相应的输入点不能用于其它功能。

表 6-25 高速计数器的输入点

模式	描述	输入点			
	HSC0	I0.0	I0.1	I0.2	
	HSC1	I0.6	I0.7	I1.0	I1.1
	HSC2	I1.2	I1.3	I1.4	I1.5
	HSC3	I0.1			
	HSC4	I0.3	I0.4	I0.5	
	HSC5	I0.4			
0	带有内部方向控制的单相计数器	时钟			
1		时钟		复位	
2		时钟		复位	启动
3	带有外部方向控制的单相计数器	时钟	方向		
4		时钟	方向	复位	
5		时钟	方向	复位	启动
6	带有增减计数时钟的双相计数器	增时钟	减时钟		
7		增时钟	减时钟	复位	
8		增时钟	减时钟	复位	启动
9	A/B 相正交计数器	时钟 A	时钟 B		
10		时钟 A	时钟 B	复位	
11		时钟 A	时钟 B	复位	启动

HSC 模式举例

图 6-21 到图 6-25 中给出了每种模式下计数器功能的时序图。

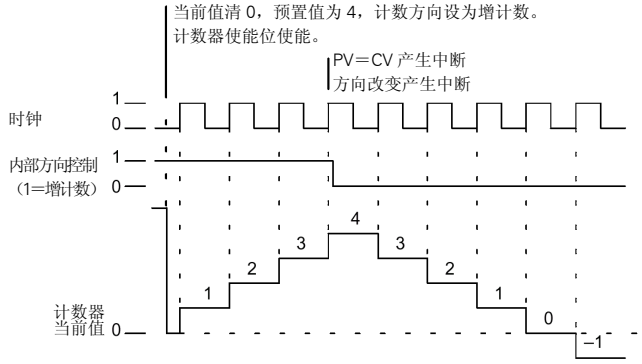


图 6-21　模式 0、1 或者 2 操作实例

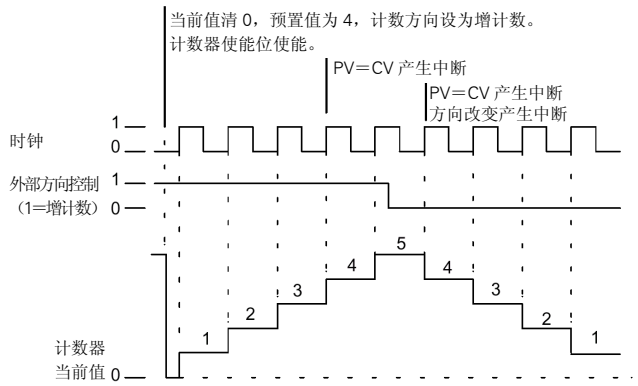


图 6-22　模式 3、4 或者 5 操作实例

当你使用模式 6、7 或者 8 时，如果增时钟输入的上升沿与减时钟输入的上升沿之间的时间间隔小于 0.3 毫秒，高速计数器会把这些事件看作是同时发生的。如果这种情况发生，当前值不变，计数方向指示不变。只要增时钟输入的上升沿与减时钟输入的上升沿之间的时间间隔大于 0.3 毫秒，高速计数器分别捕捉每个事件。在以上两种情况下，都不会有错误产生，计数器保持正确的当前值。

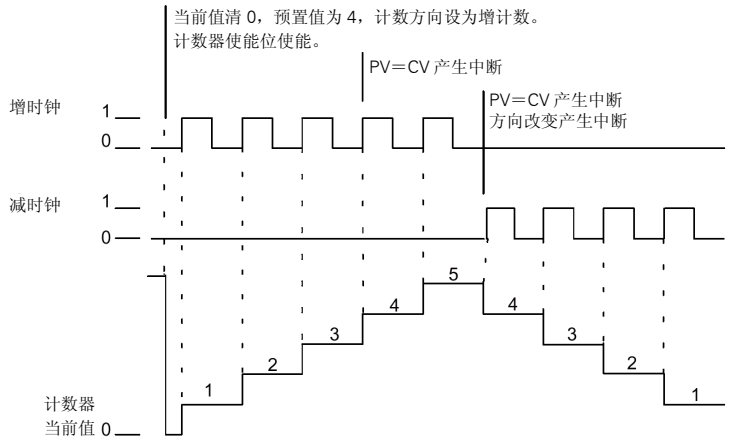


图 6-23　模式 6、7 或者 8 操作实例

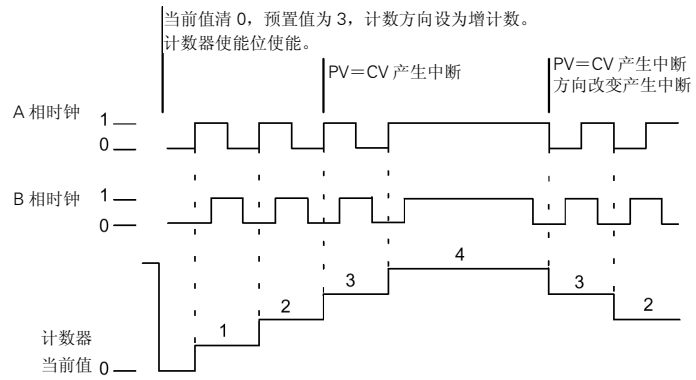


图 6-24　模式 9、10 或者 11 操作实例（一倍速正交模式）

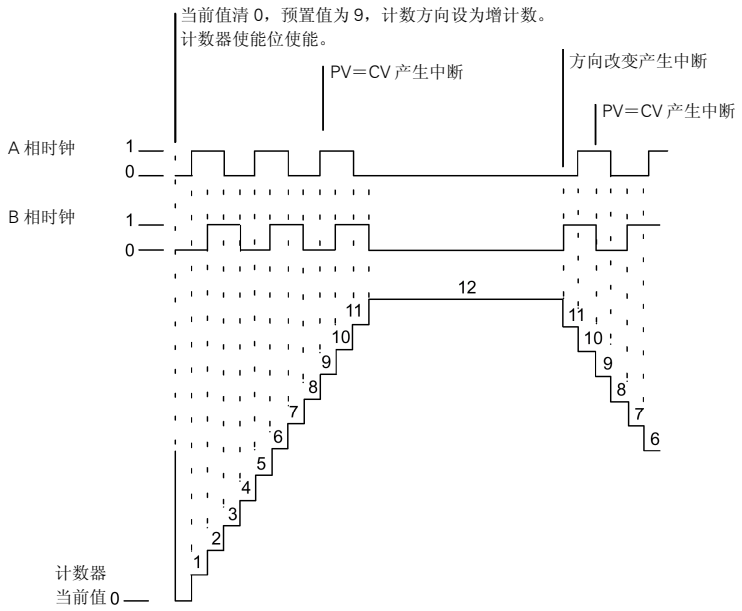


图 6-25　模式 9、10 或者 11 操作实例（四倍速正交模式）

复位和启动操作

如图 6-26 中所示的复位和启动操作适用于使用复位和启动输入的所有模式。在本图中，复位输入和启动输入都被编程为高电平有效。

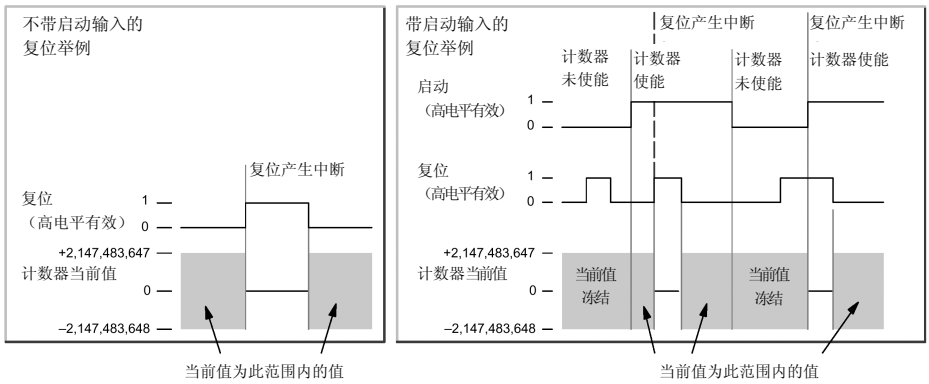


图 6-26　带有或者不带启动输入的复位操作举例

对于高速计数器，有三个控制位用于配置复位和启动信号的有效状态以及选择一倍速或者四倍速计数模式（仅用于正交计数器）。这些位位于各个计数器的控制字节中并且只有在 HDEF 指令执行时使用。在表 6-26 中给出了这些位的定义。

提示：

在执行 HDEF 指令前，必须把这些控制位设定到希望的状态。否则，计数器对计数模式的选择取缺省设置。

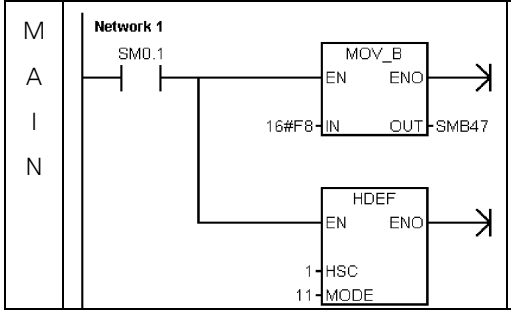
一旦 HDEF 指令被执行，你就不能再更改计数器的设置，除非先进入 STOP 模式。

表 6-26　复位和启动输入的有效电平以及 1x/4x 控制位

HSC0	HSC1	HSC2	HSC4	描述 (仅当 HDEF 执行时使用)
SM37.0	SM47.0	SM57.0	SM147.0	复位有效电平控制位： 0 = 复位高电平有效；1 = 复位低电平有效
—	SM47.1	SM57.1	—	启动有效电平控制位： 0 = 启动高电平有效；1 = 启动低电平有效
SM37.2	SM47.2	SM57.2	SM147.2	正交计数器计数速率选择： 0=4X计数率；1= 1X计数率

1　缺省设置为：复位输入和启动输入高电平有效，正交计数率为四倍速（四倍输入时钟频率）。

高速计数器定义指令程序举例




```
Network 1    //首次扫描：
             //1. 选择启动和复位输入高电平
             //有效，选择四倍速模式。
             //2. 配置HSC1为带有复位和启动输入
             // 的正交模式。
LD      SM0.1
MOVB    16#F8, SMB47
HDEF    1, 11
```

设置控制字节

只有定义了计数器和计数器模式，才能对计数器的动态参数进行编程。每个高速计数器都有一个控制字节，包括以下内容：

- 使能或者禁止计数器
- 控制计数方向（只对模式 0、1 和 2 有效）或者对所有其它模式定义初始化计数方向。
- 装载初始值
- 装载预置值

在执行 HSC 指令时，要检验控制字节和相关的初始值和预置值。表 6-27 中对这些控制位逐一做了说明。

表 6-27　HSC0 到 HSC5 的控制位

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	描述
SM37.3	SM47.3	SM57.3	SM137.3	SM147.3	SM157.3	计数方向控制位： 0=减计数；1＝增计数
SM37.4	SM47.4	SM57.4	SM137.4	SM147.4	SM157.4	向 HSC 中写入计数方向： 0＝不更新；1＝更新计数方向
SM37.5	SM47.5	SM57.5	SM137.5	SM147.5	SM157.5	向 HSC 中写入预置值： 0＝不更新；1＝更新预置值
SM37.6	SM47.6	SM57.6	SM137.6	SM147.6	SM157.6	向 HSC 中写入新的初始值： 0＝不更新；1＝更新初始值
SM37.7	SM47.7	SM57.7	SM137.7	SM147.7	SM157.7	HSC 允计： 0＝禁止 HSC； 1=允许 HSC

设置初始值和预置值

每个高速计数器都有一个 32 位的初始值和一个 32 位的预置值。初始值和预置值都是符号整数。为了向高速计数器装入新的初始值和预置值，必须先设置控制字节，并且把初始值和预置值存入特殊存储器中，然后执行 HSC 指令，从而将新的值传送到高速计数器。表 6-28 中对保存新的初始值和预置值的特殊存储器作了说明。

除去控制字节和新的初始值与预置值保存字节外，每个高速计数器的当前值可利用数据类型 HC（高速计数器当前值）后面跟计数器号（0 到 5）的格式读出。因此，可用读操作直接访问当前值，但写操作只能用 HSC 指令来实现。

表 6-28　HSC0 到 HSC5 的初始值和预置值

要装入的值	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5
初始值	SMD38	SMD48	SMD58	SMD138	SMD148	SMD158
预置值	SMD42	SMD52	SMD62	SMD142	SMD152	SMD162

高速计数器寻址（HC）

如果要指定高速计数器的地址，访问高速计数器的计数值，要使用存储器类型 HC 和计数器号（例如 HC0）。高速计数器的当前值为只读数据，其数据长度为双字（32 位），如图 6-27 所示。

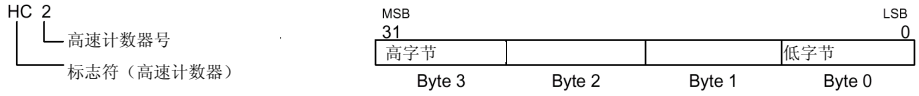


图 6-27　访问高速计数器的当前值

指定中断

所有的计数器模式都会在当前值等于预置值时产生中断，使用外部复位端的计数模式支持外部复位中断。除去模式 0、1 和 2 之外，所有计数器模式支持计数方向改变中断。每种中断条件都可以分别使能或者禁止。要得到关于使用中断的更多信息，参见通迅和中断指令一节。

注意：

当使用外部复位中断时，不要写入初始值，或者是在该中断服务程序中禁止再允许高速计数器，否则会产生一个致命错误。

状态字节

每个高速计数器都有一个状态字节，其中的状态存储位指出了当前计数方向，当前值是否大于或者等于预置值。表 6-29 给出了每个高速计数器状态位的定义。

提示：

只有在执行中断服务程序时，状态位才有效。监视高速计数器状态的目的是使其它事件能够产生中断以完成更重要的操作。

表 6-29　HSC0 到 HSC5 的状态位

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	描述
SM36.0	SM46.0	SM56.0	SM136.0	SM46.0	SM156.0	不用
SM36.1	SM46.1	SM56.1	SM136.1	SM46.1	SM156.1	不用
SM36.2	SM46.2	SM36.2	SM136.2	SM46.2	SM156.2	不用
SM36.3	SM46.3	SM56.3	SM136.3	SM46.3	SM156.3	不用
SM36.4	SM46.4	SM56.4	SM136.4	SM46.4	SM156.4	不用
SM36.5	SM46.5	SM56.5	SM136.5	SM46.5	SM156.5	当前计数方向状态位: 0=减计数 1=增计数
SM36.6	SM46.6	SM56.6	SM136.6	SM146.6	SM156.6	当前值等于预置值状态位: 0=不等; 1=相等
SM36.7	SM46.7	SM56.7	SM136.7	SM146.7	SM156.7	当前值大于预置值状态位: 0=小于等于; 1=大于

高速计数器的初始化步骤举例

以下以 HSC1 为例，对初始化和操作的步骤进行描述。在初始化描述中，假定 S7-200 已经置成 RUN 模式。因此，首次扫描标志位为真。如果不是这种情况，请记住在进入 RUN 模式之后，对每一个高速计数器的 HDEF 指令只能执行一次。对一个高速计数器第二次执行 HDEF 指令会引起运行错误，而且不能改变第一次执行 HDEF 指令时对计数器的设置。

提示：

虽然下列步骤描述了如何分别改变计数方向、初始值和预置值，但你完全可以在同一操作步骤中对全部或者任意参数组合进行设置，只要设置正确的 SMB47 然后执行 HSC 指令即可。

初始化模式 0、1 或 2

HSC1 为内部方向控制的单相增/减计数器（模式 0、1 或 2），初始化步骤如下：

1. 用初次扫描存储器位（SM0.1=1）调用执行初始化操作的子程序。由于采用了这样的子程序调用，后续扫描不会再调用这个子程序，从而减少了扫描时间，也提供了一个结构优化的程序。
2. 初始化子程序中，根据所希望的控制操作对 SMB47 置数。例如：
   SMB47=16#F8 *产生如下的结果：*
   允许计数
   写入新的初始值
   写入新的预置值
   置计数方向为增
   置启动和复位输入为高电平有效
3. 执行 HDEF 指令时，HSC 输入置 1，MODE 输入置 0（无外部复位或启动）或置 1（有外部复位和无启动）或置 2（有外部复位和启动）。
4. 将所希望的初始值装入 SMD48（双字）中，若装入 0，则清除 SMD48。
5. 将所希望的预置值装入 SMD52（双字）中。
6. 为了捕获当前值（CV）等于预置值（PV）中断事件，编写中断子程序，并指定 CV=PV 中断事件（事件号 13）调用该中断子程序。参看本章中断一节，以了解中断处理的细节。
7. 为了捕获外部复位事件，编写中断子程序，并指定外部复位中断事件（事件号 15）调用该中断子程序。
8. 执行全局中断允许指令（ENI）来允许 HSC 中断。
9. 执行 HSC 指令，使 S7-200 对 HSC1 编程。
10. 退出子程序。

初始化模式 3、4 或 5

HSC1 为外部方向控制的单相增/减计数器（模式 3、4 或 5），初始化步骤如下：

1. 用初次扫描存储器位（SM0.1=1）调用执行初始化操作的子程序。由于采用了这样的子程序调用，后续扫描不会再调用这个子程序，从而减少了扫描时间，也提供了一个结构优化的程序。
2. 初始化子程序中，根据所希望的控制操作对 SMB47 置数。例如：
   SMB47=16#F8 *产生如下的结果：*
   允许计数
   写入新的初始值
   写入新的预置值
   置 HSC 的初始计数方向为增
   置启动和复位输入为高电平有效
3. 执行 HDEF 指令时，HSC 输入置 1，MODE 输入置 3（无外部复位或启动）或置 4（有外部复位和无启动）或置 5（有外部复位和启动）。

4. 将所希望的初始值装入 SMD48（双字）中，若装入 0，则清除 SMD48。
5. 将所希望的预置值装入 SMD52（双字）中。
6. 为了捕获当前值（CV）等于预置值（PV）中断事件，编写中断子程序，并指定 CV=PV 中断事件（事件号 13）调用该中断子程序。参看本章中断一节，以了解中断处理的细节。
7. 为了捕获计数方向改变中断事件，编写中断子程序，并指定计数方向改变中断事件（事件号 14）调用该中断子程序。
8. 为了捕获外部复位中断事件，编写中断子程序，并指定外部复位中断事件（事件号 15）调用该中断子程序。
9. 执行全局中断允许指令（ENI）来允许 HSC 中断。
10. 执行 HSC 指令，使 S7-200 对 HSC1 编程。
11. 退出子程序。

初始化模式 6、7 或 8

HSC1 为具有增/减两种时钟的双相增/减计数器（模式 6、7 或 8），初始化步骤如下：

1. 用初次扫描存储器位（SM0.1=1）调用执行初始化操作的子程序。由于使用了这样的子程序调用，后续的扫描不会再调用这个子程序，从而减少了扫描时间，也提供了一个结构优化的程序。
2. 初始化子程序中，根据所希望的控制操作对 SMB47 置数。例如：
   SMB47=16#F8 *产生如下的结果：*
   允许计数
   写入新的初始值
   写入新的预置值
   置 HSC 的初始计数方向为增
   置启动和复位输入为高电平有效
3. 执行 HDEF 指令时，HSC 输入置 1，MODE 输入置 6（无外部复位或启动）或置 7（有外部复位和无启动）或置 8（有外部复位和启动）。
4. 将所希望的初始值装入 SMD48（双字）中，若装入 0，则清除 SMD48。
5. 将所希望的预置值装入 SMD52（双字）中。
6. 为了捕获当前值（CV）等于预置值（PV）中断事件，编写中断子程序，并指定 CV=PV 中断事件（事件号 13）调用该中断子程序。参看本章中断一节，以了解中断处理的细节。
7. 为了捕获方向改变中断事件，编写中断子程序，并指定计数方向改变中断事件（事件号 14）调用该中断子程序。
8. 为了捕获外部复位中断事件，编写中断子程序，并指定外部复位中断事件（事件号 15）调用该中断子程序。
9. 执行全局中断允许指令（ENI）来允许 HSC1 中断。
10. 执行 HSC 指令，使 S7-200 对 HSC1 编程。

11. 退出子程序。

初始化模式 9、10 或 11

HSC1 为 A/B 相正交计数器（模式 9、10 或 11），初始化步骤如下：

1. 用初次扫描存储器位（SM0.1=1）调用执行初始化操作的子程序。由于采用了这样的子程序调用，后续的扫描不会再调用这个子程序，从而减少了扫描时间，也提供了一个结构优化的程序。
2. 初始化子程序中，根据所希望的控制操作对 SMB47 置数。
   例如（1X 计数方式）：
   SMB47=16#FC *产生如下的结果：*
   允许计数
   写入新的初始值
   写入新的预置值
   置 HSC 的初始计数方向为增
   置启动和复位输入为高电平有效
   例如（4X 计数方式）：
   SMB47=16#F8 *产生如下的结果：*
   允许计数
   写入新的初始值
   写入新的预置值
   置 HSC 的初始计数方向为增
   置启动和复位输入为高电平有效
3. 执行 HDEF 指令时，HSC 输入置 1，MODE 输入置 9（无外部复位或启动）或置 10（有外部复位和无启动）或置 11（有外部复位和启动）。
4. 将所希望的初始值装入 SMD48（双字）中，若装入 0，则清除 SMD48。
5. 将所希望的预置值装入 SMD52（双字）中。
6. 为了捕获当前值（CV）等于预置值（PV）中断事件，编写中断子程序，并指定 CV=PV 中断事件（事件号 13）调用该子程序。参见本章中断一节，以了解中断处理的细节。
7. 为了捕获计数方向改变中断事件，编写中断子程序，并指定计数方向改变中断事件（事件号 14）调用该中断子程序。
8. 为了捕获外部复位中断事件，编写中断子程序，并指定外部复位中断事件（事件号 15）调用该中断子程序。
9. 执行全局中断允许指令（ENI）来允许 HSC1 中断。
10. 执行 HSC 指令，使 S7-200 对 HSC1 编程。
11. 退出子程序。

改变模式 0、1 或 2 的计数方向

对具有内部方向控制（模式 0、1 或 2）的单相计数器 HSC1，改变其计数方向的步骤如下：

1. 向 SMB47 写入所需的计数方向：
   SMB47=16#90　　允许计数
   置 HSC 计数方向为减。
   SMB47=16#98　　允许计数
   置 HSC 计数方向为增
2. 执行 HSC 指令，使 S7-200 对 HSC1 编程。

写入新的初始值（任何模式下）

以下步骤描述了如何改变 HSC1 的初始值（任何模式下）：

在改变初始值时，迫使计数器处于非工作状态，此时计数器不再计数，也不产生中断。

1. 向 SMB47 写入新的初始值的控制位：
   SMB47=16#C0　　允许计数
   写入新的初始值
2. 向 SMD48（双字）写入所希望的初始值（若写入 0，则清除）。
3. 执行 HSC 指令，使 S7-200 对 HSC1 编程。

写入新的预置值（任何模式下）

以下步骤描述了如何改变 HSC1 的预置值（任何模式下）：

1. 向 SMB47 写入允许写入新的预置值的控制位：
   SMB47=16#A0　　允许计数
   写入新的预置值
2. 向 SMD52（双字）写入所希望的预置值。
3. 执行 HSC 指令，使 S7-200 对 HSC1 编程。

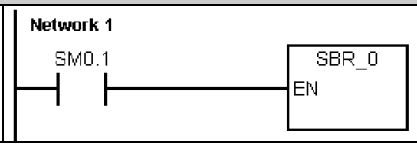
禁止 HSC（任何模式下）

以下步骤描述了如何禁止 HSC1 高速计数器（任何模式下）：

1. 写入 SMB47 以禁止计数：
   SMB47=16#00　　禁止计数
2. 执行 HSC 指令，以禁止计数。

高速计数器指令程序举例：

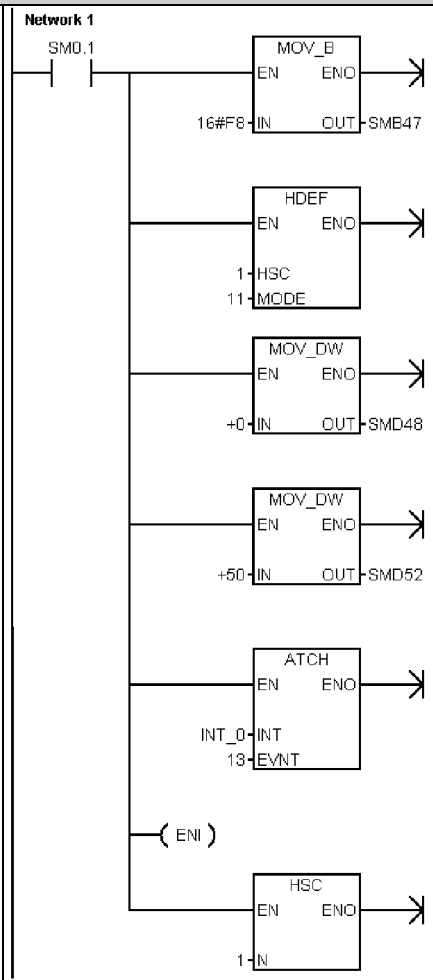
MAIN




```
Network1  //在首次扫描，调用 SBR_0
LD    SM0.1
CALL  SBR_0
```

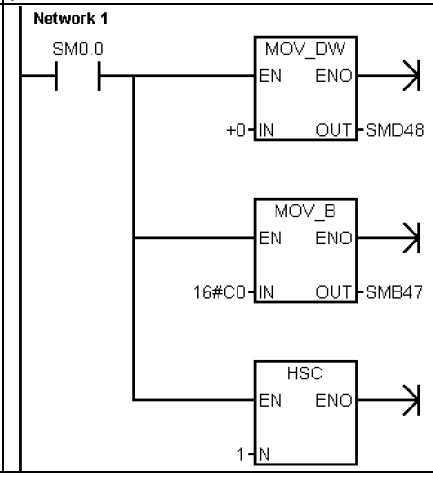
高速计数器指令程序举例：

SBR0




```
Network 1   //在首次扫描，配置 HSC1：
            //1. 使能计数器。
            //一写初始值。
            //一写预置值。
            //一设初始方向为增计数。
            //一选择启动和复位输入高电平有效。
            //一选择 4 倍速模式。
            //2. 配置 HSC1 为带启动和复位输入的正交模式。
            //3. 清除 HSC1 的初始值。
            //4. 置 HSC1 的预置值为 50。
            //5. 当 HSC1 的当前值=预置值时，执行 INT_0。
            //6. 全局中断允许。
            //7. 执行 HSC1
LD      SM0.1
MOVB    16#F8，SMB47
HDEF    1，11
MOVD    +0，SMD48
MOVD    +50，SMD52
ATCH    INT_0，13
ENI
HSC     1
```

INT0

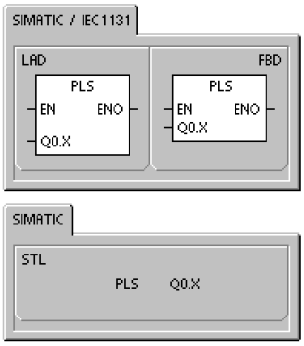



```
Network1    //执行 HSC1
            //1. 清除 HSC1 的初始值。
            //2. 选择写入新的初始值和 HSC1 使能。
LD      SM0.0
MOVD    +0，SMD48
MOVB    16#C0，SMB47
HSC     1
```

脉冲输出指令

运动控制

脉冲输出指令（PLS）用于在高速输出点（Q0.0 和 Q0.1）上实现脉冲串输出（PTO）和脉宽调制（PWM）功能。你可以使用运动控制向导来配置脉冲输出。



PTO 可以输出一串脉冲（占空比 50%），用户可以控制脉冲的周期和个数。

PWM 可以输出一串占空比可调的脉冲，用户可以控制脉冲的周期和脉宽。

S7-200 有两个 PTO/PWM 发生器，它们可以产生一个高速脉冲串或者一个脉宽调制波形。一个发生器是数字输出点 Q0.0，另一个发生器是数字输出点 Q0.1。特殊存储器中为每一个发生器存储下列数据：一个控制字节（8 位），一个计数值（32 位无符号数）和一个周期或脉宽值（16 位无符号数）。

PTO/PWM 发生器与过程映象寄存器共用 Q0.0 和 Q0.1。当在 Q0.0 或 Q0.1 上激活 PTO 或者 PWM 功能时，PTO/PWM 发生器对输出拥有控制权，同时普通输出点功能被禁止。输出波形不受过程映象区状态、输出点强制值或者立即输出指令执行的影响。当不使用 PTO/PWM 发生器功能时，对输出点的控制权交回到过程映象寄存器。过程映象寄存器决定输出波形的起始和结束状态，以高低电平产生波形的启动和结束。

表 6-30 脉冲输出指令的有效操作数

输入/输出	数据类型	操作数
Q0.X	WORD	常数：0（=Q0.0）或者 1（=Q0.1）

提示：

在使能 PTO 或者 PWM 操作之前，将 Q0.0 和 Q0.1 过程映象寄存器清 0。所有控制位、周期、脉宽和脉冲计数值的缺省值均为 0。**PTO/PWM 的输出负载至少为 10%的额定负载，才能提供陡直的上升沿和下降沿。**

应用示例

请参阅资料光盘上应用示例中关于 PTO/PWM 操作和 PLS 指令的使用。见应用示例 7、22、23、30 和 50。

脉冲串操作（PTO）

PTO 按照给定的脉冲个数和周期输出一串方波（占空比 50%）。（如图 6-28 所示。）PTO 可以产生单段脉冲串或者多段脉冲串（使用脉冲包络）。你可以指定其脉冲个数和周期（以微秒或者毫秒为增量）：

- 脉冲个数：1 到 4，294，967，295
- 周期： 50μs 到 65，535μs 或者 2ms 到 65，535ms。

如果设定的周期为奇数，会引起占空比失真。

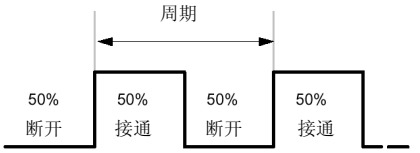


图 6-28　脉冲串输出（PTO）

表 6-31 中是对脉冲计数和周期的限定。

表 6-31　PTO 功能的脉冲个数及周期

脉冲个数/周期	结果
周期<2 个时间单位	将周期缺省地设定为 2 个时间单位
脉冲个数=0	将脉冲个数缺省地设定为 1 个脉冲

PTO 功能允许脉冲串“链接”或者“排队”。当当前脉冲串输出完成时，会立即开始输出一个新的脉冲串。这保证了多个输出脉冲串之间的连续性。

PTO 脉冲串的单段管线

在单段管线模式，需要为下一个脉冲串更新特殊寄存器。一旦启动了起始 PTO 段，就必须按照第二个波形的要求改变特殊寄存器，并再次执行 PLS 指令。第二个脉冲串的属性在管线中一直保持到第一个脉冲串发送完成。在管线中一次只能存储一段脉冲串的属性。当第一个脉冲串发送完成时，接着输出第二个波形，此时管线可以用于下一个新的脉冲串。重复这个过程可以再次设定下一个脉冲串的特性。

除去以下两种情况之外，脉冲串之间可以作到平滑转换：

时间基准发生了变化或者在利用 PLS 指令捕捉到新脉冲之前，启动的脉冲串已经完成。

PTO 脉冲串的多段管线

在多段管线模式，CPU 自动从 V 存储器区的包络表中读出每个脉冲串的特性。在该模式下，仅使用特殊存储器区的控制字节和状态字节。选择多段操作，必须装入包络表在 V 存储器中的起始地址偏移量（SMW168 或 SMW178）。时间基准可以选择微秒或者毫秒，但是，在包络表中的所有周期值必须使用同一个时间基准，而且在包络正在运行时不能改变。执行 PLS 指令来启动多段操作。

每段记录的长度为 8 个字节，由 16 位周期值、16 位周期增量值和 32 位脉冲个数值组成。表 6-32 中给出了包络表的格式。你可以通过编程的方式使脉冲的周期自动增减。在周期增量处输入一个正值将增加周期；输入一个负值将减少周期；输入 0 将不改变周期。

当 PTO 包络执行时，当前启动的段的编号保存在 SMB166（或 SMB176）。

表 6-32 多段 PTO 操作的包络表格式

从包络表开始的字节偏移	包络段数	描述
0		段数（1 到 255[1]）
1	#1	初始周期（2 到 65535 时间基准单位）
3		每个脉冲的周期增量（有符号值）（-32768 到 32767 时间基准单位）
5		脉冲数（1 到 4294967295）
9	#2	初始周期（2 到 65535 时间基准单位）
11		每个脉冲的周期增量（有符号值）（-32768 到 32767 时间基准单位）
13		脉冲数（1 到 4294967295）
: :	#3 :	: :

[1] 输入 0 作为脉冲串的段数会产生一个非致命错误，将不产生 PTO 输出。

脉宽调制（PWM）

PWM 产生一个占空比变化周期固定的脉冲输出，如图 6-29 所示。你可以以微秒或者毫秒为单位指定其周期和脉冲宽度。

- 周期：50μs 到 65,535μs 或者 2ms 到 65,535ms
- 脉宽：0μs 到 65,535μs 或者 0ms 到 65,535ms。

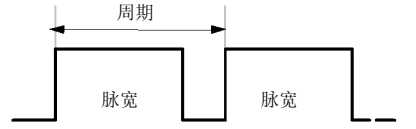


图 6-29 脉宽调制（PWM）

如表 6-33 中所示，设定脉宽等于周期（使占空比为 100%），输出连续接通。设定脉宽等于 0（使占空比为 0%），输出断开。

表 6-33 脉宽、周期和 PWM 功能的执行结果。

脉宽/周期	结果
脉宽≥周期值	占空比为 100%：输出连续接通。
脉宽=0	占空比为 0%：输出断开。
周期<2 个时间单位	将周期缺省地设定为 2 个时间单位。

有两个方法改变 PWM 波形的特性：同步更新和异步更新。

- 同步更新：如果不需要改变时间基准，就可以进行同步更新。利用同步更新，波形特性的变化发生在周期边沿，提供平滑转换。
- 异步更新：PWM 的典型操作是当周期时间保持常数时变化脉冲宽度。所以，不需要改变时间基准。但是，如果需要改变 PTO/PWM 发生器的时间基准，就要使用异步更新。异步更新会造成 PTO/PWM 功能被瞬时禁止，和 PWM 波形不同步。这会引起被控设备的振动。由于这个原因，建议采用 PWM 同步更新。选择一个适合于所有周期时间的时间基准。

控制字节中的 PWM 更新方法位（SM67.4 或 SM77.4）用来指定更新类型。执行 PLS 指令激活这些改变。注意，如果改变了时间基准，会产生一个异步更新，而和这些控制位无关。

提示：

控制字节中的 PWM 更新方式位（SM67.4 或 SM77.4）用于指定更新方式。当 PLS 指令执行时变化生效。

如果改变了时间基准，会产生一个异步更新，而与 PWM 更新方式位的状态无关。

使用 SM 来配置和控制 PTO/PWM 操作

PLS 指令会从特殊存储器 SM 中读取数据，使程序按照其存储值控制 PTO/PWM 发生器。SMB67 控制 PTO0 或者 PWM0，SMB77 控制 PTO1 或者 PWM1。表 6-34 对用于控制 PTO/PWM 操作的存储器给出了描述。你可以使用表 6-35 作为一个快速参考，用其中的数值作为 PTO/PWM 控制存储器的值来实现需要的操作。你可以通过修改 SM 存储区（包括控制字节），然后执行 PLS 指令来改变 PTO 或 PWM 波形的特性。你可以在任意时刻禁止 PTO 或者 PWM 波形，方法为：首先将控制字节中的使能位（SM67.7 或者 SM77.7）清 0，然后执行 PLS 指令。

PTO 状态字节中的空闲位（SM66.7 或者 SM76.7）标志着脉冲串输出完成。另外，在脉冲串输出完成时，你可以执行一段中断服务程序。（参考中断指令和通讯指令中的描述）。如果你使用多段操作，可以在整个包络表完成之后执行中断服务程序。

下列条件使 SM66.4（或 SM76.4）或 SM66.5（或 SM76.5）置位：

- 如果周期增量使 PTO 在许多脉冲后产生非法周期值，会产生一个算术溢出错误，这会终止 PTO 功能并在状态字节中将增量计算错误位（SM66.4 或者 SM76.4）置 1，PLC 的输出变为由映象寄存器控制。
- 如果要手动终止一个正在进行中的 PTO 包络，要把状态字节中的用户终止位（SM66.5 或 SM76.5）置 1。
- 当管线满时，如果试图装载管线，状态存储器中的 PTO 溢出位（SM66.6 或者 SM76.6）置 1。如果想用该位检测序列的溢出，必须在检测到溢出后手动清除该位。当 CPU 切换至 RUN 模式时，该位被初始化为 0。

提示：

如果要装入新的脉冲数（SMD72 或 SMD82）、脉冲宽度（SMW70 或 SMW80）或周期（SMW68 或 SMW78），应该在执行 PLS 指令前装入这些值和控制寄存器。如果要使用多段脉冲串操作，在使用 PLS 指令前也需要装入包络表的起始偏移量（SMW168 或 SMW178）和包络表的值。

表 6-34　PTO/PWM 控制寄存器的 SM 标志

Q0.0	Q0.1	状态字节
SM66.4	SM76.4	PTO 包络由于增量计算错误而终止 0=无错误；　1=终止
SM66.5	SM76.5	PTO 包络由于用户命令而终止 0=无错误；　1=终止
SM66.6	SM76.6	PTO 管线上溢/下溢 0=无溢出；　1=上溢/下溢
SM66.7	SM76.7	PTO 空闲　0=执行中；　1=PTO 空闲
Q0.0	**Q0.1**	**控制字节**
SM67.0	SM77.0	PTO/PWM 更新周期值　0=不更新；　1=更新周期值
SM67.1	SM77.1	PWM 更新脉冲宽度值　0=不更新；　1=脉冲宽度值
SM67.2	SM77.2	PTO 更新脉冲数　0=不更新；　1=更新脉冲数
SM67.3	SM77.3	PTO/PWM 时间基准选择　0=1μs/时基；　1=1ms/时基
SM67.4	SM77.4	PWM 更新方法：　0=异步更新；　1=同步更新
SM67.5	SM77.5	PTO 操作：　0=单段操作；　1=多段操作
SM67.6	SM77.6	PTO/PWM 模式选择　0=选择 PTO；　1=选择 PWM
SM67.7	SM77.7	PTO/PWM 允许　0=禁止；　1=允许
Q0.0	**Q0.1**	**其它 PTO/PWM 寄存器**
SMW68	SMW78	PTO/PWM 周期值（范围：2 到 65535）
SMW70	SMW80	PWM 脉冲宽度值（范围：0 到 65535）
SMD72	SMD82	PTO 脉冲计数值（范围：1 到 4294967295）
SMB166	SMB176	进行中的段数（仅用在多段 PTO 操作中）
SMW168	SMW178	包络表的起始位置，用从 V0 开始的字节偏移表示 （仅用在多段 PTO 操作中）

表 6-35　PTO/PWM 控制字节参考

控制寄存器（16 进制）	执行 PLS 指令的结果							
	允许	模式选择	PTO 段操作	PWM 更新方法	时基	脉冲数	脉冲宽度	周期
16#81	Yes	PTO	单段		1μs /周期			装入
16#84	Yes	PTO	单段		1μs /周期	装入		
16#85	Yes	PTO	单段		1μs /周期	装入		装入
16#89	Yes	PTO	单段		1ms/周期			装入
16#8C	Yes	PTO	单段		1ms/周期	装入		
16#8D	Yes	PTO	单段		1ms/周期	装入		装入
16#A0	Yes	PTO	单段		1μs /周期			
16#A8	Yes	PTO	单段		1ms/周期			
16#D1	Yes	PWM		同步	1μs /周期			装入
16#D2	Yes	PWM		同步	1μs /周期		装入	
16#D3	Yes	PWM		同步	1μs /周期		装入	装入
16#D9	Yes	PWM		同步	1ms/周期			装入
16#DA	Yes	PWM		同步	1ms/周期		装入	
16#DB	Yes	PWM		同步	1ms/周期		装入	装入

计算包络表的值

PTO/PWM 发生器的多段管线功能在许多应用中非常有用，尤其在步进电机控制中。

例如：你可以用带有脉冲包络的 PTO 来控制一台步进电机，来实现一个简单的加速、匀速和减速过程或者一个由最多 255 段脉冲包络组成的复杂过程，而其中每一段包络都是加速、匀速或者减速操作。

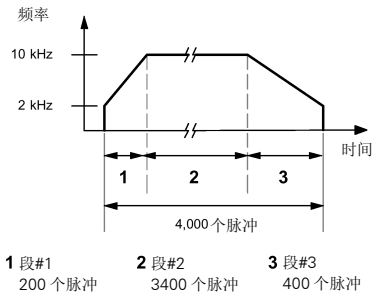


图 6-30　频率/时间图

图 6-30 中的示例给出的包络表值要求产生一个输出波形包括三段：步进电机加速（第一段）；步进电机匀速（第二段）和步进电机减速（第三段）。

对该例，假定需要 4000 个脉冲达到要求的电机转动数，启动和结束频率是 2kHz，最大脉冲频率是 10kHz。由于包络表中的值是用周期表示的，而不是用频率，需要把给定的频率值转换成周期值。所以，启动和结束的脉冲周期为 500μs，最高频率的对应周期为 100μs。在输出包络的加速部分，要求在 200 个脉冲左右达到最大脉冲频率。也假定包络的减速部分，在 400 个脉冲完成。

在该例中，使用一个简单公式计算 PTO/PWM 发生器用来调整每个脉冲周期所使用的周期增量值：

给定段的周期增量=｜ECT-ICT｜/Q
ECT=该段结束周期时间
ICT=该段初始化周期时间
Q=该段的脉冲数量

利用这个公式，加速部分（第 1 段）的周期增量是-2。相似地，减速部分（第 3 段）的周期增量是 1。由于第 2 段是恒速控制，因此，该段的周期增量是 0。

假定包络表存放在从 VB500 开始的 V 存储器区，表 6-36 给出了产生所要求波形的值。

表 6-36　包络值表

V 存储器地址	值
VB500	3（总段数）
VW501	500（初始周期－段#1）
VW503	-2（周期增量－段#1）
VW505	200（脉冲数－段#1）
VW509	100（初始周期－段#2）
VW511	0（周期增量－段#2）
VW513	3400（脉冲数－段#2）
VW517	100（初始周期－段#3）
VW519	1（周期增量－段#3）
VD521	400（脉冲数－段#3）

该表的值可以在用户程序中用指令放在 V 存储器中。一种方法是在数据块中定义包络表的值。

段的最后一个脉冲的周期在包络中不直接指定，但必须计算出来（除非周期增量是 0）。

如果在段之间需要平滑转换，知道段的最后一个脉冲的周期是有用的。计算段的最后一个脉冲周期的公式是：

段的最后一个脉冲的周期时间=ICT+（DEL*（Q-1）） ICT=该段初始化周期时间 DEL=该段的增量周期时间 Q=该段的脉冲数量

作为介绍，上面的简例是有用的，实际应用可能需要更复杂的波形包络。记住：

- 周期增量只能以微秒数或毫秒数指定
- 周期的修改在每个脉冲上进行

这两项的影响使对于一个段的周期增量的计算可能需要叠代方法。对于结束周期值或给定段的脉冲个数，可能需要作调整。

在确定校正包络表值的过程中，包络段的持续时间很有用。按照下面的公式可以计算完成一个包络段的时间长短：

包络段的持续时间=Q*（ICT+（（DEL/2）*（Q-1））） Q=该段的脉冲数量 ICT=该段的初始化周期时间 DEL=该段的增量周期时间

PWM 输出操作实例

提示：

以下对于 PWM 初始化和操作步骤的描述，建议使用首次扫描标志位（SM0.1）来对脉冲输出进行初始化。用首次扫描标志位来调用初始化子程序减少了扫描时间，因为在接下来的扫描周期中不再调用该子程序。（首次扫描标志位只在转入 RUN 模式的第一个周期内置位。）但是，你的应用有可能要求你在某种条件下初始化（或者重新初始化）脉冲输出。在这种情况下，你可以使用其它条件来调用初始化子程序。

初始化 PWM 输出

通常，你使用一个子程序来初始化 PWM 脉冲输出。你在主程序中调用初始化子程序。用首次扫描标志位（SM0.1）将 PWM 使用的输出点清 0 并且调用子程序来完成初始化操作。由于采用了这样的子程序调用，后续扫描不会再调用该子程序，从而减少了扫描时间，也使程序更加结构化。

在主程序中调用初始化子程序之后，在初始化子程序中要使用以下步骤创建控制逻辑，配置 Q0.0 的脉冲输出。

1. 把 16#D3 送入 SMB67，使 PWM 以微秒为增量单位（或 16#DB 使 PWM 以毫秒为增量单位）。用这些值设置控制字节的目的是：允许 PTO/PWM 功能，选择 PWM 操作，选择以微秒或毫秒为增量单位，设置更新脉宽和周期值。
2. 向 SMW68（字）写入所希望的周期值。
3. 向 SMW70（字）写入所希望的脉宽。
4. 执行 PLS 指令，以使 S7-200 对 PTO/PWM 发生器编程。
5. 向 SMB67 写入 16#D2 选择以微秒为增量单位(或 16#DA 选择以毫秒为增量单位)，这复位了控制字节中的更新周期值位但允许改变脉宽。可以装入一个新的脉宽值，然后不需要修改控制字节就执行 PLS 指令。
6. 退出子程序。

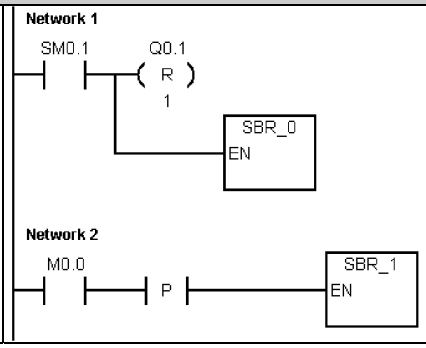
修改 PWM 输出的脉冲宽度

如果在 SMB67 中装入 16#D2 或者 16#DA（见以上第 5 步），你可以编制子程序来改变 Q0.0 输出脉冲的脉宽。在创建了子程序的调用之后，使用以下步骤来创建控制逻辑改变脉宽：

1. 把所需脉宽装入 SMW70（字）中。
2. 执行 PLS 指令，使 S7-200 对 PTO/PWM 发生器编程。
3. 退出子程序。

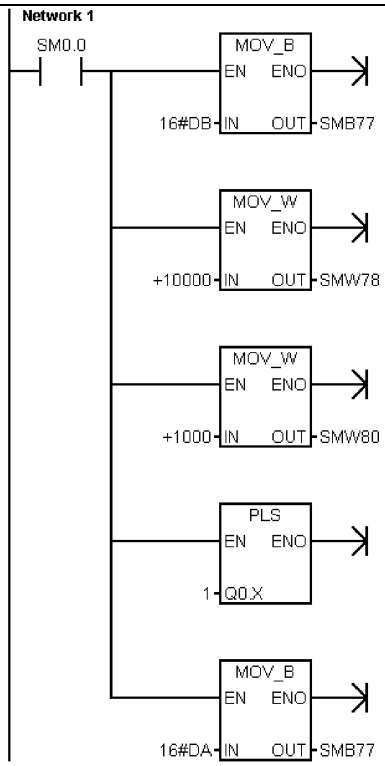
脉宽调制（PWM）程序举例：

MAIN



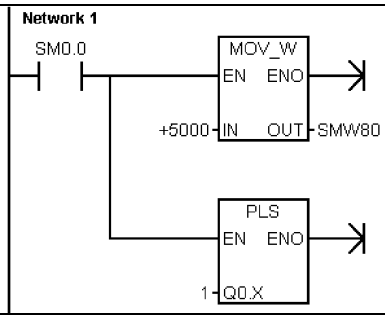

```
Network1  //首次扫描复位映象寄存器并调用 SBR_0。
LD      SM0.1
R       Q0.1，1
CALL    SBR_0
Network2  //当脉宽达到占空比 50%时，将 M0.0 置位。
LD      M0.0
EU
CALL    SBR_1
```

SBR 0



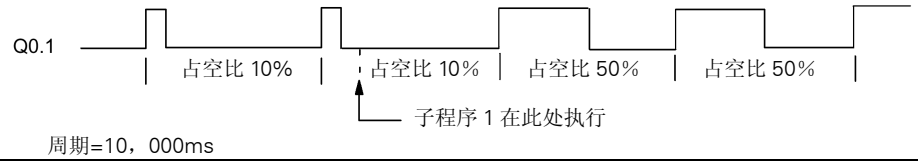

```
Network 1   //子程序 0 开始：
            //1.  设置控制字节。
            //－选择 PWM 操作。
            //－选择 ms 增量和同步更新
            //－允许装载脉冲宽度和周期。
            //－使能 PWM 功能。
            //2.  设定周期为 10，000ms。
            //3.  设定脉宽为 1，000ms。
            //4.  启动 PWM 操作：PLS1=>Q0.1。
            //5.  为以后的脉宽改变预装控制字节。
LD          SM0.0
MOVB        16#DB，SMB77
MOVW        +10000，SMW78
MOVW        +1000，SMW80
PLS         1
MOVB        16#DA，SMB77
```

SBR 1




```
Network1    //子程序 1 开始：
            //设定脉冲宽度为 5000ms。
            //修改脉宽。
LD          SM0.0
MOVW        +5000，SMW80
PLS         1
```

时序图



PTO 输出操作实例

提示：

以下对于 PTO 初始化和操作步骤的描述，建议使用首次扫描标志位（SM0.1）来对脉冲输出进行初始化。用首次扫描标志位来调用初始化子程序减少了扫描时间，因为在接下来的扫描周期中不再调用该子程序。（首次扫描标志位只在转入 RUN 模式的第一个周期内置位。）但是，你的应用有可能要求你在某种条件下初始化（或者重新初始化）脉冲输出。在这种情况下，你可以使用其它条件来调用初始化子程序。

初始化单段 PTO 输出

通常，你使用一个子程序来初始化 PTO 脉冲输出。你在主程序中调用初始化子程序。用首次扫描标志位（SM0.1）将 PTO 使用的输出点清 0 并且调用子程序来完成初始化操作。由于采取了这样的子程序调用，后续扫描不会再调用该子程序，从而减少了扫描时间，也使程序更加结构化。

在主程序中调用初始化子程序之后，在初始化子程序中要使用以下步骤创建控制逻辑，配置 Q0.0 的脉冲输出：

1. 把 16#85 送入 SMB67，使 PTO 以微秒为增量单位（或 16#8D 使 PTO 以毫秒为增量单位）。用这些值设置控制字节的目的是：允许 PTO/PWM 功能，选择 PTO 操作，选择以微秒或毫秒为增量单位，设置更新脉冲计数和周期值。
2. 向 SMW68（字）写入所希望的周期值。
3. 向 SMD72（双字）写入所希望的脉冲计数。
4. 可选步骤。如果你想在一个脉冲串输出（PTO）完成时立刻执行一个相关功能，则可以编程，使脉冲串输出完成中断事件（事件号 19）调用一个中断子程序，并执行全局中断允许指令。参见本章中断指令节，以了解中断处理的详细内容。
5. 执行 PLS 指令，使 S7-200 对 PTO/PWM 发生器编程。
6. 退出子程序。

修改 PTO 的周期（单段操作）

对于单段 PTO 操作，你可以使用一个中断服务程序或者子程序来改变周期。当使用单段 PTO 操作时，要有中断服务程序或者子程序中改变周期，使用以下步骤：

1. 把 16#81 送入 SMB67，使 PTO 以微秒为增量单位（或 16#89 使 PTO 以毫秒为增量单位）。用这些值设置控制字节的目的是：允许 PTO/PWM 功能，选择 PTO 操作，选择以微秒或毫秒为增量单位，和设置更新周期值。
2. 向 SMW68（字）写入所希望的周期值。
3. 执行 PLS 指令，使 S7-200 对 PTO/PWM 发生器编程，在更新周期的 PTO 波形开始前，CPU 必须完成已经启动的 PTO。
4. 退出中断程序或子程序。

改变 PTO 脉冲数（单段操作）

对于单段 PTO 操作，你可以使用一个中断服务程序或者子程序来改变脉冲数。当使用单段 PTO 操作时，要在中断服务程序或者子程序中改变脉冲数，使用以下步骤：

1. 把 16#84 送入 SMB67，使 PTO 以微秒为增量单位（或 16#8C 使 PTO 以毫秒为增量单位）。用这些值设置控制字节的目的是：允许 PTO/PWM 功能，选择 PTO 操作，选择以微秒或毫秒为增量单位，和设置更新脉冲个数。
2. 向 SMD72（双字）写入所希望的脉冲个数。
3. 执行 PLS 指令，使 S7-200 对 PTO/PWM 发生器编程，在更新周期的 PTO 波形开始前，CPU 必须完成已经启动的 PTO。
4. 退出中断程序或子程序。

改变 PTO 的周期和脉冲数（单段操作）

对于单段 PTO 操作，你可以使用一个中断服务程序或者子程序来改变周期和脉冲数。当使用单段 PTO 操作时，要在中断服务程序或者子程序中改变周期和脉冲数，使用以下步骤：

1. 把 16#85 送入 SMB67，使 PTO 以微秒为增量单位（或 16#8D 使 PTO 以毫秒为增量单位）。用这些值设置控制字节的目的是：允许 PTO/PWM 功能，选择 PTO 操作，选择以微秒或毫秒为增量单位，设置更新周期脉冲个数。
2. 向 SMW68（字）写入所希望的周期值。
3. 向 SMD72（双字）写入所希望的脉冲个数。
4. 执行 PLS 指令，使 S7-200 对 PTO/PWM 发生器编程，在更新周期的 PTO 波形开始前，CPU 必须完成已经启动的 PTO。
5. 退出中断程序或子程序。

初始化多段 PTO 输出

通常，你使用一个子程序来初始化多段 PTO 脉冲输出。你在主程序中调用初始化子程序。用首次扫描标志位（SM0.1）将 PTO 使用的输出点清 0 并且调用子程序来完成初始化操作。由于采取了这样的子程序调用，后续扫描不会再调用该子程序，从而减少了扫描时间，也使程序更加结构化。

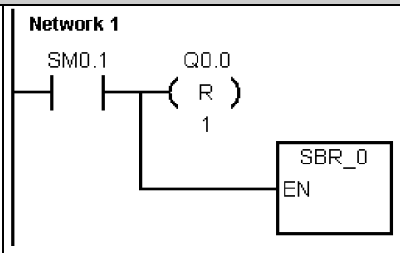
在主程序中调用初始化子程序之后，在初始化子程序中要使用以下步骤创建控制逻辑，配置 Q0.0 的脉冲输出：

1. 把 16#A0 送入 SMB67，使 PTO 以微秒为增量单位（或 16#A8 使 PTO 以毫秒为增量单位）。用这些值设置控制字节的目的是：允许 PTO/PWM 功能，选择 PTO 操作，选择以微秒或毫秒为增量单位，设置更新脉冲个数和周期值。
2. 向 SMW168（字）写入包络表的起始 V 存储器偏移值。
3. 在包络表中设定段数，确保段数区（表的第一个字节）正确。
4. 可选步骤。如果你想在一个脉冲串输出（PTO）完成时立刻执行一个相关功能，则可以编程，使脉冲串输出完成中断事件（事件号 19）调用一个中断子程序，并执行全局中断允许指令。

5. 执行 PLS 指令，使 S7-200 对 PTO/PWM 发生器编程。
6. 退出子程序。

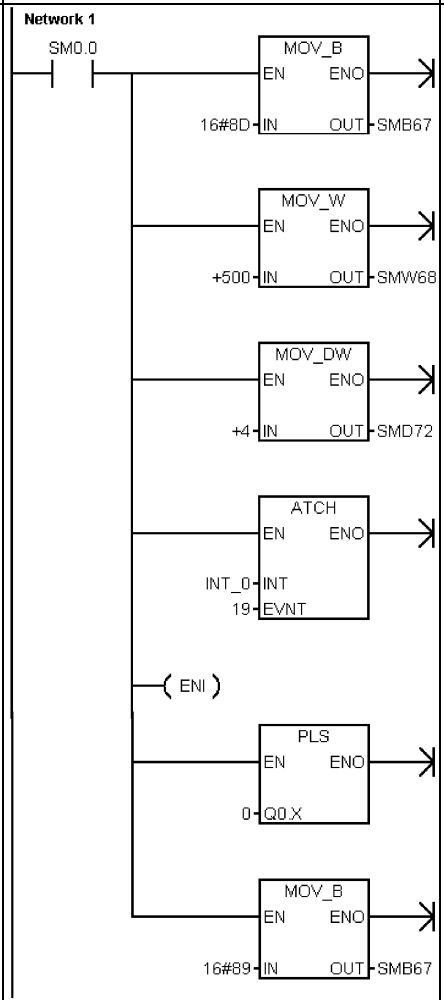
单段脉冲串操作（PTO）程序举例：

MAIN




```
Network1  //在首次扫描将映象寄存器清 0 并调用子程序 0。
LD      SM0.1
R       Q0.0，1
CALL    SBR_0
```

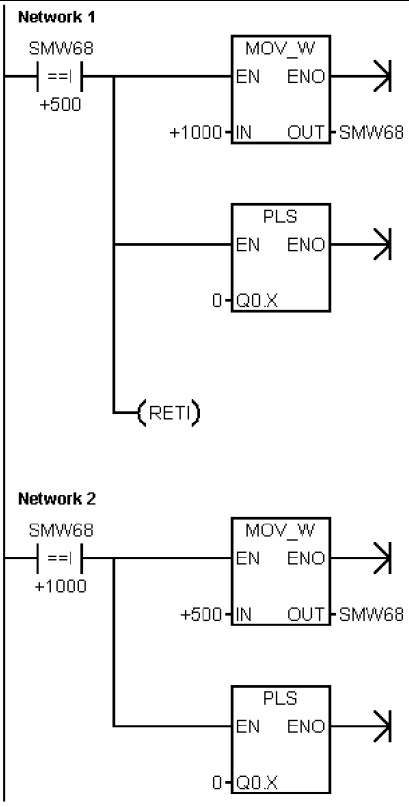
SBR0




```
Network 1    //子程序 0 开始：配置 PTO
             //1.  设置控制字节。
             //－选择 PTO 操作。
             //－选择单段操作。
             //－选择 ms 增量。
             //－使能脉冲数和周期装载。
             //－使能 PTO 功能
             //2.  设定周期为 500ms。
             //3.  设定脉冲数为 4。
             //4.  定义中断服务程序 0 为 PTO 完成中断。
             //5.  全局中断允许。
             //6.  启动 PTO 操作，PLS0=>Q0.0。
             //7.  为改变周期预装控制字节。
LD      SM0.0
MOVB    16#8D，SMB67
MOVW    +500，SMW68
MOVD    +4，SMD72
ATCH    INT_0，19
ENI
PLS     0
MOVB    16#89，SMB67
```

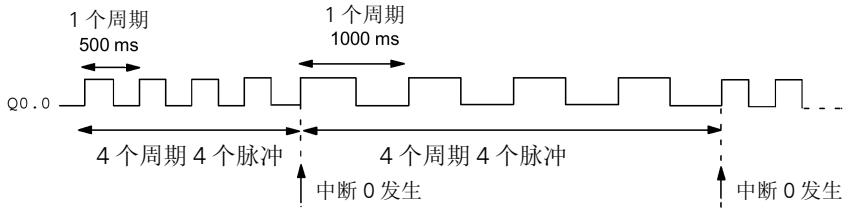
单段脉冲串操作（PTO）程序举例续：

INT 0



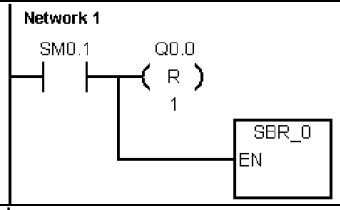

```
Network1    //如果当前周期为 500ms，
            //设置周期为 1000ms，输出 4 个脉冲。
LDW=   SMW68，+500
MOVW   +1000，SMW68
PLS    0
CRETI
Network2    //如果当前周期的 1000ms，设置周期为
            //500ms，输出 4 个脉冲。
LDW=   SMW68，+1000
MOVW   +500，SMW68
PLS    0
```

时序图



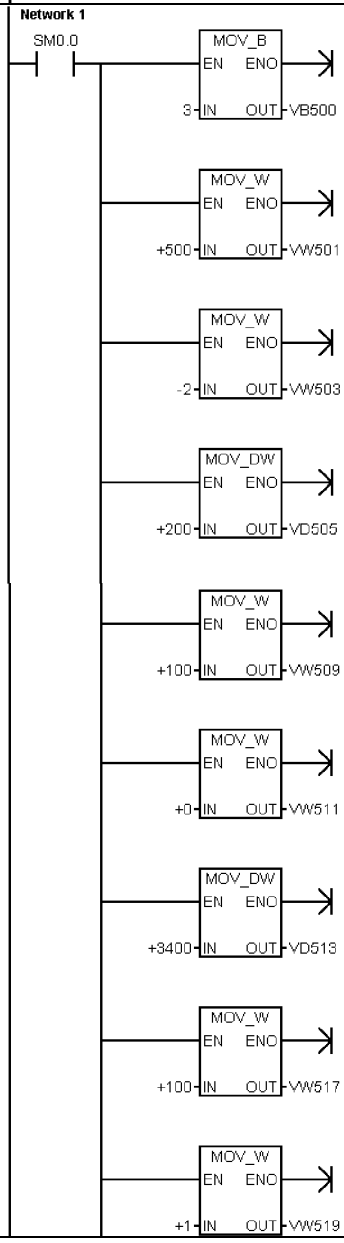
多段脉冲串操作（PTO）程序举例：

MAIN




```
Network1  //在首次扫描将映象寄存器清 0 并调用子程序 0。
LD      SM0.1
R       Q0.0，1
CALL    SBR_0
```

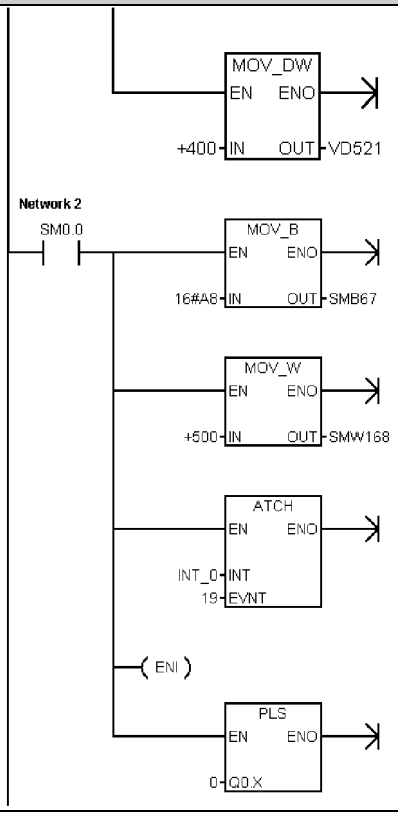
SBR0




```
Network 1    //预装 PTO 包络表：
             //设包络表段数为 3。
             //配置每一段。
             //1.  配置第一段：
             //－设置初始周期=500ms。
             //－设置周期增量为-2ms。
             //－设置脉冲数为 200。
             //2.  配置第二段：
             //－设置初始周期=100ms。
             //－设置周期增量为 0ms。
             //－设置脉冲数为 3400。
             //3.  配置第三段：
             //－设置初始周期=100ms。
             //－设置周期增量为 1ms。
             //－设置脉冲数为 400。
LD       SM0.0
MOVB     3，VB500
MOVW     +500，VW501   //段 1
MOVW     –2，VW503
MOVD     +200，VD505
MOVW     +100，VW509   //段 2
MOVW     +0，VW511
MOVD     +3400，VD513
MOVW     +100，VW517   //段 3
MOVW     +1，VW519
MOVD     +400，VD521
```

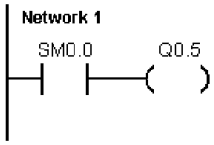
多段脉冲串操作（PTO）程序举例（续）

SBR0 Continued




```
Network2    //1.  设置控制字节
            //－选择 PTO 操作
            //－选择多段操作
            //－选择 ms 增量
            //－使能 PTO 功能
            //2.  设置包络表起始地址为 V500。
            //3.  定义中断服务程序 0 为 PTO 完成中断。
            //4.  全局中断允许。
            //5.  启动 PTO 操作，PLS0=>Q0.0。
LD      SM0.0
MOVB    16#A8，SMB67
MOVW    +500，SMW168
ATCH    INT_0，19
ENI
PLS     0
```

INT0

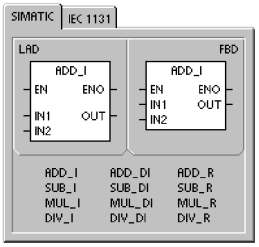



```
Network 1   //当 PTO 包络完成时，输出 Q0.5 接通。
LD      SM0.0
=       Q0.5
```

数字运算指令

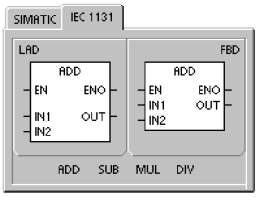
加、减、乘、除指令

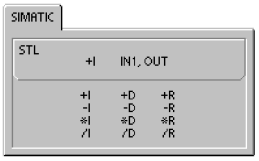
加法	减法	
IN1+IN2=OUT	IN1-IN2=OUT	*LAD 和 FBD*
IN1+OUT=OUT	OUT-IN1=OUT	*STL*



整数加法（+I）或者整数减法（-I）指令，将两个 16 位整数相加或者相减，产生一个 16 位结果。双整数加法（+D）或者双整数减法（-D）指令，将两个 32 位整数相加或者相减，产生一个 32 位结果。实数加法（+R）或者实数减法（-R）指令，将两个 32 位实数相加或者相减，产生一个 32 位实数结果。

乘法	除法	
IN1* IN2=OUT	IN1/IN2=OUT	*LAD 和 FBD*
IN1* OUT=OUT	OUT/IN1=OUT	*STL*





整数乘法（*I）或者整数除法（/I）指令，将两个 16 位整数相乘或者相除，产生一个 16 位结果。（对于除法，余数不被保留。）双整数乘法（*D）或者双整数除法（/D）指令，将两个 32 位整数相乘或者相除，产生一个 32 位结果。（对于除法，余数不被保留。）实数乘法（*R）或者实数除法（/R）指令，将两个 32 位实数相乘或者相除，产生一个 32 位实数结果。

SM 位和 ENO

SM1.1 表示溢出错误和非法值。如果 SM1.1 置位，SM1.0 和 SM1.2 的状态不再有效而且原始输入操作数不会发生变化。如果 SM1.1 和 SM1.3 没有置位，那么数字运算产生一个有效的结果，同时 SM1.0 和 SM1.2 有效。在除法运算中，如果 SM1.3 置位，其它数学运算标志位不会发生变化。

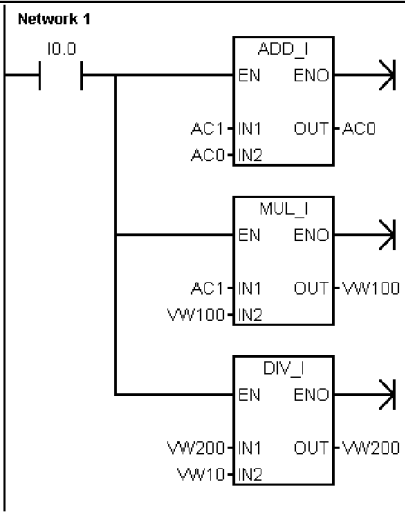
使 ENO=0 的错误条件	受影响的特殊存储器位
• SM1.1（溢出） • SM1.3（被 0 除） • 0006（间接寻址）	• SM1.0（结果为 0） • SM1.1（溢出，运算过程中产生非法数值或者输入参数非法） • SM1.2（结果为负） • SM1.3（被 0 除）

表 6-37　加、减、乘、除指令的有效操作数

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	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

实数（或者浮点数）的表示格式采用 ANSI/IEEE 754-1985 标准（单精度）。要得到更多信息请参考该标准。

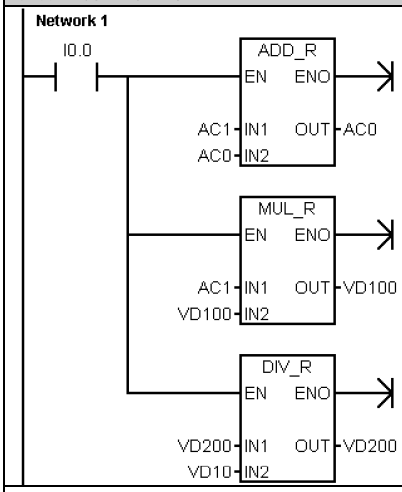
整数运算指令程序举例




```
Network1
LD      I0.0
+I      AC1，AC0
*I      AC1，VW100
/I      VW10，VW200
```

加法				乘法				除法				
40	+	60	=	100	40	*	20	=	800	4000	/	40	=	100
AC1		AC0		AC0	AC1		VW100		VW100	VW200		VW10		VW200

实数运算指令程序举例




```
Network1
LD      I0.0
+R      AC1，AC0
*R      AC1，VD100
/R      VD10，VD200
```

加法

4000.0 + 6000.0 = 10000.0
AC1 AC0 AC0

乘法

400.0 * 200.0 = 80000.0
AC1 VD100 VD100

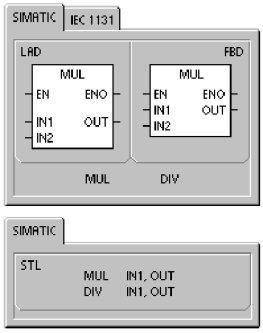
除法

4000.0 / 41.0 = 97.5609
VD200 VD10 VD200

整数乘法产生双整数和带余数的整数除法

整数乘法产生双整数

IN1*IN2=OUT　　LAD 和 FBD
IN1*OUT=OUT　　STL



整数乘法产生双整数指令（MUL），将两个 16 位整数相乘，得到 32 位结果。在 STL 的 MUL 指令中，OUT 的低 16 位被用作一个乘数。

带余数的整数除法

IN1/IN2=OUT　　LAD 和 FBD
OUT/IN1 =OUT　　STL

带余数的整数除法指令（DIV），将两个 16 位整数相除，得到 32 位结果。其中 16 位为余数（高 16 位字中），另外 16 位为商（低 16 位字中）。

在 STL 的 DIV 指令中，OUT 的低 16 位被用作除数。

SM 位和 ENO

对于在本页中介绍的两条指令，特殊存储器（SM）标志位表示错误和非法值。如果在除法指令执行时，SM1.3 置位，其它数字运算标志位不会发生变化。否则，当数字运算完成时，所有数字运算标志位都会包含有效状态。

使 ENO=0 的错误条件：

- SM1.1（溢出）
- SM1.3（被 0 除）

受影响的特殊存储器位：

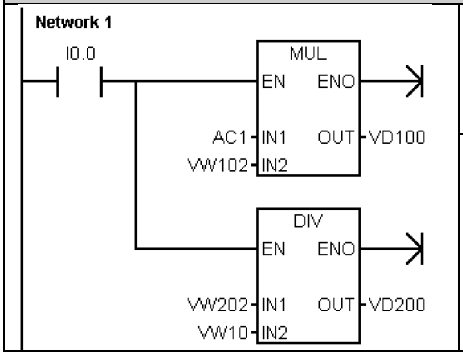
- SM1.0（结果为 0）
- SM1.1（溢出）

- 0006（间接寻址）
- SM1.2（结果为负）
- SM1.3（被 0 除）

表 6-38　整数乘法产生双整数和带余数的整数除法指令的有效操作数

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

整数乘法产生双整数和带余数的整数除法指令程序举例




```
Network1
LD      I0.0
MUL     AC1，VD100
DIV     VW10，VD200
```

整数乘法产生双整数　400 (AC1) * 200 (VW102) = 80000 (VD100)

带余数的整数除法　4000 (VW202) / 41 (VW10) = 23 (余数, VW200) 97 (商, VW202)，VD200

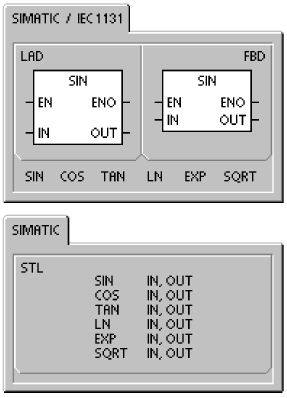
数学功能指令

正弦、余弦和正切

正弦（SIN）、余弦（COS）和正切（TAN）指令计算角度值 IN 的三角函数值，并将结果存放在 OUT 中。输入角度值为弧度值。

SIN（IN）=OUT　COS（IN）=OUT　TAN（IN）=OUT

要将角度从度数变为弧度，可以使用 MUL_R（*R）指令，将度数乘以 1.745329E-2（接近 π/180）即可。



自然对数和自然指数

自然对数指令（LN）计算输入值 IN 的自然对数，并将结果存放在 OUT 中。

自然指数指令（EXP）计算输入值 IN 为指数的自然指数值，并将结果存放在 OUT 中。

LN（IN）=OUT　　EXP（IN）=OUT

要从自然对数计算出以 10 为底的对数值，可以使用除法指令，将自然对数值除以 2.302585（接近 10 的自然对数）即可。

要计算任意实数的任意实数次方，包括分数形式的指数，需要将自然对数指令和自然指数指令结合在一起使用。例如：要计算 X 的 Y 次方，使用以下公式：EXP（Y*LN（X））。

平方根

平方根指令（SQRT）计算实数 IN 的平方根，并将结果存放在 OUT 中。

SQRT（IN）=OUT

如果要求其它次数的方根值：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

数学功能指令的 SM 位和 ENO

对于本页中描述的所有指令，SM1.1 用来表示溢出错误或者非法的数值。如果 SM1.1 置位，SM1.0 和 SM1.2 的状态不再有效而且原始输入操作数不会发生变化。如果 SM1.1 没有置位，那么数字运算产生一个有效的结果，同时 SM1.0 和 SM1.2 状态有效。

使 ENO=0 的错误条件：

- SM1.1（溢出）
- 0006（间接寻址）

受影响的特殊存储器位

- SM1.0（结果为 0）
- SM1.1（溢出）
- SM1.2（结果为负）

表 6-39　数学功能指令的有效操作数

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
OUT	REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC

实数（或者浮点数）的表示格式采用 ANSI/IEEE 754-1985 标准（单精度）。要得到更多信息请参考该标准。

递增和递减指令

递增

IN+1=OUT　　*LAD 和 FBD*

OUT+1=OUT　　*STL*

递减

IN-1=OUT　　*LAD 和 FBD*

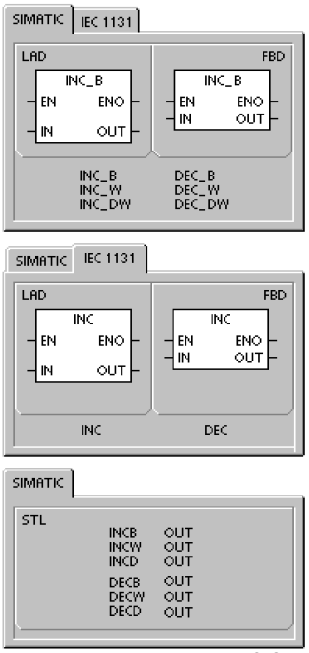
OUT-1=OUT　　*STL*

递增或者递减指令将输入 IN 加 1 或者减 1，并将结果存放在 OUT 中。

字节递增（INCB）和字节递减（DECB）操作是无符号的。

字递增（INCW）和字递减（DECW）操作是有符号的。

双字递增（INCD）和双字递减（DECD）操作是有符号的。



使 ENO=0 的错误条件：

- SM1.1（溢出）
- 0006（间接寻址）

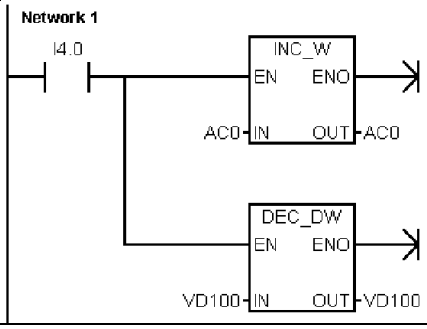
受影响的特殊存储器位

- SM1.0（结果为 0）
- SM1.1（溢出）
- SM1.2（结果为负）对于字和双字操作有效

表 6-40 递增和递减指令的有效操作数

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
	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
OUT	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD
	INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC,*VD、*AC、*LD、
	DINT	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC

递增和递减指令程序举例

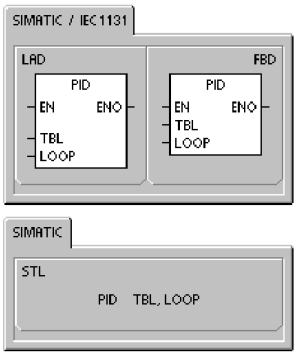



```
Network1
LD      I4.0
INCW    AC0
DECD    VD100
```

字递增 125 (AC0) + 1 = 126 (AC0)

双字递减 128000 (VD100) − 1 = 127999 (VD100)

比例/积分/微分（PID）回路控制指令



PID 回路控制指令（PID）运用回路表中的输入和组态信息，进行 PID 运算。

使 ENO=0 的错误条件：

- SM1.1（溢出）
- 0006（间接寻址）

受影响的特殊存储器位

- SM1.1（溢出）

PID 回路指令（包含比例、积分、微分回路）可以用来进行 PID 运算。但是，可以进行这种 PID 运算的前提条件是逻辑堆栈栈顶（TOS）值必须为 1。该指令有两个操作数：TBL 和 LOOP。其中 TBL 是回路表的起始地址；LOOP 是回路号，可以是 0 到 7 的整数。

在程序中最多可以用 8 条 PID 指令。如果两个或两个以上的 PID 指令用了同一个回路号，那么即使这些指令的回路表不同，这些 PID 运算之间也会相互干涉，产生不可预料的结果。

回路表包含 9 个参数，用来控制和监视 PID 运算。这些参数分别是过程变量当前值（PVn），过程变量前值（PV_{n-1}），给定值（SP_n），输出值（M_n），增益（K_c），采样时间（T_s），积分时间（TI），微分时间（TD）和积分项前值（MX）。

为了让 PID 运算以预想的采样频率工作，PID 指令必须用在定时发生的中断程序中，或者用在主程序中被定时器所控制以一定频率执行。采样时间必须通过回路表输入到 PID 运算中。

表 6-41 PID 回路控制指令的有效操作数

输入/输出	数据类型	操作数
TBL	BYTE	VB
LOOP	BYTE	常数（0 到 7）

指令向导

STEP 7-Micro/WIN 提供了 PID 指令向导，指导你定义一个闭环控制过程的 PID 算法。在命令菜单中选择 **Tools>Instruction Wizard**，然后在指令向导窗口中选择 **PID** 指令。

理解 PID 算法

PID 控制器调节输出，保证偏差（e）为零，使系统达到稳定状态，偏差（e）是给定值（SP）和过程变量（PV）的差。PID 控制的原理基于下面的算式：输出 M（t）是比例项、积分项和微分项的函数。

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

输出 = 比例项 + 积分项 + 微分项

其中：

M(t)　　PID 回路的输出，是时间的函数

K_c　　PID 回路的增益

e　　PID 回路的偏差（给定值与过程变量之差）

$M_{initial}$　　PID 回路输出的初始值

为了能让数字计算机处理这个控制算式，连续算式必须离散化为周期采样偏差算式，才能用来计算输出值。数字计算机处理的算式如下：

M_n	=	$K_c * e_n$	+	$K_I * \sum_{1}^{n} + M_{initial}$	+	$K_D * (e_n - e_{n-1})$
输出	=	比例项	+	积分项	+	微分项

其中：

M_n　　在第 n 采样时刻，PID 回路输出的计算值

K_C　　PID 回路增益

e_n　　在第 n 采样时刻的偏差值

e_{n-1}　　在第 n-1 采样时刻的偏差值（偏差前项）

K_I　　积分项的比例常数

$M_{initial}$　　PID 回路输出的初值

K_D　　微分项的比例常数

从这个公式可以看出，积分项是从第 1 个采样周期到当前采样周期所有误差项的函数，微分项是当前采样和前一次采样的函数，比例项仅是当前采样的函数。在数字计算机中，不保存所有的误差项，实际上也不必要。

由于计算机从第一次采样开始，每有一个偏差采样值必须计算一次输出值，只需要保存偏差前值和积分项前值。利用计算机处理的重复性，可以化简以上算式为：

M_n	=	$K_c * e_n$	+	$K_I * e_n + MX$	+	$K_D * (e_n - e_{n-1})$
输出	=	比例项	+	积分项	+	微分项

其中：

M_n　　在第 n 采样时刻，PID 回路输出的计算值

K_C　　PID 回路增益

e_n　　在第 n 采样时刻的偏差值

e_{n-1}　　在第 n-1 采样时刻的偏差值（偏差前项）

K_I　　积分项的比例常数

MX　　积分项前值

K_D　　微分项的比例常数

CPU 实际使用以上简化算式的改进形式计算 PID 输出。这个改进型算式是：

M_n	=	MP_n	+	MI_n	+	MD_n
输出	=	比例项	+	积分项	+	微分项

其中：

M_n 第 n 采样时刻的计算值

MP_n 第 n 采样时刻的比例项值

MI_n 第 n 采样时刻的积分项值

MD_n 第 n 采样时刻的微分项值

理解 PID 方程的比例项

比例项 MP 是增益（K_C）和偏差（e）的乘积。其中 K_C 决定输出对偏差的灵敏度，偏差（e）是给定值（SP）与过程变量值（PV）之差。CPU 执行的求比例项算式是：

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

其中：

MP_n 第 n 采样时刻比例项的值

K_c 增益

SP_n 第 n 采样时刻的给定值

PV_n 第 n 采样时刻的过程变量值

理解 PID 方程的积分项

积分项值 MI 与偏差和成正比。CPU 执行的求积分项算式是：

$$MI_n = K_c * T_S/T_I * (SP_n - PV_n) + MX$$

其中：

MI_n 第 n 采样时刻的积分项值

K_c 增益

T_S 采样时间间隔

T_I 积分时间

SP_n 第 n 采样时刻的给定值

PV_n 第 n 采样时刻的过程变量值

MX 第 n-1 采样时刻的积分项（积分项前值）（也称积分和或偏置）

积分和（MX）是所有积分项前值之和。在每次计算出 MI_n 之后，都要用 MI_n 去更新 MX。其中 MI_n 可以被调整或限定（详见“变量和范围”一节）。MX 的初值通常在第一次计算输出以前被设置为 $M_{initial}$(初值)。积分项还包括其他几个常数：增益（K_C），采样时间间隔（T_S）和积分时间（T_I）。其中采样时间是重新计算输出的时间间隔，而积分时间控制积分项在整个输出结果中影响的大小。

理解 PID 方程的微分项

微分项值 MD 与偏差的变化成正比。CPU 执行的求微分项算式是：

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

为了避免给定值变化的微分作用而引起的跳变，假定给定值不变（$SP_n = SP_{n-1}$）。这样，可以用过程变量的变化替代偏差的变化，计算算式可改进为：

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

或

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

其中：

MD_n	第 n 采样时刻的微分项值
K_c	回路增益
T_s	回路采样时间
T_D	微分时间
SP_n	第 n 采样时刻的给定值
SP_{n-1}	第 n-1 采样时刻的给定值
PV_n	第 n 采样时刻的过程变量值
PV_{n-1}	第 n-1 采样时刻的过程变量值

为了下一次计算微分项值，必须保存过程变量，而不是偏差。在第一采样时刻，初始化为 $PV_{n-1}=PV_n$。

回路控制类型的选择

在许多控制系统中，只需要一种或两种回路控制类型。例如只需要比例回路或者比例积分回路。通过设置常量参数，可先选中想要的回路控制类型。

如果不想要积分回路，可以把积分时间设为无穷大。即使没有积分作用，积分项还是不为零，因为有初值 MX。

如果不想要微分回路，可以把微分时间置为零。

如果不想要比例回路，但需要积分或积分微分回路，可以把增益设为 0.0，系统会在计算积分项和微分项时，把增益当作 1.0 看待。

回路输入的转换和标准化

每个 PID 回路有两个输入量，给定值（SP）和过程变量（PV）。给定值通常是一个固定的值，比如设定的汽车速度。过程变量是与 PID 回路输出有关，可以衡量输出对控制系统作用的大小。在汽车速度控制系统中，过程变量可以是测速仪的输入（衡量车轮转速高低）。

给定值和过程变量都可能是现实世界的值，它们的大小、范围和工程单位都可能不一样。PID 指令在对这些量进行运算以前，必须把他们转换成标准的浮点型实数。

转换的第一步是把 16 位整数值转成浮点型实数值。下面的指令序列提供了实现这种转换的方法：

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

转换的下一步是把实数值进一步标准化为 0.0~1.0 之间的实数。下面的算式可以用来标准化给定值或过程变量：

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

其中：

R_{Norm}	标准化的实数值
R_{Raw}	没有标准化的实数值或原值
Offset	单极性为 0.0，双极性为 0.5
Span	值域大小，可能的最大值减去可能的最小值 单极性为 32,000（典型值） 双极性为 64,000（典型值）

下面的指令把双极性实数标准化为 0.0~1.0 之间的实数。通常用在第一步转换之后：

```
/R      64000.0, AC0      //累加器中的标准化值
+R      0.5, AC0          //加上偏置，使其在 0.0~1.0 之间
MOVR    AC0, VD100        //标准化的值存入回路表
```

回路输出值转换成刻度整数值

回路输出值一般是控制变量，比如，在汽车速度控制中，可以是油阀开度的设置。同时，输出是 0.0~1.0 之间的标准化了的实数值，在回路输出驱动模拟输出之前，必须把回路输出转换成相应的 16 位整数。这一过程，是给定值或过程变量的标准化转换的逆过程。该过程的第一步把回路输出转换成相应的实数值，公式如下：

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

其中：

R_{Scal}	回路输出的刻度实数值
M_n	回路输出的标准化实数值
Offset	单极性为 0.0，双极性为 0.5
Span	值域大小，可能的最大值减去可能的最小值 单极性为 32,000（典型值） 双极性为 64,000（典型值）

这一过程可以用下面的指令序列完成：

```
MOVR    VD108, AC0      //把回路输出值移入累加器
-R      0.5, AC0        //仅双极性有此句
*R      64000.0, AC0    //在累加器中得到刻度值
```

下一步是把回路输出的刻度转换成 16 位整数，可通过下面的指令序列来完成：

```
ROUND    AC0, AC0       //把实数转换为 32 位整数
DTI      AC0, LW0       //把 32 位整数转换为 16 位整数
MOVW     LW0, AQW0      //把 16 位整数写入模拟输出寄存器
```

正作用或反作用回路

如果增益为正，那么该回路为正作用回路。如果增益为负，那么是反作用回路。对于增益为零的积分或微分控制来说，如果指定积分时间、微分时间为正，就是正作用回路；指定为负值，则是反作用回路。

变量和范围

过程变量和给定值是 PID 运算的输入值，因此在回路表中，这些值只能被回路指令读而不能改写。

输出变量是由 PID 运算产生的，所以在每一次 PID 运算完成之后，需更新回路表中的输出值，输出值被限定在 0.0~1.0 之间。当 PID 指令从手动方式转变到自动方式时，回路表中的输出值可以用来初始化输出值（有关 PID 指令的方式详见下面的“控制方式”一节）。

如果使用积分控制，积分项前值要根据 PID 运算结果更新。这个更新了的值用作下一次 PID 运算的输入，当输出值超过范围（大于 1.0 或小于 0.0），那么积分项前值必须根据下列公式进行调整：

$MX = 1.0 - (MP_n + MD_n)$　　*当计算输出 $M_n > 1.0$*

或

$MX = - (MP_n + MD_n)$　　*当计算输出 $M_n < 0.0$*

其中：

MX	经过调整了的积分和（积分项前值）
MP_n	第 n 采样时刻的比例项值
MD_n	第 n 采样时刻的微分项值
M_n	第 n 采样时刻的输出值

这样调整积分前值，一旦输出回到范围后，可以提高系统的响应性能。而且积分项前值也要限制在 0.0~0.1 之间，然后在每次 PID 运算结束之后，把积分项前值写入回路表，以备在下次 PID 运算中使用。

用户可以在执行 PID 指令以前修改回路表中积分项前值。在实际运用中，这样做的目的是找到由于积分项前值引起的问题。手工调整积分项前值时，必须小心谨慎，还应保证写入的值在 0.0~1.0 之间。

回路表中的给定值与过程变量的差值（e）是用于 PID 运算中的差分运算，用户最好不要去修改此值。

控制方式

S7－200 的 PID 回路没有设置控制方式，只要 PID 块有效，就可以执行 PID 运算。在这种意义上说，PID 运算存在一种“自动”运行方式。当 PID 运算不被执行时，我们称之为“手动”方式。

同计数器指令相似，PID 指令有一个使能位。当该使能位检测到一个信号的正跳变（从 0 到 1），PID 指令执行一系列的动作，使 PID 指令从手动方式无扰动地切换到自动方式。为了达到无扰动切换，在转变到自动控制前，必须用手动方式把当前输出值填入回路表中的 M_n 栏。PID 指令对回路表中的值进行下列动作，以保证当使能位正跳变出现时，从手动方式无扰动切换到自动方式：

- 置给定值（SP_n）= 过程变量（PV_n）
- 置过程变量前值（PV_{n-1}）= 过程变量现值（PV_n）
- 置积分项前值（MX）= 输出值（M_n）

PID 使能位的默认值是 1，在 CPU 启动或从 STOP 方式转到 RUN 方式时建立。CPU 进入 RUN 方式后首次使 PID 块有效，没有检测到使能位的正跳变，那么就没有无扰动切换的动作。

报警与特殊操作

PID 指令是执行 PID 运算的简单而功能强大的指令。如果其他过程需要对回路变量进行报警等特殊操作，那么可以用 CPU 支持的基本指令实现这些特殊操作功能。

出错条件

如果指令指定的回路表起始地址以及回路号操作数超出范围，那么在编译期间，CPU 指令产生编译错误（范围错误），从而编译失败。PID 指令不检查回路表中的值是否在范围之内，所以必须小心操作以保证过程变量和设定值不超界。

PID 指令不检查回路表中的值是否超界，你必须保证过程变量和设定值（以及偏置和前一次过程变量）必须在 0.0 到 1.0 之间。

如果 PID 计算的算术运算发生错误，那么特殊存储器标志位 SM1.1(溢出或非法值)会被置 1，并且中止 PID 指令的执行。

（要想消除这种错误，单靠改变回路表中的输出值是不够的，正确的方法是在下一次执行 PID 运算之前，改变引起算术运算错误的输入值，而不是更新输出值）。

回路表

36 个字节的回路表的格式如表 6-42 所示。

表 6-42 回路表格式

偏移地址	域	格式	类型	描述
0	过程变量 (PV_n)	双字 – 实数	输入	过程变量，必须在 0.0~1.0 之间
4	设定值 (SP_n)	双字 – 实数	输入	给定值，必须在 0.0~1.0 之间
8	输出值 (M_n)	双字 – 实数	输入/输出	输出值，必须在 0.0~1.0 之间
12	增益 (K_C)	双字 – 实数	输入	增益是比例常数，可正可负
16	采样时间 (T_S)	双字 – 实数	输入	单位为秒，必须是正数
20	积分时间 (T_I)	双字 – 实数	输入	单位为分钟，必须是正数
24	微分时间 (T_D)	双字 – 实数	输入	单位为分钟，必须是正数
28	积分项前项 (MX)	双字 – 实数	输入/输出	积分项前项，必须在 0.0~1.0 之间
32	过程变量前值 (PV_{n-1})	双字 – 实数	输入/输出	最近一次 PID 运算的过程变量值

PID 指令编程举例

在本例中，有一水箱需要维持一定的水位，该水箱里的水以变化的速度流出。这就需要有一个水泵以不同的速度给水箱供水，以维持水位不变，这样才能使水箱不断水。

本系统的给定值是水箱满水位的 75%时的水位，过程变量由漂浮在水面的水位测量仪给出。输出值是进水泵的速度，可以从允许最大值的 0%变到 100%。

给定值可以预先设定后直接输入到回路表中，过程变量值是来自水位表的单极性模拟量，回路输出值也是一个单极性模拟量，用来控制进水泵速度。这两个模拟量的范围是 0.0~1.0，分辨率为 1/32000（标准化）。

在本系统中，只使用比例和积分控制，其回路增益和时间常数可以通过工程计算初步确定。但还需要进一步调整以达到最优控制效果。初步确定的增益和时间常数为：

K_C 是 0.25
T_S 是 0.1 秒
T_I 是 30 分钟

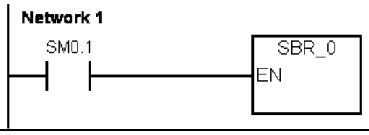
系统启动时，关闭出水口，用手动控制进水泵速度，使水位达到满水位的 75%，然后打开出水口，同时水泵控制从手动方式切换到自动方式。这种切换由一个输入的数字量控制，描述如下：

I0.0 位控制手动到自动的切换，0 代表手动；1 代表自动。

当工作在手动控制方式下，可以把水泵速度（0.0~1.0 之间的实数）写到 VD108（VD108 是回路表中保存输出的寄存器）。

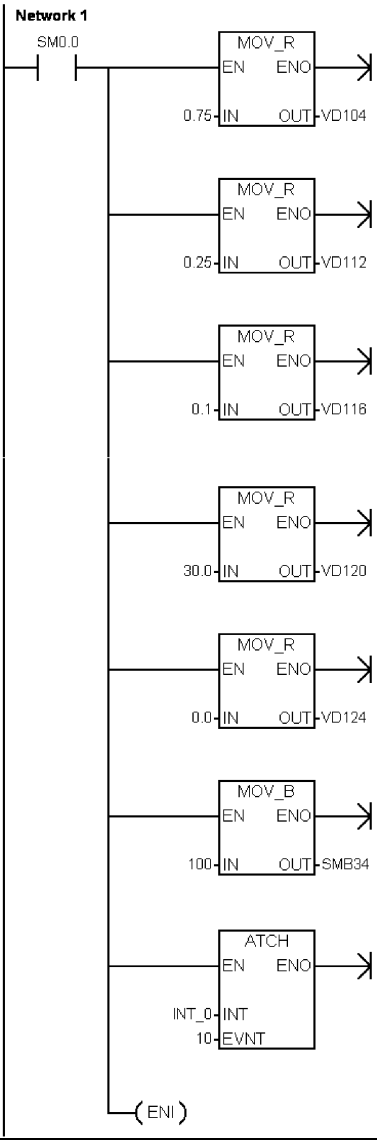
PID 回路控制指令程序举例

MAIN




```
Network 1      //在首次扫描调用初始化子程序
LD             SM0.1
CALL           SBR_0
```

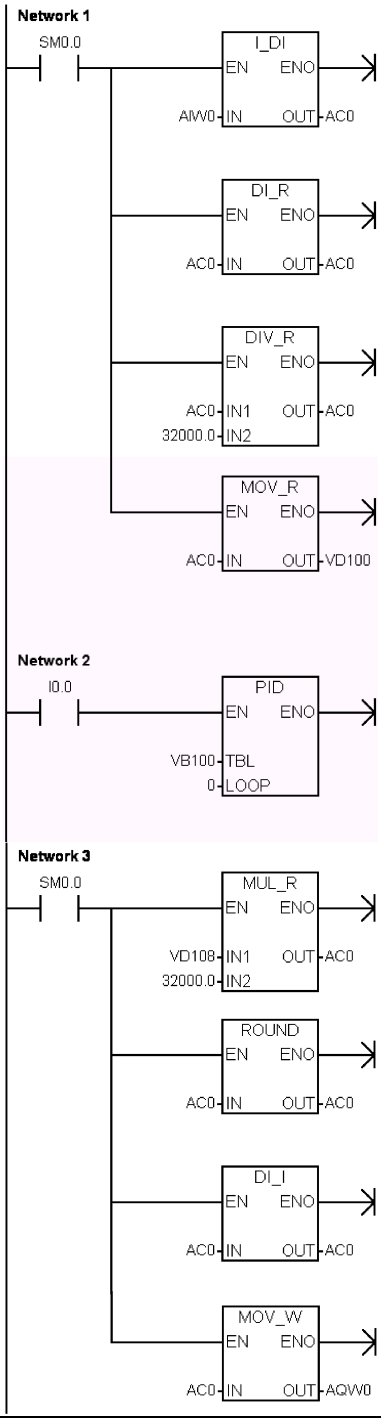
SBR0




```
Network 1      //装载PID参数和连接PID中断服务程序：
               //1. 装入回路设定值 = 75%。
               //2. 装入回路增益 = 0.25。
               //3. 装入回路采样时间= 1秒。
               //4. 装入积分时间 = 30 分钟。
               //5. 关闭微分作用。
               //6. 设定定时中断0的时间间隔为100ms。
               //7. 设定定时中断执行PID程序。
               //8. 允许中断。
LD             SM0.0
MOVR           0.75, VD104
MOVR           0.25, VD112
MOVR           0.1, VD116
MOVR           30.0, VD120
MOVR           0.0, VD124
MOVB           100, SMB34
ATCH           INT_0, 10
ENI
```

PID 回路控制指令程序举例（续）

INT 0



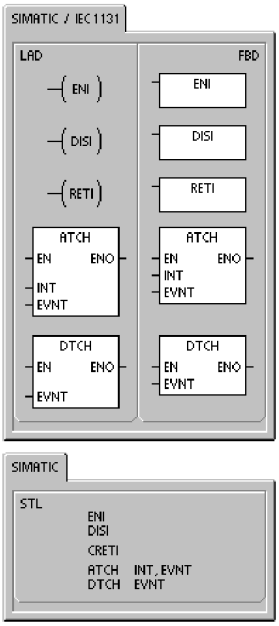

```
Network 1       //将PV转换为标准化的实数：
                //1. 将整数转换成双整数。
                //2.将双整数转换成实数。
                //3.将数值标准化。
                //4.将标准化之后的PV存入回路表。
LD      SM0.0
ITD     AIW0, AC0
DTR     AC0, AC0
/R      32000.0, AC0
MOVR    AC0, VD100
Network 2       //在自动方式下执行PID指令。
LD      I0.0
PID     VB100, 0
Network 3       //把Mn转换为16位整数。
                //Mn为单极性且非负的数。
                //1. 把输出值送到累加器。
                //2. 标准化累加器中的值。
                //3. 实数转换成双整数。
                //4. 双整数转换成整数。
                //5. 将数值写入模拟量输出。
LD      SM0.0
MOVR    VD108, AC0
*R      32000.0, AC0
ROUND   AC0, AC0
DTI     AC0, AC0
MOVW    AC0, AQW0
```

中断指令

中断允许和中断禁止

中断允许指令（ENI）全局地允许所有被连接的中断事件。中断禁止指令全局地禁止处理所有中断事件。

当进入 RUN 模式时，中断被禁止。在 RUN 模式，你可以执行全局中断允许指令（ENI）允许所有中断。全局中断禁止指令（DISI）不允许处理中断服务程序，但中断事件仍然会排队等候。



使 ENO＝0 的错误条件：

- 0004（试图在中断服务程序中执行 ENI、DISI 或者 HDEF 指令。）

中断条件返回

中断条件返回指令（CRETI）可以用来根据逻辑操作的条件，从中断服务程序中返回。

中断连接

中断连接指令（ATCH）将中断事件 EVNT 与中断服务程序号 INT 相关联，并使能该中断事件。

使 ENO＝0 的错误条件：

- 0002（与 HSC 的输入分配相冲突）

中断分离

中断分离指令（DTCH）将中断事件 EVNT 与中断服务程序之间的关联切断，并禁止该中断事件。

表 6-43　中断连接和中断分离指令的有效操作数

输入/输出	数据类型	操作数
INT	BYTE	常数（0 到 127）
EVNT	BYTE	常数 *CPU221 和 CPU222*：　0 到 12，19 到 23 和 27 到 33 *CPU224*：　0 到 23 和 27 到 33 *CPU226 和 CPU226XM*：　0 到 33

对中断连接和中断分离指令的理解

在激活一个中断程序前，必须在中断事件和该事件发生时希望执行的那段程序间建立一种联系。中断连接指令（ATCH）指定某中断事件（由中断事件号指定）所要调用的程序段（由中断程序号指定）。多个中断事件可调用同一个中断程序，但一个中断事件不能同时指定调用多个中断程序。在中断允许时，某个中断事件发生，只有为该事件指定的最后一个中断程序被执行。当为某个中断事件指定其所对应的中断程序时，该中断事件会自动被允许。如果用全局中断禁止指令（DISI）禁止所有中断，则每个出现的中断事件就进入中断队列，直到用全局中断允许指令（ENI）重新允许中断。

当把中断事件和中断程序连接时，自动允许中断。如果采用禁止全局中断指令不响应所有中断，每个中断事件进行排队，直到采用允许全局中断指令重新允许中断。

可以用中断分离指令（DTCH）截断中断事件和中断程序之间的联系，以单独禁止中断事件。中断分离指令（DTCH）使中断回到不激活或无效状态。

表 6-44　中断事件

事件号	中断描述	CPU 221 CPU 222	CPU 224	CPU 226 CPU 226XM
0	上升沿，I0.0	Y	Y	Y
1	下降沿，I0.0	Y	Y	Y
2	上升沿，I0.1	Y	Y	Y
3	下降沿，I0.1	Y	Y	Y
4	上升沿，I0.2	Y	Y	Y
5	下降沿，I0.2	Y	Y	Y
6	上升沿，I0.3	Y	Y	Y
7	下降沿，I0.3	Y	Y	Y
8	端口 0：接收字符	Y	Y	Y
9	端口 0：发送字符	Y	Y	Y
10	定时中断 0，SMB34	Y	Y	Y
11	定时中断 1，SMB35	Y	Y	Y
12	HSC0 CV=PV（当前值＝预置值）	Y	Y	Y
13	HSC1 CV=PV（当前值＝预置值）		Y	Y
14	HSC1 输入方向改变		Y	Y
15	HSC1 外部复位		Y	Y
16	HSC2CV=PV（当前值＝预置值）		Y	Y
17	HSC2 输入方向改变		Y	Y
18	HSC2 外部复位		Y	Y
19	PLS 0 脉冲数完成中断	Y	Y	Y
20	PLS 1 脉冲数完成中断	Y	Y	Y
21	定时器 T32 CT=PT 中断	Y	Y	Y
22	定时器 T96 CT=PT 中断	Y	Y	Y
23	端口 0：接收信息完成	Y	Y	Y
24	端口 1：接收信息完成			Y
25	端口 1：接收字符			Y
26	端口 1：发送字符			Y
27	HSC0 输入方向改变	Y	Y	Y
28	HSC0 外部复位	Y	Y	Y

事件号	中断描述	CPU 221 CPU 222	CPU 224	CPU 226 CPU 226XM
29	HSC4 CV=PV（当前值＝预置值）	Y	Y	Y
30	HSC4 输入方向改变	Y	Y	Y
31	HSC4 外部复位	Y	Y	Y
32	HSC3 CV=PV（当前值＝预置值）	Y	Y	Y
33	HSC5 CV=PV（当前值＝预置值）	Y	Y	Y

理解 S7-200 对中断服务程序的处理

执行中断服务程序用于响应与其相关的内部或者外部事件。一旦执行完中断服务程序的最后一条指令，控制权会回到主程序。你可以执行中断条件返回指令（CRETI）退出中断服务程序。表 6-45 中对于在应用程序中使用中断服务程序给出了一些指导和限定。

表 6-45　使用中断服务程序的指导和限定

指导
中断处理提供了对特殊的内部或外部事件的响应。用户应当优化中断程序以执行一个特殊的任务，然后把控制返回主程序。应当使中断程序短小而简单，执行时对其他处理也不要延时过长。如果做不到这些，意外的条件可能会引起由主程序控制的设备操作异常。对中断而言，其格言是“越短越好”。
限定
在中断程序中不能使用 DISI、ENI、HDEF、LSCR 和 END 指令。

系统对中断的支持

由于中断指令影响触点、线圈和累加器逻辑，所以系统保存和恢复逻辑堆栈、累加寄存器以及指示累加器和指令操作状态的特殊存储器标志位（SM）。这避免了由中断程序返回后对用户主程序执行现场所造成的破坏。

在主程序和中断程序间共享数据

你可以在主程序和一个或多个中断程序间共享数据。例如，用户主程序的某个地方可以为某个中断程序提供要用到的数据，反之亦然。如果用户程序共享数据，必须考虑中断事件异步特性的影响，这是因为中断事件会在用户主程序执行的任何地方出现。共享数据一致性问题的解决要依赖于主程序被中断事件中断时中断程序的操作。

这里有几种可以确保在用户主程序和中断程序间正确共享数据的编程技巧。这些技巧或限制共享存储器单元的访问方式，或让使用共享存储器单元的指令序列不会被中断。

- STL 程序共享单个变量：如果共享数据是单个字节、字、双字变量，而且用户程序用 STL 编写，那么通过把对共享数据操作得到的中间值只存储到非共享的存储器单元或累加器中，可以保证正确的共享访问。
- LAD 程序共享单个变量：如果共享数据是单个字节、字或双字变量，而且用户程序用梯形图编写，那么通过建立只用 Move 指令（MOVB、MOVW、MOVD、MOVR）访问共享存储器单元的约定，可以保证正确的共享访问。这些 Move 指令由执行时不受中断事件影响的单条 STL 指令组成，而其它许多梯形图指令是由可被中断的 STL 指令序列组成的。

- STL 或 LAD 程序共享多个变量：如果共享数据由一些相关的字节、字或双字组成，那么可以用中断禁止/允许指令（DISI 和 ENI）来控制中断程序的执行。在用户程序开始对共享存储器单元操作的地方禁止中断，一旦所有影响共享存储器单元的操作完成后，再允许中断。在访问共享存储器单元期间，中断被禁止，中断程序不能执行，因而也无法访问共享存储器单元，但这种方法导致了对中断事件响应的延迟。

在中断服务程序中调用子程序

你可以在一个中断服务程序中调用一个子程序。中断服务程序与被调用的子程序共享累加器和逻辑堆栈。

S7-200 支持的中断类型

S7-200 支持以下中断类型

- 通讯口中断：S7-200 产生的事件允许你使用程序控制通讯端口。
- I/O 中断：S7-200 对 I/O 点状态的各种变化产生中断事件。这些事件允许你对高速计速器、脉冲输出以及输入点的上升沿和下降沿作出响应。
- 时基中断：S7-200 以指定的时间间隔产生中断事件。

通讯口中断

PLC 的串行通讯口可由 LAD 或 STL 程序来控制。通讯口的这种操作模式称为自由端口模式。在自由端口模式下，用户可用程序定义波特率、每个字符位数、奇偶校验和通讯协议。利用接收和发送中断可简化程序对通讯的控制。请参看发送/接收指令以了解更多的信息。

I/O 中断

I/O 中断包含了上升沿或下降沿中断、高速计数器中断和脉冲串输出（PTO）中断。S7-200 CPU 可用输入 I0.0 至 I0.3 的上升沿或下降沿产生中断。表 9-21 给出了允许中断的输入，上升沿事件和下降沿事件可被这些输入点捕获。这些上升沿或下降沿事件可被用来指示当某个事件发生时必须引起注意的错误条件。

高速计数器中断允许响应诸如当前值等于预置值、相应于轴转动方向变化的计数方向改变和计数器外部复位等事件而产生的中断。每种高速计数器可对高速事件实时响应，而 PLC 扫描速率对这些高速事件是不能控制的。

脉冲串输出中断给出了已完成指定脉冲数输出的指示。脉冲串输出的一个典型应用是步进电机。

可以通过将一个中断程序连接到相应的 I/O 事件上来允许上述的每一个中断。

时基中断

时基中断包括定时中断和定时器 T32/T96 中断。CPU 可以支持定时中断。可以用定时中断指定一个周期性的活动。周期以 1ms 为增量单位，周期时间可从 0ms 到 255ms。对定时中断 0，把周期时间写入 SMB34；对定时中断 1，把周期时间写入 SMB35。

每当定时器溢出时，定时中断事件把控制权交给相应的中断程序。通常可用定时中断以固定的时间间隔去控制模拟量输入的采样或者执行一个 PID 回路。

当把某个中断程序连接到一个定时中断事件上，如果该定时中断被允许，那就开始计时。在连接期间，系统捕捉周期时间值，因而后来的变化不会影响周期。为改变周期时间，首先必须修改周期时间值，然后重新把中断程序连接到定时中断事件上。当重新连接时，定时中断功能清除前一次连接时的任何累计值，并用新值重新开始计时。

一旦允许，定时中断就连续地运行，指定时间间隔的每次溢出时执行被连接的中断程序。如果退出 RUN 模式或分离定时中断，则定时中断被禁止。如果执行了全局中断禁止指令，定时中断事件会继续出现，每个出现的定时中断事件将进入中断队列（直到中断允许或队列满）。请参见定时中断的例子程序。

定时器 T32/T96 中断允许及时地响应一个给定的时间间隔。这些中断只支持 1ms 分辨率的延时接通定时器（TON）和延时断开定时器（TOF）T32 和 T96。T32 和 T96 定时器在其它方面工作正常。一旦中断允许，当有效定时器的当前值等于预置值时，在 CPU 的正常 1ms 定时刷新中，执行被连接的中断程序。首先把一个中断程序连接到 T32/T96 中断事件上，然后允许该中断。

中断优先级和中断队列

在各个指定的优先级之内，CPU 按先来先服务的原则处理中断。任何时间点上，只有一个用户中断程序正在执行。一旦中断程序开始执行，它要一直执行到结束。而且不会被别的中断程序，甚至是更高优先级的中断程序所打断。当另一个中断正在处理中，新出现的中断需排队等待，以待处理。

表 6-46 给出了 3 个中断队列以及它们能够存储的中断个数。

表 6-46　每个中断队列的最大数目

队列	CPU 211、CPU 222、CPU 224	CPU 226、CPU 226XM
通讯中断队列	4	8
I/O 中断队列	16	16
定时中断队列	8	8

有时，可能有多于队列所能保存数目的中断出现，因而，由系统维护的队列溢出存储器位表明丢失的中断事件的类型。中断队列溢出位如表 6-47 所示。你应当只在中断程序中使用这些位，因为在队列变空或控制返回到主程序时，这些位会被复位。

表 6-48 给出了所有中断事件的优先级和事件号。

表 6-47　中断队列溢出标志位

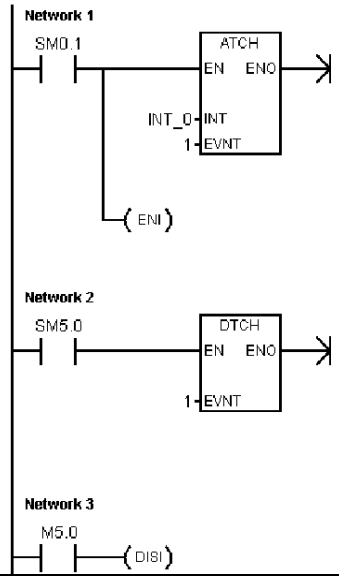
描述（0＝不溢出，1＝溢出）	SM 位
通讯中断队列溢出	SM4.0
I/O 中断队列溢出	SM4.1
定时中断队列溢出	SM4.2

表 6-48　中断事件的优先级顺序

事件号	中断描述	优先级	优先组中的优先级
8	端口 0：接收字符	通讯（*最高*）	0
9	端口 0：发送完成		0
23	端口 0：接收信息完成		0
24	端口 1：接收信息完成		1
25	端口 1：接收字符		1
26	端口 1：发送完成		1
19	PTO 0 完成中断	I/O（*中等*）	0
20	PTO 1 完成中断		1
0	上升沿，I0.0		2
2	上升沿，I0.1		3
4	上升沿，I0.2		4
6	上升沿，I0.3		5
1	下降沿，I0.0		6
3	下降沿，I0.1		7
5	下降沿，I0.2		8
7	下降沿，I0.3		9
12	HSC0 CV=PV（当前值＝预置值）		10
27	HSC0 输入方向改变		11
28	HSC0 外部复位		12
13	HSC1 CV=PV（当前值＝预置值）		13
14	HSC1 输入方向改变		14
15	HSC1 外部复位		15
16	HSC2 CV=PV		16
17	HSC2 输入方向改变		17
18	HSC2 外部复位		18
32	HSC3 CV=PV（当前值＝预置值）		19
29	HSC4 CV=PV（当前值＝预置值）		20
30	HSC4 输入方向改变		21
31	HSC4 外部复位		22
33	HSC5 CV=PV（当前值＝预置值）		23
10	定时中断 0	定时（*最低*）	0
11	定时中断 1		1
21	定时器 T32 CT=PT 中断		2
22	定时器 T96 CT=PT 中断		3

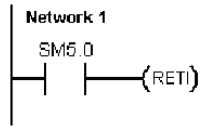
中断指令程序举例：

MAIN




```
Network 1      //首次扫描
               //1.定义 I0.0 的下降沿中断服务程序为 INT_0
               //2.全局中断允许。
LD     SM0.1
ATCH   INT_0,1
ENI
Network 2      //如果出现 I/O 错误，禁止 I0.0 的下降沿中断。
               //该程序段可选。
LD     SM5.0
DTCH   1
Network 3      //当 M5.0 接通时，禁止所有中断。
LD     M5.0
DISI
```

INT 0



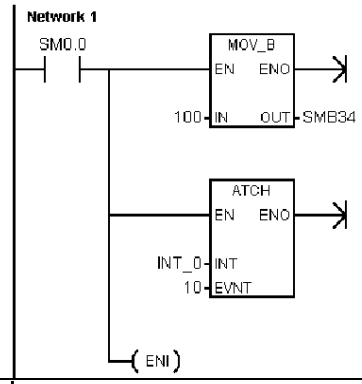

```
Network        //I0.0 的下降沿中断服务程序：
               //当有 I/O 错误时返回。
LD     SM5.0
CRETI
```

用定时中断读取模拟量的数值程序举例：

MAIN

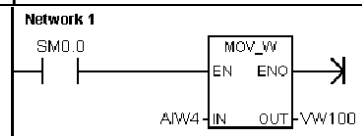

```
Network 1      //首次扫描，调用子程序 0。
LD     SM0.1
CALL   SBR_0
```

SBR 0




```
Network 1      //1.设置定时中断的时间间隔为 100ms。
               //2.连接 INT_0 到定时中断 0（事件 10）。
               //3.全局中断允许。
LD     SM0.0
MOVB   100,SMB34
ATCH   INT_0,10
ENI
```

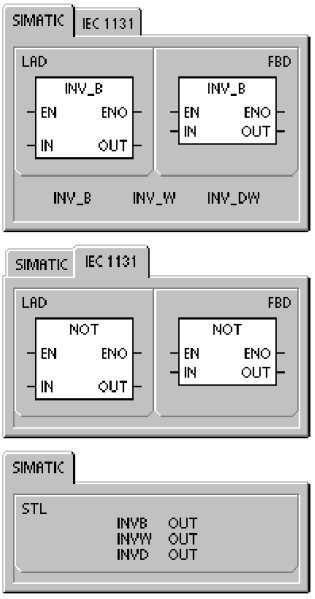
INT 0




```
Network 1      //每 100ms 读 AIW4 的值。
LD     SM0.0
MOVW   AIW4,VW100
```

逻辑操作指令

取反指令



字节、字和双字取反

字节取反（INVB）、字取反（INVW）和双字取反（INVD）指令将输入 IN 取反的结果存入 OUT 中。

使 ENO＝0 的错误条件：

- 0006（间接寻址）

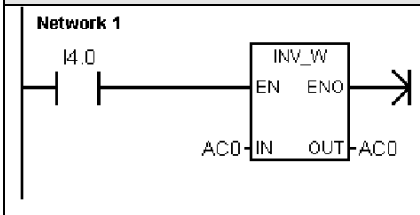
受影响的 SM 标志位：

- SM1.0（结果为 0）

表 6-49　取反指令的有效操作数

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

取反指令程序举例：

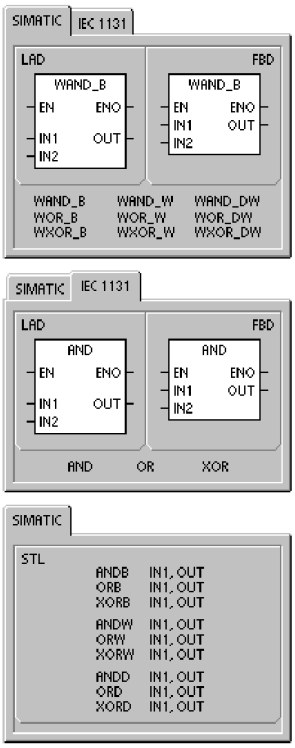



```
Network 1
LD      I4.0
INVW    AC0
```

字取反　AC0　1101 0111 1001 0101

执行后　AC0　0010 1000 0110 1010

与、或和异或指令



字节与、字与和双字与

字节与（ANDB）、字与（ANDW）和双字与（ANDD）指令将输入 IN1 和 IN2 的相应位作与操作，将结果存入 OUT 中。

字节或、字或和双字或

字节或（ORB）、字或（ORW）和双字或（ORD）指令将输入 IN1 和 IN2 的相应位作或操作，将结果存入 OUT 中。

字节异或、字节或和双字异或

字节异或（XORB）、字异或（XORW）和双字异或（XORD）指令将输入 IN1 和 IN2 的相应位作异或操作，将结果存入 OUT 中。

SM 位和 ENO

对于本页中描述的所有指令，下列情况影响 SM 位和 ENO。

使 ENO＝0 的错误条件：

- 0006（间接寻址）

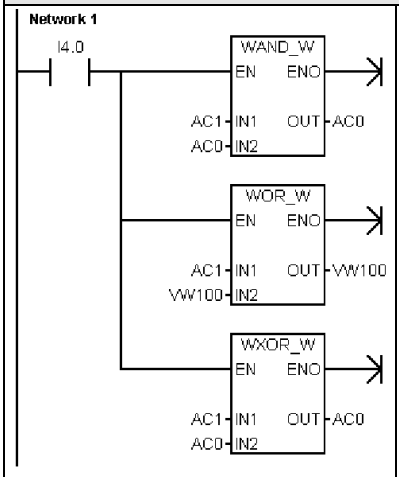
受影响的 SM 位：

- SM1.0（结果为 0）

表 6-50 与、或和异或指令的有效操作数

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
	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、＊AC、＊LD
	WORD	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、＊VD、＊AC、＊LD
	DWORD	ID、QD、VD、MD、SMD、SD、LD、AC、＊VD、＊AC、＊LD

与、或和异或指令程序举例：




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

字与

AC1 0001 1111 0110 1101

与

AC0 1101 0011 1110 0110

等于

AC0 0001 0011 0110 0100

字或

AC1 0001 1111 0110 1101

或

VW100 1101 0011 1010 0000

等于

VW100 1101 1111 1110 1101

字异或

AC1 0001 1111 0110 1101

异或

AC0 0001 0011 0110 0100

等于

AC0 0000 1100 0000 1001

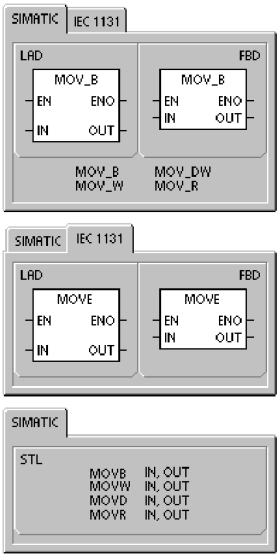
传送指令

字节、字、双字或者实数传送

字节传送（MOVB）、字传送（MOVW）、双字传送（MOVD）和实数传送指令在不改变原值的情况下将 IN 中的值传送到 OUT。

使用双字传送指令可以创建一个指针。要得到更多信息，请参考第 4 章中指针和间接寻址一节。

对于 IEC 传送指令，输入和输出的数据类型可以不同，但数据长度必须相同。



传 ENO＝0 的错误条件：

- 0006（间接寻址）

表 6-51 传送指令的有效操作数

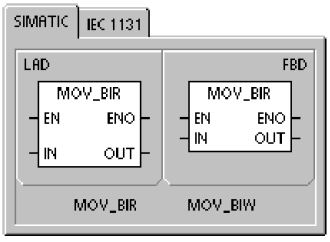
输入/输出	数据类型	操作数
IN	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数
	WORD、INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*AC、*LD、常数
	DWORD、DINT	ID、QD、VD、MD、SMD、SD、LD、AC、HC、&IB、&QB、&VB、&MB、&SB、&T、&C、*VD、*LD、*AC、常数
	REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC、常数
OUT	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC
	WORD、INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AQW、*VD、*LD、*AC
	DWORD、DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC

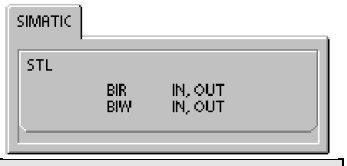
字节立即传送（读和写）

字节立即传送指令允许你在物理 I/O 和存储器之间立即传送一个字节数据。

字节立即读指令（BIR）读物理输入 IN，并将结果存入 OUT 中，但过程映象寄存器并不刷新。

字节立即写指令（BIW）从存储器 IN 读取数据，写入物理输出 OUT，同时刷新相应的过程映象区。





使 ENO＝0 的错误条件：

- 0006（间接寻址）
- 不能访问扩展模块

表 6-52 字节立即读指令的有效操作数

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

表 6-53 字节立即写指令的有效操作数

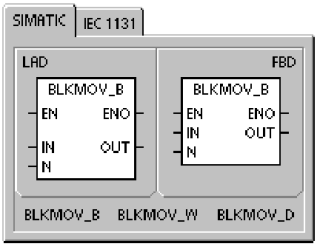
输入/输出	数据类型	操作数
IN	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数
OUT	BYTE	QB、*VD、*LD、*AC

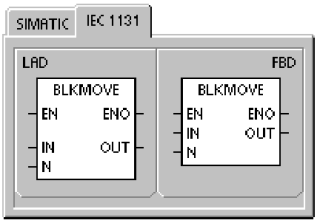
块传送指令

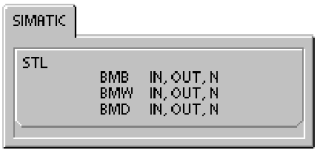
字节、字、双字的块传送

字节块传送（BMB）、字块传送（BMW）和双字块传送（BMD）指令传送指定数量的数据到一个新的存储区，数据的起始地址 IN，数据长度为 N 个字节、字或者双字，新块的起始地址为 OUT。

N 的范围从 1 到 255。







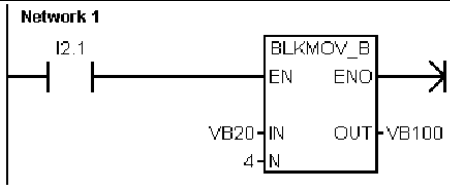
使 ENO＝0 的错误条件：

- 0006（间接寻址）
- 0091（操作数超出范围）

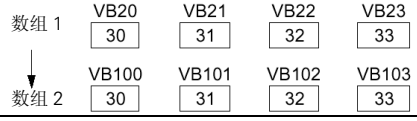
表 6-54 块传送指令的有效操作数

输入/输出	数据类型	操作数
IN	BYTE WORD、INT DWORD、DINT	IB、QB、VB、MB、SMB、SB、LB、＊VD、＊LD、＊AC IW、QW、VW、SMW、SW、T、C、LW、AIW、＊VD、＊LD、＊AC ID、QD、VD、MD、SMD、SD、LD、＊VD、＊LD、＊AC
OUT	BYTE WORD、INT DWORD、DINT	IB、QB、VB、MB、SMB、SB、LB、＊VD、＊LD、＊AC IW、QW、VW、MW、SMW、SW、T、C、LW、AQW、＊VD、＊LD、＊AC ID、QD、VD、MD、SMD、SD、LD、＊VD、＊LD、＊AC
N	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、常数、＊VD、＊LD、＊AC

块传送指令程序举例

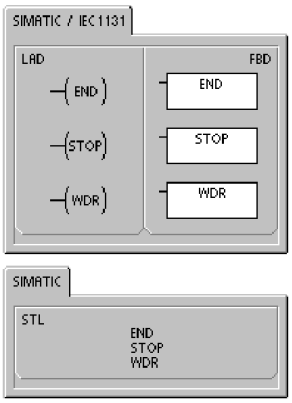



```
Network 1     //将数组 1（VB20 到 VB23）传送至数组 2
              //（VB100 到 VB103）
LD            I2.1
BMB           VB20，VB100，4
```



程序控制指令

条件结束



条件结束指令（END）根据前面的逻辑关系终止当前扫描周期。可以在主程序中使用条件结束指令，但是不能在子程序和中断服务程序中使用该指令。

停止

停止指令（STOP）导致 CPU 从 RUN 到 STOP 模式从而可以立即终止程序的执行。

如果 STOP 指令在中断程序中执行，那么该中断立即终止，并且忽略所有挂起的中断，继续扫描程序的剩余部分。在本次扫描的最后，完成 CPU 从 RUN 到 STOP 的转变。

看门狗复位

看门狗复位（Watchdog Reset）指令（WDR）允许 CPU 的看门狗定时器重新被触发。在没有看门狗错误的情况下，这样可以增加一次扫描所允许的时间。

使用 WDR 指令时要小心，因为如果你用循环指令去阻止扫描完成或过度的延迟扫描完成的时间，那么在终止本次扫描之前，下列操作过程将被禁止：

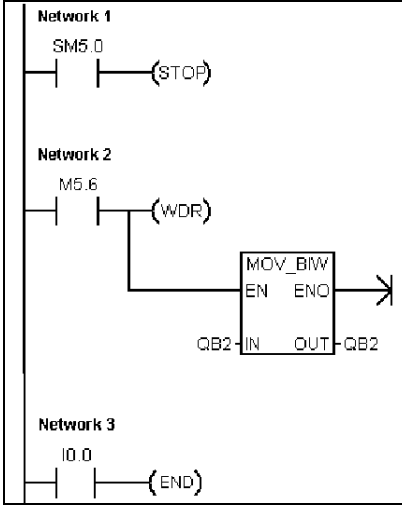
- 通讯（自由端口方式除外）
- I/O 更新（立即 I/O 除外）
- 强制更新
- SM 位更新（SM0，SM5～SM29 不能被更新）
- 运行时间诊断
- 由于扫描时间超过 25 秒，10ms 和 100ms 定时器将不会正确累计时间。
- 在中断程序中的 STOP 指令

提示：

如果希望程序的扫描周期超过 500ms，或者在中断事件发生时有可能使程序的扫描周期超过 500ms 时，你应该使用看门狗复位指令来重新触发看门狗定时器。

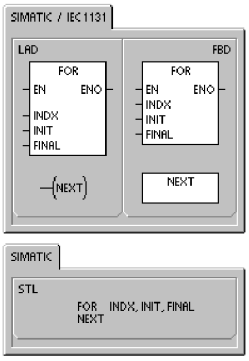
每次使用看门狗复位指令，你应该对每个扩展模块的某一个输出字节使用一个立即写指令来复位每个扩展模块的看门狗。如果你使用了看门狗复位指令允许程序的执行有一个很长的扫描时间，此时将 S7-200 的模式开关切换到 STOP 位置，则在 1.4 秒内，CPU 转到 STOP 方式。

停止、结束和看门狗复位指令程序举例




```
Network 1     //当检测到 I/O 错误时，强制切换到
              // STOP 模式。
LD        SM5.0
STOP

Network 2     //当 M5.6 接通时，允许扫描周期扩展：
              // 1. 重新触发 S7-200 CPU 的看门狗。
              // 2. 重新触发第一个输出模块的看门狗。
LD        M5.6
WDR
BIW       QB2,QB2

Network 3     //当 I0.0 接通时，终止当前扫描周期。
LD        I0.0
END
```

For-Next 循环指令



FOR 和 NEXT 指令可以描述需重复进行一定次数的循环体。每条 FOR 指令必须对应一条 NEXT 指令。FOR 和 NEXT 循环嵌套（一个 FOR 和 NEXT 循环在另一个 FOR 和 NEXT 循环之内）深度可达 8 层。

FOR-NEXT 指令执行 FOR 指令和 NEXT 指令之间的指令。必须指定计数值或者当前循环次数 INDX、初始值（INIT）和终止值（FINAL）。

NEXT 指令标志着 FOR 循环的结束。

使 ENO＝0 的错误条件：

- 0006（间接寻址）

如果允许 FOR/NEXT 循环，除非在循环内部修改了终值，循环体就一直循环执行直到循环结束。当 FOR/NEXT 循环执行的过程中可以修改这些值。

当循环再次允许时，它把初始值拷贝到 INDX 中（当前循环次数）。

当下一次允许时，FOR/NEXT 指令复位它自己。

例如，给定初值（INIT）为 1，终值（FINAL）为 10，那么随着当前计数值（INDX）从 1 增加到 10，FOR 与 NEXT 之间的指令被执行 10 次。

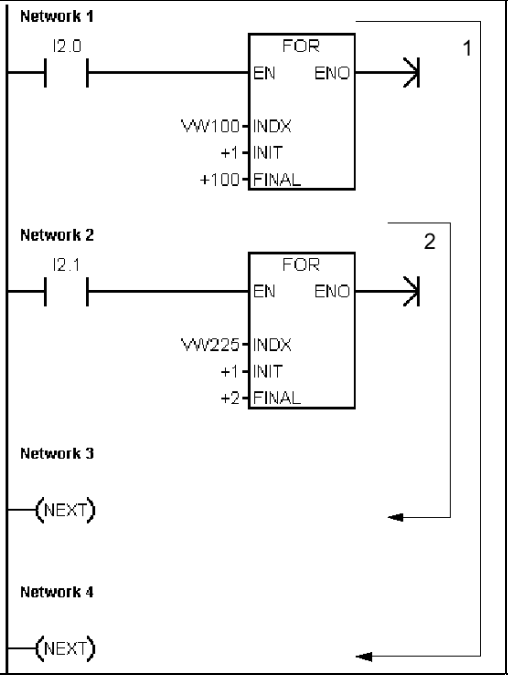
如果初值大于终值，那么循环体不被执行。每执行一次循环体，当前计数值增加 1，并且将其结果同终值作比较，如果大于终值，那么终止循环。

如果程序进入 FOR-NEXT 循环时，栈顶值为 1，则当程序退出 FOR-NEXT 循环时，栈顶值也将为 1。

表 6-55　FOR-NEXT 指令的有效操作数

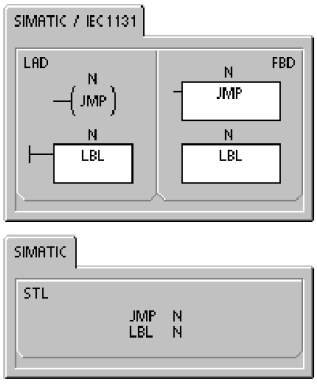
输入/输出	数据类型	操作数
INDX	INT	IW、QW、VW、MW、SMW、SW、T、C、LW、AC、*VD、*LD、*AC
INIT、FINAL	INT	VW、IW、QW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*AC、常数

FOR-NEXT 循环指令程序举例：




```
Network 1      //当 I2.0 接通时，外循环（标识
               //1）执行 100 次。
LD       I2.0
FOR      VW100,+1,+100

Network 2      //当 I2.1 接通时，外循环
               //每执行一次，内循环执行两次。
LD       I2.1
FOR      VW225,+1,+2

Network 3      //回路 2 结束。
NEXT

Network 4      //回路 1 结束。
NEXT
```

跳转指令



跳转指令可以使程序流程跳转到指定的标号 N 处的程序分支。

标号指令标记跳转目的地的位置 N。

你可以在主程序、子程序或者中断服务程序中，使用跳转指令。跳转和与之相应的标号指令必须位于同一段程序代码（无论是主程序、子程序还是中断服务程序）。

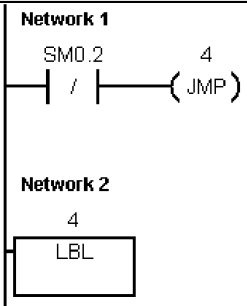
不能从主程序跳到子程序或中断程序，同样不能从子程序或中断程序跳出。

可以在 SCR 程序段中使用跳转指令，但相应的标号指令必须也在同一个 SCR 段中。

表 6-56　跳转指令的有效操作数

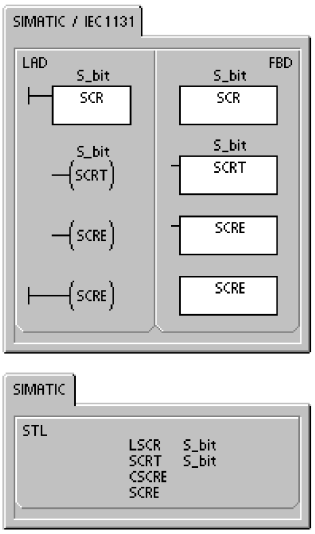
输入/输出	数据类型	操作数
N	WORD	常数（0 到 255）

跳转到标号指令程序举例：




```
Network 1     //如果掉电保持的数据没有丢失，跳
              //转到 LBL4。
LDN     SM0.2
JMP     4
Network 2
LBL     4
```

顺控继电器（SCR）指令



SCR 指令使你能够按照自然工艺段在 LAD、FBD 或 STL 中编制状态控制程序。

无论如何，由一系列操作组成的应用程序都会反复执行，而 SCR 可以使程序更加结构化，以至于直接针对应用。这样可以使编程和调试更加便捷。

装载 SCR 指令（LSCR）将 S 位的值装载到 SCR 和逻辑堆栈中。

SCR 堆栈的结果值决定是否执行 SCR 程序段。SCR 堆栈的值会被复制到逻辑堆栈中，因此可以直接将盒或者输出线圈连接到左侧的能流线上而不经过中间触点。

限定：

SCR 指令的限制如下：

- 不能把同一个 S 位用于不同程序中。例如：如果在主程序中用了 S0.1，在子程序中就不能再使用它。
- 在 SCR 段之间不能使用 JMP 和 LBL 指令，就是说不允许跳入、跳出。可以在 SCR 段附近使用跳转和标号指令或者在段内跳转。
- 在 SCR 段中不能使用 END 指令。

表 6-57　顺控继电器指令的有效操作数

输入/输出	数据类型	操作数
S_bit	BOOL	S

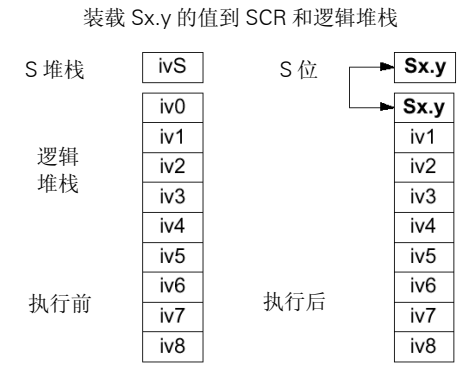


图 6-31　LSCR 对逻辑堆栈的影响

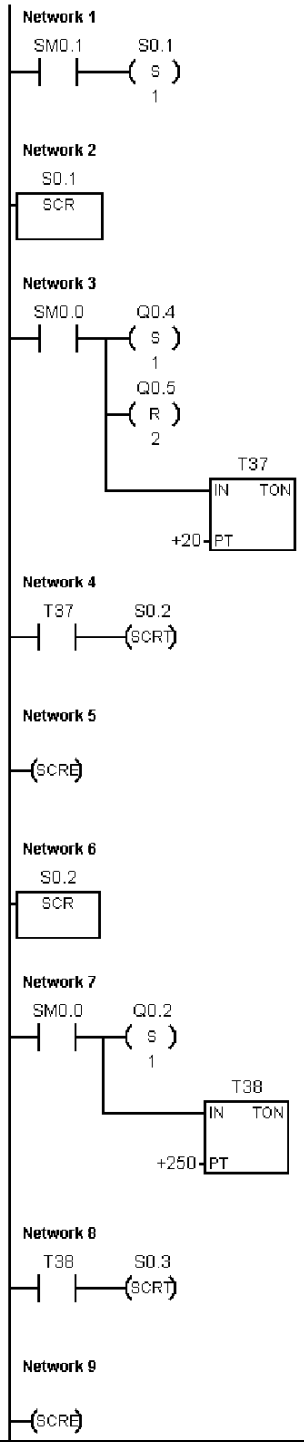
图 6-31 给出了 S 堆栈和逻辑堆栈以及执行 LSCR 指令产生的影响。以下是对顺控继电器指令的正确理解：

- 装载 SCR 指令（LSCR）标志着 SCR 段的开始，SCR 结束指令则标志着 SCR 段的结束。在装载 SCR 指令与 SCR 结束指令之间的所有逻辑操作的执行取决于 S 堆栈的值。而在 SCR 结束指令和下一条装载 SCR 指令之间的逻辑操作则不依赖于 S 堆栈的值。
- SCR 传输指令（SCRT）将程序控制权从一个激活的 SCR 段传递到另一个 SCR 段。执行 SCRT 指令可以使当前激活的程序段的 S 位复位，同时使下一个将要执行的程序段的 S 位置位。

  在 SCRT 指令指行时，复位当前激活的程序段的 S 位并不会影响 S 堆栈。SCR 段会一直保持能流直到退出。
- SCR 条件结束指令(CSCRE)可以使程序退出一个激活的程序段而不执行 CSCRE 与 SCRE 之间的指令。CSCRE 指令不影响任何 S 位，也不影响 S 堆栈。

在以下实例中，首次扫描位 SM0.1 置位 S0.1，从而在首次扫描中，激活状态 1。延时 2 秒后，T37 导致切换到状态 2。切换使状态 1 停止，激活状态 2。

顺控继电器指令程序举例：




```
Network 1     //在首次扫描使能状态 1。
LD        SM0.1
S         S0.1,1

Network 2     //状态 1 控制开始。
LSCR      S0.1

Network 3     //控制第一条街的信号：
              //1. 置位：接通红灯。
              //2. 复位：关断黄灯和绿灯。
              //3. 启动 2 秒定时器。
LD        SM0.0
S         Q0.4,1
R         Q0.5,2
TON       T37,+20

Network 4     //延时 2 秒后，切换到状态 2。
LD        T37
SCRT      S0.2

Network 5     //状态 1 的 SCR 区结束。
SCRE

Network 6     //状态 2 的控制区开始。
LSCR      S0.2

Network 7     //控制第二条街的信号：
              //1. 置位：接通绿灯。
              //2. 启动 25 秒定时器。
LD        SM0.0
S         Q0.2,1
TON       T38,+250

Network 8     //延时 25 秒后，切换到状态 3。
LD        T38
SCRT      S0.3

Network 9     //状态 2 的 SCR 区结束。
SCRE
```

分支控制

在许多实例中，一个顺序控制状态流必须分成两个或多个不同分支控制状态流。当一个控制状态流分离成多个分支时，所有的分支控制状态流必须同时激活，如图 6-32 所示。

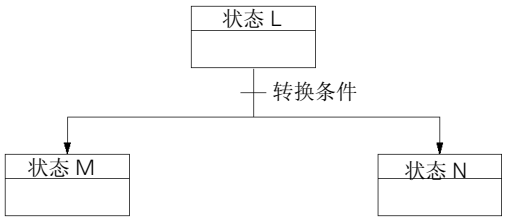
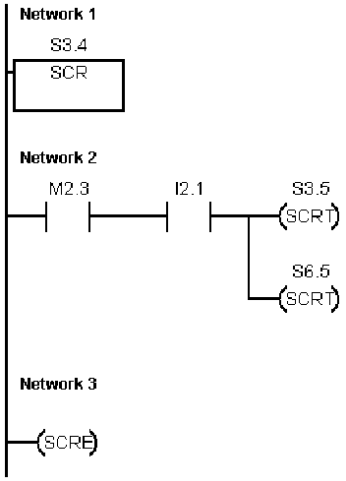


图 6-32　控制流的分支

在同一个转移条件下，使用多条 SCRT 指令可以在一段 SCR 程序中实现控制流的分支，如下例所示。

控制流分支程序举例




```
Network 1     //状态 L 控制区开始
LSCR    S3.4

Network 2
LD      M2.3
A       I2.1
SCRT    S3.5          //切换到状态 M
SCRT    S6.5          //切换到状态 N

Network 3     //状态 L 的状态区结束
SCRE
```

合并控制

与分支控制的情况类似，两个或者多个分支状态流必须合并为一个状态流。当多个状态流汇集成一个时，我们称之为合并。当控制流合并时，所有的控制流必须都完成，才能执行下一个状态。图 6-33 给出了两个控制流合并的示意图。

在 SCR 程序中，通过从状态 L 转到状态 L′，以及从状态 M 转到状态 M′的方法实现控制流的合并。当状态 L′、M′的 SCR 使能位为真时，即可激活状态 N，如下例所示。

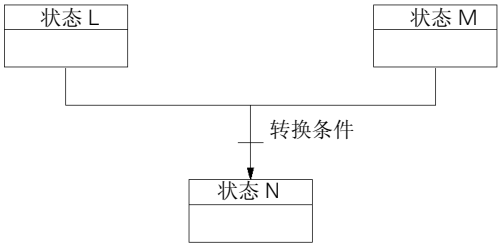
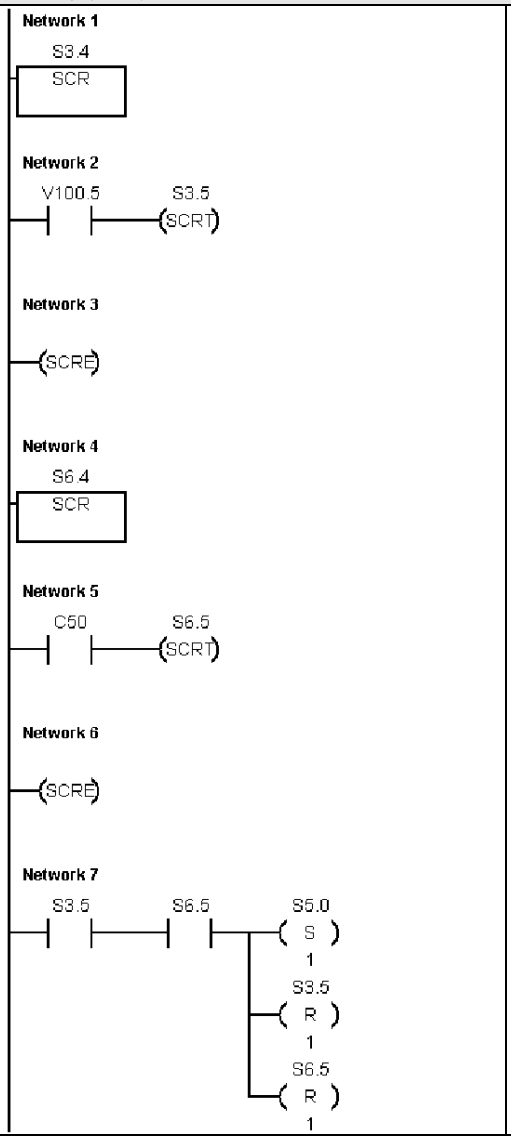


图 6-33　控制流的合并

控制流合并程序举例：




```
Network 1     //状态 L 控制区开始
LSCR    S3.4

Network 2     //切换到状态 L'
LD      V100.5
SCRT    S3.5

Network 3     //状态 L SCR 区的结束
SCRE

Network 4     //状态 M 控制区开始
LSCR    S6.4

Network 5     //切换到状态 M'
LD      C50
SCRT    S6.5

Network 6     //状态 M SCR 区的结束
SCRE

Network 7     //当状态 L'和 M'同时激活时：
              //1. 使能状态 N（S5.0）
              //2. 复位状态 L'（S3.5）
              //3. 复位状态 M'（S6.5）
LD      S3.5
A       S6.5
S       S5.0,1
R       S3.5,1
R       S6.5,1
```

在有些情况下，一个控制流可能转入多个可能的控制流中的某一个。到底进入哪一个，取决于控制流前面的转移条件，哪一个首先为真，如图 6-34 所示。

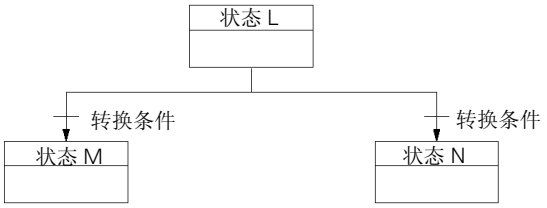
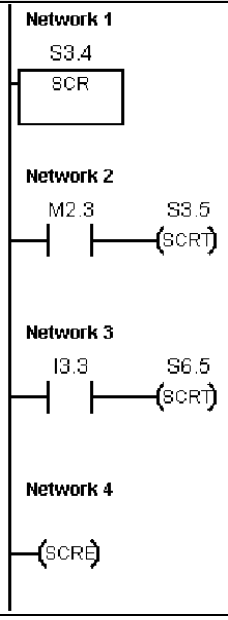


图 6-34　条件转换控制流分支

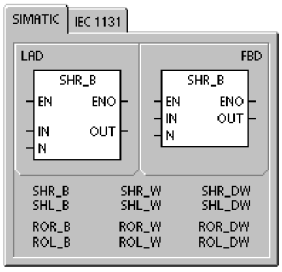
条件转换程序举例：

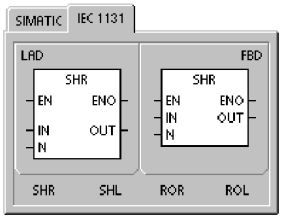


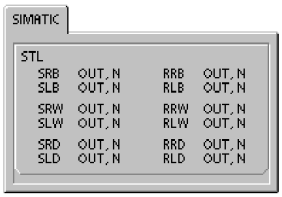

```
Network 1      //状态 L 控制区开始
LSCR      S3.4

Network 2      //切换到 M 状态
LD        M2.3
SCRT      S3.5

Network 3      //切换到 N 状态
LD        I3.3
SCRT      S6.5

Network 4      //状态 L 的 SCR 区结束
SCRE
```

移位和循环指令

右移和左移指令







移位指令将输入值 IN 右移或者左移 N 位，并将输出结果装载到 OUT 中。

移位指令对移出的位自动补零。如果位数 N 大于或等于最大允许值（对于字节操作为 8，对于字操作为 16，对于双字操作为 32），那么移位操作的次数为最大允许值。如果移位次数大于 0，溢出标志位（SM1.1）上就是最近移出的位值。如果移位操作的结果为 0，零存储器位（SM1.0）置位。

字节操作是无符号的。对于字和双字操作，当使用符号数据类型时，符号位也被移动。

使 ENO＝0 的错误条件：

- 0006（间接寻址）

受影响的 SM 标志位：

- SM1.0（结果为 0）
- SM1.1（溢出）

循环右移和循环左移指令

循环移位指令将输入值 IN 循环右移或者循环左移 N 位，并将输出结果装载到 OUT 中。如果位数 N 大于或者等于最大允许值（对于字节操作为 8，对于字操作为 16，对于双字操作为 32），S7-200 在执行循环移位之前，会执行取模操作，得到一个有效的移位次数。取模操作的结果对于字节操作为 0 到 7，对于字操作为 0 到 15，对于双字操作为 0 到 31。如果移位次数为 0，循环移位指令不执行。如果循环移位指令执行，最后一位的值会复制到溢出标志位（SM1.1）。

如果移位次数不是 8（对于字节操作）、16（对于字操作）和 32（对于双字操作）的整数倍，最后被移出的位会被复制到溢出标志位（SM1.1）。当循环移位的结果为 0 时，结果为零标志位（SM1.0）被置位。

字节操作是无符号的。对于字和双字操作，当使用符号数据类型时，符号位也被移位。

使 ENO＝0 的错误条件：

- 0006（间接寻址）

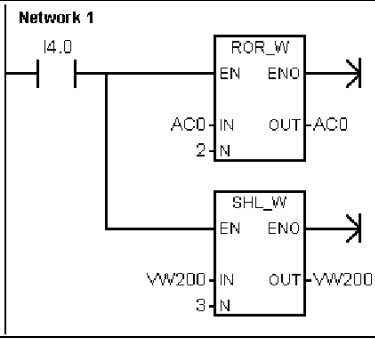
受影响的 SM 标志位：

- SM1.0（结果为 0）
- SM1.1（溢出）

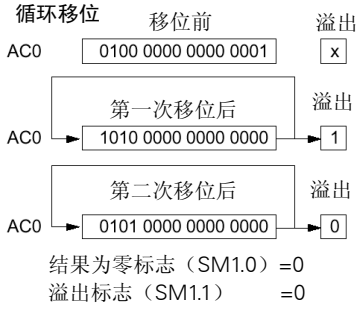
表 6-58　移位和循环移位指令的有效操作数

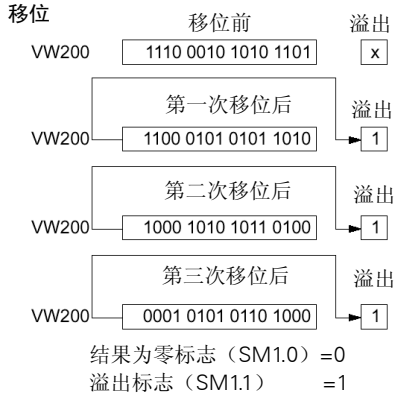
输入/输出	数据类型	操作数
IN	BYTE WORD DWORD	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数 IW、QW、VW、MW、SMW、SW、T、C、LW、AC、AIW、*VD、*LD、*AC、常数 ID、QD、VD、MD、SMD、SD、LD、AC、HC、*VD、*LD、*AC、常数
OUT	BYTE WORD DWORD	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC IW、QW、VW、MW、SMW、SW、T、C、LW、AC、*VD、*LD、*AC ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*LD、*AC
N	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*LD、*AC、常数

移位和循环移位指令程序举例




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



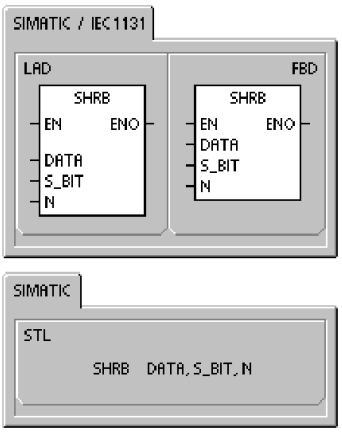


移位寄存器指令

移位寄存器指令将一个数值移入移位寄存器中。移位寄存器指令提供了一种排列和控制产品流或者数据的简单方法。使用该指令时，每个扫描周期，整个移位寄存器移动一位。

移位寄存器指令把输入的 DATA 数值移入移位寄存器。其中，S_BIT 指定移位寄存器的最低位，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）。

SHRB 指令移出的每一位都被放入溢出标志位（SM1.1）。这条指令的执行取决于最低有效位（S_BIT）和由长度（N）指定的位数。



使 ENO＝0 的错误条件：

- 0006（间接寻址）
- 0091（操作数超出范围）
- 0092（计数区错误）

受影响的 SM 标志位：

- SM1.1（溢出）

表 6-59　移位寄存器指令的有效操作数

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

位移位寄存器的最高位（MSB.b）可通过下面公式计算求得：

MSB.b＝*［（S-IT 的字节号）＋（［N-1］＋（S-BIT 的位号））/8 的商］.［除 8 的余数］*

例如，如果 S-BIT 是 V33.4，N 是 14，那么 MSB.b 是 V35.1，或：

MSB.b＝V33+（[14]-1+4）/8

＝V33+17/8

＝V33+2（余数为 1）

＝V35.1

当反向移动时，N 为负值，输入数据从最高位移入，最低位（S_BIT）移出。移出的数据放在溢出标志位（SM1.1）中。

当正向移动时，N 为正值，输入数据从最低位（S_BIT）移入，最高位移出。移出的数据放在溢出标志位（SM1.1）中。

移位寄存器的最大长度为 64 位，可正可负。

图 6-35 中给出了 N 为正和负两种情况下的移位过程。

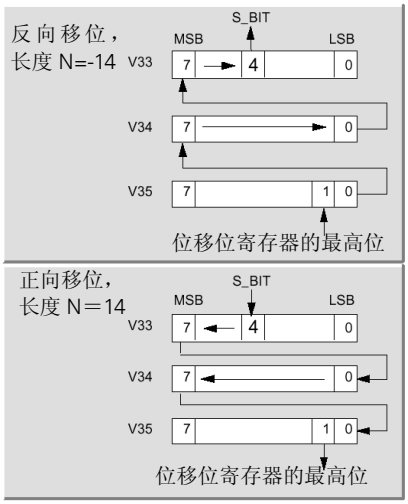
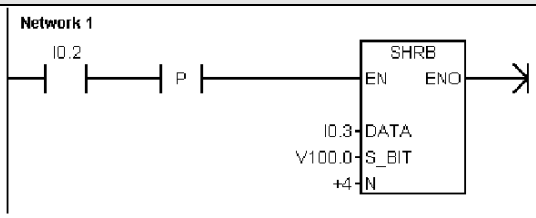
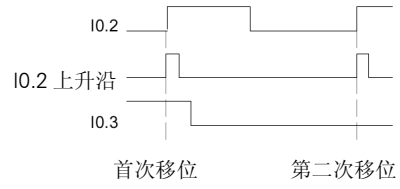


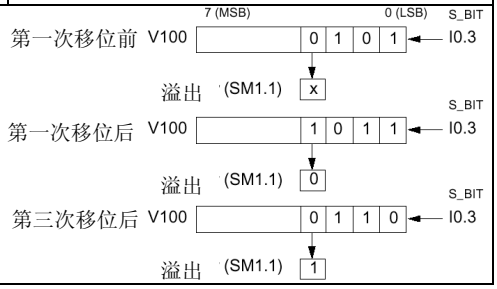
图 6-35　移位寄存器的入口和出口

移位指令程序举例：

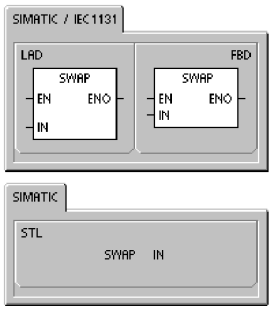



```
Network 1
LD      I0.2
EU
SHRB    I0.3,V100.0,+4
```





字节交换指令



字节交换指令用来交换输入字 IN 的高字节和低字节。

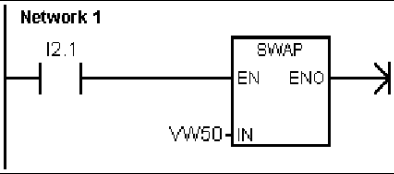
使 ENO＝0 的错误条件：

- 0006（间接寻址）

表 6-60　字节交换指令的有效操作数

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

交换指令程序举例：

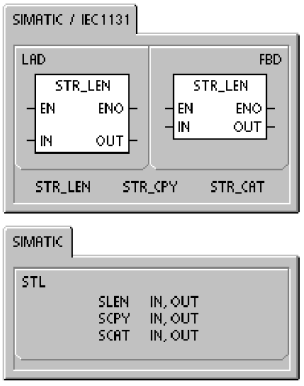



```
Network 1
LD      I2.1
SWAP    VW50
```

交换 VW50 [D6 C3] → VW50 [C3 D6]

字符串指令

字符串长度



字符串长度指令（SLEN）返回 IN 中指定的字符串的长度值。

字符串复制

字符串复制指令（SCPY）将 IN 中指定的字符串复制到 OUT 中。

字符串连接

字符串连接指令（SCAT）将 IN 中指定的字符串连接到 OUT 中指定字符串的后面。

SM 位和 ENO

对于字符串长度、字符串复制和字符串连接指令，下列条件影响 ENO。

使 ENO=0 的错误条件：

- 0006（间接寻址）
- 0091（操作数超出范围）

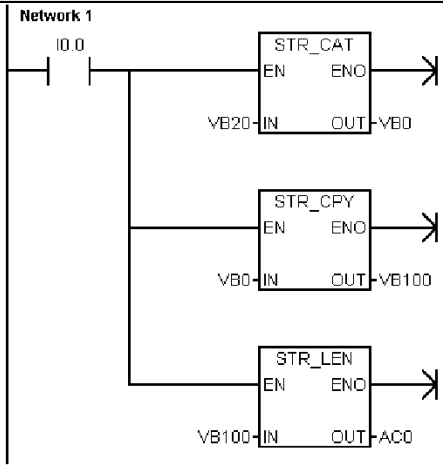
表 6-61　字符串长度指令的有效操作数

输入/输出	数据类型	操作数
IN	BYTE（字符）	VB、LB、＊VD、＊LD、＊AC
OUT	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、＊VD、＊LD、＊AC

表 6-62　字符串复制和字符串连接

输入/输出	数据类型	操作数
IN、OUT	BYTE（字符）	VB、LB、＊VD、＊LD、＊AC

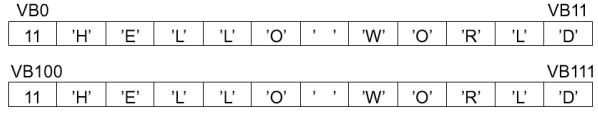
字符串连接、字符串复制和字符串长度指令程序举例




```
Network 1      //1.将 VB20 中的字符串连接到 VB0
               //中字符串的后面。
               //2.将 VB0 中的字符串复制到 VB100
               //中。
               //3.得到 VB100 中存储的字符串的
               //长度。
LD      I0.0
SCAT    VB20,VB0
STRCPY  VB0,VB100
STRLEN  VB100,AC0
```

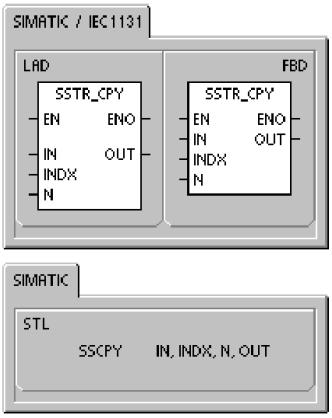
程序执行前

程序执行后



从字符串中复制子字符串

从字符串中复制子字符串指令（SSCPY）从 INDX 指定的字符号开始，将 IN 中存储的字符串中的 N 个字符复制到 OUT 中。



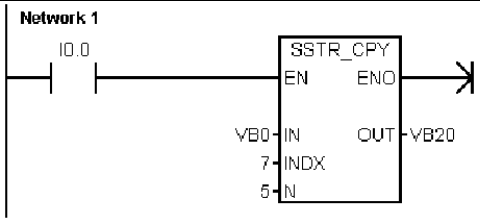
使 ENO=0 的错误条件：

- 0006（间接寻址）
- 0091（操作数超出范围）
- 009B（INDX＝0）

表 6-63　从字符串中复制子字符串指令

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

从字符串中复制子字符串指令程序举例




```
Network 1     //从 VB0 中字符串的第 7 个字符开始，
              //复制 5 个字符到 VB20 开始的新字符串。
LD      I0.0
SSCPY   VB0,7,5,VB20
```

程序执行前

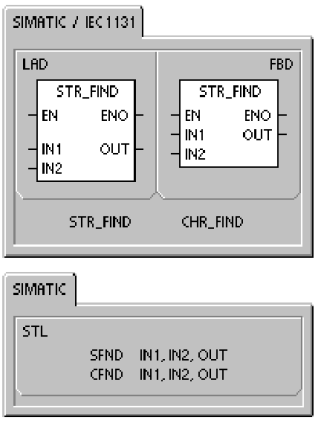
VB0											VB11
11	'H'	'E'	'L'	'L'	'O'	' '	'W'	'O'	'R'	'L'	'D'

程序执行后

VB20					VB25
5	'W'	'O'	'R'	'L'	'D'

字符串搜索

字符串搜索指令（SFND）在 IN1 字符串中寻找 IN2 字符串。由 OUT 指定搜索的起始位置。如果在 IN1 中找到了与 IN2 中字符串相匹配的一段字符，则 OUT 中会存入这段字符中首个字符的位置。如果没有找到，OUT 被清 0。



使 ENO＝0 的错误条件：

- 0006（间接寻址）
- 0091（操作数超出范围）
- 009B（INDX＝0）

字符搜索

字符搜索指令（CFND）在 IN1 字符串中寻找 IN2 字符串中的任意字符。由 OUT 指定搜索的起始位置。如果找到了匹配的字符，字符的位置被写入 OUT 中。如果没有找到，OUT 被清 0。

使 ENO＝0 的错误条件：

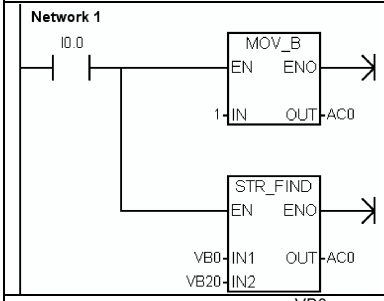
- 0006（间接寻址）
- 0091（操作数超出范围）
- 009B（INDX＝0）

表 6-64　字符串搜索和字符搜索指令的有效操作数

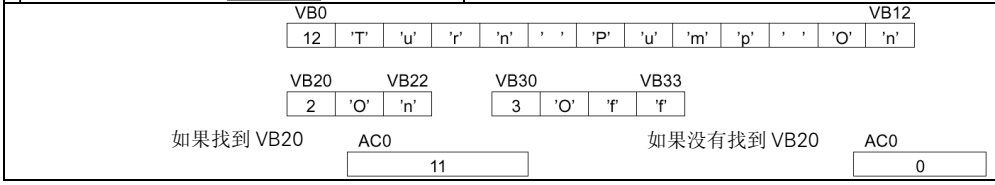
输入/输出	数据类型	操作数
IN1、IN2	BYTE（字符）	VB、LB、＊VD、＊LD、＊AC
OUT	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、＊VD、＊LD、＊AC

字符串搜索指令程序举例

在以下例子中，用存储在 VB0 中的字符串作为泵的启/停命令。字符串‘On’存储在 VB20 中，字符串‘Off’存储在 VB30 中。搜索结果在 AC0 中(OUT 参数)。如果结果不是 0，就说明在命令字符串中找到了字符串‘On’(VB12)。

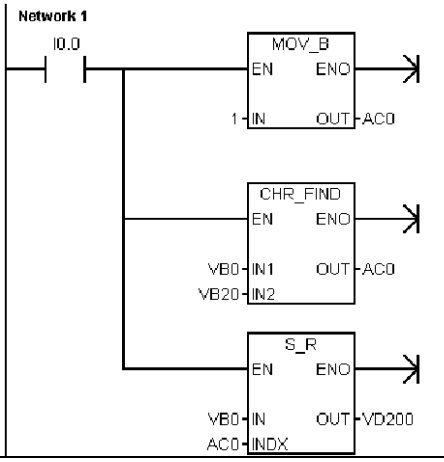



```
Network 1     //1. 置 AC0 为 1。
              //（AC0 用作 OUT 参数。）
              //2. 在 VB0 字符串中搜索 VB20 字符串(‘On’)，
              //从首个字符开始（AC0＝1）。
LD      I0.0
MOVB    1,AC0
SFND    VB0,VB20,AC0
```



字符搜索指令程序举例

在以下例子中，存储在 VB0 的字符串包含温度值。存储在 VB20 中的字符串包括所有的数字（包括＋和－），用于识别字符串中的温度值。该例子程序在字符串中找到数字的启始位置，并将其转换为实数，温度值存放在 VD200 中。



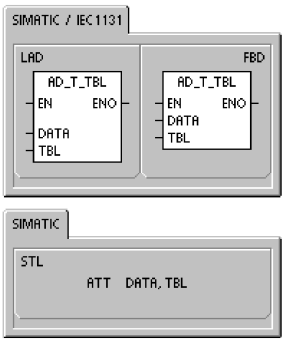

```
Network 1     //1. 置 AC0 为 1。
              //（AC0 用作 OUT 参数并指向字符串的首个字符。）
              //2. 在 VB0 字符串中寻找数字字符。
              //3. 将字符串转换为实数。
LD      I0.0
MOVB    1,AC0
CFND    VB0,VB20,AC0
STR     VB0,AC0,VD200
```

表指令

填表

ATT 指令向表（TBL）中增加一个数值（DATA）。表中第一个数是最大填表数（TL），第二个数是实际填表数（EC），指出已填入表的数据个数。新的数据填加在表中上一个数据的后面。每向表中填加一个新的数据，EC 会自动加 1。

一个表最多可以有 100 条数据。



使 ENO＝0 的错误条件：

- SM1.4（表溢出）
- 0006（间接寻址）
- 0091（操作数超出范围）

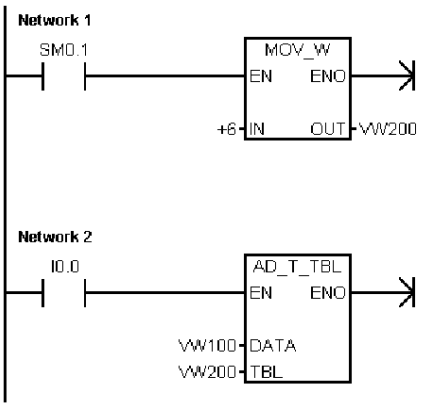
受影响的 SM 标志位：

- 如果表出现溢出，SM1.4 会置 1。

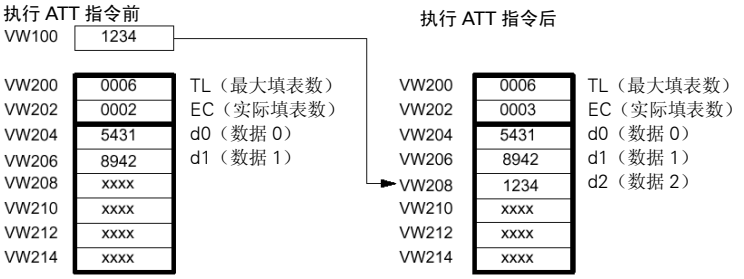
表 6-65　表指令的有效操作数

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

填表指令程序举例：

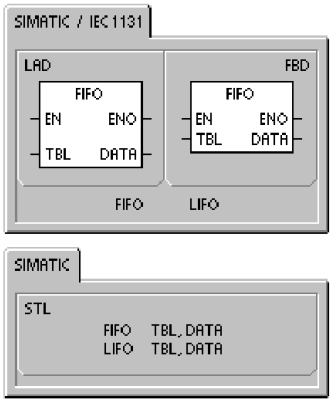



```
Network 1     //装载表的最大长度
LD        SM0.1
MOVW      +6,VW200
Network 2
LD        I0.0
ATT       VW100,VW200
```



先进先出和后进先出

一个表可以有最多 100 条数据。



先进先出

先进先出（FIFO）指令从表（TBL）中移走第一个数据，并将此数输出到 DATA。剩余数据依次上移一个位置。每执行一条本指令，表中的实际填表数（EC）减 1。

后进先出

后进先出（LIFO）指令从表（TBL）中移走最后一个数据，并将此数输出到 DATA。每执行一条本指令，表中的实际填表数（EC）减 1。

使 ENO=0 的错误条件：

- SM1.5（表空）
- 0006（间接寻址）
- 0091（操作数超出范围）

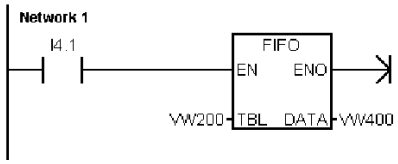
受影响的 SM 标志位：

- 当你试图从一个空表中删除一条数据时，SM1.5 会置 1。

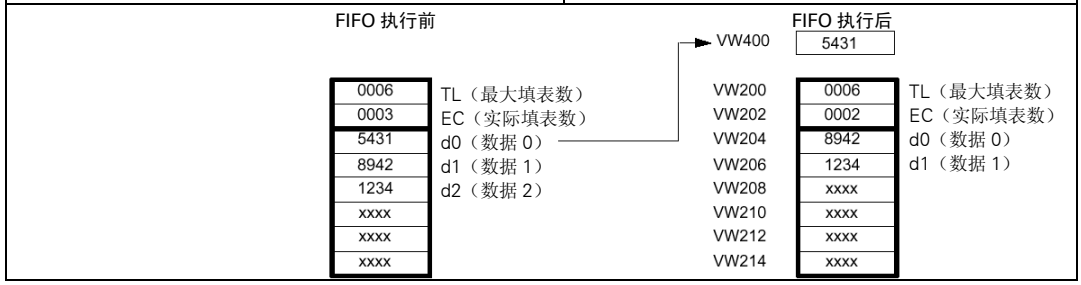
表 6-66　先进先出和先进后出指令的有效操作数

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

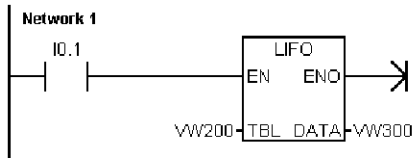
先进先出指令程序举例：



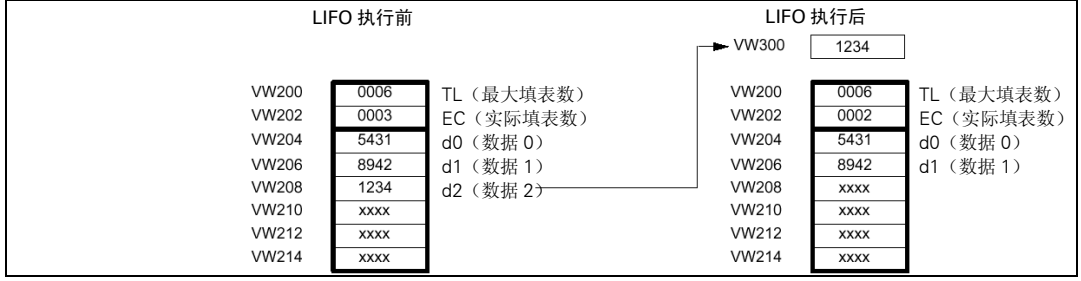

```
Network 1
LD      I4.1
FIFO    VW200，VW400
```



后进先出指令程序举例：

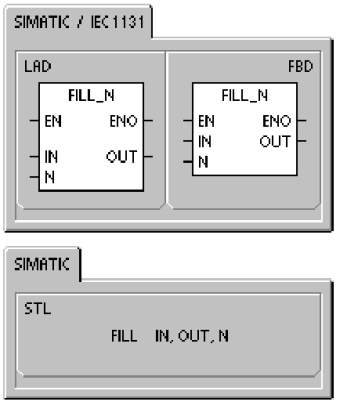



```
Network 1
LD      I0.1
LIFO    VW200，VW300
```



存储器填充

存储器填充指令（FILL）用输入值（IN）填充从输出（OUT）开始的 N 个字的内容。N 可取 1～255 之间的整数。



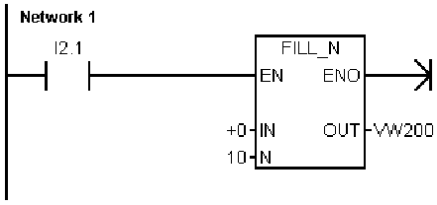
使 ENO＝0 的错误条件：

- 0006（间接寻址）
- 0091（操作数超出范围）

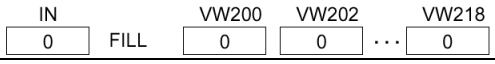
表 6-67　存储器填充指令的有效操作数

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

存储器填充指令程序举例：




```
Network 1
LD      I2.1
FILL    +0，VW200,10
```

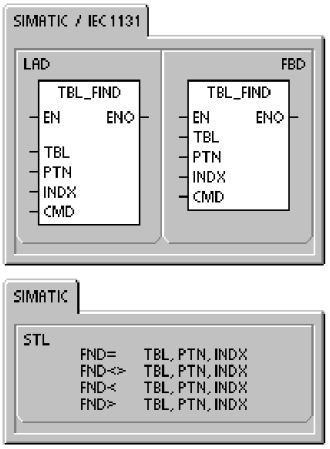


查表

查表指令从 INDX 开始搜索表（TBL），寻找符合 PTN 和条件（=、<>、<或>）的数据。命令参数 CMD 是一个 1～4 的数值，分别代表=、<>、<和>。

如果发现了一个符合条件的数据，那么 INDX 指向表中该数的位置。为了查找下一个符合条件的数据，在激活查表指令前，必须先对 INDX 加 1。如果没有发现符合条件的数据，那么 INDX 等于 EC。

一个表可以有最多 100 条数据。数据条标号从 0 到 99。



使 ENO＝0 的错误条件：

- 0006（间接寻址）
- 0091（操作数超出范围）

表 6-68　查表指令的有效操作数

输入/输出	数据类型	操作数
TBL	WORD	IW、QW、VW、MW、SMW、T、C、LW、＊VD、＊LC、＊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：等于（＝），2：不等于（<>），3：小于（<），4：大于（>）

提示：

当你用 FND 指令查找由指令 ATT、LIFO 和 FIFO 生成的表时，实际填表数（EC）和输入数据相符，直接对应。最大填表数（TL）对 ATT、LIFO 和 FIFO 指令是必需的，但 FND 指令并不需要它。

因此，FND 指令的操作数 SRC 是一个字地址（指向 EC），比相应的 ATT、LIFO 或 FIFO 指令的操作数 TABLE 要高 2 个字节，如图 6-36 所示。

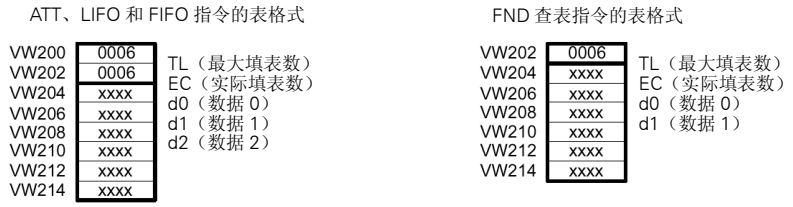
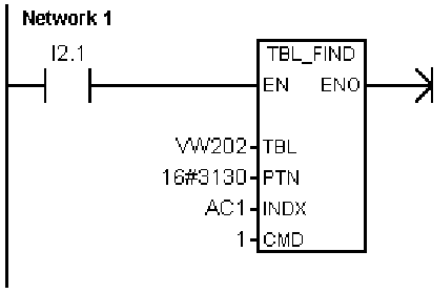
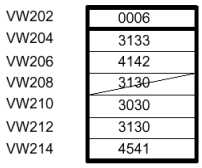


图 6-36　FND 指令与 ATT、LIFO 和 FIFO 指令所使用的表格式上的差异

查表指令程序举例：




```
Network 1
LD      I2.1
FND=    VW202,16#3130,AC1
```



这是一个你正在查找的表，如果表是用 ATT、LIFO 和 FIFO 指令创建， VW200 包含最大填表数，但是 FND 查表指令并不需要它。

AC1 | 0 | 从表头查找，AC1 必须置 0

执行查表

AC1 | 2 | AC1 中保存了第 1 个符合查表条件的数据编号（d2）

AC1 | 3 | 查表中剩余数据前，INDEX 加 1

执行查表

AC1 | 4 | AC1 中保存了第 2 个符合查表条件的数据编号（d4）

AC1 | 5 | 查表中剩余数据前，INDEX 加 1

执行查表

AC1 | 6 | AC1 中保存了已填表数（EC），这表明整个表已经查完，没有找到符合条件的数据

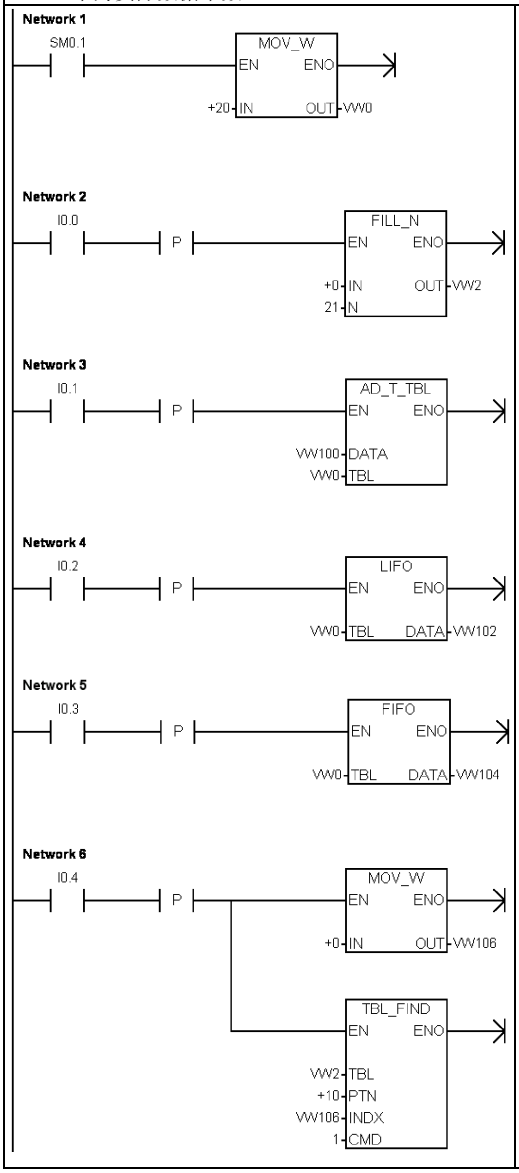
AC1 | 0 | 再次查表前，INDEX 值必须置为 0

创建一个表的程序举例

下列程序创建一个包含 20 条数据的表。存储区中的第一个数据为表的长度（在本例中为 20）。存储区中的第二个数据为表中数据的实际个数。其它存储区单元为数据。一个表可以有最多 100 条数据。其中不包括定义表长度和实际数据个数的两个单元（在本例中为 VW0 和 VW2）。当 CPU 执行第一条指令时，表中的实际数据个数（VW2）会自动增或者减。

在使用表之前，必须为表指定数据的最多个数。否则你将无法在表中插入数据。同时，要确保使用边沿触发来激活读写指令。

在查表之前，INDX（VW106）必须清 0。如果找到匹配的数据，INDX 中会存入表中的条目号；如果没有找到，INDX 中为实际数据个数（VW2）。




```
Network 1     //从 VW4 开始创建一个 20 条数据的表。
              //1. 在首次扫描定义表的最大长度。
LD       SM0.1
MOVW     +20,VW0

Network 2     //用 I0.0 将表清 0。
              //当 I0.0 产生上升沿时，从 VW2 开始清 0。
LD       I0.0
EU
FILL     +0,VW2,21
Network 3     //用 I0.1 向表中写数据。
              //当 I0.1 产生上升沿时，将 VW100 的数
              //据写入表中。
LD       I0.1
EU
ATT      VW100,VW0
Network 4     //用 I0.2 读表中的最后一个数据。
              //将表中最后一个数据值移入 VW102 中。
              //这会减少条目的数量。
              //当 I0.2 产生上升沿时，将表中最后一个数
              //据移入 VW102 中。
LD       I0.2
EU
LIFO     VW0,VW102
Network 5     //用 I0.3 读表中的第一个数据。
              //将表中的第一个数据值移入 VW104 中。
              //这会减少条目的数量。
              //当 I0.3 产生上升沿时，将表中第一个数
              //据移入 VW104 中。
LD       I0.3
EU
FIFO     VW0,VW104
Network 6     //在表中搜索是否有数据为 10。
              //1．当 I0.4 产生上升沿时，将 INDX 指针
              // 清 0。
              //2．在表中搜索等于 10 的数据。
LD       I0.4
EU
MOVW     +0,VW106
FND=     VW2,+10,VW106
```

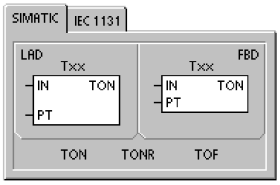
定时器指令

SIMATIC 定时器指令

接通延时定时器

有记忆的接通延时定时器

接通延时定时器（TON）和有记忆的接通延时定时器在使能输入接通时记时。定时器号决定了它的分辨率。



断开延时定时器

断开延时定时器用于在输入断开后延时一段时间断开输出。定时器号决定了它的分辨率。

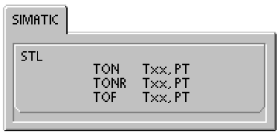


表 6-69　SIMATIC 定时器指令的有效操作数

输入/输出	数据类型	操作数
TXX	WORD	常数（T0 到 T255）
IN	BOOL	I、Q、V、M、SM、S、T、C、L、能流
PT	INT	IW、QW、VW、MW、SMW、T、C、LW、AC、AIW、*VD、*LD、*AC、常数

提示：

不能将同一个定时器号同时用作 TOF 和 TON。例如，不能够既有 TON T32 又有 TOF T32。

使用定时器可以完成基于时间的计数功能，S7-200 提供了下述 3 种定时器指令。

- 接通延时定时器（TON）用于单一间隔的定时
- 有记忆接通延时定时器（TONR）用于累计许多时间间隔
- 断开延时定时器（TOF）用于关断或者故障事件后的延时（例如：在电机停后，需要冷却电机）

表 6-70　定时器指令的操作数

定时器类型	当前值>=预设值	使能输入接通	使能输入断开	上电周期/首次扫描
TON	定时器位 ON， 当前连续计数到 32767	当前值计数时间	定时器位 OFF， 当前值＝0	定时器位 OFF， 当前值＝0
TONR	定时器位 ON， 当前连续计数到 32767	当前值计数时间	定时器位和当前值保持最后状态	定时器位 OFF， 当前值保持 [1]
TOF	定时器位 OFF， 当前值＝预设值，停止计数	定时器位 ON， 当前值＝0	发生 ON 到 OFF 的跳变之后，定时器计数	定时器位 OFF， 当前值＝0

[1] 有记忆定时器的当前值通过电源扫描选择有记忆，有关 S7-200 CPU 有记忆存储器的详细内容参阅第 4 章。


应用示例

使用接通延时定时器时，请参阅资料光盘上应用示例中的例子程序 31。

当使能输入接通时，接通延时定时器和有记忆接通延时定时器开始计时，当定时器的当前值（Txxx）大于等于预设值时，该定时器位被置位。

- 当使能输入断开时，清除接通延时定时器的当前值，而对于有记忆接通延时定时器，其当前值保持不变。
- 可以用有记忆接通延时定时器累计输入信号的接通时间，利用复位指令（R）清除其当前值。
- 当达到预设时间后，接通延时定时器和有记忆接通延时定时器继续计时，一直计到最大值 32767。

断开延时定时器（TOF）用来在输入断开后延时一段时间断开输出。当使能输入接通时，定时器位立即接通，并把当前值设为 0。当输入断开时，定时器开始定时，直到达到预设的时间。

- 当达到预设时间时，定时器位断开，并且停止计时当前值。当输入断开的时间短于预设时间时，定时器位保持接通。
- TOF 指令必须用输入信号的接通到断开的跳变启动计时。
- 如果 TOF 定时器在顺控（SCR）区，而且顺控区没有启动，TOF 定时器的当前值设置为 0，定时器位设置为断开，当前值不计时。

提示：

复位（R）指令用来对定时器复位。复位指令执行如下的操作：

- 定时器位＝OFF
- 定时器当前位置＝0

TONR 定时器只能通过复位指令进行复位操作。复位后，为了再启动，TOF 定时器需要使能输入有一个从 ON 到 OFF 的跳变。

为定时器选择分辨率

定时器对时间间隔记数。定时器的分辨率（时基）决定了每个时间间隔的时间长短。例如：一个以 10ms 为时基的延时接通定时器，在使能位接通后，以 10ms 的时间间隔计数，10ms 的定时器计数值为 50 代表 500ms。SIMATIC 定时器有三种分辨率：1ms、10ms 和 100ms。如表 6-71 中所示，定时器号决定了定时器的分辨率。

提示：

为确保时间间隔的最小值，预置值必须比它大 1。例如：为确保最小时间间隔 2100ms，要将 100ms 定时器的预置值 PV 设为 22。

表 6-71　定时器号和分辨率

定时器类型	用毫秒（ms）表示的分辨率	用秒（s）表示的最大当前值	定时器号
TONR	1ms	32.767s	T0,T64
	10ms	327.67s	T1-T4, T65-T68
	100ms	3276.7s	T5-T31, T69-T95
TON、TOF	1ms	32.767s	T32,T96
	10ms	327.67s	T33-T36, T97-T100
	100ms	3276.7s	T37-T63, T101-T255

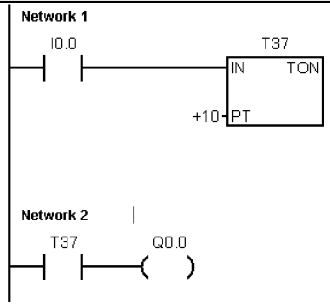
分辨率对定时器的影响

对于 1ms 分辨率的定时器来说，定时器位和当前值的更新不与扫描周期同步。对于大于 1ms 的程序扫描周期，在一个扫描周期内，定时器位和当前值刷新多次。

对于 10ms 分辨率的定时器来说，定时器位和当前值在每个程序扫描周期的开始刷新。定时器位和当前值在整个扫描周期过程中为常数。在每个扫描周期的开始会将一个扫描累计的时间间隔加到定时器当前值上。

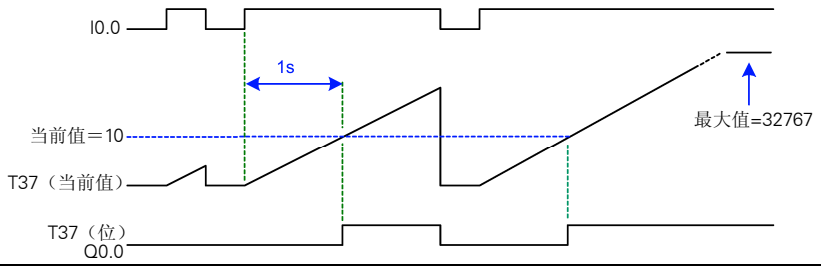
对于 100ms 分辨率的定时器来说，定时器位和当前值在指令执行时刷新。因此，为了使定时器保持正确的定时值，要确保在一个程序扫描周期中，只执行一次 100ms 定时器指令。

SIMATIC 接通延时定时器指令程序举例




```
Network 1     //100ms 定时器 T37 在 1s 后到时。I0.1 接
              //通，T37 使能，I0.1 断开，复位 T37。
LD      I0.0
TON     T37,+10

Network 2     //定时器 T37 控制 Q0.0。
LD      T37
=       Q0.0
```

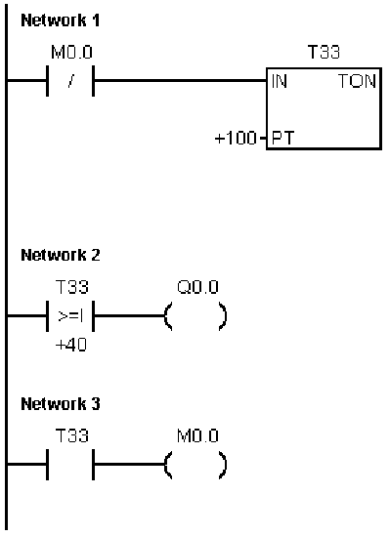
时序图



提示：

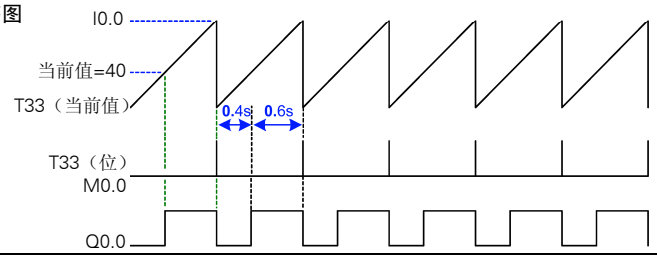
为了确保在每一次定时器达到预置值时，自复位定时器的输出都能够接通一个程序扫描周期，用一个常闭触点来代替定时器位作为定时器的使能输入。

SIMATIC 自复位接通延时定时器程序举例

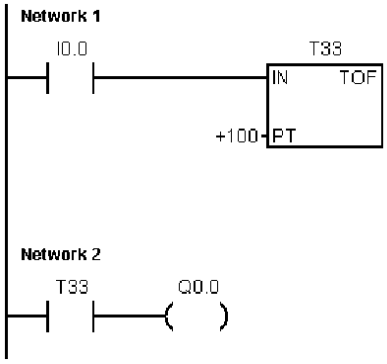



```
Network 1    //10ms 定时器 T33 在 1 秒后到时。
             // M0.0 的脉冲过窄，在状态表中无法监视。
LDN          M0.0
TON          T33,+100

Network 2    //比较指令为真的时间较长，可以在状态表中
             //监视，Q0.0 的占空比为 40%。
LDW>=        T33,+40
=            Q0.0

Network 3    // T33（位）的脉冲过窄，在状态表中无法监视。
             //在 1 秒后复位 M0.0。
LD           T33
=            M0.0
```

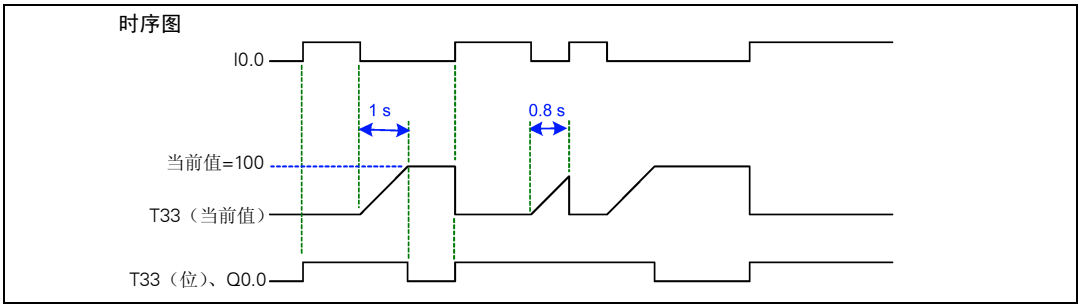
时序图



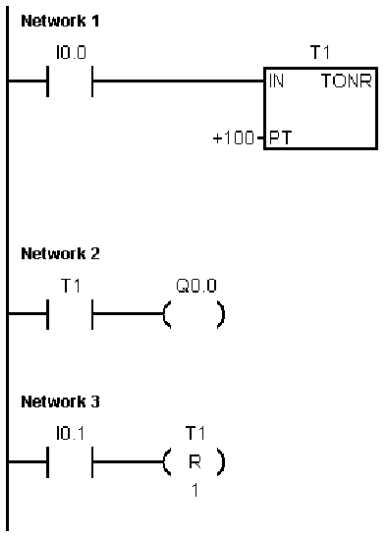
SIMATIC 断开延时定时器程序举例



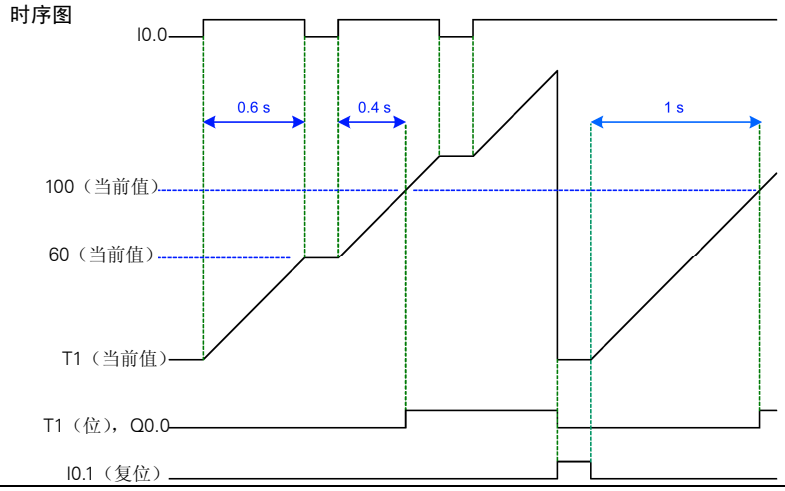

```
Network 1    //10ms 定时器 T33 在 1 秒后到时。
             //I0.0 关断使能 T33。
             //I0.0 接通 T33 复位。
LD           I0.0
TOF          T33,+100

Network 2    //定时器 T33 用其输出位控制 Q0.0。
LD           T33
=            Q0.0
```



SIMATIC 有记忆的接通延时定时器程序举例



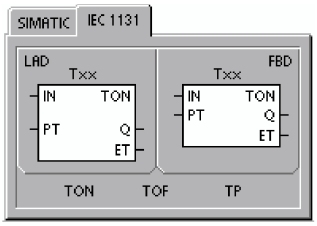

```
Network 1     //10ms TONR 定时器 T1 在 1 秒后到时。
LD            I0.0
TONR          T1,+100

Network 2     //T1 位控制 Q0.0。
              //1 秒后 T1 使 Q0.0 接通。
LD            T1
=             Q0.0

Network 3     //TONR 定时器必须用复位
              //指令才能复位。
              //当 I0.1 接通时，复位 T1。
LD            I0.1
R             T1,1
```



IEC 定时器指令

接通延时定时器

当使能输入接通时，接通延时定时器指令（TON）对时间间隔计数。



断开延时定时器

断开延时定时器（TOF）用于在输入断开后，延时一段时间后断开输出。

脉冲定时器

脉冲定时器（TP）以指定的周期产生脉冲。

表 6-72　IEC 定时器指令的有效操作数

输入/输出	数据类型	操作数
Txx	TON、TOF、TP	常数（T32 到 T63，T96 到 T255）
IN	BOOL	I、Q、V、M、SM、S、T、C、L、能流
PT	INT	IW、QW、VW、MW、SMW、SM、LW、AC、AIW、*VD、*LD、*AC、常数
Q	BOOL	I、Q、V、M、SM、S、L
ET	INT	IW、QW、VW、MW、SMW、SW、LW、AC、AQW、*VD、*LD、*AC

提示：

一个定时器号不能同时用于 TOF、TP 和 TON，例如：不能有两个定时器 TON T32 和 TOF T32。

- 当使能输入端（IN）为 1 时，接通延时定时器功能块开始启动定时，一直到预置值。当经过时间（ET）大于等于预置值（PT）时，定时器输出位（Q）变为 1。当使能输入端（IN）为 0 时，定时器输出复位。当预置时间（PT）到达时，定时停止并且定时器不工作。
- 当输入断开时，断开延时定时器功能块把输出断开的时间延迟一个固定的时间。当使能输入（IN）变为 0 时，定时器的值又变为预置值。当经过时间（ET）大于等于预置时间（PT）时，定时器输出位（Q）接通。一旦达到预置值，定时器的输出位变为 0，一直保持到使能输入（IN）再变为 1。如果使能输入（IN）变为 0 的持续时间小于预置值（PT），定时器的输出位一直保持接通。
- 脉冲定时器用于产生一个指定宽度的脉冲。当使能输入（IN）变为 1 时，输出位（Q）接通，在预置时间内输出位保持接通。一旦经过时间（ET）达到预置时间（PT），输出位变为 0。经过的时间会被保存到使能输入断开。当输出接通时，它会一直保持到脉冲周期结束。

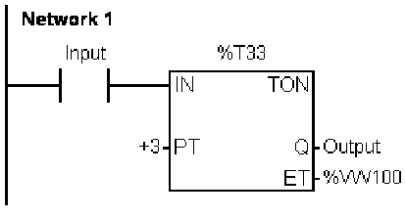
当前值的每次计数为时基的倍数。例如：以 10ms 为时基的定时器，计数值为 50，代表

500ms。IEC 定时器（TON、TOF 和 TP）有三种分辨率。分辨率由定时器号决定，如表 6-73 所示。

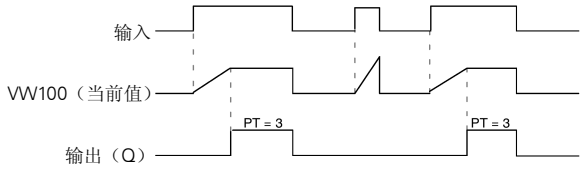
表 6-73　IEC 定时器的分辨率

定时器类型	用毫秒（ms）表示的分辨率	用秒（s）表示的最大值	定时器号
TON, TOF TP	1ms	32.767s	T32,T96
	10ms	327.67s	T33-T36, T97-T100
	100ms	3276.7s	T37-T63, T101-T255

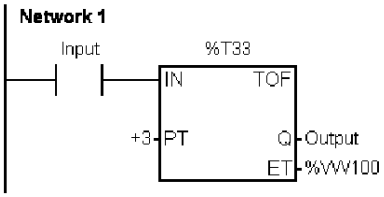
IEC 接通延时定时器指令程序举例



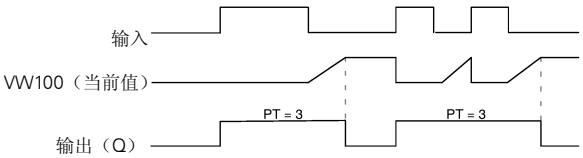
时序图



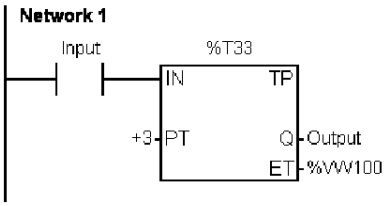
IEC 断开延时定时器指令程序举例



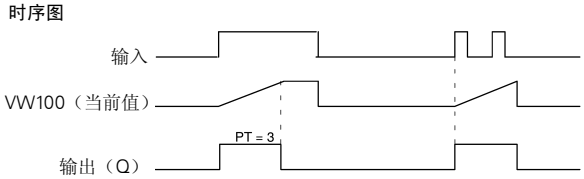
时序图



IEC 脉冲定时器指令程序举例



时序图

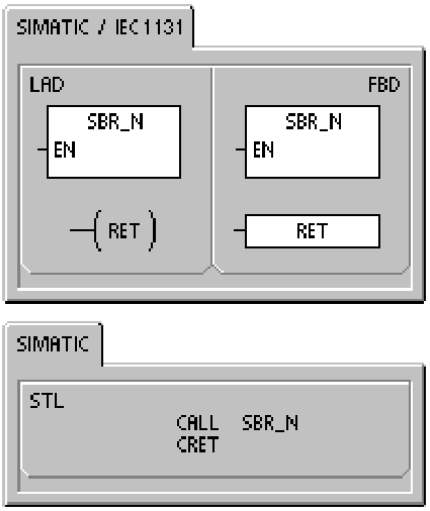


子程序指令

子程序调用指令（CALL）将程序控制权交给子程序 SBR_N。调用子程序时可以带参数也可以不带参数。子程序执行完成后，控制权回到子程序调用指令的下一条指令。

子程序条件返回指令（CRET）根据它前面的逻辑关系决定是否终止子程序。

要添加一个子程序可以在命令菜单中选择：Edit > Insert > Subroutine。



使 ENO＝0 的错误条件：

- 0008（超过子程序嵌套最大限制）
- 0006（间接寻址）

在主程序中，可以嵌套调用子程序（在子程序中调用子程序），最多嵌套 8 层。在中断服务程序中，不能嵌套调用子程序。

在被中断服务程序调用的子程序中不能再出现子程序调用。递归调用（子程序自己调用自己）不被禁止使用，但使用时应慎重。

表 6-74　子程序指令的有效操作数

输入/输出	数据类型	操作数	
SBR_N	WORD	常数　*对于 CPU221、CPU222、CPU224 和 CPU226*	0 到 63
		对于 CPU226XM	0 到 127

提示：

STEP 7-Micro/WIN 为每个子程序自动加入返回指令。

当有一个子程序被调用时，系统会保存当前的逻辑堆栈，置栈顶值为 1，堆栈的其他值为零，把控制交给被调用的子程序。当子程序完成之后，恢复逻辑堆栈，把控制权交还给调用程序。

因为累加器可在主程序和子程序之间自由传递，所以在子程序调用时，累加器的值既不保存也不恢复。

带参数调用子程序

子程序可以包含要传递的参数。参数在子程序的局部变量表中定义。参数必须有变量名（最多 23 个字符）、变量类型和数据类型。一个子程序最多可以传递 16 个参数。局部变量表中的变量类型区定义变量是传入子程序（IN）、传入和传出子程序（IN_OUT）或者传出子程序（OUT）。表 6-75 中描述了一个子程序中的参数类型。

要加入一个参数，把光标放到要加入的变量类型区（IN、IN_OUT、OUT）。点击鼠标右键可以得到一个菜单选择。选择插入选项，然后选择下一行选项。这样就出现了另一个所选类型的参数项。

表 6-75　子程序的参数类型

参　数	描　　述
IN	参数传入子程序。如果参数是直接寻址（如：VB10），指定位置的值被传递到子程序。如果参数是间接寻址（如：＊AC1），指针指定位置的值被传入子程序；如果参数是常数（如：16＃1234），或者一个地址（如：&VB100），常数或地址的值被传入子程序。
IN_OUT	指定参数位置的值被传到子程序，从子程序的结果值被返回到同样地址。常数（如：16＃1234）和地址（如：&VB100）不允许作为输入/输出参数。
OUT	从子程序来的结果值被返回到指定参数位置。常数（如：16＃1234）和地址（如：&VB100）不允许作为输出参数。
TEMP	任何局部存储器都不能用来传递参数，只能在子程序内部暂时存贮数据。

如图 6-37 中所示，局部变量表中的数据类型区定义了参数的大小和格式。参数的类型列举如下：

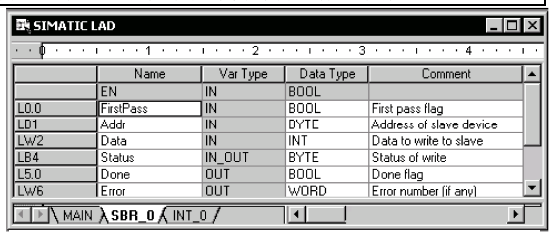
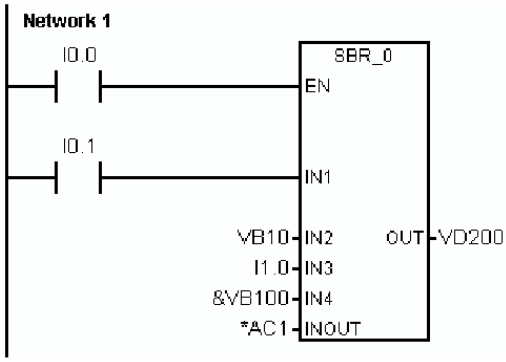


图 6-37　局部变量表

- 布尔：该数据类型用于单独位的输入和输出。下例中的 IN3 是布尔输入。
- 字节、字和双字：这些数据类型分别指明一个 1、2 或者 4 个字节的无符号输入或输出参数。
- 整数、双整数：这些数据类型分别指明 1、2 或者 4 个字节的有符号输入或输出参数。
- 实数：该数据类型指明一个 4 字节的单精度 IEEE 浮点参数。
- 能流：布尔能流仅允许对位输入操作。该变量声明告诉 STEP 7-Micro/WIN 32 此输入参数是位逻辑指令组合的能流结果。在局部变量表中布尔能流输入必须出现在其它类型的前面。只有输入参数可以这样使用。下列例子程序中的允许输入（EN）和 IN1 输入都使用布尔逻辑。

调用子程序的程序举例：

以下有两个 STL 程序。第一个程序只能在 STL 编辑器中以 STL 的形式显示，因为用作能流输入的 BOOL 参数没有存储在 L 存储区中。
第二个程序能够在 LAD 和 FBD 编辑器中显示，因为使用了 L 存储器来存储用作能流输入的 BOOL 输入参数。



只能显示 STL：

```
Network 1
LD      I0.0
CALL    SBR_0,I0.1,VB10,I1.0,&VB100,*AC1,VD200
```

可以在 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
```

地址参数（如 IN4 处的&VB100）以一个双字（无符号）的值传送到子程序。在带常数调用程序时必须指明常数类型。例如，把值为 12345 的无符号双字作为参数进行传递，常数参数必须用 DW＃12345 指明。如果参数中缺少了常数描述符，常数可能被当作不同的类型。

输入或输出参数上没有自动数据类型转换功能。例如，如果局部变量表明一个参数具有实型，而在调用时使用一个双字，子程序中的值就是双字。

当给子程序传递值时，它们放在子程序的局部存储器中。局部变量表的最左列是每个被传递参数的局部存储器地址。当子程序调用时，输入参数值被拷贝到子程序的局部存储器。当子程序完成时，从局部存储器区拷贝输出参数值到指定的输出参数地址。

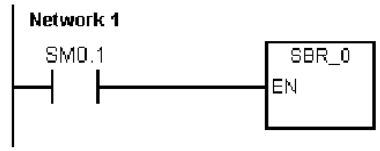
数据单元的大小和类型用参数的代码表示。在子程序中局部存储器的参数值的分配如下所示：

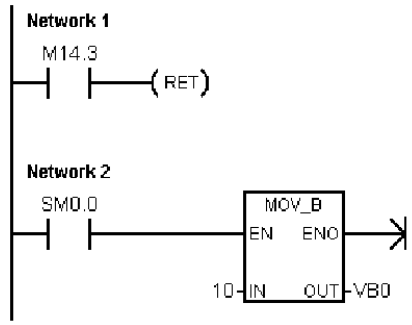
- 按照子程序指令的调用顺序，参数值分别给局部存储器，起始地址是 L0。
- 1 到 8 连续位参数值分配一个字节，从 Lx.0 到 Lx.7。
- 字节、字和双字值按照字节顺序分配在局部存储器中（LBx、LWx 或 LDx）。

在带参数调用子程序指令中，参数必须按照一定顺序排列，输入参数在最前面，其次是输入/输出参数，然后是输出参数。

如果用语句表编程，CALL 指令的格式是：

CALL 子程序号，参数 1，参数 2，…，参数

子程序和子程序返回指令程序举例		
M A I N		Network 1 //在首次扫描，调用初始化子程序 0。 LD SM0.1 CALL SBR_0

S
B
R
0
		Network 1 //你可以使用条件返回指令在子程序
		//结束之前返回。
		LD M14.3
		CRET
		
		Network 2 //如果 M14.3 接通，本段程序会被跳过。
		LD SM0.0
		MOVB 10,VB0

网络通讯

7

S7-200 可以满足你的通讯和网络需求，它不仅支持简单的网络，而且支持比较复杂的网络。S7-200 提供了通讯手段，使你可以用它与那些使用自己的通讯协议的设备，例如：打印机和称重天平等进行通讯。

STEP 7-Micro/WIN 使得你建立和配置网络简便快捷。

本章内容：

理解 S7-200 网络通讯的基本概念

在网络中使用主站和从站

S7-200 支持主-从通讯方式并且可以被配置为主站或者从站。STEP 7-Micro/WIN 只能是主站。

提示：

当使用 Windows NT 和 PC/PPI 电缆时，在网络上不能有其它主站。

主站

网络上的主站器件可以向网络上的其它器件发出要求。主站也可以对网络上其它主站的要求作出响应。典型的主站器件包括：STEP 7-Micro/WIN、TD200 等 HMI 产品和 S7-300 或 S7-400 PLC。当 S7-200 需要从另外一个 S7-200 读取信息时被定义为主站（点对点通讯）。

提示：

如果网络上有其它主站，TP070 将无法工作。

从站

配置为从站的器件只能对其它主站的要求作出响应，自己不能发出要求。对于多数情况，S7-200 被配置为从站。作为从站，S7-200 响应主站的要求。主站可以是操作面板或者 STEP7-Micro/WIN 等。

设置波特率和站地址

数据通过网络传输的速度是波特率。其单位通常为 Kbaud 或者 Mbaud。波特率用于量度在给定时间内传输数据的多少。例如，19.2Kbaud 表示在 1 秒内传输 19,200 位数据。在同一个网络中通讯的器件必须被配置成相同的波特率。因此，网络的最高波特率取决于连接在该网络上的波特率最低的设备。

表 7-1 中列出了 S7-200 支持的波特率。

表 7-1　S7-200 支持的波特率

网络	波特率
标准网络	9.6k 到 187.5k
使用 EM 277	9.6k 到 12M
自由口模式	1200 到 115.2K

在网络中要为每个设备指定唯一的站地址。唯一的站地址可以确保数据发送到正确的设备或者来自正确的设备。S7-200 支持的网络地址为从 0 到 126。对于有两个通讯口的 S7-200，每一个通讯口可以有自己的站地址。表 7-2 中列出了 S7-200 设备的缺省设置。

表 7-2　S7-200 设备的缺省站地址

S7-200 设备	缺省地址
STEP 7-Micro/WIN	0
HMI(TD200、TP 或 OP)	1
S7-200 CPU	2

为 STEP 7-Micro/WIN 设置波特率和站地址

你必须为 STEP 7-Micro/WIN 配置波特率和站地址。波特率的设置必须与网络上的其它设备相同，而站地址则必须是唯一的。

通常，你不需要改变 STEP 7-Micro/WIN 的缺省站地址 0。如果你的网络中包含其它编程设备使用 STEP 7 之类的编程软件，你需要改变 STEP 7-Micro/WIN 的站地址。

如图 7-1 所示，为 STEP 7-Micro/WIN 配置波特率和站地址非常简单。在操作栏中点击通讯图标，然后执行以下步骤：

1．在通讯设置窗口中双击图标。

2．在 Set PG/PC Interface 对话框中点击属性按钮。

3．为 STEP 7-Micro/WIN 选择站地址。

4．为 STEP 7-Micro/WIN 选择波特率。

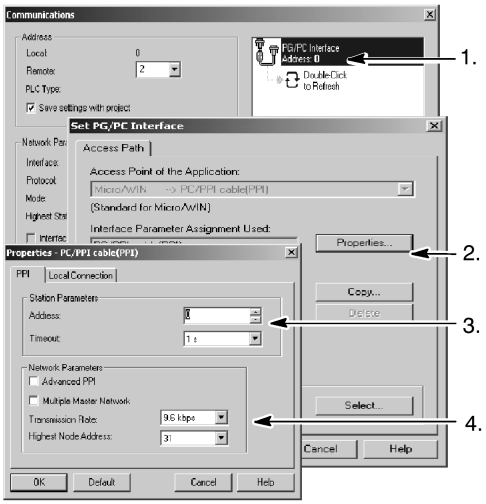


图 7-1　配置 STEP 7-Micro/WIN

为 S7-200 设置波特率和站地址

你也必须为 S7-200 配置波特率和站地址。S7-200 的波特率和站地址存储在系统块中。在为 S7-200 选择了参数之后，你必须将系统块下载到 S7-200 中。

每一个 S7-200 通讯口的波特率缺省设置为 9.6k，站地址的缺省设置为 2。如图 7-2 所示，使用 STEP 7-Micro/WIN 为 S7-200 设置波特率和站地址。你可以在操作栏中点击系统块图标或者在命令菜单中选择 View>Component>System Block，然后执行以下步骤：

1．为 S7-200 选择站地址。

2．为 S7-200 选择波特率。

3．下载系统块到 S7-200。

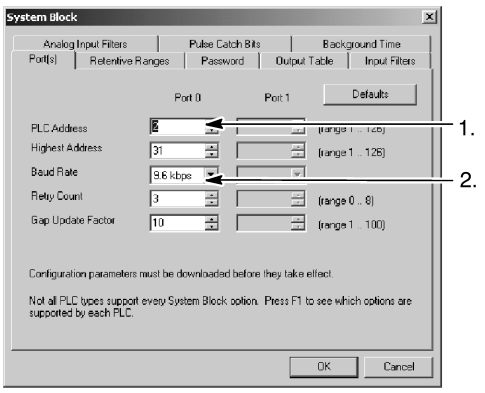


图 7-2　配置 S7-200 CPU

设置远端地址

在将最新设置下载到 S7-200 之前，你必须为 STEP 7-Micro/Win 的通讯口和 S7-200 的远端站地址作配置，使它与远端的 S7-200 的当前设置相匹配，如图 7-3 所示。

在下载最新设置之后，你必须重新配置 COM 口（如果对远端 S7-200 的设置与当前设置不同）。要显示通讯对话框，可以在操作栏中选择通讯图标或者在命令菜单中选择 View>Component>Communications。

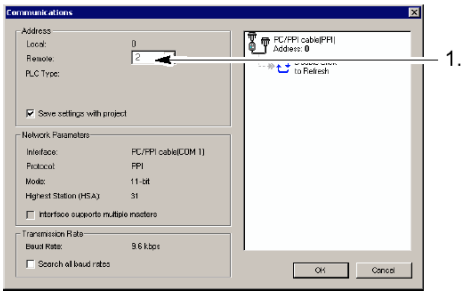


图 7-3　配置 S7-200 CPU

1．选择远端地址。

2．确认 COM 口、远端 S7-200 通讯口和 PC/PPI 电缆的波特率参数匹配，否则，会通讯失败。

在网络上寻找 S7-200 CPU

你可以寻找并且识别连接在网络上的 S7-200。你也可以选择是使用特定的波特率还是用所有波特率来寻找网络上的 S7-200。

如果使用 PC/PPI 电缆，STEP 7-Micro/WIN 只能在 9.6k 和 19.2k 两个波特率寻找。对于 CP 卡，STEP 7-Micro/WIN 可以在 9.6k、19.2k 和 187.5k 三个波特率寻找。搜寻从当前选择的波特率开始。

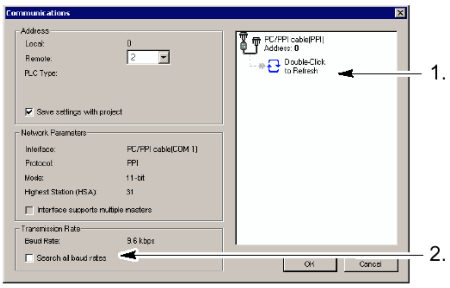


图 7-4　在网络上寻找 S7-200

1．打开通讯对话框并双击刷新图标开始搜寻。

2．要使用所有波特率搜寻，选中在所有波特率下搜寻复选框。

为网络选择通讯协议

S7-200 CPU 支持以下通讯协议中的一种或多种，它允许你配置网络实现你的应用要求：

- 点对点接口（PPI）
- 多点接口（MPI）
- PROFIBUS

在开放系统互联（OSI）七层模式通讯结构的基础上，这些通讯协议在一个令牌环网络上实现。令牌环网络符合欧洲标准 EN 50170 中定义的 PROFIBUS 标准。这些协议是非同步的字符协议，有 1 位起始位、8 位数据位、偶校验位和 1 位停止位。通讯结构依赖于特定的起始字符和停止字符、源和目地站地址，报文长度和数据校验和。如果使用相同的波特率，这些协议可以在同一个网络中同时运行而互不干扰。

PPI 协议

PPI 是一种主-从协议：主站设备发送要求到从站设备，从站设备响应，如图 7-5 所示。从站不发信息，只是等待主站的要求和对要求作出响应。

主站靠一个 PPI 协议管理的共享连接来与从站通讯。PPI 并不限制与任意一个从站通讯的主站数量，但是在一个网络中，主站的个数不能超过 32。

选择 PPI 高级允许网络设备建立一个设备与设备之间的逻辑连接。对于 PPI 高级，每个设备的连接个数是有限制的。S7-200 支持的连接个数如表 7-3 所示。

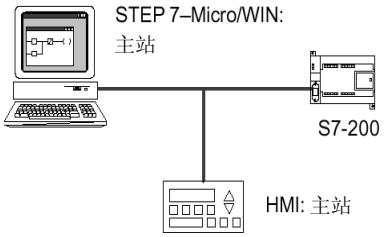


图 7-5　PPI 网络

如果在用户程序中使能 PPI 主站模式，S7-200 CPU 在运行模式下可以作主站。（参见附录 D 中 SMB 30 的描述）在使能 PPI 主站模式之后，可以使用网络读写指令来读写另外一个 S7-200。当 S7-200 作 PPI 主站时，它仍然可以作为从站响应其它主站的请求。

你可以使用 PPI 协议与所有 S7-200 CPU 通讯。当与 EM 277 通讯时，你必须使能 PPI 高级。

表 7-3　S7-200 CPU 和 EM 277 模块的连接个数

模块	波特率	连接数
S7-200 CPU 通讯口 0	9.6k、19.2k、187.5k	4
S7-200 CPU 通讯口 1	9.6k、19.2k、187.5k	4
EM 277	9.6K 到 12M	6（每个模块）

MPI 协议

MPI 允许主-主通讯和主-从通讯，如图 7-6 所示。与一个 S7-200 CPU 通讯，STEP 7-Micro/WIN 建立主-从连接。MPI 协议不能与一个作为主站的 S7-200 CPU 通讯。

网络设备通过任意两个设备之间的连接通讯（由 MPI 协议管理）。设备之间通讯连接的个数受 S7-200 CPU 或者 EM 277 模块所支持的连接个数的限制。表 7-3 中给出了 S7-200 支持的连接个数。

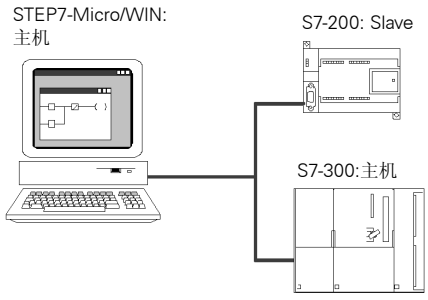


图 7-6　MPI 网络

对于 MPI 协议，S7-300 和 S7-400 PLC 可以用 XGET 和 XPUT 指令来读写 S7-200 的数据。要得到更多关于这些指令的信息，参见 S7-300 或者 S7-400 的编程手册。

PROFIBUS 协议

PROFIBUS 协议通常用于实现与分布式 I/O（远程 I/O）的高速通讯。可以使用不同厂家的 PROFIBUS 设备。这些设备包括普通的输入/输出模块、电机控制器和 PLC。

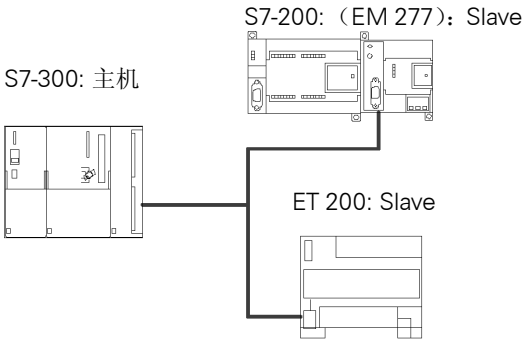


图 7-7　PROFIBUS 网络

PROFIBUS 网络通常有一个主站和若干个 I/O 从站，如图 7-7 所示。主站设备通过配置可以知道 I/O 从站的类型和站号。主站初始化网络使网络上的从站设备与配置相匹配。主站不断地读写从站的数据。当一个 DP 主站成功配置了一个 DP 从站之后，它就拥有了这个从站设备。如果在网上有第二个主站设备，它对第一个主站的从站的访问将受到限制。

仅仅使用 S7-200 设备的网络配置实例

单主站 PPI 网络

对于一个简单的单主站网络来说，编程设备可以通过 PC/PPI 电缆或者通讯卡（CP）与 S7-200 CPU 通讯。

在图 7-8 上面的网络实例中，编程站（STEP 7-Micro/WIN）是网络的主站。在图 7-8 下面的网络实例中，人机界面（HMI）设备（例如：TD200、TP 或者 OP）是网络的主站。

在两个网络中，S7-200 CPU 都是从站响应来自主站的要求。

对于单主站 PPI 配置，配置 STEP 7-Micro/WIN 使用 PPI 协议。可以选择单主站、多主站或者 PPI 高级。

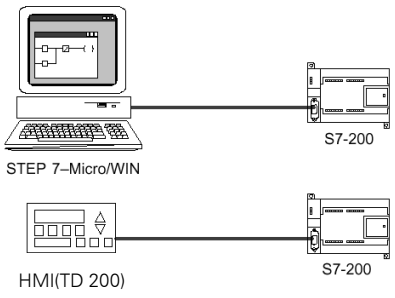


图 7-8　单主站 PPI 网络

多主站 PPI 网络

图 7-9 中给出了有一个从站的多主站网络实例。编程站（STEP 7-Micro/WIN）使用 CP 卡或者 PC/PPI 电缆。STEP 7-Micro/WIN 和 HMI 设备共享网络。

STEP 7-Micro/WIN 和 HMI 设备都是网络的主站，它们必须有不同的网络地址。S7-200 CPU 是从站。对于单从站的多主网络，配置 STEP 7-Micro/WIN 使用 PPI 协议时，应使能多主站，最好使用 PPI 高级。

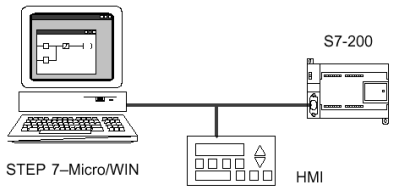


图 7-9　单个从站的多主站网络

图 7-10 中给出了有多个从站的多主站网络实例。在例子中，STEP 7-Micro/WIN 和 HMI 可以对任意 S7-200 CPU 从站读写数据。STEP 7-Micro/WIN 和 HMI 共享网络。

所有设备（主站和从站）有不同的网络地址。对于多个从站的多主网络，配置 STEP 7-Micro/WIN 使用 PPI 协议时，应使能多主站，最好使用 PPI 高级。

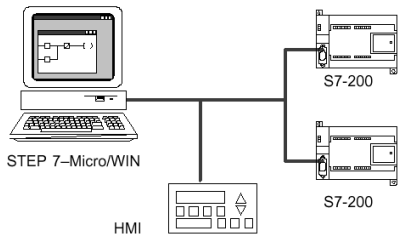


图 7-10　多个从站的多主站网络

复杂的 PPI 网络

图 7-11 中给出了一个带点对点通讯的多主网络实例。

STEP 7-Micro/WIN 和 HMI 通过网络读写 S7-200 CPU，同时 S7-200 CPU 之间使用网络读写指令相互读写数据（点对点通讯）。

对于这种复杂类型的 PPI 网络，配置 STEP 7-Micro/WIN 使用 PPI 协议时，应使能多主站，最好使用 PPI 高级。

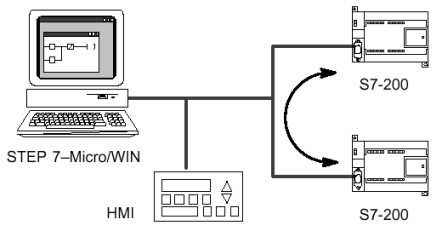


图 7-11　点对点通讯

图 7-12 中给出了另外一个带点对点通讯的多主网络的复杂 PPI 网络实例。在本例中，每个 HMI 监控一个 S7-200 CPU。

S7-200 CPU 使用 NETR 和 NETW 指令相互读写数据（点对点通讯）。

对于这个网络，配置 STEP 7-Micro/WIN 使用 PPI 协议时，应使能多主站，最好使用 PPI 高级。

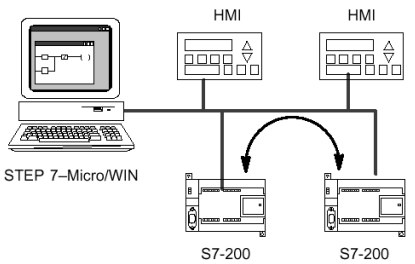


图 7-12　HMI 设备和点对点通讯

使用 S7-200、S7-300 和 S7-400 设备的网络配置实例

网络波特率可以达到 187.5k

在图 7-13 所示的网络实例中，S7-300 用 XGET 和 XPUT 指令与 S7-200 CPU 通讯。S7-300 不能与作为主站的 S7-200 通讯。

如果波特率超过 19.2k，STEP 7-Micro/WIN 必须使用通讯网卡（CP）来连接。

为了能与 S7-200 CPU 通讯，配置 STEP 7-Micro/WIN 使用 PPI 协议时，应使能多主站，最好使用 PPI 高级。

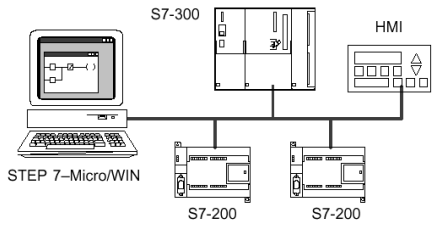


图 7-13　波特率可以达到 187.5k

波特率超过 187.5k（直至 12M）的网络

对于波特率高于 187.5k 的情况，S7-200 CPU 必须使用 EM 277 模块连接网络，如图 7-14 所示。STEP 7-Micro/WIN 必须使用通讯卡（CP）连接。

在这个配置中，S7-300 可以用 XGET 和 XPUT 指令与 S7-200 通讯，并且 HMI 可以监控 S7-200 或者 S7-300。

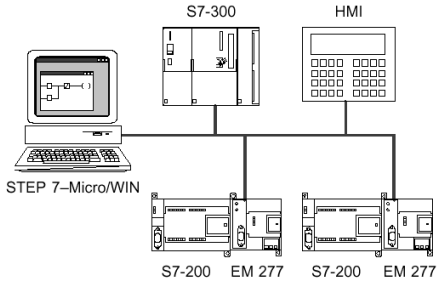


图 7-14　波特率高于 187.5k

EM 277 只能作从站。

STEP 7-Micro/WIN 可以通过 EM 277 对任意 S7-200 CPU 编程或者监控。为了与 EM 277 通讯，在配置 STEP 7-Micro/WIN 使用 PPI 协议时，使能 PPI 高级。

PROFIBUS 网络配置实例

S7-315-2DP 作 PROFIBUS 主站，EM 277 作 PROFIBUS 从站的网络

图 7-15 中给出了用 S7-315-2DP 作 PROFIBUS 主站的 PROFIBUS 网络实例。EM 277 作网络从站。

S7-315-2DP 可以发送数据到 EM 277，也可以从 EM 277 读取数据。通讯的数据量为 1 到 128 个字节。S7-315-2DP 读写 S7-200 的 V 存储器。网络支持的波特率从 9600 到 12M。

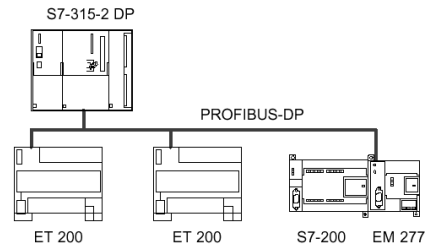


图 7-15　S7-315-2DP 网络

有 STEP 7-Micro/WIN 和 HMI 的网络

图 7-16 中给出了用 S7-315-2DP 作 PROFIBUS 主站，EM 277 作 PROFIBUS 从站的网络实例。在这个配置中，HMI 通过 EM 277 监控 S7-200。STEP 7-Micro/WIN 通过 EM 277 对 S7-200 编程。

网络支持 9.6K 到 12M 的波特率。当波特率高于 19.2K 时，STEP 7-Micro/WIN 要用 CP 卡。对于 CP 卡，配置 STEP 7-Micro/WIN 使用 PROFIBUS 协议。如果网络上只有 DP 设备，选择 DP 或者标准协议。如果网络上有非 DP 设备，为 PROFIBUS 主站设备选择通用（DP/FMS）协议。

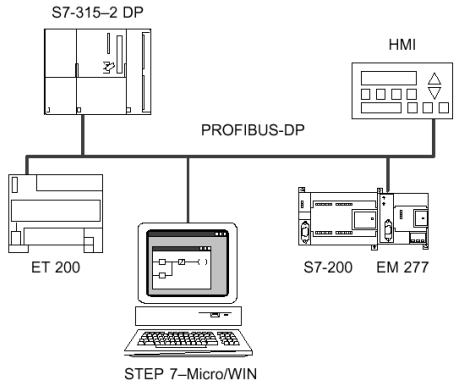


图 7-16　PROFIBUS 网络

通讯接口的安装和删除

在 Set PG/PC Interface 对话框中，你可以使用安装/删除接口对话框来安装或者删除计算机上的通讯接口。

1. 在 Set PG/PC Interface 对话框中，点击访问安装/删除接口对话框。选择框中列出了可以使用的接口，安装框中显示计算机上已经安装了的接口。
2. 要添加一个接口：选择计算机上已经安装了的通讯硬件并点击安装。当关闭安装/删除接口对话框后，Set PG/PC Interface 对话框中会在 Interface Parameter Assignment Used 框中显示接口。
3. 要删除一个接口：选择要删除的接口并点击删除。当关闭安装/删除接口对话框后，Set PG/PC Interface 对话框中会在 Interface Parameter Assignment Used 框中删除该接口。

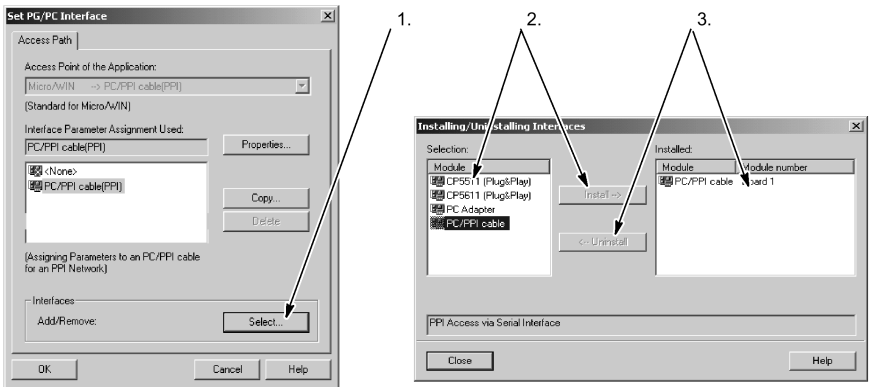


图 7-17　Set PG/PC Interface 和安装/删除接口对话框

提示：

对于 Windows NT 用户要注意以下特殊硬件安装信息

在 Windows NT 操作系统下安装硬件模块与在 Windows 95 下有轻微差异。虽然在两个系统下的硬件模块是一样的，在 Windows NT 下安装需要更多的硬件知识。Windows 95 自动的为你设置系统资源，而 Windows NT 则不能。Windows NT 只能提供缺省值，这些值与硬件配置可能匹配，也可能不匹配。但可以很容易地修改这些参数，以便于与所要求的系统设置匹配。当安装完硬件后，从安装列表中选择所安装的硬件，单击“Resources”按钮，出现资源对话框。该对话框允许为所安装的实际硬件修改系统设置。如果该按钮无效（灰色），说明你无需作任何修改。

现在你可能需要参考硬件手册，根据你的硬件设置决定对话框中所列出的每个参数设置，为了正确建立通讯，可能需要试几个不同的中断。

在 PPI 多主模式下改变计算机的端口设置

如果你在支持 PPI 多主配置的操作系统（Windows NT 不支持 PPI 多主站模式）下使用 PC/PPI 电缆，你需要调整计算机的端口设置：

1. 在桌面上用右键单击“我的电脑”图标并在命令菜单中选择属性。
2. 选择设备管理标签。对于 Windows 2000，选择第一个硬件标签然后按设备管理按钮。
3. 双击端口（COM 和 LPT）。
4. 选择当前使用的通讯端口（例如：COM1）。
5. 在端口设置页，点击高级按钮。
6. 将接收缓冲区和发送缓冲区调整到最低值（1）。
7. 点击 OK 使改变生效，关闭所有窗口并且重新启动计算机，使新的设置激活。

网络的建立

基本原则

对于任何导线安装浪涌抑制器可以避免雷击浪涌电流。应避免将低压信号线和通讯电缆与交流导线和高能量、快速开关的直流导线布置在同一线槽中。要成对使用导线，用中性线或公共线与能量线或信号线配对。

S7-200 CPU 的端口是不隔离的。如果想使网络隔离，应考虑使用 RS-485 中继器或者 EM 277。

注意：

具有不同参考电位的互联设备有可能导致不希望的电流流过连接电缆。

这种不希望的电流有可能导致通讯错误或者设备损坏。

要确保用通讯电缆连接在一起的所有设备具有相同的参考电位，或者彼此隔离，来避免产生这种不希望的电流。关于接地、选择电路参考点和使用隔离电路的相关信息请参见第 3 章。

为网络确定通讯距离、通讯速率和电缆类型

如表 7-4 所示，网段的最大长度取决于两个因素：隔离（用 RS-485 中继器）和波特率。

当你连接具有不同地电位的设备时需要隔离。当接地点之间的距离很远时，有可能具有不同的地电位。即使距离较近，大型机械的负载电流也能导致地电位不同。

表 7-4　网络电缆的最大长度

波特率	非隔离 CPU 口 [1]	有中继器的 CPU 口或者 EM 277
9.6K 到 187.5K	50m	1,000m
500k	不支持	400m
1M 到 1.5M	不支持	200m
3M 到 12M	不支持	100m

[1] 如果不使用隔离端口或者中继器，允许的最长距离为 50m。测量该距离时，从网段的第一个节点开始，到网段的最后一个节点。

在网络中使用中继器

RS-485 中继器为网段提供偏压电阻和终端电阻。你可以为以下目的使用中继器。

- 增加网络的长度：在网络中使用一个中继器可以使网络的通讯距离扩展 50 米。如果你使用两个中继器而且中间没有其它节点，网络的通讯距离按照所使用的波特率扩展一个网段的长度。在一个串联网络中，你最多可以使用 9 个中继器，但是网络的总长度不能超过 9600 米。
- 为网络增加设备：在 9600 的波特率下，50 米距离之内，一个网段最多可以连接 32 个设备。使用一个中继器允许你在网络上再增加 32 个设备。
- 使不同的网段之间电隔离：如果不同的网段具有不同的地电位，将它们隔离会提高网络的通讯质量。

一个中继器在网络中被算作网段的一个节点，尽管如此，它没有被指定站地址。

图 7-18　带中继器的网络举例

选择网络电缆

S7-200 网络使用 RS-485 标准，使用双绞线电缆。表 7-5 中列出了网络电缆的技术指标。在一个网段上最多可以连接 32 个设备。

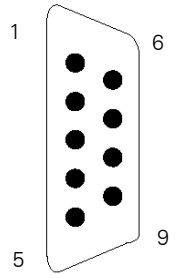
表 7-5　网络电缆的通用技术指标

技术指标	描述
电缆类型	屏蔽双绞线
回路阻抗	≤115Ω/Km
有效电容	30pF/m
标称阻抗	大约 135Ω～160Ω（频率＝3MHz～20MHz）
衰减	0.9dB/100m（频率＝200KHz）
导线截面积	0.3mm^2~0.5mm^2
电缆直径	8mm±0.5mm

引脚分配

S7-200 CPU 上的通讯端口是符合欧洲标准 EN 50170 中 PROFIBUS 标准的 RS-485 兼容 9 针 D 型连接器。表 7-6 中给出了通讯接口的物理连接和通讯接口的引脚分配。

表 7-6　S7-200 通讯端口引脚分配

连接器	针	PROFIBUS 名称	端口 0/端口 1
	1	屏蔽	机壳接地
	2	24V 返回	逻辑地
	3	RS 485 信号 B	RS-485 信号 B
	4	发送申请	RTS（TTL）
	5	5V 返回	逻辑地
	6	＋5V	＋5V，100Ω串联电阻
	7	＋24V	＋24V
	8	RS 485 信号 A	RS-485 信号 A
	9	不用	10 位协议选择（输入）
	连接器外壳	屏蔽	机壳接地

网络电缆的偏压电阻和终端电阻

为了能够把多个设备很容易地连接到网络中，西门子公司提供两种网络连接器：一种标准网络连接器（引脚分配如表 7-6 所示）和一种带编程接口的连接器，后者允许你在不影响现有网络连接的情况下，再连接一个编程站或者一个 HMI 设备到网络中。带编程接口的连接器将 S7-200 的所有信号（包括电源引脚）传到编程接口。这种连接器对于那些从 S7-200 取电源的设备（例如 TD 200）尤为有用。

两种连接器都有两组螺钉连接端子，可以用来连接输入连接电缆和输出连接电缆。两种连接器也都有网络偏置和终端匹配的选择开关。图 7-19 给出了典型的网络连接器的偏压电阻和终端电阻电路图。

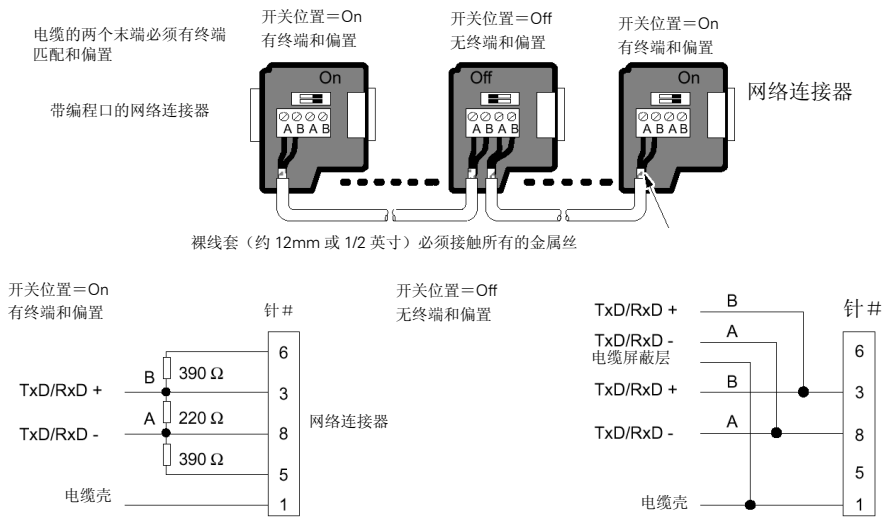


图 7-19　网络电缆的偏置和终端

为网络选择 CP 卡或者 PC/PPI 电缆

如表 7-7 所示，STEP 7-Micro/WIN 支持多种 CP 卡，并允许编程站（计算机或者 SIMATIC 编程器）作为网络的主站。

CP 卡为编程站管理多主网络提供了硬件，并且支持多种波特率下的不同协议。PC/PPI 电缆也允许你使能多主网络。

每一块 CP 卡为网络连接提供了一个单独的 RS-485 接口。CP 5511 PCMCIA 卡有一个提供 9 针 D 型接口的适配器。你可以将通讯电缆的一端接到 CP 卡的 RS-485 接口上，另一端接入网络。如果你使用 CP 卡的 PPI 通讯方式：STEP 7-Micro/WIN 不支持在同一块 CP 卡上同时运行两种应用。在通过 CP 卡将 STEP 7-Micro/WIN 连接到网络之前，你必须关掉另外一种应用。

注意：

使用非隔离的 RS-485 到 RS-232 转换电缆会损坏计算机的 RS-232 端口。

西门子公司的 PC/PPI 电缆（订货号为：6ES7 901-3BF21-0XA0）在 S7-200 CPU 的 RS-485 口和计算机的 RS-232 口之间提供电隔离。如果你不使用西门子公司的 PC/PPI 电缆，你必须为计算机的 RS-232 口提供隔离。

表 7-7　STEP 7-Micro/WIN 支持的 CP 卡和协议

配置	波特率	协议
PC/PPI 电缆[1]连接到编程站的 COM 口	9.6k或19.2k	PPI
CP 5511 类型Ⅱ，PCMCIA 卡（适用于笔记本电脑）	9.6k 到 12M	PPI、MPI 和 PROFIBUS
CP 5611（版本 3 以上）PCI 卡	9.6k 到 12M	PPI、MPI 和 PROFIBUS
MPI SIMATIC 编程器集成的 MPI 口计算机上的 CP 卡(ISA 卡)	9.6k 到 12M	PPI、MPI 和 PROFIBUS

[1] PC/PPI 电缆在 S7-200 CPU 的 RS-485 口和计算机的 RS-232 口之间提供电隔离。使用不隔离的 RS-485 到 RS-232 转换器有可能导致计算机的 RS-232 口损坏。

在网络中使用 HMI 设备

S7-200 CPU 支持西门子公司的多种 HMI 设备，同时也支持其它厂家的产品。有些 HMI 设备（如：TD 200 或 TP 070）使用的通讯协议不允许你选择，另外一些 HMI 设备（如：OP 7 或 TP 170）则允许你为其选择通讯协议。

如果 HMI 设备允许你选择通讯协议，应考虑以下原则：

- 对于直接连接在 S7-200 CPU 通讯端口上的 HMI 设备，如果网络上没有其它设备，你既可以选择 PPI 协议，又可以选择 MPI 协议。
- 对于连接在 EM 277 模块上的 HMI 设备，你可以选择 MPI 或 PROFIBUS。
  - 如果网络中有 S7-300 或 S7-400 PLC，为 HMI 设备选择 MPI 协议。
  - 如果 HMI 设备连接在一个 PROFIBUS 网络中，为 HMI 设备选择 PROFIBUS 协议与其它主站相兼容。
- 如果 HMI 设备所连接的 S7-200 CPU 已经被配置为主站，为 HMI 设备选择 PPI 协议并选中 PPI 高级。MPI 和 PROFIBUS 协议不支持 S7-200 CPU 作主站。

用自由口模式创建用户定义的协议

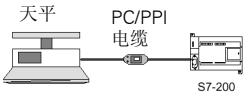
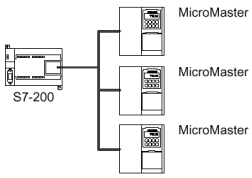
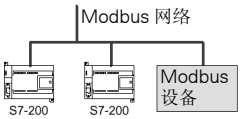
自由口模式允许应用程序控制 S7-200 CPU 的通讯端口。你可以在自由口模式下使用用户定义的通讯协议来实现与多种类型的智能设备的通讯。自由口模式支持 ASCⅡ码和二进制协议。

要使能自由口模式，你需要使用特殊存储器字节 SMB 30（端口 0）和 SMB 130（端口 1）。应用程序中使用以下步骤控制通讯口的操作：

- 发送指令（XMT）和发送中断：发送指令允许 S7-200 的通讯口上发送最多 255 个字节。发送中断通知程序发送完成。
- 接收字符中断：接收字符中断通知程序通讯口上接收到了一个字符。应用程序可以按字符执行操作。
- 接收指令（RCV）：接收指令从通讯口接收整条信息，当接收完成后产生中断通知应用程序。你需要在 SM 存储器中定义条件来控制接收指令开始和停止接收信息。接收指令可以根据特定的字符或时间间隔来启动和停止接收信息。接收指令可以实现多数通讯协议。

自由口模式只有在 S7-200 处于 RUN 模式时才能被激活。将 S7-200 转入 STOP 模式会中断所有自由口通讯，并且通讯端口会按照 S7-200 系统块中的配置转换到 PPI 协议。

表 7-8　使用自由口模式

网络配置		描述
通过 RS-232 连接使用自由口		举例：使用 S7-200 与带 RS-232 接口的电子天平通讯。 • PC/PPI 电缆连在天平的 RS-232 口和 S7-200 的 RS-485 口之间。 • S7-200 CPU 使用自由口与天平通讯。 • 波特率可以是 1200~115.2k。 • 用户程序定义通讯协议。
使用 USS 协议		举例：使用 S7-200 与 SIMODRIVE Micro Master 驱动设备通讯。 • STEP 7-Micro/WIN 提供 USS 库。 • S7-200 CPU 是主站，驱动是从站。  应用示例 参考资料光盘上应用示例中的 USS 例子程序，见 Tip28。
创建用户程序来模仿另外一种网络上的从站设备		举例：把 S7-200 CPU 连到 Modbus 网络。 • S7-200 中的用户程序模仿一个 Modbus 从站。 • STEP 7-Micro/WIN 提供 Modbus 库。  应用示例 参考资料光盘上应用示例中的 Modbus 例子程序，见 Tip41。

使用 PC/PPI 电缆和自由口模式连接 RS-232 接口设备

使用 PC/PPI 电缆和自由口通讯功能，可以将 S7-200 连接到带有 RS-232 兼容标准接口的多种设备。

当数据从 RS-232 接口传输到 RS-485 接口时，PC/PPI 电缆处于发送模式。当空闲或者数据从 RS-485 接口传输到 RS-232 接口时，电缆处于接收模式。当电缆检测到 RS-232 传送线上的字符时，会马上由接收模式转入发送模式。

PC/PPI 电缆支持的波特率为 1200～115.2K。使用 PC/PPI 电缆上的 DIP 开关来为电缆配置正确的波特率。表 7-9 中给出了波特率和开关位置的对应关系。

当 RS-232 传输线从空闲状态切换到接收模式时，需要一个时间周期，这个时间周期被定义为电缆的转换时间。如表 7-9 中所示，电缆的转换时间取决于所选择的波特率。

如果在使用自由口通讯的系统中用 PC/PPI 电缆，在以下情况下，S7-200 的程序中必须考虑转换时间：

表 7-9　转换时间和设置

波特率	转换时间	设置（1＝上）
38400～115200	0.5ms	0 0 0
19200	1.0ms	0 0 1
9600	2.0ms	0 1 0
4800	4.0ms	0 1 1
2400	7.0ms	1 0 0
1200	14.0ms	1 0 1

- S7-200 响应 RS-232 设备发送的信息。
  在 S7-200 接收到 RS-232 设备发送的要求信息之后，S7-200 必须延时一段时间才能发送数据。延时时间应该大于或者等于电缆的转换时间。
- RS-232 响应 S7-200 发送的信息。
  在 S7-200 接收到 RS-232 设备的应答信息之后，S7-200 必须延时一段时间才能发送下一条信息。延时时间应该大于或者等于电缆的转换时间。

在以上两种情况下，延时允许 PC/PPI 电缆有足够的时间从发送模式切换到接收模式，从而使数据能够从 RS-485 端口传送到 RS-232 端口。

在网络中使用 Modem 和 STEP 7-Micro/WIN

STEP 7-Micro/WIN 3.2 版本中使用标准的窗口和电话与 Modem 选项来选择和配置电话线 Modem。电话与 Modem 菜单在 Windows 的控制面板中。使用 Windows 设置菜单来设置 Modem 使你能够；

- 使用 Windows 支持的多数内置和外置 Modem。
- 使用 Windows 支持的多数 Modem 的标准配置。
- 对于选择区域、国家和区域码；选择脉冲或者语音拨号；是否支持电话卡使用标准的 Windows 拨号规则。
- 当与 EM 241 Modem 模块通讯时，使用更高的波特率。

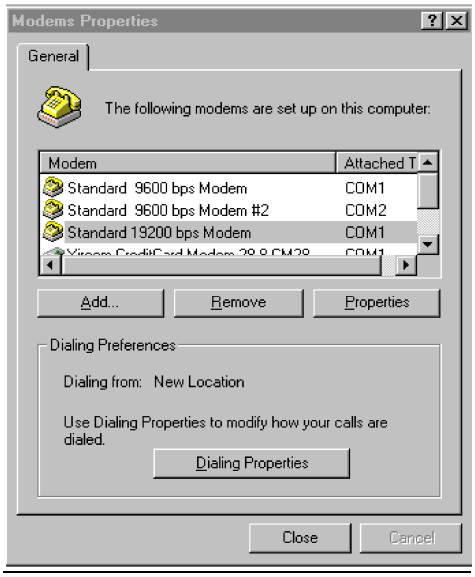


图 7-20　配置本地 Modem

使用 Windows 控制面板可以显示，Modem 属性对话框。这个对话框允许你配置本地 Modem。你可以在 Windows 支持的 Modem 列表中选择你的 Modem。如果你的 Modem 类型没有在 Windows 的 Modem 对话框中列出，你可以选择一个最相似的型号或者与销售商联系，获取 Windows 下的 Modem 配置文件。

STEP 7-Micro/WIN 也支持电台或者移动 Modem。这些 Modem 型号不会出现在 Windows 的 Modem 属性对话框中，但是在 STEP 7-Micro/WIN 中配置之后可以使用。

配置一个 Modem 连接

一个连接有一个标识名与其物理属性相关联。对于一个电话 Modem 来说，这些属性包括：Modem 的类型、选择 10 位或 11 位协议和超时时间。对于移动 Modem 来说，连接允许你设置引脚和其它参数。电台 Modem 属性包括波特率的选择、校验、数据流控制和其它参数。

连接向导

添加一个连接

使用连接向导可以添加一个新的连接，也可以删除或者编辑一个连接，如图 7-21 所示。

1. 双击通讯设置窗口中的图标。
2. 双击 PC/PPI 电缆打开 PG/PC 接口。选择 PC/PPI 电缆并点击属性按钮。在本地连接标签页中，选中 Modem 连接复选框。
3. 再次打开通讯对话框，双击 Modem 连接图标。
4. 点击设置按钮，显示 Modem 连接设置对话框。
5. 点击添加按钮，启动添加 Modem 连接向导。
6. 按照向导配置连接。

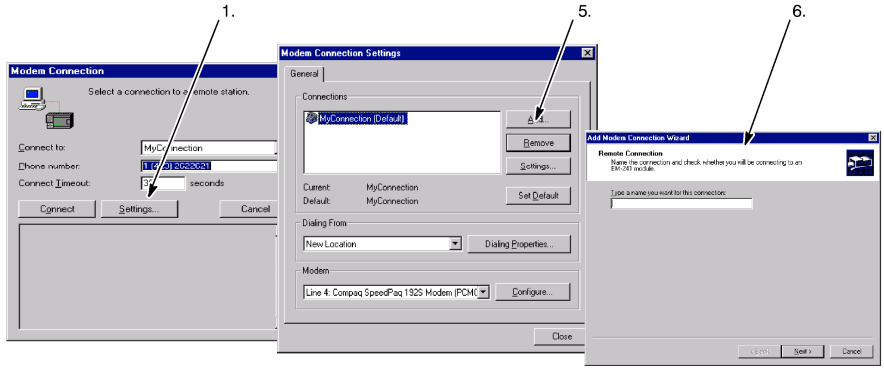


图 7-21　添加一个 Modem 连接

通过 Modem 连接 S7-200

在添加了一个 Modem 连接之后，你可以连接一个 S7-200 CPU。

1. 打开通讯对话框并双击连接图标显示 Modem 连接对话框。
2. 在 Modem 连接对话框中，点击连接按钮对 Modem 拨号。

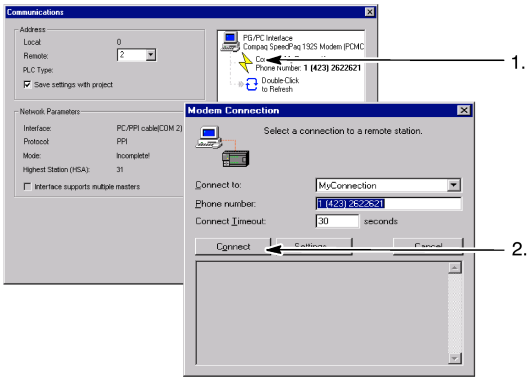


图 7-22　连接 S7-200

配置远端 Modem

Modem 扩展向导

如果远端 Modem 是 EM 241 Modem 模块，则无需配置。

如果你连接一个独立的 Modem、移动 Modem 或者电台 Modem，必须配置连接。Modem 扩展向导使配置连接变得容易。只要简单地选择 Modem 类型并按照向导提示输入信息即可。要得到更多信息，参考在线帮助。

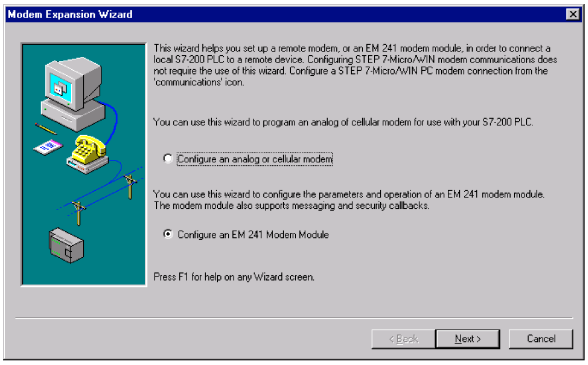


图 7-23　Modem 扩展向导

用 PC/PPI 电缆连接 Modem

你可以使用 PC/PPI 电缆将 Modem 的 RS-232 通讯口连接到 S7-200 CPU。如图 7-24 所示，开关 1、2 和 3 设置 PC/PPI 电缆的波特率。开关 4 选择使用 10 位或 11 位 PPI 协议。开关 5 连接数据通讯设备（DCE）或者数据终端设备（DTE）模式。开关 6（如果有的话）选择是否使用 PC/PPI 电缆上 RS-232 端口的 RTS 信号。

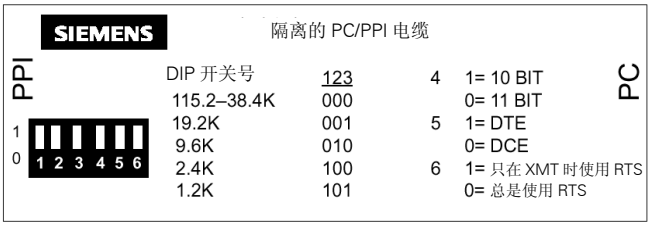


图 7-24　PC/PPI 电缆设置

Modem 通常使用 RS-232 控制信号（如：RTS、CTS 和 DTR），这样计算机可以控制 Modem。当你使用 PC/PPI 电缆连接 Modem 时，你必须配置 Modem 在操作时不使用这些信号。要确认配置 Modem 的命令要求，参考你所使用的 Modem 的资料。

PC/PPI 电缆的开关 4 选择使用 10 位或 11 位 PPI 协议。只有 S7-200 通过 Modem 连接 STEP 7-Micro/WIN 时才使用开关 4。否则，设置开关 4 为 11 位模式来确保其它设备正常运行。

PC/PPI 电缆上的开关 5 允许你设置 RS-232 端口为 DCE 或者 DTE 模式。如果你用 PC/PPI 电缆连接 STEP 7-Micro/WIN 或者计算机，将 PC/PPI 电缆置为 DCE 模式，这样免去了在 PC/PPI 与 Modem 之间安装一个空 Modem 适配器。

但如果 Modem 的连接口为 25 针，你还是需要一个 9 针转 25 针适配器。

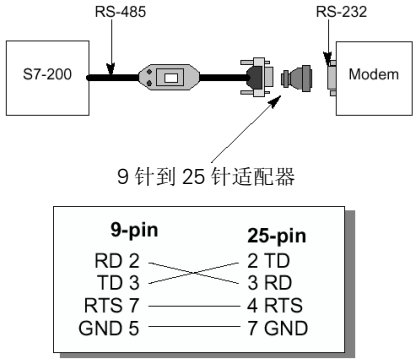


图 7-25　适配器的引脚分配

PC/PPI 电缆的开关 6 选择是否使用 RS-232 连接器的 RTS 信号。选择只在 XMT 时使用 RTS 导致只有当 S7-200 向 RS-485 端口上发送数据时，RTS 信号才被激活，而当 S7-200 接收数据时，RTS 信号被禁止。选择总是使用 RTS 信号导致无论 S7-200 发送数据还是接收数据，PC/PPI 电缆上 RS-232 端口的 RTS 信号始终被激活。只有当 PC/PPI 电缆设置为 DTE 模式时，开关 6 才会影响 RTS 信号。

图 7-25 给出了通用 Modem 适配器的引脚分配。

表 7-10 中给出了 PC/PPI 电缆在 DTE 模式时，RS-485 口和 RS-232 口的引脚和功能定义。表 7-11 中给出了 PC/PPI 电缆在 DCE 模式时，RS-485 口和 RS-232 口的引脚和功能定义。只有在 DTE 模式时，PC/PPI 电缆才支持 RTS 信号。

表 7-10　DTE 模式下的 RS-485 和 RS-232 引脚分配

RS-485 连接器引脚		RS-232 DTE 连接器引脚 [1]	
引脚号	信号说明	引脚号	信号说明
1	地（RS-485 逻辑地）	1	数据载波检测（DCD）（不用）
2	24V 返回（RS-485 逻辑地）	2	接收数据（RD） （输入到 PC/PPI 电缆）
3	信号 B（RxD/TxD）	3	发送数据（TD） （从 PC/PPI 电缆输出）
4	RTS（TTL 电平）	4	数据终端就绪（DTR）（不用）
5	地（RS-485 逻辑地）	5	地（RS-232 逻辑地）
6	+5V（带 100Ω 串联电阻）	6	数据设置就绪（DSR）（不用）
7	24V 电源	7	申请发送（RTS） （从 PC/PPI 电缆输出）
8	信号 A（RxD/TxD）	8	清除发送（CTS）（不用）
9	协议选择	9	振铃指示器（RI）（不用）

[1] 调制解调器需要一个阴到阳的 9 针到 25 针的转换

表 7-11　DCE 模式下的 RS-485 和 RS-232 引脚分配

RS-485 连接器引脚		RS-232 DTE 连接器引针	
RS-485 连接器引脚		RS-232 DTE 连接器引脚 [1]	
引脚号	信号说明	引脚号	信号说明
1	地（RS-485 逻辑地）	1	数据载波检测（DCD）（不用）
2	24V 返回（RS-485 逻辑地）	2	接收数据（RD） （输入到 PC/PPI 电缆）
3	信号 B（RxD/TxD）	3	发送数据（TD） （从 PC/PPI 电缆输出）
4	RTS（TTL 电平）	4	数据终端就绪（DTR）（不用）
5	地（RS-485 逻辑地）	5	地（RS-232 逻辑地）
6	+5V（带 100Ω 串联电阻）	6	数据设置就绪（DSR）（不用）
7	24V 电源	7	申请发送（RTS） （从 PC/PPI 电缆输出）
8	信号 A（RxD/TxD）	8	清除发送（CTS）（不用）
9	协议选择	9	振铃指示器（RI）（不用）

高级议题

优化网络性能

以下因素影响网络性能（波特率和主站的个数是影响网络性能的两个主要因素）：

- 波特率：以网上的所有设备都支持的最高波特率进行网络操作会得到最佳的通讯效果。
- 网络中的主站个数：减少网络中的主站数目可以提高网络性能。网络中的每个主站都会增加网络的负载要求，主站少可以减轻网络负载。
- 主站和从站地址的选择：所有主站的地址应该不带地址间隙，顺序地进行设定。当主站间存在地址间隙时，主站连续检查间隙内的地址，确定是否有其它主站等待进入连接。这个检查需要时间，这样会增加网络的负载。如果主站之间没有地址间隙，就不需要进行检查，这样网络的负载最小。
- 只要从站不位于主站之间，从站地址设置成任何值不会影响网络性能。位于主站之间的从站会造成主站之间的地址间隙，因而会增加网络的负载。
- 间隙刷新因子（GUF）：只有在 S7-200 CPU 作为 PPI 主站时才使用间隙刷新因子，它告诉 S7-200 检查其它主站地址间隙的时间间隔。使用 STEP 7-Micro/WIN 在 CPU 配置中为 CPU 通讯端口设置 GUF。这个配置使 S7-200 周期性地检测地址间隔。如果 GUF＝1，S7-200 每次占有令牌时都会检查地址间隔；如果 GUF＝2，S7-200 每两次占有令牌时，才会检查一次地址间隔。如果主站之间有间隙，设置高的 GUF 可以降低网络负载。如果主站之间没有间隙，GUF 不影响网络性能。由于不频繁检查地址，设置大的 GUF 会造成大的延时。GUF 的缺省设置为 10。

- 最高站地址（HSA）：只有在 S7-200 CPU 作为 PPI 主站时才使用最高站地址，它定义了一个主站寻找其它主站的最高地址。使用 STEP 7-Micro/WIN 在 CPU 配置中为 CPU 通讯端口设置 HSA。设置 HSA 限制了最后一个主站（最高地址）必须检查的地址间隙。限制地址间隙的长度可以最小化寻找和连接另一个主站所需要的时间。最高站地址对于从站地址没有影响。主站仍然可以与地址大于 HSA 的从站通讯。总的规则是应该在所有的主站上设置相同的最高站地址。这个地址应该大于或等于系统中的最高主站地址。S7-200 CPU 对于最高主站的缺省值是 31。

为网络计算令牌循环时间

在令牌传送网络中，只有拥有令牌的站有初始化通讯的权限。令牌循环时间（令牌在逻辑环中所有主站传递一周所需要的时间）标志着网络的性能。

图 7-26 为计算一个多主网络的令牌循环时间给出了一个网络实例。在这个例子中，TD 200（3 号站）与 CPU 222（2 号站）通讯；TD 200（5 号站）与 CPU 222（4 号站）通讯，以此类推。两个 CPU 224 使用网络读写指令从其它 S7-200 采集数据：CPU 224（6 号站）向 2 号站、4 号站和 8 号站发送数据；同时 CPU 224（8 号站）向 2 号站、4 号站和 6 号站发送数据。在该网络中，有 6 个主站（4 个 TD 200 和两个 CPU 224）和两个从站（两个 CPU 222）。

应用示例

要得到更多关于令牌循环时间的信息，参考资料光盘上的应用示例，见 Tip42。

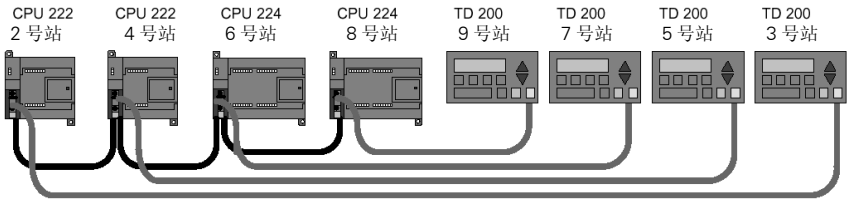


图 7-26　令牌传送网络举例

主机要发送信息，必须持有令牌。例如：当站 3 持有令牌时，它初始化到站 2 的请求，然后把令牌传给站 5。站 5 才能初始化到站 4 的请求信息，然后把令牌传给站 6。站 6 再初始化到站 2、4 或 8 的请求信息，然后把令牌传给站 7。这个初始化信息和传送令牌的过程会在逻辑环中持续进行，从站 3 到站 5，又到站 6、7、8、9，最后又返回站 3。主机要能够发出请求信息，这个令牌必须在逻辑环中完整循环。对于一个 6 个站的逻辑环，如果每个令牌持有者发送一个请求信息，为一双字值（4 个字节），则令牌循环时间在 9600 波特下为 900ms。增大每条信息存取的字节数或增加站数，都会加大令牌循环时间。

令牌循环时间是由各站占有令牌的时间决定的。对于多主网络，令牌循环时间可以由各主站占有令牌时间相加得出。如果允许 PPI 主站模式（在网络中使用 PPI 协议），S7-200 可以使用网络读写指令向其它 S7-200 发送信息。如果采用这些指令发送信息，并且下述假设成立的话，令牌循环时间可以由公式近似得出：各站在每次占有令牌时发送一个请求；该请求为数据连续的读或写请求；CPU 的通讯缓冲区使用没有冲突；CPU 的扫描时间不超过 10ms。

令牌占有时间（T_{hold}）＝（128 额外＋n 数据）字符×11 位/字符×1/波特率
令牌循环时间（T_{rot}）＝主站 1 的 T_{hold}＋主站 2 的 T_{hold}＋…＋主站 m 的 T_{hold}
n 为数据的字符（字节）数 m 为主站数

使用上面例子计算令牌循环时间，6 台主机中每个主机都有相同的令牌占用时间。

T（令牌占用时间）＝（128＋4）字符×11 位/字符×1/9600 位=151.25ms/主机

T（令牌循环时间）＝151.25ms/主机×6 主机＝907.5ms

（一位时间等于一个信号的持续时间）

提示：

SIMATIC NET COM PROFIBUS 软件提供对网络性能的分析器软件。

令牌循环时间比较

表 7-12 中给出了在不同通讯站个数、数据量以及波特率下的令牌循环时间比较。令牌循环时间考虑的是使用网络读写指令的情况。

表 7-12 令牌循环时间（单位：秒）

波特率	传输	主站的个数								
	字节数	2	3	4	5	6	7	8	9	10
9.6k	1	0.30	0.44	0.59	0.74	0.89	1.03	1.18	1.33	1.48
	16	0.33	0.50	0.66	0.83	0.99	1.16	1.32	1.49	1.65
19.2k	1	0.15	0.22	0.30	0.37	0.44	0.52	0.59	0.67	0.74
	16	0.17	0.25	0.33	0.41	0.50	0.58	0.66	0.74	0.83
187.5k	1	0.009	0.013	0.017	0.022	0.026	0.030	0.035	0.039	0.043
	16	0.011	0.016	0.021	0.026	0.031	0.037	0.042	0.047	0.052

理解网络设备的链接

网络设备通过连接来实现通讯，连接是主站与从站之间的单独链接。如图 7-27 所示，不同通讯协议的连接是不一样的：

- PPI 协议中所有网络设备共享一个连接。
  PPI 高级、MPI 和 PROFIBUS 协议中，一个主站和一个从站已经建立起来的连接不会被第二个主站干扰。

S7-200 CPU 和 EM 277 总是为 STEP 7-Micro/WIN 和 HMI 设备各保留一个连接。其它主站设备不能使用这些被保留的连接。这就保证了当正在使用诸如 PPI 高级这样的协议时，在连接其它主站的同时，至少可以连接一个编程站或 HMI 设备到 S7-200 CPU 或 EM 277。

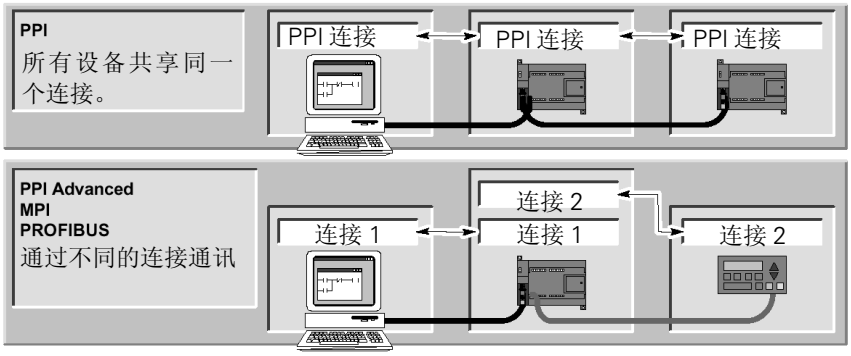


图 7-27　管理通讯连接

如表 7-13 所示，S7-200 CPU 或 EM 277 提供一定数量的连接。S7-200 CPU 的每一个端口（通讯口 0 和通讯口 1）支持 4 个独立的连接。（这意味着 S7-200 CPU 最多允许 8 个连接）这除去了共享的 PPI 连接。一个 EM 277 支持 6 个连接。

表 7-13　S7-200 CPU 和 EM 277 模块的连接个数

模块	波特率	连接数	支持的协议
S7-200 CPU　通讯口 0	9.6k、19.2k 或 187.5k	4	PPI、PPI 高级、MPI 和 PROFIBUS[1]
通讯口 1	9.6k、19.2k 或 187.5k	4	PPI、PPI 高级、MPI 和 PROFIBUS[1]
EM 277 模块	9.6k 到 12M	每块 6 个	PPI 高级、MPI 和 PROFIBUS

[1] 对于 S7-200 CPU 的通讯口 0 和通讯口 1，只有和 S7-200 从站通讯时，才能使用 MPI 和 PROFIBUS 协议。

配置复杂网络

对于 S7-200 来说，典型的复杂网络有多个 S7-200 主站，并且在 PPI 网络上用网络读写指令与其它设备通讯。复杂网络有可能产生一些特殊的问题，主站与从站的通讯有可能堵塞。

如果网络运行在较低的波特率下（9.6k 或 19.2k），每个主站在传送令牌之前完成操作（读或写）。在 187.5k 波特率下，主站对从站提出要求然后传送令牌。这就对从站提出了较高要求。图 7-28 中给出了一个有潜在的网络冲突的实例。在网络中，1 号站、2 号站和 3 号站是主站，使用网络读写指令与 4 号站通讯。网络读写指令使用 PPI 协议，因此所有 S7-200 共享 4 号站中的一个连接。

在本例中，1 号站对 4 号站提出请求。对于高于 19.2k 的波特率，1 号站将令牌传送给 2 号站。如果 2 号站也试图向 4 号站提出请求，其请求会被拒绝，因为 4 号站正在处理 1 号站的请求。在 4 号站完成对 1 号站的响应之前，所有请求都会被拒绝。只有在响应完成之后，4 号站才能接受其它主站的请求。

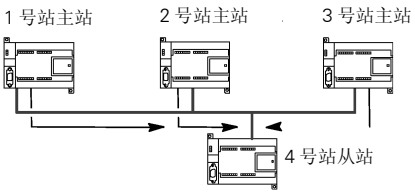


图 7-28　通讯冲突

为了避免 4 号站通讯口上的通讯冲突，应考虑使 4 号站成为网络上唯一的主站，如图 7-29 所示。4 号站可以向其它 S7-200 提出读写请求。

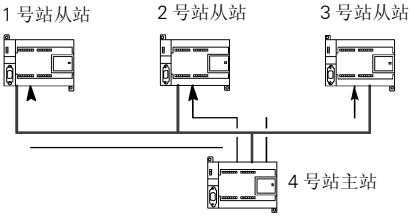


图 7-29　避免冲突

这样的配置不仅能够确保没有通讯冲突，而且减少了多主站导致网络负担，使网络更高效的运行。

对于某些应用来说，无法减少网络上的主站数量。当网上有多个主站时，你必须对令牌循环时间进行管理，并确保网络的令牌循环时间不超过目标值。（令牌循环时间是指一个主站传送令牌到再次得到令牌的时间间隔。）

如果令牌回到主站的时间长于令牌循环时间目标值，该主站不能提出请求。只有当令牌循环时间小于目标值时，主站才能提出请求。

最高站地址（HSA）和 S7-200 的波特率设置决定了令牌循环时间。表 7-14 给出了令牌循环时间目标值列表。

表 7-14　HSA 和令牌循环时间目标值

HSA	9.6k	19.2k	187.5k
HSA=15	0.613s	0.307s	31ms
HSA=31	1.040s	0.520s	53ms
HSA=63	1.890s	0.950s	97ms
HSA=126	3.570s	1.790s	183ms

对于较低的波特率，如：9.6k 或者 19.2k，主站会在传送令牌前等待应答。由于请求/应答的过程周期在扫描时间中占相当长的时间，因而很有可能每个网络上的主站在占有令牌时都作好了请求发送数据的准备。实际令牌循环时间增加并且有些主站可能不能发送任何请求。在某些情况下，某个主站只能偶尔处理其请求。

对于实际令牌循环时间大于目标值的情况，你有两种选择改善它：

- 可以通过减少网上的主站的方式来缩短令牌循环时间。
- 可以用增大网上所有主站的 HSA 的方法来增大令牌循环时间的目标值。

增大 HSA 的值会导致其它问题，它会延长 S7-200 切换到主站模式进入网络的时间。如果你使用一个定时器来确保网络读写指令在指定的时间内完成，初始化主站模式和 S7-200 作为主站进入网络时间的延时，会导致指令超时。你可以用减小间隙刷新因子（GUF）的方法来使主站进入网络的时间最小化。

由于在 187.5k 波特率时，请求传送并驻留在从站中需要一定时间，因而在选择令牌循环时间目标值时应留有余量。在 187.5k 的波特率下，实际令牌循环时间应大约为目标值的一半。

要计算令牌循环时间，使用表 7-12 中完成网络读写指令所需要的时间数据。要计算 HMI 设备（例如 TD 200）的时间需求，使用传送 16 字节的数据。将网上每个设备的时间需求加在一起，就可以计算出令牌循环时间。将所有时间需求加在一起，意味着网上所有设备在同一个令牌循环里都有请求这种最坏的情况。这样计算出的时间是网络令牌循环时间的最大值。

例如：考虑一个网络运行在 9.6k 波特率下，有 4 个 TD 200 和 4 个 S7-200，每个 S7-200 每秒钟向另一个 S7-200 写 10 个字节。用表 7-12 计算网络的传输时间为：

4 个 TD 200 传输 16 个字节数据＝0.66s

4 个 S7-200 传输 10 个字节数据＝0.63s

总的令牌循环时间　　＝1.29s

为了允许在一个令牌循环中有足够的时间处理所有请求，将 HSA 设为 63（见表 7-14）。选择的令牌循环目标值为 1.89s，大于最大令牌循环时间 1.29s，因而确保了每个设备在每一个令牌循环中都可以传输数据。

为了提高多主网络的可靠性，你还必须考虑以下措施：

- 增大 HMI 设备的刷新时间。例如：将 TD 200 的刷新速率由“尽快”改为“每秒一次”。
- 减少网络读写指令的请求数量（减少处理请求的网络负担）。例如：将两次读 4 个字节的网络读指令操作，合并为一次读 8 个字节的网络读指令操作。两次读 4 个字节操作需要的时间会远远大于一次读 8 个字节操作。
- 改变 S7-200 主站的刷新时间，使其不要试图使刷新时间小于令牌循环时间。

硬件故障诊断指导和软件调试工具

8

STEP7-Micro/WIN 提供软件工具帮助你调试和测试你的程序。这些特性包括：监视 S7-200 正在执行的用户程序，为 S7-200 指定运行程序的扫描次数，强制变量值等。

使用表 8-1 作为 S7-200 硬件故障诊断和找到解决方案的指导。

本章内容：

调试应用程序

STEP 7-Micro/WIN 为帮助用户调试程序提供了多种手段：

书签、交叉参考表和运行模式下编辑。

使用书签使编程更方便

你在程序中可以使用书签，它可以使你在一个很长的程序中，很方便的在编辑行之间前后移动。你可以移动到前一个标签行或者下一个标签行。

使用交叉参考表来检查应用程序

交叉参考表中能够显示应用程序中的交叉参考和元件使用信息。交叉参考表能够识别程序中使用的所有操作数；程序块、程序段或者程序行的位置以及每一块使用该操作数的相关指令。

你可以在符号地址和绝对地址之间切换来改变所有操作数的表现形式。

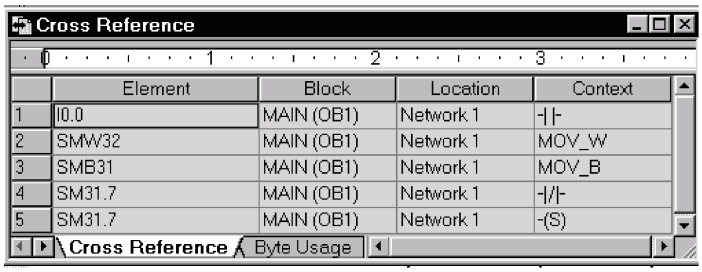


图 8-1　交叉参考表

提示：

在交叉参考表中双击某一行可以切换到程序中的相应位置。

在 RUN 模式下编辑应用程序

CPU 224 Rel.1.10 以上及 CPU 226 Rel. 1.00 以上的 CPU 模块支持 RUN 模式下编辑程序的功能。RUN 模式下编辑功能可以在对控制过程影响较小的情况下，对用户程序进行少量修改。执行该功能时也允许你对程序进行大量改动，但这样作对程序的执行影响较大，甚至是危险的。

警告：

当在 RUN 模式下向 S7-200 下载修改过的程序时，修改的程序将立即影响过程操作。在 RUN 模式下修改程序会导致不可预见的系统操作，可能会导致严重的人身伤害和财产损失。

只有了解 RUN 模式下修改程序对系统运行会造成何种影响的被授权人员，才可以执行在 RUN 模式下编辑程序。

要在 RUN 模式下编辑应用程序，在线的 S7-200 CPU 必须支持 RUN 模式下编辑，并且该 CPU 必须处于 RUN 状态。

1. 在命令菜单中选择 Debug>Program Edit in RUN。
2. 如果你打开的项目与 S7-200 中的程序不同，将提示您存盘。RUN 模式下编辑功能只能编辑 CPU 中的程序。
3. STEP 7-Micro/WIN 对于你将在 RUN 模式下编辑程序提出警告，提示您是继续下一步还是取消操作。如果你选择继续，STEP 7-Micro/WIN 会在 S7-200 中上载程序。现在你可以在 RUN 模式下编辑程序了。编辑中没有严格的限定。

提示：

上升沿（EU）和下降沿（ED）指令带一个操作数。若需要查看有关边沿指令的信息，在屏幕上的示窗部分选择交叉参考图标，边沿指令使用标签页中列出了程序中使用的边沿指令的号码。在编辑应用程序时，请注意不要使用重复的号码。

在 RUN 模式下载程序

RUN 模式编辑功能允许在 S7-200 处于 RUN 模式时下载程序块。在下载程序块之前，考虑到 RUN 模式下编辑对 S7-200 操作的影响，请注意以下情况：

- 如果在 RUN 模式编辑状态下取消一个输出控制逻辑，则输出在下一次 CPU 上电之前或 CPU 转换到 STOP 模式前将保持上一个状态。
- 如果在 RUN 模式编辑状态下取消一个正在运行的 HSC 或 PTO/PWM 功能，则这些功能在下一次 CPU 上电或 CPU 转换到 STOP 模式前将保持运行状态。
- 如果在 RUN 模式编辑状态下取消 ATCH 指令，但没有删除中断程序，则在下一次 CPU 上电或 CPU 转换到 STOP 模式之前将继续执行中断。同样，如果删除 DTCH 指令，在下一次 CPU 上电之前或 CPU 转换到 STOP 模式前中断将不会停止。
- 如果在 RUN 模式编辑状态下加入 ATCH 指令，并且满足第一次扫描标志的条件，则在下一次 CPU 上电或 CPU 从 STOP 转换到 RUN 模式前不会执行这些指令。
- 如果在 RUN 模式编辑状态下取消 ENI 指令，则在下一次 CPU 上电之前或 CPU 从 RUN 转换到 STOP 模式前将继续执行中断。
- 如果在 RUN 模式编辑状态下修改接收指令的地址表，并且在旧程序向新程序转换时接收指令处于激活状态，则所接收的数据写入旧地址表。NETR 和 NETW 指令同样如此。
- 由于 RUN 模式编辑不影响第一次扫描标志，因此在下一次 CPU 上电之前或 CPU 从 STOP 转换到 RUN 模式前第一次扫描标志的逻辑条件不执行。

提示：

在 RUN 模式下载应用程序，S7-200 必须支持 RUN 模式下编辑；程序编译必须没有错误；STEP 7-Micro/WIN 与 S7-200 之间的通讯必须畅通。

你只能下载程序块。

退出 RUN 模式编辑

要退出 RUN 模式编辑，在命令菜单中选择 **Debug>Program Edit in RUN**，然后点击 Checkmark 即可。如果你修改完后没有存盘，STEP-7 Micro/WIN 会提示你是继续编辑、下载并退出 RUN 编辑模式或者不下载退出。

显示程序状态

STEP 7-Micro/WIN 允许你在程序执行时监视其状态。当你监视程序状态时，程序编辑器会显示指令操作数的值。

要显示程序状态，可以点击程序状态按钮，也可以在命令菜单中选择 **Debug>Program Status**。

显示 LAD 和 FBD 程序的状态

对于显示 LAD 和 FBD 程序的状态，STEP 7-Micro/WIN 提供了两种选择：

- 扫描结束的状态：STEP 7-Micro/WIN 在经过多个扫描周期得到显示状态值之后，刷新屏幕显示状态。状态显示并不反映程序执行时每个元素的实际状态。扫描结束状态不显示 L 存储器或者累加器的状态。
  对于扫描结束状态显示，状态值在所有 CPU 操作模式下都刷新。
- 执行状态：STEP 7-Micro/WIN 在 S7-200 程序执行过程中，显示程序段中的状态值。要显示执行状态，在命令菜单中选择 **Debug>Use Execution Status**。
  对于执行状态，状态值只有在 CPU 处在 RUN 模式时才刷新。

提示：

STEP 7-Micro/WIN 提供一种简单的方法来改变变量的状态。只要选择变量并且用右键单击就会显示选择菜单。

对 LAD 和 FBD 程序中的状态显示进行配置

STEP 7-Micro/WIN 为在程序中显示状态提供了多种选择。要为程序状态显示作配置，在命令菜单中选择 **Tools>Options**，然后选择程序编辑器标签页即可，如图 8-2 所示。

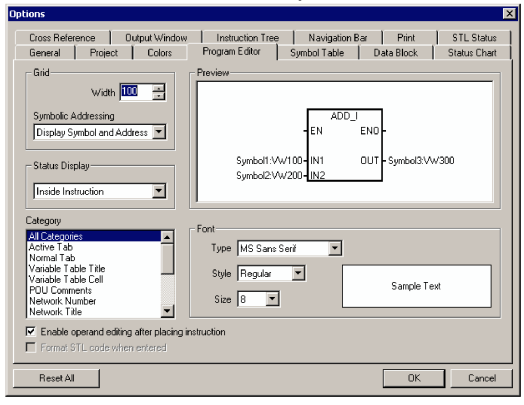


图 8-2 显示状态选择

显示 STL 程序的状态

你可以监视逐条指令编程的 STL 程序的执行状态。对于一个 STL 程序，STEP 7-Micro/WIN 在屏幕上显示指令的状态。

从编辑器窗口顶部的第一条 STL 语句开始，STEP 7-Micro/WIN 的 S7-200 采集状态信息。当你滚动编辑器窗口的屏幕时，新的信息从 S7-200 采集上来。

STEP 7-Micro/WIN 不断地刷新屏幕上的数值。要使屏幕刷新暂停，选择触发暂停按钮。当前值会保持在屏幕上，直到触发暂停按钮失效。

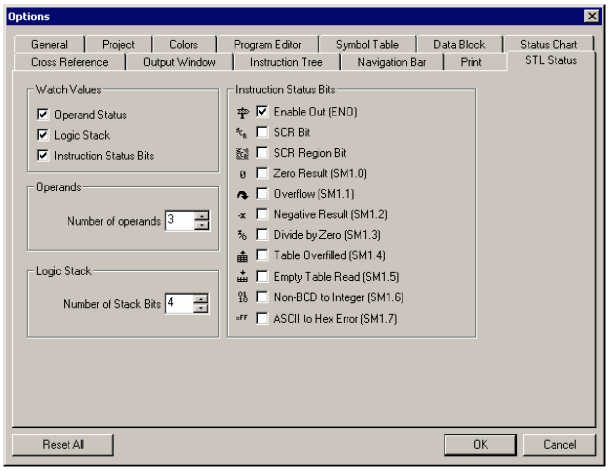


图 8-3　显示（STL）状态选项

为 STL 程序中显示的参数作配置 STEP 7-Micro/WIN 允许你在 STL 指令中显示多种参数状态。在命令菜单中选择 **Tools>Options**，然后选择 STL 状态标签页，如图 8-3 所示。

使用状态图来显示和修改 S7-200 中的数据

状态图允许你在 S7-200 运行程序时，读、写、强制和监视变量数据。在命令菜单中选择 **View>Component>Status Chart** 来创建一个状态图。图 8-4 中给出了一个状态图的例子。

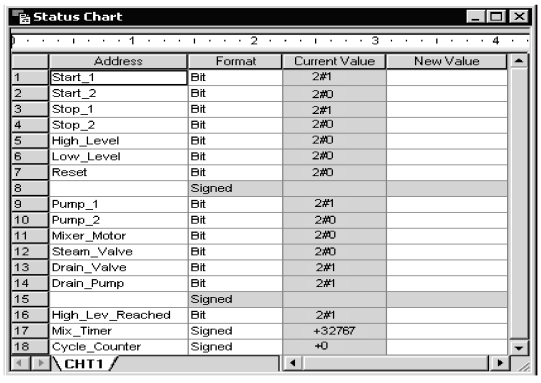


图 8-4　状态图

你可以创建多个状态图。

STEP 7-Micro/WIN 提供工具条图标来对状态图进行操作：升序排列、降序排列，单次读、全部写、强制、全部强制、和读所有强制值。

要为单元选择格式，选择该单元并单击鼠标右键会显示关联菜单。

强制指定值

S7-200 CPU 允许你用指定值来强制赋给一个或所有的 I/O 点（I 和 Q 位）。另外你也可以强制改变最多 16 个内部存储器数据（V 或 M）或模拟 I/O 量（AI 或 AQ）。V 和 M 存储器变量可以按字节、字或双字来改变。模拟量只能以字方式改变，以偶字节开始（如 AIW6 或 AQW14）。所有改变值存于 CPU 的固定 EEPROM 存储器中。

因为在扫描周期的不同阶段（执行程序、或 I/O 更新、或通讯处理阶段）可能会改变强制数据。所以在扫描周期的不同时间，CPU 又使用了这些强制变量。图 8-5 显示的扫描周期中，当 CPU 更新这些强制变量时，以高亮状态显示。

强制功能不仅取代了直接读或写，同时也取代了当 CPU 变为 STOP 方式时，以某个指定值输出：当 CPU 变为 STOP 方式后，输出将为强制值，而不是设置值。

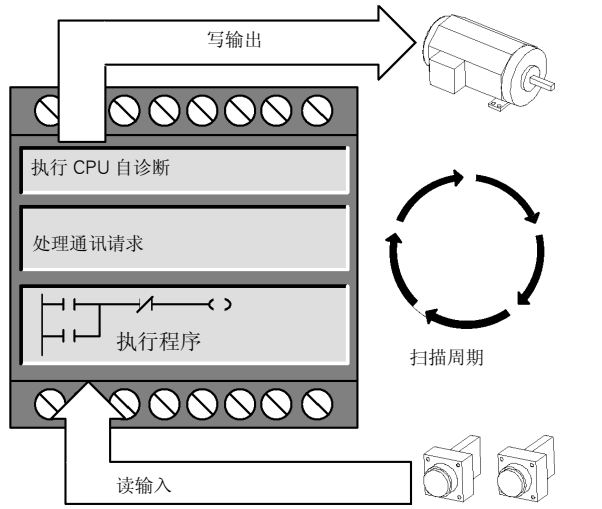


图 8-5　S7-200 扫描周期

- 读输入：S7-200 使用强制值作为输入值
- 在程序中执行控制逻辑：S7-200 使用强制值作为所有立即 I/O 值。程序执行后，强制数据最多可用于 16 个存储器。
- 处理通讯请求：强制数据用于所有读/写通讯。
- 写输出：S7-200 使用强制值作为输出。你可以使用状态表来强制变量，要强制一个新值，将其输入到状态表的新值列中，然后按工具条上的强制钮。要将一个已经存在的当前值变为强制值，在当前值列中选中该变量，然后按强制钮。

提示

强制功能优于立即 I/O 指令。强制功能同样优于在 STOP 模式下使用的输出表。如果 S7-200 进入 STOP 模式，输出点上为强制值而不是输出表中配置的值。

指定程序执行的扫描周期数

为了帮助用户调试程序，STEP 7-Micro/WIN 允许用户指定程序执行的扫描周期数。

你可以使让 S7-200 在首次扫描时执行一次程序。这使你能够监视首次扫描之后，S7-200 中的数据。在命令菜单中选择 **Debug>First Scan** 即可。

你可以指定程序执行的扫描次数（从 1 到 65,535）。这使你能在变量变化时监视程序。在命令菜单中选择 **Debug>Multiple Scans**，然后指定扫描次数即可。

硬件故障诊断指导

表 8-1 S7-200 硬件故障诊断指导

问题	可能原因	解决方法
输出不工作	• 被控制的设备产生了损坏（输出的电气浪涌） • 程序错误 • 接线松动或不正确 • 输出过载 • 输出被强制	• 当接到感性负载时，(例如电机或继器)，需要使用一个抑制电路，参考第三章。 • 修改程序。 • 检查接线，如果不正确，要改正。 • 检查输出的负载功率。 • 检查 CPU 是否有被强制的 I/O。
CPU SF(系统故障) 灯亮	下面给出了可能的原因： • 用户程序错误 0003 看门狗错误 0011 间接寻址 0012 非法的浮点数 0014 范围错误 • 电气干扰 （0001 到 0009） • 元件损坏 （0001 到 0010）	读出致命错误代码号，并参考附录 C。 • 对于编程错误，检查 FOR，NEXT，JMP，LBL 和比较指令的用法。 • 对于电气干扰： 参考第三章的接线指南。控制面板良好接地和高电压与低电压不并行引线是很重要的。 把 24VDC 传感器电源的 M 端子接到地。
LED 灯全部不亮	• 保险丝烧断 • 24V 供电线接反 • 不正确的供电电压	把电源分析器连接到系统，检查过电压尖峰的幅值和持续时间。根据检查结果，给系统加一个合适的抑制设备。 有关现场接线的安装信息，请参考第三章。
电气干扰问题	• 不合适的接地 • 在控制柜内交叉配线 • 对快速信号配置了输入滤波器	参考第三章的接线指南。控制面板良好接地和高电压与低电压不并行引线是很重要的。把 24VDC 传感器电源的 M 端子接到地。增加系统数据块中的输入滤波器的延迟时间。
当连接一个外部设备时通讯网络损坏。 (计算机接口、PLC 的接口或 PC/PPI 电缆损坏)	如果所有的非隔离设备(例如 PLC、计算机或其它设备)连到一个网络，而该网络没有共同的参考点，通讯电缆提供了一个不期望的电流通路。这些不期望的电流可以造成通讯错误或损坏电路。	• 参考第三章的接线指南和第七章的网络指南。 • 购买隔离型 PC/PPI 电缆。 • 当连接没有共同电气参考点的机器时，购买隔离型 RS-485 到 RS-485 中继器。
STEP 7-Micro/WIN 32 通讯问题		有关网络通讯的信息请参考第七章。
错误处理		有关错误代码的信息请参考附录 C。

创建位控模块程序

9

EM253 位控模块是 S7-200 的特殊功能模块，它能够产生脉冲串，用于步进电机和伺服电机的速度和位置的开环控制。它与 S7-200 通过扩展的 I/O 总线通讯，它带有八个数字输出，作为智能模块出现在 I/O 组态中。

位控模块能够产生移动控制所需的脉冲串，其组态信息存储在 S7-200 的 V 存储区中。

为了简化您应用程序中位控功能的使用，STEP 7-Micro/WIN 提供的位控向导可在几分钟内完成对位控模块的组态。STEP 7-Micro/WIN 还提供一个控制面板，可以控制、监视和测试您的位控操作。

本章内容：

位控模块的特性

位控模块可提供单轴、开环移动控制所需要的功能和性能：

- 提供高速控制，从每秒 12 个脉冲至每秒 200，000 个脉冲
- 支持急停（S 曲线）或线性的加速、减速功能
- 提供可组态的测量系统，既可以使用工程单位（如英寸或厘米）也可以使用脉冲数
- 提供可组态的螺距误差补偿
- 支持绝对、相对和手动的位控方式
- 提供连续操作
- 提供多达 25 组的移动包络，每组最多可有 4 种速度
- 提供 4 种不同的参考点寻找模式，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
- 提供可拆卸的现场接线端子便于安装和拆卸。

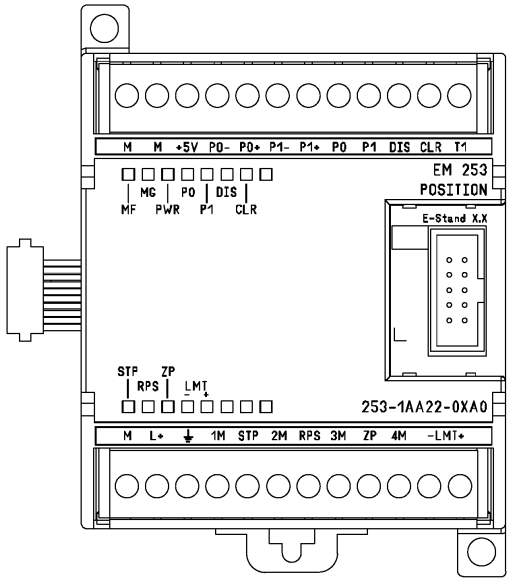


图 9-1 EM253 位控模块

使用 STEP 7-Micro/WIN 可生成位控模块所使用的全部组态和移动包络信息。

这些信息和您的程序块一起下载到 S7-200 中。由于位控模块所需要的全部信息都存储在 S7-200 中，当您更换位控模块时不必重新编程或组态。

S7-200 在输出的过程映像区中（Q 区）保留 8 位作为位控模块的接口。S7-200 的应用程序将使用这些位来控制位控模块的操作。这 8 个输出位与位控模块上的任何物理输出都不相连。

位控模块提供 5 个数字输入和 4 个数字输出与您的移动控制应用相连，见表 9-1。这些输入输出位于位控模块上。附录 A 提供了关于位控模块的详细指标，还包括位控模块与一些通用电机驱动器/放大器的接线图。

表 9-1 位控模块的输入和输出

信号	描述
STP	STP 输入可让模块停止脉冲输出。在位控向导中可选择您所需要的 STP 操作。
RPS	RPS（参考点切换）输入可为绝对移动操作建立参考点或零点位置。
ZP	ZP（零脉冲）输入可帮助建立参考点或零点位置。通常，电机驱动器/放大器每周产生一个 ZP 脉冲
LMT+ LMT-	LMT+和 LMT-是移动位置的最大限制。位控向导中可以组态 LMT+和 LMT-输入。
P0 P1 P0+、P0- P1+、P1-	P0 和 P1 是漏型晶体管输出，用以控制电机的移动和方向。P0+、P0-以及 P1+、P1-是差分脉冲输出，与 P0 和 P1 的功能一样，但所提供的信号质量更好。漏型输出和差分输出同时有效。您可以根据电机驱动器/放大器的接口要求来选择使用哪种输出。
DIS	DIS 是一个漏型输出，用来禁止或使能电机驱动器/放大器。
CLR	CLR 是一个漏型输出，用来清除伺服脉冲计数器。

位控模块编程

STEP 7-Micro/WIN 为位控模块的组态和编程提供便捷的工具。遵循以下步骤即可：

1. 组态位控模块。STEP 7-Micro/WIN 提供一个位控向导，可生成组态/包络表和位控指令。有关组态位控模块的信息请参见 9-4 页。
2. 测试位控模块的操作。STEP 7-Micro/WIN 提供一个 EM253 控制面板，用以测试位控模块的输入、输出接线组态以及移动路径的执行。有关 EM253 控制面板的信息见 9-30 页。
3. 创建 S7-200 的执行程序。位控向导自动生成位控指令。您可以将这些指令插入您的程序中。有关位控指令的信息请参见 9-16 页。将以下指令插入您的用户程序当中：
   - 要使能位控模块，插入一个 POSx_CTRL 指令。用 SM0.0（始终接通）以确保这条指令在每一个循环周期中都得到执行。
   - 要将电机移至一个指定的位置，使用一条 POSx_GOTO 或一条 POSx_ RUN 指令。POSx_GOTO 指令使电机移动到您在程序中输入的指定位置。POSx_RUN 指令则使电机按照您在位控向导中所组态的路线移动。
   - 要使用绝对座标进行移动，您必须为您的应用建立零位置。使用一条 POSx_RSEEK 或一条 POSx_LDPOS 指令建立零位置。
   - 位控向导生成的其它指令为典型应用提供功能，对于您特定的应用来说，这些指令是可选的。
4. 编译您的程序并将系统块、数据块和程序块下载到 S7-200 中。

组态位控模块

位控向导

要进行位移控制必须为位控模块创建组态/包络表。位控向导引导您一步一步完成整个组态过程，非常便捷。有关组态/包络表的详细信息请参考 9-34 页的高级议题。

使用位控向导可离线创建组态/包络表。您可以在不连接 S7-200CPU 及位控模块的情况下进行组态。

要运行位控向导，必须对项目进行编译并选择符号寻址方式。

启动位控向导，可以点击浏览条中的工具图标，然后双击位控向导图标，或者选择菜单命令 Tools>Motion Control Wizard

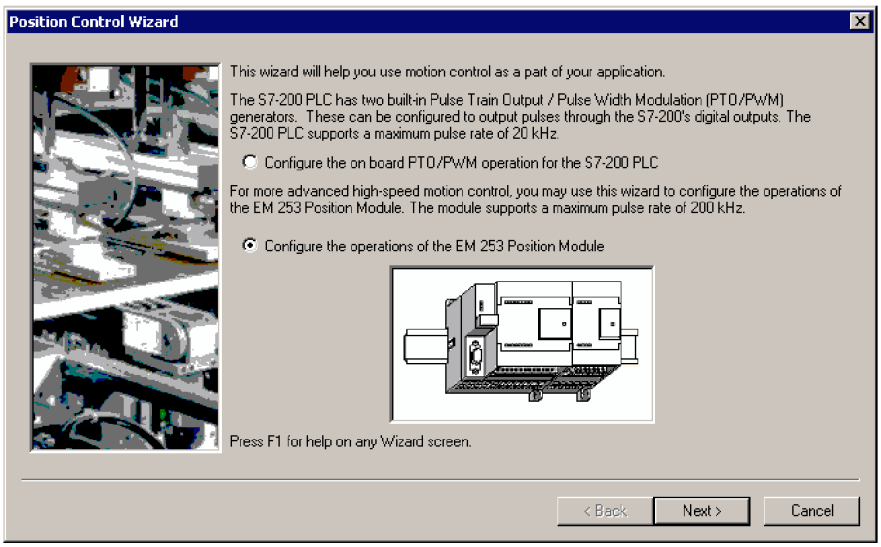


图 9-2 位控向导

输入位控模块的位置

您必须输入模块类型和位置以便定义模板参数并为您的应用定义移动包络。位控向导可自动读取智能模块的位置，从而减化这个任务。您只需点击读模块按钮。

对于硬件版本 1.2 之前的 S7-200 CPU，智能模块必须安装在紧邻 CPU 的位置以便使用位控向导对模块进行组态。

选择测量类型

您必须选择测量系统，以便在整个组态中使用。您可以选择使用工程单位或脉冲。如果您选择脉冲则不必再定义其它信息。如果选择工程单位，您必须输入以下数据：使电机转一周所需的脉冲数（参考电机或驱动的参数），测量的基本单位（如英寸、英尺、米或厘米），以及电机转一周所引起的位移量（或“单位”）。STEP 7-Micro/WIN 提供一个 EM253 控制面板，对已组态的位控模块，通过该面板可修改每周的单位数。

如果您在以后改变了测量系统，必须删除整个组态，包括位控向导生成的所有指令。您必须输入与新的测量系统一致的选项。

编辑缺省的输入和输出组态

位控向导提供一个高级选项，利用这个选项，您可以对位控模块的输入和输出的缺省组态进行查看和编辑：

- 在输入激活等级标签页可改变激活的等级设置。等级设为高时，当输入有电流时，读到逻辑 1。等级设为低时，当输入无电流时，读到逻辑 1。逻辑 1 总是解释为条件激活。不论激活等级是怎样的（缺省=高），输入有电流时 LED 灯亮。
- 输入滤波时间标签页可用来为 STP、RPS、LMT+、LMT-输入的滤波定义延时时间（范围为 0.20ms 至 12.80ms）。延时可帮助滤除输入线上的噪声，以免除输入状态的因干扰而发生改变。
- 脉冲和方向输出标签页可用来指定控制方向的方式。您必须首先指定输出的极性。

选择正极性

对于使用正极性的应用，选择下列方式之一（见图 9-3）以配合您的驱动设备以及移动的方位：

- 位控模块从 P0 发出正转脉冲，从 P1 发出反转脉冲。
- 位控模块从 P0 发脉冲的同时，如果 P1 接通正转，如果 P1 断开反转。（这是缺省设置）

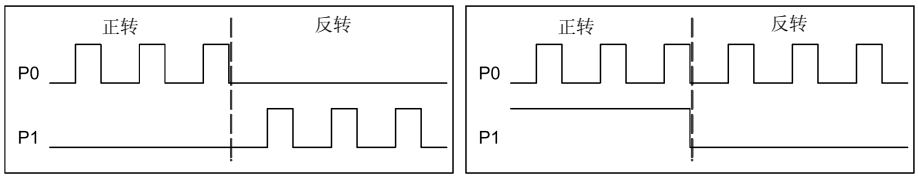


图 9-3 正极性的转向选项

选择负极性

对于使用负极性的应用，选择下列方式之一（见图 9-4）以配合您的驱动设备和移动的方位：

- 位控模块从 P0 发出反转脉冲，从 P1 发出正转脉冲。
- 位控模块从 P0 发脉冲的同时，如果 P1 接通反转，如果 P1 断开正转。

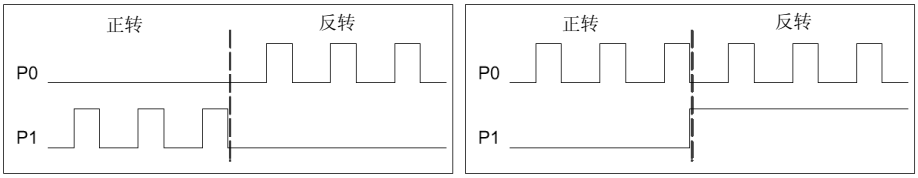


图 9-4 负极性的转向选项

组态模块对物理输入的响应

您必须指定位控模块对每个 LMT+开关、LMT-开关以及 STP 输入的响应：无动作（忽略输入条件）、减速至停止（缺省）或立即停止。

警告：

控制设备可能在不安全条件下出现故障，并可能导致被控设备不可预知的操作。这些不可预知的操作可能导致人员的伤亡以及/或设备的损坏。

位控模块的限位和停止功能是电逻辑实现的，不能够提供机电控制所能提供的保护等级。请考虑使用独立于 S7-200 CPU 和位控模块的急停功能，机电互锁或冗余的机电保护。

输入最大速度和启动/停止速度

您必须为您的应用定义最大速度和启动/停止速度：

- MAX_SPEED：该数值是您的应用中操作速度的最大值，它应在电机力矩能力的范围内。驱动负载所需的力矩由摩擦力、惯性以及加速/减速时间决定。位控向导根据指定的 MAX_SPEED，计算并显示位控模块所能控制的最小速度。
- SS_SPEED：输入该数值满足您的电机在低速时驱动负载的能力，如果 SS_SPEED 的数值过低，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 的数值过高，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过载。

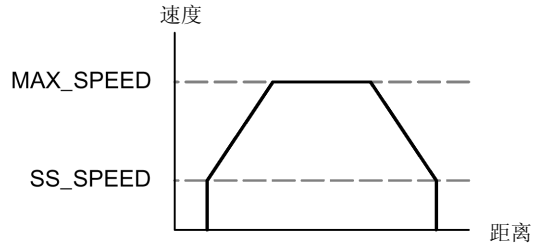


图 9-5　最大速度和启动/停止速度

在电机的数据单中，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%。SS_SPEED 值必须大于根据您指定的 MAX_SPEED 值而显示出来的最小速度。

请参考电机的数据单，为您的应用选择正确的速度。

图 9-6 所示为典型的电机力矩/速度曲线。

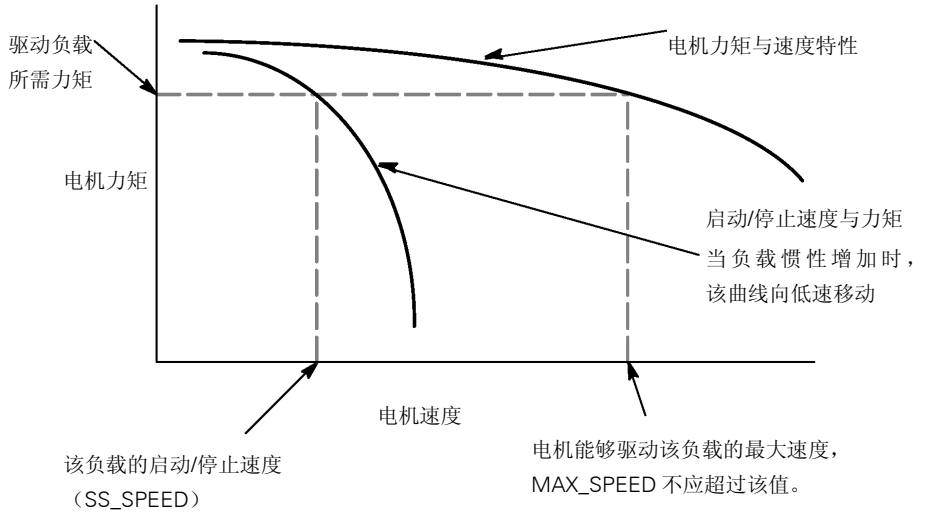


图 9-6 典型电机力矩—速度曲线

输入点动参数

点动命令用于将工件以手动方式移动到指定位置。使用位控向导，可以指定以下拖动参数值：

- JOG_SPEED：JOG_SPEED（电机的拖动速度）是点动命令有效时能够得到的最大速度。
- JOG_INCREMENT：是瞬间的点动命令能够将工件移动的距离。

图 9-7 所示为点动命令的操作。当位控模块收到一个点动命令后，它启动一个定时器。如果点动命令在 0.5 秒到时之前结束，位控模块则以定义的 SS_SPEED 速度将工件移动 JOG_INCREMENT 数值指定的距离。当 0.5 秒到时时，点动命令仍然是激活的，位控模块加速至 JOG_SPEED 速度。继续移动直至点动命令结束。位控模块随后减速停止。您可以在 EM253 控制面板中使能点动命令，或者使用位控指令。

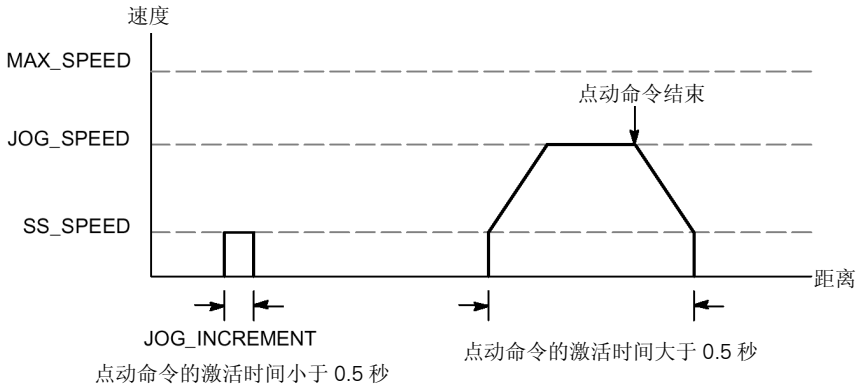


图 9-7 点动操作的说明

输入加速和减速时间

作为位控模块组态的一部分，您必须设置加速和减速时间。加速时间和减速时间的缺省设置都是 1 秒。通常，电机可在小于 1 秒的时间内工作。您要以毫秒为单位进行时间设定：

- ACCEL_TIME：电机从 SS_SPEED 速度加速到 MAX_SPEED 速度所需的时间。
  缺省值=1000ms
- DECEL_TIME：电机从 MAX_SPEED 速度减速到 SS_SPEED 速度所需要的时间。
  缺省值=1000ms。

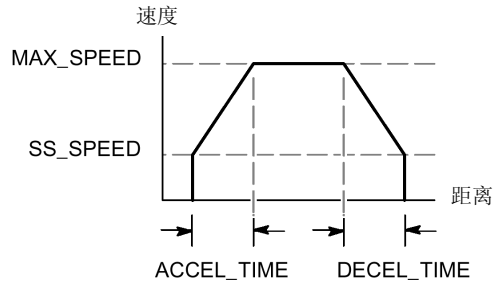


图 9-8 加速和减速时间

提示：

电机的加速和减速时间要经过测试来确定。开始时，您应在位控向导中输入一个较大的值。在作应用测试时，你可以使用 EM253 控制面板按照需要调整该数值。逐渐减少这个时间值直至电机开始停止，从而优化您应用中的这些设置。

输入急停时间

急停时间通过减小移动包络中加速和减速部分的急停（改变率）来平滑移动控制。见图 9-9。减少急停能够改善位置追踪的性能。急停时间也被称为“S 曲线包络”。急停只能用于简单的单步包络。这种补偿同样地作用于加速曲线和减速曲线的开始和结束部分。急停补偿不能够应用在介于零速和 SS_SPEED 速度之间的初始段和结束段中。

您可以输入一个时间值（JERK_TIME）来指定急停补偿。这一时间是加速度的改变从零到最大所需要的时间，由参数 MAX_SPEED、SS_SPEED 和 ACCEL_ TIME 或与之相应的 DECEL_TIME 来定义，与只是简单地增加 ACCEL_TIME 和 DECEL_TIME 相比，一个较长的急停时间由于能够对整个循环时间只有一个较小的增加，从而可以产生平滑的操作。零值代表没有补偿。

（缺省=0ms）

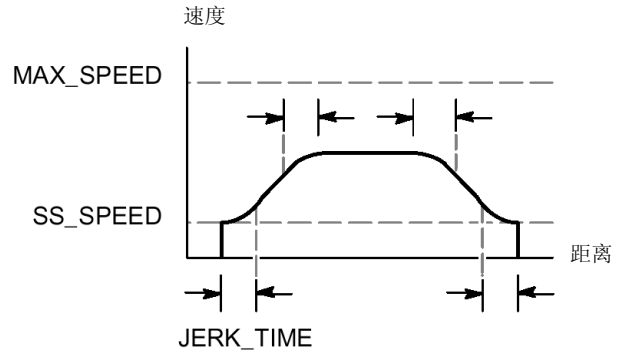


图 9-9 急停补偿

提示：

对于 JERK_TIME 来说，一个好的初始值是 ACCEL_TIME 的 40%。

组态参考点和参考点寻找参数

如果您的应用指定从一个绝对位置开始移动，您必须指定物理系统中的一个已知点作为参考点（RP），该点要与位置测量系统相吻和。参考点的缺省设置是零位置。

您可以通过组态参考点寻找（RP 寻找）参数来控制您的移动应用如何寻找 RP。RP 可以位于 RPS 有效区的中央，也可以位于 RPS 有效区的两边，或者 RP 可以位于从 RPS 有效区边缘开始的若干数量的零脉冲（ZP）之后的位置。位控模块提供一个外部参考点开关（RPS）传感器，用于寻找 RP。组态 RP，您要输入以下信息：

- 为电机指定寻找 RP 的速度：
  - RP_FAST 是模块执行 RP 寻找命令的最初的速度。通常 RP_FAST 是 MAX_SPEED 的 2/3 左右。
  - RP_SLOW 是接近 RP 的最终的速度。通常使用一个较慢的速度去接近 RP 以免错过。RP_SLOW 的典型值为 SS_SPEED。
- 为 RP 的寻找指定初始寻找方向（RP_SEEK_DIR）和最终接近方向（RP_APPR_DIR）。分为正向和反向。
  - RP_SEEK_DIR 是 RP 寻找操作最初的方向。通常，这个方向是从工作区到 RP 附近。限位开关在确定 RP 的寻找区域时扮演重要角色。当执行 RP 寻找操作时，遇到限位开关会引起方向反转，使寻找能够继续下去。（缺省=反向）。
  - RP_APPR_DIR 是最终接近 RP 的方向。为减少反冲击和提供更高的准确性，RP_APPR_DIR 按照正常工作循环中的方向移动（缺省=正向）。

位控向导提供高级参考点选项，可以指定一个 RP 偏移量（RP_OFFSET），这个偏移量是指从 RP 到零位置的距离，见图 9-10。以 RPS 为参考确定一个准确的位置作为 RP。组态 RP 偏移量，您要输入以下数值：

- RP_OFFSET：在物理的测量系统中 RP 到零位置之间的距离。（缺省=0）

- 螺距误差补偿：当方向发生变化时，为消除系统中的滞慢（螺距误差），电机必须移动的距离。螺距误差补偿总是正值。（缺省=0）

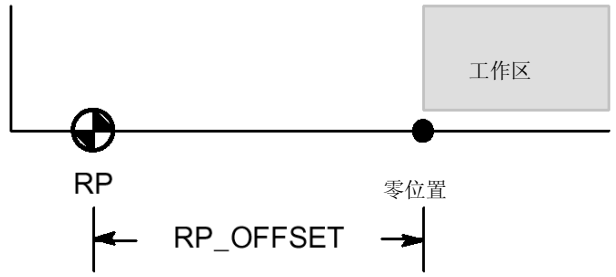


图 9-10 RP 与零位置的关系

组态 RP 寻找顺序

您可以为位控模块组态参考点寻找的顺序。图 9-11 所示为一个简化了的缺省的 RP 寻找顺序图。您可以为 RP 搜寻顺序作以下选择：

- RP 寻找模式 0：不执行 RP 寻找顺序
- RP 寻找模式 1：RP 位于 RPS 输入有效区接近工作区的一边开始有效的位置上。
- RP 寻找模式 2：RP 位于 RPS 输入有效区的中央。
- RP 寻找模式 3：RP 位于 RPS 输入有效区之外。RP_Z_CNT 指定了在 RPS 失效之后应接收多少个 ZP（零脉冲）输入。
- RP 寻找模式 4：RP 通常位于 RPS 输入的有效区内。RP_Z_CNT 指定在 RPS 激活后应接收多少个 ZP（零脉冲）输入。

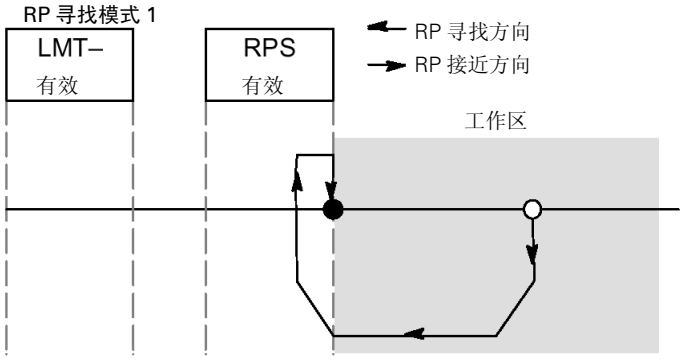


图 9-11 缺省的 RP 搜寻顺序（简化的）

更多的关于位控模块不同的 RP 寻找顺序的信息，请参见图 9-14 至图 9-17。

组态位控模块的移动包络

一个包络是一个预先定义的移动描述，它包括一个或多个速度，影响着运动从起点到终点。即使不定义包络也可以使用模块，位控向导为您提供了一个指令子程序（POSx_GOTO），可用于移动控制。

- 包络数：您最多可选择 25 个包络
- 命令字节的地址：您必须为位控模块输入一个输出（Q）区地址作为命令字节。参考图 4-10 中对 I/O 地址的说明。
- 组态/包络表的地址：您必须为组态/包络表输入起始的存储区地址，以存储位控模块的组态数据以及所有包络的数据。位控模块的组态数据需要 92 个字节的 V 存储区，每一个包络需要 34 个字节的 V 存储区。例如，带有一个包络的位控模块其组态/包络表所需占用的 V 区数量是 126 字节。

位控向导能够向您建议一个大小合适的未经使用的 V 存储区地址。

定义移动包络

位控向导提供移动包络定义，在这里，您可以为您的应用定义每一个移动包络。对每一个包络，您可以选择操作模式并为每个包络的各步定义指标。位控向导中可以为每个包络定义一个符号名，其作法是在您定义包络时为其输入一个符号名即可。当您完成包络的组态后，可以存储组态并将参数打印出来。

选择包络的操作模式

您要按照操作模式对包络进行组态，是绝对位置还是相对位置，是单一速度的连续转动还是以两种速度连续转动。图 9-12 所示为不同的操作模式。

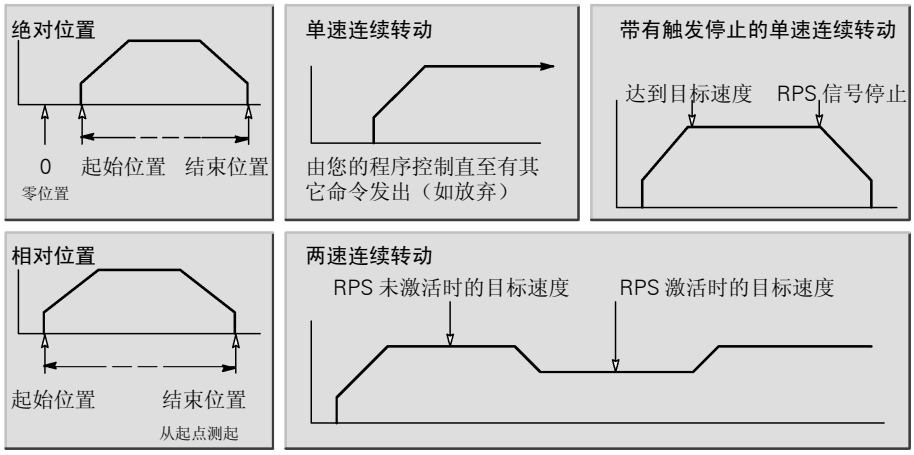


图 9-12 位控模块的模式选择

创建包络中的步

一个步是工件移动的一个固定距离，包括加速和减速时间内的距离。每个包络最多可有 4 个步。

您要为每个步指定目标速度和结束位置。如果不止一步，只需点击新一步标签（New Step）并输入包络中每一步的信息。图 9-13 所示为四种可能的包络；当然还有其它可能的组合。

点击步骤画图标签（Plot Step），您能够看到由位控向导计算，以图形方式表达出来的步。这样，您就能够很容易地随时浏览并编辑每一个步。

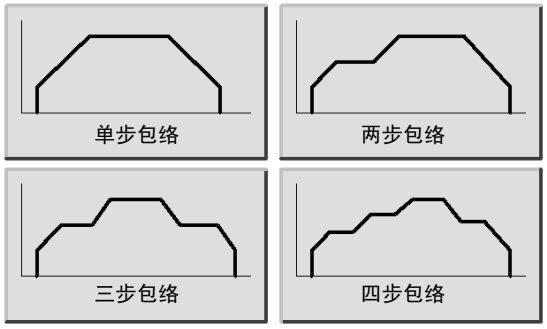


图 9-13 移动包络示例

完成对位控模块的组态

当您完成对位控模块的组态时，只需点击完成（Finish），然后位控向导会执行以下任务：

- 将模块的组态和包络表插入到你的 S7-200 程序的数据块中。
- 为位控参数生成一个全局符号表
- 在项目的程序块中增加位控指令子程序，您可在应用中使用这些指令

要修改任何组态或包络信息，您可以再次运行位控向导。

提示：

由于位控向导修改了程序块、数据块和系统块，要确保这三种块都下载到 S7-200 CPU 中。否则，位控模块可能会无法得到操作所需的所有程序组件。

理解位控模块所支持的 RP 寻找模式

下列各图是每种 RP 寻找模式的不同选项的示意图

- 图 9-14 所示是 RP 寻找模式 1 的选项。这种模式将 RP 定位在 RPS 输入开始激活，并且靠近工作区的这端。
- 图 9-15 所示是 RP 寻找模式 2 的选项。这种模式将 RP 定位在 RPS 输入有效区域的中央。
- 图 9-16 所示是 RP 寻找模式 3 的选项。这种模式将 RP 定位在 RPS 输入有效区域之外一个指定数量的零脉冲（ZP）处。

- 图 9-17 所示是 RP 寻找模式 4 的选项。这种模式将 RP 定位在 RPS 输入有效区域内一个指定数量的零脉冲（ZP）处。

对于每一种模式都有 RP 寻找方向和 RP 接近方向的四种组合。这些组合决定了 RP 寻找操作的模式。对每一种组合，又有四种不同的起始点：

1. RP 寻找操作从工作区内开始
2. RP 寻找操作从 RPS 有效区内开始
3. RP 寻找操作从限位开关区域内开始（LIM+或 LIM-）
4. RP 寻找操作从其它的一些区域开始

每个图中的工作区域都已确定，以便使从参考点到工作区的移动与 RP 接近方向一样。以这种方式选择工作区域，机械齿轮系统中的所有螺距误差都能够在参考点找到后，向工作区的第一次移动中去除。

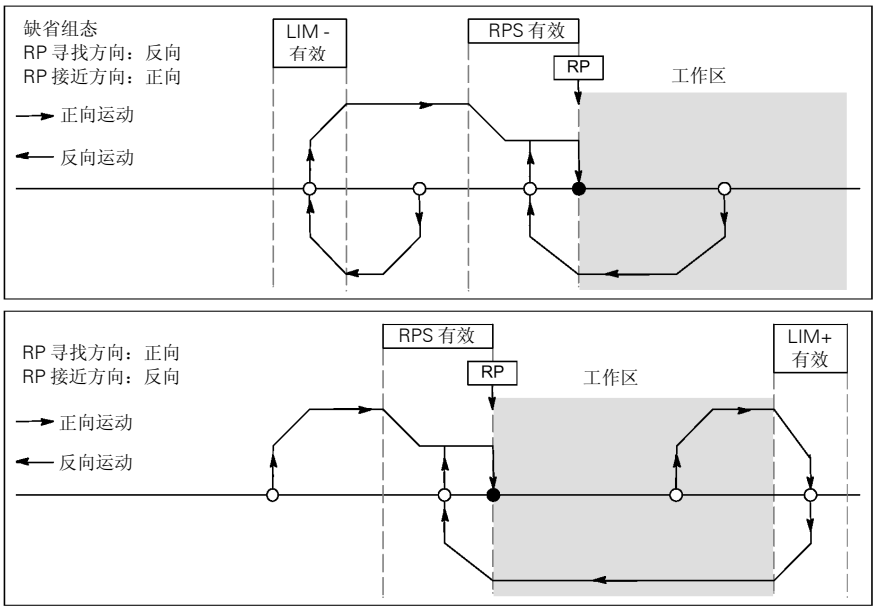


图 9-14 RP 寻找模式 1

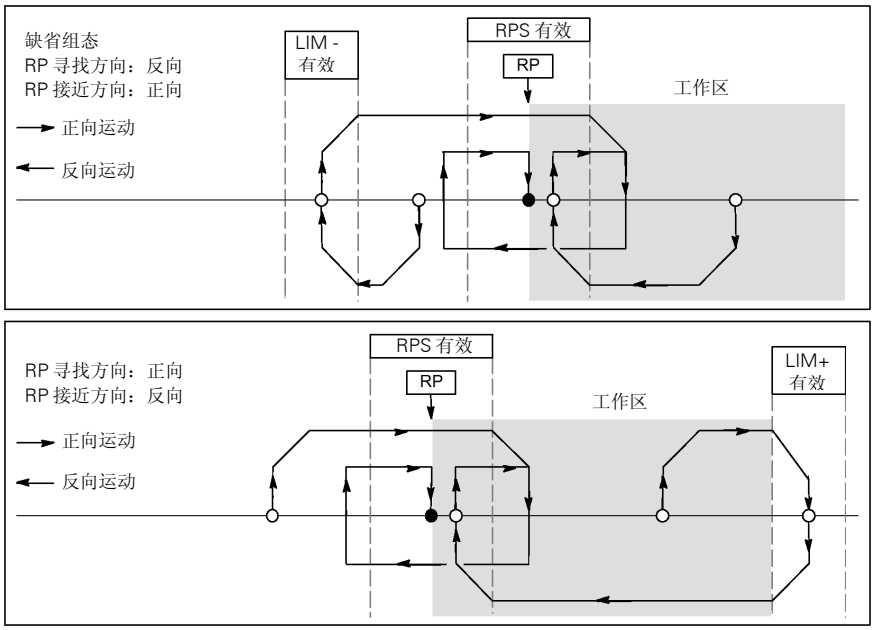


图 9-15 RP 寻找模式 2

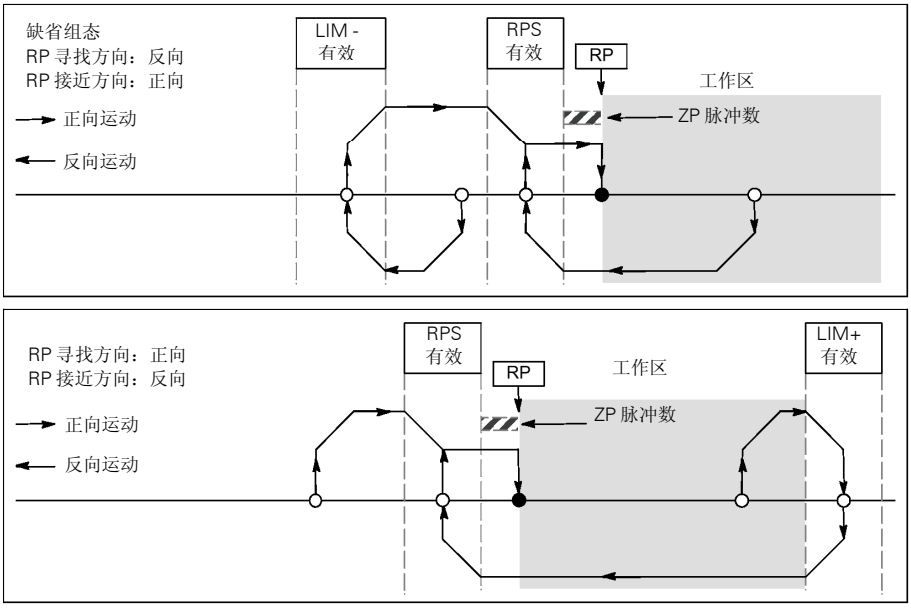


图 9-16 RP 寻找模式 3

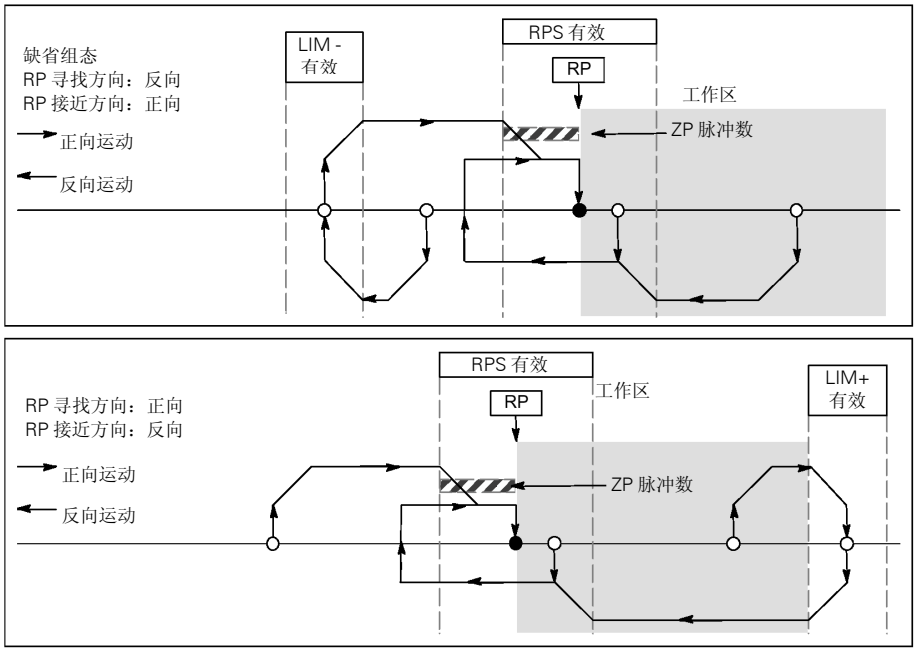


图 9-17 RP 寻找模式 4

选择工作区位置消除螺距误差

图 9-18 所示为工作区与参考点（RP）、RPS 有效区以及限位开关（LIM+和 LIM-）之间在接近方向上能够消除螺距误差的关系。图中的第二部分工作区的位置不能够消除螺距误差。图 9-18 所示为 RP 寻找模式 3。其它的 RP 寻找模式的每个搜寻顺序也可以有类似的工作区位置，但不推荐。

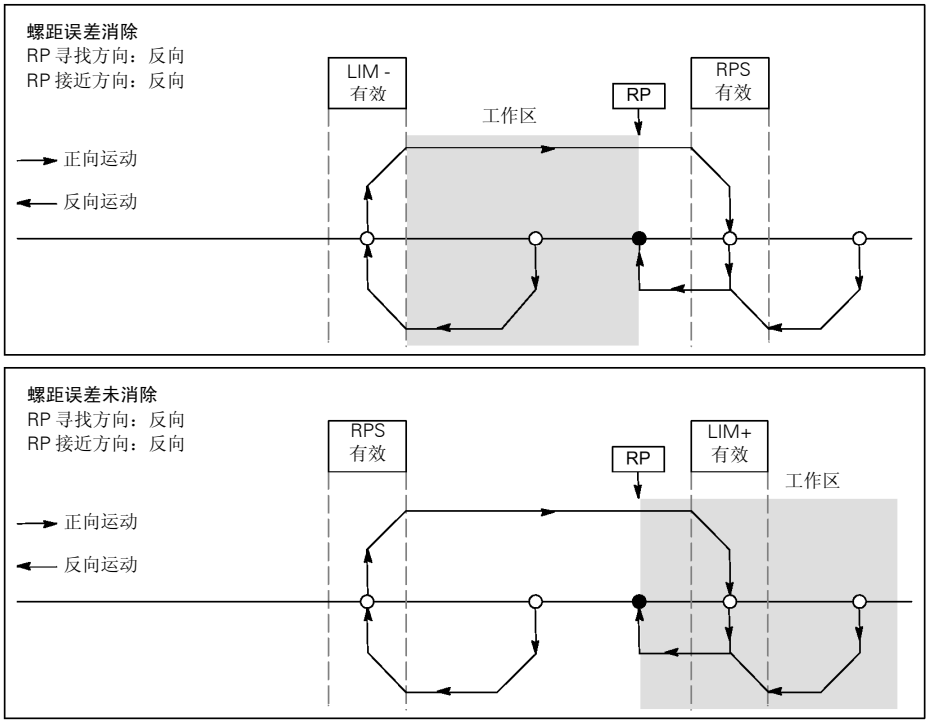


图 9-18 消除螺距误差和未消除螺距误差的工作区位置

由位控向导生成的位控模块

位控向导能够根据位控模块的位置和您对模块所作的组态生成唯一的指令子程序，从而使位控模块的控制变得非常容易。每条位控指令都有一个前缀“POSx_”这里 X 是模块位置。由于每个位控指令是一个子程序，11 条位控指令使用 11 个子程序。

提示：

位控指令使用户程序对存储空间的需求最多增加至 1700 字节。您可以删除不用的位控指令以减小对存储空间的需求。要恢复已删除的位控指令，只需再次运行位控向导即可。

位控指令使用指南

您必须确保在同时只有一条位控指令是激活的。

您可以在一个中断程序中执行 POSx_RUN 和 POSx_GOTO。但是，当模块正忙于处理其它命令时，千万不要试图在中断程序中启动指令。如果你在一个中断程序中启动一条指令，你可以使用 POSx_CTRL 指令的输出来监控位控模块是何时完成运动的。

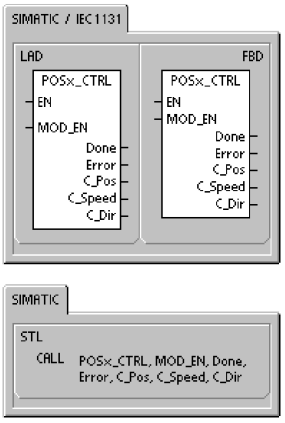
位控向导按照您所选的测量系统自动组态速度参数(Speed 和 C_Speed)和位置参数(Pos 或 C_Pos）的数值。对于脉冲，这些参数是双整数，对于工程单位，这些参数是您所选的单位的实数值。例如：选厘米（cm）为单位，位控参数则存为以厘米为单位的一个实数值，速度参数则选一个每秒若干厘米的实数值（cm/sec.）。

以下是特定的运动控制任务所需的位控指令：

- 在您的用户程序中插入 POSx_CTRL，并以 SM0.0 为条件使之每个循环都执行。
- 要指定运动到一个绝对位置，您必须首先使用 POSx_RSEEK 或 POSx_LDPOS 指令建立零位置。
- 要移动到某个特定位置，根据您程序中的输入，使用 POSx_GOTO 指令。
- 要运行您在位控向导中所组态的运动包络，使用 POSx_RUN 指令。

其它位控指令是可选的。

POSx_CTRL 指令



POSx_CTRL 指令在 S7-200 每次转换为 RUN 模式时自动向位控模块发出命令，装载组态/包络表，从而实现对位控模块的使能和初始化。

这条指令在您的项目中只使用一次，并且要确保您的用户程序在每一循环中调用该指令。使用 SM0.0（常通）作为 EN 参数的输入。

EN 参数必须为接通状态以确保其它位控指令发送命令给位控模块。如果 EN 参数为断开状态。位控模块放弃所有正在进行当中的模块。

POSx_CTRL 指令的输出参数提供位控模块当前的状态。

当位控模块完成所有指令后，参数 Done 接通。

参数 Error 包含指令的执行结果。错误代码的定义请参见表 9-13。

参数 C-Pos 是模块的当前位置。基于测量的单位，该值可以是一个脉冲数（双整数）或者工程单位数（实数）。

参数 C-Speed 提供模块的当前速度。如果您组态模块的测量系统是脉冲，C-Speed 是一个每秒脉冲数的长整数。如果您组态测量系统工程单位。C_Speed 是一个每秒若干个所选工程单位数的实数。

参数 C-Dir 指示电机的当前方向。

表 9-2 POSx_CTRL 指令的参数。

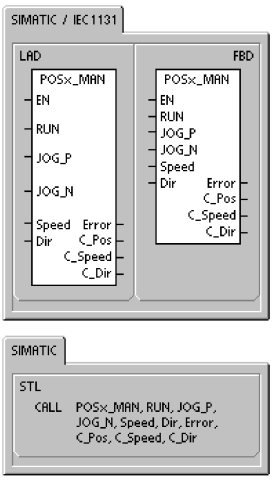
输入/输出	数据类型	操作数
MOD_EN	BOOL	I、Q、V、M、SM、S、T、C、L、能流
Done、C_Dir	BOOL	I、Q、V、M、SM、S、T、C、L
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD
C_Pos、C_Speed	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD

提示：

位控模块只在上电时或接到装载组态的命令时读取组态/包络表。

- 当使用位控向导修改组态时，POSx_CTRL 指令自动命令位控模块在 S7-200 CPU 转为 RUN 模式时装载组态/包络表。
- 如果使用 EM253 控制面板修改组态，点击刷新组态按钮命令位控模块装载新的组态/包络表。
- 如果您用其它方式修改了组态，那么您必须向位控模块发出一条重新装载组态的命令使它装载组态/包络表。否则，位控模块继续使用旧的组态/包络表。

POSx_MAN 指令



POSx_MAN 指令（手动模式）将位控模块置于手动模式。这种模式下，电机可以以不同的速度沿正向或反向点动。当 POSx_MAN 指令使能时，只能运行 POSx_CTRL 和 POSx_DIS 指令。

RUN、JOG_P 或 JOG_N 的输入你只能同时使能一个。

使能 RUN（RUN/Stop）参数则命令位控模块按指定方向（参数 Dir）加速到指定速度（参数 Speed）。你可以在电机运行时改变速度值，但参数 Dir 必须保持恒定。禁止参数 RUN 则命令位控模块减速至电机停止。

使能参数 JOG_P（点动正转）或 JOG_N（点动反转）命令位控模块沿正向或反向点动。

如果 JOG_P 或 JOG_N 有效的时间短于 0.5 秒，位控模块则输出脉冲运动到 JOG_ INCREMENT 所指定的距离。如果 JOG_P 或 JOG_N 的有效时间等于或长于 0.5 秒，位控模块则开始加速到 JOG_SPEED 所指定的速度。

参数 Speed 决定 RUN 使能时的速度。如果位控模块的测量系统组态为脉冲，该速度是一个每秒若干脉冲数的数值（双整数）。若位控模块的测量系统组态为工程单位，该速度是一个每秒若干单位的实数值。电机运行时可以修改该速度参数。

参数 Dir 决定 RUN 使能时的运动方向。当 RUN 使能时，不能修改该方向参数。

参数 Error 包含指令的执行结果。错误代码的定义请参见表 9-13。

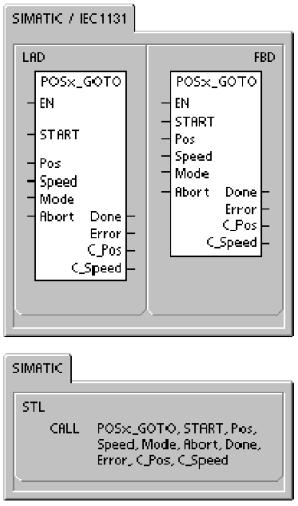
参数 C-POS 包含模块的当前位置。基于所选的测量系统，该值可以是每秒脉冲数（双整数）或每秒工程单位数（实数）。

参数 C_Dir 指示电机的当前方向。

表 9-3 POSx_MAN 指令的参数

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

POSx_GOTO 指令



指令 POSx_GOTO 命令位控模块走到指定位置。

接通 EN 位使能该指令。确保 EN 位始终保持接通直到 Done 位指示指令完成。

接通参数 START 向位控模块发送一个 GOTO 命令。当参数 START 接通且位控模块不忙时，每一循环都会向位控模块发送一条 GOTO 命令。要确保只发送一条 GOTO 命令，使用边沿检测来触发 START 参数。

参数 POS 包含一个表示运动位置（对于绝对运动）或运动距离（对于相对运动）的值。基于所选的测量系统，该值可以是一个脉冲数（双整数）或工程单位数（实数）。

参数 Speed 决定了运动的最大速度。基于测量单位，该值可以是每秒脉冲数（双整数）或每秒工程单位数（实数）。

参数 Mode 可选择运动类型：

0—绝对位置

1—相对位置

2—单速连续正向转动

3—单速连续负向转动

当位控模块完成该指令时，参数 Done 接通。

参数 Error 包含指令的执行结果。错误代码的定义请参见表 9-13。

参数 C_POS 包含模块的当前位置。基于测量单位，该值可以是一个脉冲数（双整数）或是工程单位数（实数）。

参数 C_Speed 包含模块的当前速度。基于测量系统，该值可以是一个每秒脉冲数（双整数）或是每秒工程单位（实数）。

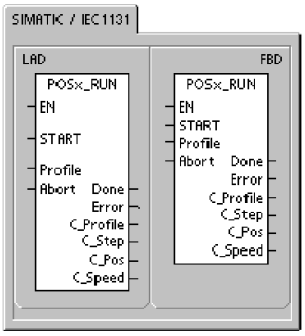
表 9-4 POSx_GOTO 指令的参数

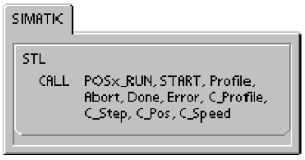
输入/输出	数据类型	操作数
START	BOOL	I、Q、V、M、SM、S、T、C、L、能流
Pos、Speed	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD、常数
Mode	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD、常数
Abort、Done	BOOL	I、Q、V、M、SM、S、T、C、L
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD
C_Pos、C_Speed	DINT、REAL	ID、QD、VD、MD、SMD、SD、LD、AC、*VD、*AC、*LD

POSx_RUN 指令

POSx_RUN 指令（运行包络）命令位控模块执行存储在组态/包络表中的某个包络的运动操作。

接通 EN 位可使能该指令。确保 EN 位保持接通直至 Done 位指示该指令完成。





接通参数 START 发送一个 RUN 命令给位控模块。每一循环周期，只要 START 参数接通且位控模块不忙，该指令发送一个 RUN 命令给位控模块，要保证该命令只发一次，使用边沿检测指令以脉冲触发 START 参数接通。

包络参数包含该运动包络的号码或符号名。你也可以选择高级运动命令。有关运动命令的信息，请参见表 9-19。

接通参数 Abort，命令位控模块停止当前的包络并减速直至电机停下。

参数 Error 包含指令的执行结果。错误代码的定义请参见表 9-13。

参数 C_Profile 包含位控模块当前正在执行的包络。

参数 C_Step 包含当前正在执行的包络的步。

参数 C_Pos 包含模块的当前位置。基于测量单位，该值可以是一个脉冲数（双整数）或工程单位数（实数）。

参数 C_Speed 包含模块的当前速度。基于测量单位，该值可以是一个每秒脉冲数（双整数）或一个每秒工程单位数（实数）。

表 9-5 POSx_RUN 指令的参数

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、常数
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

POSx_RSEEK 指令

POSx_RSEEK 指令（寻找参考点位置）触发一个参考点寻找操作，使用组态/包络表中的搜寻方式。当位控模块锁定参考点并且运动停止后，位控模块装载参数 RP_OFFSET 的值作为当前位置。

RP_OFFSET 的缺省值是 0。使用位控向导 EM253 控制面板或 POSx_LDOFF（装载偏移量）指令可以改变 RP_OFFSET 值。

接通 EN 位可使能该指令。确保 EN 位保持接通直至 Done 位指示该指令完成。

接通参数 START 则向位控模块发送一条 RSEEK 命令。

每一循环周期，当参数 START 接通且模块不忙，该指令向位控模块发送一条 RSEEK 指令，要确保该指令只发送一次，使用边沿检测以脉冲触发参数 START 接通。

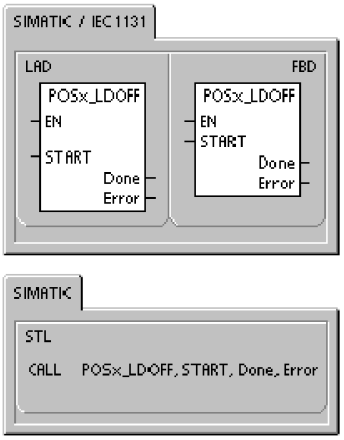
模块完成该指令时，参数 Done 接通。

参数 Error 包含该指令的结果，错误代码的定义请参见表 9-13。

表 9-6 POSx_RSEEK 指令的参数

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

POSx_LDOFF 指令



POSx_LDOFF 指令（装载参考点偏移量）建立一个新的零位置，它与参考点位置不在同一处。

执行这条指令之前，必须首先决定参考点位置，还要把机器移动到起始位置，当该指令发送 LDOFF 命令时，位控模块计算起始位置（当前位置）与参考点之间的偏移量。然后，模块将计算所得的偏移量存为参数 RP_OFFSET 的值并将当前位置设为 0。这样就将起始位置设置为零位置。

如果出现故障，电机找不到它的位置了（如，掉电或电机被手动重新定位），可以使用 POSx_RSEEK 指令自动地重建零位置。

接通 EN 位可使能该指令。确保 EN 位保持接通直至 Done 位指示该指令完成。

接通参数 START 则向位控模块发送一条 LDOFF 命令。每一循环周期，只要参数 START 接通且位控模块不忙，该指令向位控模块发送一条 LDOFF 命令。要确保该命令只发送一次，使用边沿检测以脉冲触发参数 START 接通。

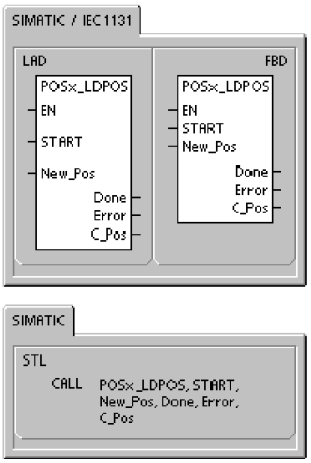
模块完成该指令时，参数 Done 接通。

参数 Error 包含该指令的结果。错误代码的定义请参见表 9-13。

表 9-7 POSx_LDOFF 指令的参数

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

POSx_LDPOS 指令



POSx_LDPOS 指令（装载位置）改变位控模块的当前位置值。你也可以使用这条指令为绝对运动命令建立一个新的零位置。

接通 EN 位可使能该指令。确保 EN 位接通直至 Done 位指示该指令完成。

接通参数 START 则向位控模块发送一条 LDPOS 命令。每一循环周期，参数 START 接通且位控模块不忙，该指令向位控模块发送一条 LDPOS 命令。要确保该命令只发送一次，使用边沿检测以脉冲触发参数 START 接通。

参数 New_Pos 提供一个新值替换位控模块在绝对运动中报告并使用的当前位置值。基于测量单位，该值可以是一个脉冲数（双整数）或是工程单位数（实数）。

模块完成该指令时，参数 Done 接通。

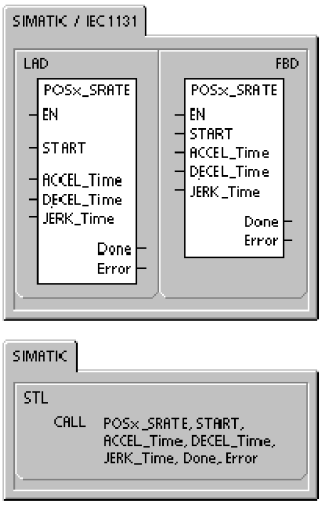
参数 Error 包含该指令的执行结果。错误代码的定义请参见表 9-13。

参数 C_Pos 包含模块的当前位置。基于测量单位，该值可以是一个脉冲数（双整数）或是工程单位数（实数）。

表 9-8 POSx_LDPOS 指令的参数

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

POSx_SRATE 指令



POSx_SRATE 指令（设置速率）命令位控模块改变加速、减速和急停时间。

接通 EN 位使能该指令。确保 EN 位接通直至 Done 位指示该指令完成。

接通参数 START 则将新的时间值拷贝到组态/包络表并向位控模块发送一条 SRATE 命令。每一循环周期，当 START 参数接通并且模块不忙时，该指令发送一条 SRATE 命令到位控模块。要确保该命令只发送一次，使用边沿检测以脉冲触发参数 START 接通。

参数 ACCEL-Time、DECEL_Time 和 JERK_Time 决定新的加速时间、减速时间和急停时间，单位为毫秒（ms）。

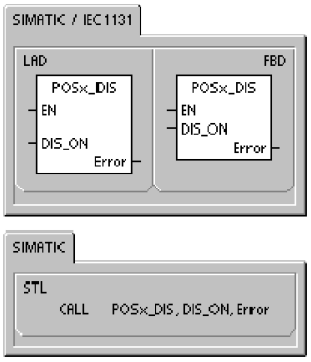
模块完成该指令时，参数 Done 接通。

参数 Error 包含指令的结果。错误代码的定义请参见表 9-13。

表 9-9 POSx_SRATE 指令的参数

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

POSx_DIS 指令



指令 POSx_DIS 可接通或断开位控模块的 DIS 输出。您可以使用 DIS 输出来使能或禁止电机控制器。如果您要使用位控模块上的 DIS 输出，那么，这条指令可以在每一循环周期中调用，或者只在您需要改变 DIS 输出时调用。

EN 位接通时使能该指令，参数 DIS_ON 控制位控模块的 DIS 输出。关于 DIS 输出的更多的信息，请参见表 9-1 或参考位控模块附录 A 中的规范。

参数 Error 包含该指令的结果。错误代码的定义请参见表 9-13。

表 9-10 POSx_DIS 指令的参数

输入/输出	数据类型	操作数
DIS_ON	BYTE	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

POSx_CLR 指令

POSx_CLR 指令（触发 CLR 输出）命令位控模块在 CLR 输出上生成一个 50ms 的脉冲。

接通 EN 位使能该指令。确保 EN 保持接通直至 Done 位指示该指令完成。

接通参数 START 则向位控模块发送一条 CLR 命令。每一循环周期，当参数 START 接通并且模块不忙时，该指令向位控模块发送一条 CLR 命令。要确保该命令只发送一次，使用边沿检测以脉冲触发参数 START 接通。

模块完成该指令时，参数 Done 接通。

参数 Error 包含该指令的执行结果。错误代码的定义请参见表 9-13。

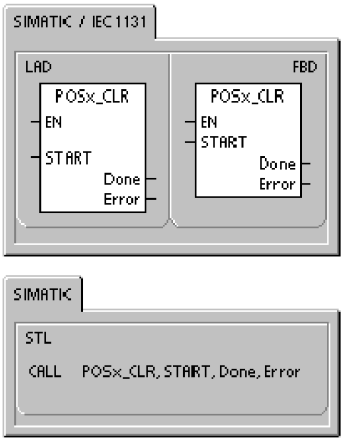
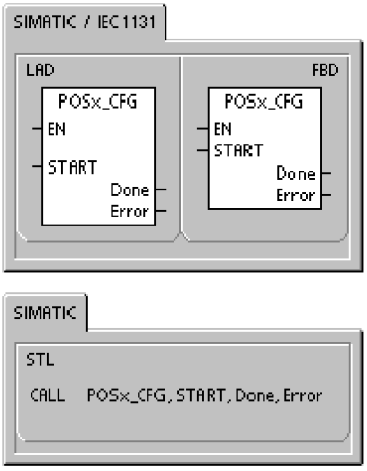


表 9-11 POSx_CLR 指令的参数

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

POSx_CFG 指令



POSx_CFG 指令（重新装载组态）命令位控模块从组态/包络表指针所指定的地方读取组态块。位控模块将新的组态与现有的组态进行比较并执行所有需要的设置改变或重新计算。

接通 EN 位可使能该指令。确保 EN 位保持接通直至 Done 位指示该指令完成。

接通参数 START 使位控模块发送一条 CFG 命令。每一循环周期，当参数 START 接通且模块不忙时，该指令都会向位控模块发送一条 CFG 命令。要确保该命令只发送一次，使用边沿检测以脉冲触发参数 START 接通。

模块完成该指令时，参数 Done 接通。

参数 Error 包含该指令的执行结果。错误代码的定义请参见表 9-13。

表 9-12 POSx_CFG 指令的参数

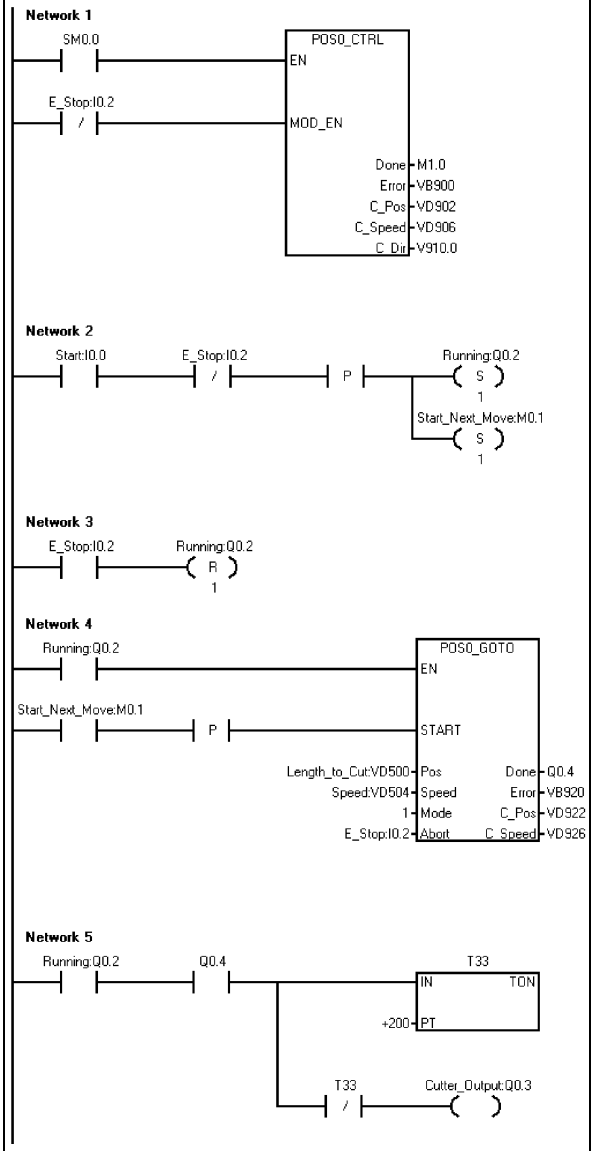
输入/输出	数据类型	操作数
START	BOOL	I、Q、V、M、SM、S、T、C、L、能流
Done	BOOL	I、Q、V、M、SM、S、T、C、L
Error	BYTE	IB、QB、VB、MB、SMB、SB、LB、AC、*VD、*AC、*LD

位控模块的示例程序

第一个示例程序是一个相对运动的示例，使用了 POSx_CTRL 和 POSx_GOTO 指令，完成一个切割长度的操作。该程序不需要 RP 寻找模式或运动包络，长度可以是脉冲数或工程单位。输入长度（VD500）和目标速度（VD504），当 I0.0（Start）接通时，设备启动。设备完成当前操作则停止。当 I0.2（E_Stop）接通时，设备放弃所有运动并立即停止。

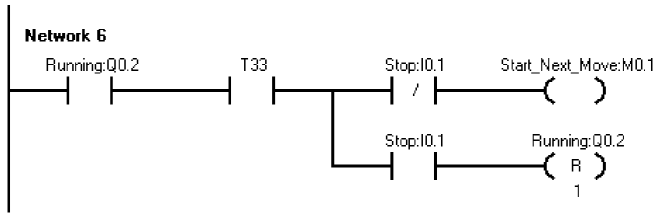
第二个示例程序提供了一个使用 POSx_CTRL、POSx_RUN、POSx_RSEEK 和 POSx_MAN 指令的示例。您必须组态 RP 寻找模式和一个运动包络。

示例程序 1：简单相对运动（长度切割应用）



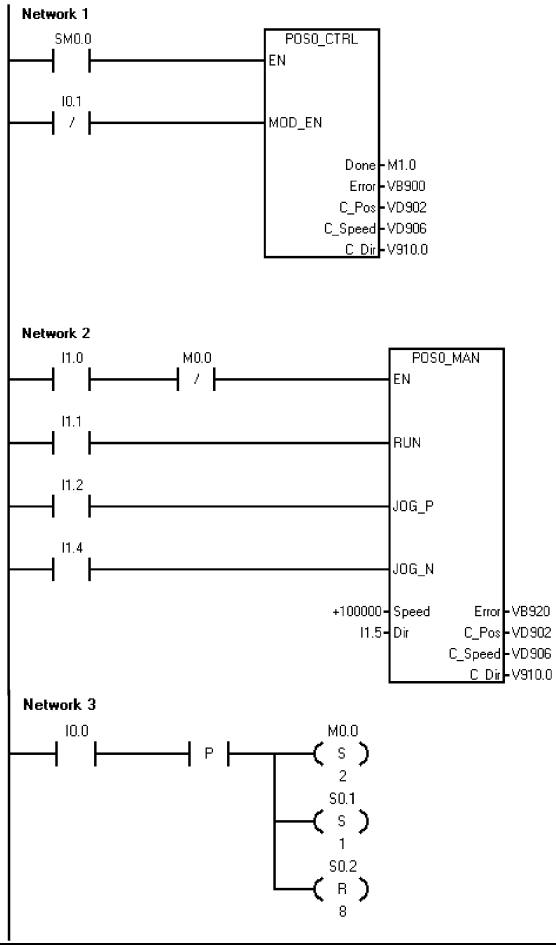

```
Network 1          //控制指令（模块位于0号槽）
LD      SM0.0
=       L60.0
LDN     I0.2
=       L63.7
LD      L60.0
CALL    POS0_CTRL, L63.7, M1.0, VB900,
        VD902, VD906, V910.0

Network 2          //Start使设备进入自动模式
LD      I0.0
AN      I0.2
EU
S       Q0.2, 1
S       M0.1, 1

Network 3          //E_Stop: 立即停止并
                   //关断自动模式
LD      I0.2
R       Q0,2, 1

Network 4          //运动到一个指定点。
                   //输入切割长度。
                   //将目标速度送给 Speed。
                   //将模式设为 1 (相对模式)。
LD      Q0.2
=       L60.0
LD      M0.1
EU
=       L63.7
LD      L60.0
CALL    POS0_GOTO, L63.7, VD500, VD504,
        1, I0.2, Q0.4, VB920, VD922, VD926

Network 5          //到达位置时，接通切割机
                   //2秒，完成切割。
LD      Q0.2
A       Q0.4
TON     T33, +200
AN      T33
=       Q0.3
```

示例程序 1：简单相对运动（长度切割应用）（续）



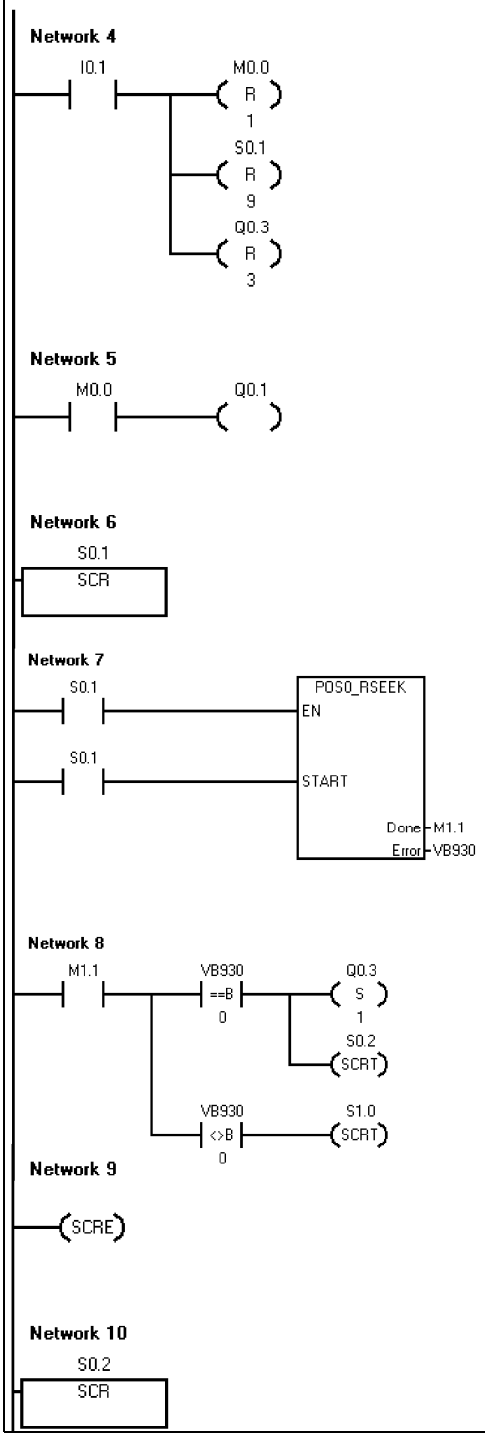

```
Network 6        //切割完成后重新启动
                 //除非Stop激活。
LD       Q0.2
A        T33
LPS
AN       I0.1
=        M0.1
LPP
A        I0.1
R        Q0.2, 1
```

示例程序 2：使用 POSx_CTRL、POSx_RUN、POSx_SEEK 和 POSx_MAN 编程



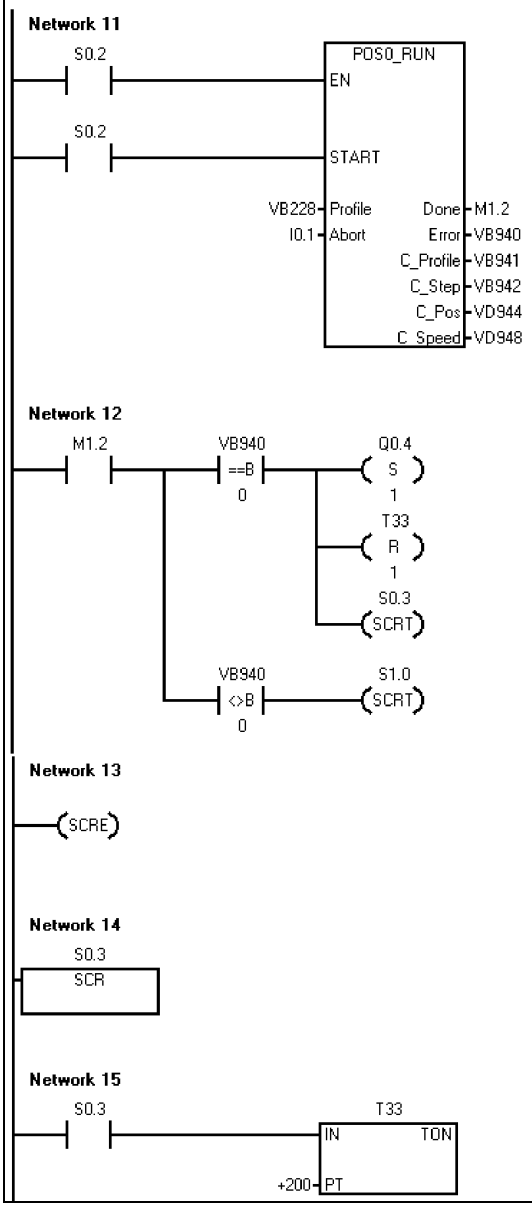

```
Network 1        //使能位控模块。
LD       SM0.0
=        L60.0
LDN      I0.1
=        L63.7
LD       L60.0
CALL     POS0_CTRL, L63.7, M1.0, VB900,
         VD902,     VD906, V910.0

Network 2        //如果不在自动模式则使能手动
                 //模式。
LD       I1.0
AN       M0.0
=        L60.0
LD       I1.1
=        L63.7
LD       I1.2
=        L63.6
LD       I1.4
=        L63.5
LD       L60.0
CALL     POS0_MAN, L63.7, L63.6, L63.5,
 +100000, I1.5, VB920, VD902, VD906, V910.0
Network 3        //使能自动模式。
LD       I0.0
EU
S        M0.0, 2
S        S0.1, 1
R        S0.2, 8
```

示例程序 2：使用 POSx_CTRL、POSx_RUN、POSx_SEEK 和 POSx_MAN 编程(续)



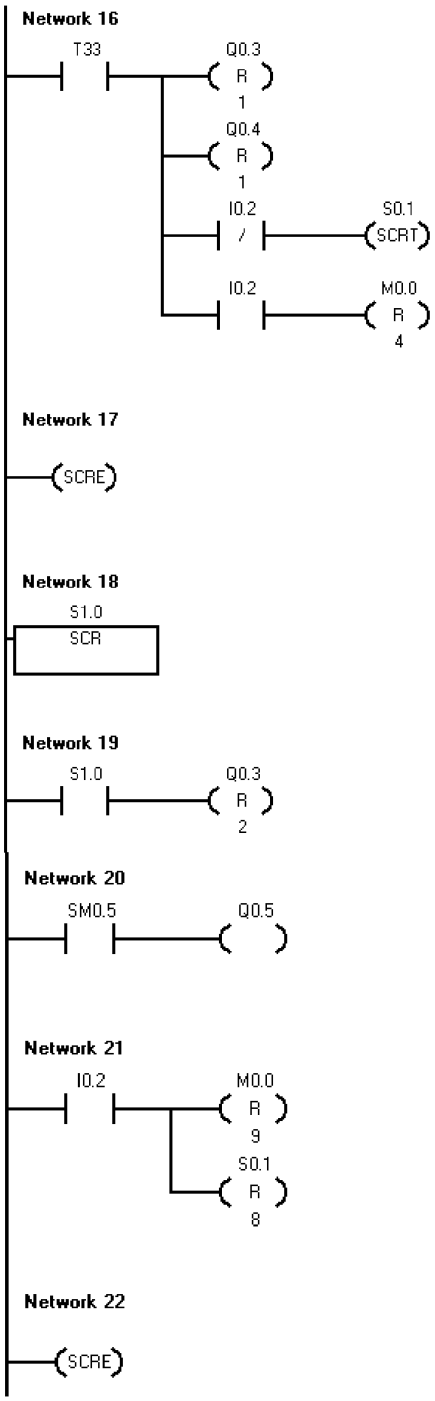

```
Network 4          //紧急停止。
                   //禁止模块和自动模式。
LD        I0.1
R         M0.0, 1
R         S0.1, 9
R         Q0.3, 3

Network 5          //自动模式时：
                   //接通运行灯。
LD        M0.0
=         Q0.1

Network 6
LSCR      S0.1

Network 7          //寻找参考点(RP)。
LD        S0.1
=         L60.0
LD        S0.1
=         L63.7
LD        L60.0
CALL      POS0_RSEEK, L63.7, M1.1, VB930

Network 8          //位于参考点(RP)时：
                   //钳起材料。
                   //进行下一步。
LD        M1.1
LPS
AB=       VB930, 0
S         Q0.3, 1
SCRT      S0.2
LPP
AB<>      VB930, 0
SCRT      S1.0

Network 9
SCRE

Network 10
LSCR      S0.2
```

示例程序 2：使用 POSx_CTRL、POSx_RUN、POSx_SEEK 和 POSx_MAN 编程(续)




```
Network 11          //使用包络1运动到相应位置。
LD      S0.2
=       L60.0
LD      S0.2
=       L63.7
LD      L60.0
CALL    POS0_RUN, L63.7, VB228, I0.1, M1.2,
        VB940, VB941, VB942, VD944, VD948

Network 12          //到达指定位置，接通切割机。
                    //进行下一步。
LD      M1.2
LPS
AB      VB940, 0
S       Q0.4, 1
R       T33, 1
SCRT    S0.3
LPP
AB<>    VB940, 0
SCRT    S1.0

Network 13
SCRE

Network 14          //等待切割结束。
LSCR    S0.3

Network 15
LD      S0.3
TON     T33, +200
```

示例程序 2：使用 POSx_CTRL，POSx_RUN，POSx_SEEK 和 POSx_MAN 编程(续)




```
Network 16          //切割结束时重新启动，
                    //除非STOP是接通的。
LD          T33
LPS
R           Q0.3, 1
R           Q0.4, 1
AN          I0.2
SCRT        S0.1
LPP
A           I0.2
R           M0.0, 4

Network 17
SCRE

Network 18
LSCR        S1.0

Network 19          //复位输出。
LD          S1.0
R           Q0.3, 2

Network 20          //故障灯闪烁。
LD          SM0.5
=           Q0.5

Network 21          //如果STOP接通，退出出错的程序。
LD          I0.2
R           M0.0, 9
R           S0.1, 8

Network 22
SCRE
```

使用 EM253 控制面板监控位控模块

为了帮助您开发您的运动控制方案，STEP7-Micro/WIN 提供 EM253 控制面板。其中的操作、组态和诊断标签可以帮助您在启动和测试阶段轻松地监视和控制位控模块的操作。

使用 EM253 控制面板可以验证位控模块是否正确地接线，可以调整组态数据并测试每个运动包络。

显示并控制位控模块的操作

控制面板的操作标签可以让您干涉位控模块的操作。控制面板显示位控模块的当前速度、当前位置和当前方向，你还可以看到输入和输出 LED 的状态（不包括 Pulse 的 LED）。

控制面板可以让你干涉位控模块、改变速度和方向、停止和启动运动、拖动工件（如果运动停止了）。

你还可以生成以下运动命令：

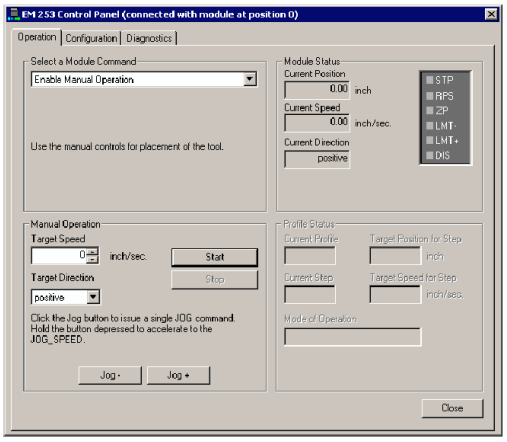


图 9-19 EM253 控制面板的操作标签

- 使能手动操作。你可以使用该命令手动定位工件。
- 运行运动包络。你可以使用这个命令选择执行某个包络。控制面板显示位控模块正在执行的包络的状态。
- 寻找参考点。该命令使用所组态的模式寻找参考点。
- 装载参考点偏移量。当你使用手动控制将工件拖动到一个新的零点后，你可以装载参考点偏移量。
- 重新装载当前位置。该命令可刷新当前位置值并建立一个新的零位置。
- 激活和禁止 DIS 输出。这些命令接通或断开位控模块上的 DIS 输出。
- 以脉冲触发 CLR 输出。该命令在位控模块的 CLR 输出产生一个 50ms 的脉冲。
- 教授一个运动包络。使用该命令你可以为一个运动包络存储目标位置和速度并记录你手动定位工件时的步。控制面板显示位控模块正在执行的包络的状态。
- 装载模块组态。该命令通过命令位控模块读取 S7-200V 区的组态块，可以装载一个新的组态。
- 运动到一个绝对位置。使用该命令，你可以以一个目标速度运动到指定位置。在使用该命令前，零位置必须已经建立。
- 以一个相对量运动。使用该命令可以从当前位置以一个目标速度移动一段指定距离。
- 复位命令接口。该命令清除位控模块的命令字节并置位 Done 位。当位控模块对命令没有响应时可以使用该命令。

显示并修改位控模块的组态

使用控制面板的组态标签可以查看并修改存储在 S7-200 数据块中的位控模块的组态设置。

修改过组态设置后，只需点击一个按钮即可同时刷新 STEP7-Micro/Win 项目和 S7-200 数据块中的设置

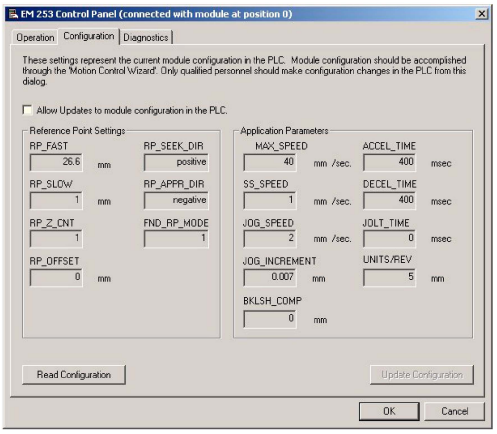


图 9-20 EM253 控制面板的组态标签

显示位控模块的诊断信息

使用控制面板的诊断标签可以查看关于位控模块的诊断信息。

您可以查看位控模块的特定信息，例如，模块在 I/O 总线上的位置，模块类型和硬件版本，以及用作该模块的命令字节的输出字节。

控制面板显示所命令的操作引起的错误条件。指令的错误代码请参见表 9-13。

您还可以查看位控模块所报告的所有错误条件。模块的错误代码请参见表 9-14。

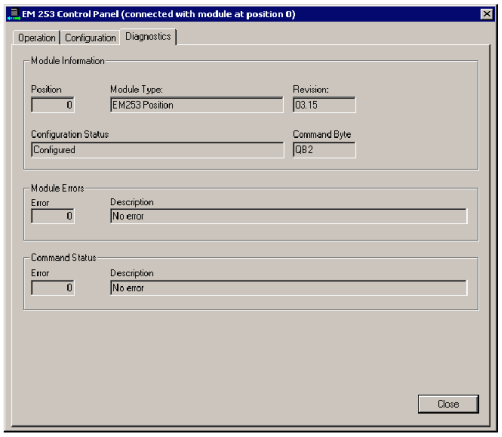


图 9-21 EM253 控制面板的诊断标签

位控模块和位控指令的错误代码

表 9-13 指令错误代码

错误代码	描述
0	无错
1	用户放弃
2	组态错误 使用 EM253 控制面板的诊断标签查看错误代码
3	非法命令
4	由于没有有效的组态而放弃 使用 EM253 控制面板的诊断标签查看错误代码
5	由于没有用户电源而放弃
6	由于没有定义的参考点而放弃
7	由于 STP 输入激活而放弃
8	由于 LMT-输入激活而放弃
9	由于 LMT+输入激活而放弃
10	由于运动执行的问题而放弃
11	没有为指定包络所组态的包络块
12	非法操作模式
13	该命令不支持的操作模式
14	包络块中非法的步号
15	非法的方向改变
16	非法的距离
17	RPS 触发在达到目标速度前出现
18	在 RP 附近时速度不是 RP_Slow
19	速度超出范围
20	没有足够的距离执行所希望的速度改变
21-127	保留
128	位控模块正在处理其它指令
129	位控模块错误
130	位控模块未使能
131	位控模块不能使用由于模块故障或未使能 （参见 POSx_CTRL 状态）

表 9-14 模块错误代码

错误代码	描述
0	无错
1	无用户电源
2	没有组态块
3	组态块指针错误
4	组态块的大小超过了可用的 V 区
5	非法的组态块格式
6	定义了太多的包络
7	非法的 STP_RSP 定义
8	非法 LIM-定义
9	非法 LIM+定义
10	非法的 FILTER_TIME 定义
11	非法的 MEAS_SYS 定义
12	非法的 RP_CFG 定义
13	非法的 PLS/REV 值
14	非法的 UNITS/REV 值
15	非法的 RP_ZP_CNT 值
16	非法的 JOG_INCREMENT 值
17	非法的 MAX_SPEED 值
18	非法的 SS_SPD 值
19	非法的 RP_FAST 值
20	非法的 RP_SLOW 值
21	非法的 JOG_SPEED 值
22	非法的 ACCEL_TIME 值
23	非法的 DECEL_TIME 值
24	非法的 JERK_TIME 值
25	非法的 BKLSH_COMP 值

高级议题

理解组态/包络表

位控向导基于您关于您的运动控制系统所给出的回答可以自动地生成组态和包络信息，帮助您轻松地完成运动程序。组态/包络表信息提供给那些想要创建他们自己的运动控制过程的高级用户。

组态/包络表位于 S7-200 的 V 区。正如表 9-15 所示，组态设置存储为下列形式的信息：

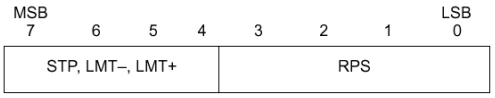
- 组态块包含用来作模块设置以备执行运动命令所需的信息。
- 这个交互作用的块支持通过用户程序对运动参数作直接设置。
- 每个包络块描述一个将由位控模块执行的预先定义的运动操作。最多可组态 25 个包络块。

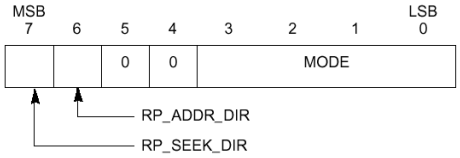
提示：

要生成多于 25 个的运动包络，您可以通过改变存储在组态/包络表指针中的数值变换组态/包络表。

表 9-15 组态/包络表

偏移量	名称	功能描述	类型
组态块			
0	MOD_ID	模块识别域	--
5	CB_LEN	以字节为单位的组态块的长度（1 字节）	--
6	IB_LEN	以字节为单位的交互块的长度（1 字节）	--
7	PF_LEN	以字节为单位的单个包络的长度（1 字节）	--
8	STP_LEN	以字节为单位的单步的长度（1 字节）	--
9	STEPS	每个包络允许的步数（1 字节）	--
10	PROFILES	从 0 到 25 的包络号（1 字节）	--
11	保留	设为 0x0000	--
13	IN_OUT-CFG	定义模块输入和输出的使用（1 字节） MSB 7 6 5 4 3 2 1 LSB 0 P/D POL 0 0 STP RPS LMT– LMT+ P/D 该位定义 P0 和 P1 的使用 正极性（POL=0）： 0 - P0 脉冲正转 P1 脉冲反转 1 - P0 脉冲转动 P1 控制转动方向（0 - 正向，1 - 反向） 负极性（POL=1）： 0 - P0 脉冲正转 P1 脉冲反转 1 - P0 脉冲转动 P1 控制转动方向（0 - 正向，1 - 反向）	--

偏移量	名称	功能描述	类型
		POL 该位为 P0 和 P1 选择极性转换 （0-正极性，1-负极性） STP 该位控制 STOP 输入的有效等级 RPS 该位控制 RPS 输入的有效等级 LMT- 该位控制反向移动限位输入的有效等级 LMT+ 该位控制正向移动限位输入的有效等级 0 - 有效等级高 1 - 有效等级低	
14	STOP_RSP	定义驱动对 STP 输入的响应（1 字节） 0 无响应，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 STP 输入 3-255 保留（指定该数值则出错）	–
15	LMT-_RSP	定义驱动对反向限位输入的响应（1 字节） 0 无响应，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 STP 输入 3-255 保留（指定该数值则出错）	–
16	LMT+_RSP	定义驱动对正向限位输入的响应（1 字节） 0 无响应，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 STP 输入 3-255 保留（指定该数值则出错）	–
17	FILTER_TIME	定义 STP、LMT-、LMT+ 和 RPS 输入的滤波时间（1 字节）  ‘0000’ 200 微秒 ‘0101’ 3200 微秒 ‘0001’ 400 微秒 ‘0110’ 6400 微秒 ‘0010’ 800 微秒 ‘0111’ 12800 微秒 ‘0011’ 1600 微秒 ‘1000’ 无滤波 ‘0100’ 1600 微秒 ‘1001’至‘1111’ 保留（指定该数值则出错）	–
18	MEAS_SYS	定义测量系统（1 字节） 0 脉冲（速度为每秒脉冲数，位置值为脉冲数）。数值存为 DINT。 1 工程单位（速度为每秒单位数，位置值为单位数）。数值存为 REAL。 2-255 保留（指定该数值则出错）	–
19	–	保留（设为 0）	–
20	PLS/REV	定义电机每转的脉冲数（4 字节） 只有当 MEAS_SYS 设为 1 时才有意义	DINT
24	UNITS/REV	定义电机每转的工程单位数（4 字节） 只有当 MEAS_SYS 设为 1 时才有意义	REAL
28	UNITS	保留给 STEP7-Micro/WIN 存储一个定制单位的字符串（4 字节）	–

偏移量	名称	功能描述	类型
32	RP_CFG	定义参考点寻找组态（1 字节）  RP_SEEK_DIR 该位定义参考点寻找的起始方向 （0－正向，1－反向） RP_APPR_DIR 该位定义结束参考点寻找的接近方向 （0－正向，1－反向） MODE 定义参考点寻找模式 ‘0000’ 参考点寻找禁止 ‘0001’ 参考点在 RPS 输入开始有效的点上 ‘0010’ 参考点在 RPS 输入有效区域中央 ‘0011’ 参考点在 RPS 输入有效区以外 ‘0100’ 参考点在 RPS 输入有效区内 ‘0101’至‘1111’ 保留（选择该数则出错）	–
33	–	保留（设为 0）	–
34	RP_Z_CNT	用来定义参考点的 ZP 输入脉冲数（4 字节）	DINT
38	RP_FAST	RP 寻找操作的高速：小于等于 MAX_SPD（4 字节）	DINT REAL
42	RP_SLOW	RP 寻找操作的低速：小于等于电机能够在瞬间停止的最大速度（4 字节）	DINT REAL
46	SS_SPEED	启动/停止速度（4 字节） 启动速度是电机能够瞬间从停止状态启动以及从运行状态瞬间停下的最大速度。允许低于该速度的操作，但加速和减速时间除外。	DINT REAL
50	MAX_SPEED	电机的最大操作速度（4 字节）	DINT REAL
54	JOG_SPEED	拖动速度。小于等于 MAX_SPEED（4 字节）	
58	JOG_INCREMENT	该拖动增量是相应于一个拖动脉冲应移动的距离（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时间，单位为毫秒（4 字节）	DINT
66	DECEL_TIME	从最大速度减速至最小速度所需时间，单位为毫秒（4 字节）	DINT
70	BKLSH_COMP	螺距误差补偿：方向转换时用于补偿系统螺距误差的距离（4 字节）	DINT REAL
74	JERK_TIME	在加速/减速曲线（S 曲线）的起始和结束两端进行急停补偿的时间。定义为零值则禁止急停补偿。急停时间以毫秒为单位（4 字节）	DINT
交互作用的块			
78	MOVE_CMD	选择操作模式（1 字节） 0 绝对位置 1 相对位置 2 单速连续正向转动 3 单速连续反向转动 4 手动速度控制，正转	–

偏移量	名称		功能描述	类型
			5 手动速度控制，反转 6 带有触发停止的单速连续正向转动（RPS 输入指示停止） 7 带有触发停止的单速连续反向转动（RPS 输入指示停止） 8-255 保留（如果指定为该数值则出错）	
79	–		保留，设为 0	–
80	TARGET_POS		该运动的目标位置（4 字节）	DINT REAL
84	TARGET_SPEED		该运动的目标速度（4 字节）	DINT REAL
88	RP_OFFSET		参考点的绝对位置（4 字节）	DINT REAL
包络块 0				
92 (+0)	步(STEPS)		该运动顺序的步数（1 字节）	–
93 (+1)	模式(MODE)		选择该包络块的操作模式（1 字节） 0 绝对位置 1 相对位置 2 单速连续正转 3 单速连续反转 4 保留（定义的错误） 5 保留（定义的错误） 6 带有触发停止的单速连续正转（RPS 选择速度） 7 带有触发停止的单速连续反转（RPS 输入指示停止） 8 两速，连续正向转动（RPS 选择速度） 9 两速，连续反向转动（RPS 选择速度） 10-255 保留（如果指定为该数值则出错）	–
94 (+2)	0	POS	运动步 0 要去的位置（4 字节）	DINT REAL
98 (+6)	0	SPEED	运动步 0 的目标速度（4 字节）	DINT REAL
102 (+10)	1	POS	运动步 1 要去的位置（4 字节）	DINT REAL
106 (+14)	1	SPEED	运动步 1 的目标速度（4 字节）	DINT REAL
110 (+18)	2	POS	运动步 2 要去的位置（4 字节）	DINT REAL
114 (+22)	2	SPEED	运动步 2 的目标速度（4 字节）	DINT RE AL
118 (+26)	3	POS	运动步 3 要去的位置（4 字节）	DINT REAL
122 (+30)	3	SPEED	运动步 3 的目标速度（4 字节）	DINT REAL

偏移量	名称		功能描述	类型
包络块 1				
126 (+34)	STEPS		该运动序列中的步数（1 字节）	–
127 (+35)	MODE		为该包络块选择操作模式（1 字节）	–
128 (+36)	0	POS	运动步 0 要去的位置（4 字节）	DINT REAL
132 (+40)	0	SPEED	运动步 0 的目标速度（4 字节）	DINT REAL
…	…	…	…	…

[1] 组态/包络表的包络块最多可包含 25 个运动包络。要生成多于 25 个的包络，你可以通过改变存储在组态/包络表指针中的数值，交换组态/包络表。

位控模块的特殊存储区分配

S7-200 按照位控模块在 I/O 系统中的物理位置给每个智能模块分配 50 个字节的特殊存储区（SM）参见表 9-16。当模块检测到错误条件或数据状态的变化，该模块会刷新这些 SM 的区域。当要求报告错误条件时，每一个模块刷新 SMB200 至 SMB249，第二个模块刷新 SMB250 至 SMB299，以此类推。

表 9-16 SMB200 至 SMB549

一个智能模块在以下各槽的 SM 字节：						
槽 0	槽 1	槽 2	槽 3	槽 4	槽 5	槽 6
SMB200 至 SMB249	SMB250 至 SMB299	SMB300 至 SMB349	SMB350 至 SMB399	SMB400 至 SMB449	SMB450 至 SMB499	SMB500 至 SMB549

表 9-17 所示是一个智能模块 SM 数据区分配的结构。这个定义是按照智能模块位于 I/O 系统的 0 号槽给出的。

表 9-17 特殊存储区：双向 RAM 区的定义（区域 8 至 15）

SM 地址	描述
SMB200 至 SMB215	模块名称（16 个 ASCII 字符）SMB200 是第一个字符：“EM253 位置”
SMB216 至 SMB219	软件版本号（4 个 ASCII 字符）。SMB216 是第一个字符。
SMW220 SMB222	模块错误代码。反映模块输入和输出的状态。 MSB 7 6 5 4 3 2 1 0 LSB DIS | 0 | 0 | STP | LMT– | LMT+ | RPS | ZP DIS 禁止输出 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
SMB223	瞬间模板状态。反映模板的组态状态和转向的状态 MSB 7 6 5 4 3 2 1 0 LSB 0 | 0 | 0 | 0 | 0 | OR | R | CFG OR 目标速度超范围 0=在范围内；1=超范围 R 转动方向 0=正转 1=反转 CFG 组态的模板 0=未组态 1=已组态
SMB224	CUR_PF 是一个字节，它指示当前正在执行的包络
SMB225	CUR_STP 是一个字节，它指示包络中当前正在执行的步
SMD226	CUR_POS 是一个双字，该值指示模块的当前位置
SMD230	CUR_SPD 是双字，该值指示模块的当前速度
SMB234	指令结果。表 9-13 是错误代码的描述。大于 127 的错误条件由向导生成的指令子程序产生 MSB 7 6 LSB 0 D | ERROR D Done 位 0=操作在进行中 1=操作完成（初始化过程中由模块设置）
SMB235 至 SMB244	保留
SMB245	与该模块用作命令接口的第一个 Q 字节之间的偏移量。该偏移量由 S7-200 自动提供以方便用户，而非模板所需。
SMD246	指向组态/包络表 V 区地址的指针。指向 V 区以外区域的指针值无效。位控模块会一直监视该指针所指向的区域直至它收到一个有效的指针值。

理解位控模块的命令字节

位控模块提供一个字节的实际输出作为命令字节。图 9-22 所示为命令字节的定义。表 9-18 所示是命令代码的定义。

当有数据向命令字节写入使得 R 由 0 变 1 时，模块认为有新的命令写入。

当一个命令正处于激活状态，模块检测到有向闲置状态的转换时（R 位状态变为 0），则放弃正在进行中的操作，并且，如果有运动在进行中，则执行减速停止。

一个操作完成后，模块必须看到一个向闲置的转换，然后才能接受新的命令。如果某个操作被放弃了，模块必须先完成所有的减速才能接受一个新命令。当有命令激活时，命令代码值的任何改变都忽略不计。

当 S7-200 出现操作模式的转换或故障条件时，位控模块的响应由 S7-200 实际输出的结果控制，而 S7-200 的实际输出的状态在 S7-200 的功能中已有定义：

R 0=闲置
1=执行命令代码所指定的命令（见表 9-18）

图 9-22 命令字节的定义

表 9-18 命令代码定义

命令代码	命令	
000 0000 到 001 1000	0~24	执行包络块 0~24 中指定的运动操作
001 1001 到 111 0101	25~117	保留（指定该数值则出错）
111 0110	118	激活 DIS 输出
111 0111	119	禁止 DIS 输出
111 1000	120	触发 CLR 脉冲输出
111 1001	121	重新装载当前位置
111 1010	122	执行交互块中指定的运动
111 1011	123	获取参考点偏移量
111 1100	124	正向拖动
111 1101	125	反向拖动
111 1110	126	寻找参考点位置
111 1111	127	重新装载组态

- 如果 S7-200 从 STOP 到 RUN：
  S7-200 中的程序控制位控模块的操作。
- 如果 S7-200 从 RUN 到 STOP：您可以选择这些实际输出在 S7-200 转为 STOP 时的状态，或者这些输出保持其最后的状态。
  - 如果 STOP 时 R 位被关断：位控模块减速所有进行中的运动至停止。
  - 如果 STOP 时 R 位是接通的：位控模块完成所有进行中的运动。如果没有正在进行的运动，则位控模块执行 ID 位所指定的包络。
  - 如果 R 位保持其最后的状态：位控模块完成所有进行中的运动。
- 如果 S7-200 检测到一个致命错误并关断所有实际输出：位控模块减速所有进行中的运动至停止。

位控模块有了看门狗定时器，如果发现与 S7-200 的通讯中断，则关断所有输出。如果看门狗时间到，位控模块减速所有进行中的运动至停止。

如果检测到模块的硬件或版本的致命错误，位控模块将输出 P0、P1、DIS 和 CLR 设为非激活状态。

表 9-19 运动命令

命令	描述
命令 0-24： 执行包络块 0-24 中定义的运动	该命令执行时，位控模块执行包络块中 MODE 域指定的运动操作，该包络块由命令的命令代码部分指示。交互块运动操作的定义是非隐藏的，所以，每次位控模块接收该命令时都会读取这些参数。 • 在模式 0（绝对位置）下，运动包络定义 1 至 4 步，每一步都包含对这个运动段进行描述的位置（POS）和速度（SPEED）参数。POS 表述的是一个基于参考点位置的绝对位置。运动方向则由当前位置与包络中第一步的位置的相互关系决定。在多步运动中禁止反向移动，并且报告反向移动造成的错误条件。 • 在模式 1（相对位置）下，运动包络 1 至 4 步，每一步都包含对这个运动段进行描述的位置（POS）和速度（SPEED）参数。位置值（POS）的符号位决定运动的方向。在多步运动中，禁止反向移动，并且报告反向移动造成的错误条件。 • 在模式 2 和 3 下（单速，连续速度模式），忽略位置（POS）参数，模块加速到第一步中指定的速度 SPEED。模式 2 用于正转，模式 3 用于反转。 • 在模式 6 和 7 下（带有触发停止的单速、连续速度模式），模块加速到第一步中指定的速度 SPEED。如果一旦 RPS 输入激活，运动在完成第一步中 POS 指定的距离后停止。模式 6 用于正转，模式 7 用于反转。 • 在模式 8 和 9 下，RPS 输入的二进制值将选择包络块中前两步所定义的两个连续速度中的一个作为速度值。 - 如果 RPS 未激活：步 0 控制驱动的速度。 - 如果 RPS 激活：步 1 控制驱动的速度。 模式 8 用于正转，模式 9 用于反转。SPEED 值控制运动速度。POS 值在该模式下忽略不计。
命令 118 激活 DIS 输出	该命令执行时，位控模块激活 DIS 输出。
命令 119 禁止 DIS 输出	该命令执行时，位控模块禁止 DIS 输出。
命令 120 触发 CLR 脉冲输出	该命令执行时，位控模块在 CLR 输出产生一个 50ms 脉冲。
命令 121 重新装载当前位置	该命令执行时，位控模块将交互块 TARGET_POS 参数值置为当前位置值。
命令 122 执行交互块中指定的运动	该命令执行时，位控模块执行交互块 MOVE_CMD 域中指定的运动操作。交互块运动操作的参数是非隐藏的，所以位控模块每次接到该命令时都会读取这些参数。 • 在模式 1 和 2 下（绝对和相对运动模式），执行单步运动，其目标速度和位置信息由交互块中的 TARGET_SPEED 和 TARGET_POS 提供。 • 在模式 2 和 3 下（单速、连续速度模式），忽略位置参数，位控模块加速到交互块中 TARGET_SPEED 指定的速度值。 • 在模式 4 和 5 下（手动速度控制模式），忽略位置参数，您的程序将变化的速度值装入交互块的 TARGET_SPEED。位控模块会持续地监视该参数区域并在速度值变化时适时地响应。
命令 123 获取参考点偏移量	当该命令执行时，位控模块建立一个不同于参考点位置的零位置。 在发出该命令前，您必须确定参考点位置并把工件拖到工作的起始位置，接收到该命令后，位控模块开始计算工作起始位置（当前位置）与参考点位置之间的偏移量，并且将计算得到的偏移量写入交互块的 RP_OFFSET 区域。随后将当前位置设为 0，即将工作起始位置设为零位置。 当步进电机无法跟踪其当前位置（如掉电或步进电机被手动重新定位），可以发出寻找参考点命令来自动重建零位置。
命令 124 正向拖动	使用该命令可以手动发出脉冲正向移动步进电机。 如果该命令的有效时间短于 0.5 秒，位控模块发出脉冲，移动 JOG_INCREMENT 所指定的距离。 如果该命令能保持 0.5 秒或更长，模块则开始加速至 JOG_SPEED 所指定的速度。 当检测到向闲置状态的转换时，位控模块减速至停止。

命令	描述
命令 125 反向拖动	使用该命令可手动发出脉冲反向移动步进电机。 如果该命令的有效时间短于 0.5 秒，位控模块发出脉冲，移动 JOG_INCREMENT 所指定的距离。 如果该命令能保持 0.5 秒或更长，模块则开始加速到 JOG_SPEED 所指定的速度。 当检测到向闲置状态的转换时，位控模块减速至停止。
命令 126 寻找参考点位置	当该命令执行时，位控模块按照指定的寻找模式发出一个参考点寻找操作。当参考点被锁定并且运动停止时，位控模块将从交互块的 RP_OFFSET 中读取的数值装载为当前位置。
命令 127 重新装载组态	该命令执行时，位控模块从 EM 区的适当区域读取组态/包络表指针，然后，到组态/包络表指针所指定的区域读取组态块。模块将刚刚得到的组态数据与现有模块组态进行比较并执行任何所需的设置改变和重新计算。放弃所有隐藏的包络。

创建您自己的运动控制指令

位控向导为位控模块的操作控制创建位控指令，但是，您也可以创建您自己的指令。以下的 STL 指令程序提供了一个示例，即如何为位控模块创建您自己的控制指令。

该示例中使用 S7-200CPU，位控模块位于 0 号槽。位控模块上电时组态，CMD_STAT 是 SMB234 的符号。CMD 是 QB2 的符号，NEW_CMD 是该包络的符号。

示例程序：控制位控模块

```
Network1        //新的运动命令状态
LSCR    Stae_0
Network2        //CMD_STAT 是 SMB234 的符号
                //CMD 是 QB2 的符号
                //NEW_CMD 是包络的符号
                //1 清除位控模块的 Done 位。
                //2 清除位控模块的命令字节。
                //3 发布新命令。
                //4 等待命令执行。
LD      SM0.0
MOVB    0,CMD_STAT
BIW     0,CMD
BIW     NEW_CMD,CMD
SCRT    State_1
Network3
SCRE
Network4        //等待命令完成。
LSCR    State_1
Network5        //如果命令完成，进入闲置状态。
LDB=    CMD_STAT,16#80
SCRT    Idle_State
Network6        //如果命令出错，进入错误处理状态。
LDB>    CMD_STAT,16#80
SCRT    Error_State
Network7
SCRE
```

创建调制解调模块程序

10

使用 EM241 调制解调模块可以将 S7-200 直接连到一个模拟电话线上，并且支持 S7-200 与 STEP 7-Micro/WIN 的通讯。该调制解调模块还支持 Modbus 从站 RTU 协议，该模块与 S7-200 之间的通讯通过扩展 I/O 总线实现。

STEP 7-Micro/WIN 提供一个调制解调扩展向导，它可以帮助您设置一个远端的调制解调器，或者设置将 S7-200 连向远端设备的调制解调模块。

本章内容：

EM241 调制解调模块特点

使用调制解调模块可将 S7-200 直接连到模拟电话线上，并且还提供以下特性：

- 提供国际电话线接口
- 提供与 STEP 7-Micro/WIN 的调制解调接口，可进行编程和诊断（teleservice）
- 支持 Modbus RTU 协议
- 支持数字和文本的寻呼
- 支持 SMS 短信息
- 允许 CPU 到 CPU 或 CPU 到 Modbus 的数据传送
- 密码保护
- 提供安全回拨功能
- 调制解调模块的组态存储在 CPU 中

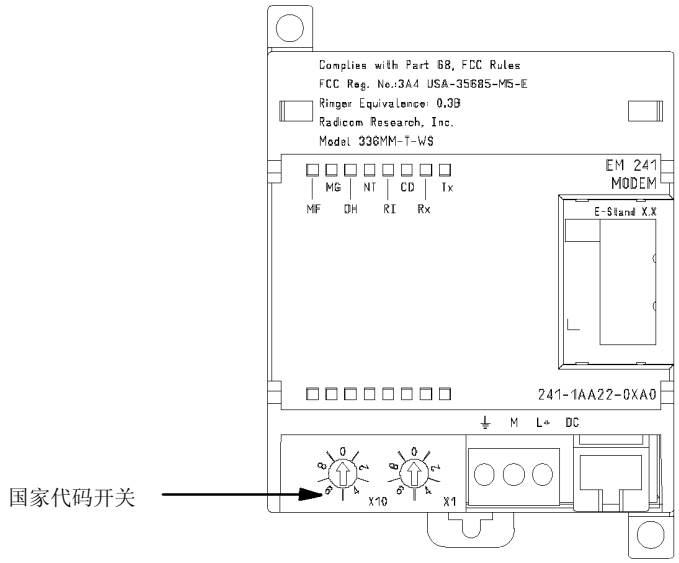


图 10-1 EM241 调制解调模块

您可以使用 STEP 7-Micro/WIN 调制解调扩展向导来组态调制解调模块。EM241 调制解调模块的技术参数如附录 A 所示。

国际电话线接口

调制解调模块是一个标准的 V.34（33.6K 波特）、10 位调制解调器，它与大多数内置或外置 PC 调制解调器相兼容。该调制解调模块不能够与 11 位调制解调器通讯。

您可以通过调制解调模块前端的六位四线 RJ11 接口（见图 10-2）将模块连到电话线上，在不同的国家可能需要适配器将 RJ11 接口转换连接到标准电话线上。更多的信息请参见您的适配器的说明书。

调制解调器和电话线接口需要外供直流 24V。可以使用 CPU 传感器电源或外部电源。将调制解调模块的接地端连到系统的地线。

当调制解调模块上电时，模块自动按特定国家的操作组态电话接口。模块前面的两个旋钮开关用来选择国家代码。您必须在模块上电前选好所要的国家。所支持的国家，请参见表 10-1。

表 10-1 EM241 所支持的国家

开关设置	国家
01	奥地利
02	比利时
05	加拿大
08	丹麦
09	芬兰
10	法国
11	德国
12	希腊
16	爱尔兰
18	意大利
22	卢森堡
25	荷兰
27	挪威
30	葡萄牙
34	西班牙
35	瑞典
36	瑞士
38	英国
39	美国

STEP 7-Micro/WIN 接口

您可以使用调制解调模块通过电话线与 STEP 7-Micro/WIN 通讯（teleservice）。当您使用 STEP 7-Micro/WIN 并将调制解调模块用作远端调制解调器时，无须在 S7-200 CPU 中组态或编程。

在 STEP 7-Micro/WIN 中按照以下步骤使用调制解调模块：

1. S7-200 断电，将调制解调模块连至 I/O 扩展总线上。不要在 S7-200 CPU 上电时连接任何 I/O 模块。
2. 将电话线连接到调制解调模块上。如需要可使用适配器。
3. 给调制解调模块连接 24V 直流供电。
4. 将调制解调模块上的地与系统的地相连接。
5. 设置国家代码开关。
6. 给 S7-200 CPU 和调制解调模块上电。
7. 组态 STEP 7-Micro/WIN 与 10 位调制解器通讯。

Modbus RTU 协议

您可以将调制解调模块组态作为 Modbus RTU 从站。调制解调模块通过调制解调接口接收 Modbus 请求，解释这些请求与 CPU 交换数据。然后，调制解调模块生成一个 Modbus 响应并通过调制解调接口传送出去。

提示：

如果调制解调模块组态为 Modbus RTU 从站，STEP 7-Micro/WIN 不能通过电话线与调制解调模块通讯。

调制解调模块所支持的 Modbus 功能如表 10-2 所示。

Modbus 功能 4 和功能 16 允许在一个请求中最多读或写 125 个保持寄存器（250 字节的 V 区）。功能 5 和功能 15 可写 CPU 的映象寄存器。这些值可被用户程序覆盖。

Modbus 地址通常写作包括数据类型和偏移量在内的 5 和 6 个字符。前一个或二个字符决定数据类型，最后的 4 个字符在数据类型的范围内选择适当的值。Modbus 主设备会将这些地址映射到正确 Modbus 功能中。

表 10-2 调制解调模块支持的 Modbus 功能

功能	描述
功能 01	读线圈（输出）状态
功能 02	读输入状态
功能 03	读保持寄存器
功能 04	读输入（模拟输入）寄存器
功能 05	写单个线圈（输出）
功能 06	预设单个寄存器
功能 15	写多个线圈（输出）
功能 16	预设多个寄存器

表 10-3 所示是调制解调模块所支持的 Modbus 地址以及 Modbus 到 S7-200 CPU 地址的映射关系。

使用调制解调扩展向导可为调制解调模块生成一个支持 Modbus RTU 协议的组态块。在使用 Modbus 协议之前，必须将调制解调模块组态块下载到 CPU 的数据块中。

表 10-3 映射 Modbus 地址至 S7-200 CPU

Modbus 地址	S7-200 CPU 地址
000001	Q0.0
000002	Q0.1
000003	Q0.2
…	…
000127	Q15.6
000128	Q15.7
010001	I0.0
010002	I0.1
010003	I0.2
…	…

Modbus 地址	S7-200 CPU 地址
010127	I15.6
010128	I15.7
030001	AIW0
030002	AIW2
030003	AIW4
…	…
030032	AIW62
040001	VW0
040002	VW2
040003	VW4
…	…
04xxxx	VW 2*(xxxx-1)

寻呼和 SMS 短信息

调制解调模块支持向移动电话发送数字和文本的寻呼信息以及 SMS（短信息服务）信息（在那些移动电话服务提供商所支持的地方），这些信息和电话号码存储在调制解调模块组态块中，这个组态块必须下载到 S7-200 CPU 的数据块中。您可以使用调制解调向导为调制解调组态块生成信息和电话号码。调制解调扩展向导还生成程序代码以供您的程序在发送这些信息时调用。

数字寻呼

数字寻呼使用语音电话的语音功能向寻呼机发送数字值。调制解调模块拨出所请求的寻呼服务，等待语音信息后完成，然后，发送数字寻呼信息。在寻呼信息中可以使用 0 到 9，星号(*)，A、B、C 和 D。星号、A、B、C 和 D 实际显示在寻呼机上的字符是非标准的，由寻呼机以及寻呼服务提供商决定。

文本寻呼

文本寻呼可以将字母信息传送给寻呼提供商，然后再传到寻呼机上，文本寻呼提供商通常使用一条与调制解调器相连的电话线来接收文本寻呼。调制解调模块使用本地电信文字数字协议（TAP）向服务提供商发送文本信息。许多文本寻呼提供商使用这个协议接收信息。

短信息服务（SMS）

一般来说，一些移动电话服务支持那些与 GSM 相兼容的短信息服务（SMS）信息。SMS 允许调制解调模块通过一条模拟电话线向 SMS 提供商发送信息。SMS 提供商再将信息传送到移动电话上，信息则以文本的形式显示在电话上，调制解调模块使用本地电信文字数字协议（TAP）和通用计算机协议（UCP）向 SMS 提供商发送信息，您只能向那些所用的调制解调设备支持这些协议的 SMS 提供商发送 SMS 信息。

文本和 SMS 信息中的嵌入变量

调制解调模块能够在文本信息中嵌入来自 CPU 的数据并且基于在信息中的定义格式化该数据。你可以定义小数点左边和右边数字的位数以及小数点是逗号还是点号。当用户程序命令调制解调模块发送一条文本信息时，调制解调模块从 CPU 获取信息，判断该信息中需要什么样的 CPU 值，从 CPU 中获取这些数值，然后，将这些数值以一定的格式放置在文本信息的适当位置上，最后传送给信息的服务提供商。

信息提供商的电话号码、信息以及信息中的嵌入数据是通过多个 CPU 循环周期从 CPU 中读取的。当一条信息正在发送时，您的用户程序不能修改电话号码或信息。信息中的嵌入数据值则可以在信息发送过程中连续地被刷新，如果信息中包含多个变量，那些变量是通过多个 CPU 循环周期读取的，如果您要信息中所有的嵌入数据保持一致，那么，在发送信息时，不要修改任何嵌入变量。

数据传送

调制解调模块允许用户程序通过电话线向其它 CPU 或 Modbus 设备传送数据，传送的数据和电话号码可通过调制解调扩展向导组态，存储在调制解调模块的组态块中。该组态块随后要下载到 S7-200 CPU 的数据块中。调制解调扩展向导还生成程序代码，在用户程序中调用这些程序进行数据传送。

数据传送可以是从远端设备中读数据，也可以是向远端设备写数据，数据传送可以读或写 1 至 100 个数据字。数据传送从或者向所连接的 CPU 中读取或写入数据。

您在调制解调扩展向导中创建的一个数据传送可以包括一个单一的从远端设备的读操作，一个单一的向远端设备的写操作，或者与远端设备间的既读又写的操作。

数据传送使用调制解调模块的组态协议，如果该模块被组态为支持 PPI 协议（当它响应 STEP 7-Micro/WIN 时），模块使用 PPI 协议传送数据。如果该模块被组态为支持 Modbus RTU 协议，数据传送则使用 Modbus 协议。

远端设备的电话号码，数据传送请求以及被传送的数据是通过多个 CPU 循环周期从 CPU 中读取的。当信息发送时，您的程序不能修改电话号码或信息，而且，在信息传送过程中也不能够修改传送的数据。

如果远端设备也是一个调制解调模块，数据传送中可以使用密码功能，只需在电话号码的组态中输入远端调制解调模块的密码即可。数据传送不能使用回拨功能。

密码保护

密码安全是调制解调模块的可选功能，可在调制解调扩展向导中使能，调制解调模块使用的密码与 CPU 密码不同。调制解调模块的密码是另外的一个 8 字符的密码，拨号者必须向调制解调模块提供该密码，否则不允许访问调制解调模块所连接的 CPU，密码作为调制解调模块组态块的一部分存在 CPU 的 V 区内。调制解调模块的组态块必须下载到其连接的 CPU 的数据块中。

如果 CPU 的密码安全在系统块中被使能了，拨号者必须提供 CPU 密码以获取对任何密码保护功能的访问权。

安全回拨

安全回拨是调制解调模块的可选功能，在调制解调扩展向导中组态，回拨功能只允许那些预先设定的电话号码访问 CPU，从而为 CPU 提供了额外的安全保护。回拨功能使能时，调制解调模块应答所有进入的拨号，验证拨号者，然后断开链接。如果是授权的拨号者，调制解调模块则回拨为该拨号者预先设定的电话号码，允许其访问 CPU。

调制解调模块支持三种回拨模式：

- 只预先定义一个号码，对此号码回拨。
- 预先设定多个号码，对这些号码皆可回拨。
- 对任何号码都可回拨。

在调制解调扩展向导中选择适当的选项确定回拨模式，然后定义回拨的电话号码。回拨电话号码存储在调制解调模块的组态块中，该组态块存储在所连接的 CPU 的数据块中。

最简单的回拨形式是一个单个的预先定义的电话号码。如果在调制解调模块的组态块中存储了一个回拨号码，每当调制解调模块应答进入的拨号时，它通知拨号者回拨功能已使能，断开拨号者，然后回拨在组态块中指定的号码。

调制解调模块还支持对多个预设电话号码的回拨功能。在该模式下，向拨号者查询电话号码，如果拨号者提供的号码与模块组态块中预先设定的号码中的某一个相匹配，调制模块则断开拨号者，然后使用与组态块中的号码相匹配的号码进行回拨。用户可最多组态 250 个回拨号码。

当存在多个预先设定的回拨号码时，向所连接的调制解调模块提供的回拨号码，除了前两个数字外，要与模块的组态块中的号码完全匹配。例如，如果组态的回拨号码是 91（123）4569999，由于可能需要拨外线（9）和长途（1），所提供的回拨号码可以是以下号码中的任何一个：

- 91（123）4569999
- 1（123）4569999
- （123）4569999

以上所有号码都可以认作是一个匹配的回拨号码。在进行回拨时，调制解调模块使用其组态块中的回拨电话号码，在本例中是 91（123）4569999。在组态多个回拨号码时，确保所有电话号码除前两位数字以外，是唯一的。进行回拨号码比较时，在一个电话号码中只使用数字字符。象逗号或括号这样的字符在回拨号码比较中忽略不计。

调制解调扩展向导中，在回拨组态中选择“Enable callbacks to any phone number”这一选项即可设置成对任意电话号码进行回拨，如果选择这一选项，调制解调模块应答到来的拨号并请求一个回拨号码，在拨号者提供电话号码后，模块断开链接并拨打该电话号码。这种回拨模式只提供一种向调制解调模块一方的连接收取电话费用的方式，并不为 S7-200 CPU 提供任何安全保护。若使用这种回拨模式，应使用调制解调模块的密码功能提供安全保护。

调制解调模块的密码和回拨功能可同时使能。模块在处理回拨功能之前，要求拨号者提供正确的密码。

EM241 组态表

所有的文本信息，电话号码，数据传输信息，回拨号码以及其它选项都存储在调制解调模块的组态表中，该组态表必须下载到 CPU 的 V 区。调制解调扩展向导引导您完成整个调制解调模块组态表的生成。STEP 7-Micro/WIN 则将组态表存储到数据块中，该数据块要下载到 S7-200 CPU 中。

调制解调模块在 CPU 运动时以及 CPU 的任何一次从 STOP 到 RUN 转换的 5 秒钟内读取该组态表。只要调制解调模块与 STEP 7-Micro/WIN 连接，模块就不会从 CPU 读取新的组态表。如果调制解调模块在线时有新的组态表下载，该模块会在任务结束时读取新的组态表。

如果调制解调模块检测到组态表中的错误，模块前部指示模块好的 LED（MG）会闪烁。查看 STEP 7-Micro/WIN 中的 PLC Information （PLC 信息），或读取 SMW220（由 0 号槽的模块占用）中关于组态错误信息的数值。调制解调模块的组态错误如表 10-4 所列。如果您使用调制解调扩展向导来创建调制解调模块的组态表，STEP 7-Micro/WIN 在生成组态表前会进行数据检查。

表 10-4 EM241 组态错误（十六进制）

错误	描述
0000	没有错误
0001	无 24V 直流外供电源
0002	模块故障
0003	无组态块 ID — 位于组态表起始处的 EM241 标识对于该模块无效。
0004	组态块超范围 — 组态表指针未指向 V 区，或者表的一部分超出了所连的 CPU 的 V 区范围
0005	组态错误 — 回拨功能使能并且回拨的电话号码数等于 0 或大于 255。信息数大于 250。信息发送的电话号码数大于 250，或者信息发送的电话号码的长度大于 120 字节。
0006	国家代码选择错误 — 不支持国家选择旋钮开关上设定的数值
0007	电话号码太大 — 回拨功能使能并且回拨号码长度大于上限
0008-00FF	保留
01xx	回拨号码 xx 有错 — 在回拨电话号码 xx 中有非法字符。xx 值为 1 意味着第一个回拨号码，xx 值为 2 是指第二个回拨号码，以此类推。
02xx	电话号码 xx 有错 — 在信息电话号码 xx 或数据传送的电话号码 xx 的某个区域中包含一个非法值。数值 xx 为 1 表示是第一个电话号码，2 则指第二个号码等等。
03xx	信息 xx 有错 — 信息或数据传输的数量 xx 超过允许的最大长度，数值 xx 为 1 代表第一条信息，2 指第二条信息等等
0400-FFFF	保留

EM241 状态 LED

位控模块的前面板有 8 个指示状态的 LED。表 10-5 描述了这些状态 LED。

表 10-5 EM241 状态 LED

LED	描述
MF	模块故障 — 当模块检测到以下故障条件时，该 LED 灯亮： • 无 24V 直流外供电源 • I/O 看门狗超时 • 模块故障 • 与本地 CPU 通讯出错
MG	模块正常 - 当没有模块故障条件时该 LED 灯亮。如果组态表中有错或者用户为电话线接口设置了非法的国家代码设置，模块正常的 LED 会闪烁。可查看 STEP 7-Micro/WIN 的 PLC Information (PLC 信息)画面，或者读取 SMW220（用于位于 0 号槽的模块）的数值以获得组态错误的信息
OH	Off Hook - 当 EM241 正在使用电话线时该 LED 亮
NT	无拨号音 - 当 EM241 接到发送信息的命令而电话线上无拨号音时，该 LED 灯亮指示有错误条件。只有当 EM241 被组态为拨号前检查拨号音的时候才会出现这个错误条件。当一次拨号尝试失败后该 LED 接通并保持 5 秒左右。
RI	振铃指示 - 该 LED 指示 EM241 正在接收一个拨入电话。
CD	载波检测 - 该 LED 指示与远程调制解调器的连接已建立。
RX	接收数据 - 当调制解调器接收数据时该 LED 闪烁。
TX	传送数据 - 当调制解调器进行数据传送时，该 LED 闪烁。

利用调制解调扩展向导组态 EM241 调制解调模块

在 STEP 7-Micro/WIN 工具菜单或浏览条的工具组件中启动调制解调扩展向导。

要使用该向导，必须对项目进行编译并设为符号寻址模式。如果您还未编译您的程序，请现在编译。

1. 在 Modem 扩展向导的第一个画面中，选择组态 EM241 Modem 模块并点击 Next>
2. 调制解调扩展向导要求调制解调模块相对于 S7-200 CPU 的位置信息以便生成正确的程序代码。点击读模块按钮可自动读取与 CPU 相连的智能模块的位置。扩展模块的号码从零开始顺序排列。双击你要组态的 Modem 模块，或在模块位置域中设置 Modem 模块的位置，点击 Next>

提示：

在硬件版本号 1.2 之前的 S7-200 CPU 中，智能模块必须位于紧靠 CPU 的位置，否则 Modem 扩展向导无法组态该模块。

3. 在密码保护画面中可以使能 Modem 模块的密码保护功能并为该模块设置一个 1 至 8 个字符的密码。该密码独立于 S7-200 CPU 的密码，当该模块设为密码保护后，任何人试图通过该 Modem 模块来连接 S7-200 CPU 时都必须输入一个正确的密码，如果需要选择密码功能，并输入一个密码，点击 Next>

4. Modem 模块支持两种通讯协议：PPI 协议（与 STEP 7-Micro/WIN 通讯）和 Modbus RTU 协议。选择什么样的通讯协议取决于远端的通讯对象。该设置控制 Modem 模块在应答拨入，以及进行 CPU 数据传送时所使用的通讯协议。选择适当的协议并点击 Next>。
5. 你可以组态该模块向呼机发送数字和文本信息，或向手机发送短信息（SMS）。点击信息使能多选框并点击组态信息...按钮，定义信息及接收方的电话号码。
6. 当设置向手机或呼机发送信息时，你必须定义信息和电话号码。选择组态信息画面的信息标签并点击新信息按钮。输入信息的文字内容并指定插入到信息中的 CPU 数据。要在信息中插入 CPU 数据，将光标放在数据的位置上然后点击插入数据...按钮。指定 CPU 数据的地址（如：VW100），显示格式（如：有符号整数）以及小数点左右的位数。你还可以指定小数点是点号还是逗号。
   - 数据寻呼仅限于数字 0-9，字母 A，B，C，D 以及星号（*）。数字寻呼信息的最大长度取决于服务提供商。
   - 文本信息最长可有 119 个字符并且可以是所有的字母数字字符。
   - 文本信息可包含任意数目的嵌入变量。
   - 嵌入变量可以是所连 CPU 中的 V、M、SM、I、Q、S、T、C 或 AI 存储区。
   - 十六进制数的显示以‘16#’开头。数值中的位数取决于变量的大小。例如，VW100 显示为 16#0123。
   - 小数点左边的位数必须足够大以显示所期望的数值范围，如果是有符号整数或浮点数，还应包括正负号。
   - 如果数据格式是整数而且小数点右边的位数为零，该整数则显示为标定的整数。例如，如果 VW100＝1234，如果小数点右边有两位，该数据则显示为‘12.34’。
   - 如果数值太大无法在指定的域中显示，Modem 模块则以＃替代所有的字符位置显示该数据。
7. 在组态信息画面选择电话号码标签可组态电话号码。点击新电话号码...按钮，并添加一个新的电话号码。一旦组态了一个电话号码，则必须将它添加到项目中。选中已有的电话号码栏中的电话号码并点击向右的箭头即可将这些电话号码添加到当前的项目中。一旦你将这些电话号码添加到当前的项目中，就可以选择这些电话号码并为这些号码添加符号名以用于您的程序。

   基于用户所选择的信息的类型，电话号码可包括几个不同的域。
   - 信息协议的选择告诉 Modem 模块使用何种协议向信息服务提供商发送信息。数字呼机只支持数字协议。文字寻呼通常要求 TAP（Telelocator Alphanumeric Pratocol）。SMS 信息提供商支持 TAP 或 UCP（通用计算机协议）。通常有三种 UCP 服务用于 SMS 信息。多数提供商支持命令 1 或 51。向 SMS 提供商进行查询以确定他们所要求的协议和命令。
   - 在描述域中，你可以为电话号码添加文字描述。

- 在电话号码域中输入信息服务提供商的电话号码。对于文本信息，这个电话号码是服务提供商用于接收文本信息的 Modem 线的号码。对于数字寻呼，这个电话号码是呼机本身的号码。Modem 模块允许电话号码域中最多 40 个字符。以下字符可用于电话号码，由 Modem 模块拨出：
  0-9 电话按键

A, B, C, D, *, #	DTMF 数字（语音拨号）
,	停止拨号 2 秒钟
!	命令 Modem 产生一个挂机闪烁
@	等待 5 秒静音
W	等待拨号音
( )	忽略（可用于格式化电话号码）

- 特定的呼机 ID 域或手机号码域，这里你要输入接收信息的呼机号码或手机号码。这个号码除了数字 0 到 9 不应包含任何字符。最多可输入 20 个数字。
- 密码域是 TAP 信息的可选项。有些提供商要求一个密码，但通常可以空着。Modem 模块允许该密码最多可有 15 个字符。
- Originating 电话号码域使 Modem 模块在 SMS 信息中能够被识别。使用 UCP 命令的服务提供商要求有这样一个域输入。有些服务提供商对这个域可能还有最少字符数的要求。Modem 模块最多允许 15 个字符。
- Modem 标准域的提供是用于 Modem 模块与服务提供商的 modem 之间无法进行 modem 标准沟通的情形。缺省设置是 V.34（33.6k 波特）。
- 数据格式域可用于向服务提供商传送信息时进行数据位和校验的调整。TAP 通常使用 7 位数据位和偶校验，但是有些服务提供商使用 8 位数据位和无校验。UCP 总是 8 位数据位，无校验。查询服务提供商以决定使用何种设置。

8. 你可以组态 Modem 模块向另一个 S7-200 CPU（如果选择了 PPI 协议）或一个 Modbus 设备（如果选择了 Modus 协议）传送数据。选中使能 CPU 数据传送的多选框并点击组态 CPU 至...按钮，定义要传送的数据以及远端设备的电话号码。

9. 当设置 CPU 至 CPU 或 CPU 至 Modbus 的数据传送时，你必须定义要传送的数据以及远端设备的电话号码。在组态数据传送画面中选择数据传送标签并点击新传送按钮。数据传送包括从远端设备读数据，向远端设备写数据，或者与远端设备之间的即读又写的操作。如果选择了即读又写，先执行读后执行写。

   在每一个读或写操作中最多可写 100 个字。数据传送必须是对 CPU 的 V 存储区。向导在对远端设备的存储区进行描述时总是将远端设备当作 S7-200 CPU。如果远端设备是一个 Modbus 设备，数据传送到 Modbus 设备（地址 04xxxx）的保持寄存器。按如下结果确定相应的 Modbus 地址（xxxx）：

   Modbus 地址 = 1 + (V 区地址 / 2)

   V 区地址 = (Modbus 地址 – 1) * 2

10. 在组态 CPU 数据传送画面的电话号码标签中，你可以定义用于 CPU 到 CPU 或 CUP 到 Modbus 数据传送的电话号码。点击“新电话号码…”按钮增加新的电话号码。完成对一个电话号码的组态后必须将它加入到项目中。在可选的电话号码栏中选中相应的电话号码，然后点击向右的箭头将该号码加入到当前的项目中，一旦你将该号码加入到项目中，就可以为它增加在程序中使用的符号名。

    对描述域和电话号码域的描述与前面信息功能相同。如果远程设备是一个 Modem 模块并且使能了密码保护功能则要求填写密码域。本地 Modem 模块的该密码域中必须填写远程 Modem 模块的密码。本地 Modem 模块在远程 Modem 模块有要求时，提供这个密码。

11. 回拨功能使得 Modem 模块在收到一个来自远程 STEP7-Micro/WIN 的拨号后自动断开连接并拨出一个预定的电话号码。选择“使能回拨”复选框然后点击“组态回拨…”按钮组态回拨的电话号码。点击 Next>。

12. 在“组态回拨…”画面中，你可以电话号码，Modem 模块在应答拨入时会使用这些号码。如果回拨号码是预先确定的，可选择“使能回拨到指定的电话号码”。如果 Modem 模块接受拨号者提供的任何回拨号码（为反向计费），选择“使能对任意电话号码的回拨”。

    如果只对特定的电话号码回拨，点击“新电话号码”按钮添加回拨电话号码。回拨属性画面中可以输入预先定义的回拨号码以及相应的描述。这里输入的电话号码是 Modem 模块在执行回拨时使用的号码。这个电话号码应包括所需要的所有数字，如连接到一个外线，等待外线时暂停，接通长途等。

    在输入一个新的回拨号码后，必须将它加入到项目中。选中可选回拨电话号码，点击向右的箭头将这个号码加入到当前项目。

13. 你可以为 Modem 模块的信息传送或数据传送功能设置试拨次数。当所有拨号或信息发送的尝试都失败以后，Modem 模块会向用户程序报告错误。

    有些电话线在话筒摘机时没有拨号音。通常，当 Modem 模块命令发信息执行回拨时，若没有拨号音，模块会向用户程序返回一个错误。要想在无拨号音的线路上拨出，可选择“使能无拨号音拨号”的选项。

14. Modem 扩展向导为 Modem 模块创建组态块并要求用户输入一个起始存储区地址用来存储 Modem 模块的组态数据。Modem 模块的组态块存储在 CPU 的 V 存储区。STEP7-Micro/WIN 将该组态块写入项目的数据块中，组态块的大小则基于所组态的信息和电话号码。

    你可以选择你所希望的组态块存储的 V 区地址；或者点击“建议地址”按钮，让向导建议一个大小合适的未使用的 V 区。点击 Next>。

15. 组态 Modem 模块的最后一步是为 Modem 模块指定命令字节的 Q 区地址。你可以计算 S7-200 上 Modem 模块之前的所有模块占用的实际的输出字节，从而决定该 Q 存储区地址。点击 Next>

16. 现在，Moden 扩展向导为所选的组态生成项目组件（程序块和数据块），你可以在程序中使用这些程序代码。最后的向导屏幕显示你所要求的组态项目组件。你必须将这个 Modem 模块组态块（数据块）以及程序块下载到 S7-200 CPU。

Modem 指令和限定概述

Modem 扩展向导可以基于模块位置和你所作的组态选项生成唯一的指令子程序，从而使 Modem 模块的控制变得非常简单。每条指令都有一个前缀“MODx_”，这里的 x 是模块位置。

使用 EM241 Modem 模块指令的要求

在使用 Modem 模块指令时请考虑以下要求：

- Modem 模块指令使用三个子程序。
- Modem 模块指令会增加你的程序对存储空间的需求，最多可达 370 字节。如果你删掉了一个无用的指令子程序，你可以在需要时重新运行 Modem 扩展向导再次生成这个指令。
- 必须确保在同一时间只有一条指令是激活的。
- 这些指令不能用在中断程序中。
- Modem 模块在它第一次上电以及从 STOP 到 RUN 时读取组态表信息。除非有模式转换或再次上电，否则模块无法查觉程序中对组态表所作的任何改变。

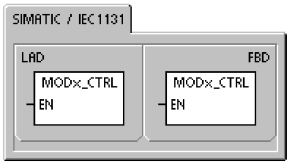
使用 EM241 Modem 模块指令

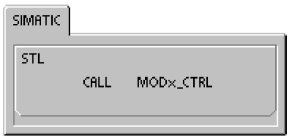
在您的 S7-200 程序中使用 Modem 模块指令请遵循以下步骤：

1. 使用 Modem 扩展向导生成 Modem 模块组态表。
2. 在程序中插入 MODx_CTRL 指令并以 SM0.0 为条件使之每一循环都执行。
3. 为每一个需要发送的信息插入一个 MODx_MSG 指令。
4. 为每一个数据传送插入一个 MODx_XFR 指令。

EM 241 Modem 模块指令

MODx_CTRL 指令





MODx_CTRL（控制）指令用于使能和初始化 Modem 模块，该指令要在每个循环周期内调用而且在项目中只能使用一次。

MODx_XFR 指令

MODx_XFR（数据传送）指令用于命令 Modem 模块读写另一个 S7-200 CPU 或 Modbus 设备的数据。该指令从 START 输入端触发到 Done 位置位需要 20 到 30 秒。

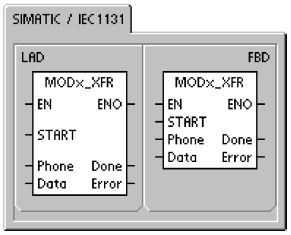
EN 位必须接通向模板发出命令，并且要保持接通直至 Done 位置位，即标志整个过程完成。当 START 输入接通且模块不忙时，每一循环向模块发送一个 XFR 命令。START 输入可以用边沿检测指令，脉冲触发，这样命令只发送一次。

Phone 是数据传送电话号码中的一个。你可以使用你在 Modem 扩展向导中为这些数据传送电话号码定义的符号名。

Data 是一个定义了的数据传送的号码。你可以使用在 Modem 扩展向导中为这个数据传送请求定义的符号名。

Done 是一个位，当 Modem 模块完成数据传送时接通。

Error 是一个字节，包含数据传送的结果。表 10-4 定义了这条指令执行时可能引起的错误条件。



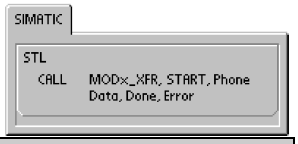
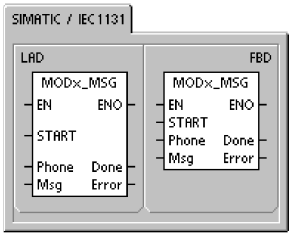
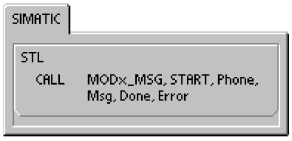


表 10-6 MODx_XFR 指令的参数

输入/输出	数据类型	操作数
START	BOOL	I、Q、M、S、SM、T、C、V、L、能流
Phone、Data	BYTE	VB、IB、QB、MB、SB、SMB、LB、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、*ID

MODx_MSG 指令





MODx_MSG（发送信息）指令用于从 Modem 模块发送寻呼或 SMS 短信息。该指令从 START 输入触发到 Done 位置位需要 20 至 30 秒。

EN 位必须接通使命令发至模块，并且保持接通至 Done 位置位，标志整个过程完成。当 START 输入接通且模块不忙时，每一循环向模块发送一个 MSG 命令。START 输入可以边沿检测脉冲触发，这样命令只发送一次。

Phone 是信息电话号码中的一个，你可以使用在 Modem 扩展向导中定义号码时您为每个信息电话号码分配的符号名。

Msg 是一个已定义的信息的号码。你可以使用在 Modem 扩展向导中为每个信息分配的符号名。

Done 是一个位，当 Modem 模块完成向服务提供商的信息发送后该位置 1。

Error 是一个字节，包含这个请求执行的结果。表 10-8 定义了这条指令的执行可能引起的错误条件。

表 10-7 MODx_MSG 指令的参数

输入/输出	数据类型	操作数
START	BOOL	I、Q、M、S、SM、T、C、V、L、能流
Phone、Data	BYTE	VB、IB、QB、MB、SB、SMB、LB、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、*ID

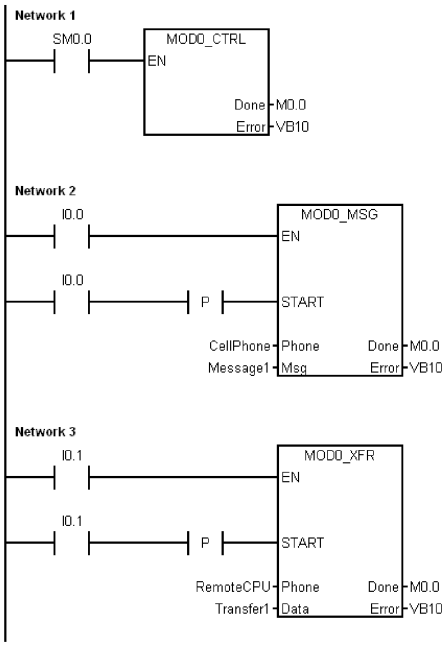
表 10 - 8 MODx_MSG 指令和 MODx_XFR 指令的错误返回值

错误	描述
0	无错误
电话线路错误	
1	无拨号音
2	线路忙
3	拨号错误
4	无应答
5	连接超时（1 分钟内未有连接）
6	连接放弃或一个未知响应
命令错误	
7	数字寻呼信息包含非法数字
8	电话号码（号码输入）超范围
9	信息或数据传送（Msg 或数据输入）超范围
10	文本信息或数据传送信息错误
11	信息或数据传送电话号码错误
12	不允许的操作（如：拨号尝试设为 0）
服务提供商错误	
13	信息服务无响应（超时）
14	信息服务由于未知原因断开
15	用户放弃信息（禁止命令位）
TAP－服务提供商返回的文本寻呼错误和 SMS 信息错误	
16	接收到远程断开（服务提供商放弃）
17	登录未被信息服务接收（不正确的密码）
18	块未被信息服务接受（校验或传送错误）
19	块未被信息服务接受（未知原因）
UCP－SMS 服务提供商返回的 SMS 信息错误	
20	未知错误
21	检验错误
22	同步错误
23	系统不支持的操作（非法命令）

错误	描述
24	该操作此时不允许
25	拨号排除激活（黑名单）
26	拨号者地址无效
27	验证失败
28	合法代码失败
29	GA 无效
30	重复不允许
31	重复合法代码失败
32	优先拨号不允许
33	优先拨号合法代码失败
34	紧急信息不允许
35	紧急信息合法代码失败
36	反向收费不允许
37	反向收费合法代码失败
UCP－服务提供商返回的 SMS 信息错误（续）	
38	延期发送不允许
39	新 AC 无效
40	新的合法代码不允许
41	标准文本无效
42	时间段无效
43	系统不支持的信息类型
44	信息太长
45	请求的标准文本无效
49	信息类型对呼机类型无效
47	SMSC 中未发现信息
48	保留
49	保留
50	子机挂机
51	不支持传真组
52	不支持传真信息类型
数据传送错误	
53	信息超时（远端设备无响应）
54	远端 CPU 正忙于上载或下载
55	访问错误（存储器超范围，非法数据类型）
56	通讯错误（未知响应）
57	响应的校验错误或 CRC 错误
58	远端 EM241 设为回拨方式（不允许）
59	远端 EM241 拒绝所提供的密码
60to127	保留
指令使用错误	
128	请求无法处理。模块可能正在处理另一个请求或该请求无启动脉冲
129	EM241 模块错误/模块故障－有关故障原因的信息请参见 EM241Modem 模块的特殊存储器（SMW220）

EM241 Modem 模块示例

子程序和子程序返回指令程序举例




```
Network 1      //在每个循环中。
               //调用 MODO_CTRL 子程序。
LD      SM0.0
CALL    MOD0_CTRL, M0.0, VB10

Network 2      //向手机发送文本信息。
LD      I0.0
EU
=       L63.7
LD      I0.0
CALL    MOD0_MSG, L63.7, CellPhone,
        Message1, M0.0, VB10

Network 3      //传送数据到远程 CPU。
LD      I0.0
EU
=       L63.7
LD      I0.1
CALL    MOD0_XFR, L63.7, RemoteCPU,
        Transfer1, M0.0, VB10
```

支持智能模块的 CPU

Modem 模块是一个智能扩展模块，它可与表 10-9 所示的 S7-200CPU 一起工作。

表 10 - 9 EM241Modem 模块与 S7-200CPU 的兼容性

CPU	描述
CPU222 Rel.1.10 或更高	CPU222 DC/DC/DC
	CPU222 AC/DC/Relay
CPU224 Rel.1.10 或更高	CPU224 DC/DC/DC
	CPU224 AC/DC/Relay
CPU226Rel.1.00 或更高	CPU226 DC/DC/DC
	CPU226 AC/DC/Relay
CPU226XM Rel.1.00 或更高	CPU226XM DC/DC/DC
	CPU226XM AC/DC/Relay

EM241 Modem 模块的特殊存储区

智能模块按其在 I/O 扩展总线上的物理位置分配有 50 个字节的特殊存储区（SM）。如果出错或检测到状态变化，模块会刷新与该模块位置相对应的 SM 区域。如果是第一个模块在需要报告状态和错误信息的时候应刷新 SMB200 到 SMB249。第二个模块则刷新 SMB250 至 SMB299 等等，见表 10-10。

表 10 - 10 特殊存储字节 SMB200 至 SMB549

特殊存储字节 SMB200 至 SMB549						
智能模块 0 号槽	智能模块 1 号槽	智能模块 2 号槽	智能模块 3 号槽	智能模块 4 号槽	智能模块 5 号槽	智能模块 6 号槽
SMB200 至 SMB249	SMB250 至 SMB299	SMB300 至 SMB349	SMB350 至 SMB399	SMB400 至 SMB449	SMB450 至 SMB499	SMB500 至 SMB549

表 10 - 11 所示为 Modem 模块的特殊存储区数据区域的分配。该区域是按照智能模块位于 I/O 系统的 0 号槽定义的。

表 10 - 11 EM241 Modem 模块的 SM 区域

地址	描述
SMB200 至 SMB215	模块名（16 个 ASCII 字符）SMB200 是第一个字符。
SMB216 至 SMB219	S/W 修订号码（4 个 ASCII 字符）SMB216 是第一个字符。
SMW220 至 SMW221	错误代码（SMB220）是错误代码的最高字节 0000 - 无错 0001 - 无用户电源 0002 - Modem 故障 0003 - 无组态块 ID 0004 - 组态块超范围 0005 - 组态错误 0006 - 国家代码选择错误 0007 - 电话号码太大 0008 - 信息太大 0009 to 00FF - 保留 01xx - 回拨号码 XX 出错 02xx - 呼机号码 XX 出错 03xx - 信息号码 X 0400 to FFFF - 保留
SMB222	模块状态－反映 LED 状态 MSB 7 6 5 4 3 2 1 0 LSB F G H T R C 0 0 F – EM_FAULT 0 – 无错误 1 – 出错 G – EM_GOODQ 0 – 不好 1 – 好 H – OFF_HOOK 0 – 挂机 1 – 摘机 T – NO DIALTONE 0 – 语音拨号 1 – 无拨号音 R – RING 0 – 无振铃 1 – 电话振铃 C – CONNECT 0 – 未连接 1 – 连通
SMB223	由开关设置的国家代码（十进制值）
SMB224 至 SMB225	建立连接所使用的波特率（无符号十进制值）。 SMB224 是高位字节，SMB225 是低位字节

地址	描述
SMB226	用户命令的结果 MSB 7 6 5 LSB 0 D | 0 | ERROR D – Done 位； 0 – 操作进行中 1 – 操作完成 ERROR：错误代码描述，见表 10-8
SMB227	电话号码选择－该字节指定发送信息时使用的信息电话号码
SMB228	信息选择－该字节指定发送哪条信息。有效值为 1 至 250。
SMB229 至 SMB243	保留
SMB244	保留
SMB245	与第一个 Q 字节之间的偏移量，用作该模块的命令接口，该偏移量由 CPU 提供，是为方便用户而非模块所需。
SMB246 至 SMB249	在 V 存储区中指向 EM241Modem 模块组态表的指针。SMB246 是最高字节，SMB249 是最低字节。指向 V 区以外其它区域的指针是不能被接受的，模块会持续地检查该区域，等待有效的指针值。

高级议题

理解组态表

Modem 扩展向导能够基于您关于您的系统的回答自动生成组态表，轻松地完成 Modem 的应用。组态表信息提供给那些想要生成自己的 Modem 模块控制程序，格式化自己的信息的高级用户。

组态表位于 S7-200 的 V 存储区。表 10-12 的字节偏移量这一栏，是与 SM 中的组态区域指针所指向的位置的偏移量。组态表信息分为四个部分。

- 组态块包含组态该模块的信息。
- 回拨电话号码块包含预定的电话号码可用于回拨安全功能。
- 信息电话号码块包含用于拨号信息服务或 CPU 数据传送的电话号码。
- 信息块包含预定的信息服务中要发送的信息。

表 10 - 12 EM241 组态表

组态块	
字节偏移量	**描述**
0to4	模块标识－5 个 ASCII 字符用于联系组态表和智能模块。 版本 1.00 的 EM241Modem 模块的标识是“M241A”
5	组态块的长度－当前为 24
6	回拨电话号码长度－有效值为 0-40
7	信息电话号码长度－有效值为 0-120
8	回拨电话号码的号码－有效值为 0-250
9	信息电话号码的号码－有效值为 0-250
10	信息号码－有效值为 0-250
11to12	保留（2 字节）
13	该字节包含所支持的特性的使能位 MSB 7 6 5 4 3 2 1 0 LSB PD CB PW MB BD 0 0 0 PD - 0=语音拨号 1 = 脉冲拨号 CB - 0=回拨禁止 1 = 回拨使能 PW - 0=密码禁止 1 = 密码使能 MB - 0=PPI 协议使能 1 = Modbus 协议使能 BD - 0=盲拨禁止 1 = 盲拨使能 模块忽略位 2，1，0。
14	保留
15	尝试次数－该值指定 Modem 模块在返回错误之前尝试拨号与并发送信息的次数。数值 0 则禁止 Modem 拨号。
16to23	密码－8 个 ASCII 字符
回拨电话号码块（可选）	
字节偏移量	**描述**
24	回拨电话号码 1－一个字符串，代表第一个授权以 EM241Modem 模块回拨访问的电话号码。按照回拨电话号码长度域中指定的长度每个回拨号码分配有相同的空间。（组态块中偏移量为 6）
24＋回拨号码	回拨电话号码 2
:	:
:	回拨电话号码 n
信息电话号码块（可选）	
字节偏移量	**描述**
M	信息电话号码 1－一个字符串，代表信息电话号码，包括协议和拨号选项。按照信息电话号码长度域中指定的长度。每个电话号码配有相同的空间。（组态块中的偏移量是 7） 对于信息电话号码格式的描述在下面。

M+ 信息号码长度	信息电话号码 2
:	:
:	信息电话号码 n
信息块（可选）	
字节偏移量	**描述**
N	第一个信息的 V 区偏移量（相对于 VB0）（2 字节）
N+2	信息 1 的长度
N+3	信息 2 的长度
:	:
:	信息 n 的长度
P	信息 1—一个字符串，最大（120 字节），代表第一个信息。该字符串包括文本和嵌入数据的规范，或者指定一个 CPU 的数据传送。 请参见下面有关文本信息格式和 CPU 数据传送格式的描述。
P+信息 1 的长度	信息 2
:	:
:	信息 n

Modem 模块在以下事件出现时会重新读取组态表：

- S7-200CPU 从 STOP 转为 RUN 的 5 秒钟之内（除非 modem 在线）
- 每 5 秒钟一次除非找到一个有效的组态（除非 modem 在线）
- 每次 modem 从在线转为离线。

信息电话号码格式

信息电话号码的结构中包含 Modem 模块发送信息所需要的信息。信息电话号码是一个字符串，第一个字节是长度，其后是 ASCII 字符。信息电话号码的最大长度是 120 字节（包括长度字节）。

信息电话号码包含由斜杠（/）隔开的 6 个区域。双斜杠之间指示空区域，在 Modem 模块中空区域改为缺省值。

格式：＜电话号码＞/＜标识＞/＜密码＞/＜协议＞/＜标准＞/＜格式＞

电话号码域是 Modem 模块发送信息时拨打的电话号码。如果要发送的信息是一个文本或 SMS 短信息，这个号码是服务提供商的电话号码。如果信息是数字寻呼，该区域是寻呼机的电话号码。如果信息是一个 CPU 的数据传送，该号码是远端设备的电话号码，该区域的最大字符数是 40。

标识：是寻呼机号或手机号。这个区域只能是数字 0 至 9。如果协议是 CPU 数据传送，该区域用来提供远端设备的地址。该区域最多可有 20 个字符。

密码区域：通过 TAP 发送信息时，如果服务提供商要求密码，则该区域提供密码。对于通过 UCP 发送信息，该区域用作原始地址或电话号码。如果信息是向另一个 Modem 模块的 CPU 数据传送，该区域可用作提供远端 Modem 模块的密码。该密码最大长度为 15 个字符。

协议域：包含一个 ASCII 字符，指示 Modem 模块该如何格式化和传送信息。它可以是以下各值：

1－数字寻呼协议（缺省）
2－TAP
3－UCP 命令 1
4－UCP 命令 30
5－UCP 命令 51
6－UCP 数据传送

标准域：强制 Modem 模块使用一个特定的 Modem 标准。该标准域是一个 ASCII 字符。可以是以下各值：

1-Bell103
2-Bell212
3-V.21
4-V.22
5-V.22bit
6-V.23c
7-V.32
8-V.32bit
9-V.34（缺省）

格式域：是三个 ASCII 字符，指定信息传送时使用的数据位和校验。如果协议设为数字寻呼，该区域无效。该区域只允许以下两种设置：

8N1-8 位数据位，无校验，一位停止位（缺省）

7E1-7 位数据位，偶校验，一位停止位

文本信息格式

文本信息格式定义了文本寻呼或 SMS 短信息的格式。这些信息格式包含文本和嵌入变量。文本信息是一个字符串，以长度字节开始，其后为 ASCII 字符。文本信息最长 120 字节（包括长度字节）。

格式：＜文本＞＜变量＞＜文本＞＜变量＞…

文本域包含 ASCII 字符。

变量域定义了一个 Modem 模块从本地 CPU 中读取的嵌入数据，其格式以及它在信息中的位置。百分号（％）用来标识变量域的开始和结束。地址和左边区域用冒号隔开。左、右区域之间的小数点可以是逗号也可以是点号。变量域格式如下：

％地址：左.右格式％

地址域指定该嵌入数据的地址、数据类型和长度（如：VD100、VW50、MB20 或 T10）。可以是以下数据类型：I、Q、M、S、SM、V、T、C 和 AI。长度可以是字节、字和双字。

左区域指定小数点左边所显示的位数。该数值应足够大以显示包括负号在内的完整的嵌入数据。如果左区域为零，则该值前面显示零。左区域的有效值为 0 到 10。

右区域定义小数点右边显示的位数。小数点右边的零总能够被显示，如果右区域的值为零，则该数值的显示不带小数点。右区域的有效值为 0 至 10。

格式域指定嵌入数据的显示格式。可以是以下字符：

i－有符号整数

u－无符号整数

h－十六进制数

f－浮点数/实数

示例：“Temperature=%VW100:3.1I% Pressure=%VD200:4.3f%”。

CPU 数据传送信息格式

无论是 CPU 到 CPU 或是 CPU 到 Modbus 的数据传送，其数据传送的格式都是由 CPU 数据传送信息格式来定义。CPU 数据传送信息是一个 ASCII 字符串。可以定义两台设备之间的任意数量的数据传送，其定义的最大数量不能超过信息长度的上限 120 字节（119 字节的字符加上一个长度字节）。ASCII 码的空格可用来分隔不同的数据传送，但不是必须的。所有的数据传送都在一个链接中执行。数据传送按照在信息中定义的顺序进行。如果在数据传送中检测到错误，与远端设备的连接断开而且随后的传送不再处理。

如果是读操作，从远端设备的“远端地址”开始读“数量”所指定的字符数，然后把这些数据写到本地 CPU 从“本地__地址”开始的 V 区中。

如果是写操作，从本地 CPU 的“本地__地址”开始读取“数量”所指定的字符数，然后将这些数据写到远端设备从“远端__地址”开始的区域。

格式：＜操作＞＝＜数量＞，＜本地__地址＞，＜远端__地址＞

操作域中是一个 ASCII 字符，指定传送类型。

R－从远端设备读数据

W－向远端设备写数据

数量域指定要传送的字符数。该区域的有效值为 1 到 100 字。

本地__地址指定本地 CPU 中数据传送的 V 区地址（如：VW100）。

远端__地址指定远端设备中的数据传送地址（如：VW500）。

即使是与 Modbus 设备传送数据，这个地址总是指定为 V 区。如果远端设备是一个 Modbus 设备，则 V 区地址与 Modbus 地址之间的对应关系如下：

Modbus 地址＝1＋（V 区地址/2）

V 区地址＝（Modbus 地址－1）*2

示例：R＝20，VW100，VW200 W＝50，VW500，VW1000 R＝100，VW1000，VW2000

CP 243-1 工业以太网通讯处理器

11

本章内容：

简介

定义和应用

CP 243-1 是一种通讯处理器，设计用于在 S7-200 自动化系统中运行。它可用于将 S7-200 系统连接到工业以太网（IE）中。CP 243-1 通讯处理器还可用于实现 S7 低端性能产品的以太网通讯。因此，可以使用 STEP 7 Micro/WIN 32，对 S7-200 进行远程组态、编程和诊断。而且，一台 S7-200 还可通过以太网与其它 S7-200、S7-300 或 S7-400 控制器进行通讯。并可与基于 OPC 的服务器进行通讯。在开放式 SIMATIC NET 通讯系统中，工业以太网可以用作协调级和单元级网络。在技术上，工业以太网是一种基于屏蔽同轴电缆、双绞电缆而建立的电气网络，或一种基于光纤电缆的光网络。工业以太网根据国际标准 IEEE 802.3 定义。

工业领域中的连续通讯

工业以太网是 SIMATIC NET 概念的一个部分，与现场总线 PROFIBUS 和 AS 接口一起，可以确保协调级、单元级和现场级的不间断网络运行。

兼容性

CP 243-1（订货号：6GK7 243-1EX00-0XE0）通讯处理器可以用于S7通讯。CP 243-1可以与各种不同类型的S7-200 CPU（222、224、226和226XM）相连接：

- CPU 222 Rel.1.10 或以上
- CPU 224 Rel.1.10 或以上
- CPU 226 Rel.1.00 或以上
- CPU 226XM Rel.1.00 或以上

在 CPU 222 上最多可以安装 2 个扩展模块。而在 224、226 和 226XM CPU 上最多可以安装 7 个扩展模块。

注意

每个 S7-200 CPU 只能连接一个 CP 243-1。如果连接有多个 CP 243-1，将不能保证 S7-200 系统的正常运行。

CP 243-1软件符合以下标准：

- S7 XPUT/XGET 和 S7 READ/WRITE
- S7-200 I/O 总线规范

规划

可以使用 STEP 7 Micro/WIN 32，版本 3.2.1 或以上，对 CP 243-1 通讯处理器进行组态。CP 243-1 使用固定的 MAC 地址供货。IP 地址和子网掩码也必须进行组态，或通过 BOOTP 协议从 BOOTP 服务器上检索。对于监控连接（“Keep Alive（持续作用）”），还必须组态所有 TCP 与主动和被动通讯伙伴的传输连接时间。一次可以保持 8 个与其它控制器的连接。

编程

在用户程序中进行通讯编程时，应使用 STEP 7 Micro/WIN 32 中的“Ethernet Wizard（以太网向导）”，详见第 4 章和第 5 章。

组态

CP 243-1 固件在生产时已被编程在闪存中，并在其中永久保存。

在 CP 243-1 运行时所生成的系统条件或动态不定内容，在失电时会丢失。

可以使用 STEP 7 Micro/WIN 32，版本 3.2.1 或以上，对 CP 243-1 通讯处理器进行组态，组态结果可以作为非易失性数据保存在 S7-200 CPU 的数据块（DB）中。在引导装入时，CP 243-1 可以从 CPU 中读取组态，并相应地进行初始化。

特性和功能

概述

CP 243-1 具有以下功能：

- S7 通讯
  - 可对通过工业以太网的数据通讯进行预先格式化。基于标准TCP/IP协议进行通讯。
  - 可通过RJ45进行以太网访问
  - 通过S7-200底板总线，即可与S7-200系统简单连接
  - 可以实现一种灵活的分布式自动化架构
  - 通过工业以太网和STEP 7 Micro/WIN 32，实现S7-200系统的远程编程、组态和诊断。
  - 为简化过程数据的进一步处理和归档打下基础
  - 可同时与最多8个S7控制器通讯
  - 可提供与S7-OPC的连接
  - 简化网络管理
  - 无需重复进行编程/组态，即可更换模板（即插即用）
  - S7通讯服务，“XPUT/XGET”，既可作为客户机，也可作服务器
  - S7通讯服务，“READ/WRITE”，作为服务器
- 看门狗定时器
  CP 243-1 中安装有一个看门狗电路。每次 CP 243-1 启动时，看门狗也启动。一般地，看门狗的监控时间为 5 秒。鉴于组件相关误差，该时间可以增加到 7 秒。如果设定了看门狗监控时间，CP 243-1 可以自动置位。这会重新启动 CP 243-1。同时，CP 243-1 会向 S7-200 CPU 报告“Parity Error（奇偶校验出错）”。对于这类错误的处理，详见《STEP 7 Micro/WIN 32》手册。
- 通过预设 MAC 地址（48 位数值），进行地址分配。
  在出厂时已对每个 CP 243-1 进行了 MAC 地址分配。MAC 地址打印在附于上盖下面的标签上。

使用 BOOTP 协议，通过预设的 MAC 地址，可以将 IP 地址分配给 CP 243-1 通讯处理器。

在工业以太网中的 S7 通讯

应用

在工业以太网中进行 S7 通讯，可以使通过 SFB/FB 通讯和组态的 S7 连接进行程控通讯成可能。使用 XPUT/XGET 和 READ/WRITE 服务，CP 243-1 可以通过工业以太网支持 S7 通讯。一般地，每个命令可以最多传送 212 个字节的用户数据。但是，如果 CP 243-1 作为服务器运行，每个读操作可以最多传送 222 个字节。

CP 243-1 可支持一个或多个远程通讯伙伴的最多 8 个通讯通道。CP 243-1 可以根据客户机/服务器原理在每个通道运行。每个通道，每次只能接收、处理或响应（主动响应或被动响应）一个命令。只有在发送响应后，CP 243-1 通讯处理器才能接受其它命令。

如果 CP 243-1 在组态为服务器使用时，在一个通道接收到了多个命令，只有第一个命令得以处理，其它命令将被忽略，直到事项处理完成，即已发送一个响应后，才被处理。CP 243-1 没有通道命令管理功能，因此不能对命令进行缓存。

与 PC/编程器进行通讯的先决条件

和以前一样，S7-200 CPU 仍可通过 PPI 接口由编程器/PC 来访问。另外，也可以通过以太网使用 CP 243-1 访问。在使用 CP 243-1 访问时，必须具备以下先决条件：

- 在编程器/PC中已插入一张以太网卡，并已组态，有一个工业以太网或TCP/IP连接至CP 243-1（可以通过路由器、防火墙等）。
- 在PC/编程器中已安装有STEP 7 Micro/WIN 32，版本3.2.1或以上。
- CP 243-1已分配有一个有效的IP地址。该地址可以在组态中进行定义，或通过BOOTP协议从一个BOOTP服务器中进行检索。

此时，只能有一个 STEP 7 Micro/WIN 32 可以通过 CP 243-1 与 S7-200 CPU 进行通讯。

通讯类型

CP 243-1 提供有三种通讯关系，可以单独使用，也可以组合使用。

1. 连接 STEP 7 Micro/WIN 32
2. 连接其它 SIMATIC S7 系列远程组件
3. 连接基于 OPC 的 PC/PG 应用程序

通讯伙伴

- S7-200 CPU 与 CP 243-1
- S7-300 CPU 与 CP 343-1 或 CP 343-1IT
- S7-400 CPU 与 CP 443-1 或 CP 443-1IT
- 编程器/PC 与 OPC 服务器
- 编程器/PC 与 STEP 7 Micro/WIN 32

在 STEP 7 HW-Config 程序中，你可以识别哪种类型的 S7-300 CPU 和 S7-400 CPU 支持 S7 协议以及 XPUT/XGET 功能，哪种类型的控制器可与 CP 243-1 进行通讯。如果你在程序中的目录窗口中选择一个 S7-300 或 S7-400 CPU，所选中的 CPU 必须支持“S7 通讯”功能。

对于 S7-300 系列系统，XPUT/XGET 只能通过以太网使用版本 1.1 或以上的通讯处理器运行。通过查看 MLFB 号，你可以识别通讯处理器的版本。如果你使用的是 CP 343-1 通讯处理器，MLFB 号必须包含序列“EX11”。

由于 CP 443-1 ISO 底板上没有 TCP/IP 和 RFC 1006，CP 443-1 ISO 不能与 CP 243-1 进行通讯。

注意

每个 S7-200 CPU 只能连接一个 CP 243-1。如果连接有多个 CP 243-1，将不能保证 S7-200 系统的正常运行。

注意

当与一个 OPC 服务器进行通讯时，应注意，CP 243-1 不支持 S7-200 中的目标自动查询服务（例如 DBxx...）。

概述：

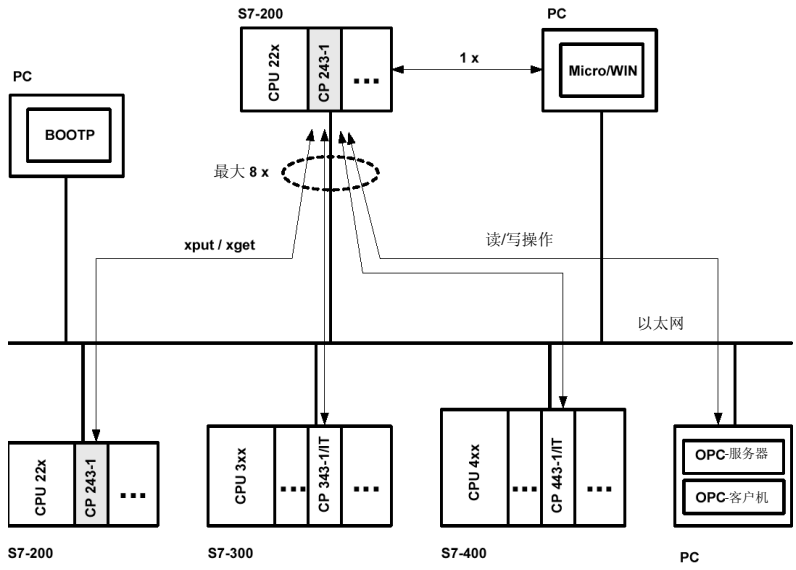


图 11-1　系统概述

你可以使选配有 CP 243-1 的 CPU 22x 与其它 S7-200、S7-300 和 S7-400 系统以及 OPC 服务器进行通讯。

在这种情况下，除了 STEP 7 Micro/WIN 连接以外，还可以提供最多 8 种连接（见图 11-1）。

S7 站连接的组态和编程

为了组态 S7-200 和 S7-300、S7-400 或 OPC 服务器之间的通讯，你可以使用 STEP 7 Micro/WIN 32（版本 3.2.1 或以上）以及 STEP 7（版本 5.1 或以上，带有 Service Pack 3）（见第 4.5 节）。

组态和编程 S7-200 站，可以使用 STEP 7 Micro/WIN 32，而 S7-300、S7-400 或 OPC 服务器的组态和编程，则需要基于以太网的 STEP 7 和 NCM。

通过工业以太网进行数据交换

通过 CP 243-1 的数据交换基于以太网设计，因此不能保证数据交换的确定性，即不能保证响应时间。不管是全双工模式，还是半双工模式，都可提供 10 和 100 Mbit 的传输速率网络支持。而且，CP 243-1 支持“自动匹配”功能，用于所使用模式和传输速率的自动匹配。模式和传输速率也可以在组态 CP 243-1 时由用户进行定义。如果 CP 243-1 的组态无效，将缺省使用“自动匹配”模式。

注意

自动匹配模式只能在所有所连接的网络组件都支持该模式时才能运行。

工业以太网和 TCP/IP 都不支持时间驱动的数据流。它不能预测一个远程 CPU 何时执行所请求的命令。来自远程 CPU 的响应也与本地 CPU 的 CPU 循环不同步。因此，对于具有时间相关要求的分布式应用，基于 TCP/IP 的通讯应用有限（例如控制回路，周期采样等）。

S7 通讯

S7 服务、XPUT 和 XGET 功能都可用于两个控制器之间的数据交换。此时，CP 243-1 既可作为一个客户机使用，也可以作为一个服务器使用。

CP 243-1 和运行在 PC/编程器上的 OPC 服务器之间的通讯，都基于 S7 服务、READ 和 WRITE 功能。在这种情况下，CP 243-1 总是作为服务器使用。其它 S7 服务，例如 S7-200（DB，...）中的目标自动查询服务，将不被支持。CP 243-1 可以支持以下数据类型或数据区：

CP 243-1 作为客户机：

读和写访问：

- 数据类型总为 BYTE（字节）
- 只能在本地系统上访问变量。
- 当 S7-200 作为一个通讯伙伴时，可在通讯伙伴系统上访问的存储区包括输入、输出、标志和变量区域。
- 对于 S7-300 或 S7-400，可在通讯伙伴系统上访问的存储区包括输入、输出、标志和数据区域。

CP 243-1 作为服务器：

写访问：

- 数据类型包括 BOOL（布尔型）、BYTE（字节）、WORD（字）或 DWORD（双字）

- 本地系统上的可访问存储区包括输入、输出、变量、标志和状态位区域。

读访问：

- 数据类型包括 BOOL（布尔型）、BYTE（字节）、WORD（字）或 DWORD（双字）
- 本地系统上的可访问存储区包括输入、输出、变量、标志、系统区和状态位区域。

与 STEP 7 Micro/WIN 32 的通讯

对于 CP 243-1 和 STEP 7 Micro/WIN 32 之间的通讯，CP 243-1 总是作为服务器使用。在这种情况下，STEP 7 Micro/WIN 32 也总是作为客户机使用。

I/O 总线通讯

所有 S7-200 CPU 的数据区都可以进行访问。读和写访问与 CPU 是处于“RUN（运行）”状态，还是处于“TERM（中止）”或“STOP（停止）”状态无关。

安全性

组态

CP 243-1 的组态将作为非易失性数据保存在 S7-200 CPU 中。使用 CRC（循环冗余校验）功能，可以保证组态的有效性。

在保存 CP 243-1 的组态时，STEP 7 Micro/WIN 32 会计算 CRC（循环冗余校验）检查和。该检查和将与组态一起保存。当 CP 243-1 读取组态时，它将检查该检查和，并以此识别所保存组态数据的异常变化。

也可以禁用这种 CRC 功能。然后，手工修改组态，或从 S7-200 用户程序中进行修改。

注意

就 CRC（循环冗余校验）校验关闭后的有意和无意修改方面，CP 243-1 不能完全检查组态数据的一致性。因此在这种情况下，就不能保证连接在网络中的 CP 或组件能够正确运行。

注意

当其组态中的某个字节设定为一个特定的数值时，CP 243-1 可以识别 CRC 是否被禁用。如果在组态中明确地设定了该数值，不管是有意还是无意，CRC 校验都将被禁用。因此，强烈建议使用集成在 STEP 7 Micro/WIN 32 中的 Ethernet Wizard（以太网向导）生成组态，检查 S7-200 程序，以检查存储操作是否在保存 CP 243-1 组态数据的区域中运行。

数据安全性

CP 243-1 是以太网和 S7-200 I/O 总线之间的一个物理连接。因此：

- CP 243-1 不能防止数据区和/或本地或远程 CPU 系统状态的有意或无意修改
- CP 243-1 不具有防火墙功能

因此，我们建议如果在本地局域网中使用 CP 243-1 时，必须采取相应的安全措施，将本地局域与公用网络隔离开来。

如果已有60秒中没有STEP 7 Micro/WIN命令发送给CPU，CP 243-1将会终止当前的STEP 7 Micro/WIN 32 连接。这可防止由于网络故障而造成 CP 243-1 中的 Micro/WIN 服务器被锁死，以及与 STEP 7 Micro/WIN 的连接受妨碍。

注意

CP 243-1 允许服务器访问 S7-200 CPU，不管 CPU 处于“RUN（运行）”模式，还是处于“STOP（停止）”模式。但是，在“STOP（停止）”模式下，不能更新程序变量或外围数值。

通讯的完整性

CP 243-1 具有符号生命周期监控功能（“Keep Alive”功能）。借助于这种功能，CP 243-1 可以在一个组态的时间段内，自动检测通讯伙伴或相关连接是否故障。

在组态 CP 243-1 时所规定的“Keep Alive（持续作用）”时间，是指内部功能到达通讯伙伴后的时间。这些功能的处理大概需要 10 秒钟的时间。如果在该时间段内不能到达通讯伙伴，CP243-1 将自动断开与该通讯伙伴的连接。如果 CP 243-1 作为一个客户机使用，它将尝试重新建立这种连接。根据第 6 章所述原理，通讯伙伴所出现的故障将通知给用户。

一般地，在进行这种通讯以及具有这种功能的所有系统中，都应激活“Keep Alive（持续作用）”功能。

注意

根据 RFC1122 和 RFC793，通讯伙伴也必须支持“Keep-Alive”这种功能。

联接

前视图：

图 11-2　连接

CP 243-1具有以下连接：

- 用于 24 V DC 电压和接地连接的接线板
- 用于以太网连接的 8 针 RJ45 插座
- I/O 总线插入式连接器
- 带有套接口的 I/O 总线集成扁平电缆

连接位于前门盖的下方。

显示：正面 LED

图 11-3　前面 LED 显示

在前面共有 5 个 LED，用以显示：

LED 显示	颜色	含义
SF	红色，连续点亮	系统错误： 在出现错误时点亮
	红色，闪亮	系统错误： 如果组态错误，并且没有找到 BOOTP 服务器，将闪亮（每秒钟一次）。
LINK	绿色，连续点亮	通过 RJ45 接口连接：已建立以太网连接
RX/TX	绿色，闪烁	以太网活动： 数据正在通过以太网进行接收和传输 **注意：** 通过以太网接收的数据包不一定用于 CP 243-1。CP 243-1 将接受每一个通过以太网传送的数据包。然后，在决定数据包是否对它有用。 如果以太网电缆还没有断开，只要 CP 243-1 一尝试发送一个数据包，RX/TX 指示灯也闪亮。
RUN	绿色，连续点亮	运行： CP 243-1 已通讯准备就绪
CFG	黄色，连续点亮	组态： 在 STEP 7 Micro/WIN 32 通过 CP 243-1 与 S7-200 CPU 保持连接时点亮。

表 11-1：LED 显示屏上的功能

在 CP243-1 引导装入时，SF 指示灯将闪烁两下。然后，LINK 指示灯和 RX/TX 指示灯闪烁几下。只要 RUN 指示灯一点亮，CP 243-1 就完成引导装入程序。

安装和调试

安装

S7-200 系列装置可以安装在控制面板中，也可以安装在 DIN 导轨中。模板既可以水平安装，也可以垂直安装。S7-200 CPU 和扩展模板都为自然对流散热设计。因此，在装置的上方和下方应留有 25 mm 的空间，以确保充分散热。如果在最大环境温度和最大负荷下长期运行，会缩短电子组件的使用寿命。

注意

CP 243-1在S7-200系统中的运行位置，取决于S7-200 CPU的固件版本。

如果使用版本 1.2 或以上的固件，CP 243-1 可以安装在 S7-200 系统中的任意位置。对于版本 1.2 以下的固件，CP 243-1 必须直接安装在 S7-200 CPU 的附近。

布线

警告

如果在设备通电时，你尝试安装或拆除 CP 243-1 或其它装置，会有电击危险，以及造成设备不能正常运行。

如果在 CP 243-1 以及所有相连装置的电源还没有断开时，安装或拆除装置，会造成人身伤害和/或设备损坏。

必须采取必要的安全预防措施，以保证在进行系统布线之前，S7-200 和 CP 243-1 的电源已断开。

总则

在进行自动化系统布线时，必须遵守以下总则：

- 应确保根据所有适用标准和相关标准对 CP 243-1 进行接线。在安装和操作装置时，应遵守相应的国家和地方规定。在实际应用中所必须遵守的标准和规定，可以向当地的主管部门咨询。
- 只能在断电状态下对 S7-200 CPU 和 CP 243-1 进行接线！
- 应根据各种电流情况，使用相应截面积的电缆。CP 243-1 的 24 V 电源电缆可以使用截面积在 0.50 mm^2-1.50 mm^2之间的电缆。对于接地端子的接线，可以使用直径为 1.50 mm^2 的电缆。
- 禁止过分拧紧连接端子。最大允许扭矩为 0.56 Nm。
- 应尽可能地短距离敷设电缆。电缆应成对敷设：中线与相线或信号电缆一起敷设。
- 应将交流线路以及开关顺序迅速的高压直流线路与低压信号线路隔离开来。
- 应对具有雷击危险的电缆采取过电压保护。
- S7-200 CPU 和 CP 243-1 应连接至同一电源！

- CP 243-1 配装有一个带有连接器套接口的集成扁平电缆，可用于与其它 S7-200 组件快速连接。
- 所用 S7-200 CPU 的固件版本要求可以运行插槽式 CP 243-1（见第 20 页的注意事项）。
- 每个 CPU 最多可以使用一个 CP 243-1。

电气要求

输入电压必须总为24 V DC。

只能使用与 120/230 V AC 电源和类似危险可靠电气隔离的 24 V DC 电源。例如，可以使用符合以下标准的可靠电气隔离：

- 符合标准EN60204-1的PELV（保安特低电压）
- Class 2或电压/电流符合UL 508标准的电路。S7-200底板总线上的电压由各自的S7-200 CPU供电。

确保 CP 243-1 正确接地。

安装空间要求

在安装模板时，应遵守以下指南：

- CP 243-1 为自然对流散热设计。因此，在装置的上方和下方应留有 25 mm 的空间，以确保充分散热。如果在最大环境温度和最大负荷下长期运行，会缩短电子组件的使用寿命。
- 如果是水平安装，CP 243-1 必须安装在 CPU 的右边。
- 如果是垂直安装，最大允许环境温度为 10℃。CP 243-1 必须布置在 CPU 的上方。如果使用的是垂直标准 DIN 导轨，你应使用标准 DIN 导轨锁挡，以防止模板移动。
- 安装深度应为 75 mm。

注意

在进行设备选型时，应注意应有足够的空间，用于输入和输出布线以及通讯电缆连接。

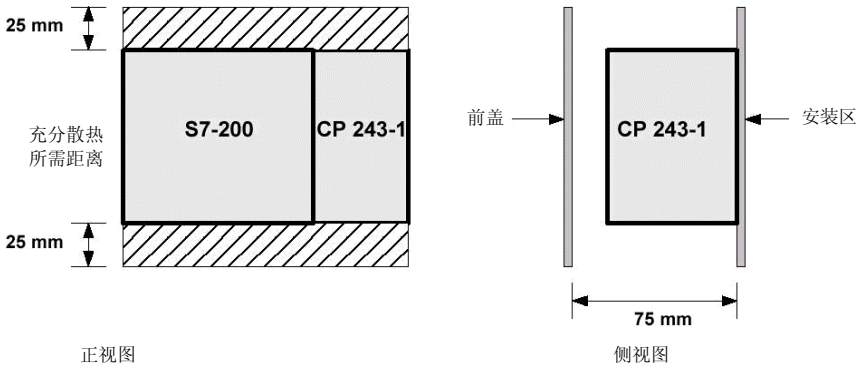


图 11-4　安装空间要求

在控制面板中的安装尺寸

CP 243-1上提供有安装孔，易于在控制面板中安装。

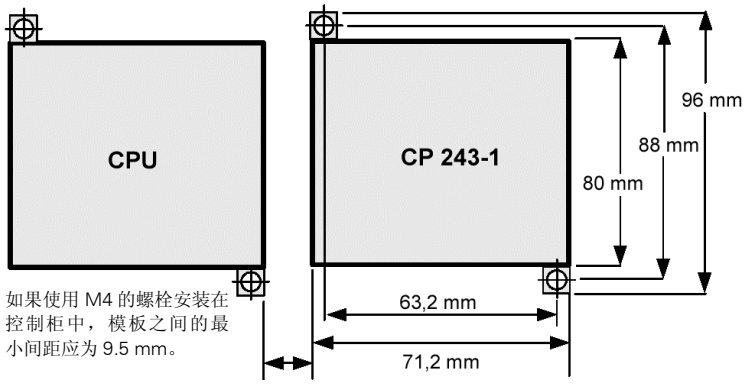


图 11-5　在控制面板中的安装尺寸

在 DIN 导轨中的安装尺寸

CP 243-1也可以安装在导轨上（DIN EN 50 022导轨）。

下图所示为标准DIN导轨尺寸：

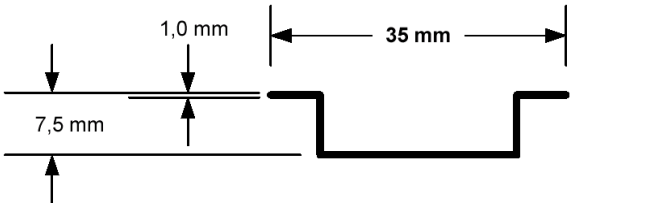


图 11-6　在 DIN 导轨中的安装尺寸

在控制面板中的安装

安装步骤

1. 准备控制面板，并备有可以使用DIN M4螺栓进行安装的安装孔。根据第3章和第3.1节所述的注意事项以及安装尺寸，在控制面板中进行安装。
2. 将CP 243-1用螺栓紧固在控制面板上。如果是水平安装，应在CPU的右侧，如果是垂直安装，应在CPU的上方。应使用DIN M4规格的螺栓。
3. 将CP 243-1的扁平电缆插入相邻模板或S7-200 CPU前盖下方的连接器中。插头的形状应易于连接。
4. 连接接地端子：

   将CP 243-1的接地端子连接到最近的接地处，以达到最大抗干扰性。建议分别连接所有接地端子。应使用截面积为1.5 mm^2的电缆。
5. 连接电源。

6. 连接以太网电缆。

   传输线路为一根2 x 2屏蔽双绞线电缆，阻抗为100Ω。该电缆的传输特性应符合5类电缆（Cat5电缆）的技术要求。对于IEEE802.3组件，端子装置和网络组件（链路段）之间的最大连接长度应在100 m以内。

   由于CP 243-1中的RJ45连接器也被屏蔽，如果使用屏幕的以太网电缆，可以形成连续的电缆屏蔽，确保以太网传输无干扰。RJ45连接器的屏蔽将被连接到CP 243-1的接地端子。

   建议应检查一下屏蔽是否在传输线路的两端已经良好接地。否则，如果使用未屏蔽的电缆或屏蔽一端没有接地或接地不良，就不能保证技术数据符合抗电磁辐射和干扰（CE标志）要求。对此，系统操作者应负责这一要求。

至此设备安装即告完成。

注意

CP 243-1 的前门在操作过程中必须保持关闭。

在安装装置时，应保证模板的上下空气管道不会阻塞，空气流通畅通。

在标准 DIN 导轨中的安装

安装步骤

1. 打开锁扣，将 CP 243-1 安装在 CPU 右侧或上方的 DIN 导轨中。
2. 锁定锁扣，将 CP 243-1 紧固在导轨上。确保锁扣已正确锁紧，使装置牢固在导轨上。

注意

对于易受强烈振动的场合或垂直安装时，应使用标准导轨锁挡，防止装置在导轨上滑动。

3. 将 CP 243-1 的扁平电缆插入相邻模板或 S7-200 CPU 前盖下方的连接器中。插头的形状应易于连接。
4. 连接接地端子：
   将 CP 243-1 的接地端子连接到最近的接地处，以达到最大抗干扰性。建议分别连接所有接地端子。应使用截面积为 1.5 mm^2 的电缆。
5. 连接电源。
6. 连接以太网电缆。

   传输线路为一根2 x 2屏蔽双绞线电缆，阻抗为100Ω。该电缆的传输特性应符合5类电缆（Cat5电缆）的技术要求。对于IEEE802.3组件，端子装置和网络组件（链路段）之间的最大连接长度应在100 m以内。

   由于CP 243-1中的RJ45连接器也被屏蔽，如果使用屏蔽的以太网电缆，可以形成连续的电缆屏蔽，确保以太网传输无干扰。RJ45连接器的屏蔽将被连接到CP 243-1的接地端子。

建议应检查一下屏蔽是否在传输线路的两端已经良好接地。否则，如果使用未屏蔽的电缆或屏蔽一端没有接地或接地不良，就不能保证技术数据符合抗电磁辐射和干扰要求。对此，系统操作者应负责这一要求。

现在设备安装即告完成。

注意

CP 243-1 的前门在操作过程中必须保持关闭。

在安装装置时，应保证模板的上下空气管道不会阻塞，空气流通畅通。

CP 243-1 的更换

如果更换 CP 243-1（订货号：6GK7 243-1EX00-0XE0），由于组态数据和用户程序已作为非易失性数据，保存在 S7-200 CPU 中，不需要进行重新编程。

CP 243-1 的拆卸

警告

如果在设备通电时，你尝试安装或拆除 CP 243-1 或其它装置，会有电击危险，以及造成设备不能正常运行。

如果在 CP 243-1 以及所有相连装置的电源还没有断开时，安装或拆除装置，会造成人身伤害和/或设备损坏。

必须采取必要的安全预防措施，以保证在进行系统拆装之前，S7-200 和 CP 243-1 的电源已断开。

应按照以下步骤，拆卸 CP 243-1 或其它 S7-200 扩展模板：

1. 切断S7-200 CPU、CP 243-1以及所有扩展模板的电源。
2. 拆下你想拆除装置的所有电缆和导线。
3. 打开前盖，并从相邻模板上拆下扁平电缆。
4. 旋出螺钉，或打开锁扣，将模板从控制面板或导轨中取下。

警告

在使用 S7-200 时，如果安装了错误的装置，会导致严重的后果。

如果更换了其它型号的 CP 243-1 或没有正确地将装置校准，会造成人身伤害和/或设备损坏。

因此，应使用相同型号的 CP 243-1，并正确校准。

组态

组态选项

通过 CP 243-1A 通讯处理器，S7-200 系统可以与其它 S7-200 系统以及 S7-300、S7-400 或基于 OPC 的系统进行通讯。

有两种方法，可以在一个 S7-200 系统中进行这种类型的通讯组态：

- 使用 STEP 7 Micro/WIN 32（版本 3.2.1 或以上）进行组态
- 使用 S7-200 用户程序进行组态

注意

如果 CRC 功能已经关闭，系统只能通过 S7-200 用户程序对系统进行组态。

由于就 CRC 校验关闭后的有意和无意修改方面，CP 243-1 不能完全检查组态数据的一致性，因此不能保证这种情况下连接到网络中的 CP 或组件能够正常运行。

在这两种情况下，组态数据都将保存在 S7-200 CPU 中的数据块中。每次 CP 243-1 重新启动时，都将从该位置读取数据一次。

注意

如果系统已进入无电压状态，为了保护 CP 243-1 中的组态数据，这些数据必须保存在 S7-200 CPU 的非易失性数据保存区中。

对于标准 S7-200 系统，整个数据块都定义为非易失性。但是，如果组态有所改变，S7-200 系统中的这种缺省设置也会更改。

如果使用 STEP 7 Micro/WIN 32 重新组态或修改 CP 243-1 的组态，只有在 CP 243-1 重新启动后，新的组态才有效。如果在这种重新组态后，S7-200 CPU 从“STOP（停止）”改变为“RUN（运行）”模式，CP 243-1 将会自动重新启动。但是，如果 CP 243-1 直接在 S7-200 用户程序中组态，通过调用用户程序中的“ETHx_CFG”子程序，也可以启用组态。调用子程序，也会导致 CP 243-1 重新启动。

TCP/IP 的地址参数（IP 地址、子网掩码、网关的 IP 地址）可以在组态过程中进行定义，也可以在引导装置入从 BOOTP 服务器中动态检索 TCP/IP 地址参数，来组态 CP 243-1。

如果 S7-200 系统通过 CP 243-1 与 S7-300、S7-400 或基于 OPC 的系统进行通讯，你可以使用 STEP 7（版本 5.1 或以上，带有 Service Pack 3）（以及用于工业以太网的 NCM S7），组态 S7-300、S7-400 或基于 OPC 的系统。

注意

由于是基于专用网络的服务（“ICMPRedirect”），在引导装入后，CP 243-1 可以独立地切换到另一个网关。大约 30 秒之后，CP 243-1 又可切回到原来组态的网关。因此，CP 243-1 实际所使用的网关于组态中定义的网关临时会有偏差。

你可以在任何特定的时间，从 S7-200 CPU 中保存 NPB 数据块的保存区，读取 CP 243-1 正在使用的网关（见第 4.4.3 节）。为此，应使用 STEP 7 Micro/WIN 32 中的 CP 243-1 诊断窗口或使用用户程序。

组态数据的取值范围

IP 地址

在组态中各个节点所指定的 IP 地址，必须符合定义 IP 地址有效性的一般约定。

根据这些约定，某些 IP 地址可以用于特定的目的。这些地址可能会不被 CP 243-1 所接受。包括：

- 回送（Loopback）： 127.0.0.0 - 127.255.255.255
- “Class D”地址： 224.0.0.0 - 239.255.255.255
- “Class E”地址： 240.0.0.0 - 247.255.255.255
- 广播地址： 例如 255.255.255.255

子网掩码

如果在组态过程中指定了一个子网掩码，其结构必须符合定义子网掩码有效性的一般约定。

请注意，一个 IP 地址和一个相关子网掩码的有效性都是独立的。

TSAP（传输层服务访问点）

TSAP 由 2 个字节组成。第一个字节规定连接，第二个字节由通讯模板的机架号和槽号组成。以下数值范围适用于第一个字节：

- 本地 TSAP 数值范围： 16#02，16#10 - 16#FE
- 远程 TSAP 数值范围： 16#02，16#03，16#10 - 16#FE

CP 243-1 不能检查第二个字节的结构。

使用 STEP 7 Micro/WIN 32 组态 CP 243-1

在你的 PC 中安装并启动 STEP 7 Micro/WIN 32 后，可以启动 CP 243-1 的向导程序（Wizard）。该向导程序位于“Ethernet Wizard...（以太网向导）”项下的“Extras（附加）”菜单中。在“Extras（附加）”项下窗口中带有导航条的 STEP 7 Micro/WIN 32 窗口的左侧，也可以找到该向导程序，在 STEP 7 Micro/WIN 32 中可以对它启用。

在你进行 CP 243-1 组态时，Ethernet Wizard 可以提供技术支持。在几个掩码中，你可以输入所有相关参数。对于这种用户向导设计，在当前掩码中的所有输入项都完成并正确之前，你不能输入一个新的掩码。否则，会显示出错报文。

使用 Ethernet Wizard 进行组态的步骤，简述如下。

注意

详细信息，请参见与 STEP 7 Micro/WIN 32 一起随附的《STEP 7 Micro/WIN 32》手册。

在你启动 Wizard 后，将出现一个包含一般信息的输入掩码。在你读取内容之后，请点击“Continue>”，继续。

CP 243-1 在 S7-200 系统中的位置定义

使用第二个掩码，可以定义 CP 243-1 在 S7-200 系统中的位置。该位置可以手工输入，或使用 Wizard 进行搜索。在定位 CP 243-1 后，其位置将自动显示在 S7-200 系统中。否则，会显示出错报文。

定义 TCP/IP 地址参数和传输类型

另外一个掩码用于定义 TCP/IP 地址参数以及所使用的传输类型。

有两种方法可以用于设定 TCP/IP 地址参数：

1. 在相应输入窗口中，手工输入参数。
2. 激活访问 BOOTP 服务器。在这种情况下，CP 243-1 可以在其引导装入时，从 BOOTP 服务器中检索 TCP/IP 地址参数。如果在你的 TCP/IP 网络中 CP 243-1 不能定位 BOOTP 服务器，它将进入“Reset（置位）”模式，并重新启动，再次尝试建立与 BOOTP 服务器的联系。CP 243-1 会一直继续这样做，直至找到可以检索 TCP/IP 地址参数的 BOOTP 服务器。

定义控制字节地址和连接数量

使用下一个掩码，可以规定你的 S7-200 系统中存储地址空间的字节地址，在此，S7-200 CPU 可以寻址 CP 243-1。该地址取决于 CP 243-1 在你的 S7-200 系统中的位置，以及你的 S7-200 系统的输出数量。在你开始组态系统时，如果你使用 Ethernet Wizard 确定 CP 243-1 在 S7-200 系统中的位置，Wizard 现在即可向你提供所使用的地址。

一般地，通过启用 STEP 7 Micro/WIN 32 中“Target system（目标系统）”菜单的“Information...”输入项，你就可确定 S7-200 系统中的模板所占用的存储地址空间。在该掩码中，你还可以定义 CP 243-1 一次可以同时保持的最大连接数量。最多可以保持 8 个这种连接。然后，在你可以组态连接的掩码中，将显示你在此指定的每个连接。

组态每个连接

你在前一掩码中建立的连接，可以在下一掩码中进行组态。对于每个连接，你必须首先定义你的 S7-200 系统是作为客户机运行，还是作为服务器运行。这决定了掩码的结构。

如果 S7-200 系统作为一个客户机连接运行，你必须指定通讯伙伴的地址以及在该通讯伙伴中的通讯终点（“TSAP（传输层服务访问点）”）。而且，在另一个掩码中，你必须指定哪些数据必须在 S7-200 系统和指定通讯伙伴之间进行交换。在该掩码中，你还可以定义这些数据是进行读操作还是进行写操作。每个连接可以最多定义 32 个读/写命令。

如果 S7-200 系统作为一个服务器连接运行，通过分配 IP 地址，你可以定义你授权可以访问你的系统的通讯伙伴。但是，你还应设定每个服务器，以便可以根据任何 IP 地址，授权访问。你还必须定义被授权访问你的 S7-200 系统的通讯伙伴的通讯终点（“TSAP”）。

对于客户机连接和服务器连接，都可启用一个“Keep Alive（持续作用）”系统。请使用在通讯伙伴组态中指定的通讯伙伴的通讯终点（“TSAP”）。在 S7-200 系统中，可以使用 STEP 7 Micro/WIN 32 生成。在 S7-300、S7-400 或基于 OPC 的系统中，使用 STEP 7 生成（见第 4.4 节）。

注意

STEP 7 和 STEP 7 Micro/WIN 32 中的通讯终点（“TSAP”）规格必须相互兼容。

启用/禁用 CRC 功能，定义“Keep Alive（持续作用）”时间

在你组态完连接后，你必须在下一掩码中规定 S7-200 CPU 中的组态数据是否需要使用 CRC 功能进行保护，防止无意修改。

如果启用了 CRC 功能，CP 243-1 可以在引导装入时，检查它从 S7-200 CPU 的存储器中读取的其组态数据是否被用户程序所修改。如果组态数据被修改，它将停止引导装入程序，并尝试从 BOOTP 服务器中检索其 TCP/IP 地址参数。如果没有被修改，它将继续引导装入程序。但是，在这种情况下，只有 MicroWN 通道被启用。因此，CP 243-1 只能与 STEP 7 Micro/WIN 32 进行通讯，不能与其它控制器进行通讯。

建议激活 CRC 功能。这是 CP 243-1 能够识别其组态数据是否被用户程序无意修改的唯一方式。

如果 CRC 功能没有启用，你可以在用户程序中修改 CP 243-1 的组态数据。但是，CP 243-1 将不能识别组态数据是否被无意修改。

注意

如果 CRC 功能已经关闭，只能使用 S7-200 用户程序组态数据。

就 CRC 校验关闭后的有意和无意修改方面，CP 243-1 不能完全检查组态数据的一致性。因此在这种情况下，就不能保证连接在网络中的 CP 或组件能够正确运行。

在同一掩码中，你可以一次设定所有已组态连接的“Keep Alive（持续作用）”时间。如果一个连接伙伴突然失踪，例如如果 TCP/TP 网络故障或通讯伙伴出现错误，在此所输入的数值可以用于确定在多长时间后，CP 243-1 开始查找故障。

在组态每个连接时，你也应定义在该时间段内应监控的连接。

定义保存组态的存储区

最后，使用下一掩码，定义你的组态数据在 S7-200 CPU 中保存的存储区。使用 Wizard，可以为此提供帮助。然后，Wizard 将告知你有关根据你的组态它正在建立的子程序信息，以及组态数据保存信息。至此，系统组态即告完成。

注意

应确保 Ethernet Wizard 保存组态数据的存储区没有被你的 S7-200 用户程序所使用。

从用户程序组态 CP 243-1

CP 243-1 的组态数据都被保存在 S7-200 CPU 存储器中，因此可以从一个 S7-200 用户程序中直接进行修改。对于组态数据，应禁用循环冗余校验（CRC），以便在下次启动时，CP 243-1 可以接收以此修改的组态数据。为此，CDB 数据结构的字节 13 必须输入为“16#AC”。只要在 Ethernet Wizard 中关闭 CRC 功能，这将会自动发生。

注意

从用户程序组态 CP 243-1 只是对有经验的程序员的一种建议。

就 CRC 校验关闭后的有意和无意修改方面，CP 243-1 不能完全检查组态数据的一致性。因此在这种情况下，就不能保证连接在网络中的 CP 或组件能够正确运行。

注意

“WORD（字）”（双字节）或“DWORD（双字）”（4 字节）类型的数据都以“big endian（高低）”格式保存在 S7-200 中。即，

地址 n： MSB

地址 n+1： LSB （类似于 DWORD）

占用系统标志区（SM 区）

CP 243-1 在 S7-200 CPU 系统标志区中占用 50 个字节。这 50 个字节的地址取决于目前 CP 243-1 在 S7-200 CPU 系统中的位置。CP 243-1 的一般信息和状态信息主要保存在这 50 个字节中。最后 4 个字节包括一个指针，通过这个指针来访问 CP 243-1 组态数据。这些组态数据按逻辑顺序保存在 S7-200 CPU 变量存储器中。并分为：

- 组态数据块（CDB）
- 网络参数块（NPB）
- 网络数据块（NDB）

下表所示为模板在 S7-200 系统中的位置与相关系统标志区之间的关系。

在 S7-200 系统中的位置	占用系统标志区（SM 区）	备注
CPU	-	-
0	200..249	-
1	250..299	只支持 CPU 固件版本 1.2 或以上
2	300..349	只支持 CPU 固件版本 1.2 或以上
3	350..399	只支持 CPU 固件版本 1.2 或以上
4	400..449	只支持 CPU 固件版本 1.2 或以上
5	450..499	只支持 CPU 固件版本 1.2 或以上
6	500..549	只支持 CPU 固件版本 1.2 或以上

表 11-2：系统标志区

组态数据块（CDB）的结构

CDB 由 Ethernet Wizard 在 STEP 7 Micro/WIN 32 中生成。下表所示为 CDB 的结构。

变量存储器中的字节偏移	说明	数据格式	举例
标题			
0-4	模板名称	5 字节 ASCII	16#4350323433“CP243”
5-6	CDB 长度	2 个字节，十六进制	16#006C（108，十进制）
7-8	NPB 长度	2 个字节，十六进制	16#0014（20，十进制）

变量存储器中的字节偏移	说明	数据格式	举例
一般信息			
9	内部使用	1 个字节，十六进制	
10	内部使用	1 个字节，十六进制	
11-12	为 STEP 7 Micro/WIN 预留	2 个字节，十六进制	---
13-14	公用标志 位 [0] 全双工模式 0: 半双工 1: 全双工 位 [1] 数据传输速率 0: 10 Mbit/s 1: 100 Mbit/s 位[2] 自动谈判 0: 无自动谈判 1: 自动谈判 位 [3] BOOTP 0: 使用组态的网络参数 1: BOOTP 位 [4-7] 没有使用 位[8-15] CRC 验证 启用 16#00 CRC 校验 启用 16#AC CRC 校验	2 个字节，十六进制	16#0004: 自动谈判，使用组态的网络参数，启用 CRC 校验 16#AC04： 自动谈判，使用组态的网络参数，禁用 CRC 校验
15-18	组态的 IP 地址 如果使用 BOOTP，该字段应设定为 16#00000000	4 个字节，十六进制	192.12.45.23:16#C00C2D17
19-22	组态的子网掩码	4 个字节，十六进制	255.255.255.0:16#FFFFFF00
	如果使用 BOOTP，该字段应设定为 16#00000000		
23-26	网关的 IP 地址 16#00000000 意思是指：不要使用网关。 如果使用 BOOTP，该字段应设定为 16#00000000	4 个字节，十六进制	192.12.45.24: 16#C00C2D18
27-28	Keep Alive（持续作用）时间参数，单位[秒]	2 个字节，十六进制	16#001E：30 秒
S7 连接第 0 分节（如果不是所有字节都用于该分节，应填充为“16#00”）			
29	标志字节 位 [0]服务器/客户机 0: 服务器 1: 客户机 位 [1] Keep Alive（持续作用） 0: 没有 Keep Alive 支持	1 个字节，十六进制	16#82: 服务器，Keep Alive（持续作用）支持，S7 连接，0，正在使用，包括有效数据。

变量存储器中的字节偏移	说明	数据格式	举例
	1: Keep Alive 支持　　位 [2-6] 没有使用 位 [7] 分节有效 0: 分节没有使用 1: 分节正在使用		
30-33	对于服务器功能： 用于访问保护的客户机 IP 地址空间 16#00000000: 没有保护 16#XXXXXX00 允许具有相同 Class-C 区段的客户机 16#XXXXXXXX 只允许具有相同地址 对于客户机功能： S7 服务器的 IP 地址	4 个字节，十六进制	192.12.45.22: 16#C00C2D16.
34-35	本地 TSAP	2 个字节，十六进制	16#1000
S7 连接 1 分节（如果不是所有字节都用于该分节，应填充为“16#00”）			
38	标志字节 见 S7 连接第 0 分节	1 个字节，十六进制	见 S7 连接第 0 分节
39-42	通讯伙伴的 IP 地址 见 S7 连接第 0 分节	4 个字节，十六进制	见 S7 连接第 0 分节
43-44	本地 TSAP	2 个字节，十六进制	16#1100
45-46	远程 TSAP	2 个字节，十六进制	见 S7 连接第 0 分节
S7 连接 2 分节（如果不是所有字节都用于该分节，应填充为“16#00”）			
47	标志字节 见 S7 连接第 0 分节	1 个字节，十六进制	见 S7 连接第 0 分节
48-49	通讯伙伴的 IP 地址 见 S7 连接第 0 分节	4 个字节，十六进制	见 S7 连接第 0 分节
52-53	本地 TSAP	2 个字节，十六进制	16#1200
54-55	远程 TSAP	2 个字节，十六进制	见 S7 连接第 0 分节
S7 连接 3 分节（如果不是所有字节都用于该分节，应填充为“16#00”）			
56	标志字节 见 S7 连接第 0 分节	1 个字节，十六进制	见 S7 连接第 0 分节
57-60	通讯伙伴的 IP 地址 见 S7 连接第 0 分节	4 个字节，十六进制	见 S7 连接第 0 分节
61-62	本地 TSAP	2 个字节，十六进制	16#1300
63-64	远程 TSAP	2 个字节，十六进制	见 S7 连接第 0 分节

变量存储器中的字节偏移	说明	数据格式	举例
S7 连接 4 分节（如果不是所有字节都用于该分节，应填充为“16#00”）			
65	标志字节 见 S7 连接第 0 分节	1 个字节，十六进制	见 S7 连接第 0 分节
66-69	通讯伙伴的 IP 地址 见 S7 连接第 0 分节	4 个字节，十六进制	见 S7 连接第 0 分节
70-71	本地 TSAP	2 个字节，十六进制	16#1400
72-73	远程 TSAP	2 个字节，十六进制	见 S7 连接第 0 分节
S7 连接 5 分节（如果不是所有字节都用于该分节，应填充为“16#00”）			
74	标志字节 见 S7 连接第 0 分节	1 个字节，十六进制	见 S7 连接第 0 分节
75-78	通讯伙伴的 IP 地址 见 S7 连接第 0 分节	4 个字节，十六进制	见 S7 连接第 0 分节
79-80	本地 TSAP	2 个字节，十六进制	16#1500
81-82	远程 TSAP	2 个字节，十六进制	见 S7 连接第 0 分节
S7 连接 6 分节（如果不是所有字节都用于该分节，应填充为“16#00”）			
83	标志字节 见 S7 连接第 0 分节	1 个字节，十六进制	见 S7 连接第 0 分节
84-87	通讯伙伴的 IP 地址 见 S7 连接第 0 分节	4 个字节，十六进制	见 S7 连接第 0 分节
88-89	本地 TSAP	2 个字节，十六进制	16#1600
90-91	远程 TSAP	2 个字节，十六进制	见 S7 连接第 0 分节
S7 连接 7 分节（如果不是所有字节都用于该分节，应填充为“16#00”）			
92	标志字节 见 S7 连接第 0 分节	1 个字节，十六进制	见 S7 连接第 0 分节
93-96	通讯伙伴的 IP 地址 见 S7 连接第 0 分节	4 个字节，十六进制	见 S7 连接第 0 分节
97-98	本地 TSAP	2 个字节，十六进制	16#1700
99-100	远程 TSAP	2 个字节，十六进制	见 S7 连接第 0 分节
STEP 7 Micro/Win 服务器分节			
101	标志类型 位[0]服务器 0: 服务器 1: 不被支持	1 个字节，十六进制	16#82: 服务器，Keep Alive 支持，STEP 7 Micro/WIN 服务分节正在使用，包含有效数据。

变量存储器中的字节偏移	说明	数据格式	举例
	位 [1] Keep Alive（持续作用） 0: 没有 Keep Alive 支持 1: Keep Alive 支持 位 [2-6] 没有使用 位 [7] 分节有效 0: 不被支持 1: 分节正在使用		
102-105	内部使用	4 个字节，十六进制	
CRC 分节			
106-107	所有 CDB CRC，无 CRC 分节	2 个字节，十六进制	

表 11-3：CBD 结构

网络参数块（NPB）的结构

该数据块由 CP 243-1 本身根据网络参数的当前设置生成。它包含当前正在使用的 TCP/IP 参数值，并已由 CP 243-1 正确组态。如果组态错误，NPB 将不包含任何有效数据。

变量存储器中的字节偏移	说明	数值计算	数据格式	举例
108-109	公用标志字节 位 [0] 全双工模式 0: 半双工 1: 全双工 位 [1] 数据传输速率 0: 10 Mbit/s 1: 100 Mbit/s 位[2] 自动谈判 0: 无自动谈判 1: 自动谈判 位 [3] BOOTP 0: 使用组态的网络参数 1: BOOTP 位 [4-15] 没有使用	取决于当前组态	2 个字节，十六进制	16#04:自动谈判，使用组态的网络参数
110-113	当前 IP 地址	取决于当前组态	4 个字节，十六进制	192.12.45.23: 16#C00C2D17
114-117	当前子网掩码	取决于当前组态	4 个字节，十六进制	255.255.255.0: 16#FFFFFF00
118-121	当前网关的 IP 地址	取决于当前组态	4 个字节，十六进制	192.12.45.24: 16#C00C2D18
122-127	MAC 地址	从硬件中读取	6 个字节，十六进制	16#080006021F04 08-00-06-02-1F-04

表 11-4：NPB 结构

网络数据块（NDB）的结构

NDB由Ethernet Wizard在STEP 7 Micro/WIN 32中生成。在该数据块中，可以组态客户机的读/写命令。8个通讯通道中的每一个都可以最多组态32个读/写命令。如果CP 243-1作为服务器在一个通道上运行，对于该通道在NDB结构中就没有入口。

下表所示为NDB的结构。读/写命令代码以字母n、m、p = 0，...，31来表示，通道代码以字母r = 0，...，7来表示。

变量存储器中的字节偏移	名称	说明	数据格式
标题			
128-129	NDB_LENGTH	规定 NDB 的长度	2 个字节，十六进制
第一个客户机通道的入口			
130	COM_CH0_ID	第一个客户机通道的代码	1 个字节，十六进制
131	COM_CH0_OFF	规定第一个通讯块（COM0）的偏移	1 个字节，十六进制
132	COM_CH0_LEN0	规定第一个通讯块（COM0）的长度	1 个字节，十六进制
	...		n 字节
n+132	COM_CH0_LENn	规定 COMn 结构的长度	1 个字节，十六进制
n+5	COM_CH0_0	第一个客户机通道的读/写命令 0 的 COM0 结构“<op>=<cnt>,<local_buffer>,<remote_buffer>”（说明见表 11-6）	ASCII
...	...	...	ASCII
n+5	COM_CH0_n	第一个客户机通道的读/写命令 n 的 COMn 结构“<op>=<cnt>,<local_buffer>,<remote_buffer>”（说明见表 11-6）	ASCII
第二个客户机通道的入口			
...	COM_CH1_ID	第二个客户机通道的代码	1 个字节，十六进制
...	COM_CH1_OFF		1 个字节，十六进制
...	COM_CH1_LEN0		1 个字节，十六进制
...	...		1个字节，十六进制
...	COM_CH1_LENm		1个字节，十六进制
...	COM_CH1_0		ASCII
...	...		ASCII
...	COM_CH1_m		ASCII
...	...	（最多 8 个通道）	
第 n 个客户机通道的入口			
...	COM_CHr_ID	最后一个客户机通道的代码	1 个字节，十六进制

变量存储器中的字节偏移	名称	说明	数据格式
...	COM_CHr_OFF		1 个字节，十六进制
...	COM_CHr_LEN0		1 个字节，十六进制
...	...		1 个字节，十六进制
...	COM_CHr_LENp		1 个字节，十六进制
...	COM_CHr_0		ASCII
...	...		ASCII
...	COM_CHr_p		ASCII
CRC 分节			
NDB 的最后 2 个字节	所有 NDB CRC，无 CRC 分节	2 个字节，十六进制	NDB 的最后 2 个字节

表 11-5：NDB 的结构

名称	说明	数据格式
<op>	命令类型 数值范围： “R”，读命令 “W”，写命令	ASCII
<cnt>	被传送字节的数量 数值范围： 1- 212	ASCII
<local_buffer>	本地系统存储区的地址 数值范围： “VB0” - “VBx”，其中“x”为最大变量地址	ASCII
<remote_buffer>	通讯伙伴存储区的地址 数值范围： “IB0” - “IBx”，其中“x”为最大输入地址（S7-200 / S7-300 / S7-400） “QB0” - “QBx”，其中“x”为最大输出地址（S7-200 / S7-300 / S7-400） “MB0” - “MBx”，其中“x”为最大标志地址（S7-200 / S7-300 / S7-400） “VB0” - “VBx”，其中“x”为最大变量地址（S7-200） “DB0.DBB0” - “DBx.DBBy”，其中“x”为最大 DB 编号， “y”为 DB 中相应数据块的最高地址（S7-300 / S7-400）	ASCII

表 11-6：读/写命令的组态

使用 STEP 7 组态通讯伙伴

以下章节将以S7-300系统为例，阐述使用STEP 7组态系统的步骤，以通过相关的以太网通讯处理器与S7-200系统进行通讯。S7-400系统的组态步骤与此类似。

详细的组态步骤，请参见《STEP 7》手册（MLFB：6ES7 810-4CC05-0YX0）或《CP 343-1手册》和《CP 443-1手册》。

在S7-300和S7-400系统中，组态的连接和自由连接不同。对于组态的连接，连接参数由用户给定。相反，自由连接不必使用STEP 7组态。

组态的连接

如果你想使用一个组态连接，你必须首先将一个新的S7连接插入STEP 7 NetPro程序包中。在“Insert new connection（插入新的连接）”掩码中，规定你想与之建立连接的站的类型。对于连接通讯伙伴，应选择“（unspecified）（不确定）”类型。

现在必须组态这些连接。为此，你必须在“Properties - S7 connection（S7连接属性）”掩码中（见图11-7），首先定义你的S7-300或S7-400系统是作为一个主动参与者还是作为被动参与者。如果你的S7-300或S7-400系统需要与S7-200系统进行通讯，也可以定义系统是作为服务器使用还是作为客户机使用。如果你想使S7-300或S7- 400系统作为客户机使用，应启用“Active connection generation（有效连接生成）”输入项。如果该输入项没有启用，系统将作为一个服务器使用。然后，根据TCP/IP协议，进行S7连接所需的必要设置。为此，应选择“TCP/IP”项。为了建立与你的通讯伙伴的TCP/IP连接，也可以规定通讯伙伴的IP地址。

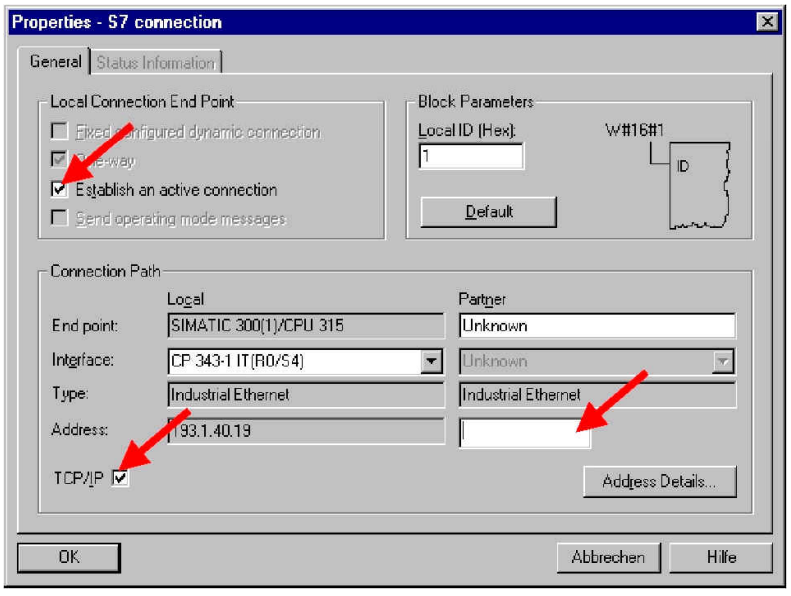


图 11-7 “S7 连接属性”掩码

最后，在“Address details（地址明细）”掩码中，定义需要使用的通讯终点（“TSAP”）。S7-200 系统中一个连接的 TSAP 可以在 STEP 7 Micro/WIN 32 中找到，在其掩码中，在“Local properties（本地属性）”输入项下，你可以组态每个连接。在专为“Address details（地址明细）”掩码中的通讯伙伴的 TSAP 提供了字段中，输入通讯终点。分配给 S7-300 或 S7-400 系统中连接的通讯终点，也可以在“TSAP”字段中，在“Local（本地）”输入项下的同一掩码中找到。在 STEP 7 Micro/WIN 32 中输入数值，从其掩码中，你可以检索所使用的通讯终点。该数值需要在该掩码中的“TSAP”字段中的“Remote properties（远程属性）”输入项中输入。

S7-300 系统作为客户机使用

如果你想使用 S7-300 系统作为客户机，即，如果你启用了“Properties - S7 connection（S7 连接属性）”掩码中的“Active connection generation（有效连接生成）”，你就不能使用在 STEP 7 Micro/WIN 32 中使用 STEP 7，在“Local（本地）”输入项中规定的 TSAP，用作 S7-300 系统的一个连接终点。你必须首先根据以下规则传送该 TSAP：

TSAP 的第 1 个字节：

从 STEP 7、“Address details（地址明细）”掩码、“Local（本地）”、“TSAP”字段中取代。

TSAP 的第 2 个字节：

可以从你的 S7-300 系统中的以太网通讯处理器的机架和槽号得出。

以太网通讯处理器所在机架的机架号插入在 S7-300 系统中规定的前 3 位。

其它 5 位包含你的 S7-300 系统中的以太网通讯处理器的槽号。

这两个数值都可以在 STEP 7 HWConfig 程序包中找到。

举例：

将 CP 343-1 插入 S7-300 系统机架 0 的第 4 号槽中。如果你使用 STEP 7 进行连接组态，数值“10.02”将显示为本地 TSAP。在 STEP 7 Micro/WIN 32 中，你必须使用数值为“10.04”的 TSAP，用于 S7-300 系统中的通讯终点（第 1 个字节（此时：10）被取代，第 2 个字节可从机架（此时为 0）和槽号（此时为 4）中得出）。

注意

STEP 7 和 STEP 7 Micro/WIN 32 中的通讯终点（“TSAP”）的定义必须相互兼容。

自由连接

自由连接只能用于 S7-300 或 S7-400 系统用作服务器时。自由连接不需要使用 STEP 7 进行组态。标准 S7-300 和 S7-400 系统都可通过自由连接建立通讯。

如果你想使用自由连接用于你的 S7- 300 或 S7-400 系统，这些连接中每一个连接的客户机侧都仍必须进行组态。在 STEP 7 Micro/WIN 32 中，这此连接的使用以及在 S7- 300 或 S7-400 系统组态的通过这些连接的通讯之间没有什么区别。对于自由连接，你只需保证 TSAP 的第 1 个字节总被赋值以“0x03”即可，通过该字节，这些连接将被传送到 S7-300 和 S7-400 系统。TSAP 的第 2 个字节可以如上从 S7-300 或 S7-400 系统中所使用的 CPU 的机架号和槽号导出。

注意

S7-200 系统不支持自由连接。这就意味着，你必须组态 S7-200 系统中的每一个连接，不管你的系统是作为客户机运行，还是作为服务器运行。

组态出错时 CP 243-1 的行为

如果 CP 243-1 识别到一个无效的组态，它将尝试通过 BOOTP 服务器检索其 TCP/IP 地址参数（IP 地址、子网掩码、网关的 IP 地址）。CP 243-1 继续该尝试将持续大约 1 分钟。如果在该时间内它不能从 BOOTP 服务器中收到一个响应，或如果响应无效或错误，红色指示灯（“SF”）将闪亮大约 30 秒钟。该步骤将循环进行，直到 CP 243-1 在 S7-200 CPU 存储器中找到一个有效的组态或从 BOOTP 服务器中接收到一个有效的响应。

如果CP 243-1接收到一个有效的BOOTP响应，它将如下自行组态：

- IP 地址、子网掩码和网关的 IP 地址都可从 BOOTP 响应中取代。
- 传输类型设定为“Auto Negotiation（自动适应）”。
- 连接的 Keep Alive（持续作用）时间设定为 30 秒。

如果以此方式进行组态，CP 243-1 就能通过以太网从 STEP 7 Micro/WIN 32 访问 S7-200 CPU。现在，一个新的有效组态即可装入。在这种情况下，不能进行与其它控制器的通讯。复位后，CP 243-1 可以根据新的组态进行自行组态。

注意

自动适应模式只能在所有所连接的网络组件都支持该模式时才能运行。

编程

可以使用 STEP 7 Micro/WIN 32，开发 S7-200 用户程序。为了能使你在这些程序中使用 CP 243-1 功能，必须使用版本为 V3.2.1 或以上的 STEP 7 Micro/WIN 32。

如果 CP 243-1 作为服务器或作为客户机使用，必须至少组态 CP 243-1 的一个通讯通道。另外，S7-200 用户程序也必须相应编程。

CP 243-1 在以下子程序中的 S7-200 用户程序中进行编程，CP 243-1 所在位置位于包括同一子程序的 S7-200 系统中。

- ETHx_CTRL （x 表示槽位，可能的数值有： 0.1, ...6)
- ETHx_CFG （x 表示槽位，可能的数值有： 0.1, ...6)
- ETHx_XFR （x 表示槽位，可能的数值有： 0.1, ...6)

在组态完成后，这些子程序可以由集成在 STEP 7 Micro/WIN 32 中的 Ethernet Wizard 生成。然后，你将可以在“Subprogram calls（子程序调用）”输入项下，在操作树中窗口中的 STEP 7 Micro/WIN 32 中找到这些子程序。在组态完成时，Wizard 生成哪些子程序，这取决于组态时你规定的数据。

注意

这些子程序不能在 S7-200 用户程序中由中断程序调用。

ETHx_CTRL

ETHx_CTRL 子程序用于初始化和监控 CP 243-1。如果你想访问 CP 243-1 的功能，你必须在每次循环开始时，在你的 S7-200 用户程序中调用该子程序。如果 CRC 校验打开，CP 243-1 识别到一个组态数据变化，调用子程序会造成 CP 243-1 重新启动。相反，如果 CRC 校验关闭，在用户程序或新的组态从 STEP 7 Micro/WIN 32 中下载到 S7-200 CPU 后，CP 243-1 总会重新启动，并且 S7-200 CPU 接着启动。

返回值可以提供 CP 243-1 的一般状态信息以及最多 8 个通讯通道的状态信息。如果在 CP 243-1 中出现错误，你可以从“Error（出错）”返回参数中读取相关的错误代码。只要你一完成 CP 243-1 的组态，Ethernet Wizard 就可在 STEP 7 Micro/WIN 32 中生成 ETHx_CTRL 子程序。

调用：

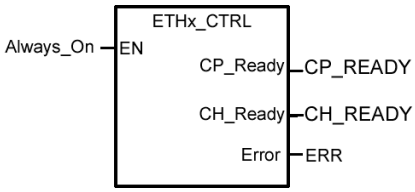


图 11- 8　调用 ETHx_CTRL 子程序

输入参数：

名称	类型	含义
-	-	-

表 11-7：输入参数（ETHx_CTRL）

返回参数：

名称	类型	含义
CP_Ready	BOOL	CP 243-1 的状态 0: CP 没有运行准备就绪 1: CP 运行准备就绪
CH_Ready	BYTE	每个通道的状态 位 0 对应通道 0 位 1 对应通道 1 位 2 对应通道 2 位 3 对应通道 3 位 4 对应通道 4 位 5 对应通道 5 位 6 对应通道 6 位 7 对应通道 7 0: 通道没有准备就绪 1: 通道准备就绪
Error	WORD	错误代码 0x0000:没有出现错误 其它： 出错（说明：见第 6.2 节）

表 11-8：返回参数（ETHx_CTRL）

如果“CH_Ready”返回参数的一个位的数值为“1”，表示相关通道已准备就绪。这就意味着，在组态中所定义的通讯伙伴的通讯连接可以根据通讯参数建立（IP 地址，TSAP）。

ETHx_CFG

通过调用 ETHx_CFG 子程序，你可以引导 CP 243-1 读取保存在 S7-200 CPU 存储器中的组态数据。在读入数据后，CP 243-1 将自动进行复位。在复位后系统重新启动时，从 S7-200 CPU 存储器中所读入的组态将为有效组态。

在 S7-200 用户程序运行时，如果你想从该用户程序中动态重新编程 CP 243-1，你就需要该子程序。如果你的组态没有启用 CRC 功能，只能由 Ethernet Wizard 在 STEP 7 Micro/WIN 32 中生成。如果调用了 ETHx_CFG 子程序，CP 243-1 将中止所有的现有连接，并复位。但是，只要你一启用 CRC 功能，你就不用再从用户程序中修改相关的组态。然后，你只需使用 STEP 7 Micro/WIN 32 中的 Ethernet Wizard 进行修改即可。

调用：

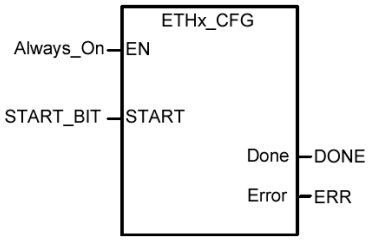


图 11-9 调用 ETHx_CFG 子程序

输入参数：

名称	类型	含义
START	BOOL	激活一个新组态的输入条件 0：不能激活新组态 1：可以激活新组态

表 11-9：输入参数（ETHx_CFG）

返回参数：

名称	类型	含义
Done	BOOL	子程序调用的状态 0: 子程序还没有执行 1: 子程序已执行，并准备下一次执行
Error	BYTE	错误代码 16#00: 没有出现错误 否则： 出错（说明：见第 6.2 节）

表 11-10：返回参数（ETHx_CFG）

ETHx_XFR

通过调用 ETHx_XFR 子程序，你可以引导 CP 243-1 将数据传送到另一个 S7 系统，或从这样一个系统中对数据进行排队。CP 243-1 所进行的数据访问类型，在组态时规定。因此，你可以在组态时定义以下参数：

- 你想访问哪些数据
- 你是想读这些数据还是写这些数据
- 你想从哪一个通讯伙伴检索这些数据，或你想将这些数据传送到哪一个通讯伙伴

当你调用 ETHx_XFR 子程序时，你应规定组态的数据访问，以用于通过调用子程序你想执行的客户机通道。

如果你至少将 CP 243-1 中的一个通道组态为客户机使用，ETHx_XFR 子程序只能由 Ethernet Wizard 在 STEP 7 Micro/WIN 32 中生成。然后，你只能从一个 S7-200 用户程序中通过 CP 243-1 进行数据访问。

每个通道一次只能有一个 ETHx_XFR 子程序激活。在一个通道不能并行进行几个数据访问。因此，建议你将“START”输入与 ETHx_XFR 子程序的“Done”返回值和 ETHx_CTRL 子程序的“CH_Ready”返回值的相应位进行关联。

调用：

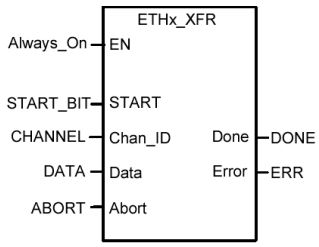


图 11- 10　调用 ETHx_XFR 子程序

输入参数：

名称	类型	含义
START	BOOL	发布读/写命令的输入条件 0: 没有发布读/写命令 1：发布读/写命令
Chan_ID	BYTE	可以访问数据的通道数量。该通道必须作为一个客户机进行组态。 数值范围： 0 .. 7
Data（数据）	BYTE	通道相关数据块的数量，在此组态可以描述要执行的读/写命令。 数值范围： 0 .. 31
Abort（中止）	BOOL	中止数据访问的输入条件 0: 不能中止数据访问 1：中止数据访问

表 11-11：输入参数（ETHx_XFR）

返回参数：

名称	类型	含义
Done	BOOL（布尔）	子程序调用的状态 0: 子程序还没有执行 1: 子程序已执行，读/写命令已完成，子程序准备下一次执行
Error	BYTE（字节）	错误代码 16#00: 没有出现错误 其它： 出错（说明：见第 6.2 节）

表 11-12：返回参数（ETHx_XFR）

注意

S7-200 用户程序的执行时间会直接影响到所使用读/写命令的持续时间。

如果你想将读/写命令的执行时间降为最低，你应使用户程序的循环时间尽可能地短。

诊断

诊断选项

对于诊断，具有以下含义：

- Ping 服务器：
  “ping”程序在所有标准计算机上都有提供，例如 Microsoft Windows 操作系统，该程序可以用于确定 CP 243-1 是否在所输入的某个地址下。
- STEP 7 Micro/WIN 32：
  在“Target system（目标系统）”菜单中的“Information ...”输入项下，你可以检索 S7-200 系统中的一般信息。包括与 S7-200 系统所连接的模板信息。当你通过双击模板中的该输入项选中“CP 243-1 ETHERNET”模板时，在你的 CP 243-1 上将出现一个窗口，包括以下信息：
- 一般模板信息（模板类型，所用版本）

- TCP/IP 地址参数（IP 地址、子网掩码、网关的 IP 地址、MAC 地址）如果组态出错，这些数值将无效，系统将通过 BOOTP 功能重新引导装入。如果用户程序还没有完成一个循环，也会如此。
- 状态报文
- CP 243-1 在 S7-200 CPU 存储地址空间中的嵌入信息
- 通道的组态和状态
- 出错报文
- 读取 SM 存储区：
  有关 CP 243-1 的信息，也可以从当前 CP 所使用的 SM 区中读取（见表 11-2），并在运行期间由 S7-200 用户程序处理。如果在 CP 243-1 中出现全局错误，其相关错误代码也可以在该区找到。下面为信息的保存位置。

SM 区中的字节偏移	含义	格式
0-15	模板类型	16 字节 ASCII
16-19	软件版本	4 字节 ASCII
20-21	错误代码（见第 6.2 节）	2 个字节，十六进制
22	CP 243-1 的状态 位 [0] 0: CP 243-1 没有引导装入 1: CP 243-1 引导装入 位 [1] 0: BOOTP 程序还没有执行 1: BOOTP 程序正在执行 位 [2] 0: CP 243-1 没有运行准备就绪 1: CP 243-1 运行准备就绪 位 [3] 0: STEP 7 Micro/Win 32 没有启用 1: STEP 7 Micro/Win 32 启用 位 [4] 0: 根据数据块组态 1: CDB 中没有有效的组态 位 [5] 预留 位 [6] 0: 没有以太网连接 1: CP 243-1 连接以太网 位 [7] 0: CP 243-1 中没有错误 1: CP 243-1 出现错误	1 个字节，十六进制
23	预留	
24	硬件版本	1 个字节，ASCII

表 11-13：全局出错及其错误代码

- 读取 NPB 存储区：
  在 CP 243-1 当前使用的 SM 区的字节 46-49 中，有一个指针可以定位保存 CP 243-1 组态数据的存储区。如果你增加该指针 108，你将发现 NPB 存储区，在该区中保存有当前 CP 243-1 正在使用的 TCP/IP 组态参数，在此可以正确组态 TCP/IP，并且用户程序至少经过一个循环。如果组态错误，NPB 中就不含有有效的数值。下表所示为这些存储区的结构。

变量存储器中的字节偏移	含义	格式
108 – 109	命令标志字节 位 [0] 双工模式 0：半双工 1：全双工 位 [1] 数据传输速率 0：10 Mbit/s 1: 100 Mbit/s 位[2] 自动适应 0：自动适应没有启用 1：自动适应启用 位 [3] BOOTP 0: 组态网络参数 1：通过 BOOTP 位 [4] – 位[15]的网络参数：保留	2 个字节，十六进制
110-113	当前 IP 地址	4 个字节，十六进制
114- 117	当前子网掩码	4 个字节，十六进制
118-121	当前所使用网关的 IP 地址	4 个字节，十六进制
122-127	MAC 地址	6 个字节，十六进制

表 11-14：NPB 存储区的结构

- LED 指示符（见第 2.5 节）

注意

如果 CP 243-1 与当前客户机的连接突然中断（例如掉电），而服务器继续运行，服务器将不能识别这种连接中断。如果客户机尝试重新连接，系统相关等待时间等于组态的 Keep Alive（持续作用）时间，即从 CP 243-1 再次打开开始，直到所有连接重新建立。

CP 243-1 的出错报文

以下所述为 CP 243-1 最重要的出错报文。所有其它出错报文，可参见 CP 243-1 的内部出错报文。如果出现这种出错报文，应与技术支持热线联系垂询。

注意

在模板出错或系统出错时，必须关闭模板，然后打开。

一般地，CP 243-1 有两种功能可以用于报告出错：

- 作为子程序的返回值：
  错误通过“Error（错误）”返回值来报告。根据子程序，该参数可以是“BYTE（字节）”或“WORD（字）”类型。
- 作为出错代码通过 S7-200 CPU 的 SM 存储区返回：
  当前所使用的 SM 存储区取决于 CP 243-1 在 S7-200 CPU 系统中的连接位置。根据错误类型，CP 243-1 的当前 SM 区中的以下字节可以用于错误信息：
  - 字节偏移20和21中的全局错误。它们必须作为字进行编译（SMW）
  - 根据相关通道，在以下字节中将出现通道错误：
    字节偏移25 通道0
    字节偏移26 通道1
    字节偏移27 通道2
    字节偏移28 通道3
    字节偏移39 通道4
    字节偏移20 通道5
    字节偏移31 通道6
    字节偏移32 通道7
  - 命令错误将在偏移33的字节中返回

每个错误的错误代码及其含义在下表中列出。这些表格还可以说明哪些功能可以返回每个错误，哪个子程序出现哪种错误代码，以及在当前 SM 存储器中的哪一个字节出现错误。如果出现了这些表格中没有列出的错误，请与技术支持热线联系垂询（见第 7 章）。

如果在下列表格中没有建议的排除办法，需要使用 STEP 7 Micro/WIN 32 手工排除。对于 CP 243-1 的组态，应使用集成在 STEP 7 Micro/WIN 32 中的 Ethernet Wizard。

如果你的 CP 243-1 仍返回组态错误，你应检查组态是否被你的用户程序所修改。

注意

如果 24 V 电源故障，CP 243-1 就不能返回错误信息。

出错字节		说明	响应/故障排除	返回功能	
十六进制	十进制			SM 区中的字节偏移	返回值 (ETHx_)
16#01	1	S7 总线中的超时	自动重新引导装入	25 - 32 33	_XFR _CFG
16#02	2	由于 ABORT 命令，数据访问被禁止	启动新的读/写命令	25 - 32	_XFR
16#03	3	“DATA”传输参数超出组态范围		25 - 32	_XFR
16#04	4	在 S7 中不能建立连接	反复尝试建立连接	25 - 32	_XFR
16#05	5	连接中止或在一个没有准备就绪的通道中尝试执行读/写命令。	检查通讯伙伴的连接路径	25 - 32	_XFR
16#06	6	在响应包中包含有逻辑错误	启动新的读/写命令 检查组态	25 - 32	_XFR
16#07	7	读命令失败	启动新的读/写命令 应检查相应读命令的参数	25 - 32	_XFR
16#08	8	写命令失败	启动新的读/写命令 应检查相应写命令的参数	25 - 32	_XFR
16#09	9	没有组态通道	启动一个具有其它参数的新读/写命令	25 - 32	_XFR
16#0A	10	尽管通道已组态为一个服务器，仍进行尝试启动读/写命令	启动一个具有其它参数的新读/写命令	25 - 32	_XFR
16#0B	11	先前的读/写命令还没有完成	启动新的读/写命令 评估先前读/写命令的“DONE”返回参数	25 - 32	_XFR
16#0C	12	无效的命令代码	启动新的读/写命令	25 - 32	_XFR
16#0D	13	由于用户程序启动重新组态，所有数据传输均中断	重新启动系统	25 – 32 33	_XFR _CFG
16#80	128	没有 24 V 外部电压	等待直至准备就绪	25 – 32 33	
16#81	129	正在使用的通道没有准备就绪或运行不正确	等待直至准备就绪 评估 ETHx_CTRL 子程序的返回参数		_XFR _CFG
16#82	130	正在使用的通道忙	等待直至准备就绪		_XFR _CFG
16#83	131	启动了一个含有禁用通道号的命令	启动新的读/写命令 应检查用户程序		_XFR
16#84	132	启动了一个含有禁用数据块号的命令	启动新的读/写命令 应检查用户程序		_XFR

表 11-15：出错报文（出错字节）

出错字		说明	响应/故障排除	返回功能	
十六进制	十进制			SM 区中的字节偏移	返回值 (ETHx_)
16#0001	1	S7 总线中的超时	自动热启动	20,21	_CTRL
16#000D	13	由于用户程序启动重新组态，所有数据传输均中断	重新启动系统	20,21	_CTRL
16#0030	48	在特定时间段内 CPU 不能调用组态	自动热启动	20,21	_CTRL
16#0031	49	在 S7-200 CPU 存储器中没有找到符合语法的正确的 CDB 组态		20,21	_CTRL
16#0032	50	组态数据（CDB，NDB）的 CRC 校验和不正确		20,21	_CTRL
16#0033	51	CP 243-1 的组态数据错误或保存不正确		20,21	_CTRL
16#0034	52	CDB 的指针错误或没有装入 CDB	确保在用户程序的一开始就调用了 ETHx_CTRL 子程序 检查 CDB 指针（SM 存储区中的字节偏移 46）	20,21	_CTRL
16#0035	53	所传输的组态有一个无效的格式代码		20,21	_CTRL
16#0036	54	TSAP 定义不正确或在组态中多次出现		20,21	_CTRL
16#0038	56	组态不正确（IP 地址错误，既没有组态为客户机也没有组态为服务器，STEP 7 Micro/WIN 32 通道没有启用）		20,21	_CTRL
16#003A	58	组态中的 CP 243-1 的模板名已经修改		20,21	_CTRL
16#003B	59	组态中包含一个无效的 IP 地址		20,21	_CTRL
16#003C	60	组态中包含一个无效的网关地址		20,21	_CTRL
16#003D	61	组态在“Keep Alive（持续作用）”参数中包含有一个无效的数值		20,21	_CTRL
16#003E	62	没有接收到一个有效的组态，不管是从 S7- 200 CPU 存储器还是通过 BOOTP	反复尝试从 S7-200 CPU 存储器或通过 BOOTP 接收一个有效的组态	20,21	_CTRL

出错字		说明	响应/故障排除	返回功能	
十六进制	十进制			SM 区中的字节偏移	返回值 (ETHx_)
16#0042	66	NDB 中包含有一个语法不正确的读/写命令或长度错误		20,21	_CTRL
16#0093	147	BOOTP 命令失败	自动热启动	20,21	_CTRL
16#0094	148	BOOTP 服务器包含有无效的数据	自动热启动	20,21	_CTRL
16#0095	149	Keep Alive（持续作用）时间不被 TCP/IP 堆栈接受	自动热启动	20,21	_CTRL
16#0096	150	一个客户机的 IP 地址不被 TCP/IP 堆栈接受	自动热启动	20,21	_CTRL
16#0097	151	子网掩码不被 TCP/IP 堆栈接受	自动热启动	20,21	_CTRL
16#0098	152	网关地址不被 TCP/IP 堆栈接受	自动热启动	20,21	_CTRL
16#00F0	240	CP 243-1 不能被 S7-200 CPU 识别	检查 S7-200 系统的组态和结构		_CTRL
16#00F1	241	CP 243-1 根据组态可以访问的输出字节地址与 CP 243-1 在 S7-200 系统中的位置不兼容	检查 S7-200 系统的组态和结构		_CTRL
16#0100 - 16#0108	256 - 264	在 S7 总线出现超时	根据需要检查 24V 电压提供	20,21	_CTRL
16#8080	32896	CP 243-1 还没有完成引导装入	根据需要检查 24V 电压提供	20,21	_CTRL

表 11-16：出错报文（出错字）

使用 USS 协议库
控制 MicroMaster 驱动

12

STEP7-Micro/WIN 指令库，该指令库包括预先组态好的子程序和中断程序，这些子程序和中断程序都是专门为通过 USS 协议与驱动通讯而设计的。使用这些 USS 指令，你可对驱动进行控制并读写参数。

您可以在 STEP7-Micro/WIN 指令树的库文件夹中找到这些指令。当您选择一个 USS 指令时，系统会自动增加一个或多个相关的子程序（USS1 到 USS7）。

本章内容：

使用 USS 协议的要求

STEP7-Micro/WIN 指令库提供 14 个子程序、3 个中断程序和 8 条指令支持 USS 协议。这些 USS 指令使用下列 S7-200 的资源：

- 初始化 USS 协议将专用作 USS 通讯
  使用 USS_INIT 指令为 Port0 选择 USS 或 PPI（USS 是指 SIMOTION MicroMaster 驱动的 USS 协议）。在选择使用 USS 协议与驱动通讯后，Port0 不能够再用作其它目的，包括与 STEP7-Micro/WIN 通讯。
  使用 USS 协议为您的应用编程时，您需要使用 CPU226、CPU226XM，或与您的计算机上 PROFIBUS CP 卡相连的 PROFIBUS-DP 模块 EM277。STEP7-Micro/WIN 可以利用第二个通讯口在 USS 协议运行时监视程序。
- USS 指令影响所有的与 Port 0 自由口通讯相关的 SM 区。
- USS 指令使用 14 个子程序和 3 个中断程序。
- USS 指令使得您的用户程序对存储空间的需求最多可增加 3450 字节。根据所使用的特定的 USS 指令，这些指令所支持的路径使控制程序对存储空间的分摊增加至少 2150 字节，最多 3450 字节。
- USS 指令的变量需要 400 字节的 V 存储区。该区域的起始地址由用户指定并保留给 USS 变量。
- 有一些 USS 指令还要求 16 字节的通讯缓存区。作为一个指令的参数，您要为该缓存区提供一个 V 区的起始地址。建议为每一例 USS 指令指定一个单独的缓存区。
- 在执行计算时，USS 指令使用累加器 AC0 至 AC3。您仍然可以在您的程序中使用这些累加器，只是累加器中的数值会被 USS 指令改变。
- USS 指令不能用在中断程序中。

提示

要将 Port0 恢复为 PPI 使之与 STEP7-Micro/WIN 通讯，您可以使用另外一条 USS_INIT 指令重新设定 Port 0。

您还可以将 S7-200 的模式开关设为 STOP，这样就复位 Port 0 的参数。请注意，停止与驱动的通讯也就停止了驱动。

计算与驱动通讯的时间要求

S7-200 的循环扫描和驱动的通讯是异步的。S7-200 在完成一个驱动的通讯传送之前通常要完成若干个循环扫描。以下各因素可帮助您确定所需要的时间：驱动的数量、波特率、S7-200 的循环扫描时间。

有一些驱动在使用参数访问指令时要求更长的时延。参数访问对时间的需求量取决于驱动的类型和要访问的参数。

在使用 USS_INIT 指令将 Port 0 指定为 USS 协议后，S7-200 会以表 12-1 所示的时间间隔轮询所有激活的驱动。您必须为每个驱动设置超时（time-out）参数以完成该任务。

表 12-1　通讯时间

波特率	对激活的驱动进行轮询的时间间隔（无参数访问指令激活）
1200	240 毫秒（最大）乘以驱动的数量
2400	130 毫秒（最大）乘以驱动的数量
4800	75 毫秒（最大）乘以驱动的数量
9600	50 毫秒（最大）乘以驱动的数量
19200	35 毫秒（最大）乘以驱动的数量
38400	30 毫秒（最大）乘以驱动的数量
57600	25 毫秒（最大）乘以驱动的数量
115200	25 毫秒（最大）乘以驱动的数量

提示

同时只能有一个 USS_RPM_x 或 USS_WPM_x 指令激活。要等到每个指令的 Done 位输出指示完成，才能通过用户逻辑触发一个新的指令。每个驱动只能使用一个 USS_CTRL 指令。

使用 USS 指令

要在您的 S7-200 控制程序中使用 USS 协议指令，请遵循以下步骤：

1. 在您的程序中插入 USS_INIT 指令并且该指令只在一个循环周期内执行一次，您可以用 USS_INIT 指令启动或改变 USS 通讯参数。
   当您插入 USS_INIT 指令时，若干个隐藏的子程序和中断服务程序会自动地加入到您的程序中。
2. 在您的程序中为每个激活的驱动只使用一个 USS_CTRL 指令。
   您可以按需求尽可能多地使用 USS_RPM_x 和 USS_WPMx 指令，但是，在同一时刻，这些指令中只能有一条是激活的。
3. 在指令树中选中程序块图标（Program Block）点击右键（显示菜单）为这些库指令分配 V 区。选择库存储区选项，显示库存储区分配对话框。
4. 组态驱动参数使之与程序中所用的波特率和站地址相匹配。

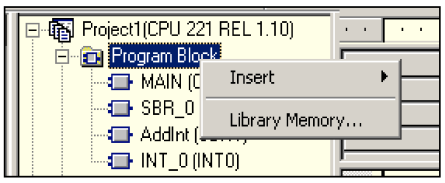


图 12-1　为库指令分配 V 区

5. 连接 S7-200 和驱动之间的通讯电缆。
   确保象 S7-200 这样的所有连接驱动的控制设备，通过一条短而粗的电缆连接到同一个地或象驱动一样星形连接。

注意

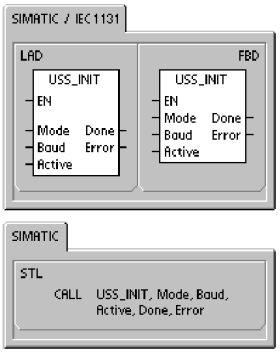
具有不同参考电位的设备相互连接时会在连接电缆中形成电流。这些电流会导致通讯错误或设备损坏。

确保所有通过通讯电缆连接在一起的设备共享一个公共参考点，或者彼此隔离以避免产生电流。

屏蔽层必须接到外壳地上或 9 针接头的针 1 上。建议您将 MicroMaster 驱动上的接线端 2-0V 接到外壳地上。

USS 协议指令

USS_INIT 指令



USS_INIT 指令用来使能、初始化或禁止 MicroMaster 驱动的通讯。USS_INIT 指令必须无错误地执行，才能够执行其它的 USS 指令。在继续执行下一条指令前，该指令结束且 Done 位立即置位。

当 EN 输入接通时，每一循环都执行该指令。

在每一次通讯状态改变时只执行一次 USS_INIT 指令。使用边沿检测指令脉冲触发 EN 输入接通。

要改变初始化参数，需执行一个新的 USS_INIT 指令。

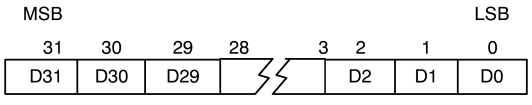
通过 USS 输入值可选择不同的通讯协议：输入值为 1 指定 Port 0 为 USS 协议并使能该协议，输入值为 0 指定 Port 0 为 PPI 并且禁止 USS 协议。Baud 设置波特率为 1200、2400、4800、9600、19200、38400、57600 或 115200。Acitive 指示哪个驱动激活，有些驱动只支持地址 0 到 30。

表 12-2　USS4_INIT 指令的参数

输入/输出	数据类型	操作数
Mode	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

图 12-2 所示为对激活的驱动输入的描述和格式。所有标为 Active（激活）的驱动都会在后台被自动地轮询，控制驱动搜索状态，防止驱动的串行链接超时。

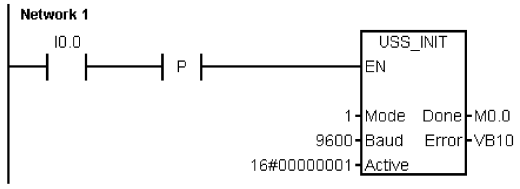
参见表 12-1 计算各状态间轮询的时间。



D0 Drive 0 激活位；0－未激活；1－激活
D1 Drive 1 激活位；0－未激活；1－激活

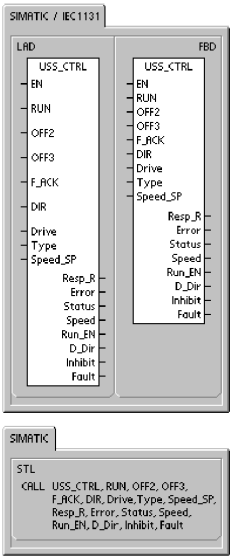
图 12-2 Active 参数的格式

当 USS_INIT 指令完成时，Done 位输出接通。Error 输出字节中包含该指令的执行结果。表 12-6 定义了由于执行该指令而引起的错误条件。

示例：USS_INIT 子程序	
	Network 1 LD I0.0 EU CALL USS_INIT, 1, 9600, 16#00000001, M0.0, VB10

USS_CRTL 指令

USS_CRTL 指令用于控制激活的 MicroMaster 驱动。USS_CTRL 指令将所选的命令存放在一个通讯缓存区中，然后发送到所寻址的驱动中（Drive 参数），该驱动应已在 USS_INIT 指令的参数 Active 中选中。



对于每一个驱动只能使用一个 USS_CTRL 指令。

有些驱动只以正值的形式报告速度。如果速度为负，驱动报告一个正的速度值同时反向 D_Dir 位（方向位）。EN 位必须接通使能 USS_CTRL 指令。该指令要始终保持使能。

RUN（RUN/STOP）指示驱动是否接通（1）或断开（0）。当 RUN 位接通时，MicroMaster 驱动接收命令，以指定的速度和方向运行。为使驱动运行，必须满足以下条件：

- 该驱动必须在 USS_INIT 中激活。
- OFF2 和 OFF3 必须设为 0。
- Fault 和 Inhibit 位必须为 0。

当 RUN 断开时，命令 MicroMaster 驱动斜坡减速直至电机停止。OFF2 位用来允许 MicroMaster 驱动斜坡减至停止，OFF3 位用来命令 MicroMaster 驱动快速停止。

Resp_R（响应收到）位应答来自驱动的响应，轮询所有激活的驱动以获得最新的驱动的状态信息。每次 S7-200 接收到来自驱动的响应时，Resp_R 位在一个循环周期内接通并且刷新以下各值。

F_ACK（故障应答）位用于应答驱动的故障。当 F_ACK 从 0 变 1 时，驱动清除该故障（Fault）。

DIR（方向）位指示驱动应向哪个方向运动。

Drive（驱动地址）是 MicroMaster 驱动的地址，USS_CTRL 命令发送到该地址。有效地址为 0 到 31。

Type（驱动类型）选择驱动的类型。对于 3 系列的（或更早的）MicroMaster 驱动，类型为 0，对于 4 系列的 MicroMaster 驱动，类型为 1。

表 12-3　USS_CTRL 指令参数

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
Status	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

Speed_SP（速度设定值）是驱动的速度，是满速度的百分比。Speed_SP 的负值使驱动反向旋转。范围是：-200.0%至 200.0%。

Error 是错误字节，包含最近一次向驱动发出的通讯请求的执行结果。表 12-6 定义了该指令执行可能引起的错误条件。

Status 是驱动返回的状态字的原始值。图 12-3 所示为标准状态字的状态位及主反馈。

Speed 是驱动速度，是满速度的百分比，范围：-200.0%至 200.0%。

Run_EN（RUN 使能）指示驱动是运行（1）还是停止（0）。

D_Dir 指示驱动转动的方向。

Inhibit 指示驱动上禁止位的状态（0—未禁止，1－禁止）。要清除禁止位，Fault（故障）位必须为零，而且 RUN、OFF2 和 OFF3 输入必须断开。

Fault 指示故障位的状态（0－无故障，1－有故障）。驱动显示故障代码。（请参考您的驱动手册）。要清除 Fault，必须排除故障并接通 F_ACK 位。

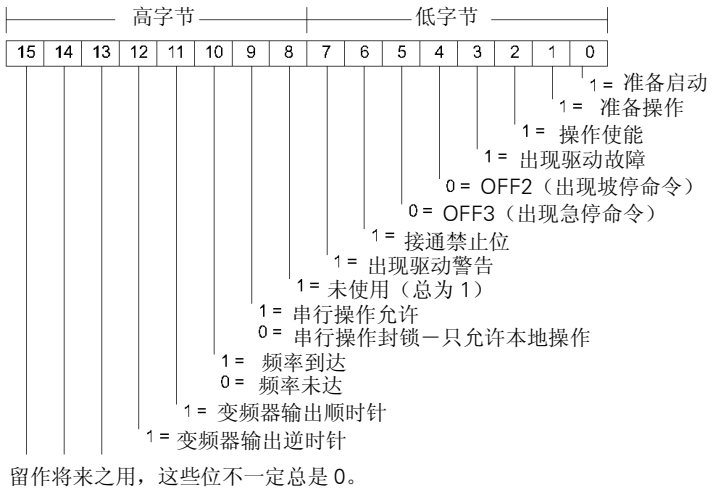


图 12-3　3 系列小变频标准状态字的状态位以及主反馈

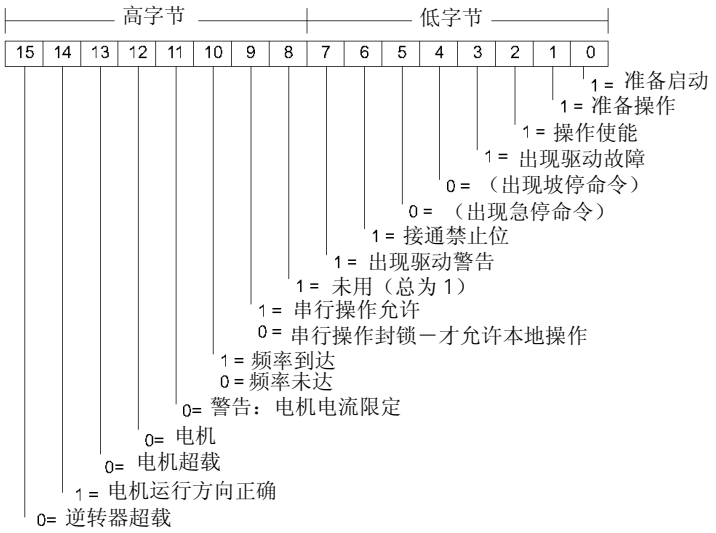
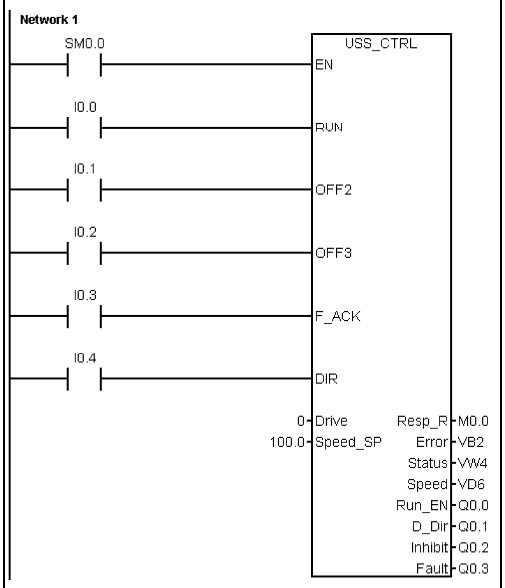


图 12-4　4 系列小变频标准状态字的状态位和主反馈

示例：USS_CTRL 子程序



只显示为STL:

```
Network 1          //0号驱动的控制
LD      SM0.0
CALL    USS_CTRL, I0.0, I0.1, I0.2, I0.3, I0.4, 0, 1, 100.0,
        M0.0, VB2, VW4, VD6, Q0.0, Q0.1, Q0.2, Q0.3
```

显示为LAD或FBD:

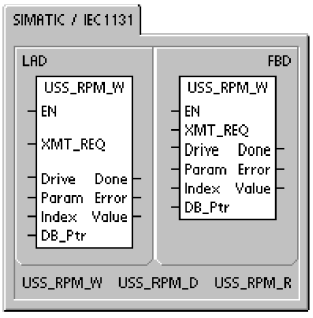
```
Network 1          //0号驱动的控制
LD      SM0.0
=       L60.0
LD      I0.0
=       L63.7
LD      I0.1
=       L63.6
LD      I0.2
=       L63.5
LD      I0.3
=       L63.4
LD      I0.4
=       L63.3
LD      L60.0
CALL    USS_CTRL, L63.7, L63.6, L63.5, L63.4, L63.3,
0, 1, 100.0, M0.0, VB2, VW4, VD6, Q0.0, Q0.1, Q0.2, Q0.3
```

USS_RPM_x 指令

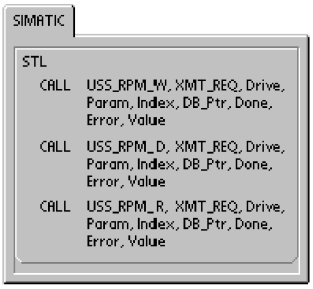
USS 协议有三种读指令：

- USS_RPM_W 指令读取一个无符号字类型的参数
- USS_RPM_D 指令读取一个无符号双字类型的参数
- USS_RPM_R 指令读取一个浮点数类型的参数

同时只能有一个读（USS_RPM_x）或写（USS_WPM_x）指令激活。



当 MicroMaster 驱动对接收的命令应答或有报错时，USS_RPM_x 指令的处理结束。在这一过程等待应答时，逻辑扫描继续执行。



要使能对一个请求的传送，EN 位必须接通并且保持为 1 直至 Done 位置 1，即意味着过程结束。例如，当 XMT_REQ 输入接通时，每一循环扫描向 MicroMaster 驱动传送一个 USS_RPM_x 请求。因此，应使用脉冲边沿检测作为 XMT_REQ 的输入，这样，每当 EN 输入有一个正的改变时，只发送一个请求。

Drive 是向其发送 USS_RPM_x 命令的 MicroMaster 驱动的地址。每个驱动的有效地址为 0 到 31。

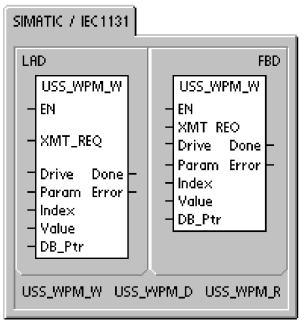
Param 是参数号码，Index 是要读的参数的索引值。Value 是返回的参数数值。DB_Ptr 输入应是一个 16 字节缓存区的地址。该缓存区用于存储向 MicroMaster 驱动发送的命令的执行结果。

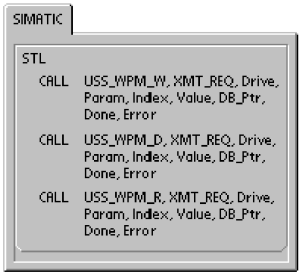
当 USS_RPM_x 指令结束时，Done 输出接通，Error 输出字节和 Value 输出包含该指令的执行结果。表 12-6 定义了执行该指令引起的错误条件。只有 Done 位输出接通时 Error 和 Value 输出才有效。

表 12-4　USS_RPM_x 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、能流，上升沿有效
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

USS_WPM_x 指令





USS 协议有三种读指令：

- USS_WPM_W 指令写一个无符号字类型的参数
- USS_WPM_D 指令写一个无符号双字类型的参数
- USS_WPM_R 指令写一个浮点数类型的参数

同时只能有一个读（USS_RPM_x）或写（USS_WPM_x）指令激活。

当 MicroMaster 驱动对接收的命令应答或有报错时，USS_WPM_x 指令处理结束，在这一过程等待应答时，逻辑扫描继续执行。

要使能对一个请求的传送，EN 位必须接通并且保持为 1 直至 Done 位置 1，指示过程结束。例如，当 XMT_REQ 输入接通时，每循环扫描向 MicroMaster 驱动传递一个 USS_RPM_x 请求。因此，应使用脉冲边沿检测作为 XMT_REQ 的输入，这样，每当 EN 输入有一个正的跳变时，只发送一个请求。

Drive 是向其发送 USS_WPM_x 命令的 MicroMaster 驱动的地址。每个驱动的有效地址为 0 到 31。

Param 是参数号码。Index 是要写的参数的索引值。Value 是要写到驱动上的 RAM 中的参数值。对于 3 系列的 MicroMaster 驱动，您还可以将该值写到驱动上的 EEPROM 中，这将基于您对 P971 的组态（EEPROM 存储控制）。

您必须为 DB_Ptr 输入提供一个 16 字节缓存区的地址。该缓存区由 USS_WPM_x 指令使用，存储向 MicroMaster 驱动发送的命令的执行结果。

当 USS_WPM_x 指令结束时，Done 输出接通，Error 输出字节包含该指令的执行结果。表 12-6 定义了执行该指令引起的错误条件。

表 12-5　USS_WPM_x 指令的有效操作数

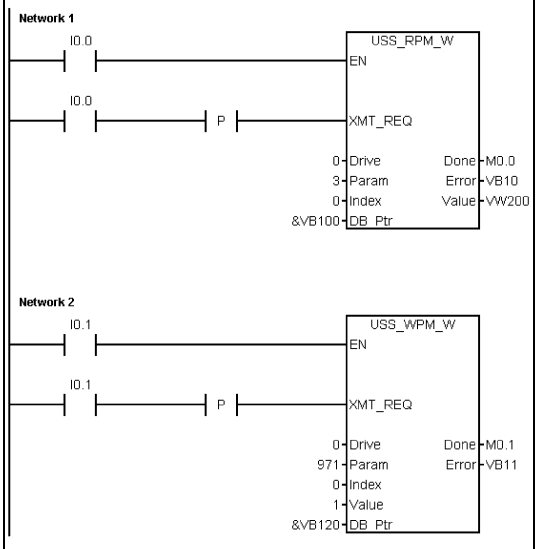
输入/输出	数据类型	操作数
XMT_REQ	BOOL	I、Q、M、S、、SM、T、C、V、L、能流，上升沿有效
Drive	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD、常数
Param、Index	WORD	VW、IW、QW、MW、SW、SMW、LW、T、C、AC、AIW、*VD、*AC、*LD、常数
DB_Ptr	DWORD	&VB
Value	WORD DWORD、REAL	VW、IW、QW、MW、SW、SMW、LW、T、C、AC、AQW、*VD、*AC、*LD VD、ID、QD、MD、SD、SMD、LD、*VD、*AC、*LD
Done	BOOL	I、Q、M、S、SM、T、C、V、L
Error	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD

注意

当您使用 USS_WPM_x 指令刷新存储在驱动的 EEPROM 中的参数设置时，必须确保不超过对 EEPROM 写周期的最大次数的限定（大约 50,000 次）。

写周期超限将引起存储数据的崩溃和数据丢失，读周期的次数没有限定，如果需要频繁地向驱动写参数，您首先要将驱动中的 EEPROM 存储控制参数设为零。

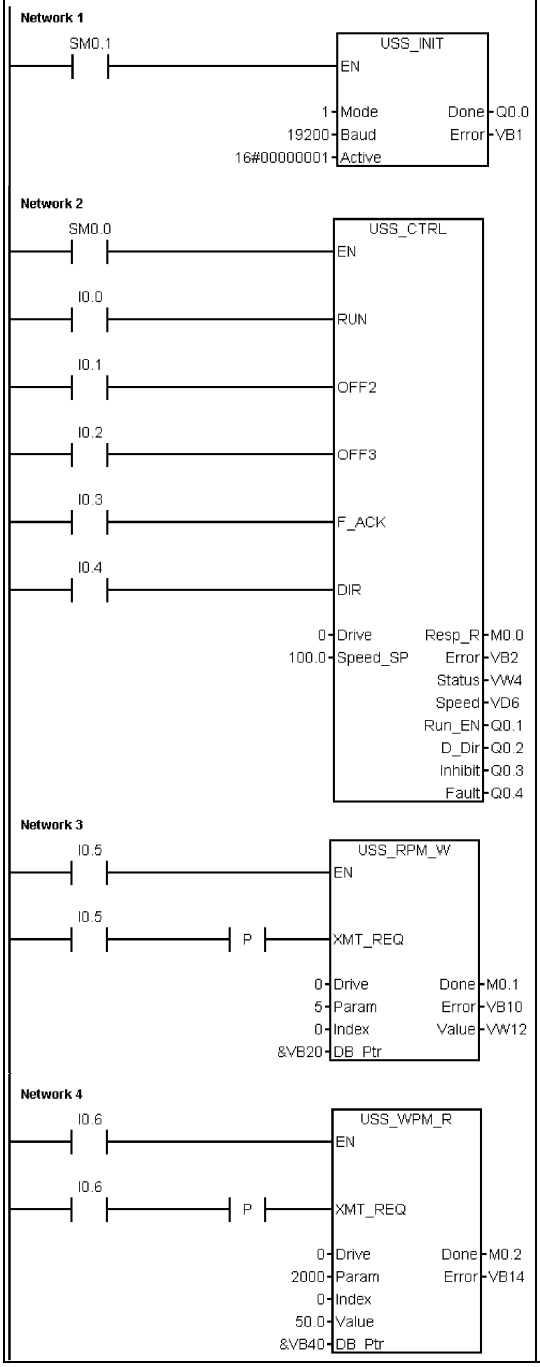
示例：USS_RPM_x 和 USS_WPM_x




```
Network 1          //这两个接触器必须有相同地址
LD      I0.0
=       L60.0
LD      I0.0
EU
=       L63.7
LD      L60.0
CALL    USS_RPM_W, L63.7, 0, 3, 0, &VB100,
        M0.0, VB10, VW200

Network 2          //这两个接触器必须有相同地址
LD      I0.1
=       L60.0
LD      I0.1
EU
=       L63.7
LDN     SM0.0
=       L63.6
LD      L60.0
CALL    USS_WPM_W, L63.7, L63.6, 0, 971, 0, 1,
        &VB120, M0.1, VB11
```

USS 协议示例程序

示例: 能够以STL方式正确显示的USS指令示例程序




```
Network 1          //初始化USS协议:
                   //在首次扫描中使能USS协议
                   //Port0的波特率为19200，
                   //地址为“0”的驱动激活。
LD        SM0.1
CALL      USS_INIT, 1, 19200, 16#00000001, Q0.0,
VB1
Network 2          //控制0号驱动的参数
LD        SM0.0
CALL      USS_CTRL, I0.0, I0.1, I0.2, I0.3, I0.4, 0, 1,
          100.0, M0.0, VB2, VW4, VD6, Q0.1, Q0.2,
          Q0.3, Q0.4

Network 3          //从0号驱动读一个字参数
                   //读索引为0的参数5。
                   //1. 将I0.5存入临时变量以便该段
                   //   程序可显示为LAD
                   //
                   //2. 将I0.5的上升沿存入临时变量
                   //    以便将它传递给子程序
                   //
LD        I0.5
=         L60.0
LD        I0.5
EU
=         L63.7
LD        L60.0
CALL      USS_RPM_W, L63.7, 0, 5, 0, &VB20,
          M0.1,VB10, VW12

Network 4          //向0号驱动写一个字参数
                   //写索引为0的参数2000。
LD        I0.6
=         L60.0
LD        I0.6
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, M0.2, VB14
```

注意: 该STL指令并不与LAD或FBD兼容

USS 执行错误代码

表 12-6　USS 指令的执行错误代码

错误代码	描述
0	无错
1	驱动没响应
2	来自驱动的响应中检测到校验和错误
3	来自驱动的响应中检测到奇偶校验错误
4	由来自用户程序的干扰引起的错误
5	尝试非法命令
6	提供非法驱动地址
7	通讯口未设为 USS 协议
8	通讯口正忙于处理某条指令
9	驱动速度输入超限
10	驱动响应的长度不正确
11	驱动响应的第一个字符不正确
12	驱动响应的长度字符不被 USS 指令所支持
13	错误的驱动响应
14	提供的 DB_Ptr 地址不正确
15	提供的参数号码不正确
16	所选协议无效
17	USS 激活；不允许改变
18	指定的波特率非法
19	没有通讯：该驱动未激活
20	驱动响应的参数或数值不正确或包含错误代码
21	请求一个字类型的数值却返回一个双字类型值。
22	请求一个双字类型的数值却返回了一个字类型值。

连接并设置 3 系列 MicroMaster 驱动

连接 3 系列 MicroMaster 驱动

您可以使用标准的 PROFIBUS 电缆和接头来连接 S7-200 和 3 系列的 MicroMaster 驱动（MM3），连接电缆的终端和偏置设置请参见图 12-5。

注意：

具有不同参考电位的设备连接在一起可能在连接电缆上形成未预期的电流。

这些未预期的电流可能造成通讯错误或设备损坏。

确保您要用通讯电缆连接在一起的设备要么共享一个公共参考点，要么彼此隔离以防止未预期电流的形成。

屏蔽层必须接到底盘地或 9 针接头的针 1。建议您将 MicroMaster 驱动上的接线端 2-0V 接到外壳地。

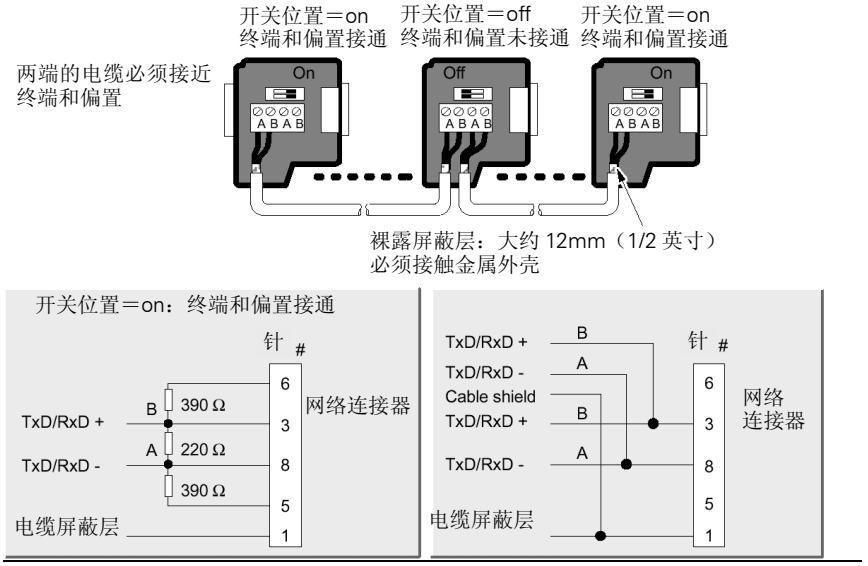


图 11-5　网络电缆的终端和偏置

设置 3 系列 MicroMaster 驱动

在将驱动连至 S7-200 之前，您必须确保该驱动有以下系统参数。使用驱动上的键盘放置这些参数：

1. 将驱动恢复为出厂设置（可选）。按 P 键：显示 P000，按向上或向下的键头直至显示 P944，按 P 输入该参数。
   P944=1
2. 使能对所有参数的读/写访问。按 P 键。按向上或向下的键头直至显示 P009。按 P 输入该参数。
   P009＝3
3. 检查您的驱动的电机设置。设置因使用的电机的不同而不同。按 P 键，按上、下键头直至显示您的驱动的电机设置。按 P 输入这些参数。
   P081＝电机的标识频率（HZ）
   P082＝电机的标识速度（RPM）
   P083＝电机的标识电流（A）
   P084＝电机的标识电压（V）
   P085＝电机的标识功率（kW/HP）
4. 设置本地/远程控制模式。按 P 键。按上、下键头直至显示 P910。按 P 输入参数。
   P910＝1 远控模式
5. 设置 RS-485 串口的波特率。按 P 键，按上、下键直至显示 P092。按 P 输入参数。按上、下键直至显示 RS-485 串口波特率的数值。按 P 输入。
   P092　3（1200 波特）
   4（2400 波特）

5（4800 波特）
6（9600 波特－缺省）
7（19200－波特）

6. 输入从站地址。每个驱动（最多 31）都可通过总线操作。按 P 键，按上、下键头直至显示 P091。按 P 输入参数。按上、下键头直至显示您想要的从站地址，按 P 输入。
   P091＝0 至 31
7. 斜坡上升时间（可选）。这是一个以秒为单位的时间，在这个时间内，电机加速至最高频率。按 P 键，按上、下键头直至 P002 显示。按 P 输入参数。按上、下键头直至显示您想要的斜坡加速时间。按 P 输入。
   P002＝0－650.00
8. 斜坡下降时间（可选）。这是一个以秒为单位的时间，在这个时间内，电机减速至完全停止。按 P 键。按上、下键头直至 P003 显示。按 P 输入参数。按上、下键直至显示您想要的减速时间。按 P 输入。
   P003＝0－650.00
9. 串行链接超时。这是到来的两个数据报文之间最大的间隔时间。该特性可用来在通讯失败时关断变频器。
   当收到一个有效的数据报文后，计时启动。如果在指定的时间内未收到下一个数据报文，变频器将触发并显示故障代码 F008，该值设为零则关断该控制。使用表 12-1 计算对驱动的状态轮询的时间。
   按 P 键。按上、下键头直至显示 P093。按 P 输入参数。按上、下键直至显示您想要的串行链接超时。按 P 输入。
   P093＝0－040（0 为缺省；时间单位为秒）
10. 串行链接标识系统设定值。该值可能不一样，但典型值为 50Hz 或 60Hz，它定义了 PV 或 SP 的 100％值。按 P 键。按上、下键直至显示 P094。按 P 输入参数。按上、下键直至显示您想要的串行链接标识系统设定。按 P 输入。
    P094＝0－400.00
11. USS 兼容性（可选）。按 P 键。按上、下键直至显示 P095。按 P 输入参数。按上、下键直至显示与您想要的 USS 兼容性相对应的号码。按 P 输入。
    P095＝0　　0.1Hz 分辨（缺省）
    　　1　　0.01Hz 分辨
12. EEPROM 存储控制（可选）。按 P 键。按上、下键直至显示 P971。按 P 输入参数。按上、下键直至显示与 EEPROM 存储控制相对应的号码。按 P 输入。
    P971＝0掉电时参数设置的改变（包括 P971）丢失
    P971＝1（缺省）掉电时，参数设置的改变保留。
13. 操作显示。按 P 退出参数模式。

连接和设置 4 系列 MicroMaster 驱动

连接 4 系列 MicroMaster 驱动

连接 4 系列 MicroMaster 驱动，将 485 电缆的两端插入为 USS 操作提供的两个卡式接线端。在 S7-200 上可使用标准 PROFIBUS 电缆和接头。

注意：

具有不同参考电位的设备连接在一起可能在连接电缆上形成未预期的电流。

这些未预期的电流可能造成通讯错误或设备损坏。

确保您要用通讯电缆连接在一起的设备要么共享一个公共参考点，要么彼此隔离以防止未预期电流的形成。

屏蔽层必须接到底盘地或 9 针接头的针 1。建议您将 MicroMaster 驱动上的接线端 2-0V 接到底盘地。

如图 12-6 所示，RS485 另外一端的两根接线必须插在 MM4 驱动的终端。在做 MM4 驱动的电缆连接时，取下驱动的前盖板露出接线终端。关于如何取下您的驱动的前盖板，请参见 MM4 用户手册。

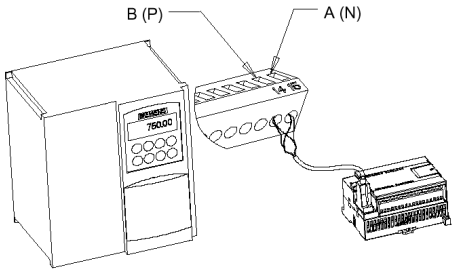


图 11-6　连接到 MM420 的接线终端

接线终端的连接以数字标识。在 S7-200 端使用 PROFIBUS 连接器，将 A 端连至驱动端的 15（对 MM420 而言）或 30（对 MM440 而言），将 B 端连到接线端 14（MM420）或 29（MM440）。

如果 S7-200 是网络中的端点，或者，如果是点到点的连接，则必须使用连接器的端子 A1 和 B1（而非 A2 和 B2），因为这样可以接通终端电阻（例如，使用 DP 接头 6ES7 972-0BA40-0XA0）。

注意

确保通电前驱动的前盖板正确地安置回原位。

如果驱动在网络中组态为端点站，那么终端和偏置电阻必须正确地连接至连接终端上。例如，图 12-7 是对 MM4，6SE6420 驱动必须做的终端和偏置连接。

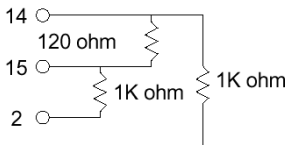


图 11-7　终端和偏置示例

设置 MM4 驱动

在将驱动连至 S7-200 之前，您必须确保驱动具有以下系统参数。使用驱动上的键盘设置参数：

1. 将驱动恢复为出厂设置（可选）： P0010＝30
   P0970＝1

   如果您忽略该步骤，确保以下参数的设置：

   USS PZD 长度：P2012 Index 0=2

   USS PKW 长度：P2013 Index 0=127

2. 使能对所有参数的读/写访问（专家模式）：P0003＝3
3. 检查您的驱动的电机设置： P0304＝额定电机电压（V）
   P0305＝额定电机电流（A）
   P0307＝额定功率（W）
   P0310＝额定电机频率（Hz）
   P0311＝额定电机速度（RPM）

   这些设置因使用的电机而不同。

   要设置参数 P304，P305，P307，P310 和 P311，您必须先将参数 P010 设为 1（快速调试模式）。当您完成参数设置后，将参数 P010 再设为 0。参数 P304，P305，P307，P310 和 P311 只能在快速调试模式下修放。
4. 设置本地/远程控制模式： P0700 Index0＝5
5. 在 COM 链接中设置到 USS 的频率设定值：P1000 Index0＝5
6. 斜坡上升时间（可选）：P1120＝0 至 650.00

   以秒为单位，是电机加速到频率所需时间。
7. 斜坡下降时间（可选）：P1121＝0 至 650.00

   以秒为单位，是电机减速至完全停止所需的时间。
8. 设置串行链接参考频率：P2000＝1 至 650Hz
9. 设置 USS 标准化：P2009 Index0＝0
10. 设置 RS-485 串口波特率：P2010 Index0＝4（2400 波特）
    5（4800 波特）
    6（9600 波特）
    7（19200 波特）
    8（38400 波特）
    9（57600 波特）
    12（115200 波特）

11. 输入从站地址： P2011 Index 0＝0 至 31

    每个驱动（最多 31 个）都可以通过总线操作。

12. 设置串行链接超时： P2014 Index 0＝0 至 65,535 毫秒

    （0＝超时禁止）

    这是到来的两个数据报文之间的最大的允许间隔时间。可使用该持性在通讯失败时关断变频器。收到一个有效数据后计时启动。如果在指定时间内未收到下一个数据报文，变频器触发并显示故障代码 F0070。该值设为 0 则关断该控制。使用表 12-1 计算对驱动状态轮询的时间。

13. 从 RAM 向 EEPROM 传送数据：

    P0971=1（启动传送）将参数设置的改变存入 EEPROM。

使用 Modbus 协议库

13

STEP7-Micro/WIN 指令库包含有专门为 Modbus 通讯设计的预先定义的子程序和中断服务程序，使得与 Modbus 主站的通讯简单易行。使用 Modbus 从站协议指令，您可以将 S7-200 组态作为一个 Modbus RTU 从站，与 Modbus 主站通讯。

您可以在 STEP7-Micro/WIN 指令树中找到这些指令。通过这些新指令，可以将 S7-200 作为 Modbus 从站。当选择一个 Modbus 从站指令时，会有一个或多个相关的子程序自动添加到您的项目中。

本章内容：

使用 Modbus 协议的要求

Modbus 从站协议指令占用 S7-200 以下资源：

- 初始化 Modbus 从站协议占用 Port0 作为 Modbus 从站协议通讯。
  当 Port0 用作 Modbus 从站协议通讯时，它不能再用作任何其它目的，包括与 STEP7-Micro/WIN 通讯。MBUS_INIT 指令控制 Port0 的设定是 Modbus 从站协议还是 PPI。
- Modbus 从站协议指令响应与 Port0 自由口通讯相关的所有 SM。
- Modbus 从站协议指令使用 3 个子程序和 2 个中断服务程序。
- Modbus 从站协议指令的两个 Modbus 从站指令及其支持子程序需占用 1857 字节的程序空间。
- Modbus 从站协议指令的变量要求 779 字节的 V 区块。该块的起始地址由用户指定，保留给 Modbus 变量。

提示

使用 MBUS_INIT 指令重新设置 Port0 则可将 Port0 改为 PPI 操作，这样就可以和 STEP7-Micro/WIN 通讯了。

Modbus 协议的初始化和执行时间

Modbus 通讯使用 CRC（循环冗余检验）以确保通讯信息的完整性。Modbus 从站协议使用一个预计算值的表以减少信息处理所需的时间。CRC 表的初始化需要大约 425 毫秒。该初始化在 MBUS_INIT 内部完成，而且通常是在进入 RUN 模式的第一个用户程序周期完成。如果 MBUS_INIT 子程序和任何其它用户初始化所需的时间超过 500 毫秒的循环时间监控，您需要复位时间看门狗并保持输出使能（如果扩展模块要求）。输出模块时间看门狗可通过写模板输出复位。请参考第六章看门狗复位指令。

当 MBUS_SLAVE 子程序进行请求服务时循环时间增加。由于大部分时间消耗在计算 Modbus CRC 上，所以对于每一字节的请求和响应，循环时间增加 650 微秒。最大的请求/响应（读或写 120 字）可增加循环时间大约 165 毫秒。

Modbus 地址

Modbus 地址通常是包含数据类型和偏移量的 5 个或 6 个字符值。第一个或前两个字符决定数据类型，最后的四个字符是符合数据类型的一个适当的值。Modbus 主站则将这个地址对应到正确的功能上。以下是 Modbus 从站指令支持的地址：

- 000001 至 000128 是实际输出，对应于 Q0.0-Q15.7
- 010001 至 010128 是实际输入，对应于 I0.0-I15.7
- 030001 至 030032 是模拟输入寄存器，对应于 AIW0 至 AIW2
- 040001 至 04XXXX 是保持寄存器，对应于 V 区。

表 13-1　映射 Modbus 地址到 S7-200

Modbus 地址	S7-200 地址
000001	Q0.0
000002	Q0.1
000003	Q0.2
…	…
000127	Q15.6
000128	Q15.7
010001	I0.0
010002	I0.1
010003	I0.2
…	…
010127	I15.6
010128	I15.7
030001	AIW0
030002	AIW2
030003	AIW4
…	…
030032	AIW62
040001	HoldStart
040002	HoldStart+2
040003	HoldStart+4
…	…
04xxxx	HoldStart+2x(xxxx-1)

所有 Modbus 地址都是一个基础。表 13-1 所示为 Modbus 地址与 S7-200 地址的对应关系。

Modbus 从站协议允许你对 Modbus 主站可访问的输入、输出、模拟输入和保持寄存器（V 区）的数量进行限定。

MBUS_INIT 指令的参数 MaxIQ 指定 Modbus 主站允许访问的实际输入或输出（I 或 Q）的最大数量。

MBUS_INIT 指令的 MaxAI 参数指定 Modbus 主站允许访问的输入寄存器（AIW）的最大数量。

MBUS_INIT 指令的 MaxHold 参数指定 Modbus 主站允许访问的保持寄存器（V 存储区字）的最大数量。

请查看 MBUS_INIT 指令的描述，了解更多的关于为 Modbus 从站设置存储区的限制信息。

组态符号表

当您为第一个符号输入一个地址后，符号表会自动计算并分配符号表中其余的符号地址。

您要为这个占用 779 字节的表分配一个起始的 V 区地址。确保对 Modbus 从站符号表的 V 区地址分配与 MBUS_INIT 指令中 HoldStart 和 MaxHold 参数为 Modbus 保持寄存器分配的 V 区地址不重叠。如果有存储区域的地址重叠，MBUS_INIT 指令会返回一个错误。

使用 Modbus 从站协议指令

在 S7-200 程序中使用 Modbus 从站协议指令请遵循以下步骤：

1. 在您的程序中插入 MBUS_INIT 指令并且只在一个循环周期中执行该指令，MBUS_INIT 指令可用于对 Modbus 通讯参数的初始化或修改。
   当您插入 MBUS_INIT 指令时，几个隐藏的子程序和中断服务程序会自动地添加到您的程序中。
2. 为 Modbus 从站协议指令所要求的 V 存储区设定一个起始地址，该区域是一个 779 字节的连续区域。
3. 在您的程序中只使用一个 MBUS_SLAVE 指令。该指令在每个循环周期中执行，为接收到的所有请求提供服务。
4. 用通讯电缆将 S7-200 Port0 和 Modbus 主站连接起来。

注意

具有不同参考电位的设备连接在一起可能会在连接电缆上引起未预期的电流。这种电流会引起通讯错误或设备损坏。

确保所有由通讯电缆连接在一起的设备要么共享一个公共参考点，要么彼此隔离以防止产生未预期的电流。

累加器（AC0、AC1、AC2、AC3）由 Modbus 从站指令使用并显示在交叉参考列表中。在执行前，Modbus 从站指令在累加器中的数值被存储并在 Modbus 从站指令完成前恢复到累加器中，确保在执行 Modbus 从站指令时，所有在累加器中的用户数据都得到保护。

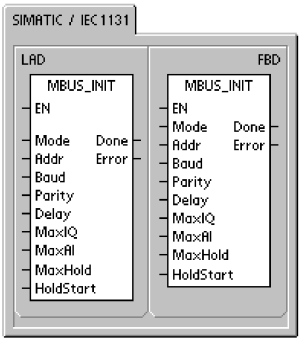
Modbus 从站协议指令支持 Modbus RTU 协议。这些指令使用 S7-200 的自由口功能，支持大部分常用 Modbus 功能。以下是所支持的 Modbus 功能：

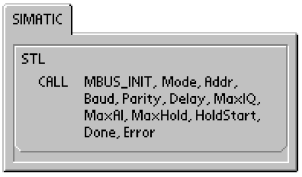
表 13-2 支持的 Modbus 从站协议功能

功能	描述
1	读单个/多个线圈（实际输出）状态，功能 1 返回任意数量输出点的接通/断开状态（Q）
2	读单个/多个接触器（实际输入）状态。功能 2 返回任意数量的输入点的接通/断开状态（I）
3	读单个/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 下是字类型，在一个请求中最多可读 120 个字。
4	读单个/多个输入寄存器。功能 4 返回模拟输入值。
5	写单个线圈（实际输出）。功能 5 将实际输出点设置为指定值。该输出点不是被强制，用户程序可以应 Modbus 的请求重写该值。
6	写单个保持寄存器。功能 6 写一个单个保持寄存器的值到 S7-200 的 V 存储区。
15	写多个线圈（实际输出）。功能 15 写多个实际输出值到 S7-200 的 Q 映象区。起始输出点必须是一个字节的开始（如，Q0.0 或 Q2.0），并且要写的输出的数量是 8 的倍数。这是 Modbus 从站协议指令的限定。这些点不是被强制，用户程序可以应 Modbus 的请求重写这些点的值。
16	写多个保持寄存器。功能 16 写多个保持寄存器到 S7-200 的 V 区。在一个请求中最多可写 120 字。

Modbus 从站协议指令

MBUS_INIT 指令





MBUS_INIT 指令用于使能和初始化或禁止 Modbus 通讯。MBUS_INIT 指令必须无错误的执行，然后才能够使用 MBUS_SLAVE 指令。在继续执行下一条指令前，MBUS_INIT 指令必须执行完并且 Done 位立即置位。

当 EN 输入接通时，该指令在每一循环周期内执行。

MBUS_INIT 指令应该在每次通讯状态改变时只执行一次。因此，EN 输入端应使用边沿检测元素以脉冲触发，或者只在第一个循环周期内执行一次。

Mode 值选择通讯协议：输入 1 值将 Port0 定义为 Modbus 协议并使能该协议，输入值为 0 将 Port0 定义为 PPI 并禁止 Modbus 协议。

参数 Baud 设置波特率为 1200、2400、4800、9600、19200、38400、57600 或 115200。

参数 Addr 设置地址，其数值在 1 到 247 之间。

表 13-3　MBUS_INIT 指令的参数

输入/输出	数据类型	操作数
Mode、Addr、Parity	BYTE	VB、IB、QB、MB、SB、SMB、LB、AC、Constant、*VD、*AC、*LD
Baud、HoldStart	DWORD	VD、ID、QD、MD、SD、SMD、LD、AC、Constant、*VD、*AC、*LD
Delay 、 MaxIQ 、MaxAI、MaxHold	WORD	VW、IW、QW、MW、SW、SMW、LW、AC、Constant、*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

参数 Parity 用于设置校验使之与 Modbus 主站相配匹。可接受值为：

- 0—无校验
- 1—奇校验
- 2—偶校验

参数 Delay 通过为标准 Modbus 信息超时增加指定数量的毫秒，扩展标准 Modbus 信息结束超时条件。当在一个连接的网络上操作时，该参数的典型值为 0。如果您使用具有纠错功能的 modem 时，将延迟时间设为 50 至 100 毫秒。如果您使用宽频电台，设置该延迟值为 10 至 100 毫秒。Delay 的数值可以是 0 到 32767 毫秒。

参数 MaxIQ 设置可使用的 I 和 Q 点数，相应于 Modbus 地址 00xxxx and 01xxxx。其数值可为 0 到 128。数值为 0 则禁止对输入和输出的读写。建议 MaxIQ 的取值为 128，即允许访问 S7-200 的所有 I 点和 Q 点。

参数 MaxHold 设置可使用的输入寄存器（AI）的字数，相应于 Modbus 地址 03xxx，其数值为 0 到 32。值为 0 则禁止读模拟输入。要允许访问所有的 S7-200 模拟输入，MaxAI 的建议值如下：

- CPU 221 为 0
- CPU 222 为 16
- CPU 224、226 和 CPU226XM 为 32。

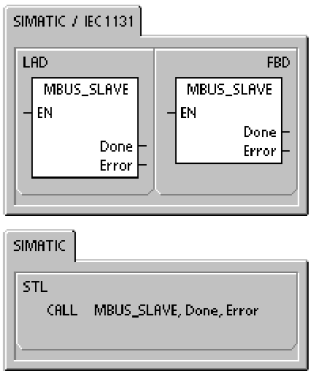
参数 MaxHold 设置可以使用的 V 区的保持寄存器的字数，相应于 Modbus 地址 04xxx。例如，要允许主站访问 2000 字节的 V 存储区，则设置 MaxHold 为 1000 字（保持寄存器）。

参数 HoldStart 是 V 存储区的保持寄存器的起始地址。通常设为 VB0，所以参数 HoldStart 设为 VB0（VB0 的地址）。也可以将其它的 V 区地址指定为保持寄存器的起始地址，以便使 VB0 可以在项目中用作其它目的。Modbus 主站可以访问起始地址为 HoldStart，字数为 MaxHold 的 V 存储区。

当 MBUS_INIT 指令完成时，Done 输出接通。Error 输出字节包含该指令的执行结果。表 13-5 定义了执行该指令可能引起的错误代码。

MBUS_SLAVE 指令

MBUS_SLAVE 指令用于服务来自 Modbus 主站的请求，必须在每个循环周期都执行，以便检查和响应 Modbus 请求。



当 EN 输入接通时，该指令在每一循环周期内执行。

MBUS_SLAVE 指令无输入参数。

当 MBUS_SLAVE 指令响应 Modbus 请求时 Done 输出接通。如果没有服务的请求，Done 输出会断开。

Error 输出包含该指令的执行结果。该输出只有 Done 接通时才有效。如果 Done 断开，错误代码不会改变。表 13-5 定义了执行该指令可能引起的错误条件。

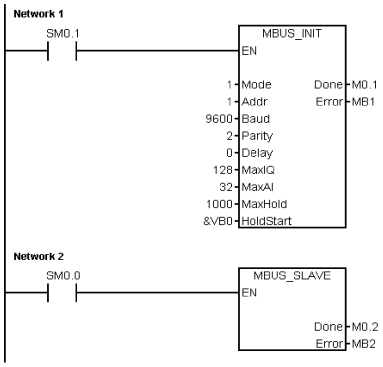
表 13-4　MBUS_SLAVE 指令的参数

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

表 13-5　Modbus 从站协议执行错误代码

错误代码	描述
0	无错误
1	存储区范围错误
2	非法波特率或校验
3	非法从站地址
4	Modbus 参数的非法值
5	保持寄存器与 Modbus 从站符号地址重叠
6	接收校验错误
7	接收 CRC 错误
8	非法功能请求/不支持的功能
9	请求中有非法存储区地址
10	从站功能未始能。

Modbus 从站协议编程示例




```
Network 1
        // 在第一循环扫描中初始化Modbus从站协议。设置从站地址为1，设置port0为9600，偶检验，可以访问所有的I、Q和AI，允许访问1000个保持寄存器(2000字节)，起始地址为VB0。
LD      SM0.1
CALL    MBUS_INT,1,1,9600,2,0,128,32,1000,
        &VB0,M0.1,MB1

Network 2  //每循环周期内执行Modbus从站协议

LD      SM0.0
CALL    MBUS_SLAVE, M0.2, MB2
```

技术规范

A

本章概述:

表 A-1　技术规范

环境条件—运输和存贮	
IEC 68-2-2，Test Bb，干热 IEC 68-2-1，Test Ab，低温	-40℃～＋70℃
IEC 68-2-30，Test Db，湿热	25℃～55℃，95％湿度
IEC 68-2-31，倒下	100mm，4 次倒下，未包装
IEC 68-2-32，自由落下	1m，5 次，运输包装
环境条件—工作	
控制柜温度范围 （单元下部 25mm 进入的空气）	0℃～55℃水平安放 0℃～45℃垂直安放 95％非冷凝湿度
IEC 68-2-14，Test Nb	5℃～55℃，3℃/分钟
IEC 68-2-27 机械震动	15G，11ms 脉冲，每轴向（3 轴）震动 6 次
IEC 68-2-6 正弦波振动	面板安装：峰-峰值 0.30mm，频率 10~57Hz；2G 频率 57~150Hz
	导轨安装：0.15mm，频率 10~57Hz；1G 57 Hz~150Hz；每轴向 10 次振动，1 倍频程/分。
EN 60529，IP20 机械保护	防止高压指状物接触设备。需要外部保护，以防止灰尘、污物、水和直径小于 12.5mm 的异物造成破坏。
电磁兼容性—抗干扰按照 EN61000-6-2	
EN 61000-4-2 静电放电	对所有的面和通讯接口 8kV 空气放电 对暴露的传导面 4kV 接触放电
EN 61000-4-3 辐射电磁场	80MHz~1GHz 10 V/m，用 1kHz 信号 80%调制
EN 61000-4-11 电压波动，短暂干扰和电压变化	对 8.3ms，83ms，833ms 和 4167ms，>95%减少
EN 61000-4-6 传导干扰	0.15~80 MHz 10 V RMS 1kHz 下 80%调幅
EN 61000-4-4 瞬间冲击	对 AC 和 DC 电源系统的连接网络，2kV，5kHz； 对数字量 I/O 和通讯口的连接端子，2kV，5kHz
EN 61000-4-5 浪涌防护	2kV 非对称，1kV 对称 5 正/5 负脉冲，0°，＋90°，- 90°相角 （24 VDC 电路要求外部浪涌保护）
VDE 0160 非周期过电压	对 85 VAC 线，90°相角，允许峰值 390V，1.3ms 脉冲 对 180VAC 线，90°相角，允许峰值 750V，1.3ms 脉冲
电磁兼容性—传导和辐射按照 EN50081 –1[2] 和-2	
EN 55011，Class A，Group 1，传导[1]	
0.15 MHz~0.5 MHz	<79 dB（μV）准峰值；< 66 dB（μV）平均值
0.5 MHz~5 MHz	<73 dB（μV）准峰值；< 60 dB（μV）平均值
5 MHz~30 MHz	<73 dB（μV）准峰值；< 60 dB（μV）平均值
EN 55011，Class A，Group 1，辐射[1]	
30 MHz～230 MHz	30 dB（μV/m）准峰值；30 米测量
230 MHz～1 GHz	37 dB（μV/m）准峰值；30 米测量
EN 55011，Class B，Group 1，传导[2]	
0.15~0.5 MHz	<66 dB (μV) 准峰值按对数频率减少到 56 dB (μV)； <56 dB (μV) 准峰值按对数频率减少到 46 dB (μV)
0.5 MHz~5 MHz	<56 dB（μV）准峰值；< 46 dB（μV）平均值
5 MHz~30 MHz	<60 dB（μV）准峰值；< 50 dB（μV）平均值

EN 55011，Class B，Group 1，辐射 [2]	
30 MHz～230 kHz	30 dB（μV/m）准峰值；10 米测量
230 MHz～1 GHz	37 dB（μV/m）准峰值；10 米测量
高压绝缘测试	
24 V/5 V 额定值电路	500 VAC (光电隔离限制)
115/230 V 电路对地	1,500 VAC
115/230 V 电路对 115/230V 电路	1,500 VAC
230 V 电路对 24 V/5 V 电路	1,500 VAC
115 V 电路对 24V/5 V 电路	1,500 VAC

[1] S7-200 必须安装在接地金属架上，并将其地线直接连到接地金属架上。电缆沿金属架布线。

[2] 设备必须安装在接地的金属壳中。AC 输入电源必须接有一个 EPCOS B84115—E—A30 滤波器或等效设备。滤波器和 S7-200 间的导线不能超过 25cm。24VDC 供电线和传感器供电线必须屏蔽。

CPU 规范

表 A-2　CPU 定货号

定货号	CPU 模板	CPU 供电（常量）	CPU 输入	CPU 输出	可拆卸连接器
6ES7 211-0AA22-0XB0	CPU 221	24 VDC	6 x 24 VDC	4 x 24 VDC	否
6ES7 211-0BA22-0XB0	CPU 221	120 至 240 VAC	6 x 24 VDC	4 x 继电器	否
6ES7 212-1AB22-0XB0	CPU 222	24 VDC	8 x 24 VDC	6 x 24 VDC	否
6ES7 212-1BB22-0XB0	CPU 222	120 至 240 VAC	8 x 24 VDC	6 x 继电器	否
6ES7 214-1AD22-0XB0	CPU 224	24 VDC	14 x 24 VDC	10 x 24 VDC	是
6ES7 214-1BD22-0XB0	CPU 224	120 至 240 VAC	14 x 24 VDC	10 x 继电器	是
6ES7 216-2AD22-0XB0	CPU 226	24 VDC	24 x 24 VDC	16 x 24 VDC	是
6ES7 216-2BD22-0XB0	CPU 226	120 至 240 VAC	24 x 24 VDC	16 x 继电器	是
6ES7 216-2AF22-0XB0	CPU 226XM	24 VDC	24 x 24 VDC	16 x 24 VDC	是
6ES7 216-2BF22-0XB0	CPU 226XM	120 至 240 VAC	24 x 24 VDC	16 x 继电器	是

表 A-3　CPU 通用规范

定货号	模板名称和描述	尺寸(毫米)	重量	损耗	电流供应	
					+5VDC	+24VDC
6ES7 211-0AA22-0XB0	CPU 221 DC/DC/DC 6 输入/4 输出	90 x 80 x 62	270 g	3 W	0 mA	180 mA
6ES7 211-0BA22-0XB0	CPU 221 AC/DC/ Relay 6 输入/4 继电器	90 x 80 x 62	310 g	6 W	0 mA	180 mA
6ES7 212-1AB22-0XB0	CPU 222 DC/DC/DC 8 输入/6 输出	90 x 80 x 62	270 g	5 W	340 mA	180 mA
6ES7 212-1BB22-0XB0	CPU 222 AC/DC/ Relay 8 输入/6 继电器	90 x 80 x 62	310 g	7 W	340 mA	180 mA
6ES7 214-1AD22-0XB0	CPU 224 DC/DC/DC 14 输入/10 输出	120.5 x 80 x 62	360 g	7 W	660 mA	280 mA
6ES7 214-1BD22-0XB0	CPU 224 AC/DC/ Relay 14 输入/10 继电器	120.5 x 80 x 62	410 g	10 W	660 mA	280 mA
6ES7 216-2AD22-0XB0	CPU 226 DC/DC/DC 24 输入/16 输出	196 x 80 x 62	550 g	11 W	1000 mA	400 mA
6ES7 216-2BD22-0XB0	CPU 226 AC/DC/ Relay 24 输入/16 继电器	196 x 80 x 62	660 g	17 W	1000 mA	400 mA
6ES7 216-2AF22-0XB0	CPU 226XM DC/DC/DC 24 输入/16 输出	196 x 80 x 62	550 g	11 W	1000 mA	400 mA
6ES7 216-2BF22-0XB0	CPU 226XM AC/DC/Relay 24 输入/16 继电器	196 x 80 x 62	660 g	17W	1000 mA	400 mA

表 A-4　CPU 规范

	CPU 221	CPU 222	CPU 224	CPU 226	CPU 226XM
存储器					
用户程序空间	2048 字		4096 字	4096 字	**8192 字**
用户数据(EEPROM)	1024 字(永久存储)		2560 字 (永久存储)	2560 字 (永久存储)	5120 字 (永久存储)
装备(超级电容) (可选电池)	50 小时/典型值(40℃时最少 8 小时) 200 天/典型值		190 小时/典型值(40℃时最少 120 小时) 200 天/典型值		
I/O					
本机数字输入/输出	6 输入/4 输出	8 输入/6 输出	14 输入/10 输出	24 输入/16 输出	
数字 I/O 映象区	256(128 入/128 出)				
模拟 I/O 映象区	无	32(16 入/16 出)	64(32 入/32 出)		
允许最大的扩展模块	无	2 模块	7 模块		
允许的最大的智能模块	无	2 模块	7 模块		
脉冲捕捉输入	6	8	14		
高速计数 单相 两相	4 个计数器 4 个 30kHz 2 个 20kHz		6 个计数器 6 个 30kHz 4 个 20kHz		
脉冲输出	2 个 20 kHz(仅限于 DC 输出)				
常规					
定时器	256 定时器；4 个定时器(1ms)；16 定时器(10 ms)；236 定时器(100 ms)				
计数器	256(由超级电容或电池备份)				
内部存储器位 掉电保存	256(由超级电容或电池备份) 112(存储在 EEPROM)				
时间中断	2 个 1ms 分辨率				
边沿中断	4 个上升沿和/或 4 个下降沿				
模拟电位器	1 个 8 位分辨率		2 个 8 位分辨率		
布尔量运算执行速度	0.37μs 每条指令				
时钟	可选卡件		内置		
卡件选项	存储卡、电池卡和时钟卡		存储卡和电池卡		
集成的通讯功能					
接口	一个 RS-485 口				两个 RS-485 口
PPI，DP / T 波特率	9.6、19.2、187.5 K 波特				
自由口波特率	1.2K—115.2 K 波特				
每段最大电缆长度	使用隔离的中继器：187.5K 波特可达 1000 米，38.4 K 波特可达 1200 米 未使用隔离中继器：50 米				
最大站点数	每段 32 个站，每个网络 126 个站				
最大主站数	32				
点到点(PPI 主站模式)	是（NETR / NETW）				
MPI 连接	共 4 个，2 个保留（1 个给 PG，1 个给 OP）				

表 A-5　CPU 电源规范

DC			AC	
输入电源				
输入电压	20.4-28.8 VDC		**85-264 VAC(47-63 Hz)**	
输入电流	仅 CPU， 24 VDC	最大负载 24 VDC	仅 CPU	最大负载
CPU 221	80 mA	450 mA	30/15 mA 120/240 VAC	120/240 VAC 时 120/60 mA
CPU 222	85 mA	500 mA	40/20 mA 120/240 VAC	120/240 VAC 时 140/70 mA
CPU 224	110 mA	700 mA	60/30 mA 120/240 VAC	120/240 VAC 时 200/100 mA
CPU 226/CPU 226XM	150 mA	1050 mA	80/40 mA 120/240 VAC	120/240 VAC 时 320/160 mA
冲击电流	28.8 VDC 时 10 A		264 VAC 时 20 A	
隔离（现场与逻辑）	不隔离		1500　VAC	
保持时间（掉电）	10 ms，24 VDC		20/80 ms，120/240 VAC	
保险（不可替换）	3A，250 V 慢速熔断		2 A，250　V 慢速熔断	
24 VDC 传感器电源				
传感器电压	L+减 5V		20.4-28.8 VDC	
电流限定	1.5A 峰值，终端限定非破坏性			
纹波噪声	来自输入电源		小于 1V 峰分值	
隔离(传感器与逻辑)	非隔离			

表 A-6　CPU 输入规范

常规	24　VDC 输入	
类型	漏型/源型(IEC 类型 1 漏型)	
额定电压	24VDC，4mA 典型值	
最大持续允许电压	30 VDC	
浪涌电压	35 VDC，0.5s	
逻辑 1（最小）	15 VDC，2.5mA	
逻辑 0（最大）	5 VDC 1 mA	
输入延迟	可选(0.2 至 12.8ms) CPU 226，CPU 226XM：输入点 I1.6 至 I2.7 具有固定延迟(4.5ms)	
连接 2 线接近开关传感器(Bero) 允许漏电流（最大）	 1mA	
隔离(现场与逻辑) 光电隔离 隔离组	是 500 VAC，1 分钟 见接线图	
高速输入速率(最大) 逻辑 1=15 至 30 VDC 逻辑 1=15 至 26 VDC	单相 20 kHz 30 kHz	两相 10 kHz 20 kHz
同时接通的输入	55℃时所有的输入	
电缆长度(最大) 屏蔽 非屏蔽	 普通输入 500 米，HSC 输入 50 米 普通输入 300 米	

表 A-7　CPU 输出规范

常规	24　VDC 输出	继电器输出
类型	固态－MOSFET[1]	干触点
额定电压	24 VDC	24 VDC 或 250 VAC
电压范围	20.4 至 28.8VDC	5 至 30 VDC 或 5 至 250 VAC
浪涌电流(最大)	8A，100ms	7A 触点闭合
逻辑 1(最小)	20 VDC，最大电流	-
逻辑 0(最大)	0.1 VDC，10KΩ 负载	-
每点额定电流(最大)	0.75A	2.0A
每个公共端的额定电流(最大)	6A	10A
漏电流(最大)	10μA	-
灯负载(最大)	5W	30W DC；200W AC
感性嵌位电压	L＋减 48 VDC，1W 功耗	-
接通电阻(接点)	0.3Ω 最大	0.2Ω(新的时候的最大值)
隔离 光电隔离(现场到逻辑) 逻辑到接点 接点到接点 电阻(逻辑到接点) 隔离组	 500 VAC，1 分钟 - - - 见接线图	 - 1500 VAC，1 分钟 750 VAC，1 分钟 100MΩ 见接线图
延时 断开到接通/接通到断开(最大) 切换(最大)	2/10 μs (Q0.0 和 Q0.1) 15/100μs (其它) -	- 10ms
脉冲频率(最大)Q0.0 和 Q0.1	20　kHz	1 Hz
机械寿命周期	-	10,000,000(无负载)
触点寿命	-	100,000(额定负载)
同时接通的输出	55℃时，所有的输出	55℃时，所有的输出
两个输出并联	是	否
电缆长度(最大) 屏蔽 非屏蔽	 500m 150m	 500m 150m

[1] 当一个机械触点接通 S7-200 CPU 或任意扩展模块的供电时，它发送一个大约 50 毫秒的“1”信号到数字输出，您需要考虑这一点，尤其是您使用触够响应短脉冲的设备时。

接线图

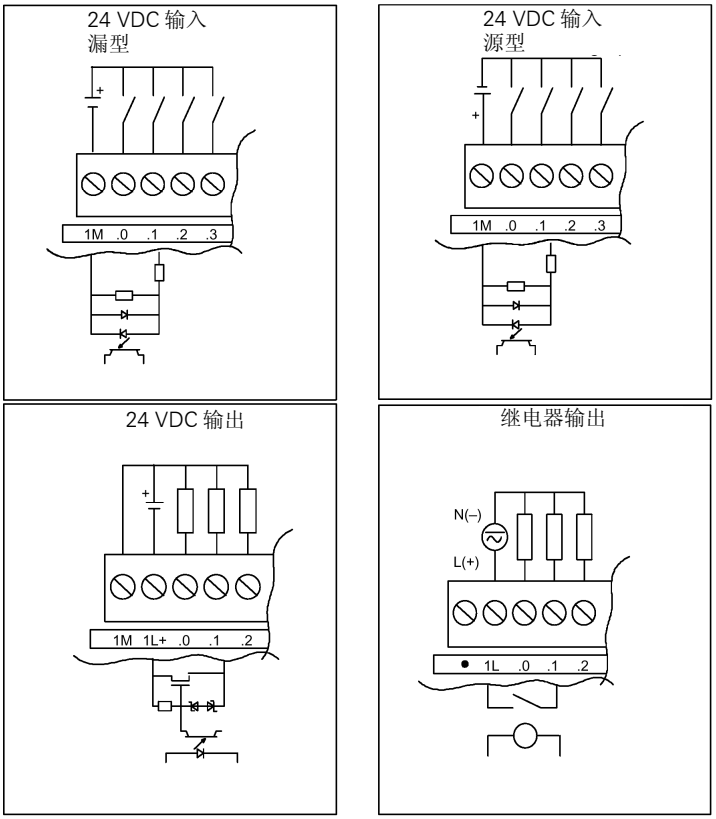


图 A-2　CPU 输入和输出

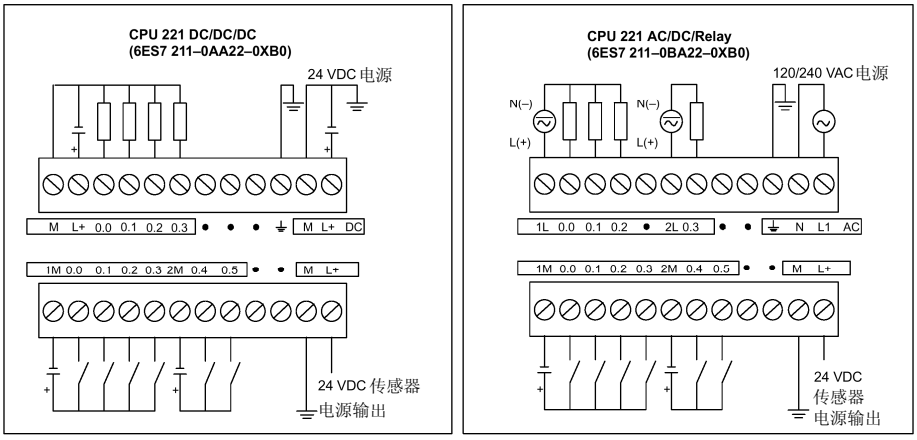


图 A-3　CPU 221 接线图

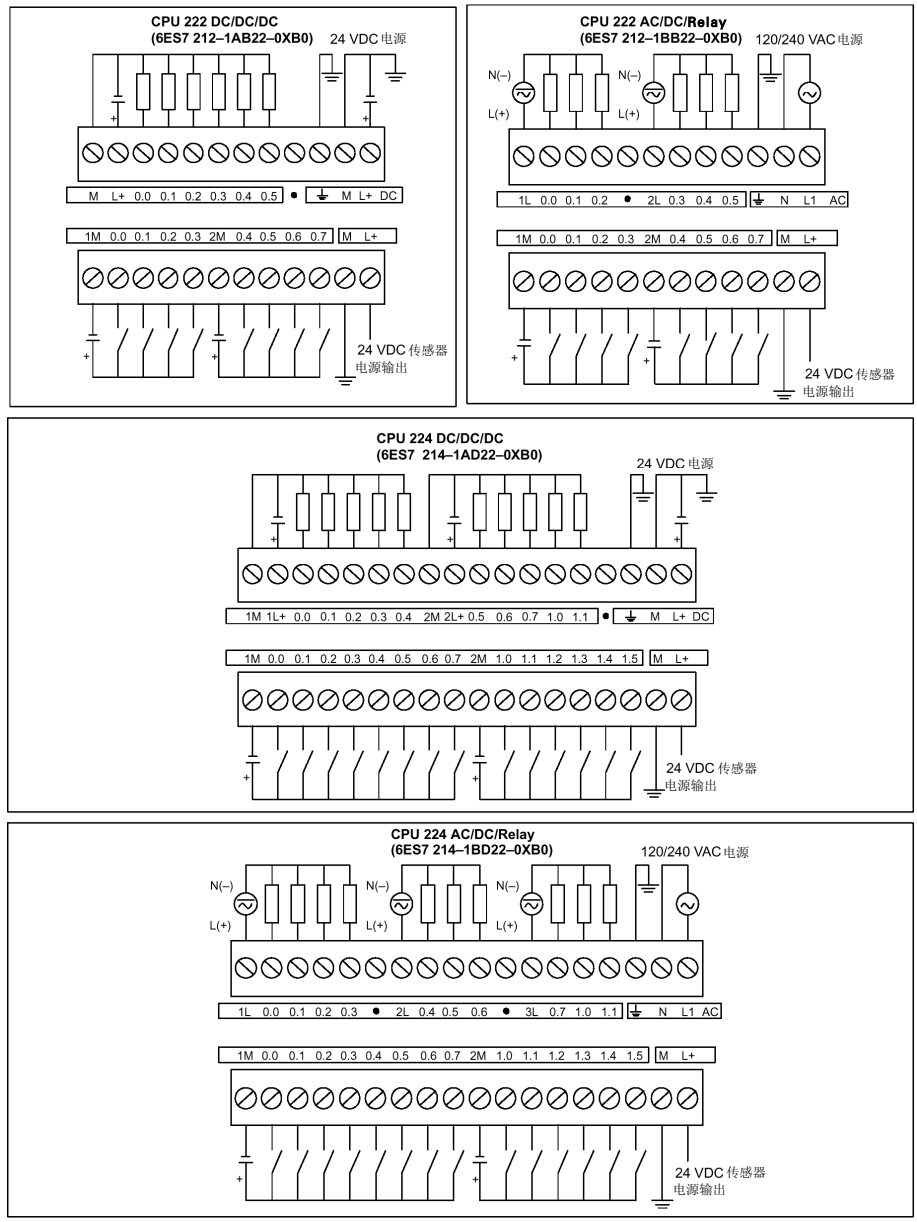


图 A-4　CPU 222 和 CPU 224 接线图

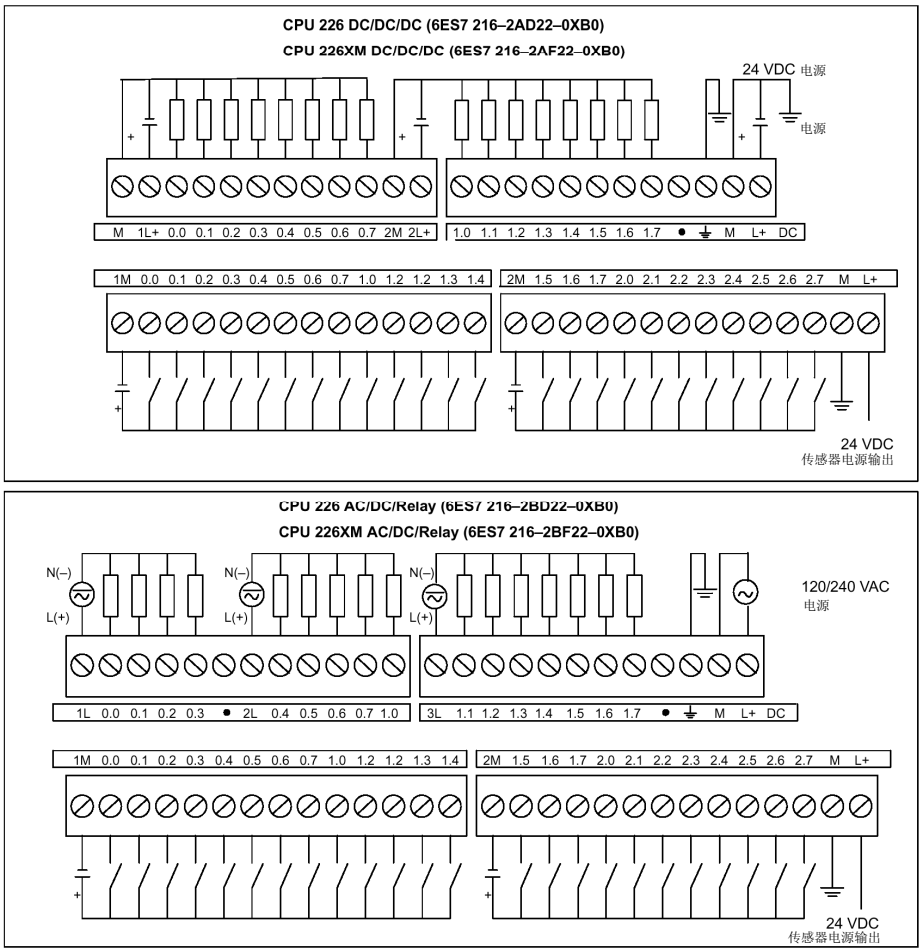
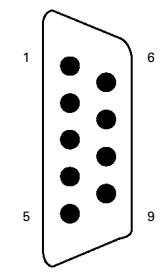


图 A-5　CPU 226 和 CPU 226XM 接线图

图 A-8　S7-200 通讯口针脚分配

连接器	针脚	信号	口 0/口 1
	1	屏蔽	机壳地
	2	24 V 返回	逻辑地
	3	RS-485 信号 B	RS－485 信号 B
	4	请求发送	RTS(TTL)
	5	5 V 返回	逻辑地
	6	＋5V	+5V，100Ω串行电阻
	7	＋24V	＋24V
	8	RS-485 信号 A	RS-485 信号 A
	9	未用	选择 10 位协议(输入)
	连接器外壳	屏蔽	机壳地

数字扩展模块规范

表 A-9　数字扩展模块定货号

定货号	扩展模块	EM 输入	EM 输出	可拆卸连接器
6ES7 221–1BF22–0XA0	EM 221 数字输入 8 x 24 VDC	8 x 24 VDC	-	是
6ES7 221–1EF22–0XA0	EM 221 数字输入 8 x AC 120/230V	8 x AC 120/230V	-	是
6ES7 222–1BF22–0XA0	EM 222 数字输出 8 x 24 VDC	-	8 x 24 VDC	是
6ES7 222–1HF22–0XA0	EM 222 数字输出 8 x 继电器	-	8 x 继电器	是
6ES7 222–1EF22–0XA0	EM 222 数字输出 8 x AC 120/230 V	-	8 x AC 120/230V	是
6ES7 223–1BF22–0XA0	EM 223 24 VDC 数字混合模块 4 输入/4 输出	4 x 24 VDC	4 x 24 VDC	是
6ES7 223–1HF22–0XA0	EM 223 24 VDC 数字混合模块 4 输入/4 继电器输出	4 x 24 VDC	4 x 继电器	是
6ES7 223–1BH22–0AX0	EM 223 24 VDC 数字混合模块 8 输入/8 输出	8 x 24 VDC	8 x 24 VDC	是
6ES7 223–1PH22–0XA0	EM 223 24 VDC 数字混合模块 8 输入/8 继电器输出	8 x 24 VDC	8 x 继电器	是
6ES7 223–1BL22–0XA0	EM 223 24 VDC 数字混合模块 16 输入/16 输出	16 x 24 VDC	16 x 24 VDC	是
6ES7 223–1PL22–0XA0	EM 223 24 VDC 数字混合模块 16 输入/16 继电器	16 x 24 VDC	16 x 继电器	是

表 A-10　数字扩展模块通用规范

定货号	模块名称和描述	尺寸 (毫米) (W x H x D)	重量	损耗	VDC 要求 +5 VDC	 +24 VDC
6ES7 221-1BF22-0XA0	EM 221 DI 8 x 24 VDC	46 x 80 x 62	150 g	2 W	30 mA	-
6ES7 221-1EF22-0XA0	EM 221 DI 8 x AC 120/230 V	71.2 x 80 x 62	160 g	3 W	30 mA	-
6ES7 222-1BF22-0XA0	EM 222 DO 8 x 24 VDC	46 x 80 x 62	150 g	2 W	50 mA	-
6ES7 222-1HF22-0XA0	EM 222 DO 8 x 继电器	46 x 80 x 62	170 g	2 W	40 mA	接通：9mA/输出 20.4 至 28.8VDC
6ES7 222-1EF22-0XA0	EM 222 DO 8 x AC 120/230 V	71.2 x 80 x 62	165 g	4 W	110 mA	-
6ES7 223-1BF22-0XA0	EM 223 24 VDC 4 入/4 出	46 x 80 x 62	160 g	2 W	40 mA	-
6ES7 223-1HF22-0XA0	EM 223 24 VDC 4 入/4 继电器	46 x 80 x 62	170 g	2 W	40 mA	接通：9mA/输出 20.4 至 28.8VDC
6ES7 223-1BH22-0AX0	EM 223 24 VDC 8 入/8 出	71.2 x 80 x 62	200 g	3 W	80 mA	-
6ES7 223-1PH22-0XA0	EM 223 24 VDC 8 入/8 继电器	71.2 x 80 x 62	300 g	3 W	80 mA	接通：9mA/输出 20.4 至 28.8VDC
6ES7 223-1BL22-0XA0	EM 223 24 VDC 16 入/16 出	137.3 x 80 x 62	360 g	6 W	160 mA	-
6ES7 223-1PL22-0XA0	EM 223 24 VDC 16 入/16 继电器	137.3 x 80 x 62	400 g	6 W	150 mA	接通：9mA/输出 20.4 至 28.8VDC

表 A-11　数字扩展模块输入规范

常规	24　VDC 输入	120/230 VAC 输入(47 至 63 Hz)
类型	漏型/源型(IEC 类型 1 漏型)	IEC 类型 1
额定电压	24 VDC，4 mA	120 VAC，6 mA 或 230 VAC， 9 mA(通常)
最大持续允许电压	30 VDC	264 VAC
浪涌电压(最大)	35 VDC，0.5s	-
逻辑 1(最小)	15 VDC，2.5 mA	79 VAC，2.5mA
逻辑 0(最大)	5 VDC，1 mA	20 VAC 或 1mA AC
输入延时(最大)	4.5 ms	15 ms
连接 2 线接近开关传感器(Bero) 允许的漏电流(最大)	1 mA	1 mA AC
隔离 光电隔离(现场到逻辑) 隔离组	 500 VAC，1 分钟 见接线图	 1500 VAC 1 分钟 1 点
同时接通的输入	55℃时所有输入	55℃时所有输入
电缆长度(最大) 屏蔽 非屏蔽	 500m 300m	 500m 300m

表 A-12　数字扩展模块输出规范

常规	24 VDC 输出	继电器输出	120/230 VAC 输出
类型	MOSFET[1]	干触点	直通
额定电压	24 VDC	24 VDC 或 250 VAC	120/230 VAC
电压范围	20.4 至 28.8 VDC	5 至 30 VDC 或 5 至 250 VAC	40 至 264 VAC(47 至 63Hz)
24 VDC 线圈电源电压范围	-	20.4 至 28.8 VDC	-
浪涌电流(最大)	8 A，100 ms	7 A，触点闭合	5A rms，2AC 周期
逻辑 1(最小)	20 VDC	-	L1(-0.9V rms)
逻辑 0(最大)	0.1 VDC	-	-
额定电流/每点(最大)	0.75 A	2.00A	0.5A AC[3]
额定电流/每个公共点(最大)	6 A	8A	0.5A AC
漏电流(最大)	10 μA	-	1.1mA rms，132 VAC 以及 1.8mA rhesus，264 VAC
灯负载(最大)	5 W	30W DC/200 W AC	60W
感性钳位电压	L＋减 48V	-	-
接通状态电阻(触点)	0.3Ω(最大)	0.2Ω最大，新的时候	410Ω最大，当负载电流小于 0.05A 时
隔离 光电隔离(现场到逻辑) 线圈到逻辑 线圈到触点 触点到触点 电阻(线圈到触点) 隔离组	 500 VAC，1 分钟 - - - - 见接线图	 - 无 1500 VAC，1 分钟 750 VAC，1 分钟 100MΩ最小，新的时候 4 点	 1500VAC，1 分钟 - - - - 1 点
延时 断开到接通/接通到断开 切换 (最大)	 50μs 最大/200μs -	 - 10ms	 0.2ms＋1/2 AC 周期 -
切换频率(最大)	-	1Hz	10Hz
机械寿命周期	-	10,000,000(无负载)	-
触点寿命	-	100,000(额定负载)	-
同时接通的输出	55℃时所有输出	55℃时所有输出	55℃时所有输出
并联两个输出	是	否	否
电缆长度(最大) 屏蔽 非屏蔽	 500 m 150 m	 500 m 150 m	 500 m 150 m

1　当一个机械触点接通 S7-200 CPU 或任意扩展模块的供电时，它发送一个大约 50 毫秒的“1”信号到数字输出，您需要考虑这一点，尤其是您使用能够响应短脉冲的设备时。

2　当一个机械触点接通 AC 扩展模块的输出电源时，它向 AC 输出发出一个宽度为大约 1/2AC 周期的“1”信号。您必须考虑这一点。

3　由于是直通电路，负载电流必须是完整的 AC 波型而非半波。最小负载电流是 0.05A AC。当负载电流在 5mA 和 50mA AC 之间时，该电流是可控的，但是，由于 410 欧姆串行电阻的存在会有额外的压降。

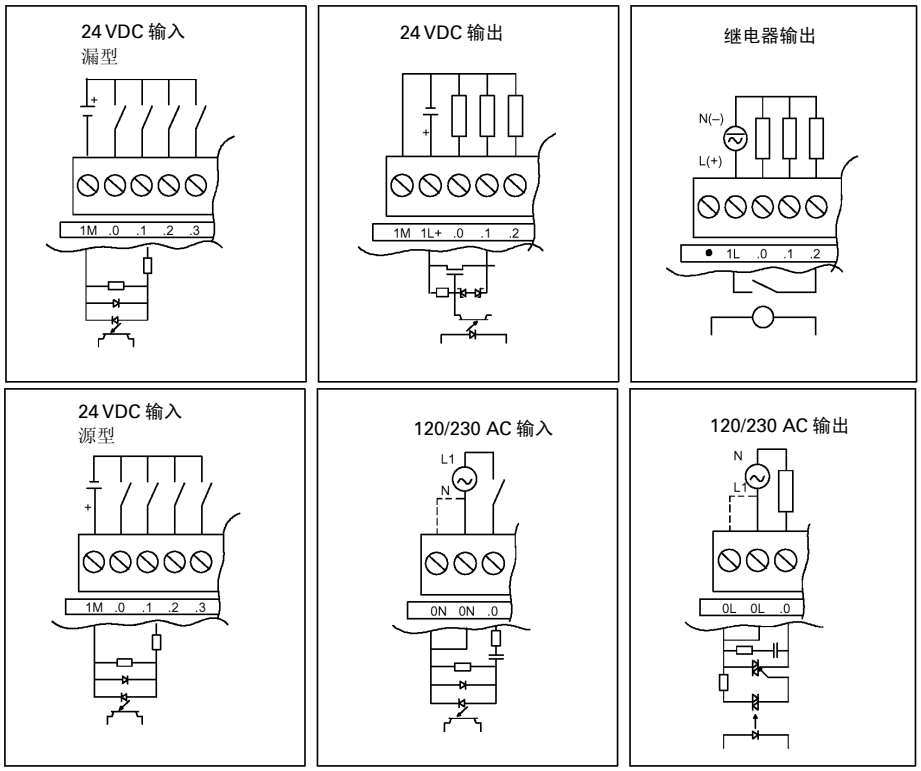


图 A-6　S7-200 数字扩展模块输入和输出

接线图

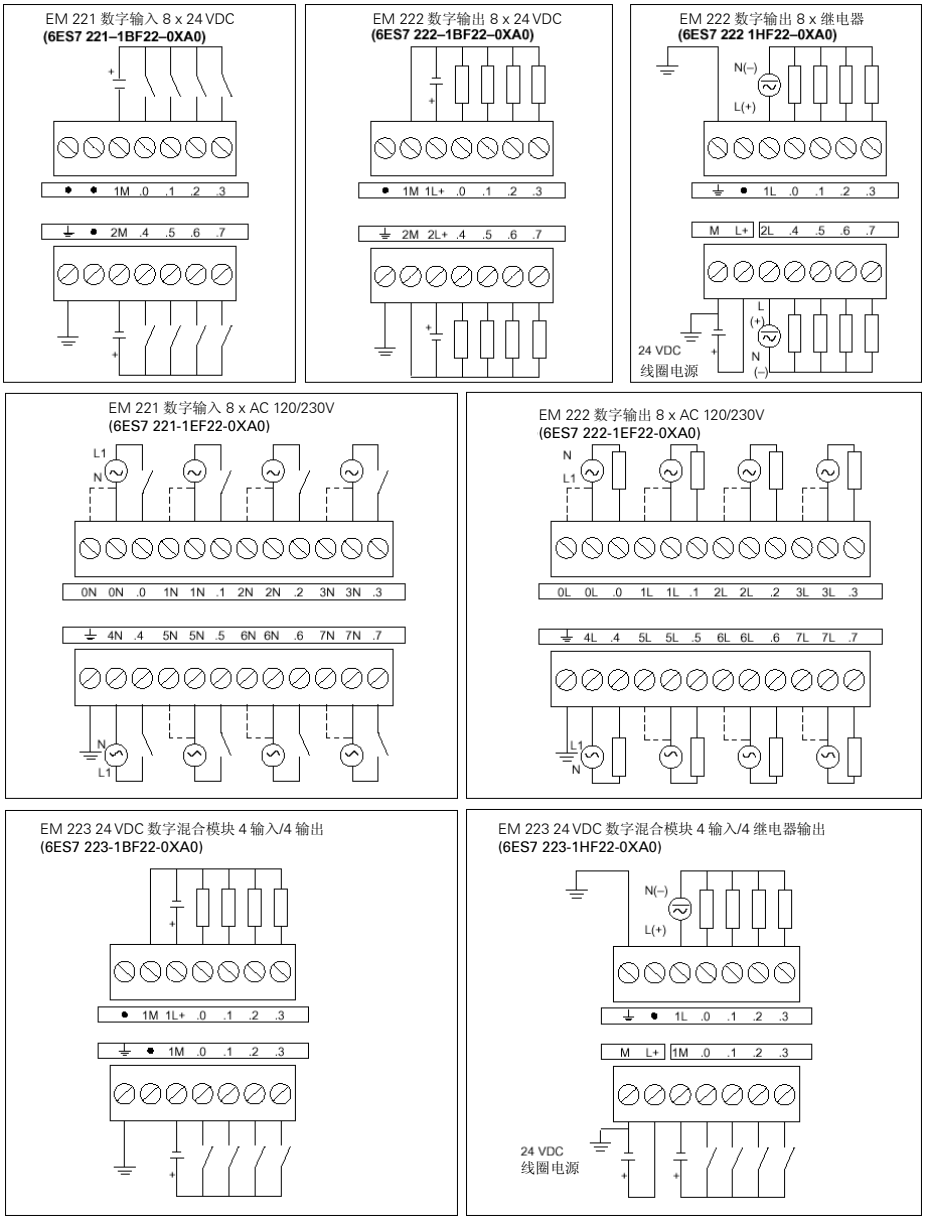


图 A-7　EM 221，EM 222，和 EM 223 扩展模块接线图

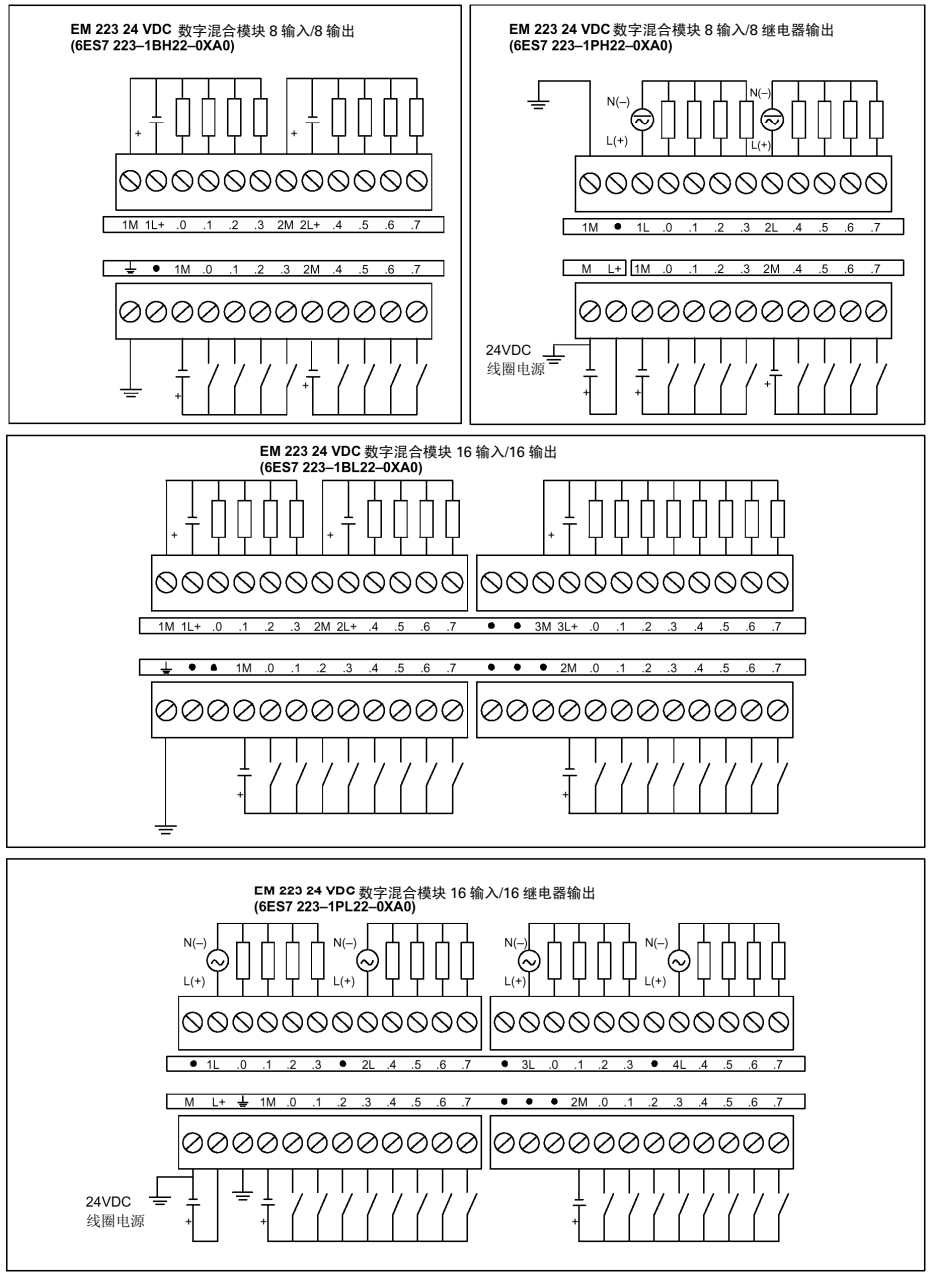


图 A-8　EM 223 扩展模块接线图

模拟量扩展模块规范

表 A-13　模拟扩展模块定货号

定货号	扩展模块	EM 输入	EM 输出	可拆卸连接器
6ES7 231-0HC22-0XA0	EM 231 模拟输出，4 输入	4	-	否
6ES7 232-0HB22-0XA0	EM 232 模拟输出，2 输出	-	2	否
6ES7 235-0KD22-0XA0	EM 235 模拟量混合模块 4 输入/1 输出	4	1[1]	否

[1]　CPU 为该模块保留 2 个模拟输出通道

表 A-14　模拟扩展模块通用规范

定货号	模块名称和描述	尺寸(毫米) (W x H x D)	重量	损耗	VDC 要求 +5 VDC	VDC 要求 +24 VDC
6ES7 231-0HC22-0XA0	EM 231 模拟输入，4 输入	71.2 x 80 x 62	183g	2W	20mA	60mA
6ES7 232-0HB22-0XA0	EM 232 模拟输出，2 输出	46 x 80 x 62	148g	2W	20mA	70mA(两 个 输 出都是 20mA)
6ES7 235-0KD22-0XA0	EM 235 模拟量混合模块 4 输入/1 输出	71.2 x 80 x 62	186g	2W	30mA	60mA(输 出 为 20mA)

表 A-15　模拟扩展模块输入规范

常规	6ES7 231-0HC22-0XA0	6ES7 235-0KD22-0XA0
数据字格式 双极性，满量程 单极性，满量程	(见图 A-11) -32000 至+32000 0 至 32000	(见图 A-11) -32000 至+32000 0 至 32000
DC 输入阻抗	≥10MΩ 电压输入 250Ω 电流输入	≥10MΩ 电压输入 250Ω 电流输入
输入滤波衰减	-3 db，3.1Khz	-3 db，3.1Khz
最大输入电压	30 VDC	30 VDC
最大输入电流	32 mA	32 mA
分辨率	12 位 A/D 转换器	12 位 A/D 转换器
隔离(现场到逻辑)	否	否
输入类型	差分	差分
输入范围 电压(单极性) 电压(双极性) 电流	 0 至 10V，0 至 5V ±5V，±2.5V 0 至 20mA	 0 至 10V，0 至 5V 0 至 1V，0 至 500mV 0 至 100mV，0 至 50mV ±10V，±5V, ±2.5V，±1V，±500mV， ±250mV，±100mV，±50mV，±25mV 0 至 20mA
输入分辨率 电压(单极性) 电压(双极性) 电流	见表 A-18	见表 A-19
模拟到数字转换时间	<250μs	<250μs
模拟输入阶跃响应	1.5ms 到 95%	1.5ms 到 95%
共模抑制	40dB，DC 到 60Hz	40dB，DC 到 60Hz
共模电压	信号电压加共模电压必须≤±12V	信号电压加共模电压必须≤±12V
24 VDC 电压范围	20.4 至 28.8	20.4 至 28.8

表 A-16　模拟量扩展模块输出规范

常规	6ES7 232-0HB22-0XA0	6ES7 235-0KD22-0XA0
隔离(现场到逻辑)	无	无
信号范围 　　电压输出 　　电流输出	 ±10V 0 至 20mA	 ±10V 0 至 20mA
分辨率，满量程 　　电压 　　电流	 12 位 11 位	 12 位 11 位
数据字格式 　　电压 　　电流	 -32000 至+32000 0 至+32000	 -32000 至+32000 0 至+32000
精度 最差情况，0° 至 55℃ 　　电压输出 　　电流输出 典型，25℃ 　　电压输出 　　电流输出	 ±2%满量程 ±2％满量程 ±0.5%满量程 ±0.5%满量程	 ±2%满量程 ±2％满量程 ±0.5%满量程 ±0.5%满量程
设置时间 　　电压输出 　　电流输出	 100μs 2mS	 100μs 2mS
最大驱动 　　电压输出 　　电流输出	 5000Ω最小 500Ω最大	 5000Ω最小 500Ω最大

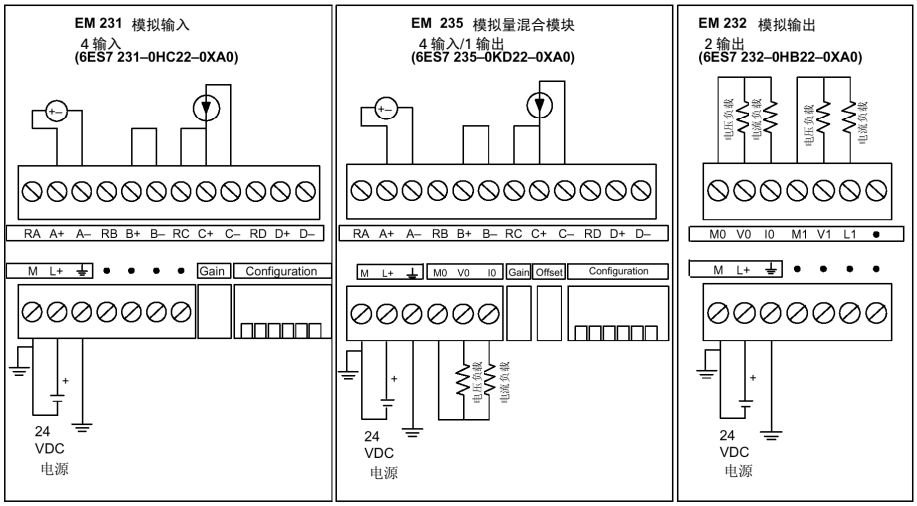


图 A-9　模拟量扩展模块接线图

模拟模块 LED 指示

模拟量模块的 LED 指示，如表 A-17 所示。

表 A-17　模拟量模块 LED 指示

LED 指示	接通	断开
24VDC 电源良好	无故障	没有 24VDC 电源

输入校准

校准调节影响模拟多路转换器运算的放大器（见 EM231 输入方框图图 A-12 和 EM235 的输入方框图图 A-13）。因此，校准影响到所有的用户输入通道。即使在校准后，在模拟多路转换器前面的输入电路中，由于每个输入通道电路元件的参数发生变化，也会使同一输入信号在不同通道之间的读数存在轻微差异。

为了达到表中所列的技术参数，应启动用于模块所有输入的模拟输入滤波器。计算平均值时，选择 64 次或更多的采样次数。

校准输入时，其步骤如下：

1. 切断模块电源。选择需要的输入范围。
2. 接通 CPU 和模块电源。使模块稳定 15 分钟。
3. 用一个变送器、一个电压源或一个电流源，将零值信号加到一个输入端。
4. 读取适当的输入通道在 CPU 中的测量值。
5. 调节 OFFSET（偏置）电位计，直到读数为零，或所需要的数字数据值。
6. 将一个满刻度值信号接到输入端子中的一个。读出送到 CPU 的值。
7. 调节 GAIN（增益）电位计，直到读数为 32000，或所需要的数字数据值。
8. 必要时，重复偏置和增益校准过程。

EM 231 和 EM 235 的校准和配置位置

如图 A-10 所示，校准电位计和配置 DIP 开关位于模块底部端子板右侧。

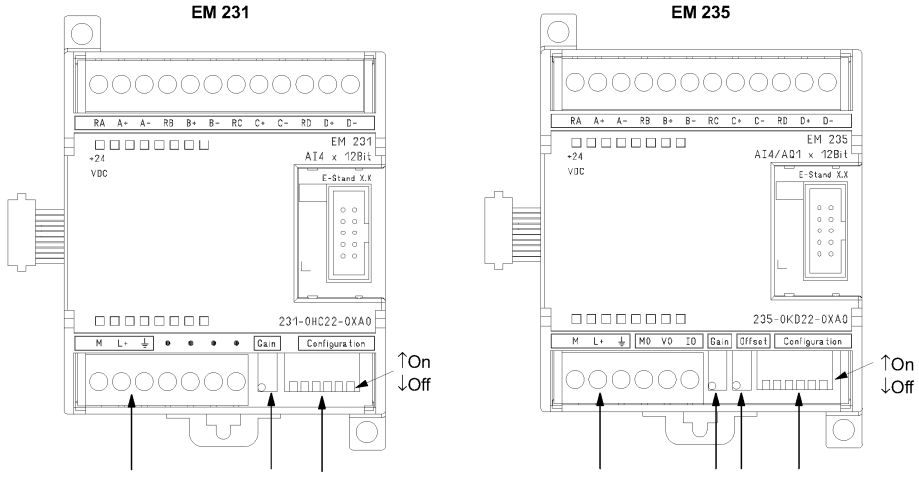


图 A-10　EM 231 和 EM 235 校准电位计和配置 DIP 开关位置

配置 EM 231

表 A-18 所示，使用 DIP 配置开关组态 EM 231。开关 1、2 和 3 选择模拟输出范围。所有输入设置为相同的模拟输入量程。该表中，ON 是闭合，OFF 是断开。

表 A-18　用于选择模拟输入量程的 EM 231 配置开关表

单极性			满量程输入	分辨率
SW1	SW2	SW3		
ON	OFF	ON	0 to 10V	2.5mV
	ON	OFF	0 to 5V	1.25mV
			0 to 20mA	5μA
双极性			满量程输入	分辨率
SW1	SW2	SW3		
OFF	OFF	ON	±5V	2.5mV
	ON	OFF	±2.5V	1.25mV

配置 EM235

表 A-19 所示为如何使用 DIP 配置开关组态 EM 235 模块。开关 1 至 6 可选择输入量程和分辨率。所有的输入都设置为相同的模拟输入量程和格式。表 A-20 所示为如何选择单极性/双极性（开关 6），增益（开关 4 和 5）以及衰减（开关 1，2 和 3），在该表中，ON 是闭合，OFF 是断开。

表 A-19　用于选择模拟量程和分辨率的 EM 235 配置开关表

单极性						满量程输入	分辨率
SW1	SW2	SW3	SW4	SW5	SW6		
ON	OFF	OFF	ON	OFF	ON	0 to 50 mV	12.5μV
OFF	ON	OFF	ON	OFF	ON	0 to 100 mV	25μV
ON	OFF	OFF	OFF	ON	ON	0 to 500 mV	125μV
OFF	ON	OFF	OFF	ON	ON	0 to 1 V	250μV
ON	OFF	OFF	OFF	OFF	ON	0 to 5 V	1.25mV
ON	OFF	OFF	OFF	OFF	ON	0 to 20 mA	5μA
OFF	ON	OFF	OFF	OFF	ON	0 to 10 V	2.5mV
双极性						满量程输入	分辨率
SW1	SW2	SW3	SW4	SW5	SW6		
ON	OFF	OFF	ON	OFF	OFF	±25 mV	12.5μV
OFF	ON	OFF	ON	OFF	OFF	±50 mV	25μV
OFF	OFF	ON	ON	OFF	OFF	±100 mV	50μV
ON	OFF	OFF	OFF	ON	OFF	±250 mV	125μV
OFF	ON	OFF	OFF	ON	OFF	±500 mV	250μV
OFF	OFF	ON	OFF	ON	OFF	±1 V	500μV
ON	OFF	OFF	OFF	OFF	OFF	±2.5 V	1.25mV
OFF	ON	OFF	OFF	OFF	OFF	±5 V	2.5 mV
OFF	OFF	ON	OFF	OFF	OFF	±10 V	5 mV

表 A-20　用于选择单极性/双极性，增益、衰减的 EM 235 配置开关表

EM 235 配置开关						单极性/双极性选择	增益选择	衰减选择
SW1	SW2	SW3	SW4	SW5	SW6			
					ON	单极性		
					OFF	双极性		
			OFF	OFF			X1	
			OFF	ON			X10	
			ON	OFF			X100	
			ON	ON			无效	
ON	OFF	OFF						0.8
OFF	ON	OFF						0.4
OFF	OFF	ON						0.2

EM 231 和 EM 235 输入数据字格式

图 A-11　所示为 CPU 中模拟输入字中的 12 位数据格式。

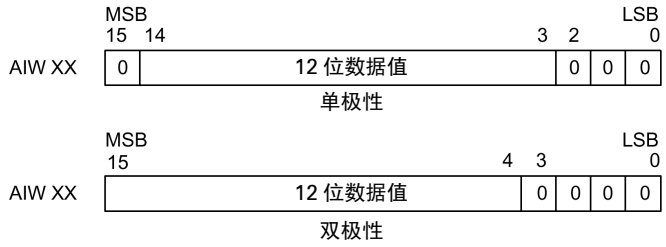


图 A-11　EM 231 和 EM 235 输入数据字格式

提示：

模拟量到数字量转换器的 12 位读数是左对齐的。最高有效位是符号位：0 表示是正值。

在单极性格式中，3 个连续的 0 使得 ADC 计数值每变化 1 个单位，数据字中则以 8 为单位变化。

在双极性格式中，4 个连续的 0 使得 ADC 计数值每变化 1 个单位，数据字中则以 16 为单位变化。

EM 231 和 EM 235 输入方框图

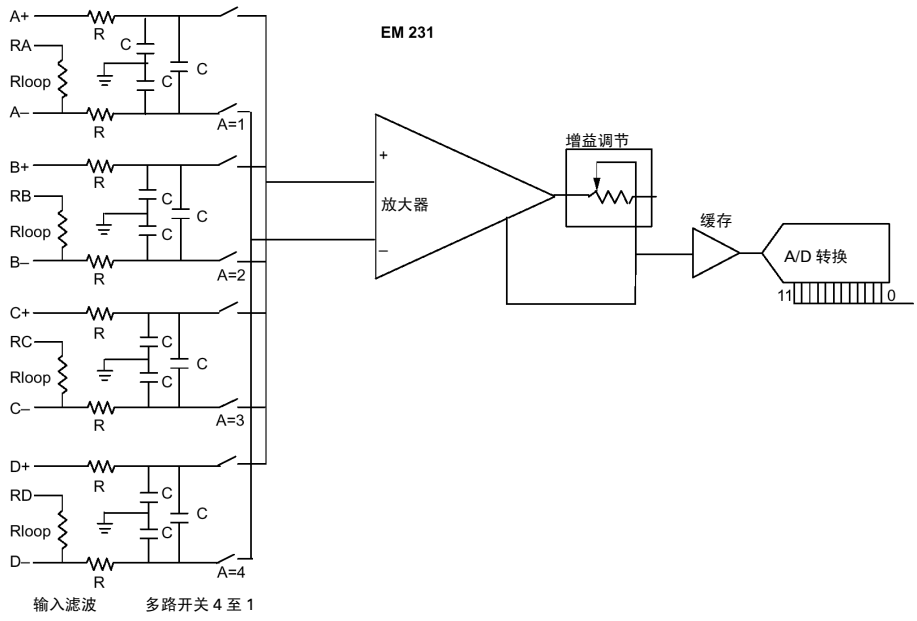


图 A-12　EM 231 输入方框图

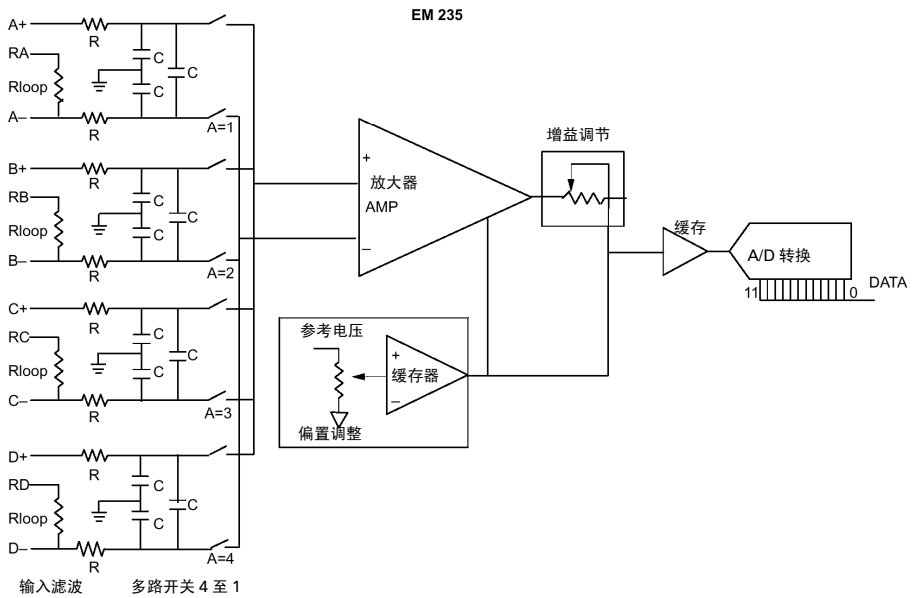


图 A-13　EM 235 输入方框图

EM 231 和 EM 235 输出数据字格式

图 A-14　所示为 CPU 中模拟输出字的 12 位数据格式。

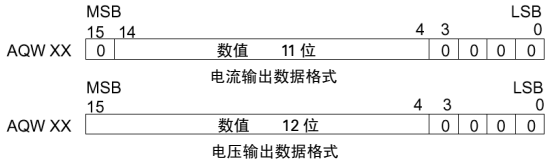


图 A-14　EM 232 和 EM 235 输出数据字格式

提示：

数字量到模拟量转换器（DAC）的 12 位读数在其输出数据格式中是左端对齐的。最高有效位（MSB）是符号位：0 表示正值。数据在装载到 DAC 寄存器之前，4 个连续的 0 是被截断的，这些位不影响输出信号值。

EM 232 和 EM 235 输出方框图

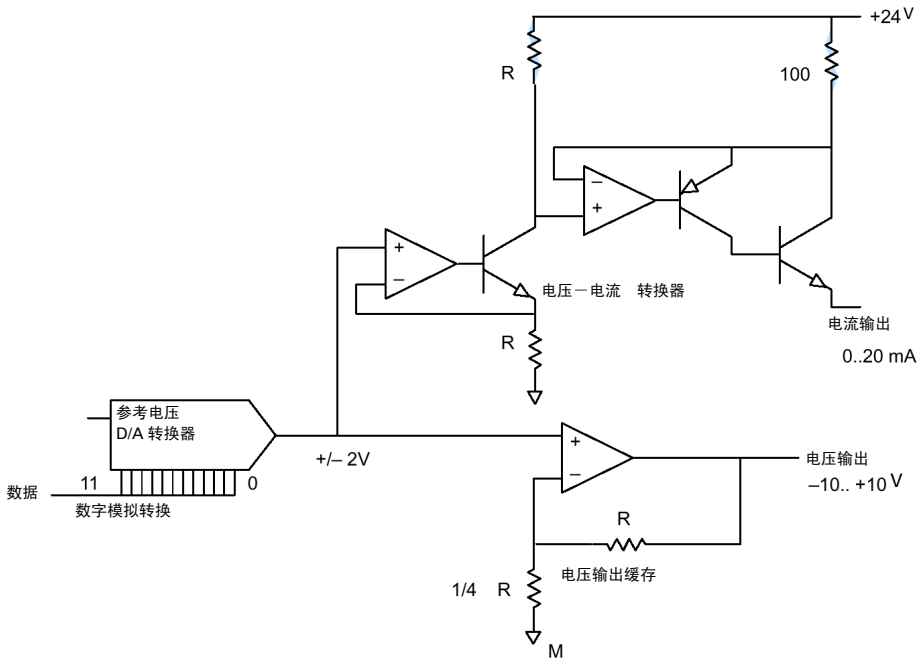


图 A-15　EM 232 和 EM 235 输出方框图

安装指南

采用下列方法确保安装正确、可靠：

- 确保 24 VDC 传感器电源无噪声、稳定。
- 传感器线尽可能短。
- 传感器线使用屏蔽的双绞线。
- 仅在传感器侧将屏蔽接终端。
- 未用通道应短接，见图 A-9。
- 避免将导线弯成锐角。
- 使用电缆槽进行敷线。
- 避免将信号线与高能量线平行布置。若两条线必须重合，应以直角度相交。
- 通过隔离输入信号或输入信号参考于模拟量模块外部 24 V 电源的公共端，从而确保输入信号范围在技术规范所规定的共模电压之内。

注意

建议 EM 231 和 EM 235 扩展模块不用于热电耦。

理解模拟量输入模块：精度和重复性

EM 231 和 EM 235 模拟量输入模块是价格适中、高速 12 位模拟量输入模块。这种模块能在 149μs 内将模拟量输入转换成相应的数字值。模拟信号输入的转换是在每次由程序访问模拟点时完成的。这些时间必须加到执行模拟量输入指令的基本时间上。

EM 231 和 EM 235 提供一个未经处理的数字值（未经线性化或滤波），它对应于模拟量输入端处出现的模拟量电压或电流。由于这种模块是高速模块，它们可以跟踪模拟量信号中的快速变化（包括内部和外部噪声）。对一个恒定或缓慢变化的模拟量输入，由噪声引起的信号读数之间的差异，可通过对读数值取平均值的方法使其影响为最小。但由于计算平均值而增加读取信号的次数（即采样次数），会相应地降低对外部输入信号的响应速度。

图 A-16 为 99%的重复性限定，各个读入值的平均值，以及平均精度的图形示意。

重复性的技术规范描述不改变输入信号时，模块每次读数之间的差异。重复性技术规范规定限制范围，要求 99% 的读数处于这限制范围以内。重复性在图 A-16 中用钟形曲线描述。

平均精度的技术规范描述误差的平均值（各个读数的平均值和实际模拟量输入信号精确值之间的差异）。

表 A-21 给出重复性技术规范和与每个可配置范围有关的平均精度。

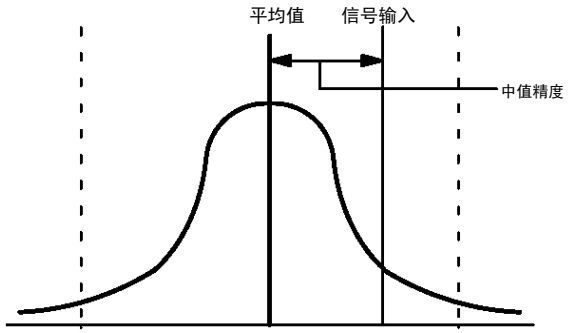


图 A-16　精度定义

模拟量规范的定义

- 精度：在某一给定点上与期望值的偏差。
- 分辨率：输出中所反映的最低位（LSB）的变化。

表 A-21　EM 231 和 EM 235 规范

满量程输入	重复性 [1]		平均精度 [1,2,3,4]	
	%满量程	数值	%满量程	数值
EM 231 规范				
0 to 5 V	±0.075%	±24	±0.1%	±32
0 to 20 mA				
0 to 10 V				
±2.5V		±48	±0.05%	
±5V				
EM 235 规范				
0 to 50mV	±0.075%	±24	±0.25%	±80
0 to 100mV			±0.2%	±64
0 to 500mV			±0.05%	±16
0 to 1V				
0 to 5V				
0 to 20mA				
0 to 10V				
±25mV	±0.075%	±48	±0.25%	±160
±50mV			±0.2%	±128
±100mV			±0.1%	±64
±250mV			±0.05%	±32
±500mV				
±1V				
±2.5V				
±5V				
±10V				

[1] 在所选择的输入量程标定后测量

[2] 接近零模拟量输入信号的偏置误差不予纠正，也不包括在精度规范内。

[3] 由于模拟量多路转换器设定时间的限制存在通道到通道的传递转换误差。最大的传递误差是通道间差分值的 0.1%。

[4] 平均精度包括非线性和 0 到 55℃漂移的影响。

热电偶和 RTD（热电阻）扩展模块规范

表 A-22 热电偶和 RTD 模块定货号

定货号	扩展模块	EM 输入	EM 输出	可拆卸连接器
6ES7 231-7PD22-0XA0	EM 231 模拟输入热电偶，4 输入	4 热电偶	-	否
6ES7 231-7PB22-0XA0	EM 231 模拟输入 RTD，2 输入	2 RTD	-	否

表 A-23 热电偶和 RTD 模块通用规范

定货号	模块名称及描述	尺寸(毫米) (W x H x D)	重量	功耗	VDC 要求 +5 VDC	VDC 要求 +24 VDC
6ES7 231-7PD22-0XA0	EM 231 模拟输入热电偶，4 输入	71.2 x 80 x 62	210g	1.8W	87mA	60mA
6ES7 231-7PB22-0XA0	EM 231 模拟输入 RTD，2 输入	71.2 x 80 x 62	210g	1.8W	87mA	60mA

表 A-24 热电偶和 RTD 模块规范

常规	6ES7 231-7PD22-0XA0 热电偶	6ES7 231-7PB22-0XA0 RTD
隔离 现场侧到逻辑 现场侧到 24 VDC 24 VDC 到逻辑	 500 VAC 500 VAC 500 VAC	 500 VAC 500 VAC 500 VAC
共模输入范围 （输入通道到输入通道）	120 VAC	0
共模抑制	>120 dB@120 VAC	>120 dB@120 VAC
输入类型	悬浮型热电偶	模块参考接地的热电阻
输入范围	TC 类型（选择一种） S，T，R，E，N，K，J 电压范围：＋/－80mV	热电阻类型（选择一种） PT-100Ω，200Ω，500，1000Ω （α 为 3850ppm，3920ppm，3850.55ppm，3916ppm，3902ppm） Pt-10000Ω(α=3850ppm) Cu-9.035Ω(α=4720ppm) Ni-10Ω，120Ω，1000Ω (α 6720ppm，6178ppm) R－150Ω，300Ω，600ΩFS
输入分辨率 温度 电压 电阻	 0.1℃/0.1℉ 15 位加符号位	 0.1℃/0.1℉ 15 位加符号位
测量原理	Sigma→delta	Sigma→delta
模块更新时间：所有通道	405mS	405mS(P t10000 为 700ms)
导线长度	到传感器最长为 100m	到传感器最长为 100m
导线回路电阻	最大为 100Ω	20Ω，2.7Ω，（Cumax）
干扰抑制	85 dB 在 50Hz/60Hz/400Hz 时	85 dB 在 50Hz/60Hz/400Hz 时
数据字格式	电压：-27648 到+27648	电阻：-27648 到+27648
传感器最大散热		1mW
输入阻抗	>1MΩ	>10MΩ
最大输入范围	30 VDC	30 VDC（检测），5 VDC（源）
分辨率	15 位加符号位	15 位加符号位
输入滤波器衰减	-3 dB，21kHz	-3 dB，3.6 kHz
基本误差	0.1%FS（电压）	0.1%FS（电阻）
重复性	0.05%FS	0.05%FS
冷端误差	±1.5℃	
24 VDC 提供电压范围	20.4—28.8 VDC	20.4-28.8VDC

[1] 输入范围的选择（温度、基于阻抗的电压）对于一个模块的所有通道使用。

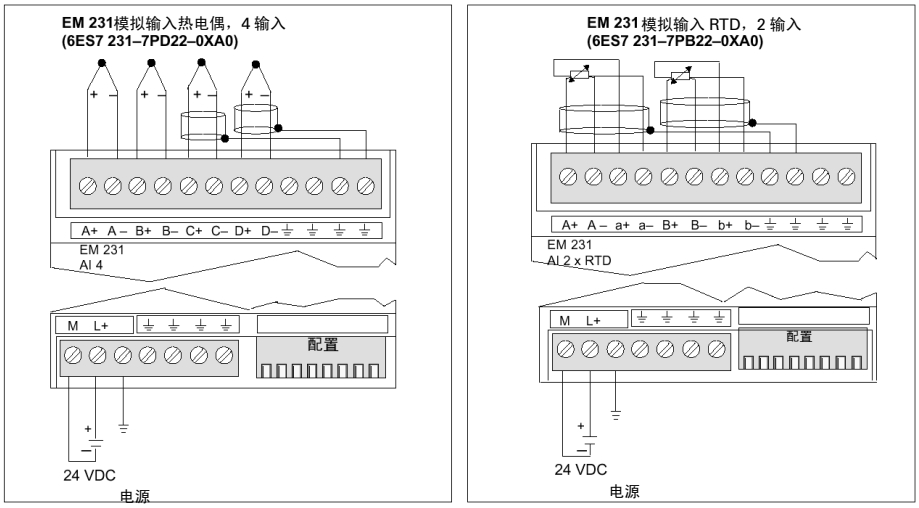


图 A-17　EM 231 热电偶和 EM 231 RTD 模块的端子接线图

兼容性

热电阻和热电偶模块设计成与 S7-200 CPU 222，CPU 224，CPU 226 和 CPU 226XM 一起工作。

提示

RTD 和热电偶模块安装在一个稳定的温度环境内时，具有最佳的性能。

例如，EM 231 热电偶模块有专门的冷端补偿电路。该电路在模块连接器处测量温度，并对测量值作出必要的修正，以补偿基准温度和模块处温度之间的温度差。如果 EM 231 热电偶模块安装环境的温度变化很剧烈，则会引起附加的误差。

为了达到最大的精度和重复性，西门子公司建议，S7-200 RTD 和热电偶模块要安装在环境温度稳定的地方。

噪声抑制

使用屏蔽线是最好的噪声抑制方法。如果热电偶的输入未使用，短接未使用的通道，或将它们并行连接到其它通道上。

EM 231 热电偶模块

EM 231 热电偶模块为 S7-200 系列提供了与 7 种热电偶类型 J、K、E、N、S、T 和 R 相连接的隔离接口。同时可以使 S7-200 能连接低电平模拟信号，±80mV 测量范围。所有连接到该模块的热电偶都必须是同一类型的。

热电偶的基本知识

任何两种金属，其连接处都会形成热电偶。热电偶产生的电压与连接点温度成正比。这个电压很低，1 微伏可能代表若干度。测量从热电偶来的电压，对外部连接点进行补偿，然后将测量值线性化。这些过程是以热电偶测量温度的基础。

当你将一个热电偶连接到 EM 231 热电偶模块时，两根不同的金属导线连接到模块的输入信号接线端子。两根不同金属导线彼此连接处形成传感器的热电偶。

在两根不同金属导线连接到输入信号接线端子的地方形成其它两个热电偶。接线端子处的温度产生一个电压，加到从传感器热电偶来的电压上。如果这个电压不校正，那末，所测量的温度会偏离传感器的温度。

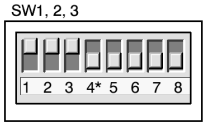
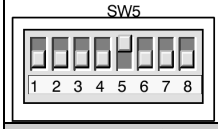
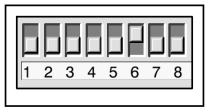
冷端点补偿用来补偿接线端子处的热电偶。热电偶表基于基准连接点温度，通常是摄氏 0 度。模块冷端补偿将接线端子处的温度补偿到摄氏 0 度。冷端补偿补偿了由于接线端子热电偶电压所引起的电压增加。模块温度是在内部测量的。这个温度转换成一个值，它加到传感器的转换值上。然后，用热电偶表线性化被修正后的传感器转换值。

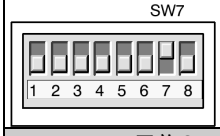
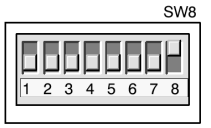
组态 EM 231 热电偶模块

组态 DIP 开关位于模块的底部，可以选择热电偶模块的类型、断线检测、温度范围和冷端补偿。为了使 DIP 的设置生效，您必须对 PLC 或用户 24V 电源重新上电。

DIP 开关 4 为以后的应用保留，将 DIP 开关 4 设定为 0 位置（向下），其他 DIP 开关的设定请参阅表 A-25。

表 A-25　用 DIP 开关配置热电偶模块

开关 1，2，3	热电偶类型	设置	描述
SW1, 2, 3  * Set DIP switch 4 to the 0 (down) position.	J（缺省）	000	开关 1 至 3 为模块上的所有通道选择热电偶类型（或 mV 操作）。例如，选 E 类型，热电偶开关 SW1=0， SW2=1，SW3=1
	K	001	
	T	010	
	E	011	
	R	100	
	S	101	
	N	110	
	+/−80mV	111	
开关 5	**断线检测方向**	**设置**	**描述**
SW5  	正标定 (+3276.7 度)	0	0 指示正向断线检测 1 指示负向断线检测
	负标定 (-3276.8 度)	1	
开关 6	**断线检测使能**	**设置**	**描述**
SW6  	使能	0	通过注入 25μA 电流到输入端进行断线检测。断线检测使能开关可以使能或禁止检测电流。断线检测始终在进行，即使关闭了检测电流。如果输入信号超过 ±200mV，则 EM 231 热电偶模块开始断线检测，如检测到断线，测量读数被设定成由断线检测所选定的值。
	禁止	1	

开关 7	温度范围	设置	描述
SW7  	摄氏度(℃)	0	EM 231 热电偶模块能够报告摄氏温度和华氏温度。摄氏温度与华氏温度的转换在内部进行。
	华氏度(℉)	1	
开关 8	**冷端补偿**	**设置**	**描述**
SW8  	冷端补偿使能	0	使用热电偶必须进行冷端补偿，如果没有使能冷端补偿，模块的转换则会出现错误。因为热电偶导线连接到模块连接器时会产生电压，当选择±80mV 的范围时，冷端补偿会自动禁止。
	冷端补偿禁止	1	

提示

- 断线检测电流源可能干扰某些低电平信号，例如热电偶模拟器。
- 输入电压超过±200mV 将触发断线检测，即使断线检测电流被禁止。

提示

- 当环境温度变化时，模块误差有可能超过技术规范中的数据。
- 超过模块温度范围的规范时，有可能导致模块冷端补偿出错。

使用热电偶：状态指示器

热电偶模块提供 PLC 测量温度或出错类型的数据字。状态位指示输入范围错误和用户电源/模块故障。LED 指示模块状态，用户程序也可检测出相应错误状态并采取相应的措施。热电偶状态指示器见表 A-26。

表 A-26　热电偶状态指示器

出错类型	通道数据	SF 指示灯	24 指示灯	范围状态位 [1]	24V 模块不良状态位 [2]
没有出错	转换数据	断开	接通	0	0
24V 丢失	32766	断开	断开	0	1
使能断线检测和检测电流源	-32768/32767	闪烁	接通	1	0
超出输入范围	-32768/32767	闪烁	接通	1	0
诊断出错 [3]	0000	接通	断开	0	注释 3

[1] 范围状态位是模块出错寄存器字节中的位 3（模块 1 为 SMB 9，模块 2 为 SMB11，等等）

[2] 不良状态位是模块出错寄存器字节中的位 2（SMB 9，SMB11 等，参阅附录 D）

[3] 诊断出错造成模块组态出错。在模块组态出错前，模块不良状态位可能设置或没有设置

提示

通道数据格式是 2 的补码，16 位字，表示温度的单位为 0.1 度例如测量的温度为 100.2 度，则报告数据是 1002，电压数据标定到 27648。例如，- 60.0mV 则报告为 – 20736（= -60mV/80mV * 27648）

如 PLC 已读取到数据，则每 405ms 更新所有 4 个通道的数据，如在一个更新时间内，PLC 没有读数据，则模块报告原有的数据一直到 PLC 读数据后的下一次模块更新，为了保持通道数据总是为当前值，建议 PLC 读数据的频度至少和模块更新频率相同。

提示

如正在使用热电偶模块，应该禁止在 PLC 中使用模拟量滤波。在以定时方式进行检查时，模拟量滤波会防碍出错条件的检测。

表 A-27　各种类型热电偶的温度范围（℃）和准确度。

数据字 (1 个数字位=0.1℃)		类型 J	类型 K	类型 T	类型 E	类型 R，S	类型 N	±80mV	
十进制	十六进制								
32767	7FFF	>1200.0℃	>1372.0℃	>400.0℃	>1000.0℃	>1768.0℃	>1300.0℃	>94.071mV	OF
↑	↑							↑	↑
32511	7EFF							97.071mV	OR
:	:								
27649	6C01							80.0029mV	
27648	6C00					↑		80mV	NR
:	:								
17680	4510		↑			1768.0℃			
:	:								
13720	3598		1372.0℃				↑		
:	:		超出范围						
13000	32C8	↑	1300.0℃				1300.0℃		
:	:								
12000	2EE0	1200.0℃			↑				
:	:								
10000	2710			↑	1000.0℃				
:	:								
4000	0FA0			-400.0℃		400.0℃			
:	:								
1	0001	0.1℃	0.1℃	0.1℃	0.1℃	0.1℃	0.1℃	0.0029mV	
0	0000	0.0℃	0.0℃	0.0℃	0.0℃	0.0℃	0.0℃	0.0mV	
-1	FFFF	-0.1℃	-0.1℃	-0.1℃	-0.1℃	-0.1℃	-0.1℃	-0.0029mV	
:	:					低于范围			
-500	FE0C					-50.0℃			
-1500	FA24	-150.0℃				↓			
:	:								
-2000	F830	低于范围	-200.0℃						
:	:								
-2100	F7CC	-210.0℃							
:	:		低于范围						
-2550	F60A			-255.0℃	-255.0℃				
:	:			低于范围	低于范围				
-2700	F574	↓	-270.0℃	-270.0℃	-270.0℃		-270.0℃		
:	:								
-27648	9400		↓	↓	↓		↓	-80.mV	
-27649	93FF							-80.0029mV	UR
:	:								
-32512	8100							-94.071mV	
#	#							↓	↓
-32768	8000	<-210.0℃	<-270.0℃	<-270.0℃	<-270.0℃	<-50.0℃	<-270.0℃	<-94.07mV	UF
全量程范围的精度		S0.1%	S0.3%	S0.6%	S0.1%	S0.6%	S0.1%	S0.1%	
精度(无冷端补偿的额定范围)		S1.5℃	S1.7℃	S1.4℃	S1.3℃	S3.7℃	S1.6℃	S0.10℃	
冷端误差		S1.5℃	S1.5℃	S1.5℃	S1.5℃	S1.5℃	S1.5℃	N/A	

* OF=下溢；OR=超出范围；NR=额定范围；VR=低于范围；UF=下溢

↑表示所有大于该值但小于断线阀值的模拟量都报告为上溢出值，32767（10x7FFF）。

↓表示所有小于该值但大于断线阀值的模拟量都报告为下溢出值，-32768（0x8000）。

表 A-28　温度范围（℉）用于热电偶类型。

数据字 (1 个数字位=0.1℉)		类型 J	类型 K	类型 T	类型 E	类型 R，S	类型 N	±80mV	
十进制	十六进制								
32767		>2192.0℉	>2502.0℉	>752.0℉	>1832.0℉	>3214.0℉	>2372.0℉	>94.071mV	
↑	↑					↑		↑	↑
32511	7EFF							94.071mV	
27649	6C01					3214.0℉			OR
27648	6C00		↑					80.0029mV	
:	:					2764.8℉		80mV	
25020	61B8		2502.0				↑		
:	:		超出范围						
23720	5CA8	↑	23720.0				2372.0℉		
:	:								
21920	55A0	2192.0℉			↑				
:	:			↑					NR
18320	4790				1832.0℉				
:	:			7520℉					
7520	1D60					752.0℉			
:	:								
320	0140					低于范围	32.0℉		
:	:								
1	0001	0.1℉	0.1℉	0.1℉	0.1℉	0.1℉	0.1℉	0.0029mV	
0	0000	0.0℉	0.0℉	0.0℉	0.0℉	0.0℉	0.0℉	0.0mV	
-1	FFFF	-0.1℉	-0.1℉	-0.1℉	-0.1℉	-0.1℉	-0.1℉	-0.0029mV	
:	:								
-580	FDBC					-58.0℃			
:	:								
-2380	F6B4	-238.0℃							
:	:								
-3280	F330	低于范围	-328.0℃						
:	:								
-3460	F27C	-346.0℃							
:	:		低于范围						
-4270	EF52			-427.0℉	-427.0℉				
:	:			低于范围	低于范围	↓	低于范围		
-4540	EE44	↓	-454.0℉	-454.0℉	-454.0℉		-454.0℉		
:	:								
-27648	9400		↓	↓	↓		↓	-80.mV	
-27649	93FF							-80.0029mV	
:	:								
-32512	8100							-94.071mV	OR
↓	↓							↓	↓
-3268	8000	<-346.0℉	<-454.0℉	<-454.0℉	<-454.0℉	<-58.0℉	<-454.0℉	<-94.07mV	UF

* OF=下溢；OR=超出范围；NR=额定范围；VR=低于范围；UF=下溢

↑表示所有大于该值但小于断线阀值的模拟量都报告为上溢出值，32767（10x7FFF）。

↓表示所有小于该值但大于断线阀值的模拟量都报告为下溢出值，-32768（0x8000）。

EM 231 热电阻模块

EM 231 热电阻模块为 S7-200 连接各种型号的热电阻提供了方便的接口。它也允许 S7-200 测量三个不同的电阻范围。连接到模块的热电阻必须是相同的类型。

配置 EM 231 RTD（热电阻）模块

使用 DIP 开关可以选择热电阻的类型，接线方式，温度测量单位和传感器熔断方向。DIP 配置开关位于模块的底部，如图 A-18 所示，为了使 DIP 开关的设置生效，应该对 PLC 或 24V 电源重新上电。

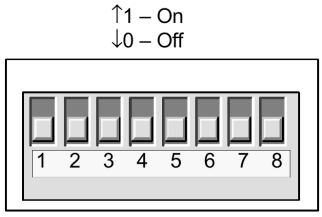


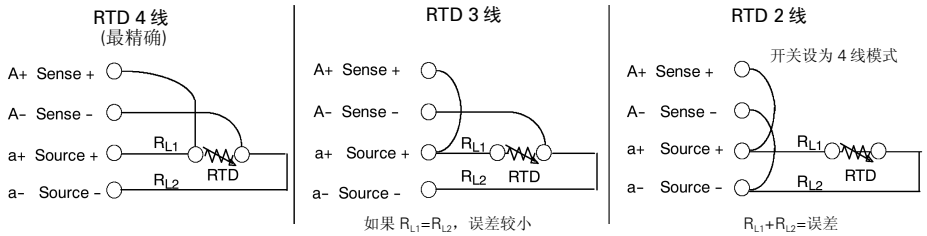
图 A-18　EM 231 RTD 模块 DIP 开关

表 A-29　选择 RTD 类型：DIP 开关 1-5。

RTD 类型	SW1	SW2	SW3	SW4	SW5	RTD 类型	SW1	SW2	SW3	SW4	SW5
100ΩPt 0.003850 (缺省)	0	0	0	0	0	100ΩPt 0.00302	1	0	0	0	0
200ΩPt 0.003850	0	0	0	0	1	200ΩPt 0.003902	1	0	0	0	1
500ΩPt 0.003850	0	0	0	1	0	500ΩPt 0.003902	1	0	0	1	0
1000ΩPt 0.003850	0	0	0	1	1	1000ΩPt 0.003902	1	0	0	1	1
100ΩPt 0.003920	0	0	1	0	0	SPARE	1	0	1	0	0
200ΩPt 0.003920	0	0	1	0	1	100ΩNi 0.00672	1	0	1	0	1
500ΩPt 0.003920	0	0	1	1	0	120ΩNi 0.00672	1	0	1	1	0
1000ΩPt 0.003920	0	0	1	1	1	1000ΩNi 0.00672	1	0	1	1	1
100ΩPt 0.00385055	0	1	0	0	0	100ΩNi 0.006178	1	1	0	0	0
200ΩPt 0.00385055	0	1	0	0	1	120ΩNi 0.006178	1	1	0	0	1
500ΩPt 0.00385055	0	1	0	1	0	1000ΩNi 0.006178	1	1	0	1	0
1000ΩPt 0.00385055	0	1	0	1	1	10000ΩPt 0.003850	1	1	0	1	1
100ΩPt 0.003916	0	1	1	0	0	10ΩCu 0.004270	1	1	1	0	0
200ΩPt 0.003916	0	1	1	0	1	150ΩFs 电阻	1	1	1	0	1
500ΩPt 0.003916	0	1	1	1	0	300ΩFs 电阻	1	1	1	1	0
1000ΩPt 0.003916	0	1	1	1	1	600ΩPHYS 电阻	1	1	1	1	1

表 A-30　设置 RTD DIP 开关

开关 6	断线检测	设置	描述
SW6 1 2 3 4 5 6 7 8 配置 ↑1 – On ↓0 – Off	正向标定 (+3276.7 度)	0	指示正向断线
	负向标定 (-3276.8 度)	1	指示负向断线
开关 7	**温度范围**	**设置**	**描述**
SW7 1 2 3 4 5 6 7 8 配置 ↑1 – On ↓0 – Off	摄氏度(℃)	0	RTD 模块可报告摄氏温度或华氏温度，摄氏到华氏的转换在模块内部进行。
	华氏度(℉)	1	
开关 8	**接线方式**	**设置**	**描述**
SW8 1 2 3 4 5 6 7 8 配置 ↑1 – On ↓0 – Off	3 线	0	RTD 模块与传感器的接线有 3 种方式(如图所示)。精度最高的是 4 线连接。2 线连接精度最低，推荐只用于可忽略接线误差的应用场合。
	2 线或 4 线	1	



注意：R_{L1}=从 a+到 RTD 接线端的导线电阻
R_{L2}=从 a-到 RTD 接线端的导线电阻

图 A-19　RTD 到传感器的接线，4 线、3 线和 2 线

EM 231 RTD 状态指示器

RTD 模块提供 PLC 温度或出错类型的数据字、状态位指示输入范围错误以及电源/模块故障。LED 指示模块的状态。用户程序也可检测出相应错误状态并采取相应的措施。表 A-31 说明由 EM 231 RTD 模块所提供的状态指示器。

提示

通道的数据格式是 2 的补码，16 位字，表达温度所采用的单位是 0.1 度（例如测量的温度为 100.2 度，则报告数据为 1002），电阻数据标度到 27648。例如，全量程电阻范围的 75%报告为 20736。（=255Ω/330Ω＊27648）

表 A-31 EM 231 热电阻状态指示器

出错类型	通道数据	SF 指示灯 红色	24V 指示灯 绿色	范围 状态位 [1]	24VDC 用户电源故障 [2]
无错误	转换数据	断	通	0	0
24V 消失	32766	断	断	0	1
SW 断线检测	-32768/32767	闪烁	通	1	0
超出输入范围	-32768/32767	闪烁	通	1	0
诊断出错 [3]	0000	通	断	0	注释 3

[1] 范围状态位是模块出错寄存器字节中的位 3（SMB9 于用模块 1，SMB11 用于模块 2，等等）

[2] 用户电源故障状态位是出错寄存器字节中的位 2（SMB9，SMB11 等。参阅附录 D）

[3] 诊断出错引起模块组态错误。在模块组态错误之前，用户电源故障状态位可能设置或没有设置。

如 PLC 已读到数据，则需 405ms 更新所有通道的数据，如果在一个更新时间内，PLC 没有读数据，则模块报告原有的数据一直到 PLC 读数据后的下一次模块更新，为了保持通道数据总为当前值，建议 PLC 读数据的频度至少和模块更新频率相同。

提示

如正在使用热电阻模块，应禁止在 PLC 中使用模拟量滤波，在以定时方式进行检查时，模拟量滤波会防碍出错条件的检测。

断线检测是由软件在 RTD 模块内部完成的。超过输入范围或断线时，测量值都被设定为熔断的标定值。断线检测至少需要三个模块扫描周期或更长时间，这通常取决于具体的断线类型。Source+和 Source-的断线检测通常需要的时间最小，而 Sense+或 Sense-则需要 5 秒钟或更长的时间来检测。在电气噪声严重的环境中，间歇地检测到断线时，开路的 Sense 线上（测量线）也会有随机的有效数据出现。电气噪声会延长断线检测的时间。为此建议，在程序收到有效数据后，还应在应用程序中对断线检测/超输入范围的状态指示进行监控及锁定。

表 A-32 用于热电阻类型的温度范围（℃）和精度

系统字 (1 个数字=0.1℃)		Pt10000	Pt100 Pt200 Pt500 Pt1000	Ni100, Ni120, Ni1000	Cu9.035	0-150 Ω	0-300 Ω	0-160 Ω	
10 进制	16 进制								
32767	7FFF								
32766	7FFE					↑	↑	↑	
32511	7EFF					176.383 Ω	352.767 Ω	705.534 Ω	
29649	6C01					150.005 Ω	300.011 Ω	600.022 Ω	
27648	6C00					150.000 Ω	300.000 Ω	600.000 Ω	
25000	61AB								↑
18000	4650								OR
15000	3A98								
13000	32C8	↑	↑						
10000	2710	1000.0℃	1000.0℃						
8500	2134		850.0℃						
6000	1770	600.0℃			↑				
3120	0C30			↑	312.0℃				
2950	0B86			295.0℃					
2600	0A28				260.0℃				
2500	09C4			250.0℃					
1	0001	0.1℃	0.1℃	0.1℃	0.1℃	0.005 Ω	0.011 Ω	0.022 Ω	
0	0000	0.0℃	0.0℃	0.0℃	0.0℃	0.000 Ω	0.000 Ω	0.000 Ω	
-1	FFFF	-0.1℃	-0.1℃	-0.1℃	-0.1℃	（负值是不可能的）			
						↓	↓	↓	
-600	FDA8			-60.0℃					
-1050	FBE6			-105.0℃					
				↓					
-2000	F830	-200.0℃	-200.0℃		-200.0℃				
-2400	F6A0				-240.0℃				
-2430	F682	-243.0℃	-243.0℃		↓				
		↓	↓						
-5000	EC78								
-6000	E890								UR
-10500	D6FC								↓
-12000	D120								
-20000	4E20								
-32767	8001								
-32768	8000								
全量程范围的精度		±0.4%	±0.1%	±0.2%	±0.5%	±0.1%	±0.1%	±0.1%	
精度（额定范围）		±4℃	±1℃	±0.6℃	±2.8℃	±0.15℃	±0.3℃	±0.6℃	

*OF=上溢出；OR=超出范围；NR=额定范围；UR=低于范围；UF=下溢出

↑或↓表示所有超过或低于这个限制值的模拟量其测量值都报告为所选的熔断标定值，32767（0X7FFF）或-32768（0X8000）

表 A-33 RTD 类型的温度范围（° F）

系统字（1 个数字=0.1° F）		PT1000	PT100, Pt200, Pt500, Pt1000	Ni100, Ni120, Ni1000	Cu 9.035	
10 进制	16 进制					
32767	7FF.					
32766	7PHAGE					
						↑
						超出范围
		↑	↑			
18320	4790	1832.0°F	1832.0 °F			
15620	3D04		1562.0°F			
11120	2B70	1112.0°F				
					↑	
5936	1730			↑	593.6°F	
5630	15FE			563.0°F		
5000	1388				500.0°F	
4820	12D4			482.0°F		
						额定范围
1	0001	0.1°F	0.1°F	0.1°F	0.1°F	
0	0000	0.0°F	0.0°F	0.0°F	0.0°F	
–1	FFFF	–0.1°F	–0.1°F	–0.1°F	–0.1°F	
–760	FD08			–76.0°F		
–1570	F9DE			–157.0°F		
				↓		
–3280	F330	–328.0°F	–328.0°F		–328.0°F	
–4000	F060				–400.0°F	
–4054	F02A	–405.4°F	–405.4°F		↓	
		↓	↓			
–5000	EC78					
–6000	E890					低于范围
–10500	D6FC					↓
–32767	8001					
–32768	8000					

↑或↓表示所有超过或低于这个限制值的模拟量其测量值都报告为所选的熔断标定值，即 32767（0X7FFF）或-32768（0X8000）

EM 277 PROFIBUS-DP 模板规范

表 A-34 EM 277 PROFIBUS-DP 模块定货号

定货号	扩展模块	输入	输出	可拆卸连接器
6ES7 277-0AA22-0XA0	EM 277 PROFIBUS-DP	-	-	否

表 A-35 EM 277 PROFIBUS-DP 模板适用规范

定货号	模板名称和描述	尺寸(mm) (W×H×D)	重量	功耗	VDC 要求	
					+5VDC	+24VDC
6ES7 277-0AA22-0XA0	EM 277 PROFIBUS-DP	71×80×62	175g	2.5W	150mA	见下文

表 A-36 EM 277 PROFIBUS-DP 模块规范

常规	6ES7 277-0AA22-0XA0
接口数	1
电气接口	RS-485
PROFIBUS-DP/MPI 波特率（自动设置）	9.6,19.2,45.45,93.75,187.5,和 500K 波特；1,1.5,3,6,和 12M 波特
协议	PROFIBUS-DP 从站和 MPI 从站
电缆长度	
最高 93.75 波特	1200m
187.5 波特	1000m
500 波特	400m
1 到 1.5 波特	200m
3 到 12 波特	100m
连网能力	
站地址设置	0-99(由旋钮开关设定)
每段最大站数	32
每个网络最大站数	126，最多 99 个 EM 277 站
MPI 连接	一共 6 个，2 个保留（1 个给 PG，1 个给 OP）
输入电源要求	
电压范围	20.4-28.8VDC（等级 2 或来自 PLC 的传感器电源）
最大电流 仅当模块端口激活时 加 90mA,5V 端口负载 加 120mA,24V 端口负载	 30mA 60mA 180mA
纹波噪声（<10 MHz）	<1V 峰峰值（最大）
隔离（现场到逻辑）[1]	500VAC，1 分钟
通讯口的 5VDC 电源	
每个口的最大电流	90mA
隔离（24VDC 到逻辑）	500VAC，1 分钟
通讯口的 24VDC 电源	
电压范围	20.4-28.8VDC
每个端口最大电流	120mA
电流限定	0.7-2.4A
隔离	无隔离，与输入 24VDC 电路相同

[1] 24VDC 电源不对模块逻辑供电，24V 电源为通讯端口供电。

支持智能模块的 S7200CPU

EM 277 PROFIBUS-DP 从站模块，是和 S7-200 PLC 一起工作的智能扩展模块，见表 A-37。

表 A-37 EM 277 PROFIBUS-DP 模块和 S7-200 PLC 的兼容性

CPU	说明
CPU 222 版本 1.10 或更高	CPU 222 DC/DC/DC
	CPU 222 AC/DC/Relay
CPU 224 版本 1.10 或更高	CPU 224 DC/DC/DC
	CPU 224 AC/DC/Relay
CPU 226 版本 1.00 或更高	CPU 226 DC/DC/DC
	CPU 226 AC/DC/Relay
CPU 226XM 版本 1.00 或更高	CPU 226XM DC/DC/DC
	CPU 226XM AC/DC/Relay

地址开关和 LED

地址开关和状态 LED 位于模块的正面，见图 A-20。DP 从站的接口插针输出也如下图所示。

EM 277 PROFIBUS-DP 前视图

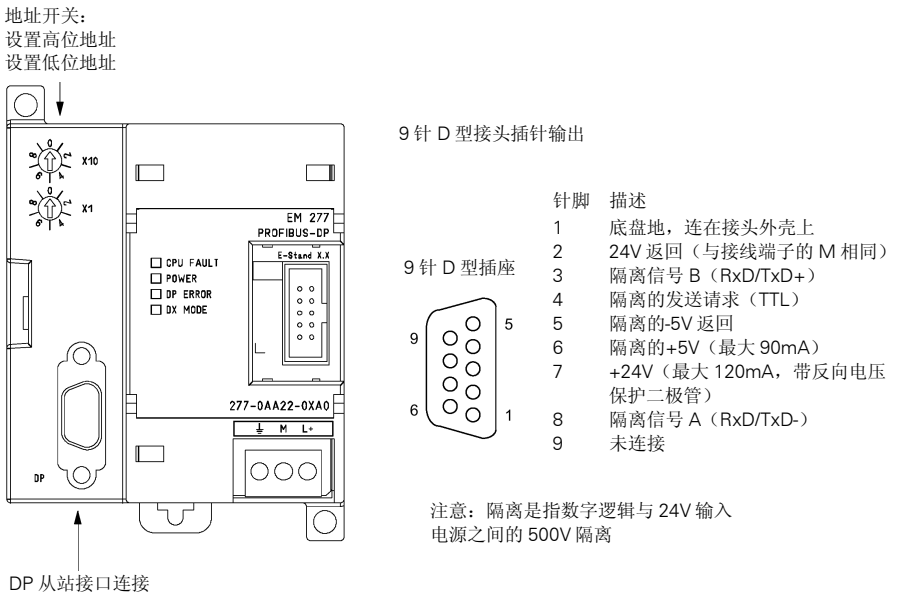


图 A-20 EM 277 PROFIBUS-DP

分布式（DP）外围设备的标准通信

PROFIBUS-DP（或 DP 标准）是由欧洲标准 EN 50170 定义的一种远程 I/O 通信协议。遵守这种标准的设备，即使是由不同公司制造的，也是兼容的。DP 表示分布式外围设备，亦即远程 I/O。PROFIBUS 表示过程现场总线。

EM 277 PROFIBUS-DP 模块已定义作为以下通信协议标准中的从站，实现 DP 标准协议：

- EN 50170（PROFIBUS）描述总线访问和传送协议，并规定数据传送介质的性能。
- EN 50170（DP 标准）描述 DP 主站和 DP 从站之间的高速循环交换数据。这个标准规定组态和参数赋值过程，解释具有分布式 I/O 功能的循环数据如何进行交换，并列出支持的诊断选择。

一个 DP 主站组态应包含地址，从站类型以及从站所需要的任何参数赋值信息。还应告诉主站由从站（输入）读入的数据应放置何处，以及从何处获得写入从站（输出）的数据。DP 主站建立网络，然后初始化其 DP 从站。主站将参数赋值信息和 I/O 配置写入到从站。然后，主站从从站那里读出诊断信息，并验证 DP 从站已接受参数和 I/O 配置。然后，主站开始与从站交换 I/O 数据。每次对从站的数据交换为写输出和读输入。这种数据交换方式无限期地继续下去。如果有一个例外事件，从站会通知主站，然后，主站从从站那里读出诊断信息。

一旦 DP 主站已将参数和 I/O 配置写入到 DP 从站，而且从站已从主站那里接收到参数和配置，则主站就拥有那个从站。从站只能接收自其主站的写请求。网络上的其它主站可以读取该从站的输入和输出，但是它们不能向该从站写入任何信息。

使用 EM 277 将 S7-200 CPU 作为 DP 从站连接到网络

通过 EM 277 PROFIBUS-DP 扩展从站模块，可将 S7-200 CPU 连接到 PROFIBUS-DP 网络。EM 277 经过串行 I/O 总线连接到 S7-200 CPU。PROFIBUS 网络经过其 DP 通信端口，连接到 EM 277 PROFIBUS-DP 模块。这个端口可运行于 9600 波特和 12M 波特之间的任何 PROFIBUS 波特率。支持的波特率请参见 EM 277 PROFIBUS 模块规范。

作为 DP 从站，EM 277 模块接受从主站来的多种不同的 I/O 配置，向主站发送和接收不同数量的数据。这种特性使用户能修改所传输的数据量，以满足实际应用的需要。与许多 DP 站不同的是，EM 277 模块不仅仅是传输 I/O 数据。EM 277 能读写 S7-200 CPU 中定义的变量数据块。这样，使用户能与主站交换任何类型的数据。首先将数据移到 S7-200 CPU 中的变量存储器，就可将输入、计数值、定时器值或其它计算值传送到主站。类似地，从主站来的数据存储在 S7-200 CPU 中的变量存储器内，并可移到其它数据区。

EM 277 PROFIBUS-DP 模块的 DP 端口可连接到网络上的一个 DP 主站上，但仍能作为一个 MPI 从站与同一网络上如 SIMATIC 编程器或 S7-300/S7-400 CPU 等其它主站进行通信。图 A-21 所示为一个 CPU 224 和一个 EM 277 PROFIBUS-DP 模块的 PROFIBUS 网络。

- CPU-315-2 是 DP 主站，并且已通过一个带有 STEP 7 编程软件的 SIMATIC 编程器进行组态。

- CPU 224 是 CP 315-2 所拥有的一个 DP 从站，ET 200 I/O 模块也是 CPU 315-2 的从站。
- S7-400 CPU 连接到 PROFIBUS 网络，并且藉助于 S7-400 CPU 用户程序中的 XGET 指令，可从 CPU 224 读取数据。

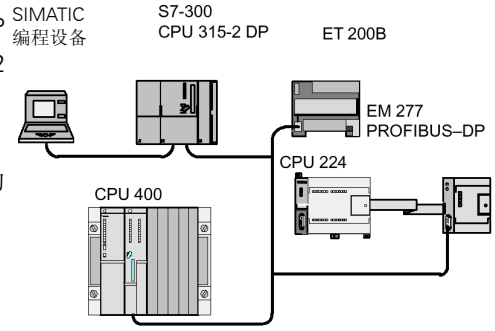


图 A-21 PROFIBUS 网络中 EM 277 PROFIBUS-DP 模块和 CPU 224

组态

为了将 EM 277 作为一个 DP 从站使用，用户必须设定与主站组态中的地址相匹配的 DP 端口地址。从站地址是使用 EM 277 模块上的旋转开关设定的。在变动旋转开关之后，用户必须重新启动 CPU 电源，以便使新的从站地址起作用。

主站通过将其输出区来的信息发送给从站的输出缓冲区（称为“接收信箱”），与其每个从站交换数据。从站将其输入缓冲区（称为发送信箱）的数据返回给主站的输入区，以响应从主站来的信息。

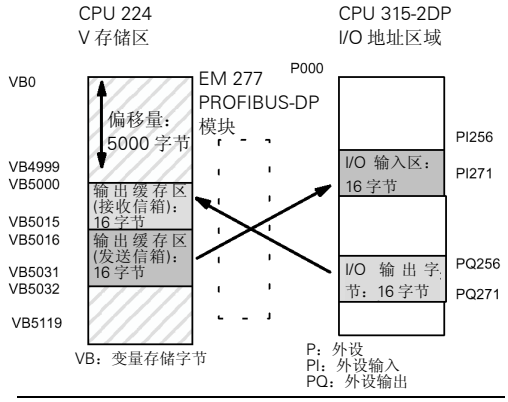


图 A-22 V 存储器和 I/O 地址区域

图 A-22 为 PROFIBUS-DP 主站的 V 存储器和 I/O 地址区域模型。

EM 277 可用 DP 主站组态，以接收从主站来的输出数据，并将输入数据返回给主站。输出和输入数据缓冲区驻留在 S7-200 CPU 的变量存储器（V 存储器）内。当用户组态 DP 主站时，应定义 V 存储器内的字节位置。从这个位置开始为输出数据缓冲区，它应作为 EM 277 的参数赋值信息的一个部分。用户也要定义 I/O 配置，它是写入到 S7-200 CPU 的输出数据总量和从 S7-200 CPU 返回的输入数据总量。EM 277 从 I/O 配置确定输入和输出缓冲区的大小。DP 主站将参数赋值和 I/O 配置信息写入到 EM 277 PROFIBUS-DP 模块，然后，EM 277 将 V 存储器地址和输入及输出数据长度传送给 S7-200 CPU。

图 A-22 表示 CPU 224 中的 V 存储器的一个存储器模型，以及一个 DP 主站 CPU 的 I/O 地址区。在这个例子中，DP 主站已定义了 16 输出字节和 16 输入字节的一种 I/O 配置，以及 V 存储器偏移为 5000。CPU 224 中的输出缓冲区和输入缓冲区长度（由 I/O 配置确定）都是 16 字节。输出数据缓冲区从 V5000 开始；输入缓冲区紧紧跟随输出缓冲区，并在 V5016 处开始。输出数据（从主站来）放置在 V 存储器中的 V5000。输入数据（传送到主站）取自 V 存储器的 V5016。

提示

如果处理的数据单位（一致性数据）是 3 字节，或数据单位（一致性数据）大于 4 字节，则必须使用 SFC 14，以便读出 DP 从站的输入，并使用 SFC-15，以便对 DP 从站的输出进行编址。详细情况见 S7-300 和 S7-400 系统的系统软件和标准功能参考手册。

表 A-38 列出由 EM 277 PROFIBUS-DP 模块支持的组态。EM 277 模块的缺省组态是输入 2 个字和输出 2 个字。

表 A-38 EM277 组态的选择

组态	输入到主站	从主站输出	数据的一致性
1	1 字	1 字	字一致性
2	2 字	2 字	
3	4 字	4 字	
4	8 字	8 字	
5	16 字	16 字	
6	32 字	32 字	
7	8 字	2 字	
8	16 字	4 字	
9	32 字	8 字	
10	2 字	8 字	
11	4 字	16 字	
12	8 字	32 字	
13	2 字	2 字	字节一致性
14	8 字	8 字	
15	32 字	32 字	
16	64 字	64 字	
17	4 字	4 字	缓存区一致性
18	8 字	8 字	
19	12 字	12 字	
20	16 字	16 字	

输入和输出缓存区的地址可以配置在 S7-200 CPU V 存储器中的任何位置。输入和输出缓冲器的缺省值地址为 VB0。输入和输出缓冲地址是主站写入 S7-200 CPU 赋值参数信息的一部分，用户必须组态主站以识别所有的从站以及将需要的参数和 I/O 配置写入每一个从站。

使用以下工具以组态 DP 主站：

- 对于 SIMATIC S5 主站，使用 COM PROFIBUS Windows 软件
- 对于 SIMATIC S7 主站，使用 STEP 7 编程软件
- 对于 SIMATIC 505 主站，使 COM PROFIBUS 和 TISOFT2 或 Softshop 两种软件之一

关于使用这些组态和编程软件的详细信息，请参阅这些软件工具的使用手册，关于 PROFIBUS 网络和其部件的详细信息，请参阅 ET 200 分布式 I/O 系统使用手册。

数据的一致性

PROFIBUS 支持三种类型的数据一致性：

- 字节一致性保证字节作为整个单元传送。
- 字一致性保证字的传送不会被 CPU 中的其它处理所中断。就是说，组成字的二个字节总是一起移动，不会被拆散。
- 缓冲区一致性保证数据的整个缓冲区是作为一个单独单元传送的，不会被 CPU 的任何其它处理过程中断。如果数据值是双字或浮点值，则必须采用缓冲区一致性。当一组值都与一种计算或项目有关时，也需要采用缓冲区一致性。

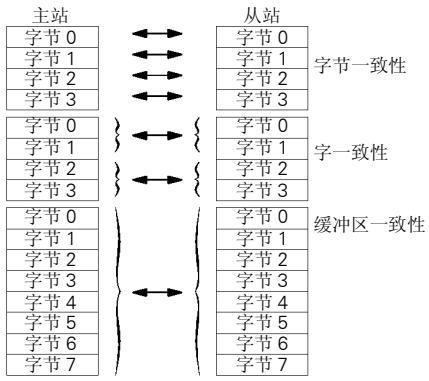


图 A-23 字节，字和缓冲区数据的一致性。

用户将数据一致性设置成主站中的 I/O 组态部分。数据一致性选择作为从站的初始化部分写入 DP 从站。DP 主站和 DP 从站都利用数据一致性选择，以便保证数据值（字节、字或缓冲区）在主站和从站内的传送是不中断的。图 A-23 表示不同类型的一致性。

用户程序需考虑的事项

一旦 EM 277 PROFIBUS-DP 模块已用一个 DP 主站成功地进行了组态，EM 277 和 DP 主站就进入数据交换模式。在数据交换模式中，主站将输出数据写入到 EM 277 PROFIBUS-DP 模块，然后，EM 277 模块响应最新的 S7-200 CPU 输入数据。EM 277 模块不断地更新其从 S7-200 CPU 来的输入，以便向 DP 主站提供最新的输入数据。然后，该模块将输出数据传送给 S7-200 CPU。从主站来的输出数据放在 V 存储器中（输出缓冲区）由某地址开始的区域内，而该地址是在初始化期间，由 DP 主站所提供的。传送到主站的输入数据取自 V 存储器存储单元（输入缓冲区），其地址是紧随输出缓冲区的。

从主站来的输出数据必须通过 S7-200 CPU 中的用户程序，从输出缓冲区转移到其它所用的数据区。类似地，传送到主站的输入数据也必须通过用户程序从各种数据区转移到传送到输入缓冲区，进而发送到 DP 主站。

从 DP 主站来的输出数据，在执行程序扫描后立即放置在 V 存储区。输入数据（传送到主站）从 V 存储器复制到 EM 277 中，以便同时传送到主站。

当主站提供有新的数据时，则从主站来的输出数据才写入到 V 存储器内。

在下次与主站交换数据时，将送到主站的输入数据发送到主站。

在建立 S7-200 CPU 用户程序时，必须知道 V 存储器中的数据缓冲区的开始地址和缓冲区大小。

状态信息

基于其物理位置，每个智能模块都分配有 50 个字节的专用存储区（SM）。模块按照它与 CPU 的相对位置，刷新 SM 区域。如果它是第一个智能模块，则刷新 SMB 200 到 SMB 249。如果是第二个智能模块，那么，它刷新 SMB 250 到 SMB 299，等等。请参见表 A-39。

注意

为智能模块定义 SM 区域的方式在版本 2.2 和其后的版本作了改变。

如果您使用版本 2.2 之前的 CPU，您要将所有的智能模块放置在所有非智能模块之前紧邻 CPU 的位置，以确保其兼容性。

表 A-39 特殊存储字节 SMB200-SMB549

特殊存储字节 SMB200-SMB549						
智能模块位于槽 0	智能模块位于槽 1	智能模块位于槽 2	智能模块位于槽 3	智能模块位于槽 4	智能模块位于槽 5	智能模块位于槽 6
SMB200-SMB249	SMB250-SMB299	SMB300-SMB349	SMB350-SMB399	SMB400-SMB449	SMB450-SMB499	SMB500-SMB549

如果 DP 尚未建立与主站的通信，那么，这些 SM 存储单元显示缺省值。当主站已将参数和 I/O 组态写入到 EM 277 PROFIBUS-DP 模块后，这些 SM 存储单元显示 DP 主站的组态设置。用户应检查协议状态字节（例如，0 号槽的 SMB 224），并确保在使用 SM 的信息以及 V 寄存器缓冲区中的信息之前，EM 277 已处于与主站交换数据的工作模式见表 A-40。

提示

用户不能通过写入 SM 存储单元来组态 EM 277 PROFIBUS-DP I/O 缓冲区的大小，或缓冲区的位置。只有 DP 主站才可以组态运行于 DP 方式下的 EM 277 PROFIBUS-DP 模块。

表 A-40 EM 277 PROFIBUS-DP 特殊存储字节

智能模块在 0 号槽	智能模块在 6 号槽	说明
SMB 200 至 SMB 215	SMB 500 至 SMB 515	模块名（16ASCII 字符） “EM 277 Profibus DP”
SMB 216 至 SMB 219	SMB 516 至 SMB 519	S/W 版本号（4ASCII 字符） XXXX
SMW 220	SMW 520	错误代码 16#0000 无错误 16#0001 无用户电源 16#0002 至 16#FFFF 保留
SMB 222	SMB 522	DP 从模块的站地址，由地址开关(0-99 十进制)设定
SMB 223	SMB 523	保留

智能模块在 0 号槽	智能模块在 6 号槽	说明
SMB 224	SMB 524	DP 标准协议状态字节 MSB LSB 0 0 0 0 0 0 S1 S0 S1 S2 DP 标准状态字节描述 0 0 上电后，DP 通讯未初始化 0 1 组态/参数化错误 1 0 处于数据交换状态 1 1 退出数据交换状态
SMB 225	SMB 525	DP 标准协议 - 从站的主站地址（0 至 126）
SMW 226	SMW 526	DP 标准协议 - 输出缓冲区的 V 存储器地址，作为从 VB0 开始的输出缓冲区的偏移量
SMB 228	SMB 528	DP 标准协议 - 输出数据的字节数
SMB 229	SMB 529	DP 标准协议 - 输入数据的字节数
SMB 230 至 SMB 249	SMB 530 至 SMB 549	保留-电源接通时清除

注：每当 DP 从站模块接收组态/参数化信息时更新 SM 区域。即使探测出一个组态/参数化错误时，这些存储单元也要更新。每次电源接通时，这些存储单元被清除。

EM 277 PROFIBUS-DP 模块 LED 指示灯

EM 277 模块在前面板上存 4 个 LED 指示灯以指示 DP 口的运行状态。

- S7-200 上电后，DX MODE 灯一直熄灭直到 DP 通讯开始。
- 当 DP 的通讯成功地初始化后（EM 277 PROFIBUS-DP 模块进入和主站交换数据的状态时），DX MODE 灯变绿直到数据交换状态结束。
- 如果 DP 通讯中断，强迫 EM 277 模块退出数据交换模式，此时，DX MODE 灯熄灭而 DP ERROR 灯变红。此状态一直保持到 S7-200 CPU 断电或数据交换重新开始。
- 如果主站写入 EM 277 模块的 I/O 配置或参数信息错误，则 DP ERROR 的红灯将闪烁。
- 如果没有 24VDC 供电，POWER（电源）灯将熄灭。

表 A-41 总结了 EM 277 状态指示灯的各种状态。

LED 指示灯	熄灭	红灯	红灯闪烁	绿灯
CPU Fault	模块良好	内部模块故障	-	-
POWER	没有 24VDC 用户电源	-	-	24VDC 用户电源良好
DP ERROR	没有错误	脱离数据交换模式	参数化/组态错误	-
DX MODE	不在数据交换模式	-	-	在数据交换模式

注意：当 EM 277 PROFIBUS-DP 模块专用作 MPI 从站时，仅绿色电源 LED 接通。

附加的组态特性

EM 277 PROFIBUS-DP 模块可作为连接到其它 MPI 主站的通信接口，而不论该模块是否用作 PROFIBUS-DP 从站。该模块利用 S7-300/400 的 XGET/XPUT 功能，提供从 S7-300/400 到 S7-200 的连接。应用 MPI 或 PROFIBUS 参数设置的 STEP 7-Micro/WIN 软件和一个网络卡（例如 CP 5611），一个 OP 设备或 TD 200（版本 2.0 或以上，订货号 6ES7 272 0AA20-0YA0），可经过 EM 277 PROFIBUS-DP 模块，与 S7-200 进行通信。

除 DP 主站外，最多可以有 6 个连接（6 个设备）与 EM 277 PROFIBUS-DP 模块相连接。一个连接是为编程器（PG）而保留的，一个连接是为操作员面板（OP）而保留的。其它 4 个连接可被任何一个 MPI 主站使用。为了使 EM 277 PROFIBUS-DP 模块与多个主站进行通信，所有主站都必须在相同的波特率下运行。见图 A-24 是一个可能的网络配置。

当 EM 277 PROFIBUS-DP 模块用于 MPI 通信时，MPI 主站必须使用 EM 277 模块的站址向 S7-200 CPU 发送信息。发送给 EM 277 PROFIBUS-DP 模块的 MPI 信息将通过 EM 277 传送给 S7-200 CPU。

EM 277 PROFIBUS-DP 模块是一种从站模块，不能用来通过 NETR 和 NETW 语句进行不同的 S7-200 PLC 之间的通讯。EM 277 PROFIBUS-DP 模块不能用于自由端口的通信，尽管 S7-200 本机上的通信端口都具有这种通讯功能。

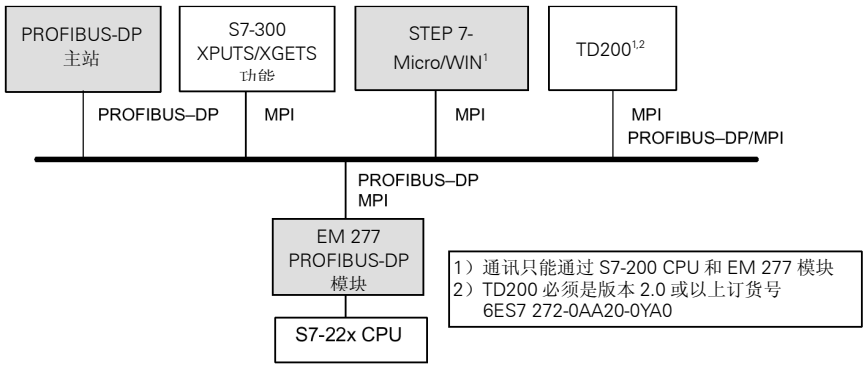


图 A-24 PROFIBUS-DP/MPI 网络

设备数据库文件：GSD

不同的 PROFIBUS 设备有不同的性能特性。这些特性就功能（例如，I/O 信号的数量和诊断信息）或总线参数（例如，传输速度和时间监视）而言是不同的。这些参数对每个设备类型和供应商来说都是不同的，而且通常汇编在技术手册内。为了帮助用户完成 PROFIBUS 的简单组态，通常把包含特定设备性能参数的电子表格称为设备数据库文件，即 GSD 文件。基于 GSD 文件的组态工具能简单地将由不同供应商来的设备集成在一个单一的网络内。

设备数据库文件以精确的格式提供对设备特性的全面描述。这些 GSD 文件是供应商为每种类型设备而准备并提供给 PROFIBUS 用户的。GSD 文件能使组态系统读入 PROFIBUS 设备的特性，并在组态系统时利用这个信息。

COM PROFIBUS 或 STEP 7 软件的最新版本包括 EM 277 PROFIBUS-DP 模块的组态文件。如果你的软件版本不包括用于 EM 277 的组态文件，你可在网址www.profibus.com 下载最新的 GSD 文件（SIEM089D.GSD）。

如果你正在使用一个非西门子的主站，可参考由制造商提供的文件，了解如何用 GSD 文件组态主站。

```
;=============================================
; 带有 DPC31 的 EM 277 PROFIBUS-DPGSD 文件
; 定货号    : 6ES7 277-0AA22-0XA0
; 日  期    : 26-March-2001
;=============================================
#Profibus_DP
;General parameters
GSD_Revision          = 1
Vendor_Name           = "Siemens"
Model_Name            = "EM 277 PROFIBUS-DP"
Revision              = "V1.02"
Ident_Number          = 0x089D
Protocol_Ident        = 0
Station_Type          = 0
FMS_supp              = 0
Hardware_Release      = "1.00"
Software_Release      = "1.02"
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           = 60
MaxTsdr_19.2          = 60
MaxTsdr_45.45         = 250
MaxTsdr_93.75         = 60
MaxTsdr_187.5         = 60
MaxTsdr_500           = 100
MaxTsdr_1.5M          = 150
MaxTsdr_3M            = 250
MaxTsdr_6M            = 450
MaxTsdr_12M           = 800
Redundancy            = 0
Repeater_Ctrl_Sig     = 2
24V_Pins              = 2

; Slave-Specification:
OrderNumber="6ES7 277-0AA2.-0XA0"
Periphery="SIMATIC S5"
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
Max_Diag_Data_Len     = 6
Modul_Offset          = 0
Modular_Station       = 1
Max_Module            = 1
Max_Input_len         = 128
Max_Output_len        = 128
Max_Data_len          = 256

; UserPrmData-Definition
ExtUserPrmData=1 "I/O Offset in the V-memory"
Unsigned16 0 0-10239
EndExtUserPrmData
; UserPrmData: Length and Preset:
User_Prm_Data_Len=3
User_Prm_Data= 0,0,0
Max_User_Prm_Data_Len=3
Ext_User_Prm_Data_Const(0)=0x00,0x00,0x00
Ext_User_Prm_Data_Ref(1)=1
```

```
;=============================================
; Continuation of GSD File
;=============================================

; Module Definition List
Module = "2 Bytes Out/ 2 Bytes In      -" 0x31
EndModule
Module = "8 Bytes Out/ 8 Bytes In      -" 0x37
EndModule
Module = "32 Bytes Out/ 32 Bytes In       -"
0xC0,0x1F,0x1F
EndModule
Module = "64 Bytes Out/ 64 Bytes In       -"
0xC0,0x3F,0x3F
EndModule
Module = "1 Word Out/ 1 Word In        -" 0x70
EndModule
Module = "2 Word Out/ 2 Word In        -" 0x71
EndModule
Module = "4 Word Out/ 4 Word In        -" 0x73
EndModule
Module = "8 Word Out/ 8 Word In        -" 0x77
EndModule
Module = "16 Word Out/ 16 Word In      -" 0x7F
EndModule
Module = "32 Word Out/ 32 Word In          -"
0xC0,0x5F,0x5F
EndModule
Module = "2 Word Out/ 8 Word In            -"
0xC0,0x41,0x47
EndModule
Module = "4 Word Out/ 16 Word In           -"
0xC0,0x43,0x4F
EndModule
Module = "8 Word Out/ 32 Word In           -"
0xC0,0x47,0x5F
EndModule
Module = "8 Word Out/ 2 Word In            -"
0xC0,0x47,0x41
EndModule
Module = "16 Word Out/ 4 Word In           -"
0xC0,0x4F,0x43
EndModule
Module = "32 Word Out/ 8 Word In           -"
0xC0,0x5F,0x47
EndModule
Module = "4 Byte buffer I/O            -" 0xB3
EndModule
Module = "8 Byte buffer I/O            -" 0xB7
EndModule
Module = "12 Byte buffer I/O           -" 0xBB
EndModule
Module = "16 Byte buffer I/O           -" 0xBF
EndModule
```

图 A-25 EM 277 PROFIBUS 模块的 GSD 文件列表

CPU 的 DP 通信的示例程序

以下是一个用语句表生成的 CPU 的例子程序，PROFIBUS-DP 模块位于其 0 号槽，它使用如下所示的 SM 存储器中的 DP 端口信息。这个程序由 SMW 226 确定 DP 缓冲区的地址，由 SMB 228 和 SMB 229 确定了 DP 缓冲区的大小。程序使用这些信息以复制 DP 输出缓冲器中的数据到 CPU 224 的过程映象输出寄存器。与此相似，在 CPU 224 过程映象输入寄存器中的数据可被复制到 V 存储器的输入缓冲区。

注意

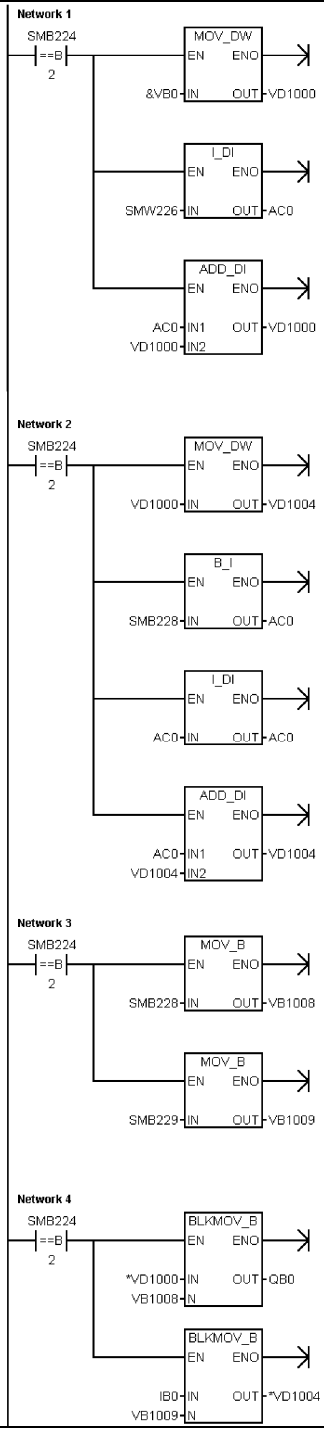
为智能模块定义 SM 区域的方式在版本 2.2 和其后的版本作了改变。

如果您使用版本 2.2 之前的 CPU，您要将所有的智能模块放置在所有非智能模块之前，紧邻 CPU 的位置以确保其兼容性。

以下示例程序中，DP 模块位于 0 号槽，SM 中的 DP 组态信息提供了 DP 从站的组态。程序使用以下数据：

```
//SMW220    DP 模块出错状态
//SMB224    DP 状态
//SMB225    主站地址
//SMW226    V 存储器中输出的偏移
//SMB228    输出数据的字节数
//SMB229    输入数据的字节数
//VD 1000   输出数据的指针
//VD 1004   输入数据的指针
```

到 CPU 的 DP 通讯示例




```
Network1  //计算输出数据指针。如果在数据交换模式：
          //1. 输出缓存区是从 VB0 开始的一个偏移量
          //2. 将 V 区偏移量转换为双整数
          //3. 加上 VB0 的地址，得出输出数据的指针
LDB=    SMB224,2
MOVD    &VB0,VD1000
ITD     SMW226,AC0
+D      AC0,VD1000
Network2  //计算输入数据指针。如果在数据交换模式：
          //1. 拷贝输出数据指针
          //2. 取得输出字节数
          //3. 加上输出数据指针得到一个起始输入数据指针。
LDB=    SMB224,2
MOVD    VD1000,VD1004
BTI     SMB228,AC0
ITD     AC0,AC0
+D      AC0,VD1004
Network3  //设置要拷贝的数据的数量。如果在数据交换模式：
          //1. 取得要拷贝的输出字节数
          //2. 取得要拷贝的输入字节数
LDB=    SMB224,2
MOVB    SMB228,VB1008
MOVB    SMB229,VB1009
Network4  //传送主站输出到 CPU 的输出。拷贝 CPU 的输入
          //到主站输入。如果在数据交换模式：
          //1. 拷贝主站输出到 CPU 输出
          //2. 拷贝 CPU 输入到主站输入。
LDB=    SMB224,2
BMB     *VD1000,QB0,VB1008
BMB     IB0,*VD1004,VB1009
```

EM 241 Modem 模块规范

表 A-42 EM 241 Modem 模块定货号

定货号	扩展模块	输入	输出	可拆卸连接器
6ES7 241-1AA22-0XA0	EM 241 Modem 模块	-	8[1]	否

[1] 八个输出 Q 用作 Modem 功能的逻辑控制，不直接控制任何外部信号。

表 A-43 EM 241 Modem 模块通用规范

定货号	模块名称及描述	尺寸(mm) (W×H×D)	重量	功耗	VDC 要求 +5VDC	VDC 要求 +24VDC
6ES7 241-1AA22-0XA0	EM 241 Modem 模块	71.2×80×62	190g	2.1W	80mA	70mA

表 A-44 EM 241 Modem 模块规范

常规	6ES7 241-1AA22-0XA0
电话连接	
隔离 （电话线到逻辑和现场）	1500 VAC（电的）
物理连接	RJ11（6 位 4 线）
Modem 标准	Bell103,Bell212,V.21,V.22bis,V.23c,V.32,V.32bis,V.24（缺省）
安全特性	密码 回拨
拨号	脉冲或语音
信息协议	数字 TAP（字母及数字） UCP 命令 1,30,51
工业协议	Modbus PPI
输入电源要求	
电压范围 隔离（现场电源到逻辑）	20.4-28.8 VDC 500VAC，1 分钟

EM 241 Modem 模块替代连于 CPU 通讯口的外部 Modem 功能。与一个连有 EM 241 的 S7-200 系统进行通讯，您只需在远端的个人计算机上连接一个外置 Modem 并安装。STEP7-Micro/WIN。

有关网络通讯的组态信息可参见第七章。为 Modem 模块编程以及该模块的高级特性可参见第十章。

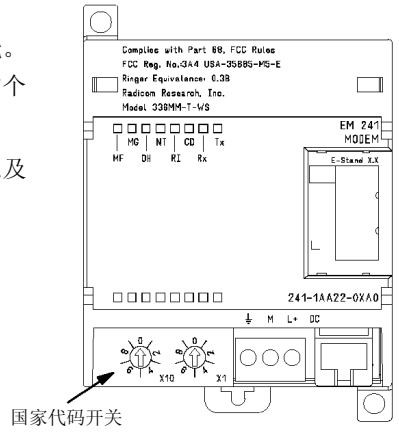


图 A-26 EM 241 Modem 模块接线端子图

安装 EM 241

按以下步骤安装 EM 241：

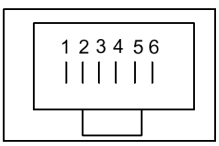
1. 将 EM 241 安装在 DIN 导轨上并插上扁平电缆。
2. 从 CPU 传感器电源或外部电源连接 24VDC，装接地端连到系统的地。
3. 将电话线连至 RJ11 插座。
4. 按照表 A-45 设置国家代码。您必须在 CPU 上电前设置开关，以便读取正确的国家代码。
5. CPU 上电。绿色的 MG（模板好）灯应接通。
6. 现在 EM 241 已为通讯作好准备。

表 A-45 EM 241 支持的国家代码

代码	国家	电话标准
01	澳大利亚	CTR21
02	比利时	CTR21
05	加拿大	IC CS03
08	丹麦	CTR21
09	芬兰	CTR21
10	法国	CTR21
11	德国	CTR21
12	希腊	CTR21
16	爱尔兰	CTR21
18	意大利	CTR21
22	卢森堡	CTR21
25	荷兰	CTR21
27	挪威	CTR21
30	波兰	CTR21
34	西班牙	CTR21
35	瑞典	CTR21
36	瑞士	CTR21
38	英国	CTR21
39	美国	FCC Part 68

RJ11 插座

图 A-27 所示为 RJ11 插座的详细图示。可以通过适配器与其它标准的电话接口相连。更多的信息请参考您的适配器的文档。



插针	描述
3	Ring
4	Tip

图 A-27 RJ11 插座视图

注意

雷击或其它不可预期的高压作用于电话线上会损坏 EM 241 Modem 模块。

使用经济实用的电话线冲击保护装置，比如常见的用于个人计算机 Modem 的保护装置。冲击保护装置为保护 EM 241 Modem 模块，可能会损坏。选择一个能够正面指示它在工作的冲击保护装置。

定期检查冲击保护装置以确保 EM 241 Modem 模块能够得到持续的保护。

EM 253 位控模块规范

表 A-46 EM 253 位控模块定货号

定货号	扩展模块	输入	输出	可拆卸连接器
6ES7 253-1AA22-0XA0	EM 253 位控模块	-	8[1]	是

[1] 八位输出 Q 用作运动功能的逻辑控制，不直接控制任何外部信号。

表 A-47 EM 253 位控模块通用规范

定货号	模块名称和描述	尺寸(mm) (W×H×D)	重量	功耗	VDC 要求	
					+5 VDC	+24 VDC
6ES7 253-1AA22-0XA0	EM 253 位控模块	71.2×80×62	0.190kg	2.5W	190mA	见下面

表 A-48 EM 253 位控模块规范

常规	6ES7 253-1AA22-0XA0
输入特性	
输入数量	5 点
输入类型	漏型/源型（IEC 类型 1 漏型，除 ZP 外）
输入电压 允许的最大持续电压 STP,RPS,LMT+,LMT- ZP 浪涌（所有输入） 额定值 STP,RPS,LMT+,LMT- ZP 逻辑“1”信号（最小） STP,RPS,LMT+,LMT- ZP 逻辑“0”信号（最大） STP,RPS,LMT+,LMT- ZP	 30 VDC 30 VDC，20mA,最大 35 VDC，0.5 秒 24 VDC，4mA，正常 24 VDC，15mA，正常 15 VDC，2.5mA，最小 3 VDC，8.0mA，最小 5 VDC，1mA，最大 1 VDC，1mA，最大
隔离（现场到逻辑） 光电隔离 组隔离	 500 VAC，1 分钟 1 点用于 STP，RPS 和 ZP 2 点用于 LMT+和 LMT-
输入延迟时间 STP,RPS,LMT+,LMT- ZP（可计脉冲宽度）	 0.2ms-12.8ms，用户选择 2 μ sec，最小
连接 2 线接近开关传感器（Bero） 允许的源电流	 1mA，最大
电缆长度 未屏蔽 STP,RPS,LMT+,LMT- ZP 屏蔽 STP,RPS,LMT+,LMT- ZP	 30 米 不建议使用 100 米 10 米
同时接通的输入数 55℃	 5

表 A-48 EM 253 位控模块规范，续

常规	6ES7 253-1AA22-0XA0	
输出特性		
集成的输出数 输出字节 P0+,P0-,P1+,P1- P0,P1,DIS,CLR	6 点（4 个信号） 驱动 漏极输出	
输出电压 P0,P1,RS-422 驱动，差分输出电压 断路 接入带有 200Ω串行电阻的光耦二极管 100Ω负载 54Ω负载 P0，P1，DIS，CLR 漏型 建议电压，开路 允许电压，开路 漏电流 接通状态电阻 断开状态下漏电流，30VDC 上拉电阻，到 T1 的漏型输出	 3.5V 典型 2.8V 最小 1.5V 最小 1.0V 最小 5VDC，来自模块 30VDC[1] 50mA 最大 15Ω最大 10μA 最大 3.3KΩ[2]	
输出电流 输出组数 接通的输出数（最大） 每点漏电流 P0，P1，DIS，CLR 过载保护	 1 6 10μA 最大 无	
隔离（现场到逻辑） 光电隔离	 500 VAC，1 分钟	
输出时延 DIS，CLR：断开到接通/接通到断开	 30μs，最大	
脉冲畸变 P0,P1,输出，RS-422 驱动，100Ω外部负载 P0,P1 输出，漏型，5V/470Ω外部负载	 75ns 最大 300ns 最大	
切换频率 P0+，P0-，P1+，P1-，P0 和 P1	 200kHz	
电缆长度 未屏蔽 屏蔽	 不推荐 10 米	
电源		
L+提供电压 逻辑提供输出 L+提供相对于 5VDC 负载的电流 负载电流 0mA（无负载） 200mA（额定负载）	11-30 VDC +5 VDC+/-10%，200mA 最大 <u>12 VDC 输入</u> 120mA 300mA	 <u>24 VDC 输入</u> 70mA 130mA
隔离 L+电源到逻辑 L+电源到输入 L+电源到输出	 500 VAC，1 分钟 500 VAC，1 分钟 无	
反向极性	L+输入和+5V 输出有二极管保护。在 M 端接入正向电压，就输出点的连接而言，可能导致损害性的电流产生。	

[1] 高于 5VDC 的漏型输出可能会增加射频干扰使之超过允许的限定。您的系统或接线需要射频干扰抑制措施。

[2] 根据您的脉冲接收器和电缆，一个额外的外部上拉电阻可能会改善脉冲信号的质量和噪声抑制功能。

EM 253 位控模块状态 LED

位控模块的状态 LED 如表 A-49 所示。

表 A-49 位控模块状态 LED

本地 I/O	LED	颜色	功能描述
-	MF	红色	模板检测到一个致命故障时接通
-	MG	绿色	无故障时接通，检测到组态错误时以 1Hz 频率闪烁
-	PWR	绿色	当 L+和 M 端有 24VDC 供电时接通
输入	STOP	绿色	输入 stop 接通时亮
输入	RPS	绿色	参考点切换输入接通时亮
输入	ZP	绿色	零脉冲输入接通时亮
输入	LMT-	绿色	负向限位输入接通时亮
输入	LMT+	绿色	正向限位输入接通时亮
输出	P0	绿色	P0 输出触发时亮
输出	P1	绿色	P1 输出触发或该输出指示正向运动时亮
输出	DIS	绿色	DIS 输出激活时亮
输出	CLR	绿色	当清除偏差计数器输出激活时亮

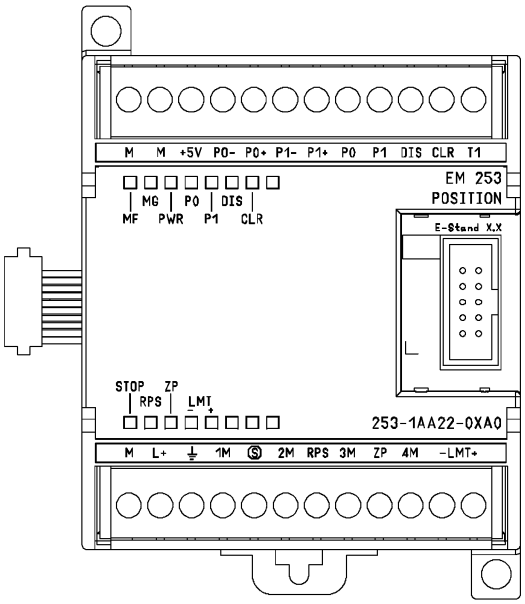


图 A-28 EM 253 位控模块

接线图

下图中各端子没有按序排列。端子的排列请参见图 A-28。

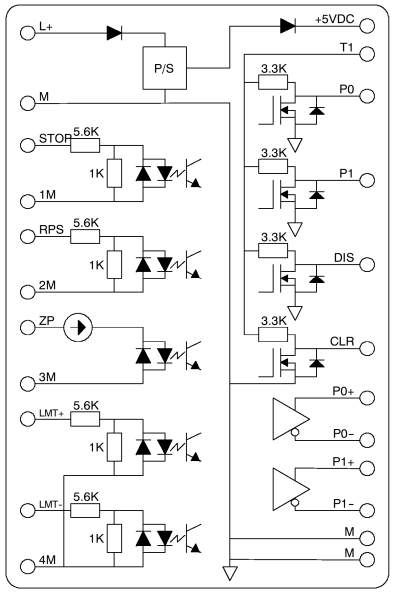
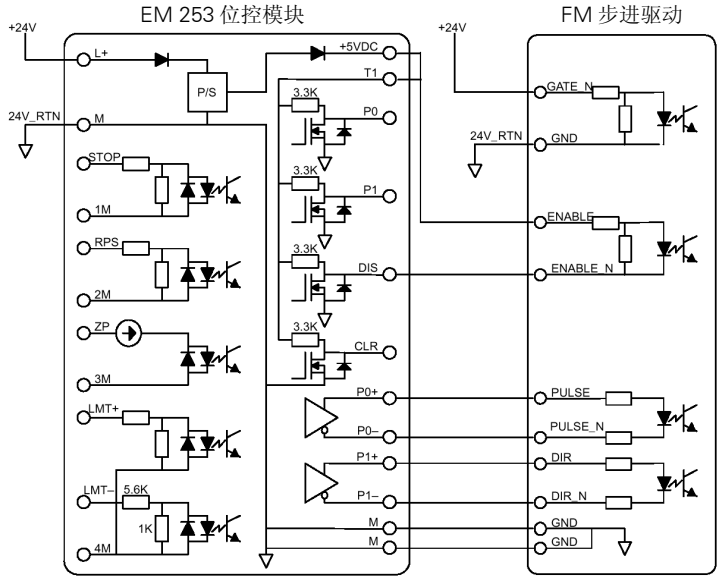
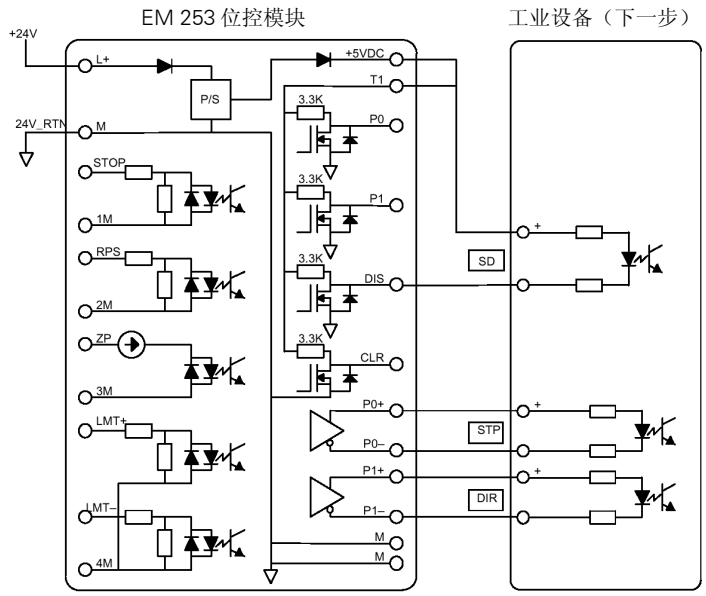


图 A-29 EM 253 位控模块输入和输出内部示意图



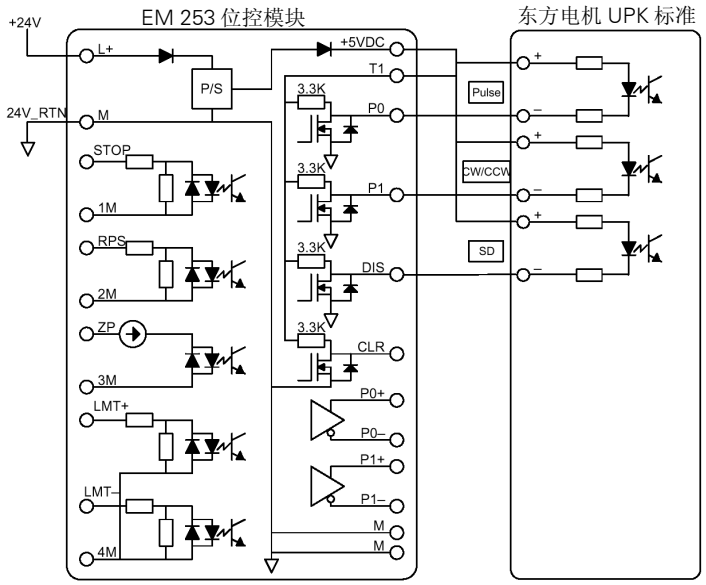
端子未按序排列。端子排列请参见图 A-28。

图 A-30 连接 EM 253 位控模块和 FM 步进驱动



端子未按序排列。端子排列请参见图 A-28。

图 A-31 连接 EM 253 位控模块和一个工业设备（下一步）



端子未按序排列。端子排列请参见图 A-28。

图 A-32 连接 EM 253 位控模块和一个东方电机 UPK 标准

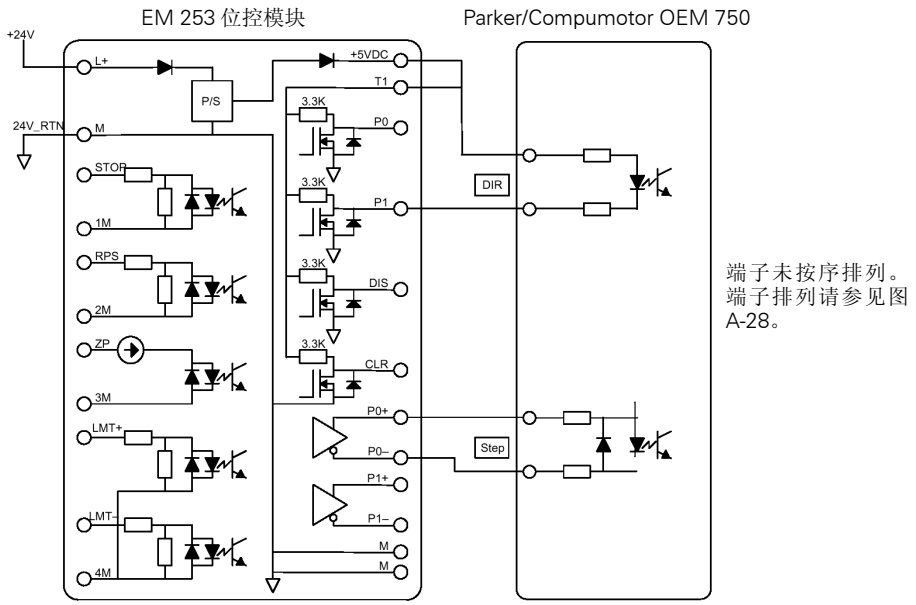


图 A-33 连接 EM 253 位控模块和 Parker/Compumotor OEM 750

AS-I 接口（CP 243-2）模块规范

表 A-50 AS-I 接口（CP 243-2）模块定货号

定货号	扩展模块	输入	输出	可拆卸连接
6GK7 243-2AX01-0XA0	ASI-接口（CP 243-2）模块	8 数字量 8 模拟量	8 数字量 8 模拟量	是

表 A-51 AS-接口（CP 243-2）模块通用规范

定货号	模块名称和描述	尺寸(mm) (W×H×D)	重量	功耗	VDC 要求 +5 VDC	 来自 AS 接口
6GK7 243-2AX01-0XA0	ASI-接口（CP 243-2）模块	71×80×62	250g 左右	3.7W	220mA	100mA

表 A-52 AS-I 接口（CP 243-2）模块规范

常规	6GK7 243-2AX01-0XA0
循环时间	5ms，31 个从站 10ms，62 个 AS-I 从站使用扩展地址模式
组态	使用前面板上的按钮进行组态，或使用完全组态命令 （参考 CP 243-2 AS-I 接口主站手册中关于 AS-I 命令的说明）
支持的 AS-I 主站协议	M1e
AS-I 电缆附件	通过 S7-200 端子块。端子 1 到 3 或从端子 2 到 4 的允许电流负载为 3A。
地址范围	一个带有 8 入/8 出的数字量模块。 一个带有 8 通道模拟输入和 8 通道模拟输出的模拟量模块

特性

图 A-34 所示为 S7-200 CPU22x 系列 AS-I 接口主站。在一个 S7-200 上同时可操作多达两个 AS-I 接口模块，增加了可使用的数字和模拟输入/输出（每个 CP AS-I 接口最多 124 数字输入/124 数字输出）。由于使用按钮组态从而节省了设置时间。LED 通过显示 CP 和所连接的从站的状态以及监控 AS-I 接口主电压，减少了停车时间。

AS-I 接口模块有以下特点：

- 支持模拟量模块
- 支持所有的主站功能并能够最多连接 62 个 AS-I 接口从站
- 前面板上的 LED 显示所连从站的运行状态和就绪状态
- 前面板上的 LED 显示错误（包括 AS-I 接口电压错误，组态错误）
- 两个端子可直接连接 AS-I 接口电缆
- 两个按钮显示从站的状态信息，切换运行模式，并可将现有的组态作为设置组态

运行

在 S7-200 的内部映像区中，CP 243-2 占用一个数字输入字节（状态字节），一个数字输出字节（控制字节），以及 8 个模拟量输入和 8 个模拟量输出字。因此，CP 243-2 占用两个逻辑模块位置。通过用户程序，状态和控制字节可用来设置 CP 243-2 的运行模式。取决于 CP 243-2 的运行模式，它可以存储 AS-I 接口从站的 I/O 数据，或是诊断值，或是在 S7-200 的模拟量地址区域中，启动主站调用（例如，改变一个从站的地址）。

所有连接的 AS-I 接口从站都可通过按压一个按钮进行配置。CP 的进一步配置是不必要的。

注意

当使用 CP 243-2 模块时，必须禁止 PLC 中的模拟量滤波。如果不禁止 PLC 中的模拟量滤波，则会破坏数字点数据，而且出错条件也不会在模拟量控制字中以位的状态来返回。

应确实保证，PLC 中的模拟量滤波已被禁止。

功能

CP 243-2 是符合 MI 主站行规的 AS-I 接口主站，这就是说，它支持所有规定的功能。因而它可以借助于双地址赋值（A-B），使 AS-I 接口上能运行最多 31 个数字从站。

CP 243-2 可设置成二种不同的工作模式：

- 标准模式：存取 AS-I 接口从站的 I/O 数据
- 扩展模式：主站调用（例如写参数）或诊断值请求

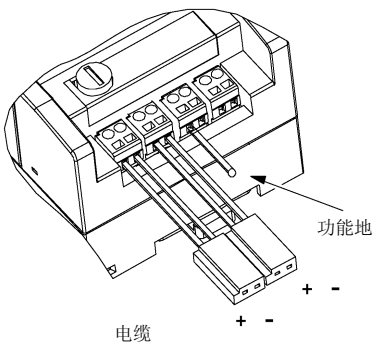


图 A-34 CP 243-2 通讯处理器接线

连接

AS-I 接口模块有以下连接：

- 两个连至 AS-I 接口模块电缆的连接（内置）
- 一个功能地的连接

如图 A-35 所示，端子位于前面板盖下。

注意

AS-I 接口模块触点的负载能力量大是 3A。如果 AS-I 接口模块电缆上的电流超过该值，该 AS-I 接口则不能连接到 AS-I 电缆上，而应由一个分开的电缆来连接（这种情况下，只使用 AS-I 接口模块的一对端子）。该 AS-I 接口必须通过接地端子接到接地导体上。

提示

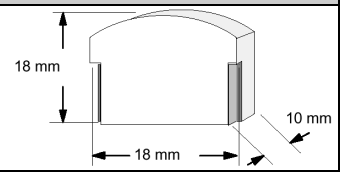
AS-I 接口模块有功能地的连接。该接口应以尽可能少的电阻连至 PE 导体。

其它信息

关于 CP 243-2 AS-I 接口主站更多的信息，请参考 SIMATIC NET CP 243-2 AS-I 接口主站手册。

可选卡件

卡件	描述	定货号
存储卡	存储卡：程序、数据和组态	6ES7 291-8GE20-0XA0
带电池的实时时钟卡	时钟卡精度： 2 分钟/月，25℃ 7 分钟/月，0℃-55℃	6ES7 297-1AA20-0XA0
电池卡	电池卡（数据保持时间）：200 天，典型	6ES7 291-8BA20-0XA0

常规特性		尺寸
电池 尺寸 类型	3V，30mA 小时 Renata CR 1025 9.9mmx2.5mm 锂电池<0.6g	

存储卡为所有 CPU（CPU221、CPU222、CPU224、CPU226 和 CPU226XM）存储全部程序块和数据块。

扩展电缆

常规特性（6ES7 290-6AA20-0XA0）	
电缆长度	0.8m
重量	25g
接口类型	10 针扁平电缆

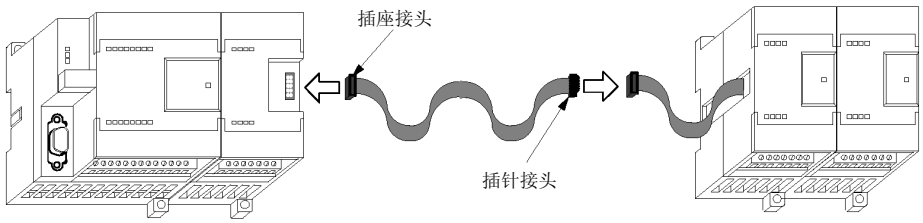


图 A-35 I/O 扩展电缆的典型安装

提示

在 CPU/扩展模块的连接中只能使用一条扩展电缆。

PC/PPI 电缆

PC/PPI 电缆（6ES7 901-3BF21-0XA0） 总体特性	
电源电压	14.4-28.8VDC
24V 电源的电流	50mA RMS max.
方向改变延时：接收的 RS232 起始位边沿到发送的 RS485 起始位边沿	12. μ smax.
方向改变延时：接收的 RS232 停止位边沿到发送的 RS485 停止位边沿	1.4 (1.4×11/baud)=1.6ms@9600 baud
传送延时	4 μ smax., RS-485 到 RS-232, 1.2 μ smax., RS-232 到 RS-485
隔离	500 VDC
RS-485 一侧的电气特性	
共模电压范围	-7V 到+12V,1second 3V RMS
接收器输入阻抗	5.4K Ω min
终端/偏置	10K Ω to+5V on B, PROFIBUS pin3 10K Ω to GND on A, PROFIBUS pin8
接收器阀值/灵敏度	+/-0.2V, 60mV typ, hysteresis
发送器差分输出电压	2V min., R_L=100 Ω 1.5V min., R_L=54 Ω
RS-232 一侧的电气特性	
接收器输入阻抗	3K Ω min
接收器阀值/灵敏度	0.8V min.low, 2.4V max,high, 0.5V typical hysteresis
发送器输出电压	+/-5V min, RL=3K Ω

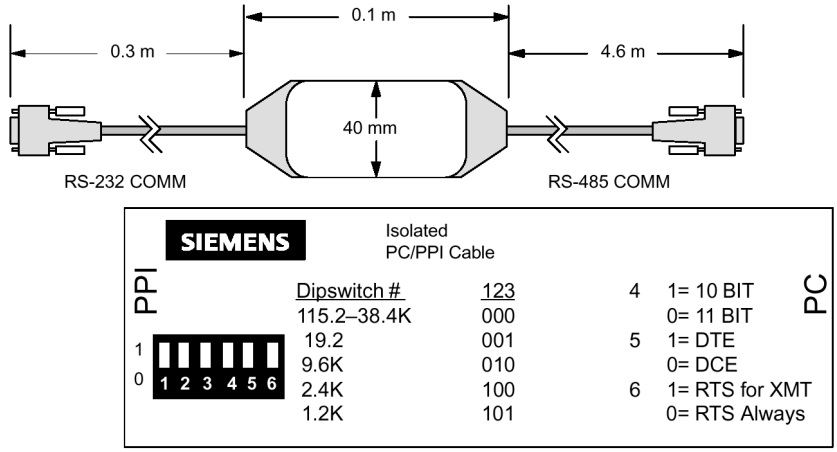


图 A-36 PC/PPI 电缆尺寸

表 A-53 PC/PPI 电缆的开关选择

波特率 开关 1,2,3*		Modem 操作开关 4*		DCE/DTE 选择开关 5*		DTE 的 RTS 选择 开关 6*	
115200-38400	000	11 位 modem	0	DCE	0	RTS 总激活	0
19200	001	10 位 modem	1	DTE	1	RTS PLC 传送时激活	1
9600	010						
4800	011						
2400	100						
1200	101						
600	110						

* 开关：1=上；0=下

表 A-54 RS-485 至 RS-232 DCE 连接器的引脚

RS-485		RS-232	
针号	信号说明	针号	信号说明
1	地（RS-485 逻辑地）	1	数据载波检测（DCD）（不用）
2	24V 返回（RS-485 逻辑地）	2	接收数据（RD） （从 PC/PPI 电缆输出）
3	信号 B（RxD/TxD+）	3	发送数据（TD） （输入到 PC/PPI）
4	RTS（TTL 电平）	4	数据终端就绪（DTR）（不用）
5	地（RS-485 逻辑地）	5	地（RS-232 逻辑地）
6	未连接	6	数据设置就绪（DSR）（不用）
7	24V 电源	7	申请发送（RTS）（不用）
8	信号 A（RxD/TxD-）	8	清除发送（CTS）（不用）
9	协议选择	9	振铃指示器（RI）（不用）

表 A-55 RS485 与 RS232 DTE 连接器的引脚

RS-485		RS-232	
针号	RS-485 连接器引针	针号	RS-232 DTE 连接器引针 [1]
1	地（RS-485 逻辑地）	1	数据载波检测（DCD）（不用）
2	24V 返回（RS-485 逻辑地）	2	接收数据（RD） （从 PC/PPI 电缆输出）
3	信号 B（RxD/TxD+）	3	发送数据（TD） （输入到 PC/PPI）
4	RTS（TTL 电平）	4	数据终端就绪（DTR）（不用）
5	地（RS-485 逻辑地）	5	地（RS-232 逻辑地）
6	未连接	6	数据设置就绪（DSR）（不用）
7	24V 电源	7	申请发送（RTS） (从 PC/PPI 电缆输出)（通过开关选择）
8	信号 A（RxD/TxD-）	8	清除发送（CTS）（不用）
9	协议选择	9	振铃指示器（RI）（不用）

[1] 调制解调器需要一个阴-阳型 9 到 25 针的转换

输入仿真器

订货号	8 输入仿真器 6ES7 274-1XF00-0XA0	14 输入仿真器 6ES7 274-1XH00-0XA00	24 输入仿真器 6ES7 274-1XK00-0XA
尺寸 （L×W×D）	61×36×22mm (2.4×1.4×0.85 in.)	91×36×22mm (3.6×1.4×0.85 in.)	147×36×25mm (3.6×1.4×0.85 in.)
重量	0.02Kg(0.04lb.)	0.03Kg(0,08lb.)	0.04Kg(0,08lb.)
点数	8	14	24

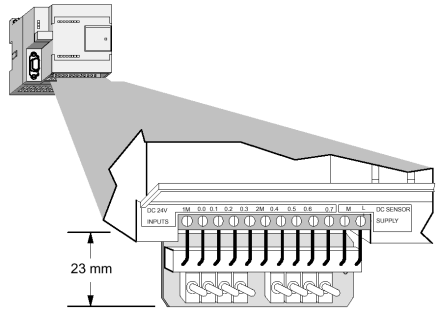


图 A-37 输入仿真器的安装

注

这些输入仿真器未被证明能够用于 Class 1 DIV 2 或 Class 1 Zone 2 危险环境。这些开关可能会引起火花。

这些输入仿真器不能用于 Class 1 DIV 2 或 Class 1 Zone 2 危险环境。

CP 243-1 工业以太网通讯处理器

说明 订货号	CP 243-1 工业以太网通讯处理器 6GK7 243-1EX00-0XE0
物理结构 • 模块化格式 • 尺寸（B×H×T） • 重量	 S7-200 扩展模板 71.2×80×62 mm 大约150 g
传输速率 闪存 SDRAM	10 Mbit/s 和 100 Mbit/s 1 Mbyte 8 Mbyte
接口 工业以太网连接（10/100 Mbit/s）	 8针RJ45插座
输入电压	DC +24 V（+/-5%）
电流消耗 • 底板总线 • 外部 DC 24 V	 55 mA 60 mA
功率损耗	1.75 W
最大连接数量	最多8个S7连接（XPUT/XGET和READ/WRITE）+ 1个STEP 7 Micro/WIN 32连接
允许环境条件： • 工作温度 • 运输/贮存温度 • 最大相对湿度 • 安装高度	 水平安装，0℃ - +55℃ 垂直安装，0℃ - +45℃ -40℃ - +70℃ 95%，+25℃时 海拔2000 m以下，海拔越高，冷却越没有效果，需要降低最大工作温度。
防护等级 以太网标准 标准	IP 20 IEEE 802.3 CE 标志 UL 508 或 cULus CSA C22.2 Number 142 或 cULus FM 3611 EN 50081-2 EN 60529 EN 61000-6-2 EN 61131-2
启动时间或复位后的重新启动时间	约10秒
用户数据数量	作为客户机：对于XPUT/XGET，212个字节 作为服务器：对于XGET或READ，222个字节 对于XPUT或WRITE，212个字节

下表中所示数值为进行相关动作后的时间。

以太网

表 18：以太网中的超时

含义	超时操作	固定时间，单位[秒]
最大等待时间，直到通过 TCP/IP 接收到一个报文	放弃电报分段，CP 关闭连接	3
最大等待时间，直到通过 TCP/IP 发送一个报文	中止发送，CP 关闭连接	3
CP 的最大等待时间，直到 CPU 响应一个由外部客户机发送给它的命令。	放弃操作，CP 等待新的命令，客户机没有接收到响应。	60
登出时间，如果所有通道都占用，并且 MW 还没有连接（MW 可以一直建立连接）	不能通过 TSAP 分配的最早连接的 TCP/IP 服务器将中止连接，并等待重新建立连接	60
处理外来连接建立请求的间隔时间	下一连接建立延迟时间	1
客户机重新建立连接未成功后的等待时间	反复尝试建立连接	10
客户机确认连接建立请求的等待时间	客户机关闭连接，并重新连接	6
BOOT 命令成功执行的最长时间	重新尝试从 S7- 200 CPU 存储器或通过 BOOTP 接收一个有效的组态	24 次尝试，间隔 2.5 秒，共 60 秒

S7 总线：

表 19：S7 总线中的超时

含义	超时操作	固定时间，单位[秒]
通过 S7，CP 243-1 和 S7-200 CPU 之间的一个通讯循环的最长时间 **注意：** 每个读/写命令，一般在客户机一侧需要 3 个循环，在服务器一侧需要 1 个循环。如果用户程序的执行时间非常长，并且同时在许多通道进行通讯，读/写命令的执行时间可以长一些。	放弃命令，CP 243-1 重新热启动	10（每个循环）
在 CP 引导装入期间 CDB / NDB 成功传输的超时时间	CP 243-1 重新热启动	120
S7 CPU 在其引导装入时物理识别 CP 243-1 后的时间	CP 243-1 重新热启动	1

电源的设计

B

S7-200 本机单元有一个内部电源，它为本机单元、扩展模块以及 24VDC 用户供电。利用下面提供的要点来决定本机单元电源能为你的配置提供多少功率（或电流）。

电源要求

每一个 S7-200 CPU 模块提供 5VDC 和 24VDC 电源：

- 每一个 CPU 模块都有一个 24VDC 传感器电源，它为本机输入点和扩展模块继电器线圈提供 24VDC。如果电源要求超出了 CPU 模块 24VDC 电源的定额，你可以增加一个外部 24VDC 电源来供给扩展模块的 24VDC。
- 当有扩展模块连接时 CPU 模块也为其提供 5V 电源。如果扩展模块的 5V 电源需求超出了 CPU 模块的电源定额，你必须卸下扩展模块，直到需求在电源预定值之内才行。

附录 A 数据表提供了有关 CPU 模块电源设计定额的信息，以及扩展模块所需电源设计定额的信息。

警告：

将 S7-200 DC 传感器电源与外部 24VDC 电源采用并联连接时，将会导致两个电源的竞争而影响它们各自的输出。

这种竞争的结果要缩短设备的寿命，或者使得一个电源或两者同时失效，并且使 PLC 系统产生不正确的操作。

S7-200 DC 传感器电源和外部电源应该在不同的点上提供电源，两者之间只能有一个连接点。

计算举例

表 B-1 所示的是一个 S7-200 Micro PLC 电源需求量计算的例子，它包括以下模块：

- CPU 224 AC/DC/继电器
- 3 个 EM 223 数字输入 8/继电器输出 8
- 1 个 EM 221 数字输入 8

该配置共有 46 个输入和 34 个输出。

在本例中，CPU 模块为扩展模块提供了足够的 5V 电源，但是它没有给所有的输入和输出线圈提供足够的 24VDC 电源。本例中 I/O 要求 400mA 的 24VDC 电源，但是 CPU 只能提供 280mA。本配置需要提供额外的 120mA 电流。

表 B-1 针对一个配置实例的电源计算

CPU 电源预算	5 VDC	24 VDC
CPU 224 AC/DC/继电器	660mA	280mA
	减	
系统要求	5 VDC	24 VDC
CPU 224,14 输入		14＊4mA= 56mA
3 EM 223,5V 电源需求	3*80mA= 240mA	
1 EM 221,5V 电源需求	1*30mA= 30mA	
3 EM 223,每个 8 输入		3*8*4mA= 96mA
3 EM 223,每个 8 继电器线圈		3*8*9mA= 216mA
1 EM 221,每个 8 输入		8*4mA= 32mA
总需求	270mA	400mA
	等于	
电流平衡	5 VDC	24 VDC
总电流平衡	剩 390mA	缺 120mA

计算你的电能需要

利用下表计算 CPU 可以为你的配置提供多少电源（或电流）。请参考附录 A 给出的 CPU 的电源预算和扩展模块的电源消耗。

电源预算	5 VDC	24 VDC
	减	
系统要求	5 VDC	24 VDC
	基本单元	
总需求		
	等于	
电流平衡	5 VDC	24 VDC
总电流平衡		

错误代码

C

有关错误代码的信息有助于查找 S7-200 CPU 的问题。

本章概述

致命错误代码和信息

致命错误会导致 CPU 停止执行用户程序。依据错误的严重性，一个致命错误会导致 CPU 无法执行某个或所有功能。处理致命错误的目标是，使 CPU 进入安全状态，可以对当前存在的错误状况进行询问并响应。

当一个致命错误发生时，CPU 执行以下任务：

- 进入 STOP（停止）方式
- 点亮系统致命系统错误和 STOP（停止）LED 指示灯
- 断开输出

这种状态将会持续到错误清除之后。在主菜单中使用菜单命令 PLC>Information 可查看错误代码。表 C-1 列出了从 S7-200 上可读到的致命错误代码及其描述。

表 C-1 从 CPU 读出的致命错误代码及其描述

错误代码	描述
0000	无致命错误
0001	用户程序校验和错误
0002	编译后的梯形图程序校验和错误
0003	扫描看门狗超时错误
0004	内部 EEPROM 错误
0005	内部 EEPROM 用户程序校验和错误
0006	内部 EEPROM 配置参数（SDB0）校验和错误
0007	内部 EEPROM 强制数据校验和错误
0008	内部 EEPROM 缺省输出表值校验和错误
0009	内部 EEPROM 用户数据 DB1 校验和错误
000A	存储器卡失灵
000B	存储器卡上用户程序校验和错误
000C	存储卡配置参数（SDB0）校验和错误
000D	存储器卡强制数据校验和错误
000E	存储器卡缺省输出表值校验和错误
000F	存储器卡用户数据 DB1 校验和错误
0010	内部软件错误
0011[1]	比较接点间接寻址错误
00012[1]	比较接点非法值错误
0013	存储器卡空，或者 CPU 不识别该程序
0014[1]	比较接点范围错误

[1] 比较接点错误是唯一的一种错误，即能产生致命错误又能产生非致命错误。产生非致命错误是因为存储错误的程序地址。

运行程序错误

在程序的正常运行中，可能会产生非致命错误（如寻址错误）。在这种情况下，CPU 产生一个非致命运行时刻错误代码。表 C-2 列出了这些非致命错误代码及其描述。

表 C-2 运行程序错误

错误代码	运行程序错误（非致命）
0000	无错误
0001	执行 HDEF 之前，HSC 不允许
0002	输入中断分配冲突，已分配给 HSC
0003	到 HSC 的输入分配冲突，已分配给输入中断
0004	在中断程序中企图执行 ENI、DISI 或 HDEF 指令
0005	第一个 HSC/PLS 未执行完之前，又企图执行同编号的第二个 HSC/PLS（中断程序中的 HSC 同主程序中的 HSC/PLS 冲突。）
0006	间接寻址错误
0007	TODW（写实时时钟）或 TODR（读实时时钟）数据错误
0008	用户子程序嵌套层数超过规定
0009	在程序执行 XMT 或 RCV 时，通讯口 0 又执行另一条 XMT/RCV 指令
000A	在同一 HSC 执行时，又企图用 HDEF 指令再定义该 HSC
000B	在通讯口 1 上同时执行 XMT/RCV 指令
000C	时钟存储卡不存在
000D	重新定义已经使用地脉冲输出
000E	PTO 个数设为 0
0091	范围错误（带地址信息）：检查操作数范围
0092	某条指令的计数域错误（带计数信息）：确认最大计数范围
0094	范围错误（带地址信息）：写无效存储器
009A	用户中断程序试图转换成自由口模式
009B	非法指针（字符串操作中起始位置值指定为 0）

编译规则错误

当你下装一个程序时，CPU 将编译该程序。如果 CPU 发现程序违反编译规则（如非法指令），那么 CPU 就会停止下装程序，并生成一个非致命编译规则错误代码。表 C-3 列出了违反编译规则所生成的这些错误代码及其描述。

表 C-3 编译规则错误

错误代码	编译错误（非致命）
0080	程序太大无法编译：你必须缩短程序
0081	堆栈溢出：把一个程序段分成多个
0082	非法指令：检查指令助记符
0083	无 MEND 或主程序中有不允许的指令：加条 MEND 或删去不正确的指令
0084	保留
0085	无 FOR 指令：加上 FOR 指令或删除 NEXT 指令
0086	无 NEXT：加条 NEXT 指令，或删除 FOR 指令
0087	无标号（LBL、INT、SBR）：加上合适标号
0088	无 RET 或子程序中有不允许的指令：加条 RET 或删去不正确指令
0089	无 RETI 或中断程序中有不允许的指令：加条 RETI 或删去不正确指令
008A	保留
008B	从/向一个 SCR 段的非法跳转
008C	标号重复（LBL、INT、SBR）：重新命名标号
008D	非法标号（LBL、INT、SBR）：确保标号数在允许范围内
0090	非法参数：确认指令所允许的参数
0091	范围错误（带地址信息）：检查操作数范围
0092	指令计数域错误（带计数信息）：确认最大计数范围
0093	FOR/NEXT 嵌套层数超出范围
0095	无 LSCR 指令（装载 SCR）
0096	无 SCRE 指令（SCR 结束）或 SCRE 前面有不允许的指令
0097	用户程序包含非数字编码的和数字编码的 EV/ED 指令
0098	在运行模式进行非法编辑（试图编辑非数字编码的 EV/ED 指令）
0099	隐含程序段太多（HIDE 指令）
009B	非法指针（字符串操作中起始位置值指定为 0）
009C	超出最大指令长度

特殊存储器（SM）标志位

D

特殊存储器标志位提供大量的状态和控制功能，并能起到在 CPU 和用户程序之间交换信息的作用。特殊存储器标志位能以位、字节、字或双字使用。

本章概述

SMB0：状态位

如表 D-1 所示，SMB0 有 8 个状态位，在每个扫描周期的末尾，由 S7-200 CPU 更新这些位。

表 D-1 特殊存储器字节 SMB0（SM0.0 – SM0.7）

SM 位	描述（只读）
SM0.0	该位始终为 1
SM0.1	该位在首次扫描时为 1，用途之一是调用初始化子程序
SM0.2	若保持数据丢失，则该位在一个扫描周期中为 1。该位可用作错误存储器位，或用来调用特殊启动顺序功能。
SM0.3	开机后进入 RUN 方式，该位将 ON 一个扫描周期，该位可用作在启动操作之前给设备提供一个预热时间
SM0.4	该位提供了一个时钟脉冲，30 秒为 1，30 秒为 0，周期为一分钟，它提供了一个简单易用的延时或 1 分钟的时钟脉冲
SM0.5	该位提供了一个时钟脉冲，0.5 秒为 1，0.5 秒为 0，周期为 1 秒钟。它提供了一个简单易用的延时或 1 秒钟的时钟脉冲
SM0.6	该位为扫描时钟，本次扫描时置 1，下次扫描时置 0。可用作扫描计数器的输入
SM0.7	该位指示 CPU 工作方式开关的位置（0 为 TERM 位置，1 为 RUN 位置）。当开关在 RUN 位置时，用该位可使自由端口通信方式有效，那么当切换至 TERM 位置时，同编程设备的正常通讯也会有效。

SMB1：状态位

如表 D-2 所示，SMB1 包含了各种潜在的错误提示。这些位可由指令在执行时进行置位或复位。

表 D-2　特殊存储器字节 SMB1（SM1.0 – SM1.7）

SM 位	描述（只读）
SM1.0	当执行某些指令，其结果为 0 时，将该位置 1
SM1.1	当执行某些指令，其结果溢出或查出非法数值时，将该位置 1
SM1.2	当执行数学运算，其结果为负数时，将该位置 1
SM1.3	试图除以零时，将该位置 1
SM1.4	当执行 ATT（Add to Table）指令时，试图超出表范围时，将该位置 1
SM1.5	当执行 LIFO 或 FIFO 指令，试图从空表中读数时，将该位置 1
SM1.6	当试图把一个非 BCD 数转换为二进制数时，将该位置 1
SM1.7	当 ASCII 码不能转换为有效的十六进制数时，将该位置 1

SMB2：自由口接收字符

SMB2 为自由端口接收字符缓冲区。如表 D-3 所示，在自由端口通信方式下，接收到的每个字符都放在这里，便于梯形图程序存取。

提示：

SMB2 和 SMB3 由 0 口和 1 口共用。当 0 口接收到字符并使得与该事件（中断事件 8）相连的中断程序执行时，SMB2 包含 0 口接收到的字符，而 SMB3 包含该字符的校验状态。当 1 口接收到字符并使得与该事件（中断事件 25）相连的中断程序执行时，SMB2 包含 1 口接收到的字符，而 SMB3 包含该字符的校验状态。

表 D-3　特殊存储器字节 SMB2

SM 位	描述（只读）
SMB2	在自由端口通讯方式下，该字符存储从口 0 或口 1 接收到的每一个字符

SMB3：自由端口奇偶校验错误

SMB3 用于自由端口方式，当接收到的字符发现有奇偶校验错时，将 SM3.0 置 1。如表 D-4 所示，当检测到校验错误时，SM3.0 接通。根据该位来废弃错误消息。

表 D-4　特殊存储器字节 SMB3（SM3.0 – SM3.7）

SM 位	描述（只读）
SM3.0	口 0 或口 1 的奇偶校验错（0=无错，1=有错）
SM3.1 – SM3.7	保留

SMB4：队列溢出

如表 D-5 所示，SMB4 包含中断队列溢出位，中断是否允许标志位及发送空闲位。队列溢出表明要么是中断发生的频率高于 CPU，要么是中断已经被全局中断禁止指令所禁止。

表 D-5　特殊存储器字节 SMB4（SM4.0-SM4.7）

SM 位	描述（只读）
SM4.0[1]	当通信中断队列溢出时，将该位置 1
SM4.1[1]	当输入中断队列溢出时，将该位置 1
SM4.2[1]	当定时中断队列溢出时，将该位置 1
SM4.3	在运行时刻，发现编程问题时，将该位置 1
SM4.4	该位指示全局中断允许位，当允许中断时，将该位置 1
SM4.5	当（口 0）发送空闲时，将该位置 1
SM4.6	当（口 1）发送空闲时，将该位置 1
SM4.7	当发生强置时，将该位置 1

[1] 只有在中断程序里，才使用状态位 SM4.0、SM4.1 和 SM4.2。当队列为空时，将这些状态位复位（置 0），并返回主程序。

SMB5：I/O 状态

如表 D-6 所示，SMB5 包含 I/O 系统里发现的错误状态位。这些位提供了所发现的 I/O 错误的概况。

表 D-6　特殊存储器字节 SMB5（SM5.0 – SM5.7）

SM 位	描述（只读）
SM5.0	当有 I/O 错误时，将该位置 1
SM5.1	当 I/O 总线上连接了过多的数字量 I/O 点时，将该位置 1
SM5.2	当 I/O 总线上连接了过多的模拟量 I/O 点时，将该位置 1
SM5.3	当 I/O 总线上连接了过多的智能 I/O 模块时，将该位置 1
SM5.4 - SM5.7	保留

SMB6：CPU 识别（ID）寄存器

如表 D-7 所示，SMB6 为 CPU 识别（ID）寄存器。SM6.4 到 SM6.7 识别 CPU 的类型，SM6.0 到 SM6.3 保留，以备将来使用。

表 D-7　特殊存储器字节 SMB6

SM 位	描述（只读）
格式	MSB 7　　LSB 0 CPU ID 寄存器 x x x x r r r r
SM6.0 – SM6.3	保留
SM6.4 – SM6.7	xxxx = 0000 = CPU 222 0010 = CPU 224 0110 = CPU 221 1001 = CPU 226/CPU 226XM

SMB7：保留

SMB7 为将来使用而保留。

SMB8 到 SMB21：I/O 模块识别和错误寄存器

SMB8 到 SMB21 是按照字节对形式（相邻两个字节）为扩展模块 0 到 6 而准备的。如表 D－8 所示，每对字节的偶数位字节为模块识别寄存器，奇数位字节为模块错误寄存器。前者标记着模块类型、I/O 类型、输入和输出点数；后者为对相应模块所测得的 I/O 错误提示。

表 D-8 特殊存储器字节 SMB8-SMB21

SM 位	描述 （只读）
格式	偶数字节：模块识别寄存器 MSB LSB 7 0 m t t a i i q q m: 模块存在 0=有模块 1=无模块 tt: 00 非智能 I/O 模块 01 智能模块 10 保留 11 保留 a: I/O 类型 0=开关量 1=模拟量 ii: 00 无输入 01 2 AI 或 8 DI 10 4 AI 或 16 DI 11 8 AI 或 16 DI qq: 00 无输出 01 2 AQ 或 8 DQ 10 4 AQ 或 16 DQ 11 8 AQ 或 32 DQ 奇数字节：模块错误寄存器 MSB LSB 7 0 c 0 0 b r p f t c: 配置错误 0= 无错误 b: 总线错误或校验错误 1=有错误 r: 超范围错误 P: 无用户电源错误 f: 熔断器错误 t: 端子块松错误
SMB8 SMB9	模块 0 识别（ID）寄存器 模块 0 错误寄存器
SMB10 SMB11	模块 1 识别（ID）寄存器 模块 1 错误寄存器
SMB12 SMB13	模块 2 识别（ID）寄存器 模块 2 错误寄存器
SMB14 SMB15	模块 3 识别（ID）寄存器 模块 3 错误寄存器
SMB16 SMB17	模块 4 识别（ID）寄存器 模块 4 错误寄存器
SMB18 SMB19	模块 5 识别（ID）寄存器 模块 5 错误寄存器
SMB20 SMB21	模块 6 识别（ID）寄存器 模块 6 错误寄存器

SMW22 到 SMW26：扫描时间

如表 D-9 所示，SMW22、SMW24 和 SMW26 提供扫描时间信息：以毫秒计的最短扫描时间、最长扫描时间及上次扫描时间。

表 D-9 特殊存储器字 SMW22 到 SMW26

SM 字	描述（只读）
SMW22	上次扫描时间
SMW24	进入 RUN 方式后，所记录的最短扫描时间
SMW26	进入 RUN 方式后，所记录的最长扫描时间

SMB28 和 SMB29：模拟电位器

如表 D-10 所示，SMB28 包含代表模拟调节器 0 位置的数字值。SMB29 包含代表模拟调节器 1 位置的数字值。

表 D-10　特殊存储器字节 SMB28 和 SMB29

SM 字节	描述（只读）
SMB28	存储模拟调节器 0 的输入值。在 STOP/RUN 方式下，每次扫描时更新该值。
SMB29	存储模拟调节器 1 的输入值。在 STOP/RUN 方式下，每次扫描时更新该值。

SMB30 和 SMB130：自由端口控制寄存器

SMB30 控制自由端口 0 的通讯方式，SMB130 控制自由端口 1 的通讯方式。你可以对 SMB30 和 SMB130 进行写和读。如表 D-11 所示，这些字节设置自由端口通讯的操作方式，并提供自由端口或者系统所支持的协议之间的选择。

表 D-11 特殊存储器字节 SMB30

口 0	口 1	描述
SMB30 的格式	SMB130 的格式	MSB LSB 7 0 p p d b b b m m 自由口模式控制字节
SM30.0 和 SM30.1	SM130.0 和 SM130.1	mm: 协议选择 00= 点到点接口协议（PPI/从站模式） 01= 自由口协议 10= PPI/主站模式 11= 保留（缺省是 PPI/从站模式） 注意当选择 mm=10 (PPI 主站)，PLC 将成为网络的一个主站，可以执行 NETR 和 NETW 指令。在 PPI 模式下忽略 2 到 7 位。
SM30.2 和 SM30.4	SM130.2 和 SM130.4	bbb: 自由口波特率 000= 38,400 波特 001= 19,200 波特 010= 9,600 波特 011= 4,800 波特 100= 2,400 波特 101= 1,200 波特 110= 115,200 波特 111= 57,600 波特
SM30.5	SM130.5	d: 每个字符的数据位 0= 8 位/字符 1= 7 位/字符
SM30.6 和 SM30.7	SM130.6 和 SM130.7	pp: 校验选择 00= 不校验 01= 偶校验 10= 不校验 11= 奇校验

SMB31 和 SMW32：永久存储器（EEPROM）写控制

在用户程序的控制下，你可以把 V 存储器中的数据存入永久存储器（EEPROM），亦称非易失存储器。先把被存数据的地址存入 SMW32 中，然后把存入命令存入 SMB31 中。一旦你发出存储命令，则直到 CPU 完成存储操作 SM31.7 被置 0 之前，你不可以改变 V 存储器的值。

在每次扫描周期末尾，CPU 检查是否有向永久存储器区中存数据的命令。如果有，则将该数据存入永久存储器中。

如表 D-12 所示，SMB31 定义了存入永久存储器的数据大小，且提供了初始化存储操作的命令。SMW32 提供了被存数据在 V 存储器中的起始地址。

表 D-12　特殊存储器字节 SMB31 和特殊存储器字 SMW32

SM 字节	描述
格式	SMB31：　MSB　LSB 软件命令　7　0 c 0 0 0 0 0 s s SMW32: V 存储器地址　MSB　LSB 15　0 V 存储器地址
SM31.0 和 SM31.1	ss:　被存数据类型 00 = 字节 01 = 字节 10 = 字 11 = 双字
SM31.7	c:　存入永久存储器（EEPROM） 0 = 无执行存储操作的请求 1 = 用户程序申请向永久存储器存储数据 每次存储操作完成后，S7-200 复位该位。
SMW32	SMW32 中是所存数据的 V 存储器地址，该值是相对于 V0 的偏移量。当执行存储命令时，把该数据存到永久存储器（EEPROM）中相应的位置。

SMB34 和 SMB35：定时中断的时间间隔寄存器

如表 D-13 所示，SMB34 分别定义了定时中断 0 和 1 的时间间隔，可以在 5ms ~ 255ms 之间以 1ms 为增量进行设定。若定时中断事件被中断程序所采用，当 CPU 响应中断时，就会获取该时间间隔值。若要改变该时间间隔，你必须把定时中断事件再分配给同一或另一中断程序，也可通过撤消该事件来终止定时中断事件。

表 D-13 特殊存储器字节 SMB34 和 SMB35

SM 字节	描述
SMB34	定义定时中断 0 的时间间隔（从 1ms~255ms，以 1ms 为增量）
SMB35	定义定时中断 1 的时间间隔（从 1ms~255ms，以 1ms 为增量）

SMB36 到 SMB65：HSC0、HSC1 和 HSC2 寄存器

如表 D-14 所示，SMB36 到 SMB65 用于监视和控制高速计数器 HSC0、HSC1 和 HSC2 的操作。

表 D-14　特殊存储器字节 SMB36 到 SMD62

SM 字节	描述
SM36.0-SM36.4	保留
SM36.5	HSC0 当前计数方向位：1=增计数
SM36.6	HSC0 当前值等于预设值位：1=等于
SM36.7	HSC0 当前值大于预设值位：1=大于
SM37.0	复位操作的有效电平控制位： 0=高电平复位有效，1=低电平复位有效
SM37.1	保留
SM37.2	正交计数器的计数速率选择： 0=4X 速率，1=1X 速率
SM37.3	HSC0 方向控制位：1=增计数
SM37.4	HSC0 更新方向：1=更新方向
SM37.5	HSC0 更新预设值：1=向 HSC0 写新的预设值
SM37.6	HSC0 更新当前值：1=向 HSC0 写新的初始值
SM37.7	HSC0 有效位：1=有效
SMD38	HSC0 新的初始值
SMD42	HSC0 新的预置值
SM46.0-SM46.4	保留
SM46.5	HSC1 当前计数方向：1=增计数
SM46.6	HSC1 当前值等于预设值位：1=等于
SM46.7	HSC1 当前值大于预设值位：1=大于
SM47.0	HSC1 复位有效电平控制位：0=高电平，1=低电平
SM47.1	HSC1 启动有效电平控制位：0=高电平，1=低电平
SM47.2	HSC1 正交计数器速率选择：0=4X 速率，1=1X 速率
SM47.3	HSC1 方向控制位：1=增计数
SM47.4	HSC1 更新方向：1=更新方向
SM47.5	HSC1 更新预设值：1=向 HSC1 写新的预设值
SM47.6	HSC1 更新当前值：1=向 HSC1 写新的初始值
SM47.7	HSC1 有效位：1=有效
SMD48	HSC1 新的初始值
SMD52	HSC1 新的预设值
SM56.0-SM56.4	保留
SM56.5	HSC2 当前计数方向：1=增计数
SM56.6	HSC2 当前值等于预设值位：1=等于
SM56.7	HSC2 当前值大于预设值位：1=大于
SM57.0	HSC2 复位有效电平控制位：0=高电平，1=低电平
SM57.1	HSC2 启动有效电平控制位：0=高电平，1=低电平
SM57.2	HSC2 正交计数器速率选择：0=4X 速率，1=1X 速率
SM57.3	HSC2 方向控制位：1=增计数
SM57.4	HSC2 更新方向：1=更新方向
SM57.5	HSC2 更新预设值：1=向 HSC2 写新的预设值
SM57.6	HSC2 更新当前值：1=向 HSC2 写新的初始值
SM57.7	HSC2 有效位：1=有效
SMD58	HSC2 新的初始值
SMD62	HSC2 新的预设值

SMB66 到 SMB85：PTO/PWM 寄存器

如表 D-15 所示，SMB66 到 SMB85 用于监视和控制脉冲输出（PTO）和脉宽调制（PWM）功能。关于这些位的完整描述见第六章高速输出指令。

表 D-15 特殊存储器字节 SMB66 到 SMB85

SM 字节	描述
SM66.0-SM66.3	保留
SM66.4	PTO 0 包络终止：0=无错，1=由于增量计算错误而终止
SM66.5	PTO 0 包络终止：0=不由用户命令终止；1=由用户命令终止
SM66.6	PTO 0 管道溢出（当使用外部包络时由系统清除，否则由用户程序清除）：0=无溢出，1=有溢出
SM66.7	PTO 0 空闲位：0=PTO 忙，1=PTO 空闲
SM67.0	PTO 0/PWM 0 更新周期：1=写新的周期值
SM67.1	PWM 0 更新脉冲宽度值：1=写新的脉冲宽度
SM67.2	PTO 0 更新脉冲量：1=写新的脉冲量
SM67.3	PTO 0/PWM 0 基准时间单元：0=1μs，1=1ms
SM67.4	同步更新 PWM 0: 0=异步更新，1=同步更新
SM67.5	PTO 0 操作：0=单段操作（周期和脉冲数存在 SM 存储器中）1=多段操作（包络表存在 V 存储器区）
SM67.6	PTO 0/PWM 0 模式选择：0=PTO，1=PWM
SM67.7	PTO 0/PWM 0 有效位：1=有效
SMW68	PTO 0/PWM 0 周期（2~65,535 个时间基准）
SMW70	PWM 0 脉冲宽度值（0~65,535 个时间基准）
SMD72	PTO 0 脉冲计数值（1~2^{32}-1）
SM76.0-SM76.3	保留
SM76.4	PTO1 包络终止；0=无错，1=由于增量计算错误终止
SM76.5	PTO1 包络终止；0=不由用户命令终止，1=通过用户命令终止
SM76.6	PTO1 管道溢出（当使用外部包络时由系统清除，否则由用户程序清除）：0=无溢出，1=有溢出
SM76.7	PTO1 空闲位：0=PTO 工作，1=PTO 空闲
SM77.0	PTO1/PWM1 更新周期值：1=写新周期
SM77.1	PWM1 更新脉冲宽度值：1=写新脉冲宽度
SM77.2	PTO1 更新脉冲计数值：1=写新的脉冲数
SM77.3	PTO1/PWM1 时间基准：0=1μs，1=1ms
SM77.4	同步更新 PWM1: 0=异步更新，1=同步更新
SM77.5	PTO1 操作：0=单段操作（周期和脉冲数存在 SM 存储器），1=多段操作（包络表存在 V 存储器）
SM77.6	PTO1/PWM1 模式选择：0=PTO，1=PWM
SM77.7	PTO1/PWM1 有效位：1=有效
SMW78	PTO1/PWM1 周期值（2 到 65,535 个时间基准）
SMW80	PWM1 脉冲宽度值（0 到 65,535 个时间基准）
SMD82	PTO1 脉冲计数值（1 到 2^{32}-1）

SMB86 到 SMB94，SMB186 到 SMB194：接收信息控制

如表 D-16 所示，SMB86 到 SMB94 和 SMB186 到 SMB194 用于控制和读出接收信息指令的状态。

表 D-16　特殊存储器字节 SMB86 到 SMB94，SMB186 到 SMB194

口 0	口 1	描述
SMB86	SMB186	MSB LSB 7 0 n r e 0 0 t c p　接收信息状态字节 n: 1=通过用户的禁止命令终止接收信息 r: 1=接收信息终止：输入参数错误或无起始或结束条件 e: 1=收到结束字符 t: 1=接收信息终止:超时 c: 1=接收信息终止：超出最大字符数 P: 1=接收信息终止：奇偶校验错误
SMB87	SMB187	MSB LSB 7 0 en sc ec il c/m tmr bk 0　接收信息状态字节 en: 0=禁止接收住处功能 1=允许接收信息功能 每次执行 RCV 指令时检查允许/禁止接收信息位。 sc: 0=忽略 SMB88 或 SMB188 1=使用 SMB88 或 SMB188 的值检测起始信息 ec: 0=忽略 SMB89 或 SMB189 1=使用 SMB89 或 SMB189 的值检测结束信息 il: 0=忽略 SMW90 或 SMW190 1=使用 SMW90 或 SMW190 的值检测空闲状态 c/m: 0=定时器是内部字符定时器 1=定时器是信息定时器 tmr: 0=忽略 SMW92 或 SMW192 1=当 SMW92 或 SMW192 中的定时时间超出时终止接收 bk: 0=忽略中断条件 1=用中断条件作为信息检测的开始
SMB88	SMB188	信息字符的开始
SMB89	SMB189	信息字符的结束
SMW90	SMW190	空闲行时间间隔用毫秒给出。在空闲行时间结束后接收的第一个字符是新信息的开始。
SMW92	SMW192	字符间/信息间定时器超时值（用毫秒表示）。如果超过时间，就停止接收信息。
SMB94	SMB194	接收字符的最大数（1 到 255 字节）。 注意：这个区一定要设为希望的最大缓冲区，即使不使用字符计数信息终止。

SMW98：扩展 I/O 总线错误

如表 D-17 所示，SMW98 给出有关扩展模块总线的错误信息。

表 D-17 特殊存储器字节 SMW98

SM 字节	描述
SMW98	当扩展总线出现校验错误时，该处每次增加 1。当系统得电时或用户程序写入零，可以进行清零。

SMB130：自由口控制寄存器（见 SMB30）

SMB131 到 SMB165：HSC3、HSC4 和 HSC5 寄存器

如表 D-18 所示，SMB131 到 SMB165 用来监视和控制高速计数器 HSC3、HSC4 和 HSC5 的操作。

表 D-18　特殊存储器字节 SMB130 到 SMB165

SM 字节	描述
SMB131-SMB135	保留
SM136.0-SM136.4	保留
SM136.5	HSC3 当前计数方向状态位：1=增计数
SM136.6	HSC3 当前值等于预设值状态位：1=等于
SM136.7	HSC3 当前值大于预设值状态位：1=大于
SM137.0-SM137.2	保留
SM137.3	HSC3 方向控制位：1=增计数
SM137.4	HSC3 更新方向：1=更新方向
SM137.5	HSC3 更新设定值：1=向 HSC3 写入新预设值
SM137.6	HSC3 更新当前值：1=向 HSC3 写新的初始值
SM137.7	HSC3 有效位：1=有效
SMD138	HSC3 新初始值
SMD142	HSC3 新预设值
SM146.0-SM146.4	保留
SM146.5	HSC4 当前计数方向状态位：1=增计数
SM146.6	HSC4 当前值等于预设值状态位：1=等于
SM146.7	HSC4 当前值大于预设值状态位：1=大于
SM147.0	复位的有效控制位： 0=高电平有效，1=低电平有效
SM147.1	保留
SM147.2	正交计数器的计数速率选择： 0=4x 计数速率，1=1x 计数速率
SM147.3	HSC4 方向控制位：1=增计数
SM147.4	HSC4 更新方向：1=更新方向
SM147.5	HSC4 更新预设值：1=向 HSC4 写新的预设值
SM147.6	HSC4 更新当前值：1=向 HSC4 写新的初始值
SM147.7	HSC4 有效位：1=有效
SMD148	HSC4 新初始值
SMD152	HSC4 预设值
SM156.0-SM156.4	保留
SM156.5	HSC5 当前计数方向状态位：1=增计数
SM156.6	HSC5 当前值等于预设值状态位：1=等于
SM156.7	HSC5 当前值大于预设值状态位：1=大于
SM157.0-SM157.2	保留
SM157.3	HSC5 方向控制位：1=增计数
SM157.4	HSC5 更新方向：1=更新方向
SM157.5	HSC5 更新预设值：1=向 HSC5 写新的预设值
SM157.6	HSC5 更新当前值：1=向 HSC5 写新的初始值
SM157.7	HSC5 有效位：1=有效
SMD158	HSC5 新初始值
SMD162	HSC5 预设值

SMB166 到 SMB185：PTO0，PTO1 包络定义表

如表 D-19 所示，SMB166 到 SMB194 用来显示包络步的数量和包络表的地址和 V 存储器区中表的地址。

表 D-19　特殊存储器字节 SMB166 到 SMB185

SM 字节	描述
SMB166	PTO0 的包络步当前计数值
SMB167	保留
SMD168	PTO0 的包络表 V 存储器地址（从 V0 开始的偏移量）
SMB170-SMB175	保留
SMB176	PTO1 的包络步当前计数值
SMB177	保留
SMD178	PTO1 的包络表 V 存储器地址（从 V0 开始的偏移量）
SMB180-SMB185	保留

SMB186 到 SMB194：接收信息控制（见 SMB86-SMB94）

参考表 D-16。

SMB200 到 SMB549: 智能模块状态

如表 D-20 所示，SMB200 到 SMB549 预留存储智能扩展模块的信息。如 EM277PROFIBUS-DP 模块。参见附录 A 可得到如何使用 SMB200 到 SMB549 的信息以及您的模块的规范。

注意

为智能模块定义 SM 区域的方式在版本 2.2 和其后版本作了改变。

如果您使用版本 2.2 之前的 CPU，您要将所有的智能模块放置在所有非智能模块之前紧邻 CPU 的位置，以确保其兼容性。

表 D-20　特殊存储字节 SMB200 – SMB549

特殊存储字节 SMB200 – SMB549							
智能模块 0	智能模块 1	智能模块 2	智能模块 3	智能模块 4	智能模块 5	智能模块 6	描述
SMB200-SMB215	SMB250-SMB265	SMB300-SMB315	SMB350-SMB365	SMB400-SMB415	SMB450-SMB465	SMB500-SMB515	模块名称（16 个 ASCII 字符）
SMB216-SMB219	SMB266-SMB269	SMB316-SMB319	SMB366-SMB369	SMB416-SMB419	SMB466-SMB469	SMB516-SMB519	SW 修订号（4 个 ASCII 字符 xxxx）
SMW220	SMW270	SMW320	SMW370	SMW420	SMW470	SMW520	错误代码
SMB222-SMB249	SMB272-SMB299	SMB322-SMB349	SMB372-SMB399	SMB422-SMB449	SMB472-SMB499	SMB522-SMB549	与特定模块类型相关的信息

S7-200 定货号

E

CPU	定货号
CPU 221 DC/DC/DC 6 输入/4 输出	6ES7 211-0AA22-0XB0
CPU 221 AC/DC/继电器 6 输入/4 继电器输出	6ES7 211-0BA22-0XB0
CPU 222 DC/DC/DC 8 输入 /6 输出	6ES7 212-1AB22-0XB0
CPU 222 AC/DC/继电器 8 输入/6 继电器输出	6ES7 212-1BB22-0XB0
CPU 224 DC/DC/DC 14 输入/10 输出	6ES7 214-1AD22-0XB0
CPU 224 AC/DC/继电器 14 输入/10 继电器输出	6ES7 214-1BD22-0XB0
CPU 226 DC/DC/DC 24 输入/16 输出	6ES7 216-2AD22-0XB0
CPU 226 AC/DC/继电器 24 输入/16 继电器输出	6ES7 216-2BD22-0XB0
CPU 226XM DC/DC/DC 24 输入/16 输出	6ES7 216-2AF22-0XB0
CPU 226XM AC/DC/继电器 24 输入/16 继电器输出	6ES7 216-2BF22-0XB0

扩展模块	定货号
EM 221 24 VDC 数字 8 输入	6ES7 221-1BF22-0XA0
EM 221 24 VDC 数字 16 输入	6ES7 221-1BH22-0XA0
EM 221 120/230 VAC 数字 8 输入	6ES7 221-1EF22-0XA0
EM 222 24 VDC 数字 8 输出	6ES7 222-1BF22-0XA0
EM 222 24 VDC 数字 4 输出（5A）	6ES7 222-1BD22-0XA0
EM 222 继电器 4 输出（10A）	6ES7 222-1HD22-0XA0
EM 222 继电器 8 输出	6ES7 222-1HF22-0XA0
EM 222 120/230 VAC 数字 8 输出	6ES7 222-1EF22-0XA0
EM 223 24 VDC 数字组合 4 输入 /4 输出	6ES7 223-1BF22-0XA0
EM 223 24 VDC 数字组合 4 输入 /4 继电器输出	6ES7 223-1HF22-0XA0
EM 223 24 VDC 数字组合 8 输入 /8 输出	6ES7 223-1BH22-0XA0
EM 223 24 VDC 数字组合 8 输入 /8 继电器输出	6ES7 223-1PH22-0XA0
EM 223 24 VDC 数字组合 16 输入 /16 输出	6ES7 223-1BL22-0XA0
EM 223 24 VDC 数字组合 16 输入 /16 继电器输出	6ES7 223-1PL22-0XA0
EM 231 24 VDC 模拟量输入，4 输入	6ES7 231-0HC22-0XA0
EM 231 24 VDC 模拟量输入热电阻，2 输入	6ES7 231-7PB22-0XA0
EM 231 24 VDC 模拟量输入热电偶，4 输入	6ES7 231-7PD22-0XA0
EM 231 24 VDC 模拟量输出，2 输出	6ES7 232-0HB22-0XA0
EM 235 24 VDC 模拟量组合，4 输入/1 输出	6ES7 235-0KD22-0XA0
EM 241 Modem 模块	6ES7 241-1A22-0XA0
EM 253 位控模块	6ES7 253-1A22-0XA0
EM 277 PROFIBUS-DP	6ES7 277-0AA22-0XA0
CP 243-2 AS 接口通讯处理器	6GK7 243-2AX01-0XA0

卡和电缆	定货号
MC 291, 32K X8 EEPROM 存储器卡	6ES7 291-8GE20-0XA0
CC 292, CPU 22x 带电池的时钟/日历卡	6ES7 297-1AA20-0XA0
BC 293, CPU 22x 电池卡	6ES7 291-8BA20-0XA0
电缆，I/O 扩展，0.8 米，CPU 22x/EM	6ES7 290-6AA20-0XA0
电缆，PC/PPI，隔离，90 度接头，RTS 开关	6ES7 901-3BF21-0XA0
编程软件	**定货号**
STEP 7-Micro/WIN 32 (V3.2) 单用户授权（CD-ROM）	6ES7 810-2BC02-0YX0
STEP 7-Micro/WIN 32 (V3.2) 升级授权（CD-ROM）	6ES7 810-2BC02-0YX3
TP-Designer 组态 TP 070，版本 1.0（CD-ROM）	6ES7 850-2BC00-0YX0
STEP 7-Micro/WIN 32 指令库 CD-	6ES7 830-2BC00-0YX0
通讯卡	**定货号**
MPI 卡：短 AT ISA	6ES7 793-2AA01-0AA0
CP 5411: 短 AT ISA	6GK1 541-1AA00
CP 5511: PCMCIA, Type II	6GK1 551-1AA00
CP 5611: PCI 卡（3 .0 版或更高版）	6GK1 561-1AA00
手册	**定货号**
TD 200 操作员界面用户手册	6ES7 272-0AA20-8BA0
TP070 触摸屏用户手册（英语）	6AV6 591-1DC01-0AB0
S7-200 点到点接口通讯手册（英语/德语）	6ES7 298-8GA00-8XH0
CP 243-2 SIMATIC NET AS 接口主站手册（英语）	6GK7 243-2AX00-8BA0
S7-200 可编程控制器系统手册（德语）	6ES7 298-8FA22-8AH0
S7-200 可编程控制器系统手册（英语）	6ES7 298-8FA22-8BH0
S7-200 可编程控制器系统手册（法语）	6ES7 298-8FA22-8CH0
S7-200 可编程控制器系统手册（西班牙语）	6ES7 298-8FA22-8DH0
S7-200 可编程控制器系统手册（意大利语）	6ES7 298-8FA22-8EH0
电缆，网络连接器和中继器	**定货号**
MPI 电缆	6ES7 901-0BF00-0AA0
PROFIBUS 网络电缆	6XV1 830-0AH10
带编程接口的网络总线连接器，垂直电缆出口	6ES7 972-0BB11-0XA0
网络总线连接器，（不带编程接口），垂直电缆出口	6ES7 972-0BA11-0XA0
RS 485 总线连接器，35° 电缆出口（不带编程接口）	6ES7 972-0BA40-0XA0
RS 485 总线连接器，35° 电缆出口（带编程接口）	6ES7 972-0BB40-0XA0
CPU 22x/EM 连接器块，7 端子，可移动式	6ES7 292-1AD20-0AA0
CPU 22x/EM 连接器块，12 端子，可移动式	6ES7 292-1AE20-0AA0
CPU 22x/EM 连接器块，14 端子，可移动式	6ES7 292-1AF20-0AA0
CPU 22x/EM 连接器块，18 端子，可移动式	6ES7 292-1AG20-0AA0
RS-485 IP 20 中继器，隔离型	6ES7 972-0AA00-0XA0
操作员界面	**定货号**
TD 200 操作员界面	6ES7 272-0AA20-0YA0
OP3 操作员界面	6AV3 503-1DB10T
OP7 操作员界面	6AV3 607-1JC20-0AX1
OP17 操作员界面	6AV3 617-1JC20-0AX1
TP 070 触摸屏	6AV6 545-0AA15-2AX0
TP 170A 触摸屏	6AV6 545-0BA15-2AX0
附件	**定货号**
DIN 导轨固定端子	6ES5 728-8MAII
12-位扇出连接器（CPU 221, CPU 222）10-包	6ES7 290-2AA00-0XA0
备用门工具包，包括以下 4 个：7、12、14、18、2x12、2x14 端子块盖，CPU 盖，EM 盖	6ES7 291-3AX20-0XA0
8 位模拟器	6ES7 274 1XF00-0XA0
14 位模似器	6ES7 274 1XH00-0XA0
24 位模拟器	6ES7 274 1XK00-0XA0

STL 指令的执行时间

F

指令执行时间对于对时间要求比较苛刻的应用来说非常重要。指令的执行时间如表 F-3 所示。

提示

当您使用表 F-3 的执行时间时、要考虑使能位，间接寻址以及使用某些存储区域对指令的执行时间的影响。这些因素可以直接地影响所列的执行时间。

使能位的影响

表 F-3 所列的执行时间是当使能位接通（顶端堆栈=1 或接通）时该指令逻辑或功能所需的时间。

当使能位未接通时，指令的执行时间为 3μs。

间接寻址的影响

表 F-3 所列为使用操作数或常数的直接寻址执行该指令的逻辑或功能所需要的时间。

当使用间接寻址的操作数时，指令中的每一个间接寻址的操作数会使指令执行时间增加 22μs。

访问某些存储区域的影响

访问某些存储区域，如 AI、AQ、L 和累加器需要额外的执行时间。

表 F-1 所示为在操作数中使用这些存储区需在指令执行时间上增加的时间。

表 F-1　访问存储区的时间增加值

存储区	执行时间增加
模拟输入（AI） 未使能模拟虑波： 使能模拟滤波：	 149 μs 0 μs
模拟输出（AQ）	73 μs
局域存储器（L）	5.4 μs
累加器（AC）	4.4 μs

使用某些 CPU 226XM 指令

执行 CPU226XM 的某一类指令时需要额外的时间。

表 F-2 是执行所列指令时需在基本执行时间上增加的值。

表 F-2 CPU 226XM 指令时间增加值

指令	执行时间增加值
ATCH	1.0 µs
CALL	4.3 µs
CSCRE	3.1 µs
FOR (时基的增加值) （循环系数的增加值）	3.1 µs 3.1 µs
INT	1.7 µs
JMP	3.1 µs
RET	2.8 µs

表 F-3　指令执行时间

指令			μs
=	使用：	I	0.37
		SM, T, C, V, S, Q, M	1.8
		L	19.2
+D			55
-D			55
*D			92
/D			376
+I			46
-I			47
*I			71
/I			115
=I	使用：	本机输出	29
		扩展输出	39
+R			110 163 最大
-R			113 166 最大
*R			100 130 最大
/R			300 360 最大
A	使用：	I	0.37
		SM, T, C, V, S, Q, M	1.1
		L	10.8
AB <=,=,>=,>,<,<>			35
AD <=,=,>=,>,<,<>			53
AENO			0.6
AI	使用：	本机输入	27
		扩展输入	35
ALD			0.37
AN	使用：	I	0.37
		SM, T, C, V, S, Q, M	1.1
		L	10.8
ANDB			37
ANDD			55
ANDW			48
ANI	使用：	本机输入	27
		扩展输入　　L	35
AR <=, =, >=, >, <, <>			54
AS=, <>	总时间 = 基本时间 + (LM*N)		
	基本时间		51
	长度系数 (LM)		9.2
	N 比较的字符数		
ATCH			20
ATH	总时间=基本时间＋（长度*LM）		
	基本时间（固定长度）		41
	基本时间（变长度）		55
	长度系数（LM）		20

指令			μs
ATT			70
AW < =, =, >=, >, <, <>			45
BCDI			66
BIR	使用：	本机输入	45
		扩展输入	53
BIW	使用：	本机输出	46
		扩展输出	56
BMB	总时间=基本时间＋（长度*LM）		
	基本时间（固定长度）		21
	基本时间（变长度）		51
	长度系数（LM）		11
BMD	总时间=基本时间＋（长度*LM）		
	基本时间（固定长度）		21
	基本时间（变长度）		51
	长度系数（LM）		20
BMW	总时间=基本时间＋（长度*LM）		
	基本时间（固定长度）		21
	基本时间（变长度）		51
	长度系数（LM）		16
BTI			27
CALL	无参数		15
	有参数		
	总时间=基本时间+Σ（操作数处理时间）		
	基本时间		32
	操作数处理时间		
	位（输入，输出）		23, 21
	字节（输入，输出）		21, 14
	字（输入，输出）		24, 18
	双字（输入，输出）		27, 20
注意：	对输出操作数的处理在子程序返回时		
CFND	最大时间=基本时间		79
	N1*((LM1*N2) + LM2)		
	基本时间		79
	长度系数 1（LM1）		9.2
	长度系数 2（LM2）		4.4
	N1 是源字符串的长度		
	N2 字符串的长度		
COS			1525 1800 最大
CRET			13
CRETI			23
CSCRE			0.9
CTD	计数输入转换		48
	否则		36
CTU	计数输入转换		53
	否则		35
CTUD	计数输入转换		64
	否则		45
DECB			30
DECD			42
DECO			36

指令	μs
DECW	37
DISI	18
DIV	119
DTA	540
DTI	36
DTCH	18
DTR	60 70 最大
DTS	540
ED	15
ENCO	39 43 最大
END	0.9
ENI	53
EU	15
EXP	1170 1375 最大
FIFO 总时间=基本时间＋（长度*LM） 基本时间 长度系数（LM）	 70 14
FILL 总时间=基本时间+（长度*LM） 基本时间（固定长度） 基本时间（变长度） 长度系数（LM）	 29 50 7
FND<, =, >, <> 总时间=基本时间+（长度*LM） 基本时间 长度系数（LM）	 85 12
FOR 总时间=基本时间+(Number of loops*LM) 基本时间 循环系数 (LM)	 64 50
GPA	31
HDEF	35
HSC	37
HTA 总时间=基本时间+（长度*LM） 基本时间（固定长度） 基本时间（变长度） 长度系数（LM）	 38 48 11
IBCD	114
INCB	29
INCD	42
INCW	37
INT 1 个中断的典型值	47
INVB	31
INVD	42
INVW	38
ITA	260
ITB	27
ITD	36

指令	μs
ITS	260
JMP	0.9
LBL	0.37
LD 使用： I, SM0.0 SM, T, C, V, S, Q, M L	0.37 1.1 10.9
LDB<=, =, >=, >, <, <>	35
LDD<=, =, >=, >, <, <>	52
LDI 使用： 本机输入 扩展输入	26 34
LDN 使用： I SM, T, C, V, S, Q, M L	0.37 1.1 10.9
LDNI 使用： 本机输入 扩展输入	26 34
LDR<=, =, >=, >, <, <>	55
LDS	0.37
LDS=, <> 总时间=基本时间+(LM*N) 基本时间 长度系数（LM） N 比较的字符数	 51 9.2
LDW<=, =, >=, >, <, <>	42
LIFO	70
LN	1130 1275 最大
LPP	0.37
LPS	0.37
LRD	0.37
LSCR	12
MEND	0.5
MOVB	29
MOVD	38
MOVR	38
MOVW	34
MUL	70
NEXT	0
NETR	179
NETW 总时间=基本时间+(长度*LM) 基本时间 长度系数（LM）	 175 8
NOP	0.37
NOT	0.37
O 使用： I SM, T, C, V, S, Q, M L	0.37 1.1 10.8
OB< =, =, >=, >, <, <>	35
OD< =, =, >=, >, <, <>	53

指令			μs
OI	使用：	本机输入	27
		扩展输入	35
OLD			0.37
ON	使用：	I	0.37
		SM, T, C, V, S, Q, M	1.1
		L	10.8
ONI	使用：	本机输入	27
		扩展输入	35
OR<=, =, >=, >, <, <>			55
ORB			37
ORD			55
ORW			48
OS=, <>		总时间＝基本时间+(LM*N)	
		基本时间	51
		长度系数（LM）	9.2
		N 比较的字符数	
OW< =, =, >=, >, <, <>			45
PID	基本时间		750
	重新计算比例积分微分的		1000
	增加时间		
PLS：	使用：	PWM	57
		PTO 单数	67
		PTO 复数	92
R	长度=1 定义为常数		
	使用操作数=C, T		17, 24
	使用其它操作数		5
	否则，总时间=基本时间+(长度*LM)		
	操作数 C，T 的基本时间		19, 19
	其它操作数的基本时间		28
	操作数 C，T 的 LM		8.6, 16.5
	其它操作数 LM		0.9
	如果长度存为变量加入到基本时间中		29
RCV			80
RET			13
RETI			23
RI	总时间=基本时间＋（长度*LM）		
	基本时间		18
	LM 使用本机输出		22
	LM 使用扩展输出		32
	如果长度存为变量加入到基本时间中		30
RLB	总时间=基本时间+（长度*LM）		
	基本时间		42
	长度系数（LM）		0.6
RLD	总时间=基本时间+（长度*LM）		
	基本时间		52
	长度系数（LM）		2.5
RLW	总时间=基本时间+（长度*LM）		
	基本时间		49
	长度系数（LM）		1.7

指令			μs
ROUND			108
			183 最大
RRB	总时间=基本时间+（长度*LM）		
	基本时间		42
	长度系数（LM）		0.6
RRD	总时间=基本时间+（长度*LM）		
	基本时间		52
	长度系数（LM）		2.5
RRW	总时间=基本时间+（长度*LM）		
	基本时间		49
	长度系数（LM）		1.7
RTA	总时间=基本时间+（LM*N）		
	基本时间（结果中的第一个数）		1000
	长度系数（LM）		240
	N 结果中的额外的数字的数量		
RTS	总时间=基本时间+（LM*N）		
	基本时间（结果中的第一个数）		1000
	长度系数（LM）		240
	N 结果中的额外的数字的数量		
S	长度=1 指定为一个常数		5
	否则：		
	总时间=基本时间＋（长度*LM）		
	基本时间		27
	长度系数（LM）		0.9
	如果长度存为变量则加入到基本时间中		29
SBR			0
SCAT	总时间=基本时间+（LM*N）		
	基本时间		55
	长度系数（LM）		8.8
	N 是附加的字符的数量		
SCPY	总时间=基本时间+（LM*N）		
	基本时间		43
	长度系数（LM）		8.8
	N 复制的字符的量数		
SCRE			0.37
SCRT			17
SEG			30
SFND	最大时间=基本时间+		
	（N1-N2）*((LM1*N2)+LM2)		
	基本时间		79
	长度系数 1（LM1）		11.5
	长度系数 2（LM2）		17.8
	N1 源字符串的长度		
	N2 搜索的字符串长度		
SHRB	总时间=基本时间+（长度*LM1）+		
	长度/8）*LM2)		
	基本时间（固定长度）		76
	基本时间（变长度）		84
	长度系数 1（LM1）		1.6
	长度系数 2（LM2）		4

指令		µs
SI	总时间=基本时间＋（长度*LM）	
	基本时间	18
	LM 使用本机输出	22
	LM 使用扩展输出	32
	如果长度存为变量，加入到基本时间中	30
SIN		1525 1800 最大
SLB	总时间=基本时间＋（长度*LM）	
	基本时间	43
	长度系数（LM）	0.7
SLD	总时间=基本时间＋（长度*LM）	
	基本时间	53
	长度系数（LM）	2.6
SLEN		46
SLW	总时间=基本时间＋（长度*LM）	
	基本时间	51
	长度系数（LM）	1.3
SPA		243
SQRT		725 830 最大
SRB	总时间=基本时间＋（长度*LM）	
	基本时间	43
	长度系数（LM）	0.7
SRD	总时间=基本时间＋（长度*LM）	
	基本时间	53
	长度系数（LM）	2.6
SRW	总时间=基本时间＋（长度*LM）	
	基本时间	51
	长度系数（LM）	1.3
SSCPY	总时间=基本时间＋（长度*LM）	
	基本时间	82
	长度系数（LM）	8.8
	N 是复制的字符数	

指令		µs
STD	总时间=基本时间+（LM*N）	
	基本时间（第一个源字符的）	84
	长度系数（LM）	59
	N 额外的源字符的数量	
STI	总时间=基本时间+（LM*N）	
	基本时间（第一个源字符的）	84
	长度系数（LM）	59
	N 额外的源字符的数量	
STOP		16
STR	总时间=基本时间+（LM*N）	
	基本时间（第一个源字符的）	100
	长度系数（LM）	120
	N 额外的源字符的数量	
SWAP		32
TAN		1825 2100 最大
TODR		2400
TODW		1600
TOF		64
TON		64
TONR		56
TRUNC		103 178 最大
WDR		16
XMT		78
XORB		37
XORD		55
XORW		48

S7-200 快速参考信息

G

表 G-1 特殊存储器位

特殊存储器位			
SM0.0	该位始终为 1	SM1.0	操作结果=0
SM0.1	首次扫描时为 1	SM1.1	结果溢出或非法数值
SM0.2	保持数据丢失时为 1	SM1.2	结果为负数
SM0.3	开机上电进入 RUN 时为 1 一个扫描周期	SM1.3	被 0 除
SM0.4	时钟脉冲：30s 闭合/30s 断开	SM1.4	超出表范围
SM0.5	时钟脉冲：0.5s 闭合/0.5s 断开	SM1.5	空表
SM0.6	时钟脉冲：闭合 1 个扫描周期/断开 1 个扫描周期	SM1.6	BCD 到二进制转换出错
SM0.7	开关放置在 RUN 位置时为 1	SM1.7	ASCII 到十六进制转换出错

表 G-2 中断事件描述

中断号	中断描述	优先级分组	按组排列的优先级
8	通讯口 0：按收字符	通讯（最高）	0
9	通讯口 0：发送完成		0
23	通讯口 0：接收信息完成		0
24	通讯口 1：接收信息完成		1
25	通讯口 1：接收字符		1
26	通讯口 1：发送完成		1
19	PTO 0 完成中断	开关量（中等）	0
20	PTO 1 完成中断		1
0	I0.0 的上升沿		2
2	I0.1 的上升沿		3
4	I0.2 的上升沿		4
6	I0.3 的上升沿		5
1	I0.0 的下降沿		6
3	I0.1 的下降沿		7
5	I0.2 的下降沿		8
7	I0.3 的下降沿		9
12	HSC0 CV=PV(当前值=预设值)		10
27	HSC0 方向改变		11
28	HSC0 外部复位		12
13	HSC1 CV=PV(当前值=预设值)		13
14	HSC1 方向改变		14
15	HSC1 外部复位		15
16	HSC2 CV=PV(当前值=预设值）		16
17	HSC2 方向改变		17
18	HSC2 外部复位		18
32	HSC3 CV=PV(当前值=预设值)		19
29	HSC4 CV=PV(当前值=预设值)		20
30	HSC4 方向改变		21
31	HSC4 外部复位		22
33	HSC5 CV=PV(当前值=预设值)		23
10	定时中断 0	定时（最低）	0
11	定时中断 1		1
21	定时器 T32 CT=PT 中断		2
22	定时器 T96 CT=PT 中断		3

表 G-3 S7-200CPU 存储器范围和特性汇总

描述	范围				
	CPU 221	CPU 222	CPU 224	CPU 226	CPU 226XM
用户程序区	2K 字	2K 字	4K 字	4K 字	8K 字
用户数据区	1K 字	1K 字	2.5K 字	2.5K 字	5K 字
输入映像寄存器	I0.0-I15.7	I0.0-I15.7	10.0-I15.7	10.0-I15.7	10.0-I15.7
输出映像寄存器	Q0.0-Q15.7	Q0.0-Q15.7	Q0.0-Q15.7	Q0.0-Q15.7	Q0.0-Q15.7
模拟输入 （只读）	-	AIW0-AIW30	AIW0-AIW62	AIW0-AIW62	AIW0-AIW62
模拟输出 （只写）	-	AQW0-AQW30	AIW0-AQW62	AIW0-AQW62	AIW0-AQW62
变量存储器 （V）[1]	VB0-VB2047	VB0-VB2047	VB0-VB5119	VB0-VB5119	VB0-VB10239
局部存储器 （L）[2]	LB0-LB63	LB0-LB63	LB0-LB63	LB0-LB63	LB0-LB63
位存储器（SM）	M0.0-M31.7	M0.0-M31.7	M0.0-M31.7	M0.0-M31.7	M0.0-M31.7
特殊存储器（SM） 只读	SM0.0-SM179.7 SM0.0-SM29.7	SM0.0-SM299.7 SM0.0-SM29.7	SM0.0-SM549.7 SM0.0-SM29.7	SM0.0-SM549.7 SM0.0-SM29.7	SM0.0-SM549.7 SM0.0-SM29.7
定时器 保持接通延时 1ms 保持接通延时 10ms 保持接通延时 100ms 接通/断开延时 1ms 接通/断开延时 10ms 接通/断开延时 100ms	256 (T0-T255) T0, T64 T1-T4. T65-T68 T5-T31 T69-T95 T32, T96 T33-T36 T97-T100, T101-T255	256 (T0-T255) T0, T64 T1-T4, T65-T68 T5-T31 T69-T95 T32, T96 T97-T100 T37-T63. T101-T255	256 (T0-T255) T0, T64 T1-T4. T65-T68 T5-T31 T69-T95 T32, T96 T37-T63, T97-T100 T101-T255	256 (T0-T255) T0, T64 T1-T4. T65-T68 T5-T31 T69-T95 T32, T96 T37-T63, T97-T100 T101-T255	256 (T0-T255) T0, T64 T1-T4. T65-T68 T5-T31 T69-T95 T32, T96 T37-T63, T97-T100 T101-T255
计数器	C0-C255	C0-C255	C0-C255	C0-C255	C0-C255
高速计数器	HC0, HC3 HC4, HC5	HC0, HC3 HC4, HC5	HC0-HC5	HC0-HC5	HC0-HC5
顺控继电器（S）	S0.0-S31.7	S0.0-S31.7	S0.0-S31.7	S0.0-S31.7	S0.0-S31.7
累加器	AC0-AC3	AC0-AC3	AC0-AC3	AC0-AC3	AC0-AC3
跳转/标号	0-255	0-255	0-255	0-255	0-255
调用/子程序	0-63	0-63	0-63	0-63	0-127
中断程序	0-127	0-127	0-127	0-127	0-127
PID 回路	0-7	0-7	0-7	0-7	0-7
通讯口	0	0	0	0/1	0/1

表 G-4　高速计数器 HSC0，HSC3，HSC4，HSC5

模式	HSC0			HSC3	HSC4			HSC5
	I0.0	I0.1	I0.2	I0.1	I0.3	I0.4	I0.5	I0.4
0	计数	-	-	计数	计数	-	-	计数
1	计数	-	复位	-	计数	-	复位	-
2	-	-	-	-	-	-	-	-
3	计数	方向	-	-	计数	方向	-	-
4	计数	方向	复位	-	计数	方向	复位	-
5	-	-	-	-	-	-	-	-
6	增计数	减计数	-	-	增计数	减计数	-	-
7	增计数	减计数	复位	-	增计数	减计数	复位	-
8	-	-	-	-	-	-	-	-
9	A 相	B 相	-	-	A 相	B 相	-	-
10	A 相	B 相	复位	-	A 相	B 相	复位	-
11	-	-	-	-	-	-	-	-

表 G-5 高速计数器 HSC1 和 HSC2

模式	HSC1				HSC2			
	I0.6	I0.7	I1.0	I1.1	I1.2	I1.3	I1.4	I1.5
0	计数	-	-	-	计数	-	-	-
1	计数	-	复位	-	计数	-	复位	-
2	计数	-	复位	启动	计数	-	复位	启动
3	计数	方向	-	-	计数	方向	-	-
4	计数	方向	复位	-	计数	方向	复位	-
5	计数	方向	复位	启动	计数	方向	复位	启动
6	增计数	减计数	-	-	增计数	减计数	-	-
7	增计数	减计数	复位	-	增计数	减计数	复位	-
8	增计数	减计数	复位	启动	增计数	减计数	复位	启动
9	A 相	B 相	-	-	A 相	B 相	-	-
10	A 相	B 相	复位	-	A 相	B 相	复位	-
11	A 相	B 相	复位	启动	A 相	B 相	复位	启动

布尔指令		
LD	N	装载
LDI	N	立即装载
LDN	N	取反后装载
LDNI	N	取反后立即装载
A	N	与
AI	N	立即与
AN	N	取反后与
ANI	N	取反后立即与
O	N	或
OI	N	立即或
ON	N	取反后或
ONI	N	取反后立即或
LDBx	N1, N2	装载字节比较的结果 N1(x:<,<=,=,>=,>,<>) N2
ABx	N1, N2	与字节比较的结果 N1(x:<,<=,=,>=,>,<>) N2
OBx	N1, N2	或字节比较的结果 N1(x:<,<=,=,>=,>,<>)N2
LDWx	N1, N2	装载字比较的结果 N1(x:<,<=,=,>=,>,<>)N2
AWx	N1, N2	与字比较的结果 N1(x:<,<=,=,>=,>,<>)N2
OWx	N1, N2	或字比较的结果 N1(x:<,<=,=,>=,>,<>)N2
LDDx	N1, N2	装载双字比较的结果 N1(x:<,<=,=,>=,>,<>)N2
ADx	N1, N2	与双字比较的结果 N1(x:<,<=,=,>=,>,<>)N2
ODx	N1, N2	或双字比较的结果 N1(x:<,<=,=,>=,>,<>)N2
LDRx	N1, N2	装载实数比较的结果 N1(x:<,<=,=,>=,>,<>)N2
ARx	N1, N2	与实数比较的结果 N1(x:<,<=,=,>=,>,<>)N2
ORx	N1, N2	或实数比较的结果 N1(x:<,<=,=,>=,>,<>)N2
NOT		堆栈取反
EU		检测上升沿
ED		检测下降沿
=	N	赋值
=1	N	立即赋值
S	S_BIT,N	置位一个区域
R	S_BIT,N	复位一个区域
SI	S_BIT,N	立即置位一个区域
RI	S_BIT,N	立即置位一个区域
LDSx	IN1,IN2	装载字符串比较结果
ASx	IN1,IN2	IN1 (x: =, <>) IN2
OSXI	IN1,IN2	与字符串比较结果 IN1 (x: =, <>) IN2 或字符串比较结果 IN1 (x: =, <>) IN2
ALD		与装载
OLD		或装载
LPS		逻辑压栈（堆栈控制）
LRD		逻辑读（堆栈控制）
LPP		逻辑弹出（堆栈控制）
LDS	N	装载堆栈（堆栈控制）
AENO		与 ENO

数学、增减指令		
+I	IN1, OUT	整数、双整数或实数加法
+D	IN1, OUT	IN1+OUT=OUT
+R	IN1, OUT	
-I	IN1, OUT	整数、双整数或实数减法
-D	IN1, OUT	OUT-IN1=OUT
-R	IN1, OUT	
MUL	IN1, OUT	整数或实数乘法
*R	IN1, OUT	IN1＊OUT＝OUT
*D, *I	IN1, OUT	整数或双整数乘法
DIV	IN1, OUT	整数或实数除法
/R	IN1, OUT	IN1/OUT＝OUT
/D,/I	IN1, OUT	整数或双整数除法
SQRT	IN1, OUT	平方根
LN	IN1, OUT	自然对数
EXP	IN1, OUT	自然指数
SIN	IN1, OUT	正弦
COS	IN1, OUT	余弦
TAN	IN1, OUT	正切
INCB	OUT	字节、字和双字增 1
INCW	OUT	
INCD	OUT	
DECB	OUT	字节、字和双字减 1
DECW	OUT	
DECD	OUT	
PID	Table, Loop	PID 回路
定时器和计数器指令		
TON	Txxx, PT	接通延时定时器
TOF	Txxx, PT	关断延时定时器
TONR	Txxx, PT	带记忆的接通延时定时器
CTU	Cxxx, PV	增计数
CTD	Cxxx, PV	减计数
CTUD	Cxxx, PV	增/减计数
实时时钟指令		
TODR	T	读实时时钟
TODW	T	写实时时钟
程序控制指令		
END		程序的条件结束
STOP		切换到 STOP 模式
WDR		看门狗复位（300ms）
JMP	N	跳到定义的标号
IBL	N	定义一个跳转的标号
CALL	N[N1,…]	调用子程序[N1，…可以有 16 个可选参数]
CRET		从 SBR 条件返回
FOR	INDX, INIT FINAL	For/Next 循环
NEXT		
LSCR	N	顺控继电器段的启动、转换，条件结束和结束
SCRT	N	
CSCRE		
SCRE		

传送、移位、循环和填充指令		
MOVB MOVW MOVD MOVR BIR BIW	OUT OUT OUT OUT IN, OUT IN, OUT	字节、字、双字和实数传送
BMB BMWI BMD	IN, OUT, N IN, OUT, N IN, OUT, N	字节、字和双字块传送
SWAP	IN	交换字节
SHRB DATA, S_BIT,	 N	寄存器移位
SRB SRW SRD	OUT, N OUT, N OUT, N	字节、字和双字右移
SLB SLW SLD	OUT, N OUT, N OUT, N	字节、字和双字左移
RRB RRW RRD	OUT, N OUT, N OUT, N	字节、字和双字循环右移
RLB RLW RLD	OUT, N OUT, N OUT, N	字节、字和双字循环左移
FILL	IN, OUT, N	用指定的元素填充存储器空间
逻辑操作		
ALD OLD		与一个组合 或一个组合
LPS LRD LPP LDS		逻辑堆栈（堆栈控制） 读逻辑栈（堆栈控制） 逻辑出栈（堆栈控制） 装入堆栈（堆栈控制）
AENO		对 ENO 进行与操作
ANDB ANDW ANDD	IN1, OUT IN1, OUT IN1, OUT	对字节、字和双字取逻辑与
ORB ORW ORD	IN1, OUT IN1, OUT IN1, OUT	对字节、字和双字取逻辑或
XORB XORW XORD	IN1, OUT IN1, OUT IN1, OUT	对字节、字和双字取逻辑异或
INVB INVW INVD	OUT OUT OUT	对字节、字和双字取反 （1 的补码）
字符串指令		
SLEN	IN, OUT	字符串长度
SCAT	IN, OUT	连接字符串
SCPY	IN, OUT	复制字符串
SSCPY	IN, INDX, N, OUT	复制子字符串
CFND	IN1, IN2, OUT	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、查找和转换指令		
ATT	TABLE, DATA	把数据加到表中
LIFO FIFO	TABLE, DATA TABLE, DATA	从表中取数据
FND= FND<> FND< FND>	SRC, PATRN, INDX SRC, PATRN, INDX SRC, PATRN, INDX SRC, PATRN, INDX	根据比较条件在表中查找数据
BCDI IBCD	OUT OUT	把 BCD 码转换成整数 把整数转换成 BCD 码
BTI ITB ITD DTI	IN, OUT IN, OUT IN, OUT IN, OUT	将字节转换成整数 将整数转换成字节 把整数转换成双整数 把双整数转换成整数
DTR TRUNC ROUND	IN, OUT IN, OUT IN, OUT	把双字转换成实数 把实数转换成双字 把实数转换成双整数
ATH HTA ITA DTA RTA	IN, OUT, LEN IN, OUT, LEN IN, OUT, FMT IN, OUT, FM IN, OUT, FM	把 ASCII 码转换成 16 进制格式 把 16 进制格式转换成 ASCII 码 把整数转换成 ASCII 码 把双整数转换成 ASCII 码 把实数转换成 ASCI 码
DECO ENCO	IN, OUT IN, OUT	解码 编码
SEG	IN, OUT	产生 7 段格式
中断		
CRETI		从中断条件返回
ENI DISI		允许中断 禁止中断
ATCH DTCH	INT, EVENT EVENT	 给事件分配中断程序 解除事件
通讯		
XMT RCV	TABLE, PORT TABLE, PORT	自由口传送 自由口接受信息
TODR TODW GPA SPA	TABLE, PORT TABLE, PORT ADDR, PORT ADDR, PORT	网络读 网络写 获取口地址 设置口地址
高速指令		
HDEF	HSC, Mode	定义高速计数器模式
HSC	N	激活高速计数器
PLS	X	脉冲输出

S7-200 应用示例

H

本章概述：

H.1 模拟电位器

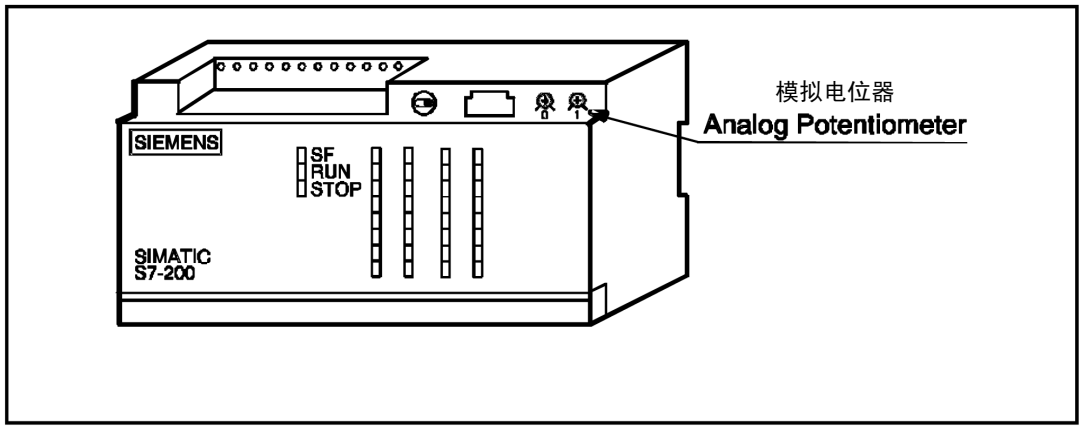
概述

本例包含了有关 SIMATIC S7-200 的模拟电位器（POT）的使用信息。电位器的位置转换为 0 至 255 之间的数字值，然后，存入两个特殊存储器字节 SMB28 和 SMB29 中，分别对应电位器0和电位器1 的值。

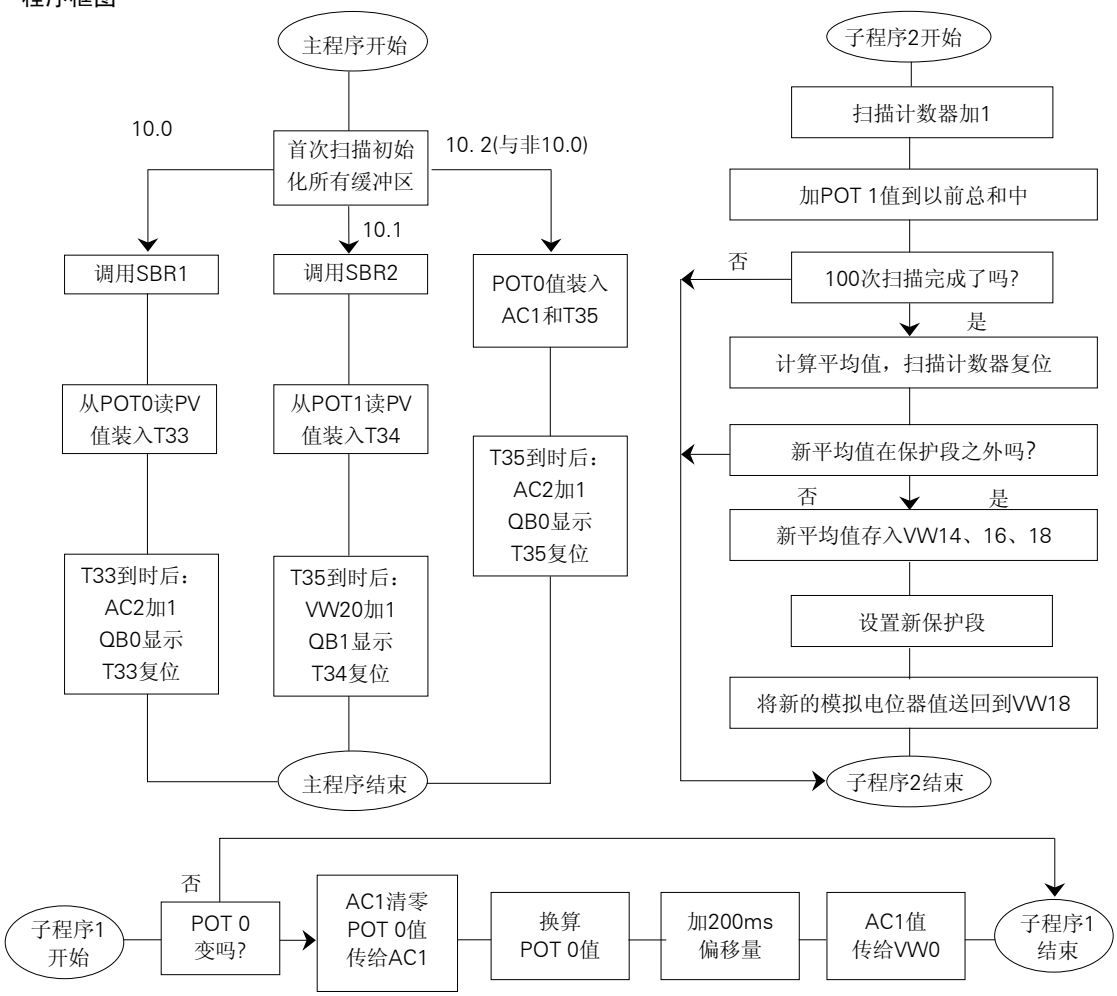
需要一把小螺丝刀用以调整电位器的位置。

本应用示例介绍了使用模拟电位器调整定时器设定值的三种方案。

例图



程序框图



程序和注释

方案1说明了用模拟电位器对定时器设定值进行细调的方法。首先通过程序中的偏移量（本例中为20ms）对定时器进行粗调，然后再用电位器能把定时器的设定值精确地调整到满意的设置。每个定时器周期之后，执行子程序1中的指令，把POT 0的值（在SMB28中）读到AC1，除以2，再加上200ms偏移量。返回主程序时，AC2中的定时器循环计数值加1，并拷贝到输出字节（QB0），以供显示。

在方案2中，对电位器1（POT 1）的100次扫描值在AC3中累加后并取平均，再存入VW12。如果该值低于低保护限值VW14，或高于高保护限值VW16（两者均在首次扫描时初始化），则将新值VW12拷贝到VW14、VW16和VW18中。然后再分别对VW16和VW14的值减、加3ms，作为新限值，而VW18中的平均值被传回主程序作为定时器T34的设定值。返回主程序时，VW20中的定时器循环计数值加1，并拷贝到输出字节（QB1），以供显示。

在方案3中，把电位器0（POT 0）的值直接作为定时器T35的设定值，AC2中的定时器循环计数值加1，并拷贝到输出字节（QB0），以供显示。

本程序长度为110个字。

```
// 标题：模拟电位器：
//  * * * * * * * * * * * * * * * *主程序 * * * * * * * * * * * * * * * *
// 这是S7-200的一个演示程序，介绍了使用模拟电位器调整定时器设定值的三种方案。
// 方案1：对来自POT 0的值进行换算并加偏移量，以调整定时器的设定值，可以从200ms调到的1.48s。
          每个定时器周期QB0加1。
// 方案2：从POT 1来的值经过滤波给定时器提供0ms到约2.55s的稳定的设定值。每个定时器周期QB1
          加1。
// 方案3：把POT 0的值直接作为定时器设定值。每个定时器周期QB0加1。

// 模拟电位器POT 0和POT 1的值可以分别从SMB28和SMB29中以一个字节读出。
// 每次扫描时，POT的值会变化一点，方案1和2都能为定时器提供稳定的设定值。
// 方案1的设定值会改变1次或2次，但每个定时器周期只装载一次。
// 方案2的设定值非常稳定，每次扫描都装载。
// 方案3的设定值每次扫描都会改变。

// 主程序：
LD SM0.1                  // 首次扫描时清除工作缓冲区：
MOVD+0，AC0               // AC0=0。
MOVD+0，AC3               // AC3=0。
MOVW+0，VW10              // VW10=0。
MOVW+32000，VW14          // 低限工作区复位。
MOVW+0，VW16              // 高限工作区复位。

// 方案1：
// 每次扫描时POT的值会改变一点。
// 下面的指令用来在每个定时器周期捕获一次换算后的值，并提供一个稳定的定时器设定值。
LD      I0.0              // 如果输入I0.0为1状态，则选方案1。
TON     T33，VW0          // POT 0的值经运算后作为T33的设定值。
CALL    1                 // 调用子程序1对POT 0的值进行换算并加偏移量。

LD      T33               // 若T33计时到，
INCW    AC2               // 则AC2加1，即定时器循环计数。
MOVB    AC2，QB0          // 把AC2的最低有效字节拷贝到输出字节QB0，以供显示。
R       T33，1            // 定时器T33复位。

// 方案2：
LD      I0.1              // 如果输入I0.1为1状态，则选方案2。
CALL    2                 // 调用子程序2，对POT 1的值进行滤波运算后存入VW18。
TON     T34，VW18         // VW18的值作为T34的设定值。
LD      T34               // 若T34计时到，
INCW    VW20              // 则VW20加1，即定时器循环计数。
MOVB    VB21，QB1         // 把VW20最低有效字节（VB21）拷贝到输出字节QB1，以供显示。
R       T34，1            // 定时器T34复位
```

```
// 方案3：
LD      I0.2              // 如果输入I0.2为1状态，
AN      I0.0              // 且方案1不在运行（I0.0=0），则选方案3。
MOVW    0，AC1            // 清除累加器1（AC1）
MOVB    SMB28，AC1        // 送POT 0的值到AC1。
TON     T35，AC1          // POT 0的值作为T35的设定值。

LD      T35               // 若T35计时到，
INCW    AC2               // 则AC2加1，即定时器循环计数。
MOVB    AC2，QB0          // 把AC2最低有效字节拷贝到输出字节QB0，以供显示。
R       T35，1            // 定时器T35复位。
MEND                      // 主程序结束
// 方案1的子程序
SBR     1                 // 子程序1。
// 换算POT 0的值并加上偏移量后存在VW0中，再返回主程序。
LD      T33               // 每个定时器周期检查POT 0的变化。
MOVW    0，AC1            // 清除累加器1（AC1）。
MOVB    SMB28，AC1        // 送POT 0的值给AC1。
DIV     2，AC1            // AC1除2，即把POT 0的输入范围从0~255换算成0~127。
+I      20，AC1           // 加200ms偏移量。
MOVW    AC1，VW0          // 把AC1值拷贝到VW0，以便能让程序员读取。
RET                       // 子程序1结束。
// 方案2的子程序
SBR     2                 // 子程序2。
// 对POT 1值采样100次，然后求平均值。
INCW    VW10              // 扫描计数器加1。
MOVB    SMB29，AC0        // 送POT 1的值到AC0。
+I      AC0，AC3          // 再加到以前的总和中（即累加POT1的值，共累加100次）。

LDW     VW10，100         // 100次扫描之后。
DIV     100，AC3          // 求平均值。
MOVW    AC3，VW12         // 存平均值。
MOVW    0，VW10           // 扫描计数器复位。
MOVD    0，AC3            // 工作内存复位。
AW<=    VW12，VW14        // 检查新的平均值是否在保护区之外。
OW>=    VW12，VW16        //
FILL    VW12，VW14，3     // 把新的平均值存入VW14，VW16，VW18。
-I      +3，VW14          // 设置新的低保护限。
+I      +3，VW16          // 设置新的高保护限。
RET                       // POT 1的滤波值存在VW18中，返回主程序
```

H.2 怎样使用高速计数器

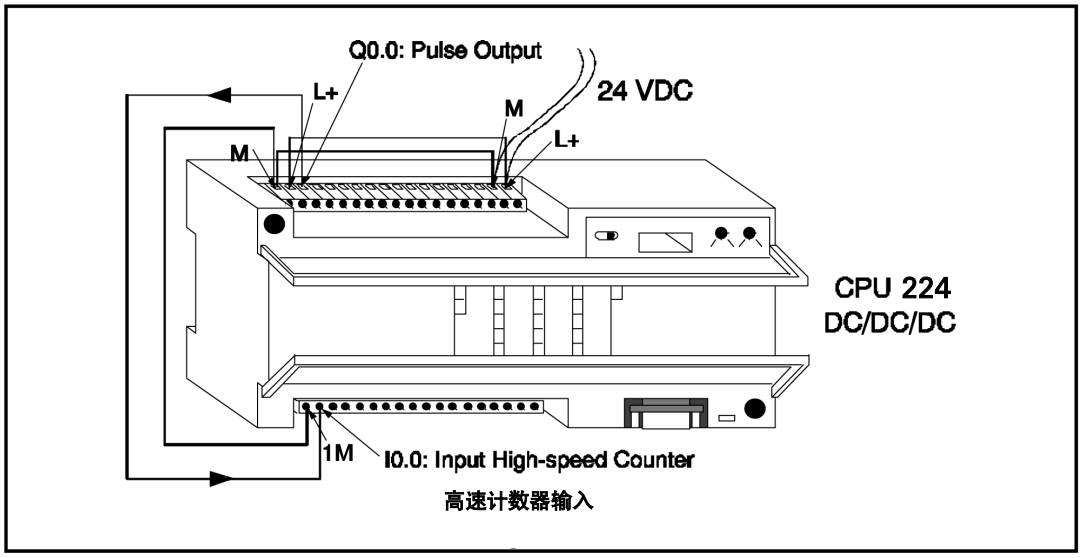
概述

本例叙述SIMATIC S7-200的高速计数器(HSC)的一种组态功能。对来自传感性（如编码器）信号的处理，高速计数器可采用多种不同的组态功能。

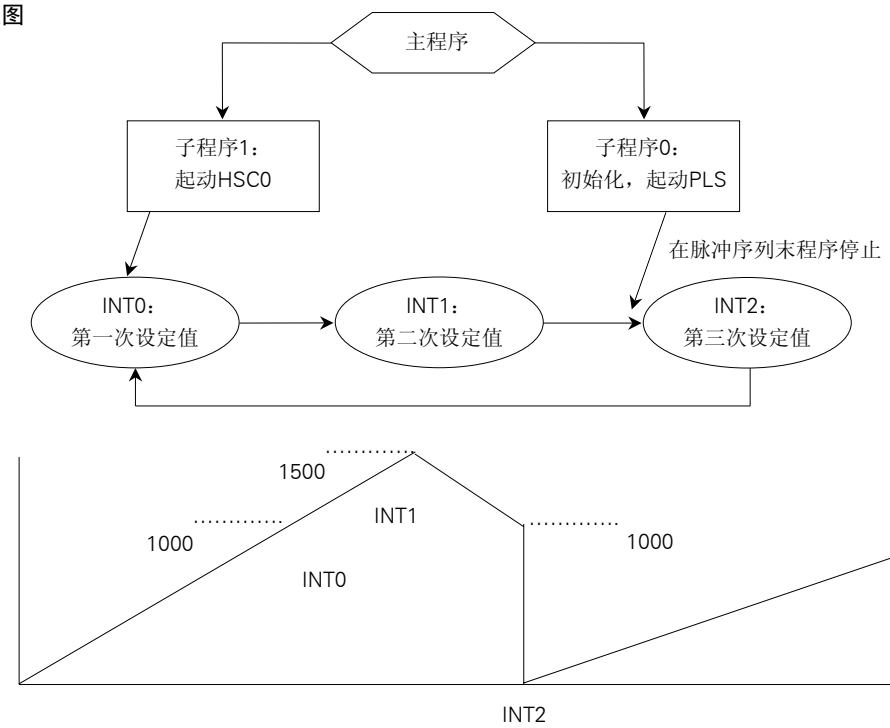
本例用脉冲输出（PLS）来为HSC产生高速计数信号，PLS可以产生脉冲串和脉宽调制信号，例如用来控制伺服电机。既然利用脉冲输出，必须选用CPU 224DC/DC/DC。

下面这个例子，展示了用HSC和脉冲输出构成一个简单的反馈回路，怎样编制一个程序来实现反馈功能。

例图



程序框图



程序和注释

本例描述了S7-200 DC/DC/DC的高速计数器(HSC)的功能。HSC计数速度比PLC扫描时间快得多，采用集成在CPU 224中的20K硬件计数器进行计数。总的来说，每个高速计数器需要10个字节内存用来存控制位、当前值、设定值、状态位。

本程序长度为91个字。

```
// 主程序：
// 在主程序中，首先将输出Q0.0置，0，因为这是脉冲输出功能的需要。再初始化高速计
// 数器HSC0，然后调用子程序0和1。
// HSC0起动后具有下列特性：可更新CV和PV值，正向计数。
// 当脉冲输出数达到SMD72中规定的个数后，程序就终止。

// 主程序
LD      SM0.1           // 首次扫描标志（SM0.1=1）。
R       Q0.0，1         // 脉冲输出Q0.0复位（Q0.0=0）。
MOVB    16#F8，SMB37    // 装载HSC0的控制位：
                        //    激活HSC0，可更新CV，可更新PV，
                        //    可改变方向，正向计数。
                        // HSC指令用这些控制位来组态HSC。

MOVD    0，SMD38        // HSC0当前值（CV）为0。
MOVD    1000，SMD42     // HSC0的第一次设定值（PV）为1000。
```

```
HDEF   0，0     // HSC0定为模式0。
CALL    0                    // 调用子程序0。
CALL    1                    // 调用子程序1。

MEND                        // 主程序结束。

//  * * * * * * * * * * * * * * * * * * * *

// 子程序0：
// 子程序0初始化，并激活脉冲输出（PLS）。
// 在特殊存储字节SMB67中定义脉冲输出特性：脉冲串（PT0），时基，可更新数值，激活PLS。
// SMW68定义脉冲周期，其值为时基的倍数。
// 最后，在SMD72中指定需要产生的脉冲数。（SMD72）为内存双字，即4个字节）。

// 子程序0
SBR     0                   // 子程序0
MOVB  16#8D，SMB67   // 装载脉冲输出（PLS0）的控制位：PT0，时基1ms，可更新，激活。
MOVW 1，SMW68        // 脉冲周期1ms。
MOVD  30000，SMD72   // 产生30000个脉冲。
PLS     0                   // 起动脉冲输出（PLS 0），从输出端Q0.0输出脉冲。
RET                         // 子程序0结束。

//  * * * * * * * * * * * * * * * * * * * *

// 子程序1：
// 子程序1起动HSC0，并把中断程序0分配给中断事件12（HSC 0的当前值CV等于设定值PV）。
// 只要脉冲计数值（当前值CV）达到设定值（PV），该事件就会发生。
// 最后，允许中断。

// 子程序1
SBR     0                   // 子程序1。
ATCH   0，12             // 把中断程序0分配给中断事件12（HSC 0的CV=PV）。
ENI                         // 允许中断。
HSC     0                   // 按主程序中对HSC 0的初始组态特性，起动HSC0。
RET                         // 子程序1结束。

//  * * * * * * * * * * * * * * * * * * * *

// 中断程序0：
// 当HSC 0的计数脉冲达到第一，设定值1000时，调用中断程序0。
// 输出端Q0.1置位（Q0.1=1）。
// 为HSC 0设置新的设定值1500（第二设定值）
// 用中断程序1取代中断程序0，分配给中断事件12（HSC 0的CV=PV）。
// 中断程序0

INT     0                   // 中断程序0。
S       Q0. 1，1          // 输出端Q0.1置位（Q0.1=1）。
MOVB  16#A0，SMB37   // 重置HSC 0的控制位，仅更新设定值（PV）。
```

```
MOVD  1500，SMD42     // HSC 0的下一个设定值为1500（第二设定值）。
ATCH  1，12           // 用中断程序1取代中断程序0，分配给中断事件12。
HSC   0               // 起动HSC 0，，为其装载新的设定值。
RETI                  // 中断程序0结束。

//  * * * * * * * * * * * * * * * * * * * * *

// 中断程序1：
// 当HSC 0的计数脉冲达到第二设定值1500时，调用中断程序1。
// 输出端Q0.2置位（Q0.2=1）。
// HSC 0改成减计数，并置新的设定值1000（第三设定值）。
// 用中断程序2取代中断程序1，分配给中断事件12（HSC 0的CV=PV）。

// 中断程序1：
INT   1               // 中断程序1。
S     Q0. 2，1        // 输出端Q0.2置位（Q0.2=1）。
MOVB  16#B0，SMB37    // 重置HSC 0的控制位，更新设定值，并改成减计数（反向计数）。
MOVD  1000，SMD42     // HSC 0的下一个设定值为1000（第三设定值）。
ATCH  2，12           // 用中断程序2取代中断程序1，分配给中断事件12。
HSC   0               // 起动HSC 0，，为其装载新的设定值和方向。
RETI                  // 中断程序1结束。

//  * * * * * * * * * * * * * * * * * * * * *

// 中断程序2：
// 当HSC 0的计数脉冲达到第三设定值1000时，调用中断程序2。
// 输出端Q0.1和Q0.2复位（Q0.1=0，Q0.2=0）。
// HSC 0的计数方向重新改为正向（增计数），并将当前计数值置为0，而设定值PV保持不变(1000)。
// 重新把中断程序0分配给中断事件12，程序再次起动HSC 0运行。
// 当脉冲数达到SMD72中规定的个数后，程序就终止。

// 中断程序2：
INT   2               // 中断程序2。
R     Q0. 1，2        // 输出端Q0.1和Q0.2复位（Q0.1=0，Q0.2=0）。
MOVB  16#D8，SMB37    // 重置HSC 0的控制位，更新CV，改为正向计数（增计数）。
MOVD  0，SMD38        // HSC 0的当前值复位（CV=0）。
ATCH  0，12           // 把中断程序0分配给中断事件12。
HSC   0               // 重新起动HSC 0。
RETI                  // 中断程序2结束。
```

H.3 自由通信口模式的简单应用

概述

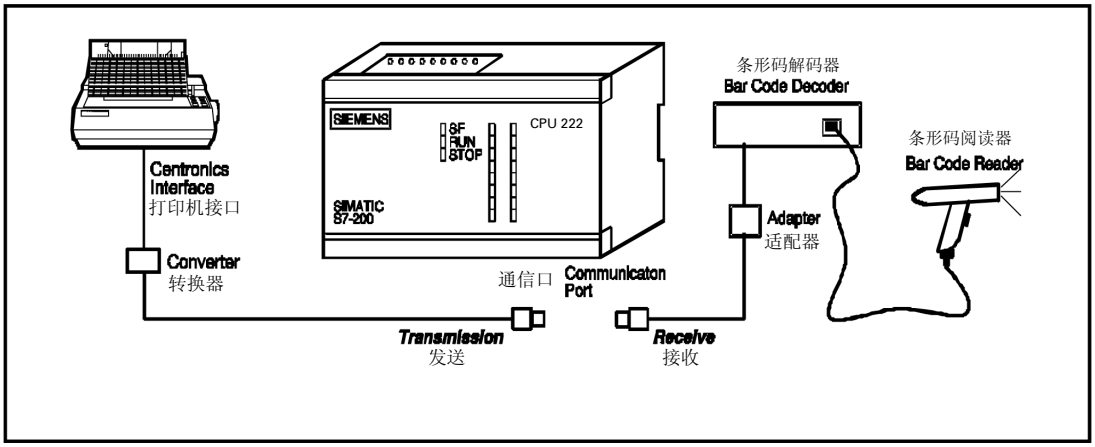
自由通信口模式(Freeport Mode)的通信协议可自由定义，通信所需要的信息存放在特殊存储字节SMB30中，用户须作如下说明：

- 奇偶校验
- 每个字符的位数
- 波特率

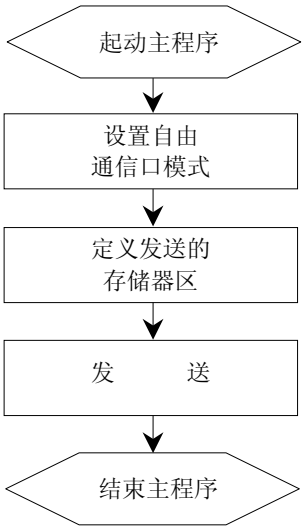
自由通信口模式可以接收和发送数据。本例用一个仿真的打印机程序来描述数据发送，再用一个条形码阅读器程序来说明数据接收。

例图

打印



机程序框图



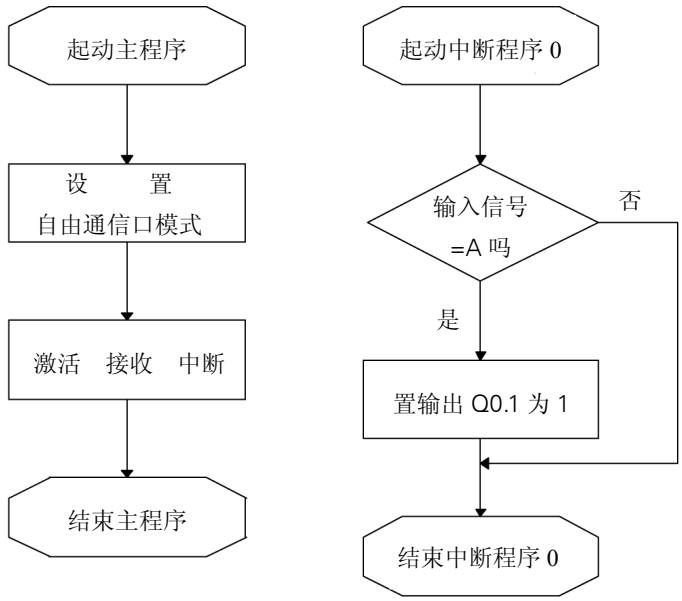
打印机程序和注解

此程序描述向打印机发送数据。为了简化此例，窗口下的终端程序可代替打印机作为接收器边接。打印机或终端的组态特性为9600波特，无奇偶校验，每字符8位。

本程序长度为13个字。

```
// 正确设置自由通信口模式对此应用很重要。
// 所需信息装载在特殊存储字节SMB30中。
// 这些输入数据可从操作手册中查询。
// 发送命令XMT包含了发送信息缓冲区的起始地址，该地址单元中只包含了发送信息的长度（以字节为单位）。

LD      SM0.1           // 第一次扫描（SM0.1=1）。
MOVB    +9,SMB30        // 自由通信口模式；9600波特，无奇偶校验，每字符8位。
MOVB    +1,VB100        // 信息长度为1个ASCII字符。
MOVB    16#41,VB101     // A字符长度为1个字节，A=41H（十六进制）。
LD      I0.1            // 输入I0.1起动发送。
EU                      // 识别脉冲上升沿。
XMT     VB100，0        // 发送。
MEND                    // 主程序结束。
```

条形码阅读器程序框图



条形码阅读器程序和注解

该程序描述数据接收，条形码阅读器通过接口把读到的数据用自由通信口模式发给SIMATIC S7-200。为简化此例，窗口下的终端程序可代替条形码阅读器作为发送器连接。

本程序长度为15个字。

```
//  * * * * * * * * * * * * 主 程 序 * * * * * * * * *

// 正确设置自由通信口模式对此应用很重要。
// 所需信息装载在特殊存储字节SMB30中。
// 这些输入数据可从操作手册中查询。
// 接收数据借助于中断实现，当数据进入自由编程接口，接收中断事件（8）。
// 就被触发了。
// 在此应用中，将中断程序0（INT0）赋予接收中断事件（8）。

LD      SM0.1           // 第一次扫描标志（SM0.1=1）。
MOVB  +9，SMB30         // 自由通信口模式：9600波特，无奇偶校验，每字符8位。
ATCH  +0，8             // 指定接收中断事件8调用中断程序0。
ENI                     // 允许中断。
MEND                    // 结束主程序。

//  * * * * * * * * 中断程序0 * * * * * * * *

// 在中断程序0，把存放在特殊存储字节SMB2中的接收字符和大写字母A作比较。
// 如果符合，则置输出位Q0.1为1。

INT                     // 接收中断程序0。
LDB     =SMB2，16#41    // 字节SMB2中的接收字符和A比较。
S       Q0.1，1         // 若字符为A，则置Q0.1为1。
RETI                    // 返回主程序。
```

H.4 处理脉宽调制

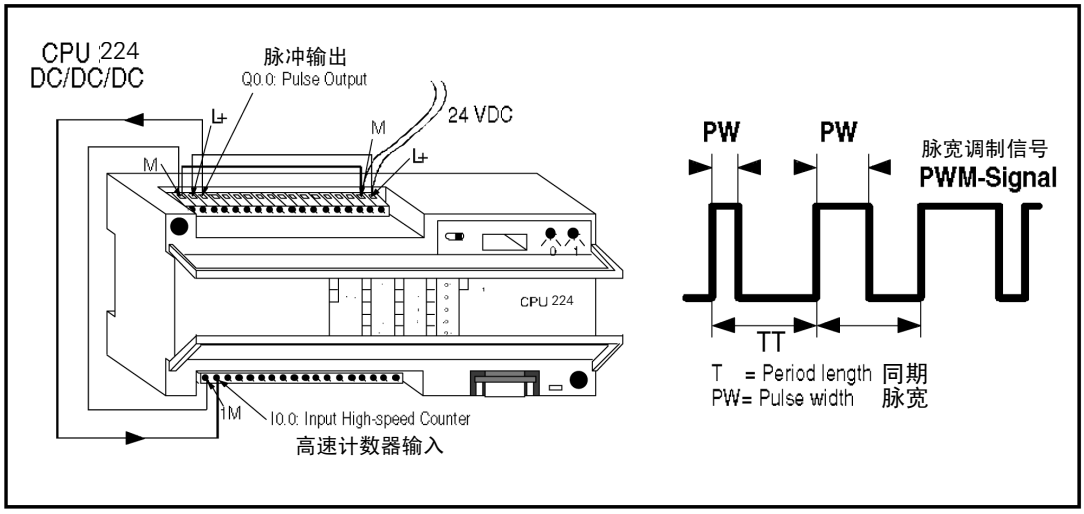
概述

在S7-200系列中输出端Q0.0和Q0.1能够输出方波信号，而且方波信号的周期和脉宽均能独立调节，其中脉宽指的是在一个周期内，输出信号处于高电平的时间长度。

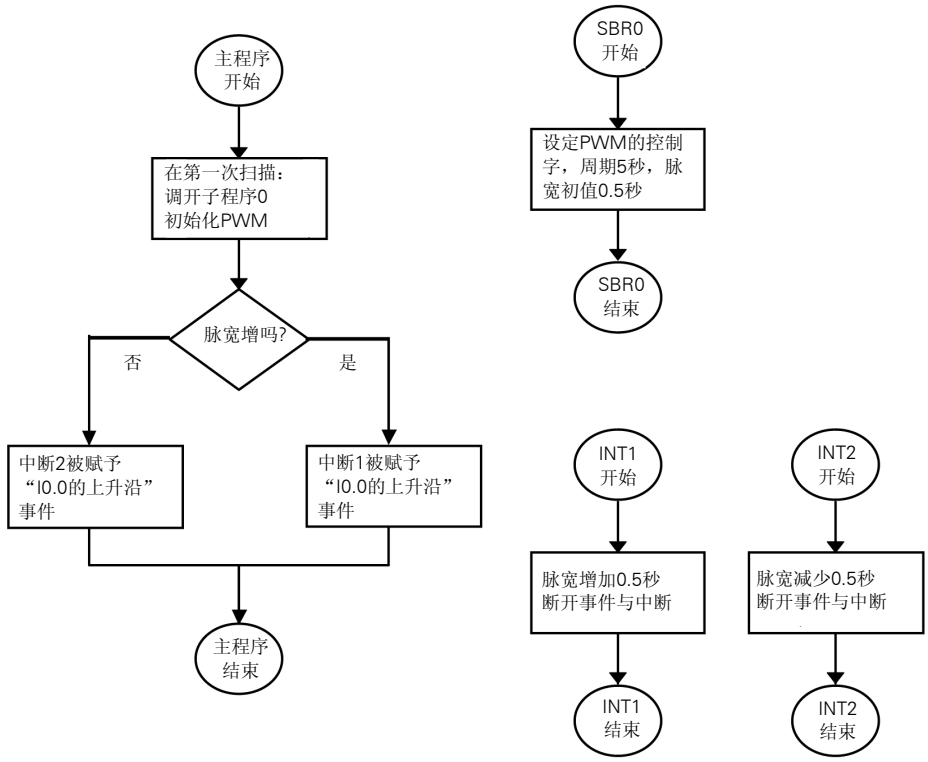
下面这个例子说明了脉宽调制（PWM）是如何工作的。输出端Q0.0输出方波信号，其脉宽每周期递增0.5秒，周期固定为5秒，并且脉宽的初始值为0.5秒。当脉宽达到设定的最大值4.5秒时，脉宽改为每周期递减0.5秒，直到脉宽为零为止。以上过程周而复始。

在这个例子中必须把输出端Q0.0与输入端I0.0连接，这样程序才能控制PWM。

例图



程序框图



程序和注解

特殊存储字节SMB67用来初始化输出端Q0.0的PWM。这个控制字内含PWM允许位，修改周期和脉宽的允许位，以及时间基数选择位等，由子程序0来调整这个控制字节。通过ENI指令，使所有的中断成为全局允许，然后通过PLS0指令，使系统接受各设定值，并初始化“PTO/PWM发生器”，从而在输出端Q0.0输出脉宽调制（PWM）信号。

另外，周期5秒是通过将数值5000置入特殊存储字SMW68来实现的，初始脉宽0.5秒则通过将500写入特殊存储字SMW70来实现的。

这个初始化过程是在程序的第一个扫描周期通过执行子程序0来实现，第一个扫描周期标志是SM0.1=1。当一个PWM循环结束，即当前脉宽为0秒时，将再一次初始化PWM。

辅助内存标记M0.0用来表明脉宽是增加，还是减少，初始化时将这个标记设为增加。输出端Q0.0与输入端I0.0相连，这样输出信号也可送到输入端I0.0。当第一个方波脉冲输出时，利用ATCH指令，把中断程序1（INT1）赋给中断事件0（I0.0的上升沿）。

每个周期中断程序1将当前脉宽增加0.5秒，然后利用DTCH指令分离中断INT1，使这个中断再次被屏蔽。如果在下次增加时，脉宽大于或等于周期，则将辅助内存标记位M0.0再次置0。这样就把中断程序2赋予事件0，并且脉宽也将每次递减0.5秒。当脉宽值减为零时，将再次执行，初始化程序（子程序0）。

本程序长度为63个字。

```
//  * * * * * * * * * * * * 主 程 序 * * * * * * * * *

LD      SM0.1                   // 在第一个扫描周期SM0.1=1。
CALL    0                       // 调用子程序0来起动PWM，即初始化PWM。
LDW>=   SMW70，VW0              // 如果脉宽大于等于（周期一脉宽），
R       M0.0，1                 // 则将辅助内存标记位M0.0置0。
LDW=    SMW70，0                // 如果脉宽为零，
CALL    0                       // 则调用子程序0来重新开始一个完整的PWM。
LD      I0.0                    // 如果输入I0.0=1。
A       M0.0                    // 且辅助内存标记位M0.0=1（脉宽增加），
ATCH    1，0                    // 则把INT1赋给事件0（输入I0.0的正向上升沿）。
LD      I0.0                    // 如果输入I0.0=1。
AN      M0.0                    // 且辅助内存标记位M0.0=0（脉宽减少），
ATCH    2，0                    // 则把INT2赋给事件0（输入I0.0的正向上升沿）。
MEND                            // 主程序结束。

//  * * * * * * * * 主 程 序 0 * * * * * * * * *

SBR     0                       // 初始化脉宽调制
S       M0.0，1                 // 将增加脉宽的辅助内存标记位M0.0置1。
MOVB    16#CB，SMB67            // 设定输出端Q0.0的PTO/PWM控制字节

// SM67.0：=1          ⇒允许接受新的周期。
// SM67.1：=1          ⇒允许接受新的脉宽。
// SM67.3：=1          ⇒时间基数为1ms（若为0，则时间基数为1μs）。
// SM67.6：=1          ⇒选择PWM模式（若为0，则PT0模式）。
// SM67.7：=1          ⇒允许高速输出功能。

MOVW    500，SMW70      // 指定初始脉宽（500ms）。
MOVW    5000，SMW68     // 周期为5s。

ENI                             // 允许全部中断。
PLS0                            // 对PTO/PWM生成器编程的指令。
MOVW    SMW68，VW0              // 将周期置入数据字VW0。
-1      500，VW0                // 将（周期－脉宽）的值置入数据字VW0。
RET                             // 子程序0结束并返回主程序。

//  * * * * * * * * 中断服务程序 1 * * * * * * * * *

INT     1                       // 增加脉宽。
+1      500，SMW70              // 脉宽增加500ms。
PLS     0                       // 对PTO/PWM生成器编程的指令。
DTCH    0                       // 将中断与事件0断开。
RETI                            // 中断服务程序1结束，并返回主程序。

//  * * * * * * * * 中断服务程序 2 * * * * * * * * *

INT     2                       // 减少脉宽。
-1      500，SMW70              // 脉宽减少500ms。
PLS     0                       // 对PTO/PWM生成器编程的指令。
DTCH    0                       // 将中断与事件0断开。
RETI                            // 中断服务程序2结束，并返回主程序。
```

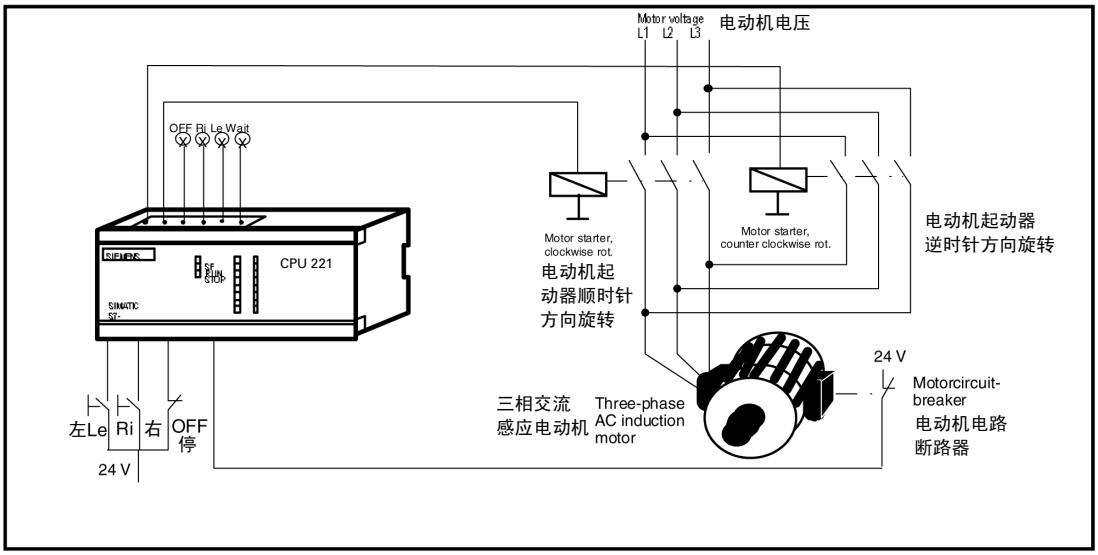
H.5 可逆电动机起动器电路——适用于改变三相交流感应电动机旋转方向

概述

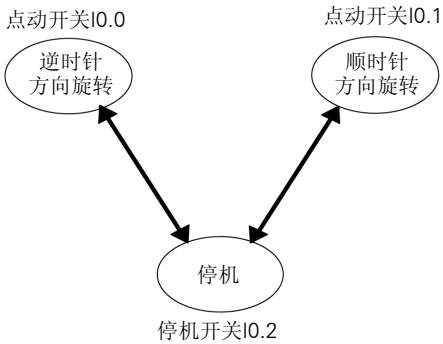
这个示例程序用于控制可双向运转的三相感应电动机。

当与输入点I0.0相连的左转点动开关（Le）闭合时，电动机逆时针方向旋转，当与输入点I0.1相连的右转点动开关（Ri）闭合时，电动机顺时针方向旋转。但这要有一个前题，即与输入点I0.3相连的电动机电路断路器和与输入点I0.2相连的停机开关（OFF）都没有动作。只有按下停机开关，并等待5秒钟之后，才可以改变电动机的旋转方向。这样做是为了让电动机有足够的时间刹车停转，然后再反向起动，如果需要电动机反转的话。如果与I0.0和I0.1相连的点动开关同时按下，电动机停转，并且不起动。

例图



程序框图



程序和注释

在程序起始部分，程序检查是否必须激活互锁电路。互锁电路防止电动机误起动，或者按错误方向起动。只有当所有点动开关都没有动作（位于起始状态），或者等待时间溢出时，互锁才清除，即M2.0被置成逻辑0。

如果电动机断路器（输入点I0.3）没有动作，停机点动开关（输入点I0.2）也没有动作（这两个触点都是常闭触点）：并且状态位M1.1没有被设置成顺时针旋转标志，则使能位M2.1被置为逻辑1。电动机才有可能逆时针旋转。代表逆时针旋转的状态位是M1.0。用类似方法可得到顺时针方向旋转的起动条件。

当点动起动开关（Ie和Ri）这一动作，并且互锁位和状态位都没有被设置成相反的旋转方向时，电动机起动。即相关的输出位和状态位被置位，状态位的作用是使输出能够自保。电动机逆时针方向旋转起动器由输出点Q0.0控制。电动机顺时针方向旋转起动器由输出点Q0.1控制。

除此外，另有一组信号灯指示电动机当前的运行状态；逆时针方向旋转指示灯（Le）与输出点Q0.4相连；顺时针方向旋转指示灯（Ri）与输出点Q0.3相连；关电机指示灯（OFF）与输出点Q0.2相连。

当电动机被停机时，“ED”的下降沿将辅助存储位M2.3置为1，进入停机模式。当M2.3被置位时，限制电动机再次起动的定时器开始计时，该定时器的预置时间是5秒（500×10ms），经过5秒钟后，内部存储器位M2.3被复位。在这段强制等待时间内与输出点Q0.5相连的信号灯（Wait）闪烁。如果状态位都没有被置位，则点亮与输出点Q0.2相连的停机状态指示灯（OFF）。

该程序的长度为61个字。

```
// 互锁：

LD      I0.1                    // 如果既命令电动机右转（Ri）。
A       I0.0                    // 又命令电动机左转（Le）。
O       M2.3                    // 或处于强制等待状态，则
S       M2.0，1                 // 设置互锁（M2.0=1）。

// 解除互锁：

LDN     I0.0                    // 如果既无左转命令（Le），
AN      I0.1                    // 也无右转命令（Ri）。
AN      M2.3                    // 并且等待时间溢出，则
R       M2.0，1                 // 解除互锁（M2.0=0）。

// 逆时针方向旋转使能

LD      I0.2                    // 如果无停机命令（OFF），
A       I0.3                    // 且电路断路器未动作
AN      M1.1                    // 且顺时针方向旋转状态位未置位，
=       M2.1                    // 则逆时针方向旋转使能位M2.1=1。

// 顺时针方向旋转使能

LD      I0.2                    // 如果无停机命令（OFF），
A       I0.3                    // 且电路断路器未动作
```

```
AN      M1.0                    // 且逆时针方向旋转状态位未置位，
=       M2.2                    // 则顺时针方向旋转使能位M2.2=1。

// 逆时针方向旋转

LD      I0.0                    // 如果命令电动机左转（Le）。
O       M1.0                    // 或逆时针方向状态位，

AN      M2.0                    // 且无互锁，
A       M2.1                    // 且逆时针方向旋转使能，则，
=       M1.0                    // 置逆时针方向旋转状态位M1.0=1。
=       Q0.0                    // 置电动机起动器输出点Q0.0=1。
=       Q0.4                    // 点亮逆时针方向旋转信号灯（Le）。

// 顺时针方向旋转

LD      I0.1                    // 如果命令电动机右转（Ri），
O       M1.1                    // 或顺时针方向状态位，
AN      M2.0                    // 且无互锁，
A       M2.2                    // 且顺时针方向旋转使能，则，
=       M1.1                    // 置顺时针方向旋转状态位M1.1=1。
=       Q0.1                    // 置电动机起动器输出点Q0.1=1。
=       Q0.3                    // 点亮顺时针方向旋转信号灯（Ri）。

// 检测边沿，关机过程

LDN     M1.0                    // 如果既无逆时针方向旋转状态位，
AN      M1.1                    // 也无顺时针方向旋转状态位，则，
=       Q0.2                    // 点亮关机输出信号指示灯（OFF）。
LD      Q0.2                    // 若关机时，
ED                              // 检测下降沿，则，
S       M2.3，1                 // 将辅助存储器标志位（代表关机状态）置位（M2.3=1）。

LD      M2.3                    // 若为关机状态，则
MOVW    500，VW20               // 装载重新起动前必须等待的时间值（500×10ms=5s）。
TON     T33，VW20               // 起动重新起动要强制等待的定时器（T33）。
A       T33                     //
R       M2.3，1                 // 超过等待时间后，将辅助存储器标志位复位（M2.3=0）。
MOVW    0，T33                  // 等待定时器清0。

// 关机状态指示，等待

LD      M2.3                    // 辅助存储器标志位（等待）。
A       SM0.5                   // 指示灯以1秒闪烁。
=       Q0.5                    // 点亮等待信号灯（Wait）。

MEND                            // 主程序结束。
```

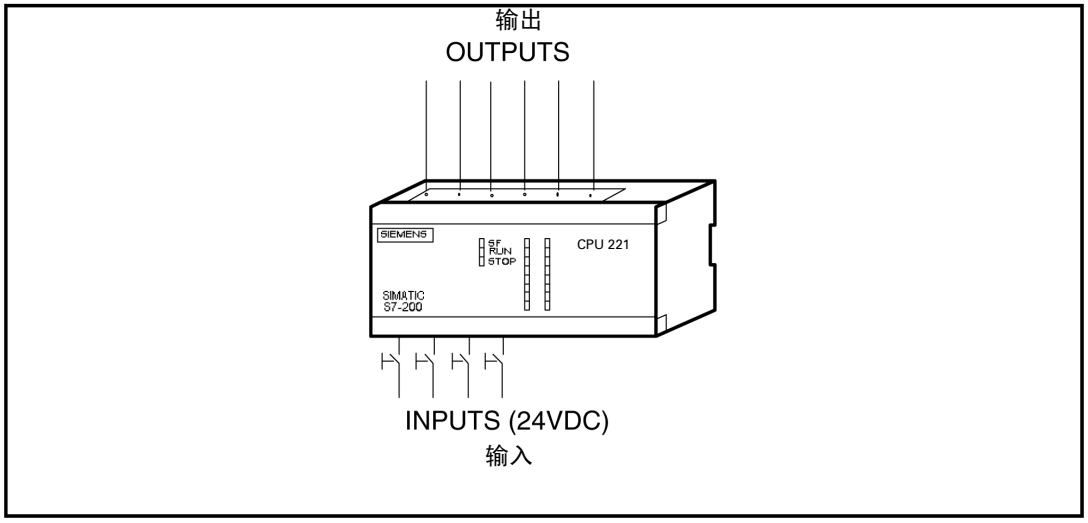
H.6 步执行顺序（事件鼓定时器）

概述

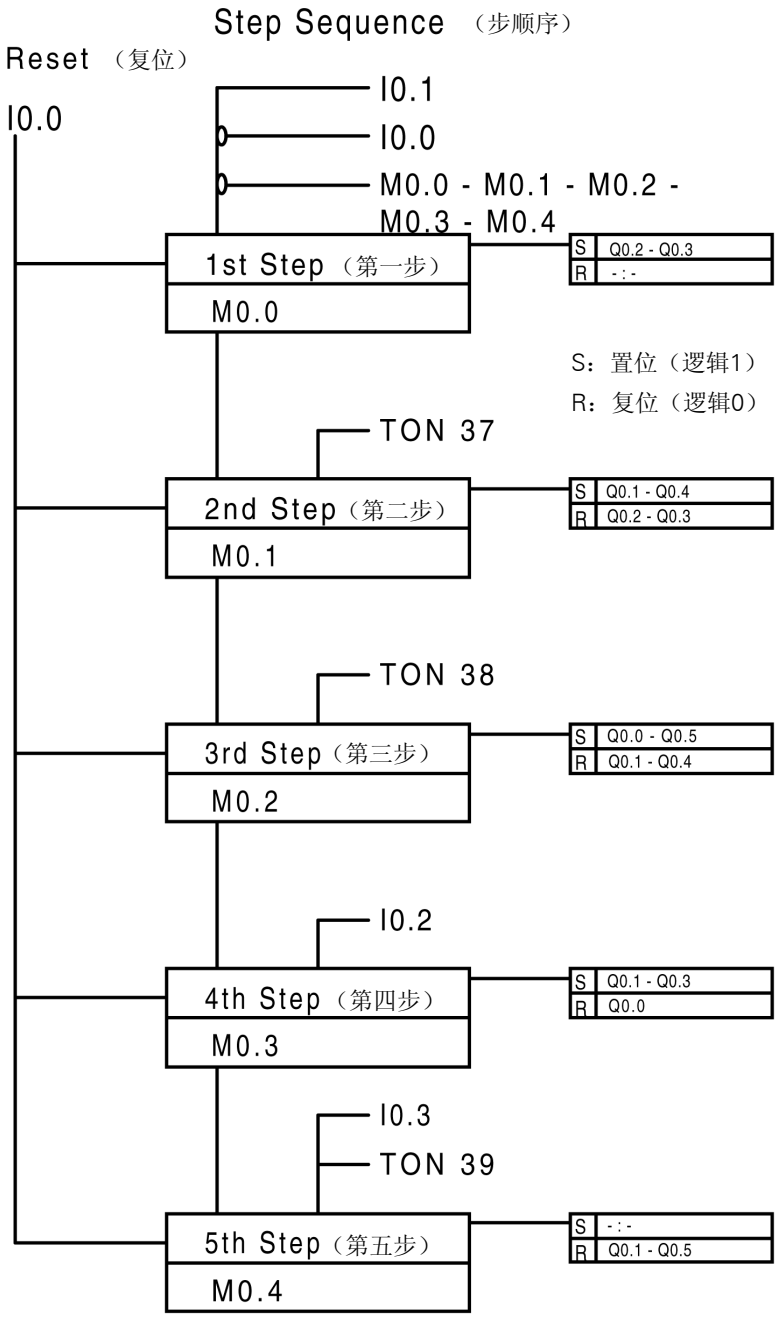
本程序实现了一个按事件步顺序执行的例子。每步均包含一系列的动作，一步紧跟一步，并且只有所有前提条件均满足时，才能执行。如下所示

	前提条件	实际输出		
第1步	I0.1已被置为1	Q0.2	Q0.3	
第2步	间隔5秒（T37定时器）	Q0.1	Q0.4	
第3步	间隔5秒（T38定时器）	Q0.0	Q0.5	
第4步	I0.2已被置为1	Q0.1	Q0.3	Q0.5
第5步	间隔5秒（T39定时器） 并且I 0.3已被置为1	Q0.3		
复位步执行顺序（I0.0已被置为1）		无		

例图



程序框图


Step Sequence （步顺序）
Reset （复位）
I0.0
I0.1
I0.0
M0.0 - M0.1 - M0.2 -
M0.3 - M0.4
1st Step（第一步）
M0.0
S Q0.2 - Q0.3
R - : -
S：置位（逻辑1）
R：复位（逻辑0）
TON 37
2nd Step（第二步）
M0.1
S Q0.1 - Q0.4
R Q0.2 - Q0.3
TON 38
3rd Step（第三步）
M0.2
S Q0.0 - Q0.5
R Q0.1 - Q0.4
I0.2
4th Step（第四步）
M0.3
S Q0.1 - Q0.3
R Q0.0
I0.3
TON 39
5th Step（第五步）
M0.4
S - : -
R Q0.1 - Q0.5

程序和注释

本示例程序由五个能连续地执行的步组成。每步实质上就是对某些输出置位和复位。在某一步可执行以前，必须满足一些必要的前提条件。例如，已按下某一开关，或满足了要求的等待时间。另外，还可以随时按开关I0.0来复位步执行顺序。这些前提条件与输出结果已在程序框图中描述过了。

本程序长度为116个字。

```
// * * * * * * * * * * 第 1 步 * * * * * * * * * *

LD    I0.1                    // 起动条件，若输入I0.1=1，
AN    I0.0                    // 且未复位（I0.0=0）
AN    M0.0                    // 且无步执行
AN    M0.1                    //
AN    M0.2                    //
AN    M0.3                    //
AN    M0.4                    //
S     M0.0，1                 // 则将第1步标志位M0.0置1。
LD    M0.0                    // 若为第1步，则
S     Q0.2，2                 // 设置输出（Q0.2=1，Q0.3=1）
TON   T37，50                 // 设与第2步之间的时间间隔为5s（100ms×50）

// * * * * * * * * * * 第 2 步 * * * * * * * * * *

LD    T37                     // 若第1个定时器的时间间隔（5s）结束（T37=1）
A     M0.0                    // 且第1步已执行完（M0.0=1），则
R     M0.0，1                 // 将第1步标志位M0.0置0，
S     M0.1，1                 // 将第2步标志位M0.1置1。
LD    M0.1                    // 若为第2步，则
S     Q0.1，1                 // 设置输出，即Q0.1=1。
S     Q0.4，1                 // 设置输出，即Q0.4=1。
R     Q0.2，2                 // 将输出Q0.2和Q0.3置0。
TON   T38，50                 // 设与第3步之间的时间间隔为5秒（100ms×50）

// * * * * * * * * * * 第 3 步 * * * * * * * * * *

LD    T38                     // 若第2个定时器时间间隔（5s）结束（T38=1）
A     M0.1                    // 且第2步已执行完（M0.1=1），则
R     M0.1，1                 // 将第2步标志位M0.1置0，
S     M0.2，1                 // 将第3步标志位M0.2置1。
LD    M0.2                    // 若为第3步，则
S     Q0.0，1                 // 设置输出，即Q0.0=1
S     Q0.5，1                 // 设置输出，即Q0.5=1
R     Q0.1，1                 // 将输出Q0.1和Q0.4置0。
R     Q0.4，1                 //
```

```
// * * * * * * * * * * 第 4 步 * * * * * * * * * *

LD     I0.2                    // 起动条件，若输入I0.2=1，
A      M0.2                    // 且第3步已执行完（M0.2=1，）则
R      M0.2，1                 // 将第3步标志位M0.2置0，
S      M0.3，1                 // 将第4步标志位M0.3置1，
LD     M0.3                    // 若为第4步，则
S      Q0.1，1                 // 设置输出，即Q0.1=1
S      Q0.3，1                 // 设置输出，即Q0.3=1
R      Q0.0，1                 // 将输出Q0.0置0。
TON    T39，50                 // 设与第5步之间的时间间隔为5s（100ms×50）

// * * * * * * * * * * 第 5 步 * * * * * * * * * *

LD     T37                     // 起动条件，若输入I0.3=1，
A      T39                     // 且第三个定时器时间间隔（5s）结束（T39=1）
A      M0.3                    // 且第4步已执行完，则
R      M0.3，1                 // 将第4步标志位M0.3置0。
S      M0.4，1                 // 将第5步标志位M0.3置1。
LD     M0.4                    // 若为第5步，则
R      Q0.1，1                 // 将输出Q0.1和Q0.5置0。
R      Q0.5，1                 //

// * * * * * * * * * * 复位步执行顺序 * * * * * * * * * *

LD     I0.0                    // 起动条件，若输入I0.0=1，则
R      M0.0，5                 // 将所有5步的标志位M0.0至M0.4置0
R      Q0.0，6                 // 将所有6个输出Q0.0至Q0.5置0

MEND                           // 结束
```

请参考SIMATIC STEP 7 编程参考手册4.1节“定时器指令”，为您提供了更多的关于定时器的信息。

H.7 S7-200用自由通信口模式和并行打印机连接

概述

本例描述了S7-200 CPU和外部设备（例如打印机）的连接方法。

该例中SIMATIC PLC自由通信口模式（Freeport Mode）向打印机发送信息。

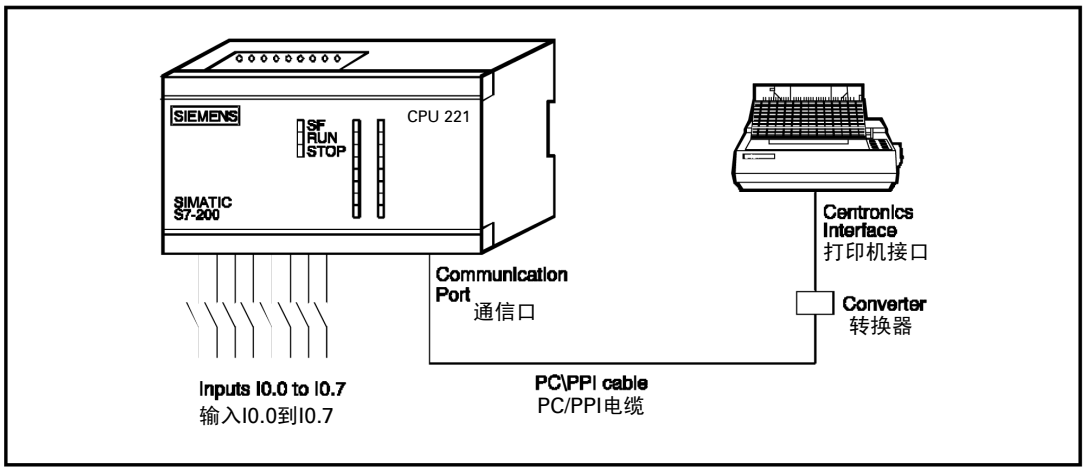
程序包含以下功能：

输入I0.0为1时，打印文字“SIMATIC S7-200”：

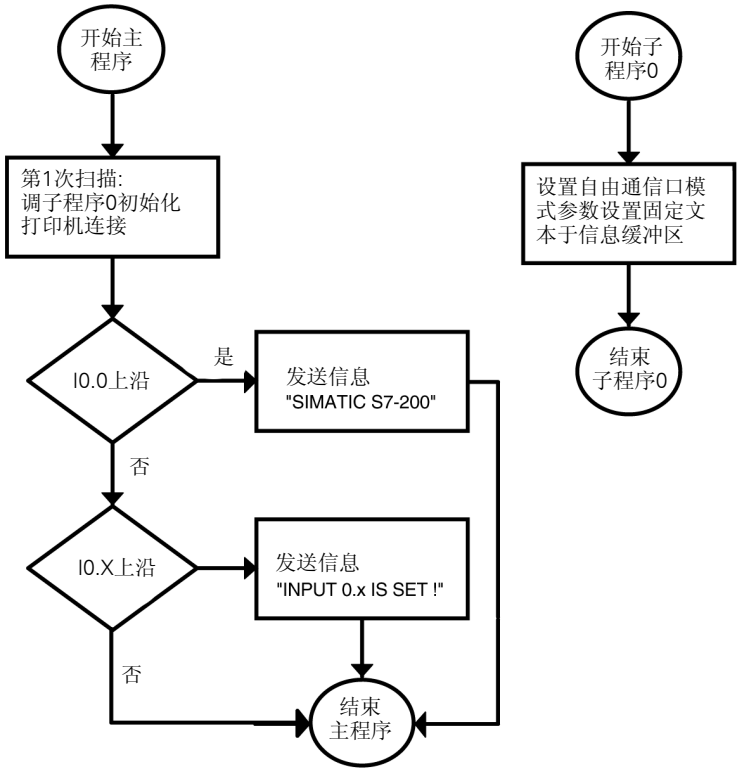
输入I0.1到I0.7为1时，打印句子“INPUT 0.×IS SET！”(其中×分别为1，2，……，7)。

假定打印机用并行接口连接，并假定发送波特率为9600波特。

例图



程序框图



程序和注解

此打印程序向并行打印机发送信息。

主程序检查S7-200模式开关，如果模式开关为RUN模式，则切换到自由通信口模式。

根据输入把相应的信息传送到打印机，主程序定义了这些内存变量。

以下的任务由子程序0完成：

子程序0包括设置自由通信口模式的参数和相应于不同输入的打印输出文本。

若输入I0.0=1，则打印“SIMATIC S7-200！”。

若输入I0.1=1，则打印“INPUT 0.1 IS SET！”。

若输入I0.2=1，则打印“INPUT 0.2 IS SET！”。

|

若输入I0.7=1，则打印“INPUT 0.7 IS SET！”。

程序结构如下：

MAIN（主程序）——初始化和输入请求

SBRO（子程序）——打印设置

本程序长度为118个字。

```
//* * * * * * * * * * 主 程 序 * * * * * * * * * *

LD      SM0.1                    // 第一次扫描标志：（SM0.1=1）。
CALL    0                        // 调用子程序0
LD      SM0.7                    // 若在TERM模式，则设置PPI（点到点接口）协议。
=       SM30.0                   // 若在RUN模式，则设置Freeport（自由通信口）协议

LD      I0.0                     // 起动打印输入I0.0
EU                               // 识别脉冲上升沿
XMT   VB80，0                    // 发送ASCII码，并打印(VB80中存放所发送的ASCII码个数)。

LD      I0.1                     // 起动打印输入I0.1
EU                               // 识别脉冲上升沿
MOVB 16#31，VB109                // 把1的ASCII码31存入VB109
XMT   VB100，0                   // 发送ASCII码，并打印(VB100中存放所发送的ASCII码个数)

LD      I0.2                     // 起动打印输入I0.2
EU                               // 识别脉冲上升沿
MOVB 16#32，VB109                // 把2的ASCII码#32存入VB109
XMT   VB100，0                   // 发送

LD      I0.3                     // 起动打印输入I0.3
EU                               //
MOVB 16#33，VB109                // 把3的ASCII码33存入VB109。
XMT   VB100，0                   //
LD      I0.4                     // 起动打印输入I0.4
EU
MOVB 16#34，VB109                // 把4的ASCII码34存入VB109。
XMT   VB100，0                   //

LD      I0.5                     // 起动打印输入I0.5。
EU
MOVB 16#35，VB109                // 把5的ASCII码35存入VB109。
XMT   VB100，0                   //

LD      I0.6                     // 起动打印输入I0.6。
EU
MOVB 16#36，VB109                // 把6的ASCII码36存入VB109。
XMT   VB100，0                   //

LD      I0.7                     // 起动打印输入I0.7。
EU
MOVB 16#37，VB109                // 把7的ASCII码37存入VB109。
XMT   VB100，0                   //
MEND                             // 主程序结束。
```

```
// * * * * * * * * * * 子 程 序 0 * * * * * * * * * *

// 子程序0

SBR    0                          // 设置打印信息。
MOVB +9，SMB30                    // 9600波特，无奇偶校验，每字符8位

MOVB +16，VB80                    // 信息长度为16个ASCII码字符：SIMATIC S7-200
MOVW 16#5349，VW81                // 字符SI
MOVW 16#4D41，VW83                // 字符MA
MOVW 16#5449，VW85                // 字符TI
MOVW 16#4320，VW87                // 字符C ␣
MOVW 16#5337，VW89                // 字符S7
MOVW 16#2D32，VW91                // 字符-2
MOVW 16#3030，VW93                // 字符00
MOVW 16#0D0A，VW95                //
MOVB +20，VB100                   // 信息长度为20个ASCII码字符：INPUT 0.×IS SET!
MOVW 16#494E，VW101               // 字符IN
MOVW 16#5055，VW103               // 字符PU
MOVW 16#5420，VW105               // 字符T␣。
MOVW 16#302E，VW107               // 字符0。
MOVB 16#20，VB110                 // 由主程序装载VB109（十六进制31，32……37）。
MOVW 16#4953，VW111               // 字符IS。
MOVW 16#2053，VW113               // 字符 ␣S。
MOVW 16#4554，VW115               //
MOVW 16#2021，VW117               // 字符ET。
MOVW 16#0D0A，VW119               // 字符␣！ 。

CRET                              // 子程序0结束。
```

H.8 通过自由通信口模式接受条形码阅读器的信息

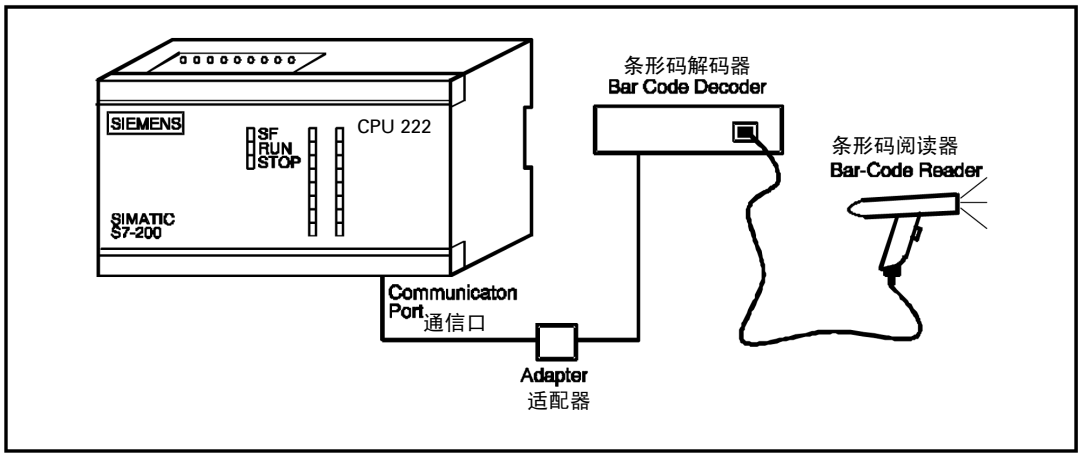
概述

本例说明如何将SIMATIC S7-222或S7-224与条形码阅读器配合作用。

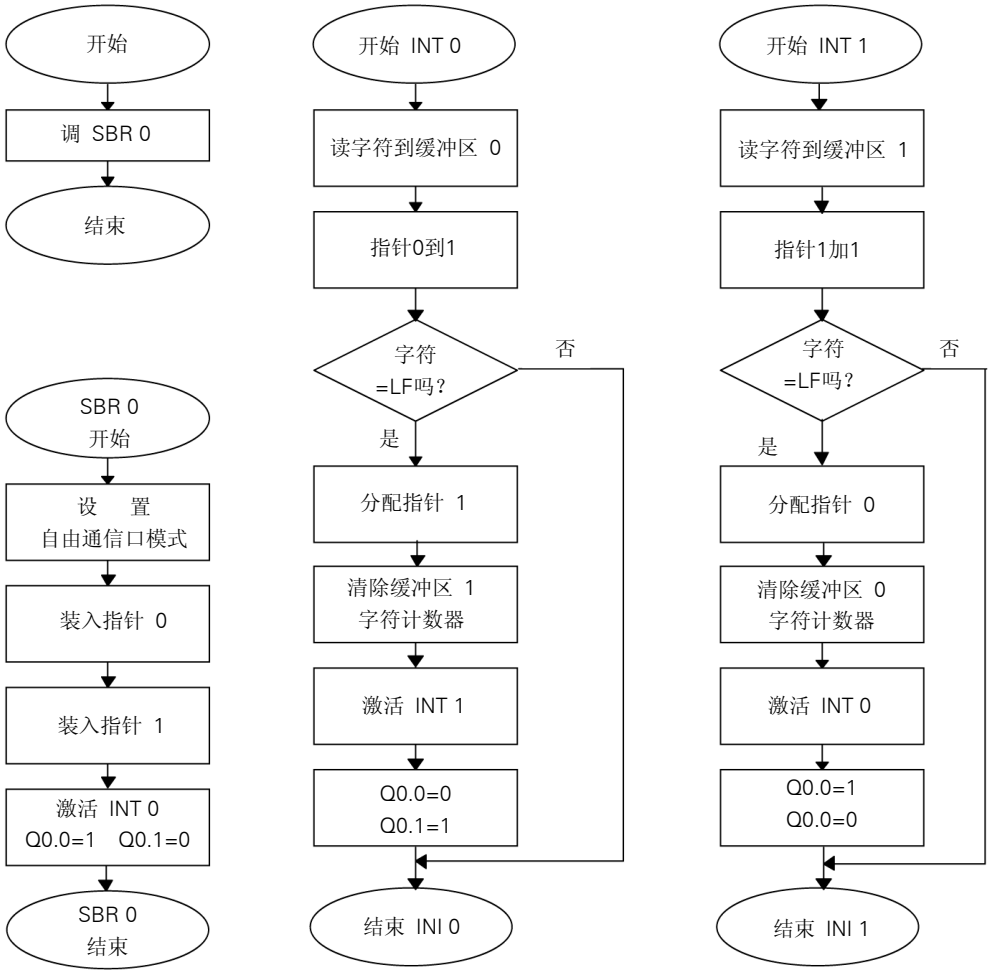
读入条形码的信息并经解码器翻译后，再通过自由通信口模式（Freeport Mode）把信息传入SIMATIC。在S7-222或S7-224的内存中有两个缓冲区，用来存储条形码信息，这两个缓冲区轮流地存储每次新读入的条形码。

通常这些数据可供程序调用。但本例中仅仅将信息存入接收缓冲区，可以用S7-200程序包来查看。

例图



程序框图



程序和注释

该程序从条形码阅读器接收信息再存入两个缓冲区。

从条形码解码器传出的信息是ASCII码形式，所接收的条形码存在SIMATIC内存中。这些数据可被程序利用，但本例中仅仅将信息存入接收缓冲区，可以用SIMATIC S7-200程序包来查看。

程序结束：

MAIN（主程序）：被始化程序

SBR0（子程序0）：接收条形码

INT0（中断程序0）：缓冲区0接收

INT1（中断程序1），缓冲区1接收

本程序长度为73个字。

```
// * * * * * * * * * * 主 程 序 * * * * * * * * * *

// 主程序
// 主程序的基本任务是初始化协议模式。
// 若开关在RUN位置，则特殊存储标志位SM0.7被设置为1，可以采用的通信口模式。准确的自由通
// 信口模式协议是通过特殊存储标志字节SM30来设定（参考Step 7编程参考手册2.6节）。若开关在
// TERM位置，则SM0.7为0，传输协议将是点到点接口协议（PPI），因此，条形码阅读器将不能向
// PLC发送信息。这是因为条形码阅读器不支持PPI协议。

LD      SM0.1                   // 第一次扫描标志位SM0.1=1
CALL    0                       // 调子程序0
LD      SM0.7                   // 若在TERM模式，则设置PPI（点到点接口）协议。
=       SM30.0                  // 若在RUN模式，则设置Freeport（自由通信口）协议。
MEND                            // 主程序结束

// * * * * * * * * * * 子 程 序 0 * * * * * * * * * *

// 子程序0
// 若开关在RUN位置，则置成自由通信口模式，SIMATIC从条形码解码器得到信息。选择了自由通信
// 口模式协议，并且定义了两个指针。缓冲区0首地址（VB100）装入指针
// VD50，缓冲区1首地址（VB200）装入指针VD60。
// 用VW54和VW64作字符计数器。激活中断程序0并允许中断。
// 读入的条形码将存到缓冲区0或1中。
// 子程序0

SBR     0                       // 准备接收条形码
MOVB    +4，SMB30               // 9600波特，无奇偶校验，每字符8位
MOVD    & VB100，VD50           // 指针指向缓冲区0
MOVD    & VB200，VD60           // 指针指向缓冲区1
MOVD    VD50，VD56              // VD56也指向缓冲区0
MOVW    +4，VW54                // 清除缓冲区0的字符计数器
ATCH    +0，8                   // 中断程序0处理缓冲区0的接收
MOVB    +1，QB0                 // 设Q0.0为1，Q0.1为0
ENI                             // 允许中断
RET                             // 结束子程序0

// * * * * * * * * * * 中断程序0 * * * * * * * * * *

// 中断0
// 若缓冲区0有效，则中断0中断程序，执行下面的程序：
// 指针VD56内容）加1，指向缓冲区的下一个位置，并且字符计数器也加1。
// 若字符是LF，则转向缓冲区1接收。
// 允许接收中断1，且设置输出Q 0.0为0、Q0.1为1。

// 中断程序0
```

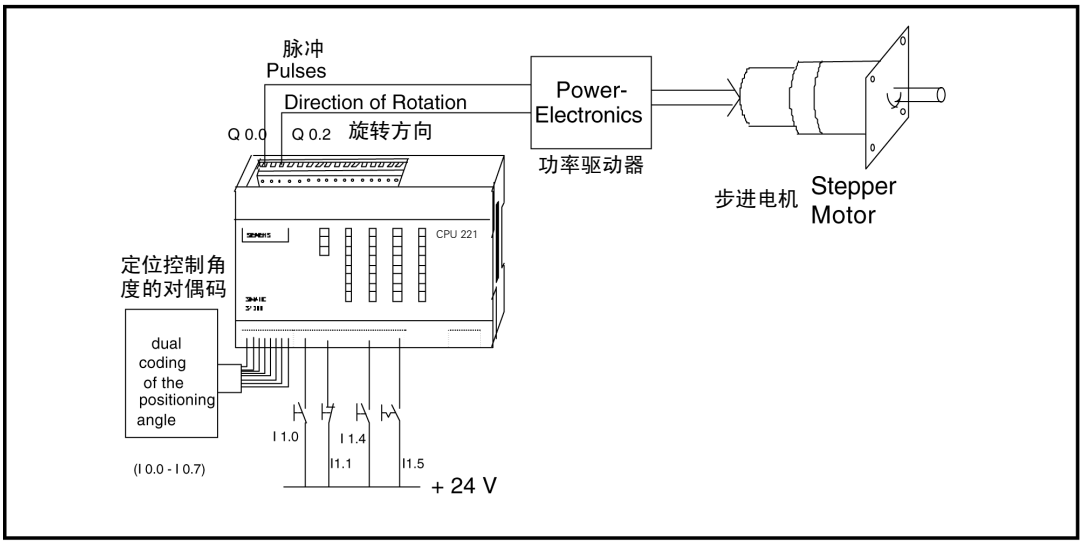
```
INT     0                       // 缓冲区0接收
MOVB SMB2，*VD56                // 字符装入缓冲区0
INCD   VD56                     // 指针加1，指向缓冲区的下一个位置。
INCW   VW54                     // 字符计数器加1
LDB=   SMB2，10                 // 若字符是LF，则
MOVD VD60，VD66                 // 使指针VD66指向缓冲区1，
MOVW+0，VW64                    // 清除缓冲区1的字符计数器，
ATCH   +1，8                    // 中断程序1处理缓冲区1的接收
MOVB +2，QB0                    // 设置Q0.0为0，Q0.1为1。
RET1                            // 中断程序0结束

// * * * * * * * * * * 中断程序1 * * * * * * * * * *

// 中断1
// 若缓冲区1有效，则中断1中断程序，执行下面的程序：
// 指针（VD66内容）加1，指向缓冲区的下一个位置，并且字符计数器也加1。
// 若字符是LF，则转向缓冲区0接收，允许接收中断0。
// 且设置输出Q0.0为1，Q0.1为0。

// 中断程序1

INT      1                      // 缓冲区1接收
MOVB   SMB2，*VD56              // 字符装入缓冲区1
INCD    VD66                    // 指针加1，指向缓冲区的下一个位置。
INCW    VW64                    // 字符计数器加1
LDB=    SMB2，1                 // 若字符是LF，则
MOVD   VD50，VD56               // 使指针VD56指向缓冲区0，
MOVW  +0，VW54                  // 清除缓冲区0的字符计数器，
ATCH    +0，8                   // 中断程序0处理缓冲区0的接收
MOVB   +2，QB0                  // 设置Q0.0为1，Q0.1为0。
RET1                            // 中断程序1结束
```

H.9 集成脉冲输出通过步进电机进行定位控制

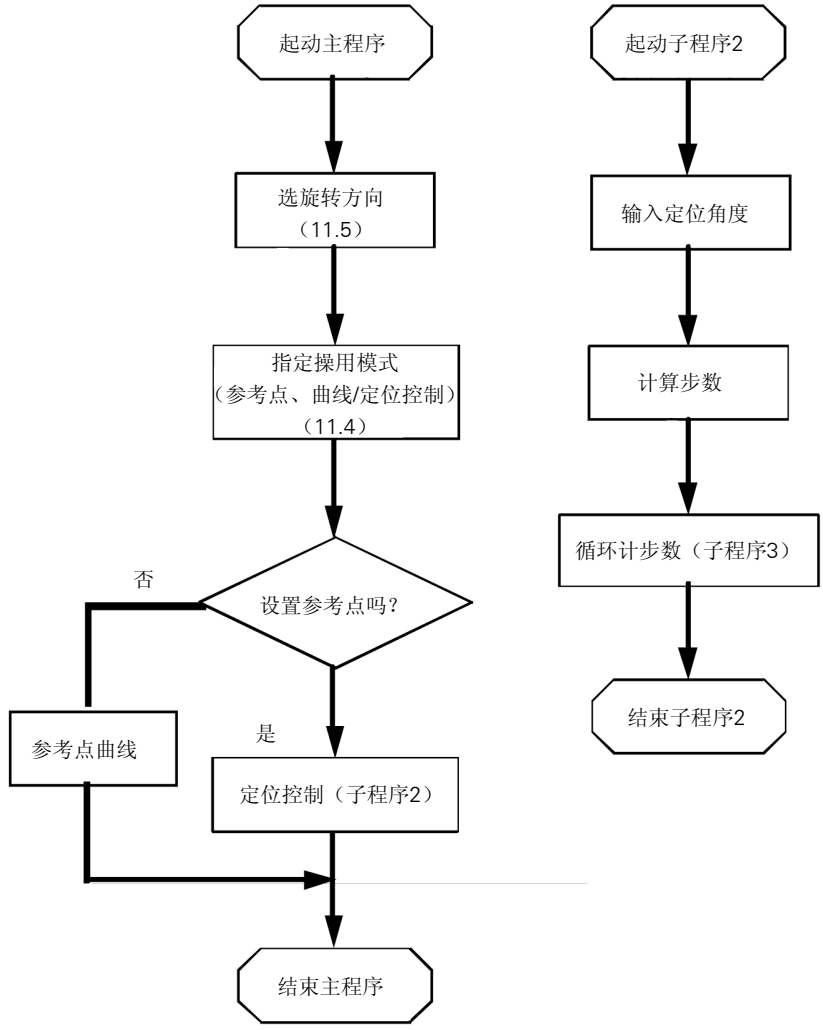
概述

关于定位控制（Positioning），调节（Regulated）和控制（Controlled）操作之间存在一些区别。步进电机不需要连续的位置控制，而在控制操作中得到应用。在以下的程序例子中，借助于S7-200产生的集成脉冲输出，通过步进电机来实现相对的位置控制。虽然这种类型的定位控制不需要参考点，本例还是初略地描述了确定参考点的简单步骤。因为实际上它总是相对一根轴确定一个固定的参考点，因此，用户借助于一个输入字节的对偶码（Duul coding）给CPU指定定位角度。用户程序根据该码计算出所需的定位步数，再由CPU输出相关个数的控制脉冲。

例图



程序框图



程序和注释

解除联锁，

“START”
参考点，按
参考点标志

“参考点曲
(Q1.0=0)。

此外，控制还为输出最大数量的控制脉冲做准备。当两次激活I1.4开关，便在两个模式之间切换。如果此信号产生，同时电机在运转，那么电机就自动停止。

实际上，一个与驱动器连接的参考点开关将代替手动操作切换开关的使用，所以，参考点标志能解决模式切换。

三、定位控制

如果确定了一个参考点（M0.3=1），而且没有联锁（请参考应用示例22），那么就执行相对的定位控制。在子程序2中，控制器从输入字节IBO读出对偶码方式的定位角度后，再存入字节MB11。与此角度有关的脉冲数，根据下面的公式计算：

$$N = \frac{\varphi}{360^\circ} \times S$$

n=控制脉冲数

φ=旋转角度（以度为单位）

S=每转所需的步数。

该示例程序所使用的步进电机采用半步操作方式（S=1000）。在子程序3中循环计算步数。如果现在按“STAR”按钮（I1.0），CPU将从输出端Q0.0输出所计算的控制脉冲个数，而且电机将根据相应的步数来转动，并在内部将“电机转动”的标志位M0.1置成1。

在完整的脉冲输出之后，执行中断程序0（请参考应用示例22），此程序将M0.1置成0，以便能够再次起动电机。象应用示例22那样，为更清晰起见，这一步并没有包含有程序流程图中。

四、停止电机

类似于应用示例22，按“STOP”（停止）按钮（I1.1），可在任何时候停止电机。执行子程序0中与此有关的指令。

本程序长度为147个字。

```
// 输入：   I0.0－I0.7      ——以度为单位的定位角（对偶码）
//          I1.0              “START”开关，起动电机
//          I1.1              “STOP”开关，停止电机
//          I1.4              “设置/取消参考点”开关
//          I1.5              选择旋转方向的开关

// 输出：   Q0.0             ——脉冲输出
//          Q0.2              旋转方向信号（Q0.2=1左转，Q0.2=0右转）
//          Q1.0              操作模式的显示
// 标志位：  M0.1             ——电机运转标志位
//          M0.2              联锁标志位
//          M0.3              参考点标志位
//          MD8，MD12         辅助标志位

// 标题：用脉冲输出进行定位控制

// 主程序
LD       SM0.1               // 仅首次扫描周期SM0.1才为1
R        M0.0，128           // MD0至MD12复位
ATCH     0，19               // 把中断程序0分配给中断事件19（脉冲串终止）
ENI                          // 允许中断
```

```
/ 脉冲输出功能的初始化
 MOVW     500，SMW68          // 脉冲周期T=500μs
 MOVW     0，，SMW70          // 脉冲宽度为0（脉宽调制）
 MOVD     429496700，SMD72    // 为参考点设定的最大脉冲数

 // 设置逆时针旋转
 LDN      M0.1                // 若电机停止
 A        I1.5                // 且旋转方向开关=1，
 S        Q0.2，1             // 则逆时针旋转（Q0.2=1）

 // 设置顺时针旋转
 LDN      M0.1                // 若电机停止
 AN       I1.5                // 且旋转方向开关=0，
 R        Q0.2，1             // 则顺时针旋转（Q0.2=0）

 // 联锁
 LD       I1.1                // 若按“STOP”（停止）按钮
 S        M0.2，1             // 则激活联锁（M0.2=1），

 // 解除联锁
 LDN      I1.1                // 若“START”（起动）按钮松开
 AN       I1.0                // 且“STOP”（停止）按钮松开
 R        M0.2，1             // 则解除联锁（M0.2=0），

 // 确定操作模式（参考点定位控制）
 LD       I1.4                // 若按“设置/取消参考点”按钮
 EU                           // 上升沿
 CALL1                        // 则调用子程序1

 // 起动电机
 LD       I1.0                // 若按“START”（起动）按钮
 EU                           // 上升沿
 AN       M0.1                // 且电机停止
 AN       M0.2                // 且无联锁
 AD>=     SMD72，1            // 且步数>=1，则

 MOVB     16#85，SMB67        // 置脉冲输出功能（PTO）的控制位
 PLS      0                   // 起动脉冲输出（Q0.0）
 S        M0.1，1             // “电机运行”标志位置位（M0.1=1）

 // 定位控制
 LD       M0.3                // 若已激活“定位控制”操作模式
 AN       M0.1                // 且电机停止
 CALL 2                       // 则调用子程序2。
```

```
// 停止电机
LD          I1.1                    // 若按“STOP”（停止）按钮
EU                                  // 上升沿
A           M0.1                    // 且电机运行，则
CALL 0                              // 调用子程序0
MEND                                // 主程序结束

//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

// 子程序1

SBR 0                               // 子程序0，“停止电机”
MOVB        16#CB，SMB67            // 激活脉宽调制
PLS         0                       // 停止输出脉冲到Q0.0
R           M0.1，1                 // “电机运行”标志位复位（M0.1=0）
RET                                 // 子程序0结束

SBR 1                               // 子程序1，“确定操作模式”
LD          M0.1                    // 若电机运行
CALL 0                              // 则调用子程序0，停止电机

// 申请“参考点曲线”
LD          M0.3                    // 若已激活“定位控制”，则
R           M0.3，1                 // 参考点标志位；复位（M0.3=0）
R           Q1.0，1                 // 取消“定位控制激活”信息（Q1.0=0）
MOVD        429496700，SMD72        // 为新的“参考点曲线”设定最大脉冲数。
CRET                                // 条件返回到主程序。

// 申请“定位控制”
LDN         M0.3                    // 若未设置参考点（M0.3=0），则
S           M0.3，1                 // 参考点标志位置位（M0.3=1）
S           Q1.0，1                 // 输出“定位控制激活”信息（Q1.0=1）
RET                                 // 子程序1结束

//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

// 子程序2

SBR 2                               // 子程序2，“定位控制”
MOVB        IB0，MB11               // 把定位角度从IBO拷到MD8的最低有效字节MB11。
R           M8.0，24                // MB8至MB10清零
DIV         9，MD8                  // 角度/9=q1+r1
MOVW        MW8，MW14               // 把r1存入MD12
MUL         25，MD8                 // q1×25→MD8
MUL         25，MD12
DIV         9，MD12                 // r1×25/9=q2+r2
```

```
CALL 3          // 在子程序3中循环步数
MOVW    0，MW12         // 删除r2
+D      MD12，MD8       // 把步数写入MD8
MOVD    MD8，SMD72      // 把步数传到SMD72
RET                     // 子程序2结束

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

// 子程序3

SBR 3                   // 子程序3，“循环步数”
LDW>=   MW12，5         // 如果r2>=5/9，则
INCW    MW14            // 步数增加1
RET                     // 子程序3结束

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

// 中断程序0，“脉冲输出终止”

INT 0                   // 中断程序0
R       M0.1，1         // “电机运行”标志位复位（M0.1=0）
RET1                    // 中断程序0结束
```

请参考SIMATIC STEP 7编程参考手册6.3节“高速输出指令”和6.2节“中断指令”，为您提供了更多的关于脉冲序列和中断程序的信息。

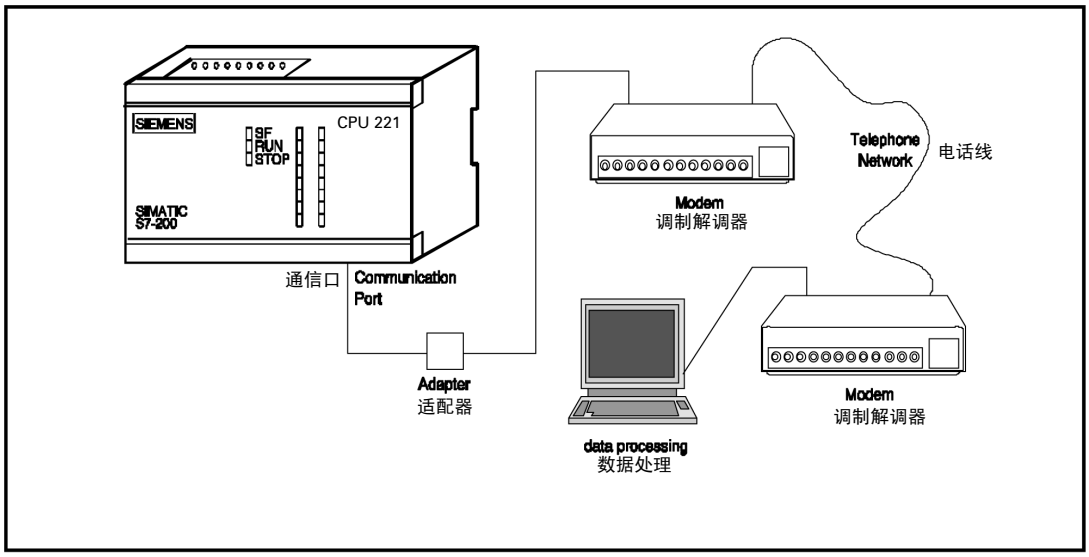
H.10 SIMATIC S7-221通过自由通信口模式控制贺氏（Hayes）调制解调器

概述

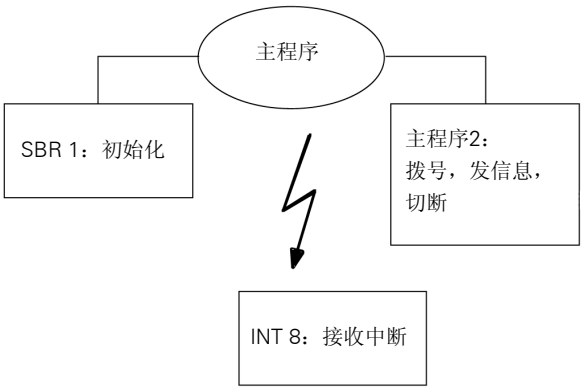
这一应用描述了如何通过SIMATIC调制解调器来使用与标准贺氏（Hayes）兼容的调制解调器，以及如何发送信息串。与S7-200相连的调制解调器，通过自由通信口模式（Freeport Mode）拨号叫另一台S7-200。由于贺氏调制解调器只能支持7个奇偶数据位，因而不能使用PPI模式，所以只能通过自由通信口模式来发送信息。

S7-200也可用作通信从设备。

例图



程序框图



程序和注解

本程序长度为189个字。

```
// 该应用将对通过调制解调器呼叫主系统进行初始化。
// 如果有信息要发送，S7-200发送信息给调制解调器，数据包括所要呼叫的电话号码和所要发出的信
// 息，信息存储在PLC的存储器中。
// 程序结构：
//          MAIN          对程序进行初始化
//          SBR 2         拨号、发送和切断
//          SBR 1         对信息和自由通信口模式进行初始化
//          INT 8         捕捉并处理来自调制解调器的响应

// * * * * * * 主 程 序 * * * * * *

// 主程序

LD        SM0.1              // 如果第一次扫描，则
CALL      1                  // 对信息进行初始化
LDB=      +0，MB0            // 如果要发信息，则
NOT                          //
CALL      2                  // 调用子程序进行发送。
LD        SM0.1              // 如果第一次扫描，则
MOVB      16#1，MB0          // 对发送的信息进行初始化
MEND                         // 主程序结束

// * * * * * * 子 程 序 2 * * * * * *

// 子程序2拨号，发送信息，然后切断。
// 子程序2

SBR       2                  // 子程序2
LDB=      16#0，MB0          // 如果命令有效，则
NOT
TON       T37，600           // 激活定时器T37（100ms×600=60s）
LD        737                // 如果定时器T37到时，则
MOVB      16#FF，MB3         // 显示错误
STOP                         // 异常结束

LD        SM0.0              // SM0.0总是1
TON       T38，15            // 运行定时器T38（15×100ms=1.5s），可用于所有情况。

// 状态1——拨号叫调制解调器

LD        M0.0               // 如果处于拨号状态，则
XMT       VB1010，0          // 拨号叫调制解调器
MOVB      16#2，MB0          // 转到等待状态
CRET                         // 异常结束
```

```
// 状态2－－等待连接和发送信息

LD      M0.1            // 如果处于等待连接状态
A       M2.1            // 且得到连接响应
A       T38             // 且等待定时器T38到时，则
XMT     VB130.0         // 发信息
MOVB    16#4，MB0       // 转到等待状态
CRET                    // 条件返回

// 状态3－－等待发送信息

LD      M0.2            // 如果正在等待发送结束
A       SM4.5           // 且发送已结束，则
MOVB    16#8，MB0       // 转到挂起（hang up）状态

// 状态4到8：挂起电话
// 挂起电话有以下5种状态（状态4至状态8）：
//      4）         等待1.5秒
//      5）         发换码（Escape）序列（+ + +）
//      6）         等待1.5秒
//      7）         发挂起命令
//      8）         等待离线

// 状态4－－第一次挂起暂停

LD      M0.3            // 如果是第一次等待，则
MOVB    16#10，MB0      // 显示发送状态，
R       T38，1          // 复位等待定时器T38
MOVW    +0，VW1008      // 清除“+”计数器
CRET                    // 跳过剩余部分

// 状态5－－发送换码（Escape）序列，这必须是发送3条信息的序列，只发送含3个字符的一条信息，
// 它将不起作用。

LD      M0.4            // 如果是发换码序列状态
AW=     +3，VW1008      // 且计数器=3，则
MOVB    16#20，MB0      // 转到第二等待状态
R       T38，1          // 复位等待定时器T38
CRET                    // 返回

LD      M0.4            // 如果是发送状态，且
A       T38             // 等待定时器T38到时，则
XMT     VB1000，0       // 发送换码序列
INCW    VW1008          // “+”计数器加1
CRET                    // 返回
```

```
// 状态6－－第二次挂起暂停（pause）

LD      M0.5                    // 如果是第二等待状态，且
A       SM4.5                   // 发送完毕，则
MOVB    15#40，MB0              // 显示挂起状态
R       T38，1                  // 复位等待定时器T38
CRET                            // 返回

// 状态7－－发送挂起命令

LD      M0.6                    // 如果是挂起状态
A       T38                     // 且等待定时器T38到时，则
XMT     VB1002,0                // 发送挂起命令
MOVB    16#80，MB0              // 转到第三等待状态。

// 状态8－－等待来自调制解调器的ACK

LD      M0.7                    // 如果是第三等待状态，且
A       T38                     // 等待定时器T38到时，则
MOVB    16#0，MB0               // 发送无效
MOVB    +0，MB2                 // 清除错误响应
RET                             // 子程序2结束

//******子 程 序 1******

// 子程序1执行信息和自由通信口模式的初始化
// 存储单元分配：

//      V1000-V1001             在线换码字符序列
//      V1002-V1007             挂起命令
//      V1008-V1009             换码字符序列计数器
//      V1010-V1???             拨号命令和电话号码

// 子程序1

SBR     1                       // 子程序1
LD      SM0.0                   // SM0.0总是1
MOVW    16#143，VW1000          // 在线换码序列字符（+）
MOVB    16#5，VB1002            // 设置挂起命令
MOVD    16#41544830，VD1003
MOVB    16#D，VB1007            // 回车

MOVB    16#9，VB1010            // 设置拨号指令
MOVD    16#41544454，VD1011     // 用按钮拨号
MOVD    16#32363137，VD1015     // 2617：一个调制解调器电话号码
MOVB    16#0D，VB1019           // 回车，发送信息
MOVB    +20，VB130              // 设置一条发送信息：S7-200 PHONE HOME
```

```
MOVD  16#53372D32，VD131     // 字符S7-2
MOVW      16#3030，VW135          // 字符00
MOVW      16#2020，VW137          // 字符 ␣ ␣
MOVD      16#50484F4E，VD139      // 字符PHON
MOVB      16#45，VB143            // 字符E
MOVB      16#20，VB144            // 字符 ␣
MOVD      16#484F4D45，VF145      // 字符HOME
MOVW      16#D0A，VW149
MOVB      16#9，SMB30             // 设置自由通信口：9600波特，每字符8位，无奇偶校
MOVB      +0，MB0                 // 显示空状态
MOVB      +0，MB2                 // 清除错误响应
ATCH      +8，8                   // 指定接收中断事件8调用中断程序8
ENI                               // 允许用户中断
R         T37，1                  // 复位定时器T37
REN                               // 子程序1结束

// * * * * * * 中断程序 8 * * * * * *

// 这个中断服务程序捕捉来自调制解调器的响应并且处理它们，代码的含义：

//        0                        // OK（好）
//        1                        // 连接
//        2                        // 振铃
//        3                        // 无载体
//        4                        // 错误
//        5                        // 连接1200
//        6                        // 无拨号按钮
//        7                        // 忙（占线）
//        8                        // 无回答
//        10                       // 连接2400
//        11                       // 连接4800
//        12                       // 连接9600

// 如果选中选项X0，则只有代码0，1，2，3，4。
// 代码0和1是可接受的，代码2、3、4是错误的。
// 实际被接收的框架是代码加CR

// 中断程序8

INT       8                       // 中断程序8

LDB=      16#D，SMB2              // 如果CR
CRETI                             // 则异常结束

LDB=      16#30，SMB2             // 如果“0”（OK）
MOVB      16#1，MB2               // 则置“OK”标志
CRETI                             // 返回
```

```
LDB    16#31，SMB2          // 如果“1”（连接），则
MOVB       16#2，MB2        // 置连接状态
R          T38，1           // 复位等待定时器T38
CRETI                       // 返回
MOVB       16#4，MB2        // 否则，显示出错标志
RETI                        // 中断子程序8结束
```

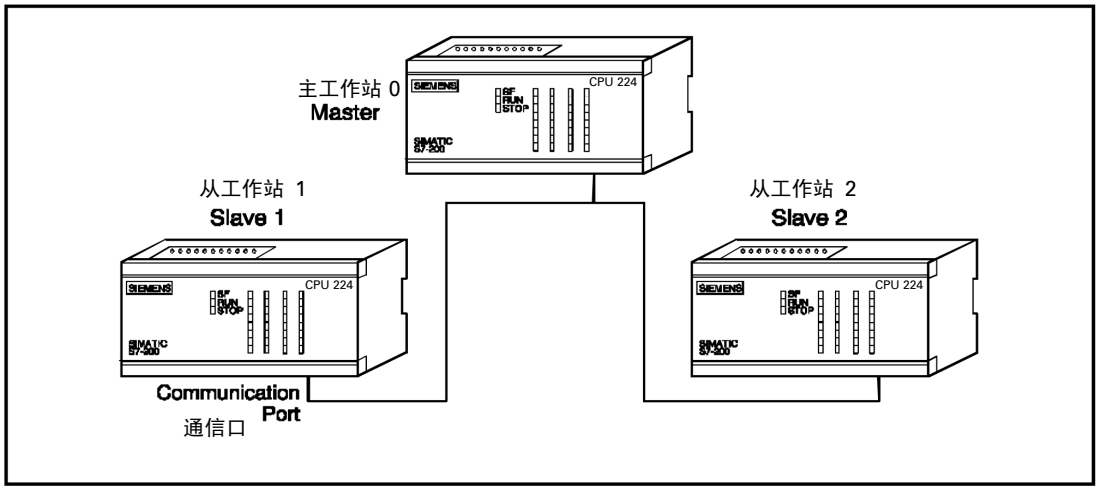
H.11 几台SIMATIC S7-200PLC使用自由通信口模式连接在一个远程I/O网络上

概述

在这个例子中连接了三台SIMATIC S7-200 CPU。工作站0被称为主工作站（Master），与工作站1和2相连，而工作站1和2被称为从工作站（Slave）。主工作站轮流发送四个字节的输出数据到每个从工作站。随之每个从工作站响应产生四个字节的输入数据。自由通信口模式（Freeprot Mode）被用来进行数据传输。

配备2个存储缓冲区，一个用作远程输入，另一个用作远程输出。发送的输出数据可从发送缓冲区获取，该数据是从输出缓冲区移到发送缓冲区的两个字长度的值。发送后，主工作站接收从工作站的响应，并且将数据存储在接收缓冲区。

例图



INT10	在发送完输出数据后的发送中断程序
INT11	接收信息第一个字符的中断程序
INT12	接收输入数据的中断程序
INT13	接收FCS字符的中断程序
INT14	静止线接收器中断程序

主工作站程序和注解

主工作台用于远程I/O的程序长度为191个字。

```
// 此程序用于连接两台、三台或更多台的SIMATIC S7-200或SIMATIC。
// S7-212，构成一个通信网络
// UART初始化如下：奇校验
                        每字符8位
                        38，400波特
                        19，200波特
// 有一点注意：只需将SMB30字节的设置值从16#C0改为16#C4，以及从16#C1改为16#C5，则应用
// 将会以19，200波特运行。

// 程序功能：
// 每条信息必须由一个检查序列字符所验证，该字符由此信息的数据字节进行异或运算形成。
// 远端I/O主工作站为工作站0，并且可以连接1台、2台或3台从工作站。如果只连接了一台从工作
// 站，则它必须为工作站1。
// 如果连接了2台从工作站，则它们必须是工作站1和2。如果连接了3台从工作站，则它们必须是工作
// 站 1、2和3。所连接的从工作站的数目必须作为一个参数提供给主工作站存入内存V0，对于从工作
// 站，将工作站的地址存入内存V0。
// 主工作站必须轮流发送四个字节的输出数据到每个从工作站，随之每个从工作站必须响应产生四个
// 字节的输入数据。配备两个V存储缓冲区，一个用作远程输入，另一个用作远程输出。主工作站中
// 的缓冲区大小如下面所示：

//      工作站1          工作站2          工作站3
//      输入缓冲区       输入缓冲区       输入缓冲区
//      VB 500字节0      VB 504字节0      VB 508字节0
//      VB 501字节1      VB 505字节1      VB 509字节1
//      VB 502字节2      VB 506字节2      VB 510字节2
//      VB 503字节3      VB 507字节3      VB 511字节3

//      工作站1          工作站2          工作站3
//      输出缓冲区       输出缓冲区       输出缓冲区
//      VB 540字节0      VB 544字节0      VB 548字节0
//      VB 541字节1      VB 545字节1      VB 549字节1
//      VB 542字节2      VB 546字节2      VB 550字节2
//      VB 543字节3      VB 547字节3      VB 551字节3

// 发送的输出数据可从发送缓冲区获取，该数据是从输出缓冲区移到发送缓冲区的两个字长度的值。
// 发送后，主工作站接收从工作站的响应，并且将数据存储在接收缓冲区内。发送和接收缓冲区的
// 格式如下面所示。其中VB 607是在产生发送检查和时所使用的存储单元。

// 发送缓冲区          发送缓冲区
// VB 600长度          VB 608字节0
// VB 601地址          VB 609字节1
// VB 602字节0         VB 610字节2
// VB 603字节1         VB 611字节3
// VB 604字节2
```

```
// VB 605字节3
// VB 606FCS
// VB 607xx

// 信息格式如下所示：
// 地址          B0    B1    B2    B3    FCS

//* * * * * * 主 程 序 * * * * * *

// 主菜单初始化程序
LD          SM0.0          // SM0.0总是1
CALL        0              // 调用子程序0
// 用户程序由此开始

// 用户程序由此结束
MEND                       // 主程序结束

//* * * * * * 子 程 序 0 * * * * * *

// 子程序0初始化自由通信口模式，并且处理指针

// 子程序0
SBR         0                    // 子程序0
LDN         SM0.7                // 当开关处于TERM位置时，则
MOVB        16#C0，SMB30         // 使自由通信口模式无效
DTCH        +8                   // 使接收中断无效
DTCH        +9                   // 使发送中断无效
DTCH        +10                  // 定时器中断无效
RET

LDN         SM30.0               // 如果不是自由通信口模式（即PPI模式），则使
MOVB        16#C1，SMB30         // 自由通信口模式有效：奇校验，每个字符8位，38.4K波特
ENI                              // 允许中断
MOVB        VB1，SMB34           // 设置定时器
CRET

LD          SM0.0                // SM0.0总是1
R           17.0，1              // 复位远程I/O指示器（I7.0=0）
MOVD        &VB540，VD630        // 指针指向输出数据缓冲区
MOVD        &VB500，VD634        // 指针指向输入数据缓冲区
MOVB        +6，VB600            // 发送缓冲区长度（6个字节）
MOVB        +1，VB601            // 工作站地址=1
MOVD        *VD630，VD602        // 置入发送的数据
MOVW        VW602，AC0           // 计算FCS
XORW        VW604，AC0           //
MOVB        AC0，VB606           //
XORW        AC0，VW606           // 存储FCS
```

```
ATCH    +1，10          // 使发送定时器有效（定时器中断事件10调中断程序0）
ATCH    +10，9          // 使发送中断有效（发送中断事件9调中断程序10）
XMT     VB600，0        // 发送数据
LBL     0               // 跳转（JMP）标识符0
LDN     17.0            // 如果远程I/O更新还没有完成
JMP     0               // 则等待它完成
LD      SM0.0
INCW    AC1
MOVW    AC1，AC2
SWAP    AC1
MOVW    AC1，VW544
SWAP    AC1
SRW     AC2，4
SWAP    AC2
MOVW    AC2，VW540
MOVB    VB500，QB0
MOVW    SMW22，VW10
EET                     // 子程序0结束

// * * * * * * 中断程序 0 * * * * * *

// 如果发送或接收数据，或重新起动定时器，则中断程序0，将禁止接收中断和定时器中断。

// 中断程序0
INT     0               // 当接收定时器到时，调中断程序0
LD      SM0.0           // SM0.0总是1
DTCH    +8              // 使接收中断无效
DTCH    +10             // 使定时器中断无效
LDB>=   VB601，VB0      // 如果这是网络中的最后一个从工作站，则
=       17.0            // 指示远程I/O循环结束
CRETI

LD      SM0.0
INCW    VW600           // 否则，工作站地址加1
+D，    +4，VD630       // 增大指针，指向下个工作站的输出数据缓冲区
+D      +4，VD634       // 增大指针，指向下个工作站的输出数据缓冲区
MOVD    *VD630，VD602   // 置入发送的数据
MOVW    VW620，AC0      // 计算FCS
XORW    VW604，AC0
MOVB    AC0，VB606
XORW    AC0，VW606      // 存储FCS
ATCH    +1，10          // 使发送定时器有效（定时器中断事件10调中断程序1）
ATCH    +10，9          // 使发送中断有效（发送中断事件9调中断程序10）
XMT     VB600，0        // 发送数据
RETI                    // 中断程序0结束
```

```
// * * * * * * 中断程序 1 * * * * * *

// 中断程序1处理XMT（发送）定时器到时

// 中断程序1
INT       1                  // 当发送信息定时器到时，调用中断程序1
LD        SM0.0              // SM0.0总是1
DTCH      +10                // 停止定时器
DTCH      +9                 // 停止发送器
STOP                         // 引起程序模式的转换
RETI                         // 中断程序1结束

// * * * * * * 中断程序 10 * * * * * *

// 中断程序10中断程序，以接收数据

// 中断程序10
INT       10                 // 发送器中断（发送中断事件9调中断程序10）
LD         SM0.0             // SM0.0总是1
DTCH      +9                 // 使发送中断无效
ATCH      +0，10             // 起动接收定时器
ATCH      +11，8             // 使接收数据有效（接收中断事件8调中断程序11）
RETI                         // 中断程序10结束

// * * * * * * 中断程序 11 * * * * * *

// 如果接收的信息的第一个字符，那么中断程序11中断程序

// 中断程序11
INT       11                 // 接收信息的第一个字符
LDN       SM3.0              // 如果没有奇偶校验错误
AB=       SMB2，VB601        // 且有正确的工作站响应，则
MOVB=     +0，AC0            // 初始化检查和寄存器
MOVW=     +4，AC1            // 置入接收字符总数
MOVD      &VB608，VD638      // VD638指针指向接收缓冲区
ATCH      +12，8             // 使接收数据有效（接收中断事件8调中断程序12）
CRETI

LD        SM0.0              // 否则，错误的工作站响应或有错误
ATCH      +0，10             // 使接收定时器有效
ATCH      +14，8             // 使静止线接收器有效
RETI                         // 中断程序11结束
```

```
// * * * * * * 中断程序 12 * * * * * *

// 如果接收到数据，那么中断程序12中断程序

// 中断程序12
INT       12                  // 接收数据
LDN       SM3.0               // 如果没有奇偶，校验错误，则
MOVB      SMB2，*VD638        // 将数据存储于接收寄存器中
INCD      VD638               // 指向下一个接收缓冲区的地址
XORW      SMW1，AC0           // 计算正在运行的检查和
DECW      AC1                 // 接收的字符数减1
LD        SM1.0               // 如果接收到四个字符，则
ATCH      +13，8              // 使接收FCS有效
CRETI

LD        SM3.0               // 如果有奇偶校验错误，则
ATCH      +0，10              // 使接收定时器有效
ATCH      +14，8              // 使静止线接收器有效
RETI                          // 中断程序12结束

// 中断程序13
INT       13                  // 接收FCS字符
LD        SM0.0               // SM0.0总是1
ATCH      +0，10              // 使接收定时器有效
LD        SM3.0               // 如果有奇偶校验错误
AB=       SMB2，AC0           // 或者如果FCS不匹配
CRETI
LD        SM0.0               // 否则
MOVD      VD608，*VD634       // 存储接收到的数据
RETI                          // 中断程序13结束

// * * * * * * 中断程序 14 * * * * * *

// 中断程序14重新激发接收定时器

// 中断程序14
INT       14                  // 静止线接收器
LD        SM0.0               // SM0.0总是1
ATCH      +0，10              // 重新激发接收定时器
RETI                          // 中断程序14结束
```

从工作站程序结构

Main	主程序
SBR 0	激活用于初始化工作站的程序
SBR 1	使通信无效
INT 0	静止线定时器
INT 1	发送定时器中断程序
INT 2	信息定时器
INT 10	发送输入数据后的发送中断
INT 11	接收信息的首个字符
INT 12	接收输出数据
INT 13	接收FCS字符
INT 14	静止线接收器

从工作站程序和注释

从工作站用于远程I/O的程序长度为148个字。

```
// 此程序连接两台、三台或更多台SIMATIC S7-200，构成通信网络
// UART初始化如下：奇校验
                每字符8位
                38，400波特（S7-214）
                19，200波特（S7-212）
// 有一点注意：只需将字节SMB 30设置值从16#C0改为16#C4，以及从16#C1改为16#C5，则应用
// 将会以19，200波特运行。

// 程序功能：
// 每条信息必须由一个检查序列字符所验证，该字符由此信息中的数据字节进行异或运算形成。
// 远端I/O主工作站为工作站0，并且可以连接一台、两台或三台从工作站。
// 如果只连接了一台从工作台，则它必须为工作站1。如果连接了2台从工作站，则它们必须为工作站
// 1和2。
// 如果连接了3台工作站，则它们必须为工作站1，2和3。所连接的从工作站的数目必须作为一个参数
// 提供给主工作站存入内存V0，对于从工作站，将工作站地址存入V0。
// 每个工作站都有一个如下所示的输入和输出缓冲区：

//                  输入缓冲区
//        IB0       字节 0
//        IB1       字节 1
//        IB2       字节 2
//        IB3       字节 3

//                  输出缓冲区
//        QB0       字节 0
//        QB1       字节 1
//        QB2       字节 2
//        QB3       字节 3
```

```
// 当主工作站发送一条信息时，则从工作站由主工作站存储在接收缓冲区中的输出数据所赋址：并且
// 响应主工作站的发送，发送存于发送缓冲区中的输入数据。
// 这些缓冲区如下所示，其中VB 607是在产生发送检查和时所使用的存储单元。

//              发送缓冲区                发送缓冲区
//        VB 600    长度          VB 608    字节 0
//        VB 601    地址          VB 609    字节 1
//        VB 602    字节 0        VB 610    字节 2
//        VB 603    字节 1        VB 611    字节 3
//        VB 604    字节 2
//        VB 605    字节 3
//        VB 606    FCS
//        VB 607    xx

// 信息格式如下所示：
// 地址          B0    B1    B2    B3    FCS

//* * * * * * 主 程 序 * * * * * *

// 主菜单初始化程序

// 主程序

LD          SM0.7                 // 如果开关在RUN位置
A           SM0.1                 // 且为首次扫描，则
CALL        0                     // 起动通信
LD          SM0.7                 // 当开关被移到RUN位置
EU                                // 上升沿
CALL        0                     // 如果开关在TERM位置，则
LD          SM0.7                 // 起动的通信
CALL        1                     // 使通信无效
MEND                              // 主程序结束

//* * * * * * 子 程 序 0 * * * * * *

// 子程序0设置自由通信口模式，并使静止线定时器和静止线接收器有效

// 子程序0
SBR         0                     // 初始化从工作站
LD          SM0.0                 // SM0.0总是1
MOVB        16#C1，SMB30          // 使自由通信口模式有效：奇校验，每字符8位，38.4K波特
ENI                               // 允许中断
MOVB        VB1，SMB34            // 设置定时器为5ms
ATCH        +0，10                // 使静止线定时器有效（定时器中断事件10调中断程序0）
ATCH        +14，8                // 使静止线接收器有效（接收中断事件8调中断程序14）
RET                               // 子程序0结束
```

```
// * * * * * * 子 程 序 1 * * * * * *

// 子程序1使自由通信口模式通信无效，且禁止接收、发送和定时器中断

// 子程序1
SBR       1                    // 使通信无效
LD        SM0.0                // SM0.0总是1
MOVB      16#C0，SMB30         // 使自由通信口模式无效
DTCH      +8                   // 使接收器无效
DTCH      +9                   // 使发送器无效
DTCH      +10                  // 使定时器无效
RET                            // 子程序1结束

// * * * * * * 中断程序 0 * * * * * *

// 中断程序0使接收输入数据有效

// 中断程序0
INT       0                    // 当静止线定时器到时，调中断程序0
LD        SM0.0                // SM0.0总是1
ATCH      +11，8               // 使接收输入数据有效
RETI                           // 中断程序0结束

// * * * * * * 中断程序 1 * * * * * *

// 如果发送或接收数据，或重新激发定时器，则中断程序1将禁止定时器中断和发送中断

// 中断程序1
INT       1                    // 当发送信息定时器到时，调中断程序1
LD        SM0.0                // SM0.0总是1
DTCH      +10                  // 停止定时器
DTCH      +9                   // 停止发送器
STOP                           // 引起程序模式的转换
RETI                           // 中断程序1结束

// * * * * * * 中断程序 2 * * * * * *

// 当接收到信息的首个字符时，中断程序2中断程序

// 中断程序2
INT       2                    // 当信息定时器到时，调中断程序2
LD        SM0.0                // SM0.0总是1
ATCH      +0，10               // 使静止线定时器有效
ATCH      +14，8               // 使静止线接收器有效
RETI                           // 中断程序2结束
```

```
// * * * * * * 中断程序10 * * * * * *

// 如果发送信息，那么中断程序10中断程序。发送中断无效，静止线定时器开始运行，且使静止线接
// 收器有效。

// 中断程序10
INT       10                  // 发送器中断
LD        SM0.0               // SM0.0总是1
DTCH      +9                  // 使发送器中断无效
ATCH      +0，10              // 起动静止线定时器
ATCH      +14，8              // 使静止线接收器有效
RETI

// * * * * * * 中断程序11 * * * * * *

// 当接收到信息的首个字符时，中断程序11有效。设置信息定时器，可以接收数据。如果有错误，则
// 使静止线定时器和静止线接收器有效。

// 中断程序11
INT       11                  // 接收信息的首个字符
LDN       SM3.0               // 如果无奇偶校验错误
AB=       SMB2，VB0           // 且是本工作站的信息，则
MOVB      +0，AC0             // 初始化检查和寄存器
MOVW      +4，AC1             // 置入接收字符数
MOVD      & VB 608，VD638     // VD638指针指向接收缓冲区
ATCH      +2，10              // 使信息定时器有效
ATCH      +12，8              // 使接收数据有效
CRETI

LD        SM0.0               // 否则，如不同工作站或有错误，则
ATCH      +0，10              // 使静止线定时器有效
ATCH      +14，8              // 使静止线接收器有效
RETI                          // 中断程序11结束

// * * * * * * 中断程序12 * * * * * *

// 如果接收到数据且无错误，那么中断程序12中断程序。
// 将信息存储于缓冲区。结构检查序列FCS可被另一个中断所接收。
// 如果有错误发生，则激活静止线定时器和静止线接收器。

// 中断程序12
INT       12                  // 接收数据
LDN       SM3.0               // 如果无奇偶校验错误，则
MOVB      SMB2，*VD638        // 将数据存储在接收寄存器中
INCD      VD638               // 指向下一个接收缓冲区地址
XORW      SMW1，AC0           // 计算正在运行的检查和
DECW      AC1                 // 接收字符计数器减1
```

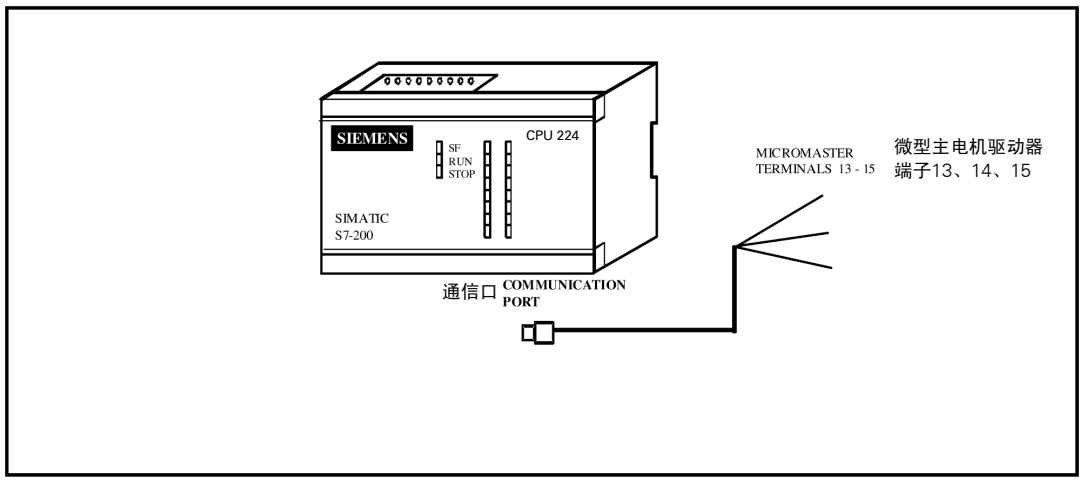
```
LD      SM1.0   // 如果接收到四个字符，则
ATCH        +13，8              // 使接收FCS有效
CRETI

LD          SM3.0               // 如果有奇偶，校验错误，则
ATCH        +0，10              // 使静止线定时器有效
ATCH        +14，8              // 使静止线接收器有效
RETI                            // 中断程序12结束

// * * * * * * 中断程序13 * * * * * *

// 中断程序13接收结构检查序列字符（FCS），无定时器和无接收中断被激活。当计算FCS后，激活
// 发送器定时中断和发送器中断。如果有错误，则激活静止线定时器和静止线接收器。

// 中断程序13
INT         13                  // 接收FCS字符
LD          SM0.0               // SM0.0总是1
DTCH        +8                  // 使接收器无效
DTCH        +10                 // 使定时器无效
LDN         SM3.0               // 如果无奇偶校验错误
AB=         SMB2，VC0           // 且FCS匹配，则
MOVD        VD608，QD0          // 存储输出数据
MOVB        +6，VB600           // 置入发送缓冲区的长度
MOVB        VB0，VB601          // 将工作站地址置入发送缓冲区
MOVD        1D0，VD602          // 将输入数据置入发送缓冲区
MOVW        VW602，AC0          // 计算FCS
XORW        VW604，ACO
MOVB        AC0，VB606
XORW        AC0，VW606          // 存储FCS
ATCH        +1，10              // 使发送器定时器有效
ATCH        +10，9              // 使发送器中断有效
XMT         VB 600，0           // 发送数据
CRETI
LD          SM0.0               // 否则，如果有奇偶校验错误，则
ATCH        +1，10              // 激活静止线定时器
ATCH        +14，8              // 激活静止线接收器

RETI                            // 中断程序13结束

// * * * * * * 中断程序14 * * * * * *

// 中断程序14激活定时器中断
INT         14                  // 静止线接收器
LD          SM0.0               // SM0.0总是1
ATCH        +0，10              // 重新激活静止线定时器
RETI                            // 中断程序14结束
```

H.12 S7-224与SIMOVERT电机驱动器之间的自由通信口通信接口

概述4

S7-224通过与一台SIMOVERT微型主电机驱动器通信来起动，停止电机，以及改变输出到电机的频率。通信是通过S7-200自由通信口模式进行，使用USS-5字协议。输入仿真器用来初始化发给电机驱动器的命令。

例图



硬件要求

这个程序假定使用者已正确地将电机和微型主电机驱动器接好线，并且所有的电机和微型主电机驱动器的参数已通过人工设定了。必须把微型主电机驱动器设置在遥控方式（P910=1），波特率19.2Kb（P92=7），地址1（P91=1）。

要求用一根带9针阳性插头的通信电缆接在S7-200的1，3，8端上，电缆另一端是无插头的，以便接到微型主电机驱动器的13，14，15端子上（1接15，3接13，8接14）。

程序结构

MAIN　主程序：监视用于自由通信口/PPI通信切换的RUN/TERM开关，寻找输入信号上升沿作业电机命令。

SBR0　设置自由通信口通信：首次扫描时设置自由通信口模式的参数。

SBR1　RUN子程序：设定电机恒速运转。

SBR2　RAMP子程序：设定电机变速运转。

SBR3　增加频率倍率的子程序：增加微型主电机驱动器的输出频率。

SBR4　降低频率倍率的子程序：降低微型主电机驱动器的输出频率。

SBR5　STOP子程序：停止电机。

SBR6　计算输出信息的BCC。

SBR7　发送信息，初始化发送定时器。

INT0　发送结束的中断处理程序，打开接收器。

INT1　发送超时的中断处理程序，再试发送，最多试发3次。

INT2　接收字符的中断程序，结束后进行有效性校验。

INT3　接收超时的中断处理程序，再试接收最多试收3次。

程序和注释

S7-214通过与一台SIMOVERT微型主电机驱动器通信来起动、停止电机，以及改变输出到电机的频率，通信是通过S7-200自由通信口，模式进行，使用USS 5字协议。输入仿真器用来初始化发给电机驱动器的命令，程序监视RUM/TERM开关，并选择相应的协议来设置自由通信口模式的控制字节（SMB 30）。程序监视如下电机命令的输入点：

I0.0	上升沿	使电机以上次命令的恒定频率运转。
I0.2	上升沿	使电机以上次命令的频率开始变频运转。 频率可用I0.6和I0.7升高或降低。
I0.4	上升沿	停止电机。
I0.5	电平	以1x或2x倍率改变频率。 I0.5： 0→1x，→2x。
I0.6	上升沿	以1x或2x频率增量（此例中为50）增加电机频率。
I0.7	上升沿	以1x或2x频率增量（此例中为50）降低电机频率。
I1.0	电平	电机旋转方向：I1.0：0→CW，1→CCW

程序检测并报告通信错误。首先对微型主电机驱动器的发送要计时，如果失败，允许再试发送，最多可试发送3次。然后，对来自微型主电机驱动器的接收亦要计时，在退出发送接收操作之前可重试多达3次。对来自微型主电机驱动器的响应信息要进行有效性校验（STX，LEN，ADR和BCC），任何被检测到的错误都要由QB0显示，直到下一次操作结束，QB0的显示含义如下：

0	无错误
1	非法的响应（除去坏的BCC）。
2	坏的BCC。
3	发送超时。
4	接收超时。

尽管这个示例程序只与一台微型主电机驱动器通信，可把它扩展用于另外的输入点，选择多站通信线路上的某一台微型主电机驱动器的地址，向它发送命令。另外，这个程序的基本通信结构还可用来发送别的信息给微型主电机驱动器，如监视电流，转矩等。

本程序长度为342个字。

```
// 这个程序是S7-200自由通信口通信接口与SIMOVERT微型主电机驱动器通信的示例
// 两者间通信用USS 5字协议。
// S7-200作为主设备，使用USS协议地址0，而微型主电机驱动器则是地址1。
// 为了正确地应用此例，你需要：
// 1台带输入仿真器的SIMATIC S7-212或S7-214。
// 1根RS 485电缆，其一端为阳型插头，另一端无插头，以接到微型主电机驱动器上。
// 1台微型主电机驱动器和电机
// 这个例子使用Herb Taylor，SE & A的示例单元（驱动器和电机）
// 你要确保正确地设置微型主电机驱动器的参数，并保证将它置于遥控方式
// ( P910=1 )，波特率设置正确 ( P092 )，从地址设置正确 ( P091 )。这个例子的波特率为19.2Kb，从
// 地址为1。

// 微型主电机驱动器使用的USS串行通信5字协议有如下14字节结构：

//      02          开始信息（STX）。
//      12          微型主电机驱动器的信息从AA到BCC长度为12字节。
//      AA          设备号地址（本例中驱动器为1）
//      PKE         参数控制高字节。
//      PKE         参数控制低字节
//      IND         陈列序引字的高字节
//      IND         陈列序引字的低字节
//      PWE         参数值和错误代码的高字节
//      PWE         参数值和错误代码的低字节
//      PZD1        控制/状态字高字节
//      PZD1        控制/状态字低字节
//      PZD2        主设定2点和返回值的高字节。
//      PZD2        主设定点和返回值的低字节
//      BCC         块校验和

// * * * * * * 注意例中对于开/关/速度、改变只使用数据字：PZD1和PZD2

//      程序结构：
//      MAIN        主程序。
//      SBR0        设定自由通信口通信。
//      SBR1        RUN子程序。
//      SBR2        RAMP子程序。
//      SBR3        增加频率倍率的子程序。
//      SBR4        降低频率倍率的子程序。
//      SBR5        STOP子程序。
//      SBR6        计算输出信息的BCC。
//      SBR7        发送消息，初始化发送定时器。
//      INT0        发送结束的中断处理程序，打开接收器。
//      INT1        发送超时的中断处理程序。
//      INT2        接收字符中断程序。
//      INT3        接收超时中断处理程序。
```

```
// 内存单元分配：
//      VB99              发送信息的和长度
//      VB100－VB113      发送缓冲区。
//      VB114－VB127      接收缓冲区。
//      VW200             缺省频率倍率，初始值=5461，频率=（5461/16384）×P094。
//      VW202             频率倍率增/减量，初始值=50。
//      VW204             发送再试次数，初始值为3次，每发一次减1。
//      VW206             接收再试次数，初始值为3次，每收一次减1。
//      VW208             信息中接收字符数，USS 10=14。
//      VB210             最后一次试操作的状态（也在输出QB0显示），其值含义如下：
//                        0  操作正确。
//                        1  来自微型主电机驱动器的非法响应（除去坏的BCC）。
//                        2  坏的BCC。
//                        3  发送超时。
//                        4  接收超时
//      VD211             接收缓冲区地址指针
//      VW215             累积接收信息为BCC，存于最低字节。
//      VW217－VB218      暂时存储区

// 输入点的功能：
//      I0.0        RUN使电机以当前频率倍率的值运转，方向由输入I1.0确定，仅在“STOP”后才影响
                    操作。上升沿操作。
//      I0.2        RAMP使电机以当前频率倍率的值运转，方向由输入I1.0确定，仅在“STOP”后才影
                    响操作。上升沿操作。
//      I0.4        STOP停止电机，允许后续的RUN/RAMP命令。
                    上升沿操作。
//      I0.5        指定倍率（1x或2x），频率倍率的增/减量，
                    电平操作：0→1x，1→2x。
//      I0.6        增加频率倍率，其值为VW202×倍率（I0.5）
                    若电机处于RAMP，则速度立即升高。
                    上升沿操作。
//      I0.7        减少频率倍率，其值为VW202×倍率（I0.5）
                    若电机处于RAMP，则速率立即下降。
                    上升沿操作。
//      I1.0        指定电机旋转方向。电平操作：0→CW，1→CCW。

// 首次扫描执行的程序
LD      SN0.1               // 首次扫描时SM0.1=1
CALL    0                   // 调子程序0
// 用RUN/TERM开关选择自由通信口/PPI通信。
LD      SM0.7               // RUN/TERM开关位置：1→RUN，0→TERM
=       SM30.0              // SM30.0=1为自由通信口协议，SM30.0=0为PPI协议。

// 检查命令
LD      I0.4                // 若STOP（停止电机）命令，且
```

```
EU                              // 上升沿，则
CALL    5                       // 往微型主电机驱动器发送信息
S       M0.0，1                 // 使后续的RUN或RAMP命令有效（M0.1=1）。
R       M0.1，1                 // M0.1=0
LD      I0.0                    // 若RUN（运行电机）命令，且
EU                              // 上升沿
A       M0.0                    // M0.0=1，则
CALL    1                       // 往微型主电机驱动器发送信息
R       M0.0，02                // 使后续的RUN命令无效（M0.0=0，M0.1=0）
LD      I0.2                    // 若RAMP命令，且
EU                              // 上升沿
A       M0.0                    // M0.0=1，则
R       M0.0，1                 // 使后续RUN命令无效（M0.0=0）
CALL    2                       // 往微型主电机驱动器发信息
S       M0.1，1                 // 显示现在状态（M0.1=1）

LD      I0.6                    // 若增加频率，且
EU                              // 上升沿，则
CALL    3                       // + + 速度。

LD      I0.7                    // 若减少频率，且
EU                              // 上升沿，则
CALL    4                       // 一速度。
MEND                            // 主程序结束。

// * * * * * * 子 程 序 0 * * * * * *

// 初始化子程序——仅在第一次扫描时执行
SBR     0                       // 初始化XMT缓冲区，设置通信参数

MOVB    16#44，SMB30            // 19.2Kb，每字符8位，偶校验
MOVB    16#0E，VB99             // XMT长度
MOVB    16#02，VB100            // STX
MOVB    16#0C，VB101            // LEN
MOVB    16#01，VB102            // ADR（微型主电机驱动器地址为1）
FILL    0，VW103，5             // 清除所有5个数据字（VW103至VW111）以后还可改变
MOVW    16#1500，VW200          // 1/3最大的频率倍率
MOVW    16#32，VW202            // 频率保率以50增/减
MOVD    16#00030003，VD204      // 设定发送和接收重试次数为3
MOVB    0，VB210                // 清除操作状态指示
MOVB    0，QB0                  // QB0=0
MOVB    0，VB219                // S7-200地址
S       M0.0，1                 // 允许RUN或RAMP或RAMP（方向改变）
ENI                             // 允许用户中断
RET                             // 子程序0结束
```

```
// * * * * * * 子 程 序 1 * * * * * *

// 运行电机子程序，旋转方向取决于输入I1.0的状态
SBR       1                    // 运行电机

MOVB      16#05，VB109         // 为CW设置STW
MOVB      16#7F，VB110         //
MOVW      VW202，VW111         // 设定频率

LD        I1.0                 // 设定旋转方向，若I1.0=0，则为CW；若I1.0=1，则为CCW
=         V109.3               //
NOT
=         V109.4

LD        SM0.0                // SM0.0总是1
CALL      6                    // 计算BCC
CALL      7                    // 初始化。XMT及XMT定时器

RET                            // 子程序1结束

// * * * * * * 子 程 序 2 * * * * * *

// RAMP电机的处理子程序，电机运转时，根据输入I0.2的上升沿脉冲，对电机频率进行+/-RAMP修正，
// 电机旋转方向取决于输入I1.0的状态。
// 子程序2
SBR       2                    // 运行电机

MOVB      16#04，1VB217        // 设置命令字节
LD        V109.3               // 保存以前的电机旋转方向
=         V217.3               //
LD        V109.4               //
=         V217.4               //
LD        SM0.0                // SM0.0总是1
MOVB      VB217，VB109         // 为RUN设置STW
MOVB      16#7F，VB110         //
MOVW      VW200，VW111         // 设置频率

LD        0.1                  // 检查状态，看是否允许改变电机旋转方向
JMP       0                    // 只有在STOP（停止）后才允许，否则不允许
                               // 保存以前的电机旋转方向
LD        I1.0                 // 根据输入I1.0的状态，设置电机旋转方向
=         V109.3               //
NOT
=         V109.4

LBL       0
LD        SM0.0
```

```
CALL   6                          // 计算BCC
CALL   7                          // 初始化XMT及XMT定时器

RET                               // 子程序2结束

// * * * * * * 子 程 序 3 * * * * * *

// 增加电机频率的处理子程序。频率增量存于VW202。若I0.5为1，则倍增；若上溢出，则设为
        32767。
// 子程序3
SBR    3                          // 增加频率

+I     VW202，VW200               // + 因子（增加）
LD     I0.5                       // 倍增吗
+I     VW202，VW200
LDW>=  VW200，16384               // 判是否上溢出（大于最大值16384）？
MOVW   16384，VW200               // 若上溢出，则设为最大值（16384）

LD     M0.1                       // 若M0.1=1，则
CALL   2                          // 发送信息，增加频率

RET                               // 子程序3结束

// * * * * * * 子 程 序 4 * * * * * *
// 降低电机频率子程序，频率减量存于VW202，若I0.5为1（ON），则倍减，若下溢出，则为0
// 子程序4
SBR    4                          // 降低频率

-I     VW202，VW200               // 一因子（降低）
LD     I0.5                       // 倍减?
-I     VW202，VW200
LD     SM1.2                      // 判是否下溢出（小于0）？
MOVW   0，VW200                   // 若下溢出，则设为0

LD     M0.1                       // 若M0.1=1，则
CALL   2                          // 发送信息，降低频率

RET                               // 子程序4结束

// * * * * * * 子 程 序 5 * * * * * *
// 停止电机的子程序
SBR    5                          // 停止电机

MOVB   16#0C，VB109               // 为STOP设置STW
MOVB   16#7E，VB110               //

CALL   6                          // 计算BCC
CALL   7                          // 初始化XMT及XMT定时器
```

```
RET                                    // 子程序5结束

// * * * * * * 子 程 序 6 * * * * * *
// 用XOR（异或）计算信息块的检查和，并存入缓冲区。
SBR       6                            // 计算USS 5字的BCC
// 十六进制信息为：02，0C，ADR，BYTE1，…，BYTE10，BCC
// 本程序入口时，AC1指向信息LEN字节（0C）的位置。
// 本程序出口时，AC1指向BCC字节位置。
// 将BCC存入信息缓冲区。
// AC2含有BCC。

MOVD      &VB101，AC1                  // 设置缓冲区地址指针
MOVD      16#0E，AC2                   // 2×OR12，信息的头两字节

FOR       AC3，1，11                   // 计算剩余11个字符的BBC。
XORW      *AC1，AC2
INCD      AC1                          // 地址指针加1
NEXT

INCD      AC1                          // 指针加1，指向BCC位置
MOVB      AC2，*AC1                    // 把BCC存到信息缓冲区中。
RET                                    // 子程序6结束。

// 初始化XMT及XMT定时器
SBR       7                            // 初始化XMT，捕捉XMT中断

XMT       VB99，0                      // 发送
ATCH      0，9                         // 捕捉XMT（发送）中断（事件9），并调用中断程序0( INT 0 )

MOVB      255，SMB34                   // 设置XMT定时器定255ms，实际只用约7ms（19.2Kb）
ATCH      1，10                        // 捕捉XMT定时器中断（事件10），并调用中断程序1( INT 1 )
RET                                    // 子程序7结束

// XMT（发送）中断处理，关掉XMT定时器，起动接收从设备响应
INT       0                            // 中断程序0，XMT中断处理

DTCH      10                           // 退出XMT定时器
DTCH      9                            // 中止XMT事件

MOVW      3，VW204                     // 刷新XMT重试次数（3次）
MOVW      14，VW208                    // 响应信息中接收的字符数（14个）
MOVW      0，VW215                     // 清除BCC累加器
MOVD      &VB114，VD211                // 设置接收缓冲区指针

ATCH      2，8                         // 捕捉RCV（接收）中断，并调用中断程序2（INT 2）
ATCH      3，10                        // 捕捉接收定时器中断（事件10），并调用中断程序3 ( INT 3 )

RETI                                   // 中断程序0结束
```

```
// 中断程序1
// 若XMT（发送）定时器时间到，这段程序取得控制权，XMT操作会重试，直到可重试次数减到0
INT       1                    // 定时器中断0处理－发送

DTCH      9                    // 停止XMT（发送）
DTCH      10                   // 退出定时器

DECW      VW204                // 重试次数减1，若为0，则
LD        SM1.0                // SM1.0=1
MOVB      3，VB210             //
VOVB      3，QB0               // 用QB0指示发送超时
MOVW      3，VW204             // 刷新发送重试计数（3次）
S         M0.01，1             // 使RUN，RAMP有效（M0.0=1）
CRETI                          // 条件返回
                               // 重试
XMT       VB99，0              // 发送
ATCH      0，9                 // 捕捉XMT（发送）中断（事件9），并调用中断程序0( INT 0 )

MOVB      255，SMB34           // 设置XMT (发送)定时器为255ms，实际只用约7ms ( 19.2Kb )
ATCH      1，10                // 捕捉定时器中断（事件10），并调用中断程序1( INT 1 )

RETI                           // 中断程序1结束

// 中断程序2
// 这段处理程序计算接收字符数，并校验错误
// 若检测到错误，则再试接收，直至可重试次数减到0
INT       2                    // 接收字符处理

MOVB      SMB2，AC0            // 得到接收字符
XORW      AC0，VW215           // 累积BCC，即用XOR（异或）计算BCC

MOVB      AC0，*VD211          // 把接收到的字符送入缓冲区
INCD      VD211                // 缓冲区指针加1
DECW      VW208                // 有待接收的字符总数减1
LDN       SM1.0                // 检验是否结束
CRETI

NOT
DTCH      10                   // 退出接收定时器
DTCH      8                    // 关掉接收

AB=       0，VB216             // 检验已算好的BCC是否为0
NOT

MOVB      2，VB210             // 坏的BCC操作码
MOVB      2，QB0
JMP       0
```

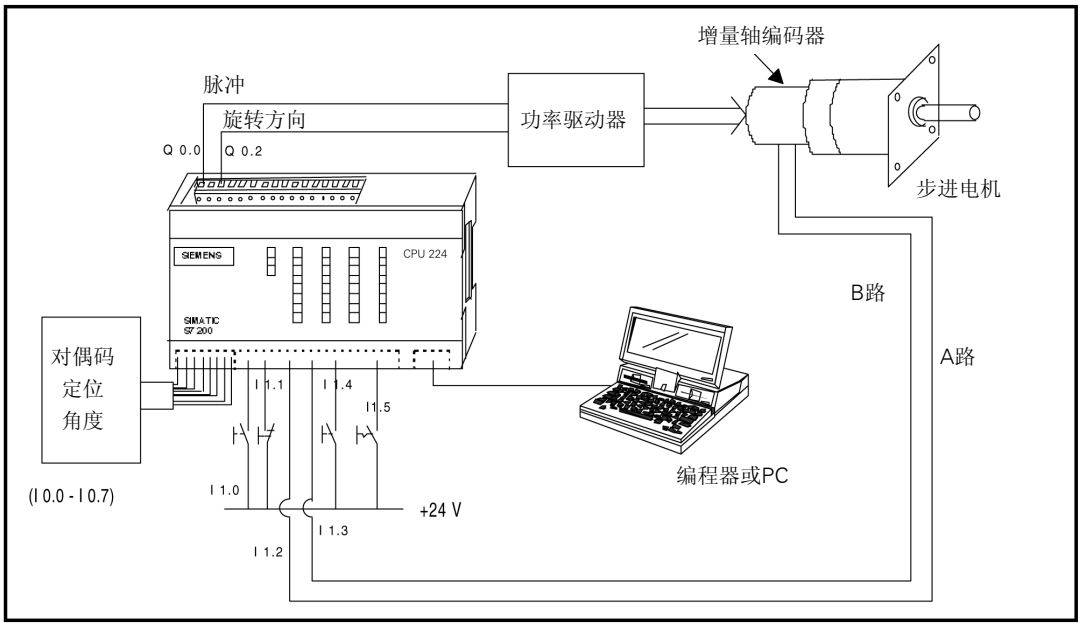
```
                                    // BCC正确，检验信息的其它部分
LDB=    VB114，16#02                // STX第一个字符吗？
AB=     VB115，16#0C                // 长度=12吗？
AB=     VB116，VB102                // 将信息发往同一从设备吗？
// * * * *

// 任何其它校验都会因相应响应而经过这一段

// * * * *
MOVB    0，VB210                    // 操作正确
MOVB    0，QB0
JMP     0

LD      SM0.0
MOVB    1，VB210                    // 信息中有不对的地方
MOVB    1,QB0

LBL     0
MOVW    3，VW206                    // 刷新接收可重试次数（3次）

RETI                                // 中断程序2结束

// 中断程序3
// 若响应的接收时间已到，这段程序取得控制权，再试发送信息，再试一次接收。
// 在超时时操作重试，直至可重试次数减为0
INT     3                           // 定时器中断0处理－－接收

DTCH    8                           // 关掉接收中断
DTCH    10                          // 退出接收定时器

DECW    VW206                       // 检查可重试次数、重试次数减1，若为0，则
LD      SM1.0                       // SM1.0=1
MOVB    4，VB210                    //
MOVB    4，QB0                      // 用QB0指示接收超时
MOVW    3，VW206                    // 刷新接收重试次数
S       M0.0，1                     // 使RUM，RAMP有效
CRETI                               // 条件返回

NOT
MOVD    &VB114，VD211               // 设置接收缓冲区的指针
MOVW    0，VW215                    // 清除BCC累加器
                                    // 再重试发送
XMT     VB99，0                     // 发送
ATCH    VB0，9                      // 捕捉XMT（发送）中断，并调用中断程序0

MOVB    255，SMB34                  //  设定XMT（发送）定时器为255ms
ATCH    1，10                       //  捕捉定时器中断（事件10），并调用中断程序1（INT 1）

RETI                                // 中断程序3结束
```

H.13 用S7-200 CPU 224 DC/DC/DC进行定位控制，并具有位置监视和位置校正

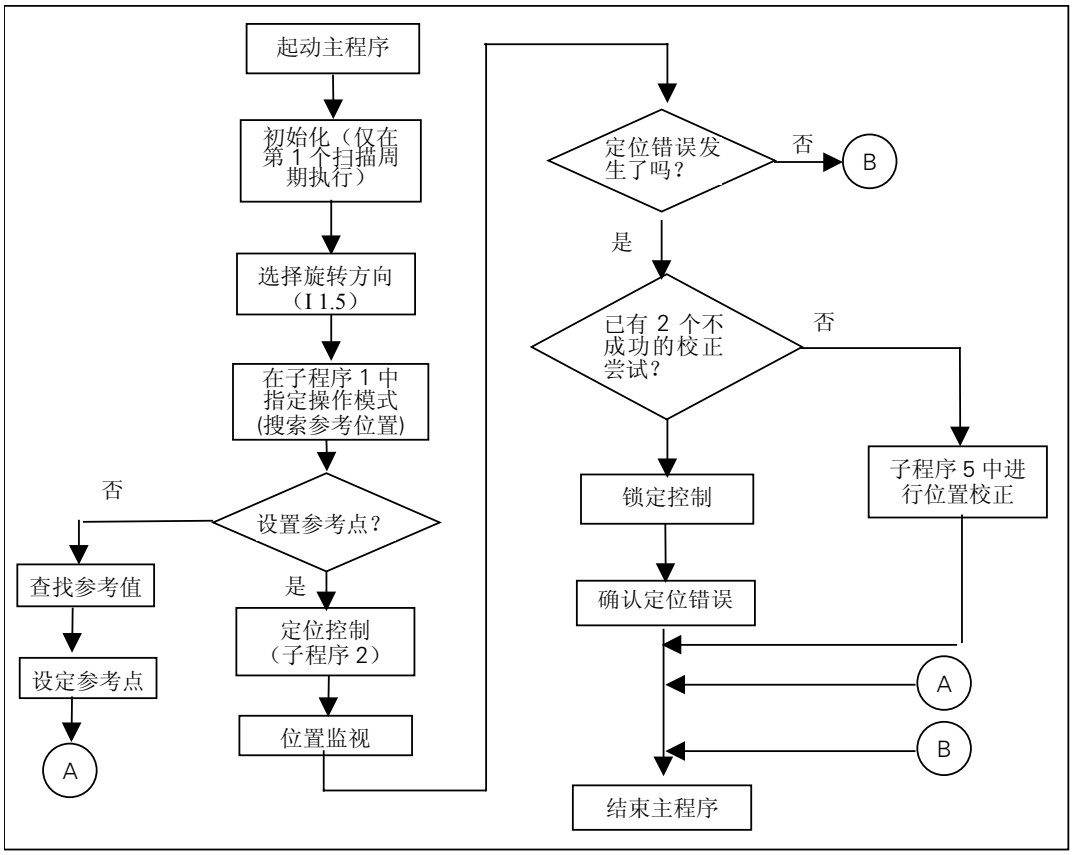
概述

本例是由增量传感器进行位置监视。为了求出传感器信号，将该信号作为CPU 224中的高速计数器的输入，这样，就可检测出位置误差。例如，当起－停频率超出时，通过步数丢失可以检测到位置错误。一旦检测出位置误差，就以较低频率进行位置校正。

例图



程序框图



程序和注解

一、初始化

在程序的第一个扫描周期（SM0.1=1）设置重要的参数。此外，高速计数器HSC2由外部复位并初始化为A/B计数器。HSC2对检测定位的增量轴编码器信号计数。传感器的A路和B路信号分别作为CPU输入端I 1.2和I 1.3的输入。

旋转方向的选择、按钮锁定、操作模式的选择及定位的过程都和例23相同（请参考此例概述）。与23不同的是，由增量传感器进行定位监视，在输出脉冲结束之后，等待T1时间，以便使连接电机和传感器的轴连接器的扭转振动消失。

二、实际值和设定值的比较

T1到时后，子程序4对实际值和设定值进行比较。如果轴的位置在设定位置的±2步范围内，定位就是正确的。如果实际位置在此目标范围之外，当超过起停频率时，那就会造成电机失步这种情况的发生，此时，Q1.1就会输出一个警告信号。

三、位置的校正

若定位错误被检测出来，则起动第二等待定时器T2。此后，根据设定值和实际值之间的差值计算出校正的步数。当校正时，电机频率低于起停频率，以防新的步数丢失。

四、校正取消

如果在两次校正尝试之后还不能达到设定位置，为安全起见，控制将被锁定（M 0.2=1）。只有按下确认按钮I 1.4之后，控制才被打开，然后，进行另一个参考点的检测。

五、信号清单：

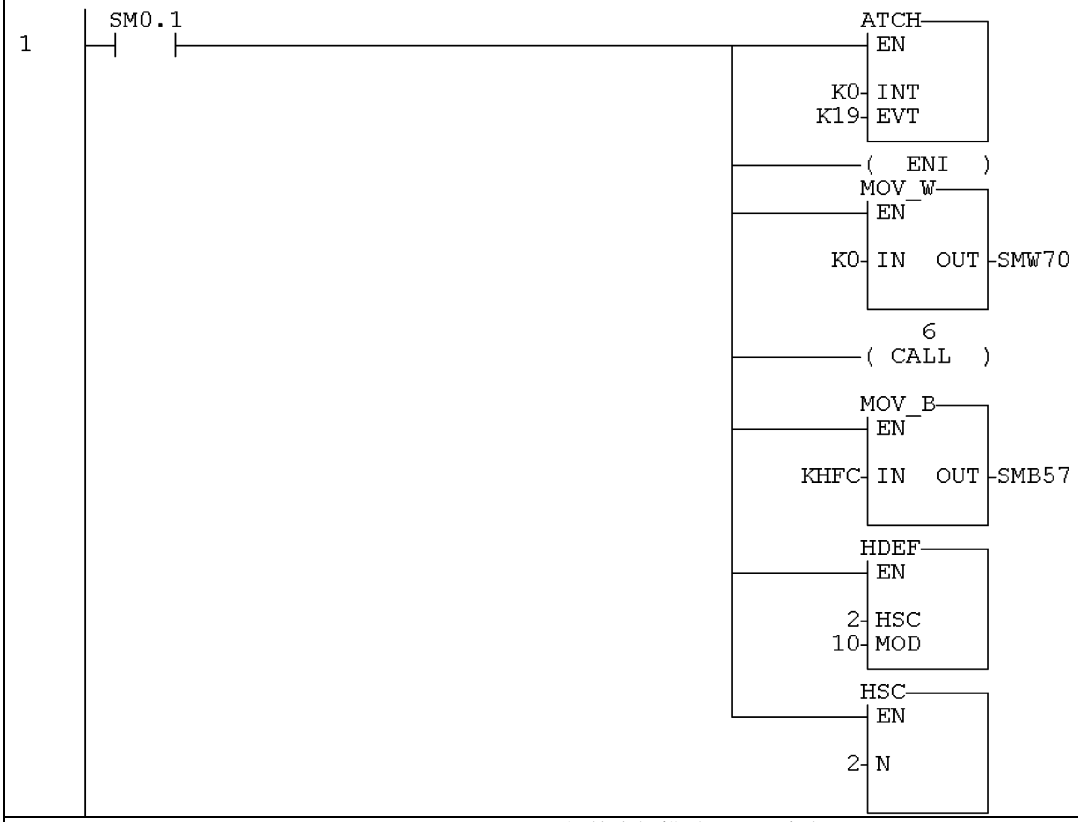
输入：	I 0.0——I 0.7	以度为单位的定位角（对偶码）
	I 1.0	“电机启动”按钮“START”
	I 1.1	“电机停止”按钮“STOP”
	I 1.2	传感器信号，A路
	I 1.3	传感器信号，B路
	I 1.4	“设置/取消参考点”按钮（确认开关）
	I 1.5	选择旋转方向的开关
输出：	Q 0.0	脉冲输出
	Q 0.2	旋转方向信号
	Q 1.0	操作模式的显示
	Q 1.1	定位错误的显示
标志位：	M 0.1	电机运转标志位
	M 0.2	锁定标志位
	M 0.3	参考点标志位
	M 0.4	完成第一次定位标志
	M 1.1	T1等待时间到标志位
	MD8,MD12	计算步数时的辅助内存单元
	M 20.0	脉冲输出结束标志位
	MW 25	错误定位计数器
精度：	AC 0	允许偏差的下限
	AC 1	允许偏差的上限
	AC 2	设定值
	AC 3	辅助寄存器

本程序的长度为310个字。

LAD	STL(IEC)

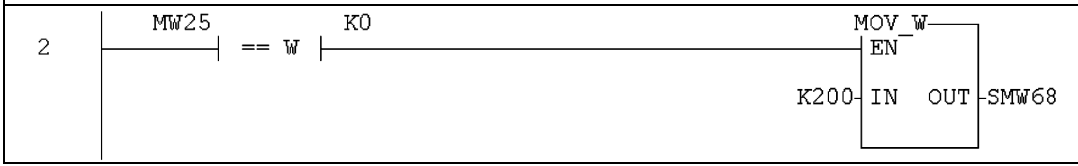
// 主程序

// 初始化



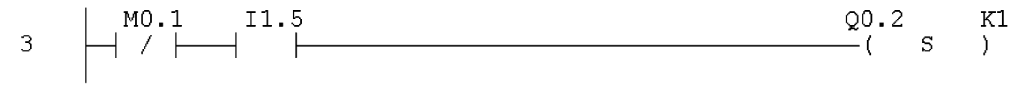

```
LD      SM0.1          //  仅首次扫描时SM0.1才为1
ATCH    0，19          //  把中断程序0分配给中断事件19（PLS0脉冲串终
ENI                    止）
MOVW    0，SMW70       //  允许中断
CALL    6              //  脉冲宽度=0（脉宽调制）
                       //  在子程序6中进行初始化
// 高速记数器HSC2
MOVB    16#FC，SMB57   // 置HSC2的控制字节
HDEF    2，10          // 置HSC2模式10：外部复位，A/B计数器
HSC     2              // 激活HSC2
```

// 定位速度



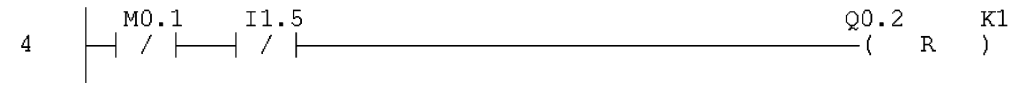
```
LDW=    MW25，0          // 若没有错误定位
MOVW    200，SMW68       // 则高速定位（T=200μs）
```

// 设置逆时针旋转



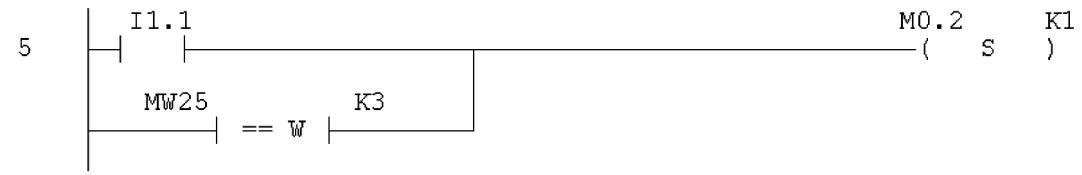

```
LDN     M0.1             // 若电机停止（M0.1=0）
A       I1.5             // 且按下旋转方向开关（I1.5=1）
S       Q0.2，1          // 则逆时针旋转（Q0.2=0）
```

// 设置顺时针旋转



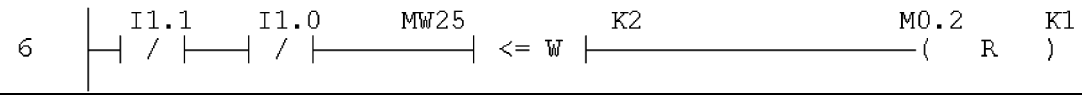

```
LDN     M0.1             // 若电机停止（M0.1=0）
AN      I1.5             // 且未按旋转方向开关（I1.5=0）
R       Q0.2，1          // 则顺时针旋转（Q0.2=0）
```

// 锁定



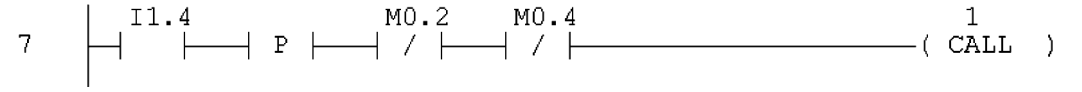

```
LD      I1.1             // 若按“电机停止（stop）”钮
OW=     MW25，3          // 或有3个错误定位
S       M0.2，1          // 则激活锁定（M0.2=1）
```

// 解除锁定



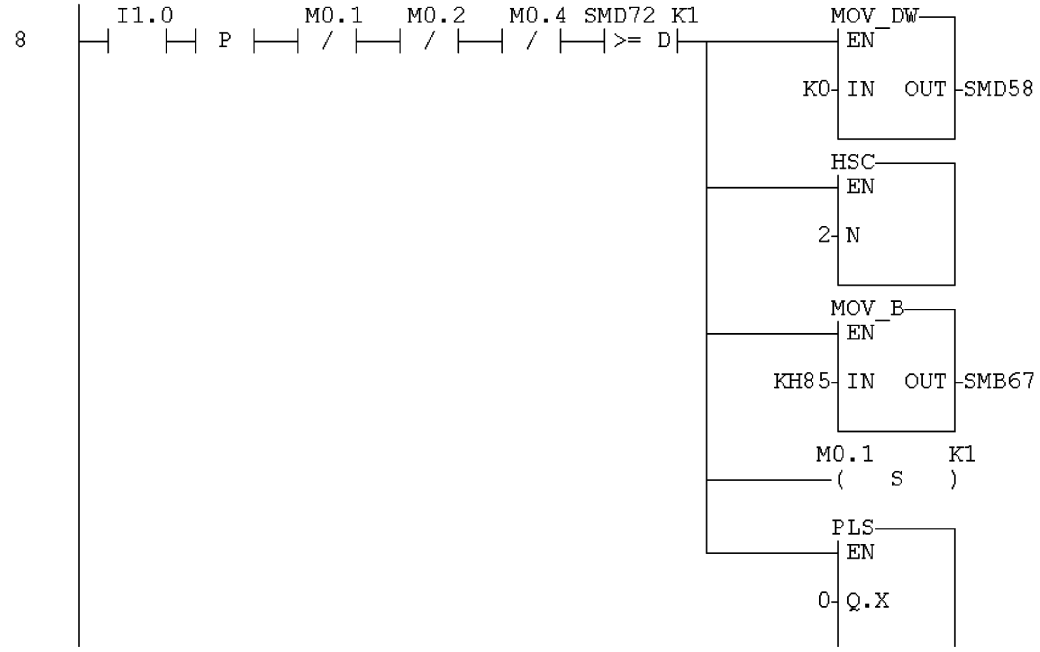

```
LDN     I1.1             // 若未按电机停止钮“STOP”（I1.1=0）
AN      I1.0             // 且未按电机起动钮“START”（I1.0=0）
AW< =   MW25，2          // 且< 2个错误定位
R       M0.2，1          // 则解除锁定
```

// 指定操作模式（检索参考/定位）



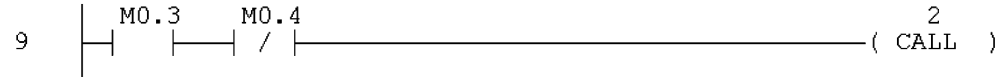
LD	I1 .4	// 若按下“设置/删除参考点”按针（I1.4=1）
EU		// 且上向沿
AN	M0.2	// 且无锁定（M0.2=0）
AN	M0.4	// 且无定位（M0.4=0）
CALL	1	// 则调用子程序1指定操作模式

// 起动电机



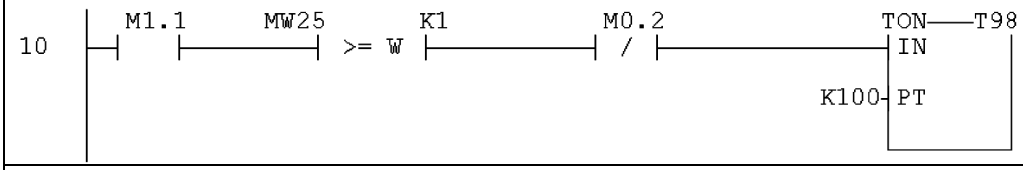
LD	I1.0	// 若按“电机起动”接钮“START”（I1.0=1）
EU		// 且上升沿
AN	M0.1	// 且电机在停止状态（M0.1=0）
AN	M0.2	// 且无联锁（M0.2=0）
AN	M0.4	// 且无定位控制（M0.4=0）
AD > =	SMD72，1	// 且步数 > 1，则
MOVD	0，SMD58	// 置HSC2的起始值为0
HSC	2	// 起动HSC2
MOVB	16#85，SMB67	// 激活脉冲输出功能PTO0（即置PTO0）的控制位）
S	M0.1，1	// “电机运转”标志置位（M0.1=1）
PLS	0	// 起动输出端Q0.0输出脉冲

// 定位

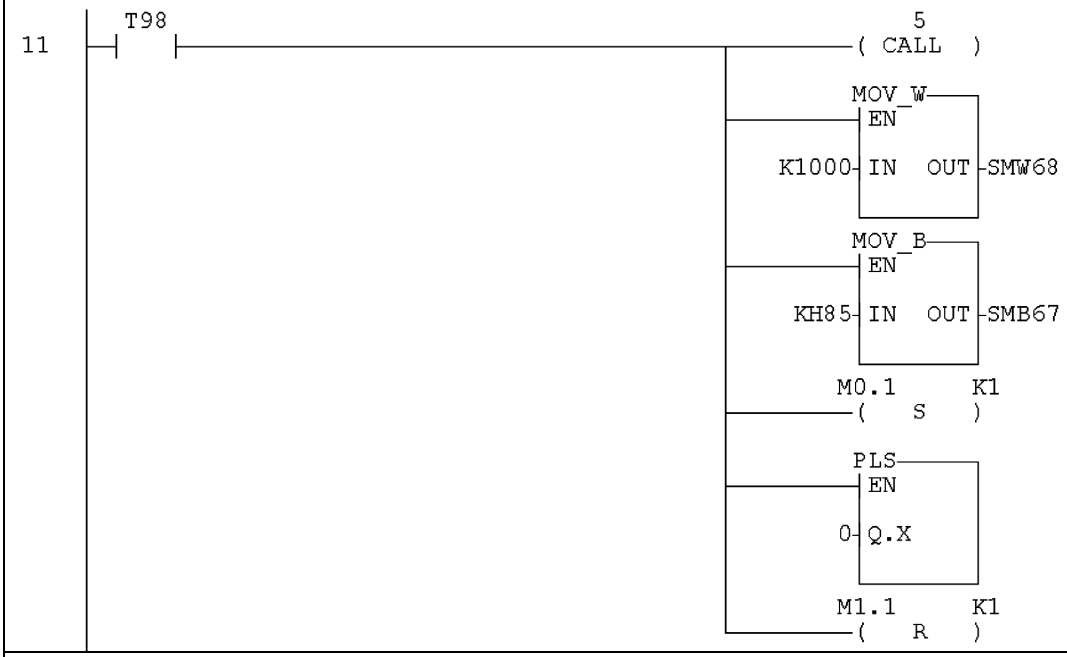


LD	M0.3	// 若“定位”操作模式（M0.3=1）
AN	M0.4	// 且尚未定位（M0.4=0）
CALL	2	// 则调用子程序2定位

// 位置校正



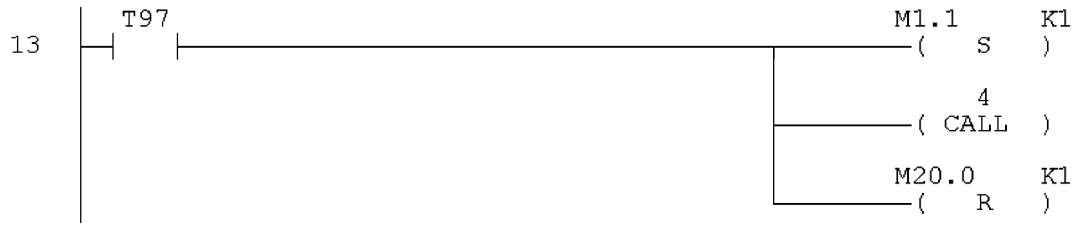
LD	M1.1	// 若T1到时（M1.1=1）
AW > =	MW25,1	// 且检测出错误定位（MW25≥1）
AN	M0.2	// 且未激活锁定（M0.2=0）
TON	T98,100	// 则起动等待定时器T2（1s）



LD	T98	// 若T2到时（T98=1），则
CALL	5	// 调用子程序5计算校正步数
MOVW	1000,SMW68	// 用1kHz 进行位置校正
MOVB	16#85,SMB67	// 激活PTO0（即置PTO0的控制位）
S	M0.1，1	// 设置“电机运转”标志（M01=1）
PLS	0	// 起动Q 0.0输出脉冲
R	M1.1，1	// T1复位（M1.1=0）

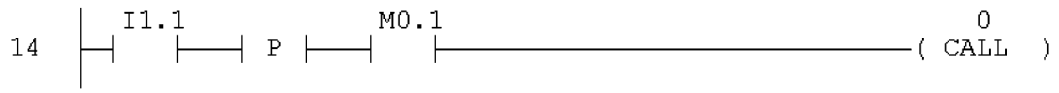
// 位置监视

```
LD      M20.0       // 若脉冲输出结束（M20.0=1
AN      M0.2        // 且未激活锁定（M0.2=0）
TON     T97，50     // 则起动等待定时器T1（500ms）
```



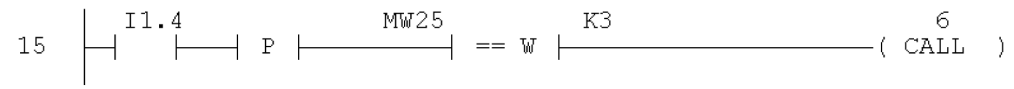

```
LD      T97         // 若T1到时（T97=1），则
S       M 1.1，1    // T1标志置位（M1.1=1）
CALL    4           // 在子程序4中调用位置监视
R       M20.0，1    // 脉冲输出结束标志位复位（M20.0=0）
```

// 电机停止




```
LD      I1.1        // 若按下“电机停止”钮“STOP”（I1.1=1）
EU                  // 且上升沿
A       M0.1        // 且电机在运转 （M0.1=1）
CALL    0           // 则调用子程序0停止电机
```

// 3次定位失败之后的错误确认




```
LD      I1.4        // 若按下确认钮（I1.4=1）
EEU                 // 且上升沿
AW=     MW25，3     // 且有3次定位失败
CALL    6           // 则调用子程序6返回初始状态
```

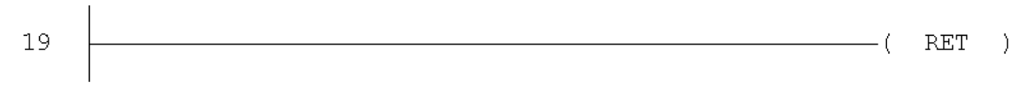

```
MEND                // 主程序结束
```

子程序

// ***** **

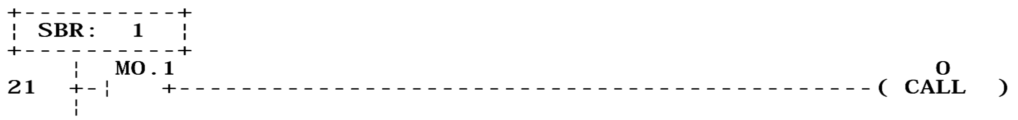
// 子程序0“停止电机”


```
SBR     0               // 子程序0
LD      SM0.0           // SM0.0总是1
MOVB    16#CB，SMB67    // 激活脉冲宽度调制（即置PTO0的控制位）
PLS     0               // Q0.0停止输出脉冲
R       M0.1，1         // 对“电机运转”标志复位（M0.1=0）
```



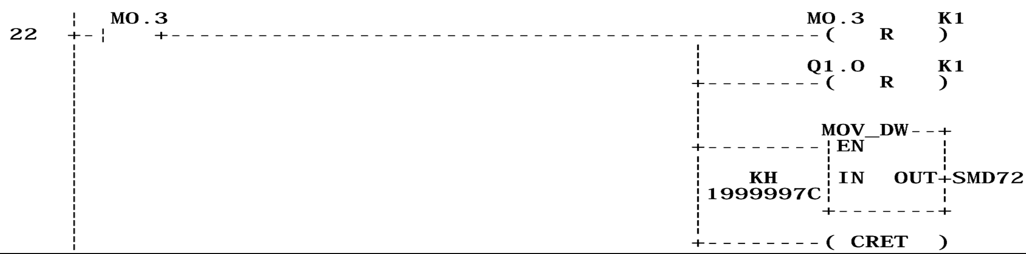

```
RET                     // 子程序0结束
```

// 子程序1“指定操作模式”



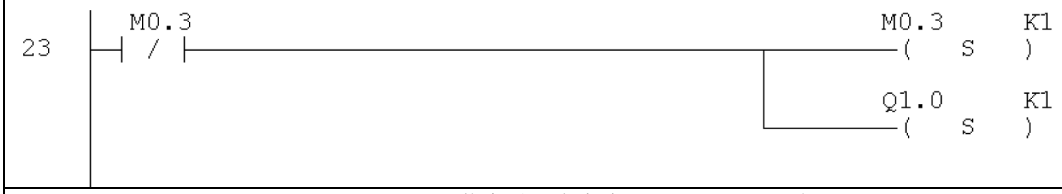

```
SBR     1               // 子程序1
LD      M0.1            // 若电机运转（M0.1=1）
CALL    0               // 则调用子程序0停止电机
```

// 开始检索参考点

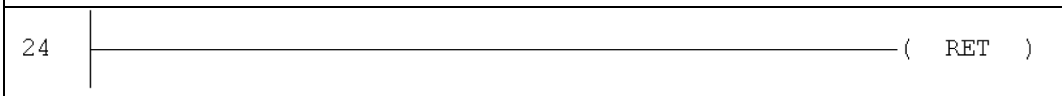


```
LD      M0.3          // 若“定位”标志激活（0.3=1），则
R       M0.3，1       // 参考点标志复位（M0.3=0）
R       Q1.0，1       // 删除“定位激活”信号(Q1.0=0)
MOVD    16#1999997C，  // 为新的参考点设置最大脉冲数
CRET    SMD72         // 条件返回
```

// 请求定位控制

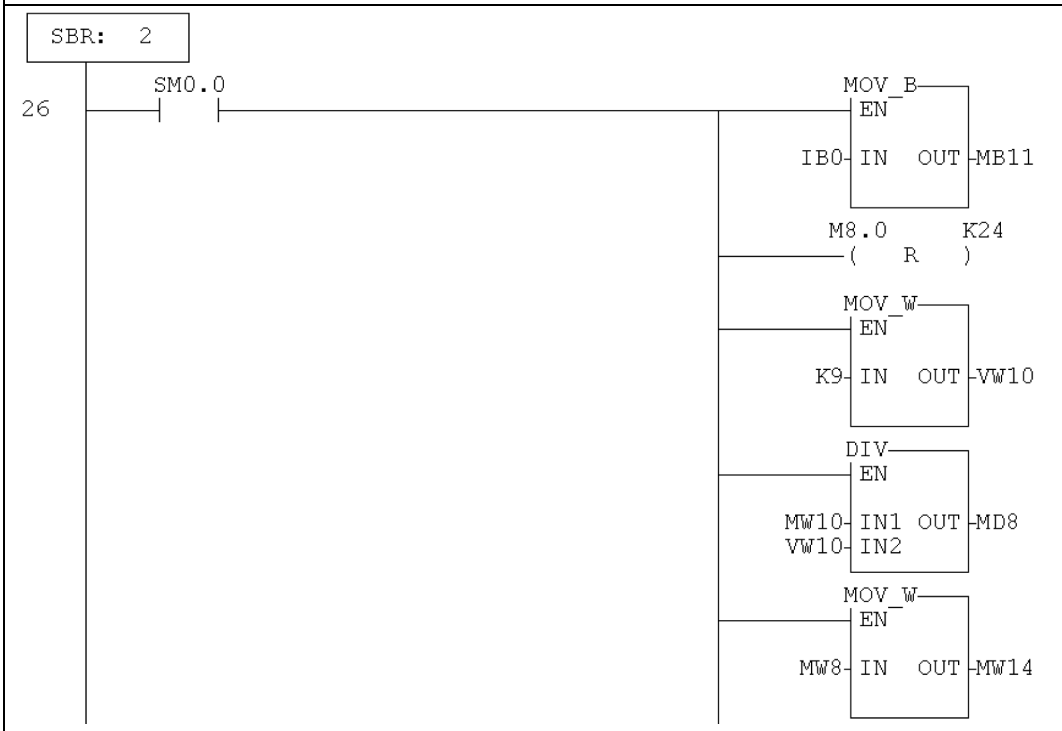


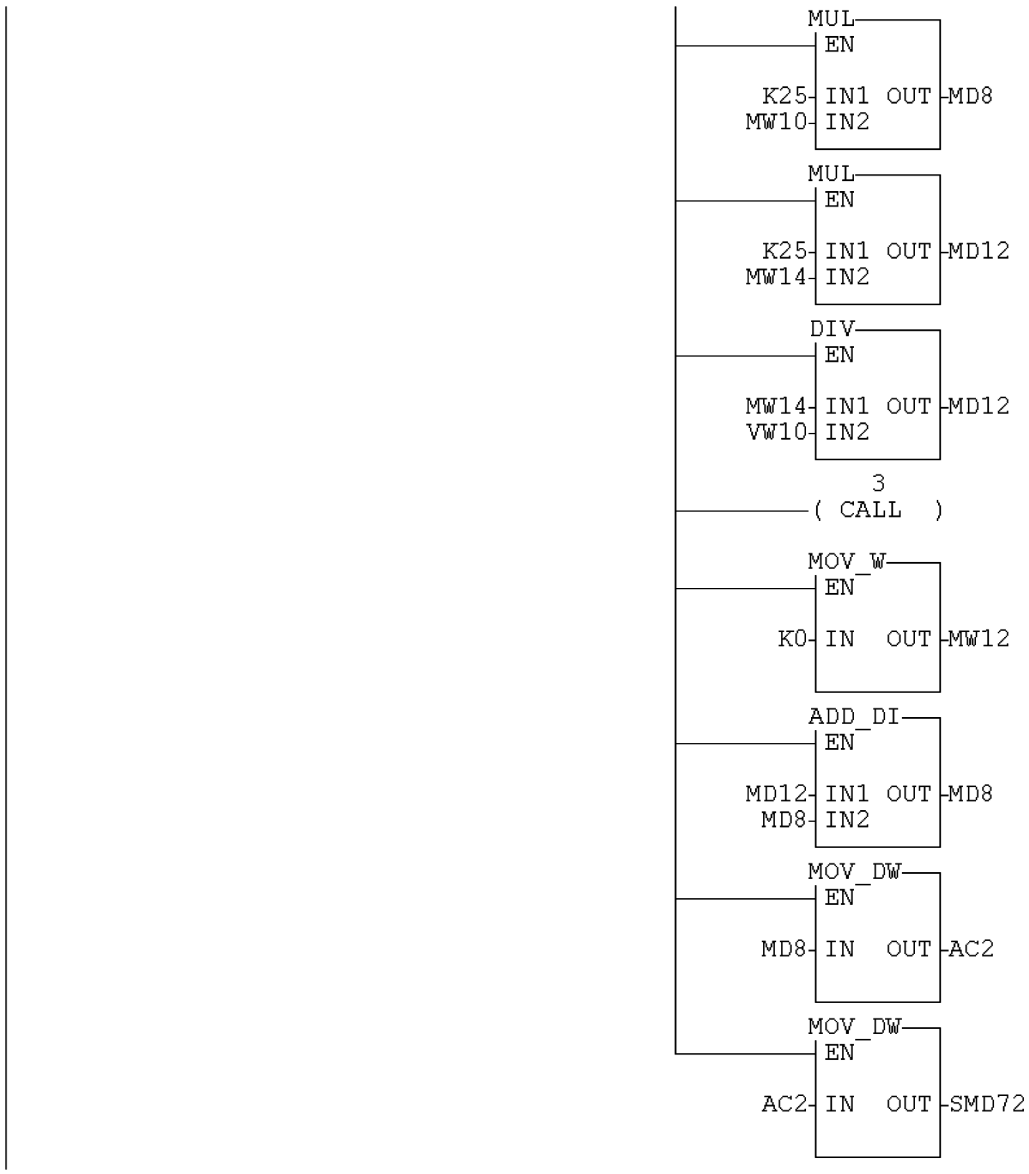

```
LDN     M0.3          // 若未设置参考点（M0.3=0），则
S       M0.3，1       // 参考点标志位置位（M0.3=1）
S       Q1.0，1       // 输出“定位激活”信号（Q 1.0=1）
```




```
RET                   // 子程序1结束
```

// 子程序2“计算步数，允许偏差极限”

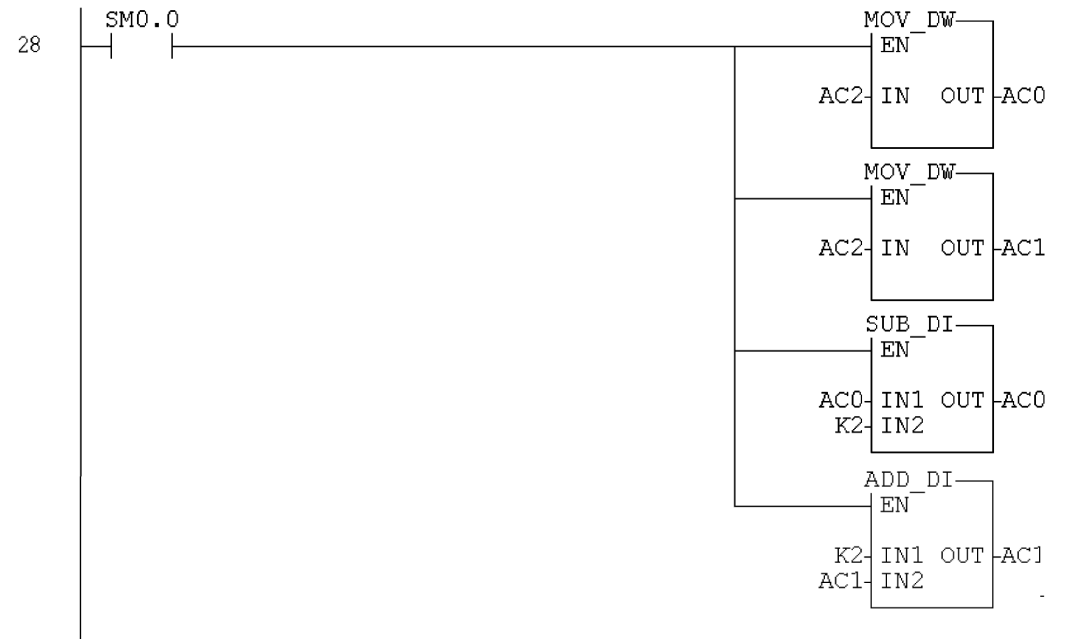





```
SBR2                    // 子程序2
LD      SM0.0           // SM0.0总为1
MOVB    IB0，MB11       // 把预设定位角从输入字节IB0复制到MD8的最低有效字节MB11
R       M8.0，24        // MB8至MB10请零
MOVW    K9，VW10        // 把9置入VW10
DIV     VW10，MD8       // 角度/9=q1+r1(q1=商，r1=余数)
MOVW    MW8，MW14       // 把r1（余数）存入MD12
MUL     25，MD8         // q1×25=MD8
MUL     25，MD12        // r1×25=MD12
```

```
DIV     VW10，MD12      // r1×25/9=q2+r2（q2=商，r2=余数）
CALL    3               // 在子程序3中4舍5入步数
MOVW    0，MW12         // 删除r2
+D      MD12，MD8       // 把步数写入MD8（MD12+MD8=MD8）
MOVD    MD8，AC2        // 步数=设定值（把步数MD8存入累加寄存器AC2）
MOVD    AC2，SMD72      // 把步数存入SMD72
```


```
LD      I1.5            // 若按下逆时针旋转钮（I1.5=1），则
INVD    AC2             // AC2取反
INCD    AC2             // AC2+1=AC2
```




```
LD      SM0.0           // SM0.0总为1
MOVD    AC 2，AC0       // AC2存入AC0
MOVD    AC2，AC1        // AC2存入AC1
-D      2，AC0          // 最低限值（AC2-2=AC0）
+D      2，AC1          // 最高限值（AC2+2=AC1）
```

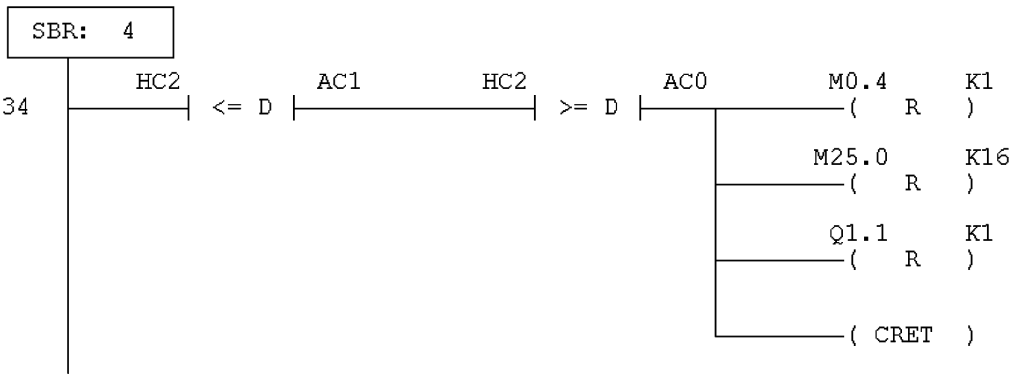

```
RET                                    // 子程序2结束
```

// 子程序3“4舍5入步数”

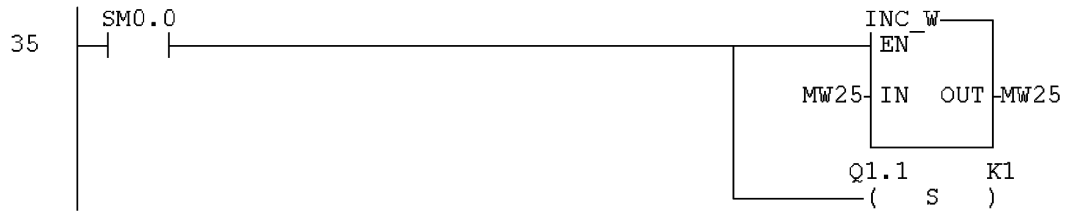

```
SBR       3                            // 子程序3
LDW>=     MW12，5                      // 如果r2>=5/9
INCW      MW14                         // 则步数增加1
```


```
RET                                    // 子程序3结束
```

// 子程序4“位置监视”



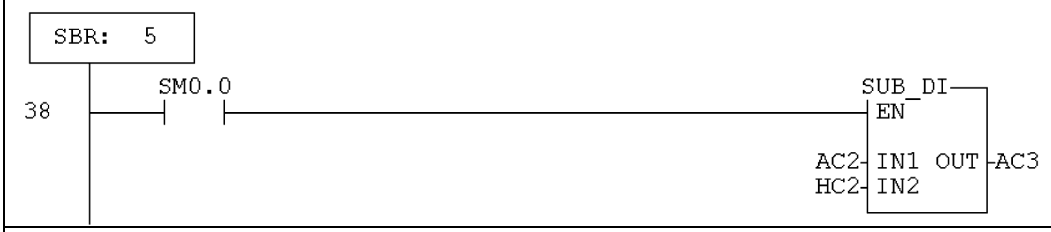

```
SBR4                                   // 子程序4
LDD<=     HC2，AC1
AD> =     HC2，AC0                     // 若当前值在限值范围内（即ACO≤HC2≤AC1），则
R         M0.4，1                      // 第1个位置标志复位（M0.4=0）
R         M 25.0，16                   // 错误定位计数器复位
R         Q1.1，1                      // 删除错误定位显示（Q1.1=0）
CRET                                   // 条件返回，否则
```



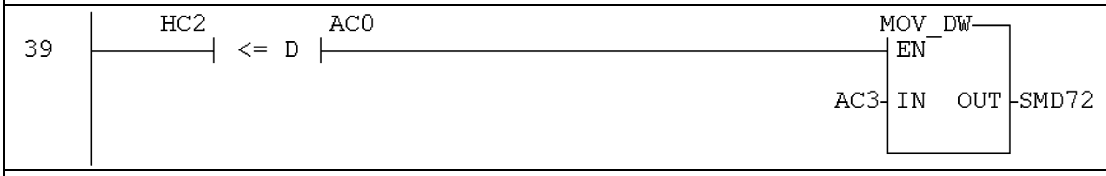
LD	SM0.0	// 错误定位
INCW	MW25	// 错误定位计数器加1
S	Q1.1，1	// 显示错误定位

RET		// 子程序4结束

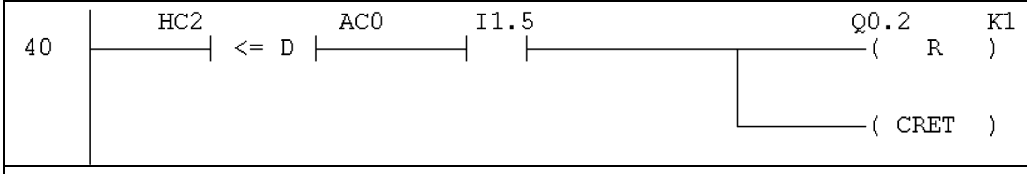
// 子程序5“计算校正步数”



SBR	5	// 子程序5
LD	SM0.0	// SM0.0总为1
MOVD	AC2，AC3	// AC2存入AC3
-D	HC2，AC3	// 设定值－实际值=AC3

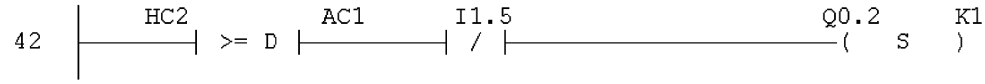


LDD<=	HC2，AC0	// 若实际值<设定值AC0
MOVD	AC3，SMD72	// 则步数=设定值－实际值

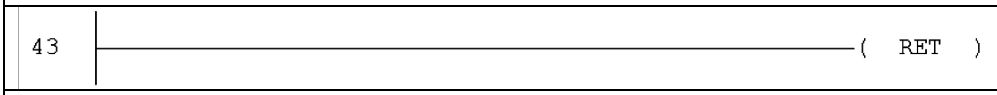


LDD<=	HC2，AC0	// 若实际值<设定值AC0
A	I1.5	// 且逆时针方向旋转（I1.5=1）
R	Q0.2，1	// 则顺时针方向校正
CRET		// 条件返回

```
LDD>=    HC2，AC1        // 若实际值>设定值AC1，则
INVD     AC3             // AC3取反
INCD     AC3             // AC3+1=AC3
MOVD     AC3，SMD72      // 步数=实际值－设定值
```

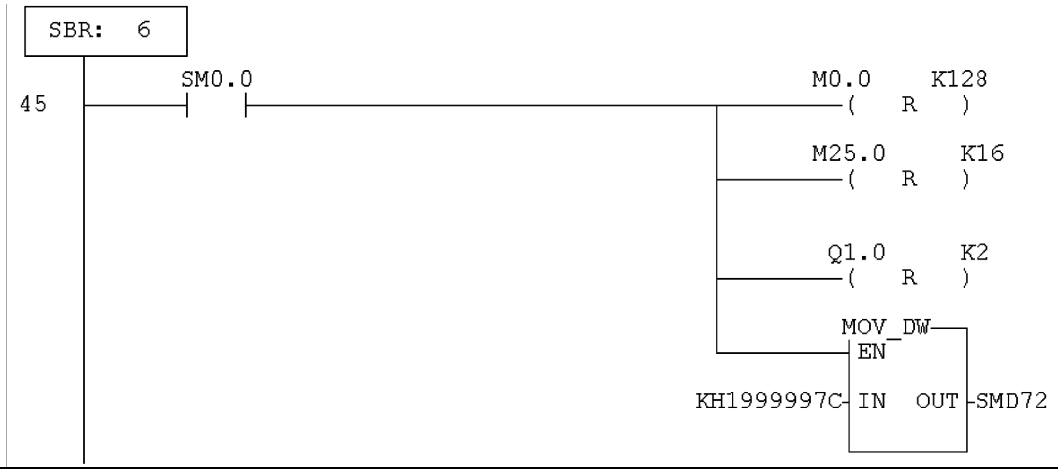



```
LDD>=    HC2，AC1        // 若实际值>设定值AC1
AN       I1.5            // 且顺时针方向旋转时（方向开关I1.5=0）
S        Q0.2，1         // 则逆时针方向校正
```

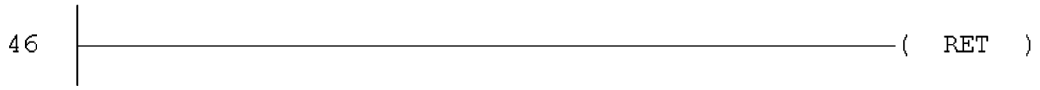



```
RET                      // 子程序5结束
```

// 子程序6“程序开始和错误确认之后的初始化”



```
SBR       6                    // 子程序6
LD        SM0.0                // SM0.0总为1
R         M0.0，128            // M0.0至M15.7复位
R         M25.0，16            // 错误定位计数器复位
R         Q1.0，2              // 操作模式和错误定位显示复位（Q1.0=0，Q1.1=0）
MOVD      16#1999997C，SMD72   // 搜索参考点的脉冲计数（PTO0）
```

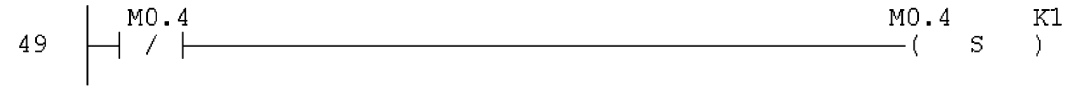



```
RET                            // 子程序6结束
```

中断程序

```
//  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// 中断0“脉冲输出终止”
```


```
INT       0                    // 中断程序0
LD        SM0.0                // SM0.0总是1
R         M0.1，1              // “电机运转”标志复位（M0.1=0）
S         M20.0，1             // “脉冲输出结束”标志置位（M20.0=1）
```




```
LDN       M0.4                 // 第1次定位控制之后
S         M0.4，1              // 设置相应标志信号（M0.4=1）
```

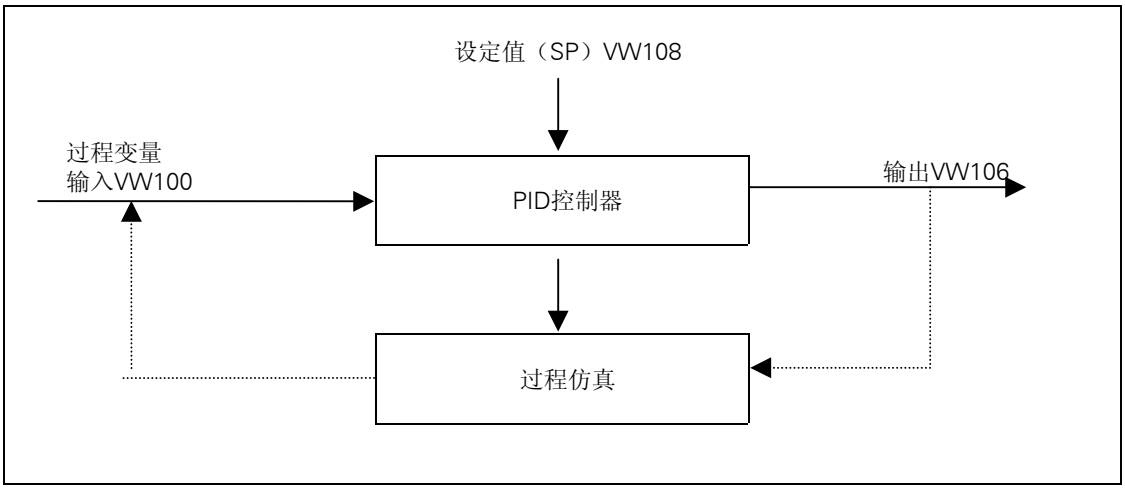

```
RET1                           // 中断程序0结束
```

H.14 用S7-200实现PID控制

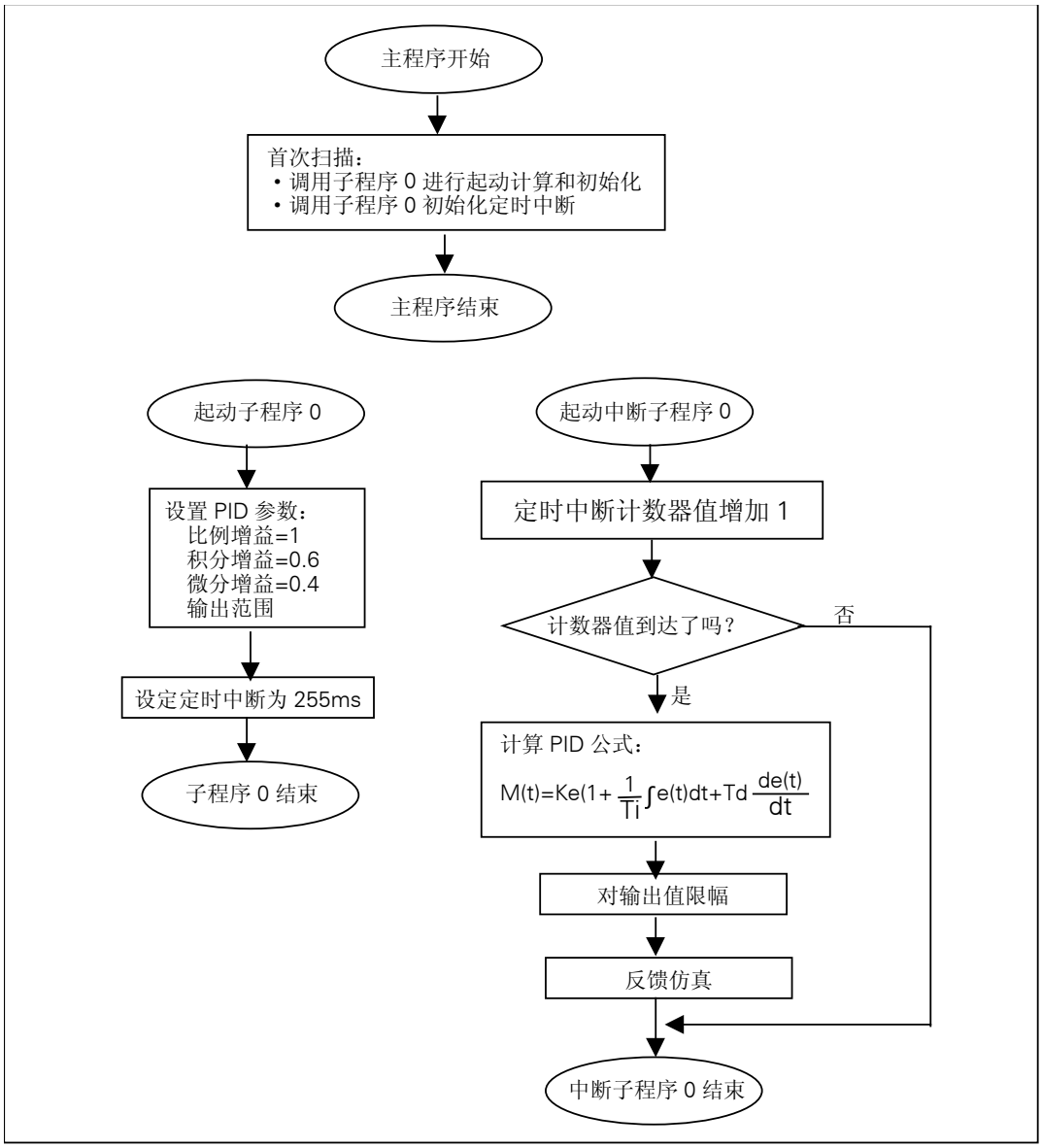
概述

本例描述了用S7-200实现PID控制功能。这个程序是一个带过程仿真的独立执行的PID例子，它很容易修改后与模拟模块EM235一起使用。

例图



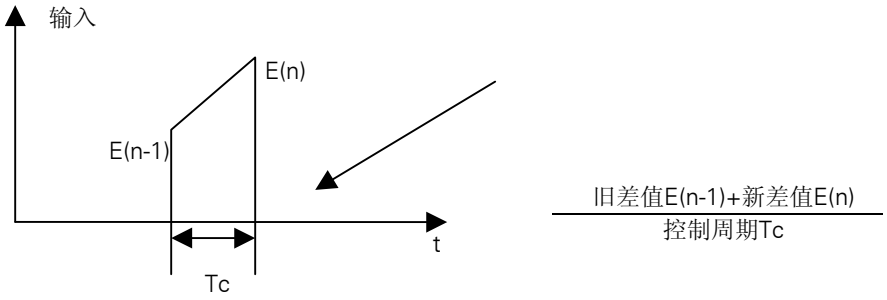
程序框图


主程序开始
首次扫描：
• 调用子程序 0 进行起动计算和初始化
• 调用子程序 0 初始化定时中断
主程序结束
起动子程序 0
设置 PID 参数：
比例增益=1
积分增益=0.6
微分增益=0.4
输出范围
设定定时中断为 255ms
子程序 0 结束
起动中断子程序 0
定时中断计数器值增加 1
计数器值到达了吗？
否
是
计算 PID 公式：
M(t)=Ke(1+ 1/Ti ∫e(t)dt+Td de(t)/dt
对输出值限幅
反馈仿真
中断子程序 0 结束

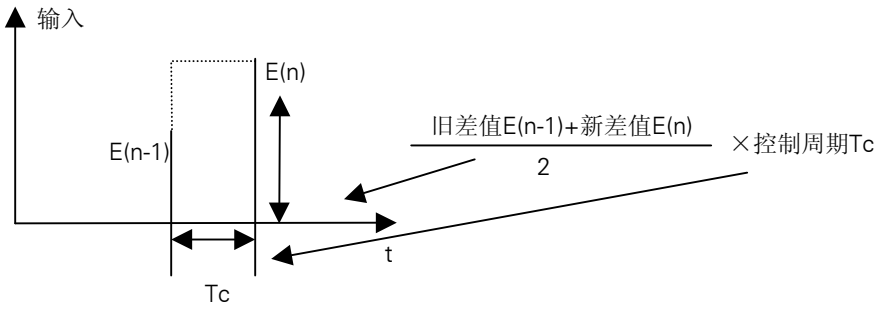
程序及注解

初始化部分将PID的所有值复位，并定义了计算PID控制器的控制周期Tc。
计算PID过程中，出现了某些数字方面的问题，以及控制周期Tc的计算。由于扫描时间的限制，除法运算通过移位来实现（1024近似为1000），而未调用专门的除法子程序。微分和积分是另外2个比较灵敏的数学运算，采用如下公式：

微分运算：



积分运算：

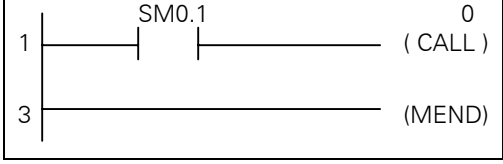


处理过程中的反馈输入变量来自仿真
程序全长370个字

LAD	STL（IEC）

//主程序

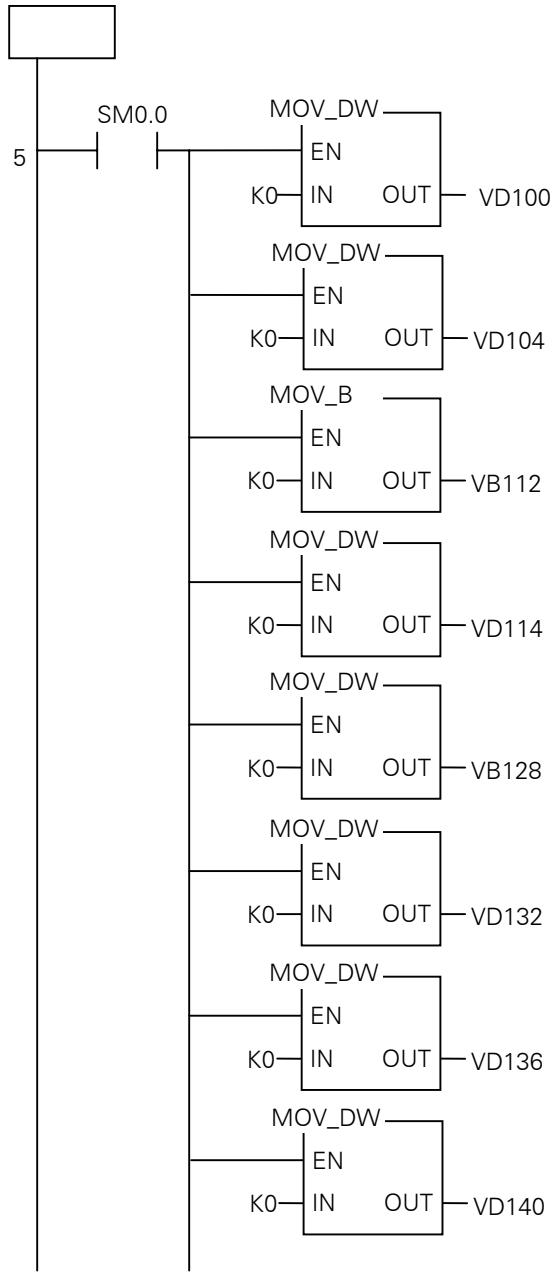
```
// VW100=过程变量（PV）输入量
// 如果模拟模块已准备好，它将是一个模拟输入字，并假设该输入的最大范围是
// -2047~+2047
// VW102=新控制差值（运行期间计算）E(n)
// VW104=为输出值而设的计算缓冲器
// VW106=输出值
// VW108=设定值（在CPU 214中，此值能够从模拟调节装置POTS中调入）
// VW110=预置定时中断个数，如果控制周期Tc大于255ms，则在此内存字中置数；
// 计算Tc的公式：
//                    VW110×VB113=Tc（单位：ms）
// VW112=未用
// VW113=定时中断0的时间（单位：ms）
// 注意：  若VW110是0，则控制周期Tc=VW113（ms）；
//         若VW110大于0，则控制周期Tc=VW110×VB113（ms）
//
// VW114=控制周期Tc（VW110与VB113之积）
// VW116=未用
// VW118=比例增益。1…1000=0.0078~7.8，
//         注意：例如，比例增益为1，可由VW118=128得到
//         注意：比例增益用于P、I、D部分，并与这3部分的和相乘。
// VW120=积分增益（1…1000=0.001~1）
// VW122=微分增益（1…1000=0.001~1）
// VW124=最大输出值（<=+2047），这是允许的最大输出值
// VW126=最小输出值（>-2047），  这是允许的最小输出值
// VW128=输出上溢出，此值由启动时，VW124值加1得到
// VW130=输出下溢出，此值由启动时，VW126值减1得到
// VW132=旧的控制差值（缓冲值，启动时置0）  E（n-1 ）
// VW134=积分寄存器（缓冲值，启动时置0）
// VW136=积分运算缓冲器
// VW138=积分值
// VW140=微分运算缓冲器
// VW142=微分值
// VW144=微分因子计算缓冲器
// VW146=微分因子，启动时，由微分增益/控制周期 [1/ms] 得到
//         （此值实际用于控制器运算
// VW148至
// VW154=用于起动运算的运算缓冲器
// VW156=定时中断计数器
// VW200至
// VD214=用于反馈仿真
```



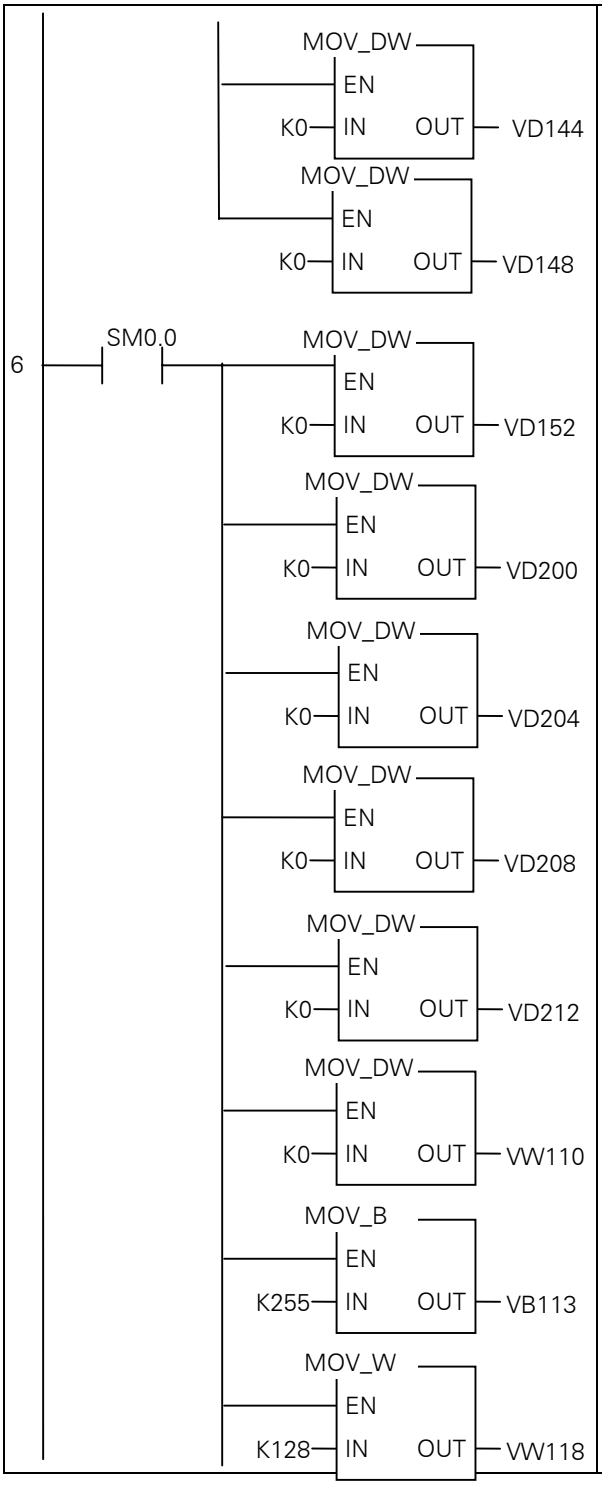

```
LD      SM0.1   // 仅首次扫描时SM0.1才为1
CALL    0       // 调用子程序0，进行起动运
                   算，初始化，设置定时中断
// 用户程序码段
MEND            // 主程序结束
```

//子程序

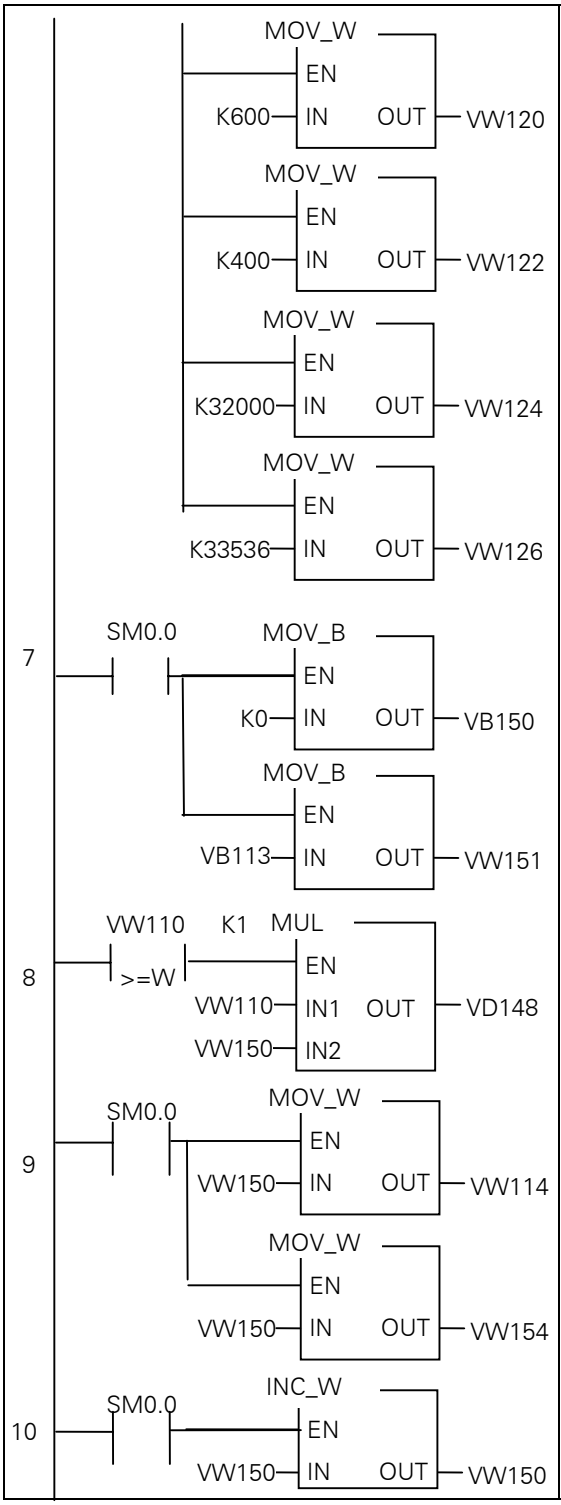
```
//  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// 功能：起动子程序
//  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```



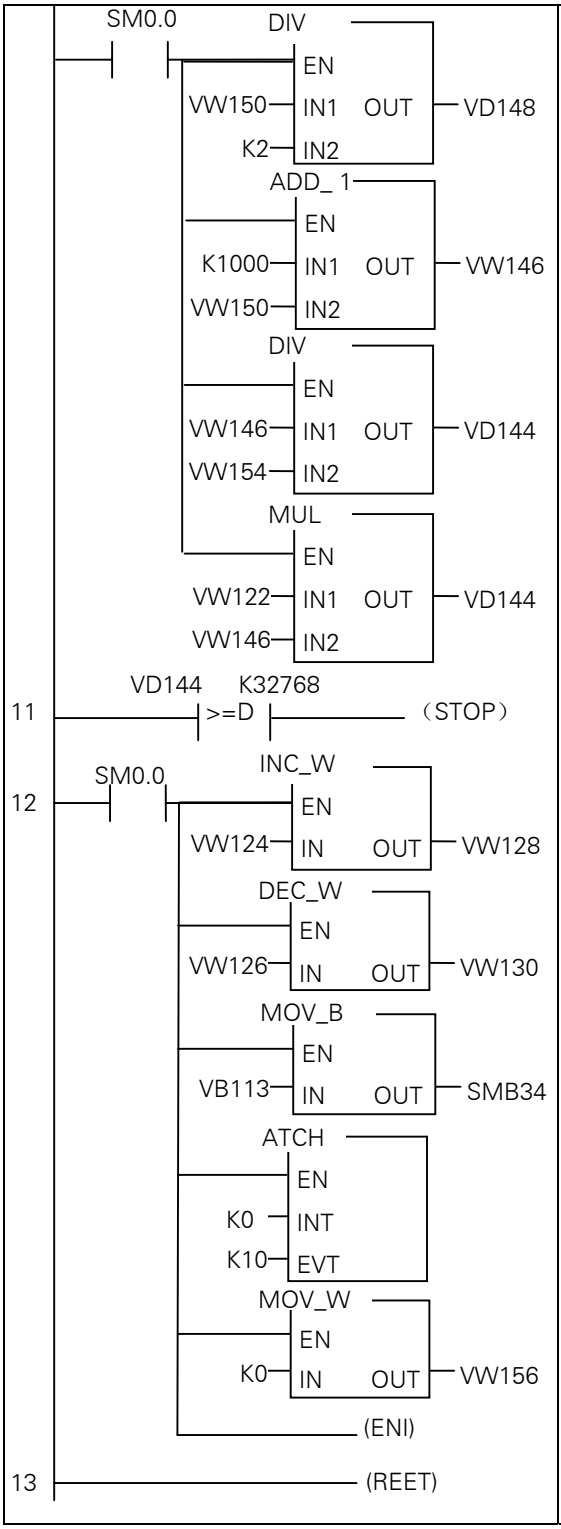

```
SBR     0               // 子程序0
// 对各变量进行正确复位
LD      SM0.0           // SM0.0总是为1
MOVD    0，VD100
MOVD    0，VD104
MOVB    0，VB112
MOVD    0，VD114
MOVD    0，VD128
MOVD    0，VD132
MOVD    0，VD136
MOVD    0，VD140
```




```
MOVD    0，VD144
MOVD    0，VD148
LD      SM0.0
MOVD    0，VD152
// 对仿真变量复位
MOVD    0，VD200
MOVD    0，VD204
MOVD    0，VD208
MOVD    0，VD212
// 设置PID控制器参数
MOVW    0，VW110      // 预置定时中断个数为0
MOVB    255，VB113    // 定时中断0的时间为255ms
MOVW    128，VW118    // 比例增益=128/128=1
```



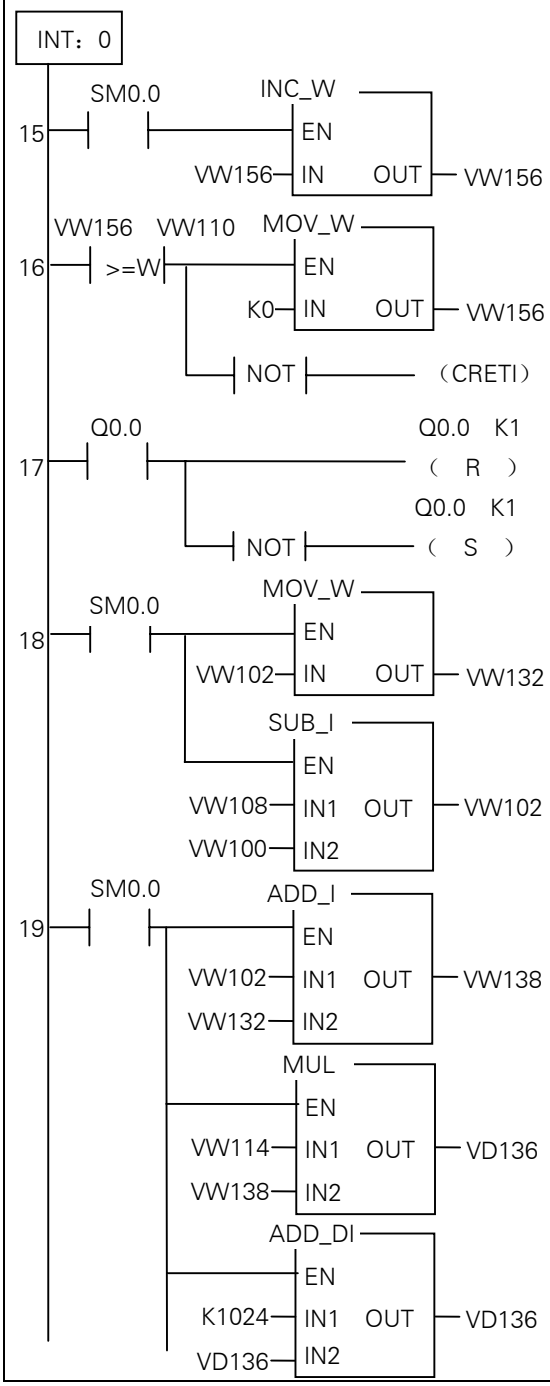

```
MOVW    600，VW120      // 积分增益= 600/1024=0.6
MOVW    400，VW122      // 微分增益=400/1024=0.4
MOVW    32000，VW124    // 最大输出值是+32000
MOVW    -32000，VW126   // 最小输出值是-32000
LD      SM0.0           //  SM0.0总为1
MOVB    0，VB150        // 运算缓冲器VB150清零
MOVB    VB113，VB151    // 将定时中断时间移入运算缓冲器VB151
LDW>=VW110，1           // 若预置定时中断个数≥1
MUL     VW110,VD148     // 则与定时中断时相乘，求得控制周期Tc=VD148，这是一个小于32767的正值（15位为1）
//
LD      SM0.0           // SM0.0总为1
MOVW    VW150，VW114    // 将控制周期Tc(ms)存入VW114
                        // 为微分部分而进行参数计算
                        // KD(1…1000)/ Tc[1/s]移入VW146
                        // Tc[ms]仍留在VW150中
MOVW    VW150，VW 154   // 将T控制周期Tc存入缓冲器VW154
LD      SM0.0           // SM0.0总为1
INCW    VW150           // Tc+1，这是为了移位
```



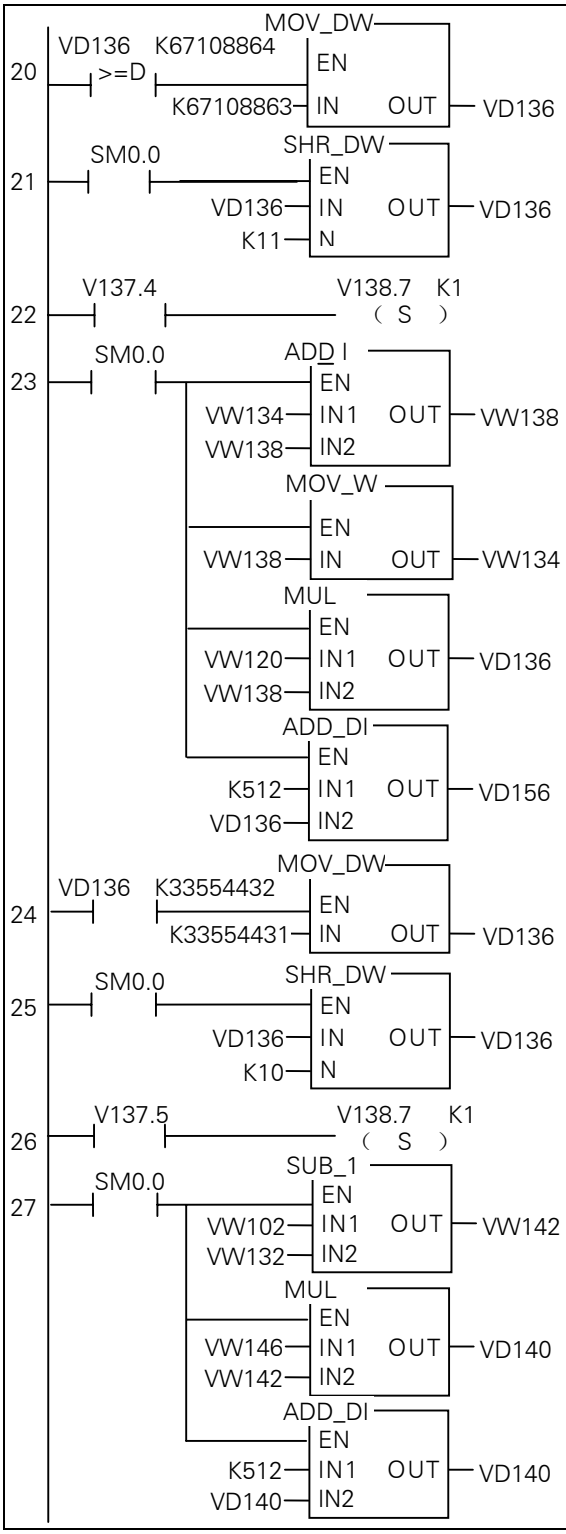

```
DIV     2，VD148            // (Tc+1)÷2

MOVW    1000,VW146          // 将1000置入微分因子缓冲器VW146
+1      VW150,VW146         // (TC+1)÷2的低端字加1000，这是为了移位

DIV     VW154,VD144         // 被Tc除

MUL     VW122,VD144         // 与微分增益相乘（1…1000）
LDD>=VD144，32768           // 检查是否上溢出，即计算值是否大于或等于2的15次方
                            // （16位为1）

STOP                        // 若有溢出，则停止‘STOP’
LD      SM0.0               // SM0.0总为1
MOVW    VW124，VW128        // 将最大输出值置入缓冲器VW128
INCW    VW128               // 加1以得到输出上溢出值
MOVW    VW126，VW130        // 将最小输出值移入缓冲器VW130
DECW    VW130               // 减1以得到输出下溢出值
// * * * * * * * * * * * * * * * * * * * * * * * * *
// 初始化定时中断
MOVB    VB113，SMB34        // 设置定时中断0的时间

ATCH    0.10                // 指定定时中断0（中断事件10）调用中断程序0

MOVW    0，VW156            // 定时中断计数器复位

ENI                         // 允许中断

RET                         // 子程序0结束
```

//中断程序

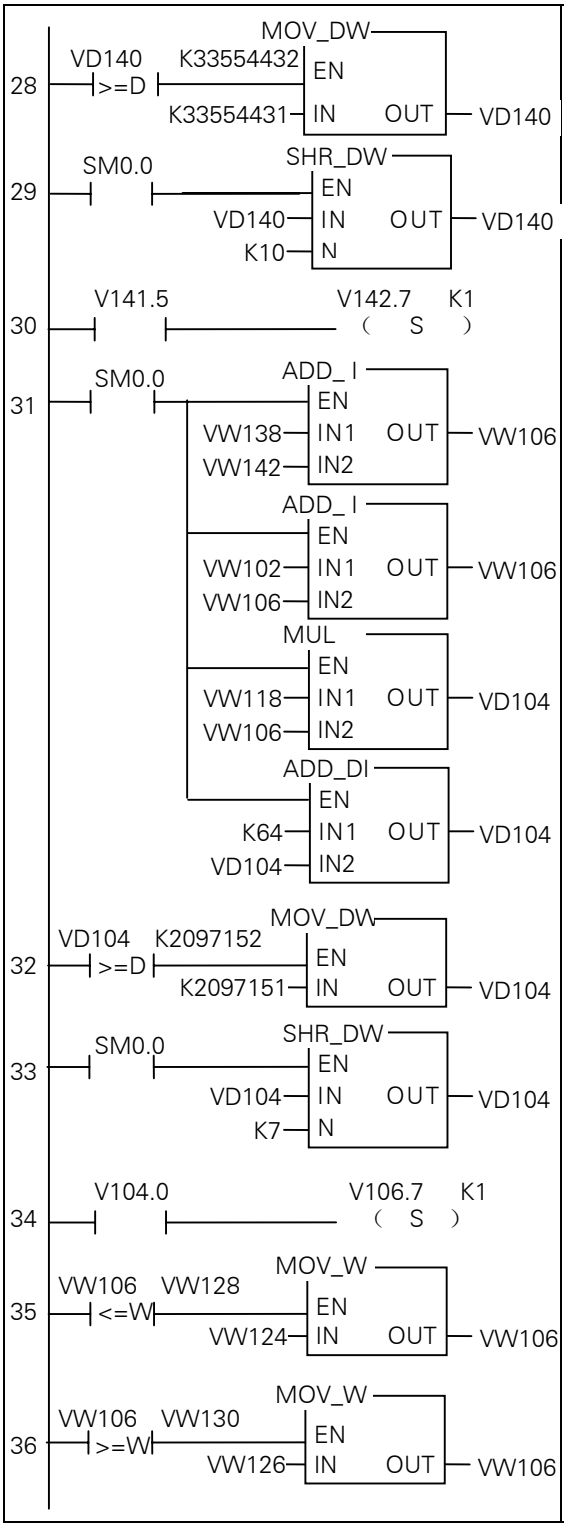
```
//*************************
//功能：由定时中断0调用的PID程序
//*************************
```



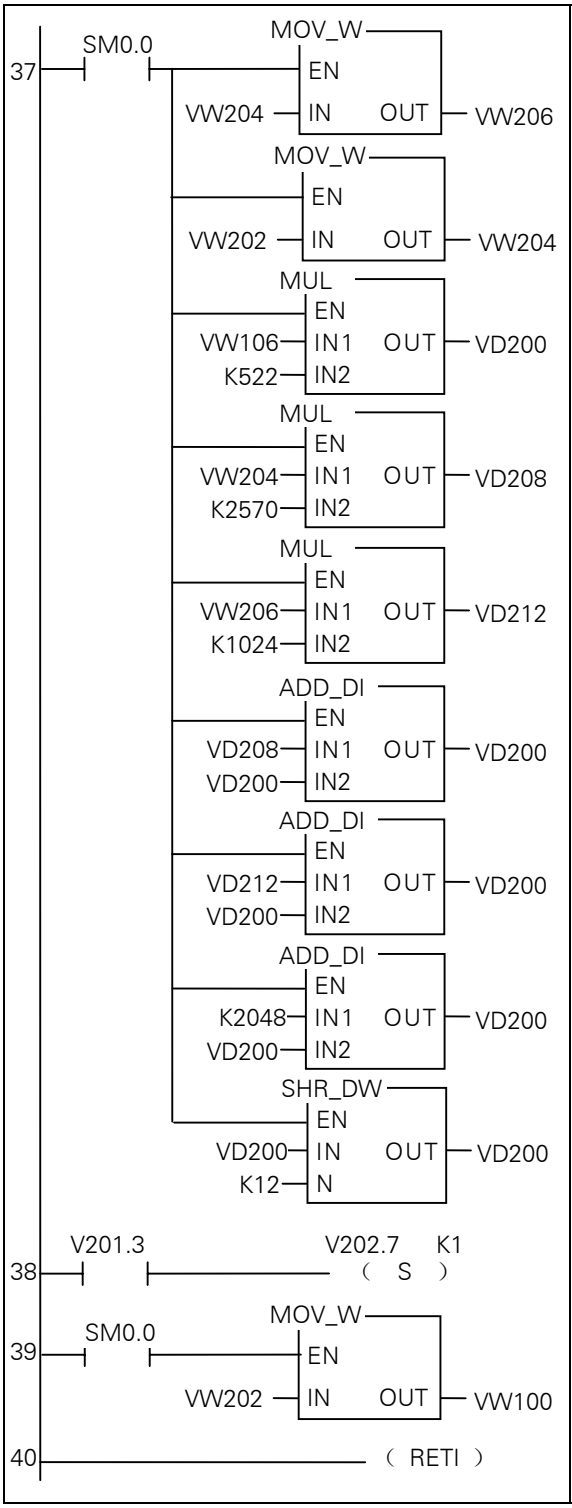

```
INT      0                  // 中断程序0
LD       SM0.0              // SM0.0总为1
INCW     VW156              // 定时中断计数器加1
LDW>=    VW156，VW110       // 若定时中断计数器数值≥预置定时中断个数
MOVW     0,VW156            // 则定时中断计数器复位；
NOT                         //
CRETI                       // 否则返回
// 下面4条指令是用来检查中断功能的，即在控制
// 器的扫描时间内输出端Q0.0的指示灯LED闪烁
LD       Q0.0
R        Q0,0，1
NOT
S        Q0.0，1
LD       SM0.0              // SM0.0总为1
MOVW     VW102，VW132       // 保存旧的控制差值E(n-1)
MOVW     VW108,VW102        // 设定值SP存入VW102
-1       VW100,VW102        // 设定值SP-过程变量PV=新的控制差值PV=新的控制差值E(n)
                            // E(n)存入VW102
//**********************
// 积分部分
LD       SM0.0              //  SM0.0总为1
MOVW     VW102，VW138       // 将新的控制差值E(n)置入缓冲器VW138
+1       VW132，VW138       //  E(n-1)+E(n)=VW138
         // 注：除以2由随后的移位实现
MUL      VW114，VD136       // 与控制周期Tc相乘（因为控制周期TC单位是
                            // 毫秒，乘积也许会是双字值！），即[E(n-1)+E(n)]×Tc=VD136
+D       1024，VD136        // 为更好的移位而加1024
```




```
LDD>=   VD136，67108864
                        // 检查是否上溢出，即计算值是否大于或等于2的26次方
                        // （27位为1)
MOVD    67108863，VD136
                        // 若有上溢出，则取最大值67108863
LD      SM0.0           //  SM0.0总为1
SRD     VD136，11       // 除以1024*2，即求得[E(n-1)+E(n)]×Tc÷2=VD136
LD      V137.4          // 注意符号位
S       V138.7，1

LD      SM0.0           // SM0.0总为1
+1      VW134，VW138    // 求积分部分累加值

MOVW    VW138，VW134    // 将新的积分部分值存入缓冲器VW134

MUL     VW120，VD136    // 与积分增益相乘（0…1000）

+D      512，VD136      // 为更好的移位而加512

LDD     >=VD136，33554432
                        // 检查是否上溢出，即计算值是否大于或等于2的25次方
                        // （26位为1）
MOVD    33554431，VD136
                        // 若上溢出，则取最大值33554431
LD      SM0.0           // SM0.0总为1
SRD     VD136，10       // 被1024除(正确值应是1000)
LD      V 137.5         // 注意符号位
S       V138.7，1       // VW138=积分值
// * * * * * * * * * * * * * * * * * * * * * *
// 积分部分
LD      SM0.0           // SM0.0总为1
MOVW    VW102，VW142    // 把新的控制差值E(n)存入VW142
-1      VW132，VW142    // E(n)-E(n-1)=VW142
MUL     VW146，VD140    // 与微分因子相乘

+D      512，VD140      // 为更好的移位而加512
```




```
LDD>=   VD140，33554432
                        // 检查是否上溢出，即计算值是否大于或等于2的25次方
                        // （26位为1）
MOVD    33554431，VD140
                        // 若上溢出，则取最大值33554431
LD      SM0.0           // SM0.0总为1
SRD     VD140，10       // 被1024除（正确值应是1000）
LD      V141.5          // 注意符号位
S       V142.7，1       // VW142=微分值
// * * * * * * * * * * * * * * * * * * * * * * * * * * *
// 比例部分及总和
LD      SM0.0           // SM0.0总为1
MOVW    VW138，VW106    // 积分结果送入输出缓冲器
+1      VW142，VW106    // 将微分和积分结果相加
+1      VW102，VW106    // 再加比例部分（新的控制差值E(n)）
MUL     VW118，VD104    // 与比例增益相乘，比例增益，1…100=0.0078~7.8
+D      64，VD104       // 为更好的移位而加64
LDD>=   VD104，2097152
                        // 检查是否上溢出。即计算值是否大于或等于2的21次方
                        // （22位为1）
MOVD    2097151，VD104  // 若上溢出，则取最大值2097151
LD      SM0.0           // SM0.0总为1
SRD     VD104，7        // 被128除
LD      V104.0          // 注意符号位
S       V106.7，1       // VW106=控制器输出
// * * * * * * * * * * * * * * * * * * * * * * * * * * *
// 对输出值限幅
LDW>=   VW106，VW128    // 若控制器输出值上溢出，即VW106>=VW128
MOVW    VW124，VW106    // 则将输出置为最大允许输出，即VW106=VW124
LDW<=   VW106，VW130    // 若控制器输出值下溢出，即VW106≤VW130
MOVW    VW126，VW106    // 则将输出置为最小允许输出，即VW106=VW126
// 控制器中断程序结束，在过程仿真求反馈输入期间此程序休息
```



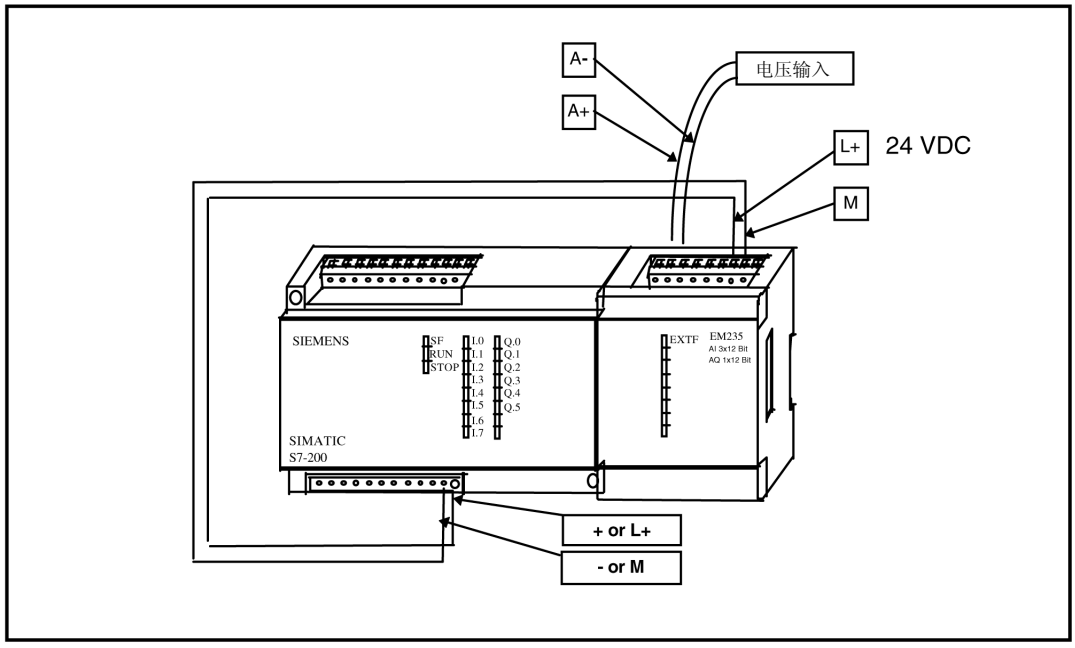

```
// 反馈输入通道
LD      SM0.0              // SM0.0总为1
MOVW    VW204，VW206       // 保存旧值，VW204存入VW206
MOVW    VW202，VW204       // 保存旧值，VW202存入VW204
MOVW    VW106，VW202       // 控制器输出值VW106存入VW202
MUL     522，VD200
MOVW    VW204，VW210
MUL     2570，VD208
MOVW    VW206，VW214
MUL     1024，VD212
+D      VD208，VD200
+D      VD212，VD200
+D      2048，VD200        // 为更好的移位而加2048
SRD     VD200，12          //被4096除
LD      V201.3             // 注意符号位
S       V202.7，1
LD      SM0.0              // SM0.0总为1
MOVW    VW202，VW100       // 作为新输入值反馈回去，即过程仿真结果作为过程变量（PV）输入VW100
RETI                       // 中断程序0结束
```

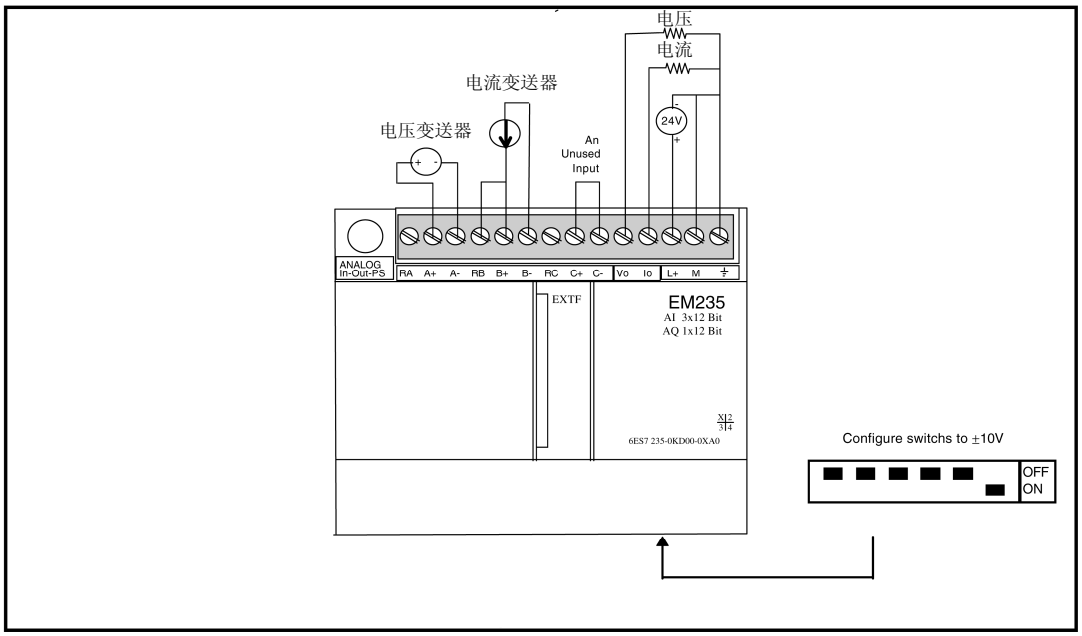
H.15 模拟量输入的处理

概述

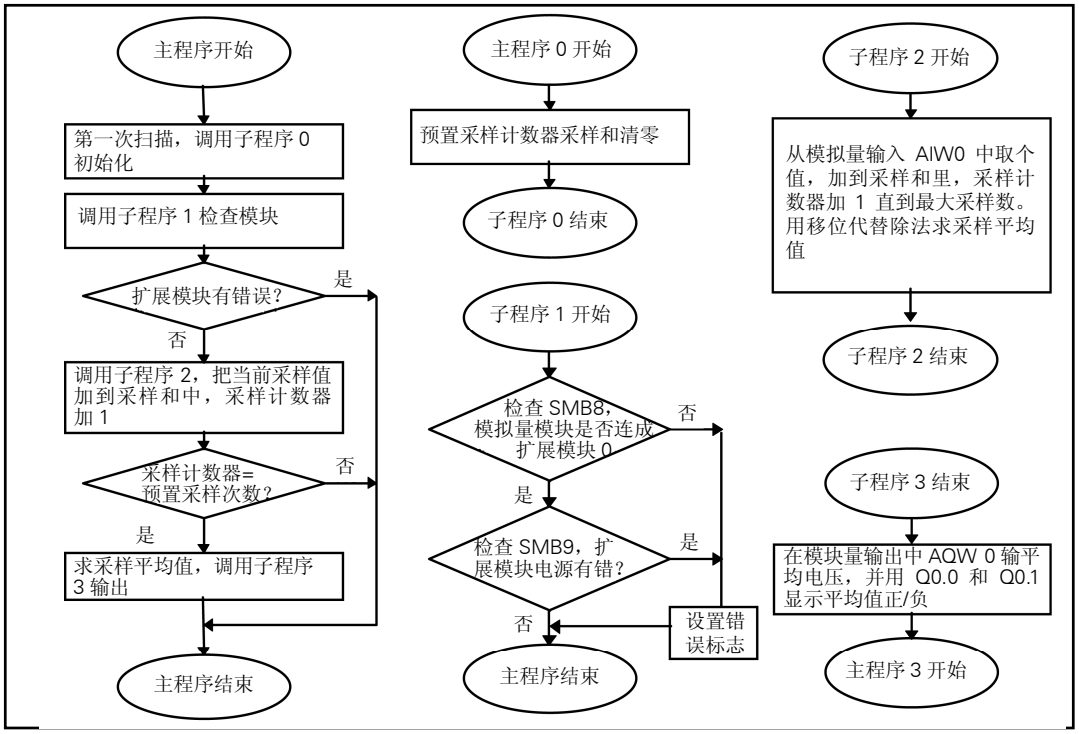
本示例描述了模拟量模块EM235 4AI 1AQ 与S7-200一起使用的一种探讨。本例中模拟量输入值是给定采样次数的采样平均值，然后试验决定怎样设置输出。EM235配置成±10V。

例图




电压
电流
电流变送器
电压变送器
24V
An
Unused
Input
ANALOG
In-Out-PS
RA A+ A- RB B+ B- RC C+ C- Vo Io L+ M
EXTF
EM235
AI 3x12 Bit
AQ 1x12 Bit
6ES7 235-0KD00-0XA0
Configure switchs to ±10V
OFF
ON

程序结果


主程序开始
第一次扫描，调用子程序 0 初始化
调用子程序 1 检查模块
扩展模块有错误?
是
否
调用子程序 2，把当前采样值加到采样和中，采样计数器加 1
采样计数器= 预置采样次数?
否
是
求采样平均值，调用子程序 3 输出
主程序结束
主程序 0 开始
预置采样计数器采样和清零
子程序 0 结束
子程序 1 开始
检查 SMB8，模拟量模块是否连成扩展模块 0
否
是
检查 SMB9，扩展模块电源有错?
是
否
设置错误标志
主程序结束
子程序 2 开始
从模拟量输入 AIW0 中取个值，加到采样和里，采样计数器加 1 直到最大采样数。用移位代替除法求采样平均值
子程序 2 结束
子程序 3 结束
在模块量输出中 AQW 0 输平均电压，并用 Q0.0 和 Q0.1 显示平均值正/负
主程序 3 开始

程序和注释

本程序描述了模拟量模块EM235 4AI 1AQ）的功能，从AIW0中取输入值，为了增加稳定性而求多次采样值的平均值，再依据计算出的平均值在AQW0中输出模拟电压。

模拟量模块经过测试可提供模块错误信息。如果第一个扩展模块不是模拟量模块，Q1.0接通。另外模拟量模块检查到的错误是电源出错，则将CPU上Q1.1 接通。模拟量模块上有EXTF字样。

本程序中所用除法是简单的移位除法（用采样次数的2的方次）。因为移位只花费较短的扫描时间，该数能从2变化到32768。

输入字是12位长。如果采样次数大于16（2的4次方），那么和的长度将大于一个字（16位）。于是需要用双字（32位）存贮采样和。为把输入值加到采样和中，你应当把它转成双字。

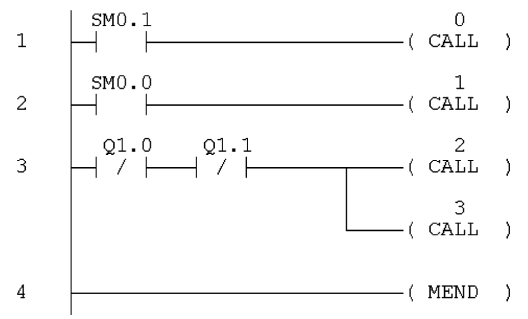
当输入数为负值时，最高有效字增添1；若为正值，最高有效字增添0来校正输入值。

本程序长度为118个字。

LAD（S7-MicroDOS）	STL(IEC)

// 主程序

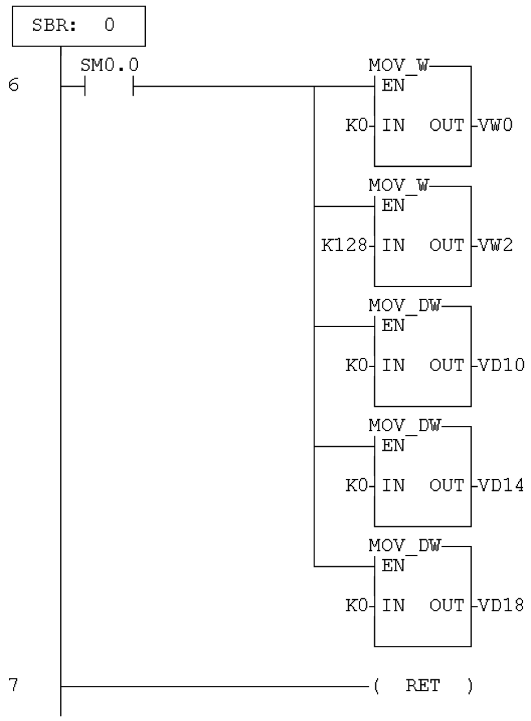
// 主程序




```
LD      SM0.1   // 仅首次扫描时SM0.1=1
CALL    0       // 初始化
LD      SM0.0   //
CALI    1       // 检查模块
LDN     Q1.0    // 若模块正常（即标志位Q1.0=0）
AN      Q1.1    // 且Q1.1=0
CALL    2       // 则采样模拟量输入
CALL    3       // 输出所要电压
MEND            // 主程序结束
```

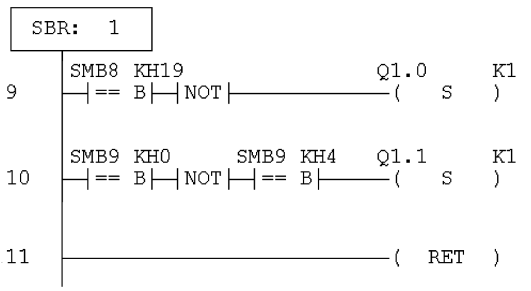
//子程序

```
//*************************************
// 功能：初始化
//*************************************
```



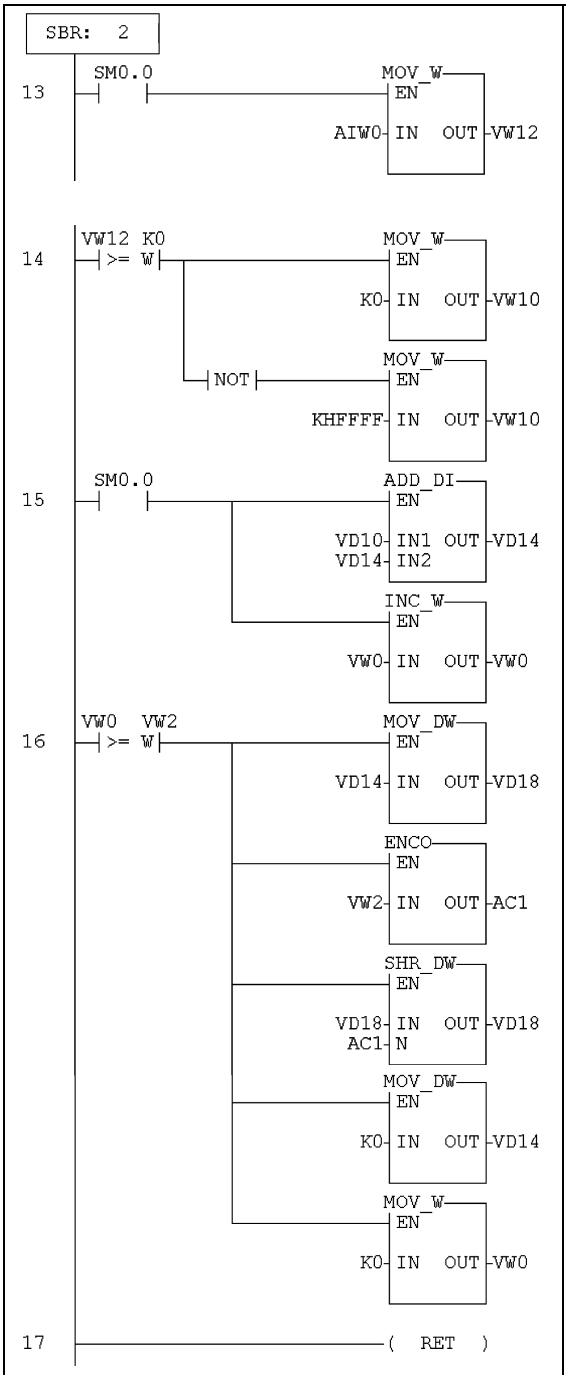

```
SBR     0           // 子程序0
LD      SM0.0       // SM0.0总为1

MOVW    0，VW0      // 计数器清零

MOVW    128，VW2    // 预置采样次数

MOVD    0，VD10     // 当前采样值清零

MOVD    0，VD14     // 当前采样和清零

MOVD    0，VD18     // 平均值清零

RET     子程序0结束
```

```
//**************************************
// 功能：检查模块是否正常工作
//**************************************
```



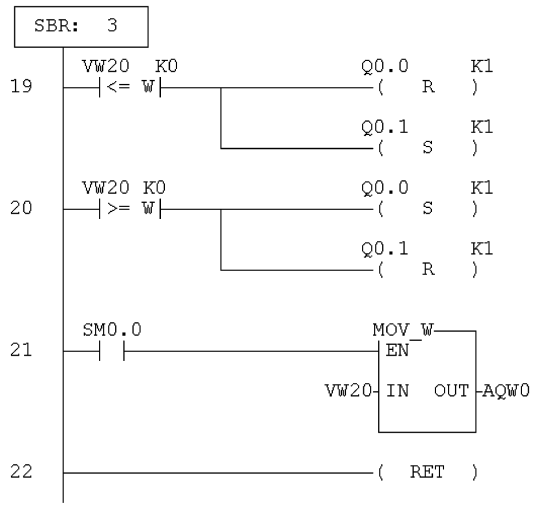

```
SBR     1           // 子程序1
LDB=    SMB8，16#19 // 检查第一个扩展模块
                       是否存在，
NOT                 //
S       Q1.0，1     // 若不存在，则置
                       Q1.0=1；
LDB=    SMB9，16#00 // 检查第一个扩展模块
                       是否有错误，
NOT                 //
AB=     SMB9，16#04 // 检查电源是否正常，
S       Q1.1，1     // 若有错，则置Q1.1
                       =1
RET
```

```
//**************************************
// 功能：采样模拟量，从AIW0取模块量输入值
//**************************************
```




```
SBR     2                 // 子程序2

LD      SM0.0             //  SM0.0总为1
MOVW    AIW0，VW12        // 在VW12中放置模拟量输入值

LDW>=   VW12，0           // 检查输入信号
MOVW    0，VW10           // 把输入值转换成双字

NOT                       // 即VD10=模拟量输入值(当前采样值)
MOVW    16#FFFF，VW10

LD      SM0.0             //  SM0.0总为1
+D      VD10，VD14        // 把当前采样值加到采样和中

INCW    VW0               // 采样计数器值加1

LDW>=   VW0，VW2          // 若达到采样次数
MOVD    VD14，VD18        // 则把采样和VD14复制到VD18中

ENCO    VW2，AC1          // 计算移位数

SRD     VD18，AC1         // 用移位实现除法，即求采样平均值

MOVD    0，VD14           // 重新初始化，即将采样和清零

MOVW    0，VW0            // 采样计数器清零

RET                       // 子程序2结束
                          // 模拟量输入平均值=VW20
```

```
//**************************************
// 功能：基于平均输入值设定输出值
//**************************************
```



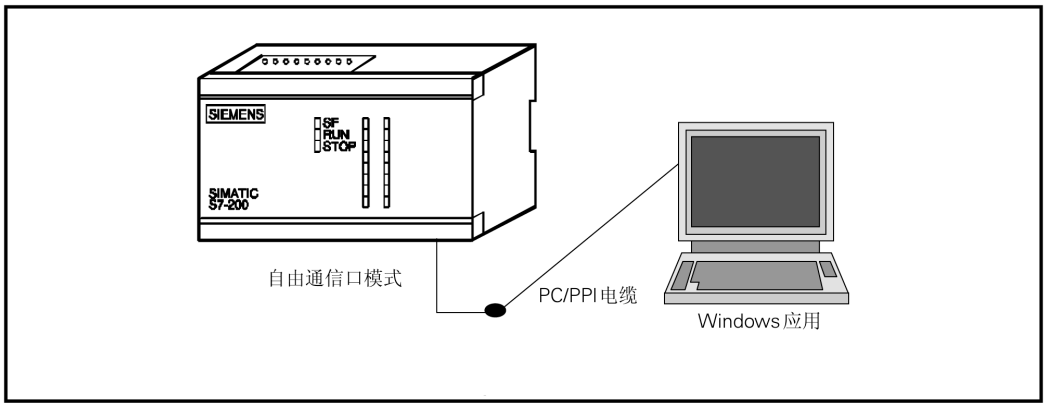

```
SBR     3              // 子程序3
LDW<=   VW20，-160     // 如果平均值为负
R       Q0.0，1        // 则关断Q0.0
S       Q0.1，1        // 且接通Q 0.1，用来显示当前平均值为负；

LDW>=   VW20，160      // 如果平均值为正
R       Q0.0，1        // 则接通Q0.0
S       Q0.1，1        // 且关断Q0.1，用来显示当前平均值为正。

LD      SM0.0          // SM0.0总为1
MOVW    VW20，AQW0     // 在模块AQW0输出平均值

RET                    // 子程序3结束
```

H.16 S7-200与PC之间的连接：从Windows应用程序中读数据

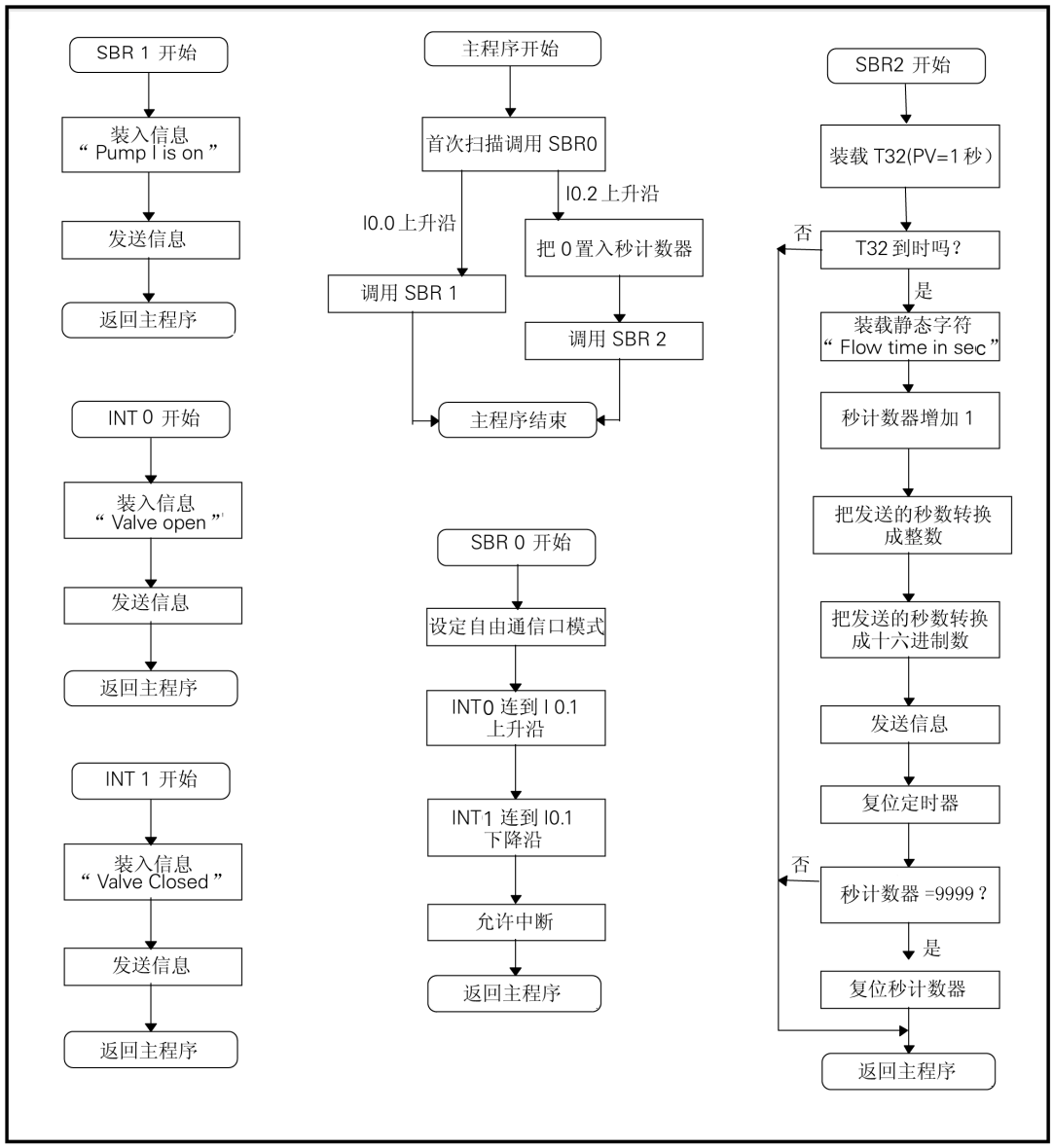
概述

本示例讲述了怎样用第三部分软件，由Windows应用程序，从SIMATIC S7-200系列CPU中读数据。本例模仿一个简单的‘泵站’系统，把数据发送到Microsoft Excel中不同的位置。

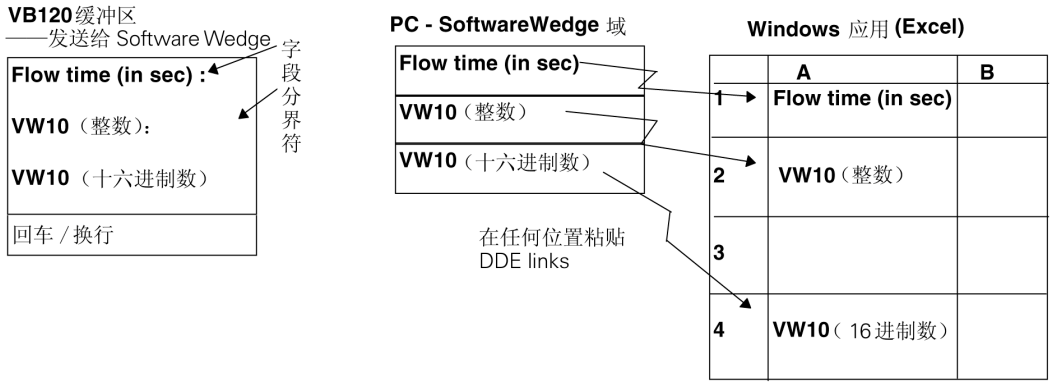
例图



程序框图


SBR 1 开始
装入信息
“ Pump I is on ”
发送信息
返回主程序
INT 0 开始
装入信息
“ Valve open ”
发送信息
返回主程序
INT 1 开始
装入信息
“ Valve Closed ”
发送信息
返回主程序
主程序开始
首次扫描调用 SBR0
I0.0 上升沿
I0.2 上升沿
调用 SBR 1
把 0 置入秒计数器
调用 SBR 2
主程序结束
SBR 0 开始
设定自由通信口模式
INT0 连到 I 0.1
上升沿
INT1 连到 I0.1
下降沿
允许中断
返回主程序
SBR2 开始
装载 T32(PV=1 秒）
否
T32 到时吗？
是
装载静态字符
“ Flow time in sec ”
秒计数器增加 1
把发送的秒数转换
成整数
把发送的秒数转换
成十六进制数
发送信息
复位定时器
否
秒计数器 =9999？
是
复位秒计数器
返回主程序

程序和注释



SIMATIC CPU S7-200能与基于Windows的程序，如SoftwareWedge for Windows之类软件相联系。所以，来自S7-200的信息能显示在任何Windows应用程序中，同时信息也能从Windows应用程序写到S7-200中。

目前，SoftwareWedge不允许发送来自不同输入的信息，在不同的时间显示和更新屏幕的不同部分。然而，从S7-200发送来的各种信息，可以显示在不同位置，每个部分必须显示在SoftwareWedge自己的区域里，每个区域被发送来的某个字段分界符分隔开。这些字符可以是用户任意要求的。此外，每次发送结束，必须有一个或多个“结束”字符，它也可由用户指定。

装载完SoftwareWedge软件包后，选择DDE服务器方式，指定DDE应用名、题目及适用的项目，接着把通信口设定为9600波特，没有奇偶校验，每个字符8位，1个停止位。记住所设定的通信口是好的。最好，要输入的记录结构必须定义。在下面程序中，从收到任一字符作为记录的开始，收到一个回车和换行作为记录的结束，选择多个数据字段，用3作为字段的最大数目，用“：”（ASCII码为58）号作为字段分界符。最后，在Windows应用中，用拷贝/粘贴联接命令把不同数据字段粘贴到屏幕上所要求的部分。

选择：在它进入另一个Windows应用前SoftwareWedge提供了取消变量格式的自动转换。

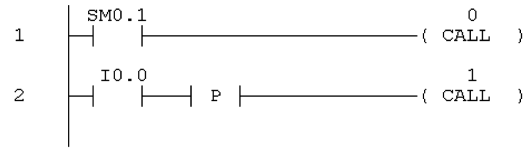
本程序长度为158个字

LAD (S7-MicroDOS)	STL(IEC)

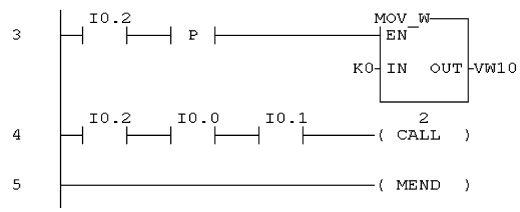
主程序

```
// 标题：Windows界面，并在Windows里读
// 该程序描述了利用适当的软件，S7-200系列CPU可以发送信息给任何Windows
// 应用产品（见上面）
// 本例给出了一个简单的“泵站”系统。假定I0.0为起动“主泵”的开关，I0.1为开或
// 关紧急阀的开关，I0.2为开或关主阀的开关（主阀控制液体流出）
// 操作员坐在计算机可以看到三个状态变化中的任一个信息。I0.0控制一个静态信息
// “Pump 1 is on”的出现。I0.1控制着信息“Valve open”或
// “Valve closed”交替出现。I0.2控制显示液体流出的时间，这个信息在 I0.2
// 接通时每秒变化一次。该程序不允许同时显示所有信息，但是可作为每个开关变化的
// 状态。然而，对程序作某些修改，那就可以同时显示所有信息。这就要求所有数据随
// 时发送（即使没有变化），并改用新的发送模式。
// 本程序字段分界符是（：），发送结束符是一个回车及换行符。
// 该程序试图写一系列同类型的信息给Windows应用程序。一条静态信息
// “Pump is on”，一条切换信息“Valve is ope/closed”，以及一条不断地变化的
// 整数或十六进制数“Flow time (in sec)####”全包括在一起，以便给用户一个多种多样的信息变化。

// 下面列出本程序用到的变量
```

// VW10	主计数存贮器，用来显示液体流出时间（秒）
// VW20	第二计数存贮器——来自VW10的拷贝，用在SBR2中进行IBCD转换，
//	而且不擦除计数器里的值。
//VB80	存贮数值14，即发送缓冲区中待发送的ASCII码（十六进制）字母数。用作XMT发送命令的前提。
// VD81-VW93	信息：“Pump 1 is on”
// VB100	依据紧急阀的状态存贮12或14，即发送缓冲区中待发送的ASCII码（十六进制）字母数。
// VD101-VD109	
// 或VD101-VD113	信息：“Valve open”或“Valve closed”
// VB120	存贮数值28，即发送缓冲区中待发送的ASCII码（十六进制）字母数。
// VD121-VD133	信息“Flow time is sec”
// VB137	以十六进制形式存贮“：”，作为字段分界符
// VB138-VB141	存贮秒计数器的ASCII码，它在Windows中作为整数显示。
// VB142	以十六进制形式存贮“：”，作为下一字段分界符
// VB143-VB146	存贮秒计数器的ASCII码，它在Windows中作为十六进制显示
// VW147	回车换行 – 表示发送信息结束

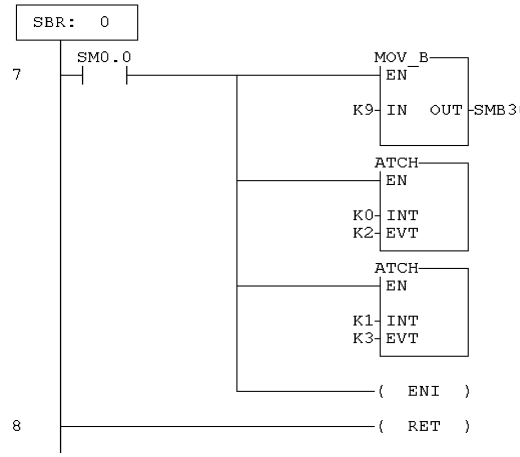



```
LD      SM0.1   // 首次扫描位 SM0.1=1
CALL    0       // 调用SBR 0
LD      I0.0    // 若主泵开关I0.0=1
EU              // 且上升沿
CALL    1       // 则调用SBR1
```



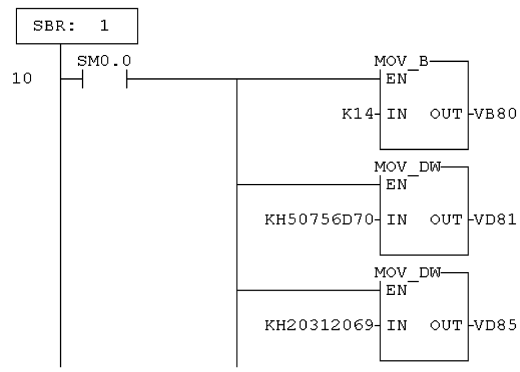

```
LD      I0.2        // 若主阀开关I0.2=1
EU                  // 且上升沿
MOVW    0，VW10     // 则定时器置0
LD      I0.2        // 若主阀开关I0.2=1
A       I0.0        // 且泵运转（I0.0=1）
A       I0.1        // 且紧急阀打开（I0.1=1）
CALL    2           // 则调用SBR2
MEND                // 主程序结束
```

子程序

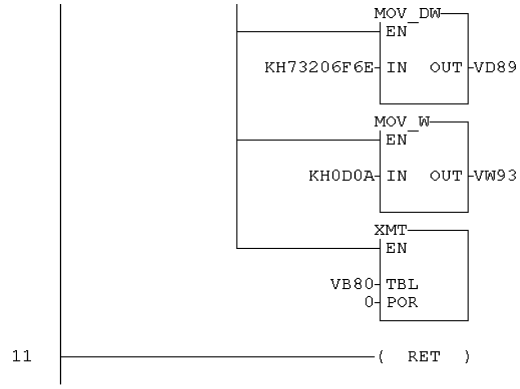



```
SBR     0           // 初始化子程序0

LD      SM0.0       // SM0.0总是1
MOVW    9，SMB30    // 自由通信口模式：9600波特，无奇偶校验，每个字符8位

ATCH    0，2        // 指定中断事件2（I0.1上升沿）调用中断程序0

ATCH    1，3        // 指定中断事件3（I0.1下降沿）调用中断程序1

ENI                 // 允许中断
RET                 // 返回主程序
```

```
// 每当主泵开关I0.0上升沿时，将“Pump 1 is on ”装入，从VB81开始的缓冲区，并在
// VB80中置发送字节数14，通过自由通信口模式发送
```



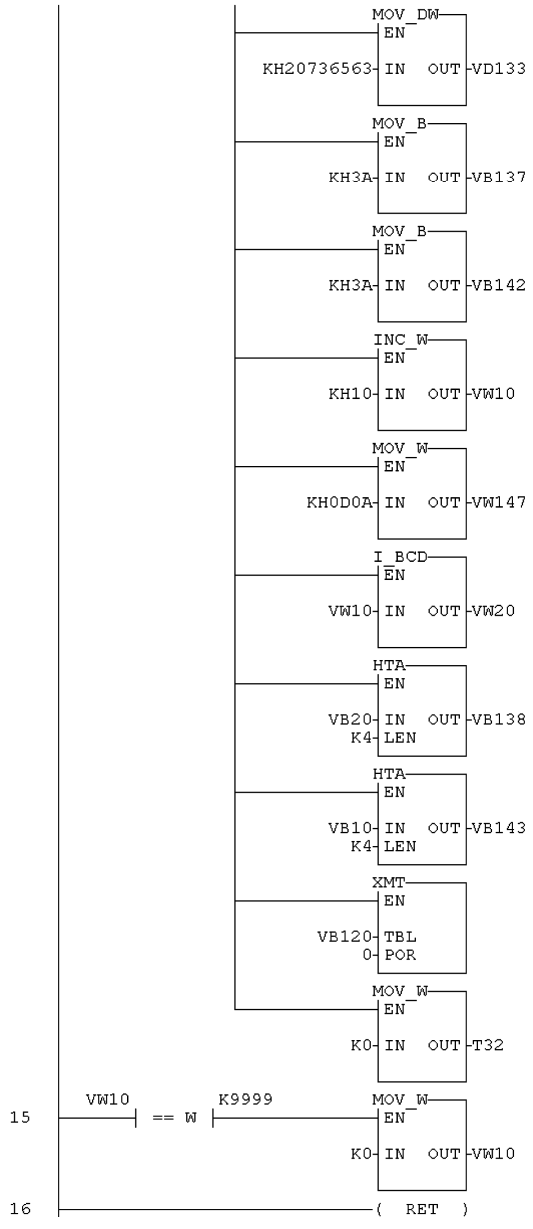

```
SBR1                // 写“Pump 1 is on”

LD      SM0.0       // SM0.0总是1

MOVB    14，VB80    // 置发送信息长度（14个字节）

MOVD    16#50756D70，VD81
                    // 十六进制信息—“Pump”

MOVD    16#20312069，VD85
                    // “1 l”
```




```
MOVD    16#73206F6E，VD89
                        // “s on”
MOVW    16#0D0A，VW93
                        // 回车换行表示信息结束
XMT     VB80，0         // 发送（VB80中已存放
                           所发送信息的字节数）
RET                     // 返回主程序
```

```
// 当I0.2接通时，给T32（TON，1ms精度的定时器）装入预设值1000（1秒）。当T32到时后，
// VW10增加1，并将新值装入VW20。VW20由整数转换成BCD码，再转换成ASCII码，
// 并存入发送缓冲区。VW10也被移到适当的缓冲器。
// 十六进制数作为实际值转换成ASCII码供发送。在子程序结束时，进行测试，
// 如果值超过9999，复位VW10。这是因为BCDI命令只允许转换一个字（或4位址六进制数）。
// 如果需要十六进制数，则任何数都能被转换。从VB120开始的发送缓冲区每秒都更新
// （由T32定时），且读“Flow time in sec（整数）（十六进制数）”
```

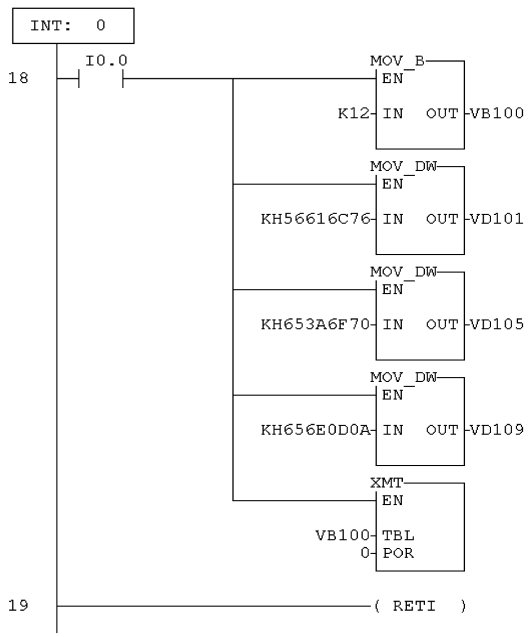
```
SBR     2               // 子程序2
LD      SM0.0           // SM0.0总为1

TON     T32，1000       //启动定时器T32(1000ms)
LD      T32             // 一秒以后
MOVB    28，VB120       // 置发送信息长度（28个
                           字节）
MOVD    16#466C6F77，VD121
                        // “FLow”
MOVD    16#2074696D,VD125
                        // “tim”
MOVD    16#6520696E,VD129
                        // “e in”
```




```
MOVD    16#20736563VD133
                        // “sec”
MOVB    16#3A,VB137
                        // “：”字段分界符
MOVB    16#3A，VB142
                        //“：”字符分界符
INCW    VW10            // VW10增加1
MOVW    16#0D0A，VW147
                        // 回车换行表示信息结束
MOVW    VW10，VW20      // 把VW10拷贝到VW20
IBCD    VW20            // 把VW20转换成BCD码
HTA     VB20，VB138，4
                        // 把BCD转换成ASCII（作数值显示），装入发送缓冲区。
HTA     VB10，VB143，4
                        // 把十六进制转换成ASCII（作十六进制显示），装入发送缓冲区
XMT     VB120，0        // 发送（VB120中已存放所发送信息的字节数）
MOVW    0，T32          // 复位定时器
LDW=    VW10，9999      // 若流出时间超过9999秒
MOVW    0，VW10         // 则复位秒计算器
RET                     // 返回主程序
```

中断程序

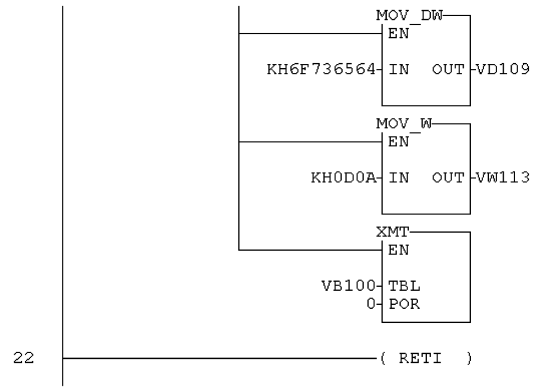
```
// 若为I0.1上升沿，（即接通），则调用中断程序0发送信息“Valve open”
// 因此，只有阀转换状态时才更新阀信息。
```




```
INT     0               // 中断程序0
LD      SM0.0           // SM0.0总为1
MOVB    12，VB100       //置发送信息长度（12个字节）
MOVD    16#56616C76，VD101
                        // “Valv”
MOVD    16#653A6F70,VD105
                        // “e：op”，“：”是字段分界符
MOVD    16#656E0D0A，VD109
                        // “en”，回车换行表示信息结束
XMT     VB100，0        // 发送（VB100中已存放所发送信息的字节数）
RETI                    // 返回主程序
```

// 若为I0.0下降沿（即断开），则调用中断程序1发送“Valve closed”。只有阀转换
// 状态时信息才再一次更新。

```
INT     1               // 中断程序1
LD      SM0.0           // SM0.0 总为1
MOVW    14，VB100       // 置发送信息长度（14个字节）
MOVD    16#56616C76，VD101
                        // “Valv”
MOVD    16#653A636C,VD105
                        // “e cl”
```




```
MOVD    16#6F736564，VD109
                        // “osed”
MOVW    16#0D0A,VW113
                        // 回车换行表示信息结束
XMT     VB100，0        // 发送（VB100中已存放所发送信息的字节数
RETI                    // 返回主程序
```

H.17 自由口模式下PLC与计算机的通信

概述

本例说明如何以自由协议实现计算机与 S7－200 的通信，计算机作为主站，可以实现对 PLC 从站各寄存器的读/写操作。

计算机通过 COM 口发送指令到 PLC 的 PORTO（或 PORT1）口，PLC 通过 RCV 接收指令，然后对指令进行译码，译码后调用相应的读/写子程序实现指令要求的操作，并返回指令执行的状态信息。

通信协议

在自由口模式下，通信协议是由用户定义的。用户可以用梯形图程序调用接收中断、发送指令（XMT）、接受指令（RCV）来控制通信操作。在自由口模式下，通信协议完全由梯形图程序控制。

指令格式定义

- 计算机每次一个 33 字节长的指令来实现一次读/写操作，指令格式见表 1 说明：

1. 起始字符
   起始字符标志着指令的开始，在本例中被定义为 ASCII 码的“g”，不同的 PLC 从站可以定义不同的起始字符以接收真对该 PLC 的指令。
2. 指令类型
   该字节用来标志指令的类型，在本例中 05H 代表读操作，06H 代表写操作。
3. 目标 PLC 站地址
   目标 PLC 站地址占用指令的 B2、B3 两个字节，以十六进制 ASCII 码的格式表示目标 PLC 的站地址。
4. 目标寄存器地址
   在 PLC 内部可以用 4 个字节来表示一个寄存器的地址（但不能表示一个位地址）。前两个字节表示寄存器类型，后两个字节表示寄存器号。

   00　00（H）：I 寄存器区

   01　00（H）：Q 寄存器区

   02　00（H）：M 寄存器区

   08　00（H）：V 寄存器区
5. 例如：

   IB000 的地址可表示为　　00　00　00　00（H）

   VB100 的地址可表示为　　08　00　00　64（H）
6. 读/写字节数 M
   当读命令时，始终读回从目标寄存器开始的连续 8 个字节的数据（转换为十六进制 ASCII 码后占用 16 个字节），可以根据自己的需要取用，M 可以任意写入。

当写命令时，M 表示的是要写入数据的十六进制 ASCII 码所占用的字节数。例如要写入 1 个字节的数据，数据在指令中以十六进制 ASCII 码表示，它将占用 2 个字节，此时应向 M 中写入“02”。同理，如果要写入 5 个字节的数据，M 中应写入“0A”。

7. 要写入的数据
   要写入的数据在指令中以十六进制 ASCII 码的格式表示，占用指令的 B14-B29 共 16 个字节。数据区必须填满，但只有前 M 个字节的数据会被写入目标寄存器。一条指令最多可以写入 8 个字节的数据（此时 M 中应写入“10”，代表十进制的 16）
8. BCC 校验码
   在传输过程中，指令有可能受到任何的干扰而使原来的数据信号发生扭曲，此时的指令当然是错误的，为了侦测指令在传输过程中发生的错误，接收方必须对指令作进一步的确认工作，以防止错误的指令被执行，最简单的方法就是使用校验码。BCC 校验码的方法就是将要传送的字符串的 ASCII 码以字节为单位作异或和，并将此异或和作为指令的一部分传送出去；同样地，接收方在接到指令后，以相同的方式对接收到的字符串作异或和，并与传送方所送过来的值作对比，若其值相等，则代表接收到的指令是正确的，反之则是错误的。
   在本例中，bcc 为指令 B1 到 B29 的异或和，BCC 为 bcc 的十六进制 ASCII 码。
   bcc=B1 xor B2 xor B3 xor B4 xor …… xor B29
9. 结束字符
   结束字符标志着指令的结束，在本例中被定义为 ASCII 码的“G”，不同的 PLC 从站可以定义不同的结束字符以接收真对该 PLC 的指令。

- PLC 在接到上位机指令后，将发送一个 21 字节长反馈信息，格式见表 2
  说明：

1. 起始字符
   起始字符标志着反馈信息的开始，在本例中被定义为 ASCII 码的“g”，不同的 PLC 从站可以定义不同的起始字符，这样上位机可以根据信息的起始字符来判断反馈信息的来源。
2. 状态信息
   该字节包含指令执行的状态信息，在本例中

01H	代表	读取正确
02H	代表	写入正确
03H	代表	BCC校验码错误
04H	代表	指令不合法

3. 数据区
   反馈信息的 B3 到 B18 为读指令所要读取的数据，以十六进制 ASCII 码表示。
4. BCC 校验码
   与上位机指令中的 BCC 校验码类似，它是反馈信息 B3 到 B18 的异或和。
5. 结束字符

结束字符标志着反馈信息的结束，在本例中被定义为 26H。

指令中为何要使用 ASCII 码

一条指令除包含数据外，还包含必要的控制字（起始字符、结束字符、指令类型等）。如果指令中的数据直接以其原本的形式传输，则不可避免的会与指令中的控制字发生混淆。

例如本例中，指令的起始字符为“g”，其 ASCII 码值为 67H，结束字符为“G”，其 ASCII 码值为 47H。假设要写入的数据中也有 47H，并且数据直接以其原本的形式传输，则 PLC 会因为接收到了数据中的 47H 而停止接收，这样 PLC 接收到的指令将是一个不完整的非法指令，很可能造成 PLC 的误动作。

为了避免这种情况的发生，可以用文本来传送二进制数据。通过以 16 进制 ASCII 码的格式描述数据，每个二进制的字节都可以表示成一对 ASCII 编码，这对编码表示这个字节的两个 16 进制字符。这种格式可以表示任何的数值，仅仅使用 ASCII 代码的 30H 到 39H（表示 0 到 9）和 41H 到 46H（表示 A 到 F）。ASCII 码的其余部分可以用作控制字（起始标志、结束标志、指令类型等）。这样，数据中的 47H 以 ASCII 码的形式进行传送就变成了 34H 37H 两个字节，从而避免了 PLC 因接收到数据中的 47H 而停止接收的错误。

表 1 上位机指令格式

Byte0	起始字符
Byte1	指令类型（读/写）
Byte2	目标 PLC 站地址（十六进制 ASCII 码）
Byte3	
Byte4	目标寄存器地址（十六进制 ASCII 码）
Byte5	
Byte6	
Byte7	
Byte8	
Byte9	
Byte10	
Byte11	
Byte12	读/写字节数 M（十六进制 ASCII 码）
Byte13	
Byte14	要写入的数据（十六进制 ASCII 码）
Byte15	
Byte16	
Byte17	
Byte18	
Byte19	
Byte20	

Byte21	
Byte22	
Byte23	
Byte24	
Byte25	
Byte26	
Byte27	
Byte28	
Byte29	
Byte30	BCC 校验码（十六进制 ASCII 码）
Byte31	
Byte32	结束字符

表 2 反馈信息格式

Byte0	起始字符
Byte1	状态信息
Byte2	数据区（十六进制 ASCII 码）
Byte3	
Byte4	
Byte5	
Byte6	
Byte7	
Byte8	
Byte9	
Byte10	
Byte11	
Byte12	
Byte13	
Byte14	
Byte15	
Byte16	
Byte17	
Byte18	BCC 校验码（十六进制 ASCII 码）
Byte19	
Byte20	结束字符

PLC 程序执行过程

PLC 在第一次扫描时执行初始化子程序，对端口及 RCV 指令进行初始化。初始化完成后，运行 RCV 指令使端口处于接受状态。

RCV 会将以“g”开头“G”结尾的指令保存到接收缓冲区，并同时产生接收完成中断。

RCVcomplete 中断服务程序用来处理接收完成中断事件，它会将接收缓冲区中的十六进制 ASCII 码还原成数据并保存，同时置位 Verify 子程序的触发条件（M0.1）。Verify 子程序首先复位本身的触发条件以防止子程序被重复调用，然后求出接收缓冲区中指令的 BCC 校验码并与指令中的 BCC 校验码进行比对。如果相等则置 BCC 码校验正确的标志位（M0.0）为 1；如果指令格式正确（指令的结束标志在接收缓冲区中特定的位置 VB133）而 BCC 码不相等，则发送代表 BCC 校验码错误的反馈信息；如果指令格式不正确（VB133 中不是指令的结束标志），则返回代表指令格式错误的反馈信息。

Read 子程序的触发条件为：指令中的站地址与本机站地址相符、指令类型为读指令、BCC 检验码正确。当条件满足时，Read 子程序被执行。Read 子程序首先禁止 RCV，然后将指令所要读取的数据转换成十六进制 ASCII 码并写入发送缓冲区、计算 BCC 检验码、最后发送反馈信息。

Write 子程序的触发条件为：指令中的站地址与本机站地址相符、指令类型为写指令、BCC 检验码正确。当条件满足时，Write 子程序被执行。Write 子程序首先禁止 RCV，然后将指令中的数据写入目标寄存器，最后发送代表写入正确的反馈信息。

PLC 每接到一条指令后都会发送一条反馈信息，当反馈信息发送完成时，会产生发送完成中断，XMTcomplete 中断服务程序用来处理发送完成中断事件。在 XMTcomplete 中断服务程序中所要执行的操作包括：复位 BCC 校验码正确的标志位（M0.0）；允许 RCV；bcc 码寄存器清零；重新装入用于计算 BCC 校验码的地址指针；接收缓冲区中存放指令结束字符的字节 VB133 清零（用来判断下一条指令格式是否正确）。

PLC 寄存器地址分配

此程序占用 PLC 寄存器的 VB100－VB199，内部继电器占用 M0.0 和 M0.1。寄存器地址分配见表 3、表 4、表 5、表 6。

表 3 接收缓冲区

VB100	字符数	
VB101	起始字符	Byte0
VB102	指令类型（读/写）	Byte1
VB103	目标 PLC 站地址（十六进制 ASCII 码）	Byte2
VB104		Byte3
VB105	目标寄存器地址（十六进制 ASCII 码）	Byte4
VB106		Byte5
VB107		Byte6
VB108		Byte7
VB109		Byte8
VB110		Byte9
VB111		Byte10
VB112		Byte11
VB113	读/写字节数 M（十六进制 ASCII 码）	Byte12
VB114		Byte13

VB115	要写入的数据（十六进制 ASCII 码）	Byte14
VB116		Byte15
VB117		Byte16
VB118		Byte17
VB119		Byte18
VB120		Byte19
VB121		Byte20
VB122		Byte21
VB123		Byte22
VB124		Byte23
VB125		Byte24
VB126		Byte25
VB127		Byte26
VB128		Byte27
VB129		Byte28
VB130		Byte29
VB131	BCC 校验码（十六进制 ASCII 码）	Byte30
VB132		Byte31
VB133	结束字符	Byte32

表 4 译码区

VB134	PLC 站号（ATH from VB103-VB104）
VB135	合成为 VD135 作为目标寄存器的地址指针
VB136	（ATH from VB105－VB112）
VB137	
VB138	
VB139	读/写字节数（ATH from VB113－VB114）
VB140	Bcc 码（ATH from VB131－VB132）
VB141	未使用
VB142	
VB143	
VB144	
VB145	
VB146	
VB147	
VB148	
VB149	和成为 VD149 作为 VB102 的地址指针用以计算 BCC 校验码
VB150	
VB151	
VB152	

表 5 发送缓冲区

VB153	字符数	
VB154	起始字符	Byte0
VB155	状态信息	Byte1
VB156	数据区（十六进制 ASCII 码）	Byte2
VB157		Byte3
VB158		Byte4
VB159		Byte5
VB160		Byte6
VB161		Byte7
VB162		Byte8
VB163		Byte9
VB164		Byte10
VB165		Byte11
VB166		Byte12
VB167		Byte13
VB168		Byte14
VB169		Byte15
VB170		Byte16
VB171		Byte17
VB172	BCC 校验码（十六进制 ASCII 码）	Byte18
VB173		Byte19
VB174	结束字符	Byte20

表 6 其它

VB175	合成为 VW175
VB176	作为接收时计算 bcc 码循环的 INDX
VB177	合成为 VW177
VB178	作为发送时计算 bcc 码循环的 INDX
VB179	接收数据的 bcc 码
VB180	发送数据的 bcc 码
VB181	合成为 VD181 作为 VB156 的地址指针 （计算发送反馈信息的 bcc 码时使用）
VB182	
VB183	
VB184	
VB185 至 VB198	未使用
VB199	本机站号

程序清单

主程序：

```
NETWORK  1
LD       SM0.1                    //第一次扫描调用初始化子程序
CALL     initialize
```

```
NETWORK  2
LDB=     VB134，VB199        //指令中的站地址与本机站地址相符
AB=      VB102，5            //指令类型为读指令
A        M0.0                //BCC 码校验正确
CALL     Read                //调用读子程序

NETWORK  3
LDB=     VB134，VB199        //指令中的站地址与本机站地址相符
AB=      VB102，6            //指令类型为写指令
A        M0.0                //BCC 码校验正确
CALL     Write               //调用写子程序

NETWORK  4
LD       M0.1                //指令接收完成后调用 BCC 码校验子程序
CALL     Verify

NETWORK  5
LD       SM4.5               //当端口空闲时启动 RCV
RCV      VB100，0
```

Read 子程序：

```
NETWORK  1
LD       SM0.0               //停止端口 0 的接收
R        SM87.7，1
R        M0.0，1
RCV      VB100，0

NETWORK  2
LD       SM0.0               //将数据写入发送缓冲区
MOVB     103，VB154
MOVB     1，VB155
HTA      *VD135，VB156，16
MOVB     26，VB174
MOVB     21，VB153

NETWORK  3
LD       SM0.0               //计算 BCC 校验码
FOR      VW177，+1，+16

NETWORK  4
LD       SM0.0
XORB     *VD181，VB180
```

```
NETWORK  5
LD       SM0.0
INCD     VD181

NETWORK  6
NEXT

NETWORK  7
LD       SM0.0
HTA      VB180，VB172，2      //BCC 校验码写入发送缓冲区

NETWORK  8
LD       SM4.5                //发送反馈信息
XMT      VB153，0
```

Write 子程序：

```
NETWORK  1
LD       SM0.0                //停止端口 0 的接收
R        SM87.7，1
R        M0.0，1
RCV      VB100，0

NETWORK  2
LD       SM0.0                //装入要写如数据源的地址指针
MOVD     &VB115，VD145

NETWORK  3
LD       SM0.0                //写入数据
ATH      *VD145，*VD135，
         VB139

NETWORK  4
LD       SM0.0                //指令执行的反馈信息写入发送缓冲区
MOVB     21，VB153
MOVB     103，VB154
MOVB     2，VB155
MOVB     26，VB174

NETWORK  5
LD       SM4.5                //发送指令执行的反馈信息
XMT      VB153，0
```

Verify 子程序：

```
NETWORK  1
LD       SM0.0
R        M0.1，1              //复位 verify 子程序的执行条件

NETWORK  2
LD       SM0.0                //计算 BCC 码
FOR      VW175，＋1，＋29

NETWORK  3
LD       SM0.0
XORB     *VD149，VB179

NETWORK  4
LD       SM0.0
INCD     VD149

NETWORK  5
NEXT

NETWORK  6
LDB=     VB179，VB140         //当 BCC 码校验正确时，M0.0 置 1
AB=      VB133，71
S        M0.0，1

NETWORK  7
LDB=     VB133，71            //BCC 码错误时发送反馈信息
AB<>     VB179，VB140
MOVB     21，VB153
MOVB     103，VB154
MOVB     3，VB155
MOVB     26，VB174
R        SM87.7，1
RCV      VB100，0
XMT      VB153，0

NETWORK  8
LDB<>    VB133，71            //指令格式错误或 RCV 超时时发送反馈信息
MOVB     21，VB153
MOVB     103，VB154
MOVB     4，VB155
MOVB     26，VB174
```

```
R       SM87.7，1
RCV     VB100，0
XMT     VB153，0
```

Initialize 子程序：

```
NETWORK  1
LD       SM0.0
MOVB     9，SMB30              //0 口“9600，N，8，1”

NETWORK  2
LD       SM0.0                 //RCV 指令初始化
MOVB     16#EC，SMB87
MOVB     103，SMB88
MOVB     71，SMB89
MOVB     +1000，SMW92
MOVB     35，SMB94
R        SM87.2，1

NETWORK  3
LD       SM0.0
ATCH     RCVComplete，23       //连接口 0 接收完成的中断

NETWORK  4
LD       SM0.0
ATCH     XMTcomplete，9        //连接口 0 发送完成的中断

NETWORK  5
LD       SM0.0
ENI                            //中断允许

NETWORK  6
LD       SM0.0
MOVB     2，VB199              //将本机站地址装入寄存器

NETWORK  7
LD       SM0.0
MOVB     &VB102，VD149         //装入地址指针
MOVB     0，VB179              //BCC 码寄存器清零
MOVB     &VB156，VD181         //装入地址指针
MOVB     0，VB180              //BCC 码寄存器清零
```

RCVcomplete 中断程序

```
NETWORK  1
```

```
LD      SM0.0
ATH     VB103，VB134，2     //指令译码（ASCII 码到十六进制）
ATH     VB105，VB135，8
ATH     VB113，VB139，2
ATH     VB131，VB140，2
S       M0.1，1             //置位 Verify 子程序的触发条件
MOVB    0，VB179            //BCC 码寄存器清零
MOVD    &VB102，VD149       //装入地址指针
```

XMTcomplete 中断程序

```
NETWORK  1
LD      SM0.0
R       M0.0，1             //复位 BCC 校验码正确的标志位
S       SM87.7，1           //允许口 0 进行接收
MOVB    0，VB179            //BCC 校验码寄存器清零
MOVB    0，VB180            //BCC 校验码寄存器清零
MOVD    &VB102，VD149       //重新装入地址指针
MOVD    &VB156，VD181
MOVB    0，VB133            //接收缓冲区中存放指令结束字符的字节清零
```

H.18 两个S7-200站通过工业以太网进行通讯

以下通讯举例阐述 CP243-1 是如何作为一个客户机进行处理的。阐述了由集成在 STEP 7 Micro/WIN 32 中的 Ethernet Wizard 生成的子程序在组态结束后在程序中的应用。Ethernet Wizard 在其中保存相关组态的数据块显示在程序代码后。

在此所提供的样板程序和相关组态都作为一个 STEP 7 Micro/WIN 32 项目，包含在 QuickStart 光盘中。在运行该程序时，相应组态服务器将要在其中运行的第二个 S7-200 系统必须可以通过以太网访问。

程序代码

项目名称： Beispielprogramm_Client

版本： 01.00

日期 2002 年 7 月 25 日

目标硬件： CPU 224 和 CP 243-1 安装在槽 0 中

说明： 2 个 S7-200 站通过工业以太网中进行通讯的样板程序

保存在地址 VB200 或以上且长度为 5 个字节的字符串（“CP243”），将被从本地 S7-200 系统发送到另一个 S7-200 系统，并保存为同一地址，然后读回。该程序将反复连续执行。

相应服务器必须作为一个对应通讯对方进行组态。只运行在服务器上的用户程序必须调用 ETHx_CTRL 子程序。调用 ETHx_XFR 子程序不需运行服务器。

CRC 校验关闭。

组态（IP 地址等）必须与当前条件相适应。“CP243”字符串必须保存在地址 VB200 或以上。

网络 1

ETHT0_CTRL 子程序初始化并监控 CP 243-1。

SM0.0 状态位在输入“EN”生成。这可保证子程序总是激活。只要 CP 243-1 一完成引导装入，“CP_Ready”和“CH_Ready”输出将置位。如果在组态中出现错误，“Error”输出将被置位。出错代码将保存在标志字 2 中。它可以在状态表中读取。

```
LD      Always_On
CALL    ETH0_CTRL, cp_ready, ch_ready, cp_error
```

符号	地址	注释
Always_On	SM0.0	对于系统相关原因，总为“1”
ch_ready	MB1	
cp_error	MW2	
cp_ready	M0.0	
ETH0_CTRL	SBR1	该 POU 由 Ethernet Wizard 生成，用于 CP243-1

网络 2

只要通道 0 传输准备就绪，并且“ch0_ready”置位，“start”变量将置位。这需要几个循环。

```
LD      ch0_ready
S       start, 1
```

符号	地址	注释
ch0_ready	M1.0	ETH0_CTRL 子程序的 CH_READY 返回之中的通道 0 位
start	M4.0	自动启动数据传输触发器

网络 3

对于 S7-200，由于系统原因，“First_Scan_On”位置只在第一个循环置位。因此，“start”变量将复位。使用该程序，“start”变量可以生成一个开关沿，用于 ETHx_XFR 子程序的启动，即使对于 CPU 重复启动/停止程序。

```
LD      First_Scan_On
R       start, 1
```

符号	地址	注释
First_Scan_On	SM0.1	第一个循环中为“1”（由于系统原因），否则为“0”
start	M4.0	自动启动数据传输触发器

网络 4

本地 S7-200 站将数据从以太网连接上的 VB200（数据长度 5 个字节）发送到另一个 S7-200 站。数据将保存在 VB200 中。

如果任何先前读命令的“Done”位被置位，只有在“START”参数的上升沿后，才进行读写命令。“Done”位保存在“ch0_done_read”变量中。对于读写命令的输出，一个通道每次只能处理一个命令。因此，新命令只能在前一命令完成后进行。

“Chan_ID”参数可以规定数据访问的通道数量。在目前情况下，适用于通道 0。“Data”参数可以规定通道相关数据块的数量。使用“Abort”参数可以中止一个已触发的命令。为此，M7.0 标志必须置位。在该例中，该标志也可以用作网络 5 中的读命令。

在标志 M7.0 清零后，由于“Done”位在一个命令中止后总与“Error”位一起置位，将再次触发读/写命令。

使用专用的标志 SM0.0，可以保证 ETH0_XFR 子程序总是启用。

只要写命令一完成，在“Done”返回参数中将返回数值“1”。所出现的任何错误将通过“Error（错误）”参数返回。否则，在该参数中将返回数值“0”。

```
LD      Always_On
=       L60.0
LD      ch0_done_read
EU
U       ch0_ready
LD      start
EU
```

OLD
= L63.7
LD L60.0
CALL ETH0_XFR, L63.7, Connection0_0, Write_1, Connection_abort, ch0_done_write, ch0_error_write

符号	地址	注释
Always_On	SM0.0	对于系统相关原因，总为“1”
ch0_done_read	M5.0	读命令的“Done”位
ch0_done_write	M5.1	写命令的“Done”位
ch0_error_write	MB14	通道“0”的错误标志，写命令
ch0_ready	M1.0	ETH0_CTRL 子程序的 CH_READY 返回之中的通道 0 位
Connection0_0	VB166	

符号	地址	注释
Connection_abort	M7.0	中止传输
ETH0_XFR	SBR2	该 POU 由 Ethernet Wizard 生成，用于 CP243-1
start	M4.0	自动启动数据传输触发器
Write_1	VB167	

网络 5

本地 S7- 200 站可以从另一个 S7- 200 站的 VB200 中读取数据，并保存在本地 S7-200 站的 VB200 中。

在执行读命令之前，应计算两个变量“"ch0_done_write”和“ch0_ready”，第一，以保证先前的写命令已完成，第二已使通道 0 的状态准备就绪。

LD Always_On
= L60.0
LD ch0_done_write
EU
U ch0_ready
= L63.7
LD L60.0
CALL ETH0_XFR, L63.7, Connection0_0, Read_1, Connec-tion_abort, ch0_done_read, ch0_error_read

符号	地址	注释
Always_On	SM0.0	对于系统相关原因，总为“1”
ch0_done_read	M5.0	读命令的“Done”位
ch0_done_write	M5.1	写命令的“Done”位
ch0_error_read	MB6	通道“0”的错误标志，读命令
ch0_ready	M1.0	ETH0_CTRL 子程序的 CH_READY 返回之中的通道 0 位
Connection0_0	VB166	
Connection_abort	M7.0	中止传输
ETH0_XFR	SBR2	该 POU 由 Ethernet Wizard 生成，用于 CP243-1
Read_1	VB168	

相关组态的数据块

```
//
// 数据块注释
//
// 按 F1 键，可获得帮助和数据块举例
//
//------------------------------------------------------------
// CP243-1 以太网模板组态模块。
// 由 Ethernet Wizard 生成。
//------------------------------------------------------------
VB0                 'CP243'         // 位置为“0”的 CP243-1 以太网模板的标识号
                                    //
VW5                 16#006C         // CDB 的长度
VW7                 16#0014         // NPB 的长度
VB9                 16#01           // 组态数据版本
VB10                16#00           // 项目组态版本
VW11                16#0000
VW13                16#0004         // 自动检测通讯，用户生成
                                    // IP 地址，CRC 保护启用
VD15                16#C1012807     // 模板的 IP 地址（193.1.40.7）
VD19                16#FFFFFF00     // 模板的子网掩地址（255.255.255.0）
VD23                16#C1012801     // 网关地址（193.1.40.1）
VW27                30              // 持续作用（Keep Alive）时间间隔，单位[秒]
    //---------------------------- 连接 0
VB29                16#83           // 客户机连接，持续作用
VD30                16#C1012812     // 该连接的服务器地址
                                    //(193.1.40.18)
VW34                16#1000         // 该连接的本地 TSAP（10.00）
VW36                16#1000         // 该连接的远程 TSAP（10.00）
                                    //---------------------------- 连接 1
VB38                16#00           // 连接没有定义。
```

```
VD39            16#00000000
VW43            16#0000
VW45            16#0000
                                //---------------------------------- 连接 2
VB47            16#00           // 连接没有定义。
VD48            16#00000000
VW52            16#0000
VW54            16#0000
                                //---------------------------------- 连接 3
VB56            16#00           // 连接没有定义。
VD57            16#00000000
VW61            16#0000
VW63            16#0000
                                //---------------------------------- 连接 4
VB65            16#00           // 连接没有定义。
VD66            16#00000000
VW70            16#0000
VW72            16#0000
                                //---------------------------------- 连接 5
VB74            16#00           // 连接没有定义。
VD75            16#00000000
VW79            16#0000
VW81            16#0000

//-------------------------------------------------------Connection 6
VB83            16#00           // Connection not defined.
VD84            16#00000000
VW88            16#0000
VW90            16#0000
//-------------------------------------------------------Connection 7
VB92            16#00           // Connection not defined.
VD93            16#00000000
VW97            16#0000
VW99            16#0000
//-------------------------------------------------------STEP 7-Micro/WIN reserved connection.
VB101           16#82
VD102           16#00000000
VW106           16#641F
//-----------------------------------------------------------
// Network Parameter Block Section
```

```
// This section is used by the CP243-1 Ethernet Module //-----------------------------------------
----------
VW108                   16#0000
VD110                   16#00000000
VD114                   16#00000000
VD118                   16#00000000
VB122                   16#00
VB123                   16#00
VB124                   16#00
VB125                   16#00
VB126                   16#00
VB127                   16#00
//-------------------------------------------------------------
// Network Data Block Section
//-------------------------------------------------------------
VW128                   16#0026
VB130                   16#00
VB131                   16#03
VB132                   16#0F
VB133                   16#0F
VB134                   'W=5,VB200,VB200'       // Message 0 for Connection 0.
VB149                   'R=5,VB200,VB200'       // Message 1 for Connection 0.
VW164                   16#7E73
//-------------------------------------------------------------
// Symbol Initializations
//-------------------------------------------------------------
VB166 0
VB167 0
VB168 1
//-------------------------------------------------------------
VB200                   'CP243'  // Module ID for testing
```