SIEMENS

SIMATIC

S7-200
可编程序控制器

系统手册

订货号：
6ES7 298-8FA21-V1-5D00

03/2000
版本 02

注册商标

SIMATIC®、ProTool/Lite®、ProTool®和 ProTool/Pro®均是西门子公司的注册商标。

该文件所用的其它一些术语也都是经注册的商标，如果第三方擅自使用这些商标，则属于侵权行为。

Siemens AG
Automatisierungs- und Antriebstechnik
Bedien- u. Beobachtungssysteme
Postfach 4848, D-90327 Nuremberg

Siemens Aktiengesellschaft

拒绝承担责任的声明

已经对打印文件的内容与所描述的硬件和软件之间进行一致性检查。由于不可能完全消除不准确性，因此不能保证绝对准确。将对文件中的信息作定期检查并且在以后的修订中进行必要的修改。非常乐意接受改进建议。

订货号6AV6594-1MA05-1AB0

安装指南

本手册包括了保证人身安全与保护本产品及连接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，并按危险等极标明如下：

危险

表示如果不采取适当的预防措施，将导致死亡、严重的人身伤害或财产损失。

警告

表示如果不采取适当的预防措施，将导致死亡、严重的人身伤害或财产损失。

小心

表示如果不采取适当的预防措施，将导致死亡、严重的人身伤害或财产损失。

合格人员

设备或系统的安装与操作必须按照本手册的要求进行。只有**合格人员**才允许安装与操作此设备。合格人员定义为授权按照既定安全惯例和标准、对线路、设备和系统进行试运、接地与加标签的人。

正确应用

注意如下：

警告

此设备及其部件只能用于产品目录与技术说明中所叙述的应用，并且只可与Siemens认可和推荐的别人厂家出产的设备或部件一起使用。

只要正确地运输、保管、配置与安装，并且按照建议操作与维护，产品才能正常、安全地运行。

商标

Siemens® 和 SIMATIC® 是 SIEMENS AG. 的注册商标。

STEP 7™ 和 S7™ 是SIEMENS AG. 的注册商标。

Microsoft®，Windows®，Windows 95 ®，Windows 98® 和Windows NT® 是微软公司的注册商标。

Underwriters Laboratories® 是 Underwriters Laboratories，Inc. 的注册商标。

拒负责任的声明

我们已核对本手册的内容与所叙述的硬件和软件相符，因为差错难以避免，我们不能保证完全一致。然而，我们将经常对手册中的数据进行检查并在后续的编辑中进行必要的更正。欢迎您提出宝贵意见。

前言

目的

S7-200 系列小型 PLC（Micro PLC）可以应用于各种自动化系统。结构紧凑，低成本以及功能强大的指令集，使得 S7-200 控制器是各种小型控制任务理想的解决方案。多种多样的 CPU 尺寸及电压范围以及基于 Windows 的编程工具，使您更加灵活、方便地解决自动化任务。

S7-200 产品系列设计更加小巧、指令执行更加快速、功能性更加提高，新一代的 S7-200 产品可以替代以前的产品。

该手册提供有关 S7-200 Micro PLC 的安装和编程信息。它包括以下内容：

- 安装和接线
- 了解 PLC 的运行、数据类型、编址方式、扫描周期、口令保护和网络通讯
- 技术特点
- SIMATIC 和 IEC 1131-3 编程指令的描述和示例
- 使用 USS 协议指令与驱动部件通讯
- SIMATIC STL 指令典型的执行时间

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的编写。

适用范围

本手册包括以下产品信息：

- S7-200 CPU：CPU 221，CPU 222、CPU 224 和 CPU 226
- STEP 7-Micro/WIN 32，3.1 版，在Windows 95，Windows 98 和 Windows NT 环境下使用的 32 位编程软件包。
- STEP 7 Micro/WIN 32软件工具包，在Windows 95，Windows 98 和 Windows NT 4.0 环境下使用的 32 位编程软件包。 该工具软件包为 S7-200 CPU 与其它微型系统部件的使用提供了便利（例如 TP 070 触摸屏或 Micro Master 变频器）。

认证

SIMATIC S7-200 系列满足以下规范：

- European Community (CE) Low Voltage Directive 73/23/EEC
- European Community (CE) EMC Diective 89/336EEC
- Underwriters Laboratories, Inc: UL 508 Listed (Industrial Control Equipment)
- Canadian Standards Association: CSA C22.2 Number 142 Certified (Process Control Equipment)
- Factory Mutual Research: FM Class 1, Division 2, Groups A, B, C&D Hazardous Locations, T4A 和 Class 1, Zone 2, 11C, T4。

详细信息参见附录 A。

相关信息

请参考以下详细信息

- STEP 7-Micro/WIN 32 CD 盘：提供在线帮助，STEP 7-Micro/WIN 使用入门（可打印在线手册），和应用示例。
- STEP 7 Micro/WIN 32 工具软件包 CD 盘：提供 TP 070 触摸屏组态软件，USS 协议指令，在线帮助，STEP 7 Micro/WIN 使用入门（可打印在线手册），和应用示例。
- 过程现场总线（PROFIBUS）标准（EN 50170）：描述 S7-200 DP 通讯能力的标准协议。
- TD 200 操作员界面用户手册：描述如何安装和使用 TD 200。
  （有关该手册的中文版电子文件请与西门子公司联系）

如何使用本手册

如果您初次使用 S7-200，您需要通读该手册。如果您是一位有经验的用户，请通过目录及索引查找相应信息。

S7-200 系统手册按以下主题编排：

- “S7-200 Micro PLC 概述”（第 1 章）提供该产品的特点概述。
- “S7-200 PLC 的安装（第 2 章）描述 S7-200 CPU 模块及扩展 I/O 模块的安装步骤、尺寸和基本安装指南。
- “S7-200 编程系统使用入门”（第 3 章）描述如何建立 S7-200 编程系统
- “S7-200 CPU 编程的基本概念”（第4章）、“CPU 存储器的数据类型及寻址式式”（第 5 章）和“CPU 和输入/输出控制组态”（第 6 章）提供 S7-200 CPU 处理数据和执行程序的有关信息。
- “设置通讯硬件和网络通讯”（第 7 章）提供如何安装和拆除通讯硬件以及如何将 S7-200 CPU 连接不同类型网络的信息。
- “S7-200 指令”（第 8 章）提供不同编程语言概念和规约的概述。
- 第 9 章提供 SIMATIC LAD、FBD 和 STL 编程指令的描述和示例。
- 第 10 章提供 IEC 1131-3 LAD 和 FBD 编程指令的描述和示例。
- 第 11 章提供“使用 USS 协议指令与驱动部件进行通讯”。提供 USS 协议指令的描述和示例），以及如何使用这些指令与驱动部件进行通讯的信息。
- 第12章提供了TD200的组态方法。
- 附录H章为用户特别提供了部分S7-200的应用示例，供用户参考。

在附录中提供其它信息（包括设备技术规范，错误代码描述，故障检测和 STL 指令执行时间）。

其它帮助

有关技术支持、产品培训及产品订货的有关事宜，请与当地西门子公司及分销商联系。

http://www.ad.siemens.de	西门子公司的通用信息
http://www.siemens.com/s7200	S7-200 产品信息

目录

S7-200 Micro PLC 的概述

1

S7-200 系列是一类可编程逻辑控制器 (Micro PLC)。这一系列产品可以满足多种多样的自动化控制需要，图1-1展示一台 S7-200 Micro PLC。由于具有紧凑的设计、良好的扩展性、低廉的价格以及强大的指令，使得 S7-200 可以近乎完美地满足小规模的控制要求。此外，丰富的 CPU 类型和电压等级使其在解决用户的工业自动化问题时,具有很强的适应性。

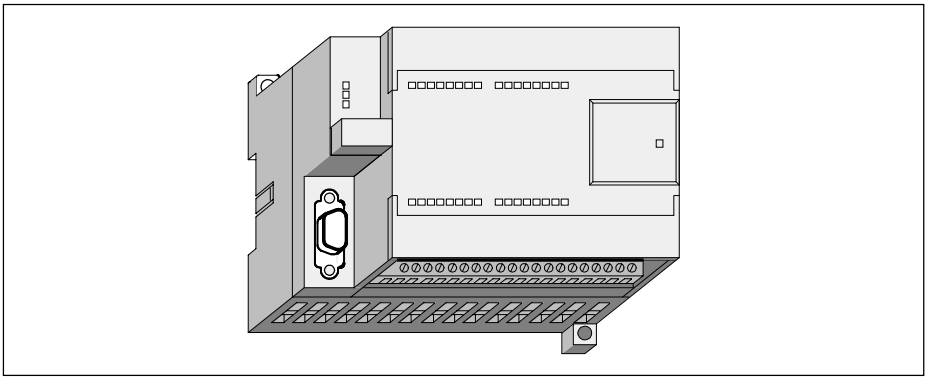


图 1-1 S7-200 Micro PLC

本章概述

节	内 容	页
1.1	S7-200 Micro PLC 系列的性能比较	1-2
1.2	S7-200 Micro PLC 主要组成部分	1-4
1.3	最大 I/O 配置	1-5

1.1 S7-200 Micro PLC 系列的性能比较

所需设备

图 1-2 展示了一个基本的 S7-200 Micro PLC。它包括一个 S7-200 CPU 模块，一台个人计算机 (PC)，STEP 7-Micro/WIN 32 (3.1版) 编程软件，以及一条通讯电缆。

为了使用个人计算机 (PC)，你必须以下一种设备：

- 一条 PC/PPI 电缆
- 一个通讯处理器 (CP) 和多点接口 (MPI) 电缆
- 一块 MPI 卡，随 MPI 卡提供一根通讯电缆。

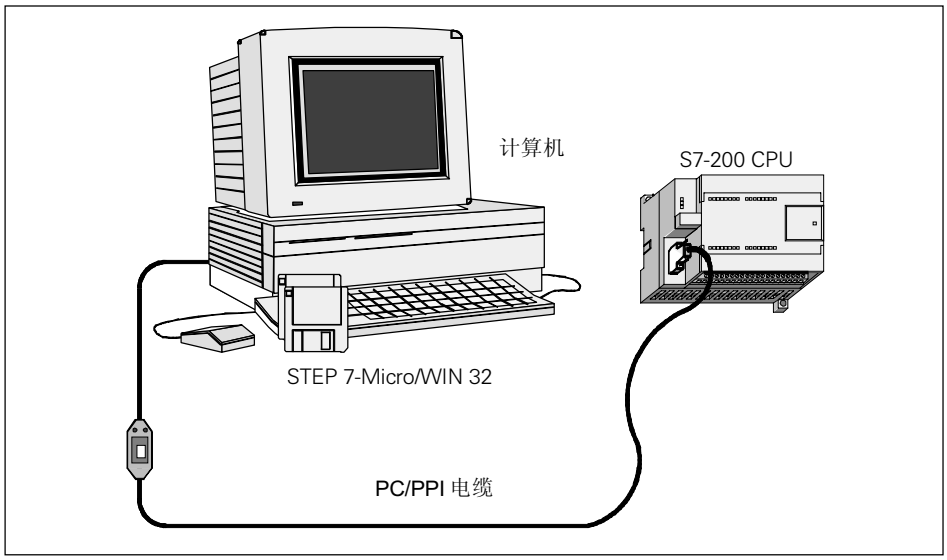


图 1-2 S7-200 Micro PLC 系统的组成

S7-200 CPU 的性能

S7-200 家族包括多种 CPU，这使得工业自动控制的设计工作得到支持，表 1-1提供了每一种 S7-200 CPU 的主要特性。

表 1-1 S7-200 CPU 主要技术指标

特性	CPU 221	CPU 222	CPU 224	CPU 226
外形尺寸 mm	90 x 80 x 62	90 x 80 x 62	120.5 x 80 x 62	190 x 80 x 62
存储器				
程序	2048 字	2048 字	4096 字	4096 字
用户数据	1024 字	1024 字	2560 字	2560 字
用户存储器类型	EEPROM	EEPROM	EEPROM	EEPROM
数据后备 (超级电容) 典型值	50 小时	50 小时	190 小时	190 小时
本机 I/O				
本机 I/O	6 入 / 4 出	8 入 / 6 出	14 入 / 10 出	24 入 / 16 出
扩展模块数量	无	2 个模块	7 个模块	7 个模块
I/O总计				
数字量 I/O 映象区大小	256 (128 入 /128 出)	256 (128 入 /128 出)	256 (128 入 /128 出)	256 (128 入 /128 出)
模拟量 I/O 映象区大小	无	16 入 /16 出	32 入 /32 出	32 入 /32 出
CPU的映象寄存器大小、模块数量、5V电源容量不同产品I/O点的物理数量决定了实际的I/O点数量。见1.3节。				
指令				
33MHz下布尔指令执行速度	0.37 μs/ 指令	0.37 μs/ 指令	0.37 μs/ 指令	0.37 μs/ 指令
I/O映象寄存器	128 I 和 128 Q	128 I 和 128 Q	128 I 和 128 Q	128 I 和 128 Q
内部继电器	256	256	256	256
计数器/定时器	256/256	256/256	256/256	256/256
字入/字出	无	16/16	32/32	32/32
顺序控制继电器	256	256	256	256
For/Next 循环	有	有	有	有
整数运算 (+ - ×/)	有	有	有	有
实数运算 (+ - ×/)	有	有	有	有
附加功能				
内置高速计数器	4H/W (20 KHz)	4H/W (20 KHz)	6H/W (20 KHz)	6H/W (20 KHz)
模拟量调节电位器	1	1	2	2
脉冲输出	2 (20 KHz, DC)	2 (20 KHz, DC)	2 (20 KHz, DC)	2 (20 KHz, DC)
通讯中断	1 发送器 / 2 接收器	1 发送器 / 2 接收器	1 发送器 / 2 接收器	2 发送器 / 4 接收器
定时中断	2 (1 ms ~ 255 ms)	2 (1 ms ~ 255 ms)	2 (1 ms ~ 255 ms)	2 (1 ms ~ 255 ms)
硬件输入中断	4，输入滤波器	4，输入滤波器	4，输入滤波器	4，输入滤波器
实时时钟	有 (时钟卡)	有 (时钟卡)	有 (内置)	有 (内置)
口令保护	有	有	有	有
通讯				
通讯口数量	1 (RS–485)	1 (RS-485)	1 (RS-485)	2 (RS-485)
支持协议 0号口： 1号口：	 PPI, DP/T,自由口 N/A	 PPI, DP/T,自由口 N/A	 PPI, DP/T,自由口 N/A	 PPI, DP/T,自由口 PPI, DP/T,自由口
PROFIBUS 点到点	(NETR/NETW)	(NETR/NETW)	(NETR/NETW)	(NETR/NETW)

1.2 S7–200 Micro PLC 主要组成部分

1台S7–200 Micro PLC包括一个单独的S7–200 CPU，或者带有各种各样的可选扩展模块。

S7–200 CPU

S7-200 CPU 模块包括一个中央处理单元 (CPU) 、电源以及数字量 I/O 点，这些都被集成在一个紧凑、独立的设备中。

- CPU 负责执行程序和存储数据，以便对工业自动控制任务或过程进行控制。
- 输入和输出是系统的控制点：输入部分从现场设备(例如传感器或开关)中采集信号，输出部分则控制泵、电机、以及工业过程中的其它设备。
- 电源向CPU及其所连接的任何模块提供电力。
- 通讯端口允许将 S7-200 CPU 同编程器或其它一些设备连接起来。
- 状态信号灯显示了 CPU 的工作模式 (运行或停止)，本机 I/O 的当前状态，以及检查出的系统错误。
- 通过扩展模块可增加CPU的 I/O点数 (CPU 221不可扩展)。
- 通过扩展模块可提供其通讯性能。
- 一些 CPU 具有内置的实时时钟，其它 CPU 则需要实时时钟卡。
- EEPROM 卡可以存储 CPU 程序，也可以将一个 CPU 中的程序传送到另一个 CPU 中。
- 通过可选的插入式电池盒可延长RAM中的数据存储时间

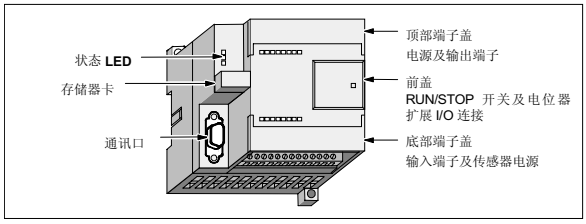


图 1-3 S7-200 CPU

扩展模块

S7-200 CPU模块提供了一定数量的本机 I/O，扩展模块提供了附加的输入输出点 (见图 1-4)。

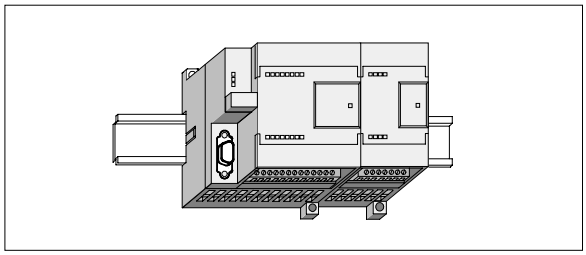


图 1-4 带有扩展模块的 CPU

1.3 最大 I/O 配置

每种CPU的最大I/O配置须服从以下限制：

- 模块数量

  CPU 221: 不能扩展

  CPU 222: 最多2个扩展模块

  CPU 224 和 CPU 226: 最多7个扩展模块

  7个模块中最多能有2个智能扩展模块 (EM 277 PROFIBUS-DP模块)。

- 数字量映象寄存器大小：每个 CPU 允许的数字量 I/O 的逻辑空间为 128 个输入和 128 个输出。由于该逻辑空间按8点模块分配，因此有些物理点无法被寻址。一个特殊模块可能不能全部寻址8个点，例如 CPU 224有10个输出点，但它占用逻辑输出区的16个点地址。一个4入/4出模块占用逻辑空间的8个输入点和8个输出点。
- 模拟量映象寄存器大小：模拟量 I/O 允许的逻辑空间为:

  CPU 222：16 输入和 16 输出

  CPU 224 和 CPU 226：32 输入和 32 输出
- 5V 电源预算：表1-2列出了每种 CPU 所能提供的最大 5V 电流。在一个系统中的所有扩展模块的电流总和不能超过该预算。详见 2.5 节。

表1-3 S7-200 CPU 所提供的电流

CPU 22x 为扩展 I/O 提供的5VDC电流－mA		扩展模块 5VDC电流消耗－mA	
CPU 222	340	EM 221 DI8 x DC24V	30
CPU 224	660	EM 222 DO8 x DC24V	50
CPU 226	1000	EM 222 DO8 x 继电器	40
		EM 223 DI4/DO4 x DC24V	40
		EM 223 DI4/DO4 x DC24V/继电器	40
		EM 223 DI8/DO8 x DC24V	80
		EM 223 DI8/DO8 x DC24V/继电器	80
		EM 223 DI16/DO16 x DC24V	160
		EM 223 DI16/DO16 x DC24V/继电器	150
		EM 231 AI4 x 12 位	20
		EM 231 AI4 x 热电耦	60
		EM 231 AI4 x RTD	60
		EM 232 AQ2 x 12 位	20
		EM 235 AI4I/AQ1 x 12 位	30
		EM 277 PROFIBUS-DP	150

表1-3　S7-200 CPU的最大 I/O 配置

模　　块	5V mA	数字量输入	数字量输出	模拟量输入	模拟量输出
CPU 221	**不能扩展**				
CPU 222					
最大数字输入/输出					
CPU	340	8	6		
2×EM 223 DI16/DO16×24VDC或者	-320或者	32	32		
2×EM 223 DI16/DO16×24VDC/继电器	-300				
总和=	**>0**	**40**	**38**		
最大模拟量输入					
CPU	340	8	6		
2×EM 235 AI4/AQ1	-60			8	2
总和=	**>0**	**8**	**6**	**8**	**2**
最大模拟量输出					
CPU	340	8	6		
2×EM 232 AQ2	-40			0	4
总和=	**>0**	**8**	**6**	**0**	**4**
CPU 224					
最大数字量输入/继电器输出					
CPU	660	14	10		
4×EM 223 DI16/DO16×24VDC/继电器	-600	64	64		
2×EM 221 DI8×24VDC	-60	16			
总和=	**=0**	**94**	**74**		
最大数字量输入/DC输出					
CPU	660	14	10		
4×EM 223 DI16/DO16×24VDC	-640	64	64		
总和=	**>0**	**78**	**74**		
数字量输入/最大继电器输出					
CPU	660	14	10		
4×EM 223 DI16/DO16×24VDC/继电器	-600	64	64		
1×EM 222 DO8×继电器	-40		8		
总和=	**>0**	**78**	**82**		
CPU 226					
最大数字量输入/继电器输出					
CPU	1000	24	16		
6×EM 223 DI16/DO16×24VDC/继电器	-900	96	96		
1×EM 223 DI8/DO8×24VDC/继电器	-80	8	8		
总和=	**>0**	**128**	**120**		
最大数字量输入/DC输出					
CPU	1000	24	16		
6×EM 223 DI16/DO16×24VDC	-960	96	96		
1×EM 221 DI8×24VDC	-30	8			
总和=	**>0**	**128**	**112**		
CPU 224或CPU 226					
最大模拟量输入					
CPU	>660	14(24)	10(16)		
7×EM 235 AI4/AQ1	-210			28	7
总和=	**>0**	**14(24)**	**10(16)**	**28**	**7**
最大模拟量输出					
CPU	>660	14(24)	10(16)		
7×EM 232 AQ2	-140			0	14
总和=	**>0**	**14(24)**	**10(16)**	**0**	**14**

S7-200 PLC 的安装

2

S7-200 PLC 设备设计成安装简便。可以利用安装孔把模块固定在控制柜的衬板上，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 系统安装和接线的指导。

本章概述

2.1 安装面板的布置

安装方法

S7-200 PLC 既可以安装在一块面板上，又可以安装在一个 DIN 导轨上。你可以把 S7-200 PLC 以垂直或水平的方向固定起来。可以通过下面的一种方法把S7–200 PLC 连接到扩展模块：

- 利用总线连接电缆可以很容易地把 I/O 模块和 PLC 或其它的扩展模块连接在一起。
- 利用 I/O 扩展电缆可以使固定方案的适应性增强。

图 2–1 揭示了一个典型的安装方式。

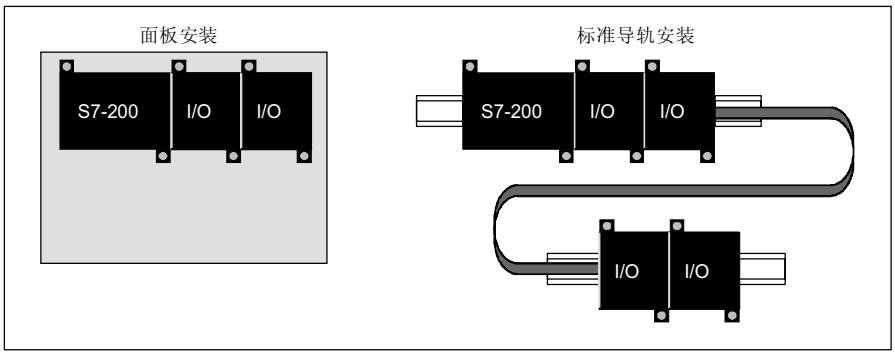


图 2-1 安装方法

安装 S7– 200 PLC 的空间要求

你可以采纳以下的安装指南：

- S7-200 CPU 和扩展模块采用自然对流散热方式，在每个单元的上方和下方都必须留有 25 mm（ 1 英寸 ）的空间，以便于正常的散热，参见图 2-2。所有的电气产品在最大负荷以及极限环境温度下连续工作，都会减少它们的使用寿命。
- 如果垂直安装，控制柜内的最高温度应该降低 10℃。并且 CPU 应该安装在其它模块的下方。如果要安装在垂直导轨上，应该使用 DIN 导轨固定端子。
- 必须使板间深度保持 75 mm，参见图 2-2。
- 在你的安装方案中，还要留出足够的空间以便容纳 I/O 线以及通信电缆。

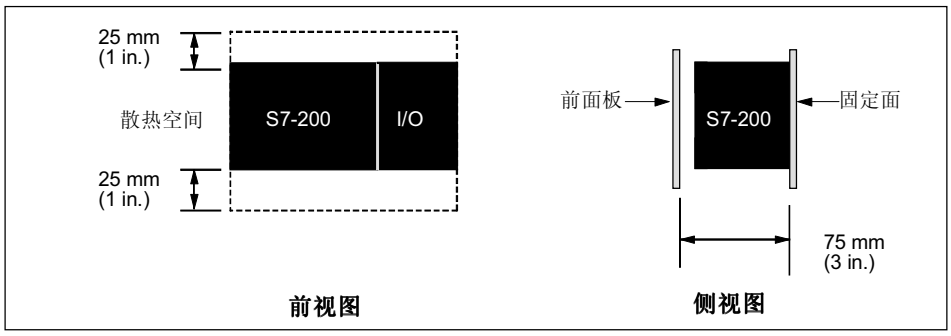


图 2-2 安装 S7-200 PLC 的水平和垂直空间要求

标准 DIN 导轨的要求

S7-200 CPU 以及扩展模块可以安装在一个标准的 DIN 导轨上 (DIN EN 50 022)，图 2-3 为 DIN 导轨的尺寸。

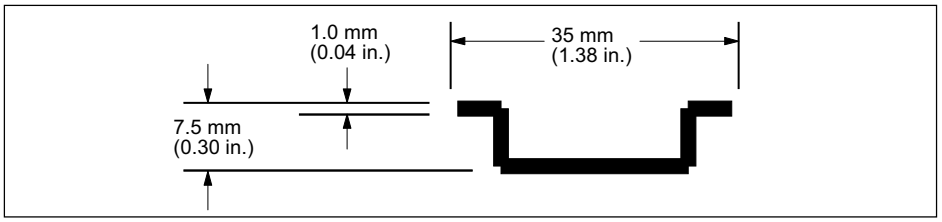


图 2-3　标准 DIN 导轨尺寸

面板固定尺寸

S7–200 CPU 及其扩展模块都具有固定孔便于安装在面板上，图 2-4 到 2-7 提供了不同的 S7–200 CPU 模块的固定尺寸。

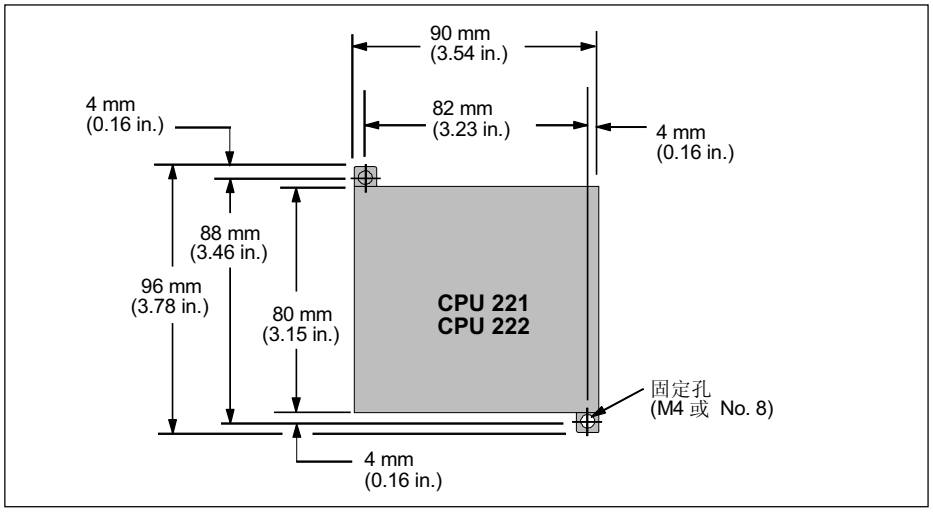


图 2-4　CPU 221 和 CPU 222 的安装尺寸

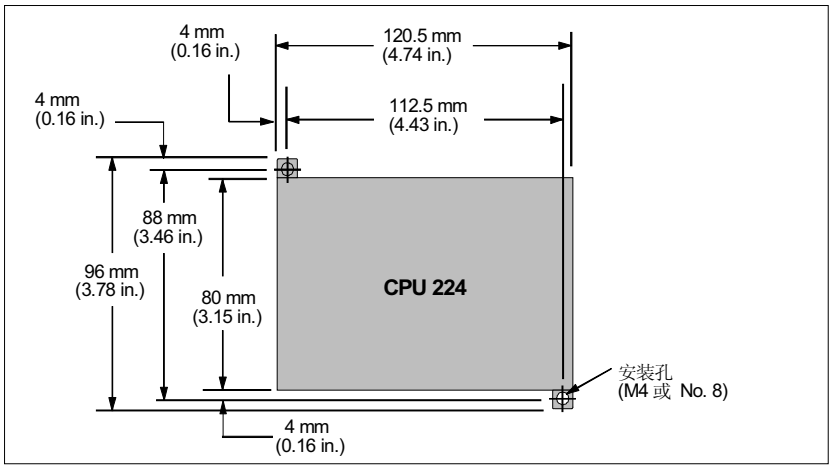


图 2-5　CPU 224 的安装尺寸

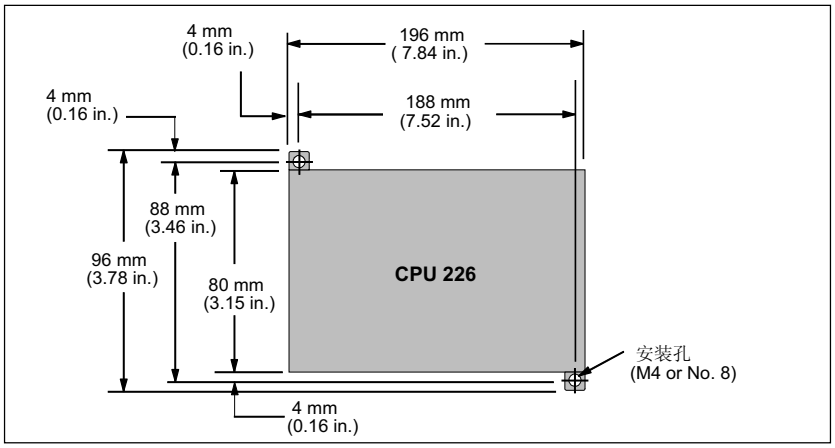


图 2-6　CPU 226 的安装尺寸

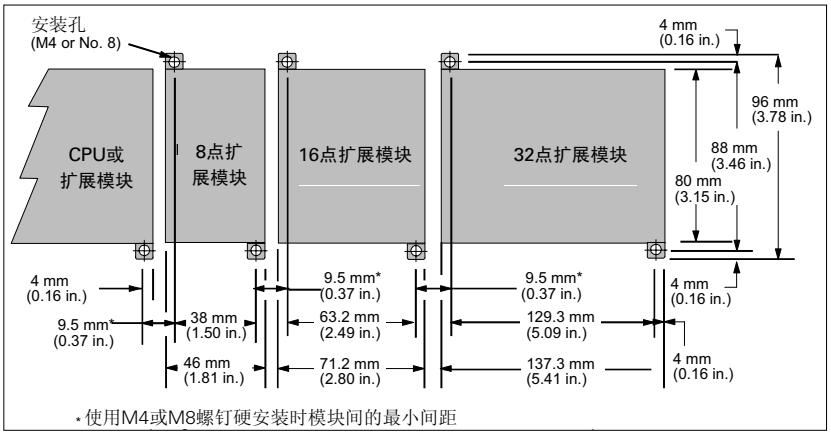


图 2-7　扩展模块的安装尺寸

2.2 S7–200 Micro PLC 或扩展模块的安装和拆卸

在面板上固定 S7-200 Micro PLC

警告：

在安装或拆卸 S7-200 模块及其相关设备时，如果没有切断电源，就有可能导致严重的人身伤害或损坏设备。因此，在安装和移动 S7-200 模块前，一定要切断所有的电源，并且要随时随地注意这一点。

S7-200 CPU 的安装步骤：

1. 用 DIN M4 或美国标准 8 号螺钉，将固定孔定位、打眼。可以参照 2-1 节中的固定方案、或其它的方法。
2. 用 DIN M4 或美国标准 8 号螺钉将 S7-200 模块固定在安装面板上。

如果你安装一个扩展模块，请遵从以下步骤：

1. 用 DIN M4 或美国标准 8 号螺钉，将固定孔定位、打眼。可以参照 2-1 节中的固定方案、或其它的方法。
2. 把 I/O 模块放到 PLC 或扩展模块的侧面，并固定住。
3. 把扩展模块的电缆插到 CPU 前盖下的连接器上，要保证正确的电缆方向。

完成安装。

在标准 DIN 导轨上安装 S7–200 Micro PLC 或扩展模块

警告：

在安装或拆卸 S7-200 模块及其相关设备时，如果没有切断电源，就有可能导致严重的人身伤害或损坏设备。因此，在安装和移动 S7-200 模块前，一定要切断所有的电源，并且要随时随地注意这一点。

安装 S7-200 CPU 模块的步骤：

1. 将 DIN 导轨每隔 75 mm (3.0 英寸)一个固定在安装面板上。
2. 打开位于模块底部的 DIN 夹子，将模块背面嵌在 DIN 导轨上。
3. 合上 DIN 夹子，仔细检查模块上 DIN 夹子与 DIN 导轨是否紧密的固定好。

安装扩展模块的步骤如下：

1. 打开 DIN 夹子，紧靠 CPU 或扩展模块把需要扩展的模块背挂到导轨上。
2. 合上 DIN 夹子，把扩展模块固定到导轨上。仔细检查模块上 DIN 夹子与DIN 导轨是否紧密的固定好。
3. 把扩展模块的电缆插到 CPU 前盖下的连接器上，要保证正确的电缆方向。
4. 安装完成。

注意：

当模块周围有潜在的振动源，或是模块要垂直安装时，就需要一个 DIN 导轨卡子。

拆卸 S7–200 Micro PLC 或扩展模块

警告：

在安装或拆卸 S7-200 模块及其相关设备时，如果没有切断电源，就有可能导致严重的人身伤害或损坏设备。因此，在安装和移动 S7-200 模块前，一定要切断所有的电源，并且要随时随地注意这一点。

拆卸 S7-200 模块应遵从以下步骤：

1. 拆除与你所要拆卸的模块相连的所有接线及电缆线。见图 2–8。一些 CPU 和扩展模块带有可移动的连接器。
2. 打开前盖，拆除临近模块的连接电缆。
3. 松开固定螺丝或打开 DIN 夹子，然后取下模块。

警告：

如果你安装了不正确的模块，那么 Micro PLC 内部的程序可能会产生错误的功能。错误地替换扩展模块和扩展电缆线将会导致严重的人身伤害或损坏设备。要用相同型号的模块进行相互替换，并且正确的定位。

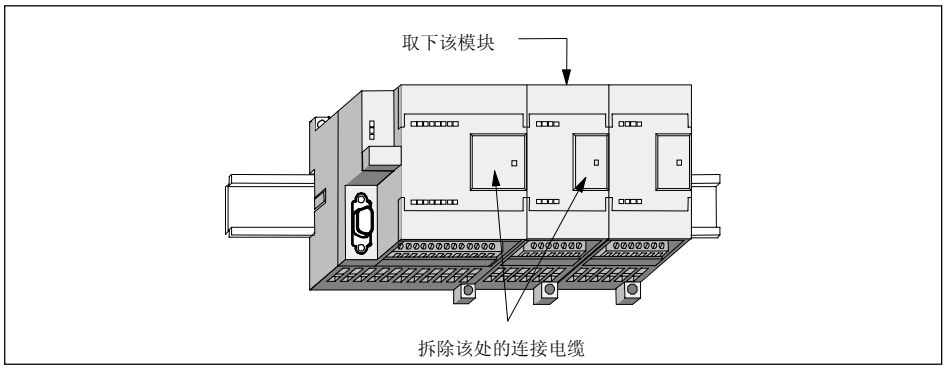


图 2-8　　拆卸扩展模块

2.3 安装现场线

警告：

在安装或拆卸 S7-200 模块及其相关设备时，如果没有切断电源，就有可能导致严重的人身伤害或损坏设备。因此，在安装和移动 S7-200 模块前，一定要切断所有的电源，并且要随时随地注意这一点。

一般性指导

以下几点是 S7-200 Micro PLC 设计安装和现场接线的一般方法：

- 在对 S7-200 PLC 接线时要确保遵从所有有效的电气编号。安装和操作所有设备要符合所有生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。
- 使用正确的导线。S7-200 模块采用的是 1.50 mm^2 ~ 0.50 mm^2 的导线 (14到 22 AWG)。
- 不要将连接器的螺钉拧得过紧，最大的扭矩不要超过 0.56Nm (5英寸镑)。
- 尽量使用短导线 (最长 500 米屏蔽线，或 300 米非屏蔽线)，导线要尽量成对使用，用一根中性或公共导线与一根热线或信号线相配对。
- 将交流线和高能量快速开关的直流线与低能量的信号线隔开。
- 正确地识别和划分 S7-200 模块的接线端子，并在线端留缓冲线圈。关于接线端子的更详细的信息，可以参考附录 A 的性能参数汇编。
- 针对闪电式浪涌，安装合适的浪涌抑制设备。
- 外部电源不要与 DC 输出点并联用作输出负载，这可能导致反向电流冲击输出，除非在安装时使用二极管或其它隔离栅。

警告：

控制设备在不安全条件下可能会失灵，导致被控制设备的误操作。这样的误动作会导致死亡或严重的人身伤害和严重损坏设备。可以考虑使用独立于可编程逻辑控制器的紧急停机功能，机电过载保护设备，或其它冗余保护。

使用隔离电路时的接地与电路参考点指南

使用隔离电路时的接地与电路参考点应遵循以下几点：

- 你应为每一个安装电路选一个参考点 (0V)，这些不同的参考点可能会连在一起，这种连接可能会导致预想不到的电流，它们会导致逻辑错误或损坏电路。产生不同参考电势的原因，通常是由于接地点在物理区域上被分隔的太远。当相距很远的设备被通讯电缆或传感器连接起来的时候，由电缆线和地之间产生的电流就会流经整个电路。即使在很短的距离内，大型设备的负载电流也可以在其与地电势之间产生变化，或者通过电磁作用直接产生不可预知的电流。那些不正确选定参考点的电源，相互之间的电路中有可能产生毁灭性的电流，以致破坏设备。
- 当把几个具有不同地电位的 CPU 连到一个 PPI 网络时，应该采用隔离的RS-485 中继器。
- S7-200 产品已在特定点上安装了隔离元件，以防止安装中所不期望的电流产生。当你打算安装时，应考虑到哪些地方有这些隔离元件，哪些地方没有。同时你也应考虑到相关电源之间的隔离以及其它设备的隔离，还有相关电源的参考点都在什么地方。
- 你最好选择一个接地参考点，并且用隔离元件来破坏可能产生不可预知电流的无用的电流回路。请记住在暂时性连接中可能引入新的电路参考点，比如说编程设备与 CPU 连接的时候。
- 在现场接地时，一定要随时注意接地的安全性，并且要正确地操作隔离保护设备。
- 在大部分的安装中，如果把传感器的供电M端子接到地上可以获得最佳的噪声抑制。

下面的概述是 S7-200 的隔离特性，但某些特性对于特殊产品可能会有所不同。请参考附录A的表格，可以查到你的产品的电路中包含哪些隔离元件及它们的隔离级别。级别小于 1500 VAC 的隔离元件只能作功能隔离，而不能用作安全隔离层。

- CPU 逻辑参考点与 DC 传感器提供的 M 点类似。
- CPU 逻辑参考点与采用 DC 电源供电的 CPU 输入电源提供的M点类似。
- CPU 通讯端口与 CPU 逻辑口具有同样的参考点。
- 模拟输入及输出与 CPU 逻辑不隔离，模拟输入采用差动输入并提供低压公共模式的滤波电路。
- 逻辑电路与地之间的隔离为 500 VAC。
- DC 数字输入和输出与 CPU 逻辑之间的隔离为 500 VAC。
- DC 数字 I/O 组的点之间隔离为 500 VAC。
- 继电器输出、AC 输出和输入与 CPU 逻辑之间的隔离为 1500 VAC。
- 继电器输出组的点之间隔离为 1500 VAC。
- AC 电源线和零线与地、CPU 逻辑以及所有的 I/O 之间的隔离为1500 VAC。

可选的现场接线端子排

选用现场接线端子排 (见图 2-9) 使得现场接线在再安装和拆卸 S7-200 时可以保持相对固定。参照附录E的 S7-200 订货号来选用现场接线端子排。

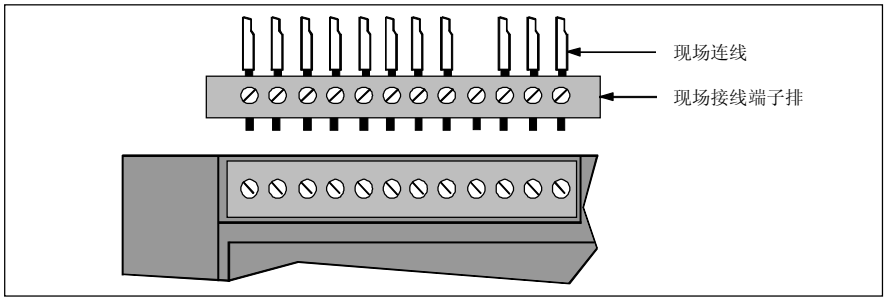


图 2-9　可选的现场接线端子排

采用可拆卸的端子连接器

采用可拆卸的端子连接器 (图 2-10) 可以保证当拆卸和重新安装 S7-200 CPU 和I/O 模块时现场接线固定不变。

按照下面的步骤从 CPU 或扩展模块上取下端子连接器：

1. 抬起 CPU 或扩展模块的端子上盖。
2. 如图 2-10 所示，把螺丝刀插入端子块中央的槽口中。
3. 如下所示，用力向下压并撬出端子连接器。

按照下面的步骤把端子连接器装入 CPU 或扩展模块：

1. 抬起 CPU 或扩展模块的端子上盖。
2. 确保新的端子连接器的引线和 CPU 或扩展模块上的引线相符合。
3. 把端子连接器向下压入 CPU 或扩展模块，直到连接器被扣住。

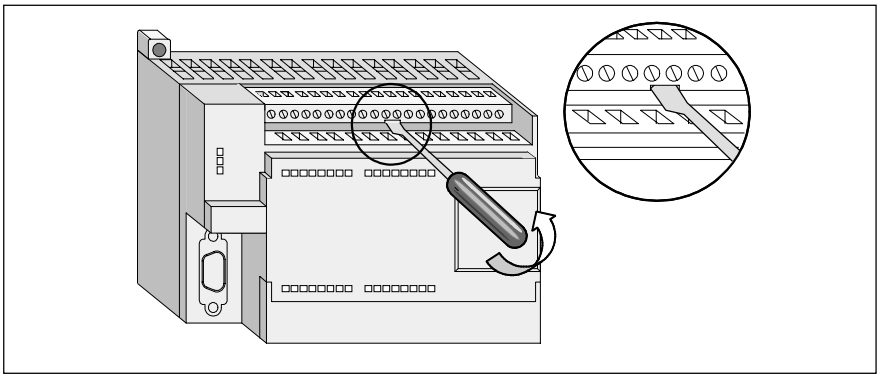


图 2-10　CPU 224 或 I/O 扩展模块的可拆卸端子连接器

交流安装指南

下列条目是 AC 交流接线安装时的一般性指南。文中括孤编号请参见图 2-11。

[a] 用一个单刀切断开关将电源与 CPU、所有的输入电路和输出（负载）电路隔离开。

[b] 用一台过流保护设备以保护 CPU 的电源、输出点以及输入点。你也可以为每个输出点加上保险丝进行范围更广的保护。

[c] 当你使用 Micro PLC 24V DC 传感器电源时，可以取消输入点的外部过流保护，因为该传感器电源具有短路保护功能。

[d] 将 S7-200 的所有地线端子同最近接地点相连接，以获得最好的抗干扰能力。建议所有的接地端子都使用 14 AWG 或 1.5 mm^2 的电线连接到独立导电点上 (亦称一点接地)。

[e] 本机单元的直流传感器电源可用来为本机单元的输入。

[f] 扩展 DC 输入以及 [g] 扩展继电器线圈供电，这一传感器电源具有短路保护功能。

[h] 在大部分的安装中，如果把传感器的供电 M 端子接到地上可以获得最佳的噪声抑制。

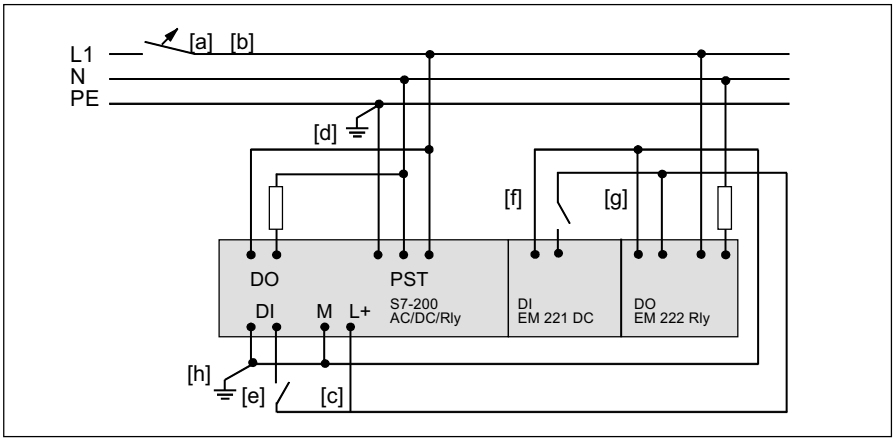


图 2-11　120 / 230VAC 使用单相过流保护开关保护 CPU 和负载电路

直流安装指南

下列条目是 DC 隔离安装接线的一般性指南，文中括孤编号请参见图 2-12。

[a] 用一个单刀开关 (a) 将电源同 CPU、所有的输入电路和输出（负载）电路隔离开。

[b] 用过流保护设备以保护 CPU 电源，[c] 输出点,以及 [d] 输入点。你也可以在每个输出点加上保险丝进行过流防护。当你使用 Micro 24 VDC 传感器电源时，可以取消输入点的外部过流保护，因为传感器电源内部具有限流功能。

[e] 确保 DC 电源有足够的抗冲击能力，以保证在负载突变时，可以维持一个稳定的电压，这时需要一个外部电容。

[f] 在大部分的应用中，把所有的 DC 电源接到地可以得到最佳的噪声抑制。在未接地 DC 电源的公共端与保护地之间并联电阻与电容 (g)。电阻提供了静电释放通路，电容提供高频噪声通路，它们的典型值是 1MΩ 和 4700pf。

[h] 将 S7-200 所有的接地端子同最近接地点 (h) 连接，以获得最好的抗干扰能力。建议所有的接地端子都使用 14 AWG 或 1.5 mm^2 的电线连接到独立导电点上(亦称一 点接地)。

24 VDC 电源回路与设备之间，以及 120/230 VAC 电源与危险环境之间，必须提供安全电气隔离。

下面是安全隔离的一些标准：

- PELV (超低电压保护) 依据 EN 60204 -1
- 2 级或电压/电流限制电路依据 UL 508

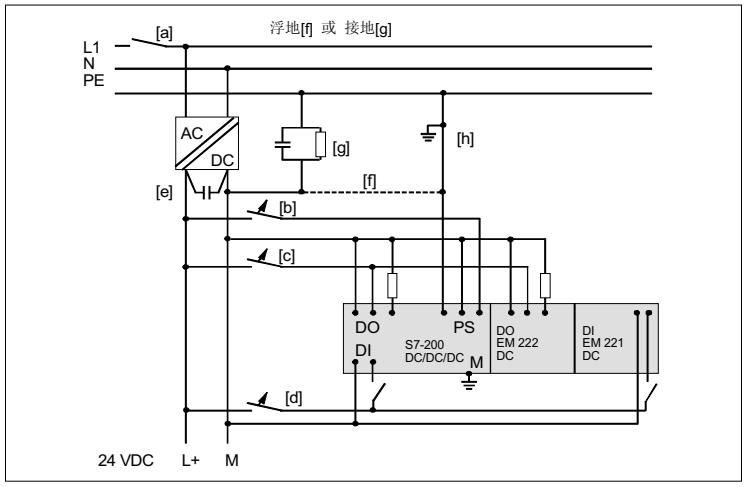


图 2-12 DC 系统的安装

2.4 抑制电路的使用

一般性指导

在感性负载中要加入抑制电路以限制在关闭电源时电压的升高。可以采用下面的指导来设计合适的抑制电路。设计的有效性取决于实际的应用，所以你必须调整其参数以适应你的应用。要保证所有的器件参数与实际应用相符合。

直流晶体管的保护

S7-200 直流晶体管输出包括了适应多种安装的齐纳二极管，对于大电感或频繁开关的感性负载可以使用外部抑制二极管来防止击穿内部二极管。图 2-13 和 2-14 所示为直流晶体管输出的典型应用。

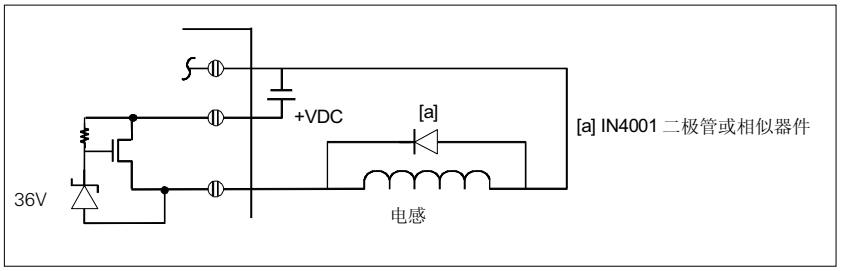


图 2-13 直流晶体管输出的普通二极管抑制

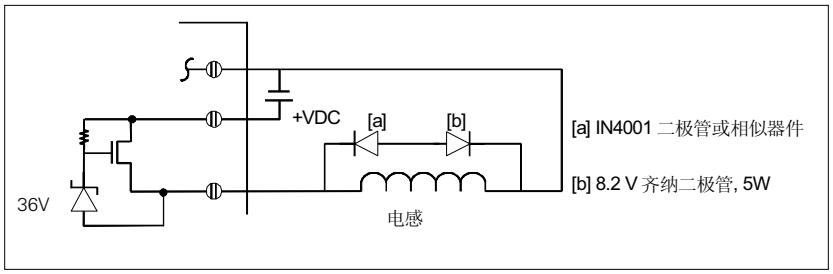


图 2-14 直流晶体管输出的齐纳二极管抑制

继电器控制直流电源的保护

如图 2-15 所示的电阻/电容网络能用于低压 (30V) 直流继电器电路，将电阻/电容网络与负载跨接。

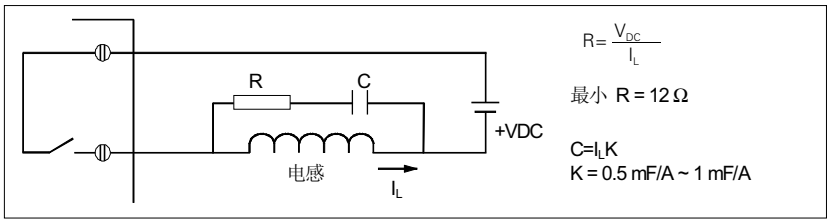


图 2-15 继电器驱动 DC 负载上跨接电阻电容网络保护电路

你也可以使用反接二级管来抑制，如图 2-13 和图 2-14 所示。若换成齐纳二级管，则阀值电压应大于 36 V。

继电器和交流输出控制交流电源的保护

当你使用继电器或 AC 输出来开关 115 VAC/230 VAC 负载时，应当在继电器触点或 AC 输出负载上跨接电阻 / 电容网络，如图 2-16 所示。你也可以使用 MOV (金属氧化物可变电阻) 来限制峰值电压，但一定要保证 MOV 的工作电压比正常的线电压至少高出 20%。

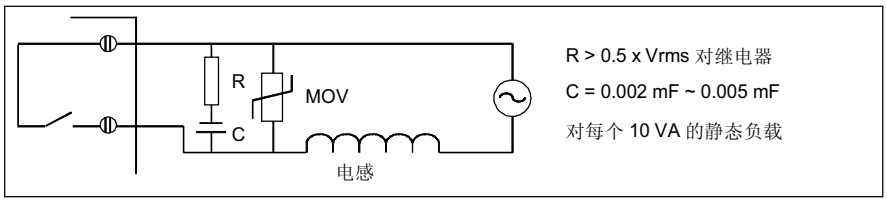


图 2-16 AC 负载继电器或 AC 输出跨接电阻电容网络保护电路

当开关断开时，电容为漏电流提供了通道，确保漏电流 I= 2x 3.14x f x C x Vrms 同应用相符。

例如：一个 NEMA 2 型交流接触器具有 183 VA 线圈冲击功率和 17VA 线圈闭合负载功率，在 115 VAC 电源下，冲击电流 I=183VA/115V=1.59 A，这在交流接触器的触点 2A 电流开关能力之内。

电阻 R= 0.5 x 115 = 57.5 Ω，选标称值为 68Ω 的电阻。

电容 C = (17 VA/10) x 0.005 = 0.0085 μF，选标称值为 0.01 μF 标准电容。

漏电流 I= 2 x 3.14 x 60 x 0.01 x 10^{-6} x 115 = 0.43 mA rms。

2.5 电源的设计

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 24V DC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 5 VDC 和 24 VDC 电源：

- 每一个 CPU 模块都有一个 24 VDC 传感器电源，它为本机输入点和扩展模块继电器线圈提供 24 VDC。如果电源要求超出了CPU 模块 24 VDC 电源的定额，你可以增加一个外部 24 VDC 电源来供给扩展模块的 24 VDC。
- 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 24 VDC 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 2-1 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 24 VDC 电源。本例中 I/O 要求 400 mA 的 24 VDC 电源，但是 CPU 只能提供 280 mA。本配置需要提供额外的 120 mA 电流。

表2-1 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	660 mA	280 mA
	减	
系统要求	**5 VDC**	**24 VDC**
CPU 224, 14 输入		14 * 4 mA = 56 mA
3 EM 223, 5 V 电源需求	3 * 80 mA = 240 mA	
1 EM 221, 5V 电源需求	1 * 30 mA = 30 mA	
3 EM 223, 每个 8 输入		3 * 8 * 4 mA = 96 mA
3 EM 223, 每个 8 继电器线圈		3 * 8 * 9 mA = 216 mA
1 EM 221, 每个 8 输入		8 * 4 mA = 32 mA
总需求	**270 mA**	**400 mA**
	等于	
电流平衡	**5 VDC**	**24 VDC**
总电流平衡	**剩 390 mA**	**缺 120 mA**

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		

S7-200 编程系统使用入门

3

本章描述如何建立一个 S7-200 编程系统。本章中所描述的编程系统包括：

- 一台 S7–200 CPU
- 一台装有 STEP 7–Micro/WIN 32 的 PC 机或编程器
- 一根连接电缆

本章概述

节	内　　容	页
3.1	概述	3–2
3.2	STEP 7-Micro/WIN 32 快速入门	3–3
3.3	如何用 PC/PPI 电缆设置通讯?	3–4
3.4	如何建立与 S7–200 CPU 的在线联系?	3–6
3.5	如何修改 PLC 的通讯参数?	3–7

3.1 概述

通用信息

按照如下的原则安装系统：

- 所使用的操作系统 (Windows 95, Windows 98, 或 Windows NT 4.0)
- 所使用的硬件类型，例如：
  - 带PC/PPI电缆的PC机
  - 带通讯处理器 (CP) 卡的 PC 机或 SIMATIC 编程器
  - CPU 221, CPU 222, CPU 224, CPU 226
  - 调制解调器 (Modem)
- 所使用的波特率

推荐设备

STEP 7-Micro/WIN 32、3.1版和STEP 7-Micro/WIN 32 软件工具包是基于Windows 的应用软件，它支持 32 位 Windows 95，Windows 98，和 WindowsNT 使用环境。 使用 STEP 7-Micro/WIN 32 时，应具有以下设备：

- TP 070 触摸屏用于使用STEP 7-Micro/WIN 32 软件工具包。
- 一台PC机，CPU为80586或更高的处理器，16M内存；或者是装有STEP 7-Micro / WIN 32 的西门子编程器 (例如PG740)。最低要求为：CPU 80486，8M 内存。
- 以下一种设备：
  - 一根连接到通讯口的 PC/PPI 电缆
  - 一个通讯处理器 (CP) 卡
- VGA 显示器，或 Microsoft Windows 所支持的其他显示器(分辨率1024×768)
- 至少 50 M 硬盘空间
- Windows 95, Windows 98, 或 Windows NT 4.0
- 推荐的选件：Microsoft Windows 所支持的鼠标

STEP 7-Micro/WIN 32 提供在线帮助和一本在线使用入门手册，从**帮助**菜单中或按 F1 键可以得到所需的帮助信息。

STEP 7-Micro/WIN 32 软件工具包可为您提供 TP 070 组态软件，以及MicroMaster变频器所用的USS协议指令。

3.2 STEP 7-Micro/WIN 32 快速入门

预装指令

在运行装载前，您应该：

- 如果已装有旧版本的 STEP 7-Micro/WIN 32，应将所有的 STEP 7-Micro/WIN 项目文件备份到软盘上。
- 关闭所有的应用软件，包括 Microsoft Office 工具条。
- 确信 PC 机和 CPU 间的通讯电缆已连接好，见 3.3 节。

安装 STEP 7-Micro/WIN 32

按以下步骤安装 STEP 7-Micro/WIN 32 软件：

1. 将 CD 盘或软盘插入相应的驱动器
2. 单击 “Start” 按钮启动 Windows 菜单。
3. 单击 Run 菜单。
4. 如果从软盘装载，则在 Run 对话框中，键入 a:\setup 并按 OK 或 ENTER键，则开始进行安装过程；
   如果从光盘装载，则在Run对话框中，键入 e:\setup (“e盘”为光驱)并按 OK 或 ENTER键 ，则开始进行安装过程。
5. 按照在线安装程序完成软件的安装。
6. 安装会自动出现设置 PG/PC 接口对话框。在本章后面将介绍如何设置 PG/PC 参数。点击 “Cancel”进行下一步。
7. 在安装结束时会出现下列两种选项：
   - 选项1：是，我要现在重新起动计算机 (缺省选项)
     否，我以后再重新起动计算机

     如果出现该选项，建议您选择缺省选项，点击 Finish 完成安装。随后请阅读ReadMe文件浏览有关STEP 7-Micro/WIN 32的最新信息。
   - 选项2：是，我现在要浏览ReadMe文件 (缺省选项)
     否，我现在要进入STEP 7-Micro/WIN 32

     如果出现该选项，建议您选择缺省选项，点击 Finish 完成安装。随后请阅读ReadMe文件浏览有关STEP 7-Micro/WIN 32的最新信息。

有关STEP 7-Micro/WIN 32 的最新信息可以在 CD 或磁盘中的 README*.X*.TXT 文件中得到。 (在 *x* 位置，字母 A = 德语，B = 英语，C = 法语，D = 西班牙语，E = 意大利语) 。

单连接用户的常见问题列表

下面的一些情况会造成通讯故障：

- 不正确的波特率：修改波特率
- 不正确的站地址：修改站地址
- PC/PPI 电缆设置不正确：检查 PC/PPI 电缆上的 DIP 开关
- 计算机上的不正确的通讯口：检查通讯口
- 自由模式下的 CPU (通讯口由用户程序控制) ：把 CPU 置为 STOP 模式
- 和其它主站冲突：把 CPU 从网络中断开

3.3 如何用 PC/PPI 电缆设置通讯?

本节说明如何利用 PC/PPI 电缆建立 S7-200 CPU 与个人计算机之间的通讯。这是单主机组态，不需要安装其它的硬件 (例如调制解调器或编程设备)。

如何把计算机连到 S7-200 CPU ?

图 3–1 给出了一个利用 PC/PPI 电缆连接计算机和 CPU 的典型组态。请按照下面的步骤在几个部件之间建立通讯：

1. 设置 PC/PPI 电缆上的 DIP 开关，选择计算机所支持的波特率。如果 PC/PPI 电缆支持这些选项，也要选择 11 位 和 DCE。
2. 把 PC/PPI 电缆的 RS–232 端 (标着 PC) 连接到计算机的通讯口，是 COM1 或者是 COM2，并拧紧连接螺丝。
3. 把 PC/PPI 电缆的 RS–485 端 (标着 PPI) 连接到 CPU 的通讯口，并拧紧连接螺丝。

有关 PC/PPI 电缆的技术细节，请参阅附录 A；有关它的定货号，请参阅附录 E；有关网络应用，请参阅第 7 章。

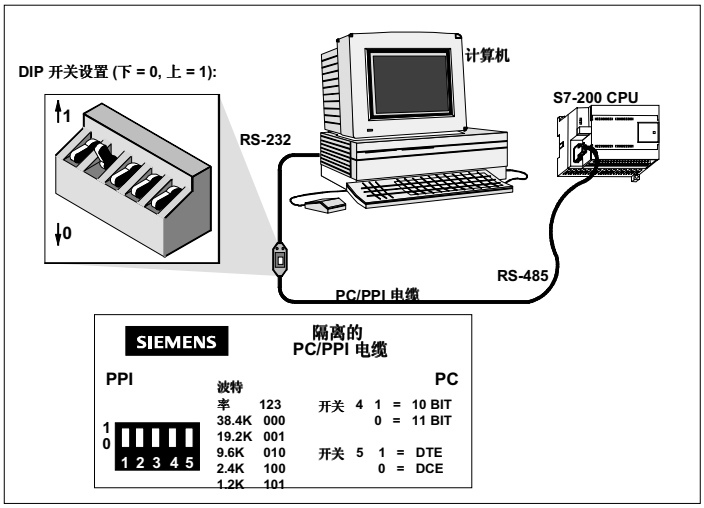


图 3-1 PPI 模式下与 CPU 的通讯

如何为我的接口核实缺省的参数?

按照下面的步骤可以为你的接口核实缺省的参数:

1. 在 STEP 7-Micro/WIN 32 下，单击通讯图标，或从菜单中选择 **View> Communications**，于是出现一个通讯设定对话框。
2. 在通讯设定对话框中，双击 PC/PPI 电缆的图标。将出现设定 PG/PC 接口的对话框。见图 3–2。
3. 选择“Properties” 钮，将出现接口属性的对话框 (见图 3–3)。检查有关属性， 确保正确。通讯速率缺省值为 9600 波特。
4. 有关修改缺省参数的帮助，参见第7章的 7.3 节。

注意:

如果在 PG/PC 接口对话框中没有列出你正在使用的硬件，你必须安装正确的硬件。第 7.2 节说明了安装和拆卸接口的方法。

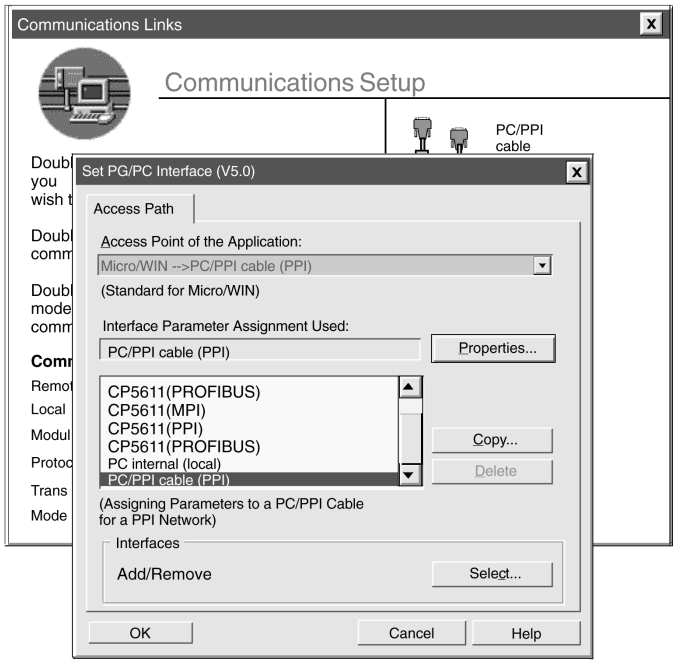


图 3-2 设定 PG/PC 接口的对话框

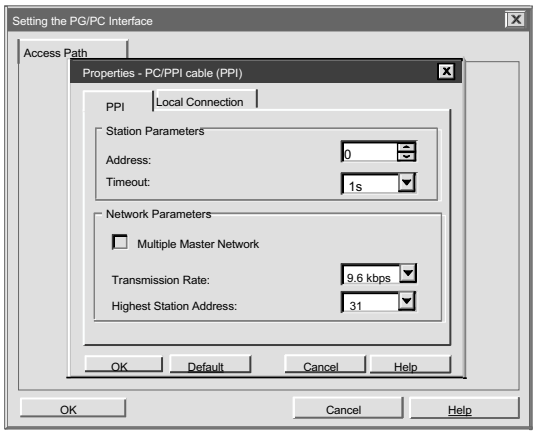


图 3-3　PG/PC 接口对话框属性

3.4　如何建立与 S7-200 CPU 的在线联系?

当你的计算机已经安装 STEP 7-Micro/WIN 32 软件，并且用 PC/PPI 电缆建立通讯， 就可以与 S7-200 CPU 建立在线联系。 (如果你使用的是编程器，编程器上已经预装了 STEP 7-Micro/WIN 32。)

按照下面的步骤建立 S7-200 CPU 的联机：

1. 在 STEP 7-Micro/WIN32 下， 单击通讯图标， 或从菜单中选择 **View>Communications**。将出现一个通讯建立对话框，显示没有连接 CPU。
2. 双击通讯建立对话框中的刷新图标。STEP 7-Micro/WIN 32检查所连接的任何 S7–200 CPU (站)。在通讯连接对话框中所连接的每个站显示为一个 CPU图标。 见图 3–4。
3. 双击要进行通讯的站，在建立通讯对话框中可以看到所选站的通讯参数。
4. 现在可以建立与 S7–200 CPU 的在线联系。

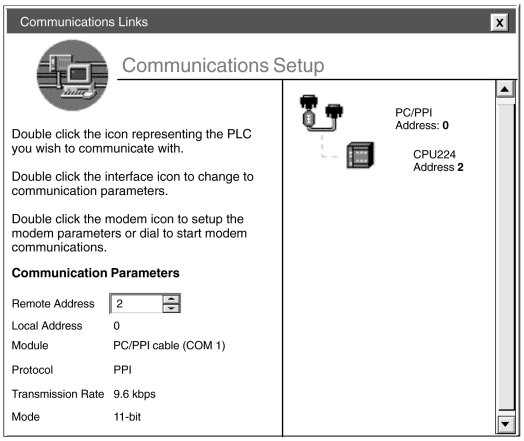


图 3-4　通讯建立对话框

3.5 如何修改 PLC 的通讯参数?

一旦和 S7–200 CPU 建立在线连接，就可以核实或修改 PLC 的通讯参数。

按照下面的步骤修改通讯参数：

1. 单击导引条中的系统块图标，或从主菜单中选择 View > System Block。
2. 出现系统块对话框。单击 Port(s) 项，（见图 3–5)。根据缺省，站地址是 2，波特率是 9.6 k 波特。
3. 选择 “OK” 保持这些参数。如果要修改参数，先进行有关的修改，然后单击 “Apply” 钮，然后再单击“OK”。
4. 单击工具条中的下装图标，把修改后的装入到 PLC。

设置的通讯参数已经接受。

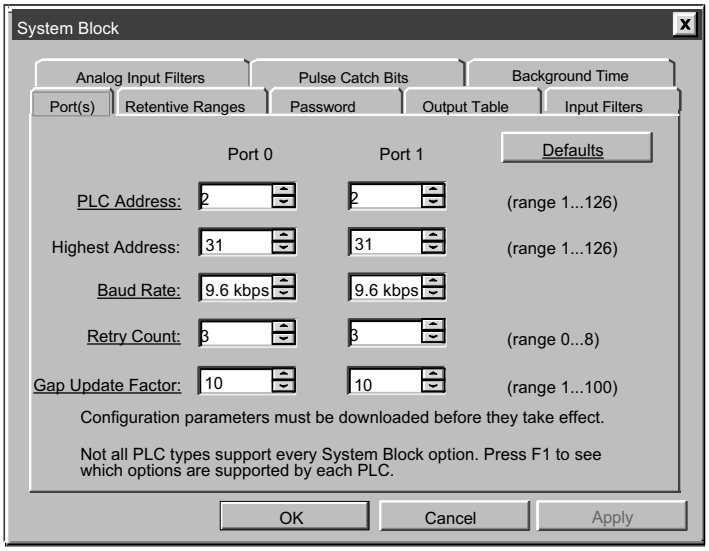


图 3-5 修改通讯参数

S7-200 CPU 编程的基本概念

4

在使用 S7–200 CPU 开始应用编程前，你应该熟悉 CPU 的一些基本操作性能。

本章概述

4.1 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。本节给出一些可以用于设计项目的通用指导原则。当然，你必须遵循所在公司的指导规程和所接受的培训内容。图 4–1 给出了设计过程的一些基本步骤。

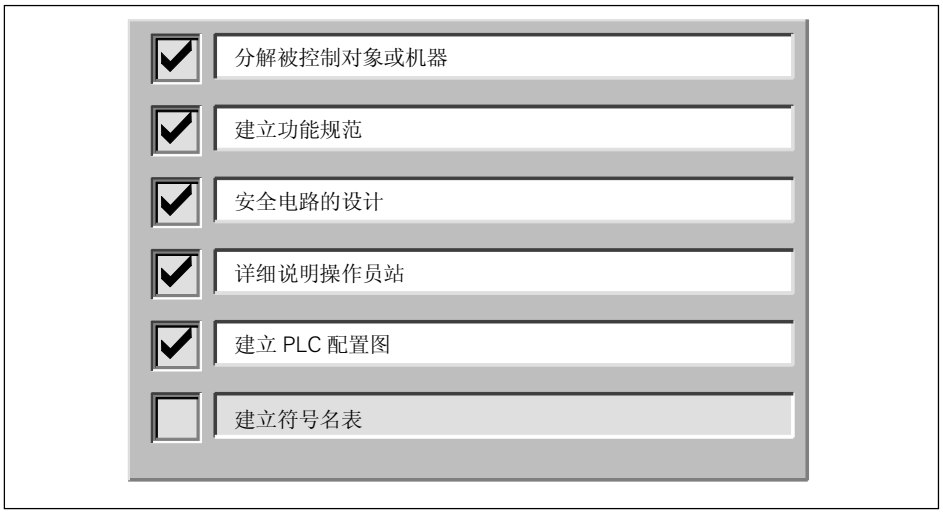


图 4-1 规划一个 PLC 系统的基本步骤

分解被控制对象或机器

把要控制的对象或机器分解成相互独立的部分。这些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。

建立功能规范

写出被控对象或机器的每部分的操作描述。它包括以下内容:

- 输入/输出 (I/O) 点
- 操作的功能描述
- 每个执行器(线圈，电机，驱动器等)的允许状态(执行前要满足的状态)
- 操作接口的描述
- 与被控对象或机器的其他部分的接口

安全电路的设计

识别要求硬接线逻辑安全的设备。控制设备在不安全的状态下出现故障，会造成不希望的启动或机器操作的变化。当不希望的或不正确的机器操作会造成人身伤害或严重的财产损失时，应该考虑采用和 CPU 独立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下面的任务：

- 识别会造成危害的不合适或不希望的执行器操作。
- 识别那些保证不危险操作的条件，并决定如何独立于 CPU 检测这些条件。
- 识别当控制对象得电或断电时 CPU 和 I/O 如何影响控制对象，识别何时错误被检测出来。这个信息只能用于常规的和希望的异常处理操作，不能用来保证安全目的。
- 设计独立于 CPU 的手动或机电冗余来阻止危险的操作。
- 向 CPU 提供独立电路的适当的状态信息，以便于程序和操作员界面得到所需要的信息。
- 识别其它的和控制对象安全操作有关的安全要求。

详细说明操作员站

根据功能描述的要求建立操作员站的配置图。包括如下的项目：

- 与控制对象或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，开关，指示灯等)
- 与 CPU 或扩展模块有关的电气图

建立 PLC 配置图

根据功能描述的要求建立控制设备的配置图。包括如下的项目：

- 和控制对象或机器有关的每个 CPU 的位置图
- CPU 和 扩展 I/O 模块的机械布局图 (包括控制柜和其他设备)
- 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

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。

4.2 S7–200 程序的一些概念

与输入和输出有关的程序

S7–200 CPU 的基本操作非常简单：

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

图 4–2 给出了一个简图说明一个电磁继电器如何连到S7–200CPU。在这个例子中，启动排水的操作面板开关状态加到其它的输入状态上。这些状态的计算决定了输出的状态，由输出驱动关闭排水的线圈。

CPU 连续地扫描程序，读写数据。

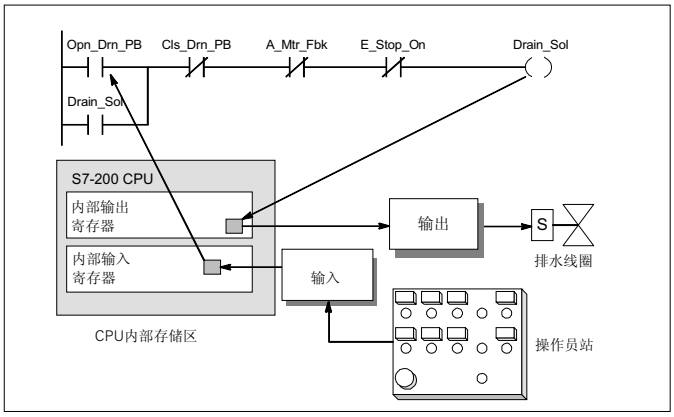


图 4-2 与输入和输出有关的程序

4.3 S7–200 编程语言和编程器的一些概念

为了完成广泛的自动化任务 S7–200 CPU 提供许多类型的指令。在 S7–200 CPU 中有两类基本指令：SIMATIC 和 IEC 1131–3。并且，基于计算机的编程软件 STEP 7-Micro/WIN 32 提供不同的编辑器选择，可以利用这些指令创建控制程序。例如，也许你喜欢在图形化环境下编程，可能在你单位别人喜欢使用汇编语言类型的编辑器。

有两种创建用户程序的基本选择。

- 使用两种指令集 (SIMATIC或 IEC 1131–3)
- 使用的编辑器 (语句表 STL，梯形图 LAD 或功能块图 FBD)

表 4-1 给出了 S7–200 指令集和编辑器的可能组合。

表 4-1 SIMATIC 与 IEC1131–3 指令集和编辑器

SIMATIC 指令集	IEC 1131–3 指令集
语句表 (STL) 编辑器	没有
梯形图 (LAD) 编辑器	梯形图 (LAD) 编辑器
功能块图 (FBD) 编辑器	功能块图 (FBD) 编辑器

语句表编辑器

STEP 7-Micro/WIN 32 语句表 (STL) 编辑器允许输入指令助记符创建控制程序。一般来说，STL 编辑器更适合于熟悉 PLC 和逻辑编程的有经验的程序员。STL 编辑器也能让你编写用梯形图或功能块图编辑器无法实现的程序。这是因为你是用 CPU 直接执行的语言进行编程，而在图形编程器中，为了正确地画出图形必须遵守一些规则。 图 4–3 给出了语句表程序的一个例子。

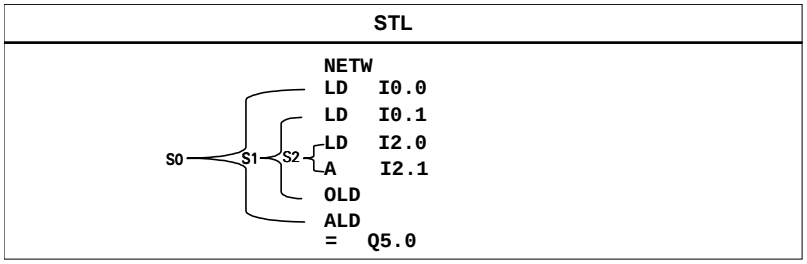


图 4-3 语句表程序实例

从图4-3可以看出，这样的文本型的程序和汇编语言程序非常相似。CPU从上到下按照程序的次序执行每一条指令，程序结束后再返回到起始位置继续执行。STL和汇编语言在另一方面也相似。S7–200 CPU使用一个逻辑堆栈分析控制逻辑 (如图4-4)。LAD和FBD 编辑器自动地插入必要的指令来处理堆栈操作。使用STL，你必须插入这些指令处理堆栈。

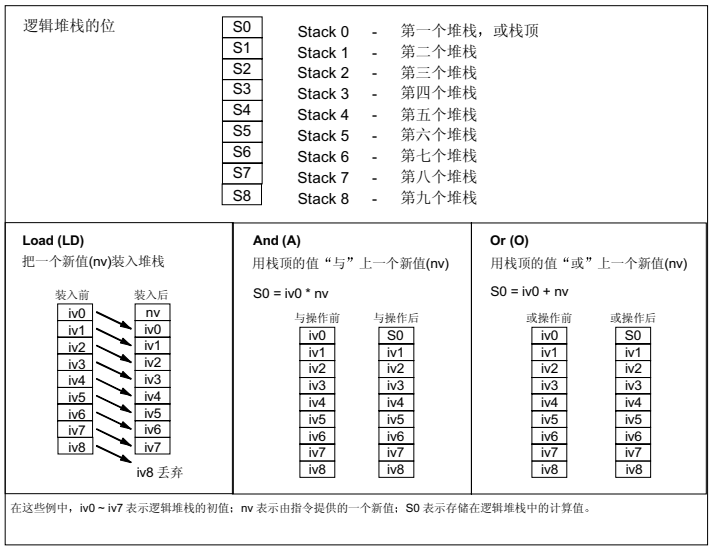


图 4-4 S7-200 CPU 的逻辑堆栈

当你选择 STL 编辑器时主要考虑：

- STL 最适合于有经验的程序员。
- STL 有时让你能够解决利用 LAD 或FBD 编辑器不容易解决的问题。
- 当使用 SIMATIC 指令集时只能使用 STL 编辑器。
- 虽然可以利用 STL 编辑器查看或编辑用 SIMATICLAD 或 FBD 编辑器编写的程序，但是反过来不一定成立。SIMATIC LAD 或 FBD 编辑器不能显示所有利用 STL 编辑器编写的程序。

梯形逻辑编辑器

利用 STEP 7-Micro/WIN 32 梯形逻辑 (LAD) 编辑器可以建立与电气接线图等价的类似程序。梯形图编程可能是许多 PLC 编程人员和维护人员选择的方法。 一般而言，梯形图程序让 CPU 仿真来自电源的电流通过一系列的输入逻辑条件，根据结果决定逻辑输出的允许条件。逻辑通常被分解成小的容易理解的片，这些片经常被称为“梯级”或“段”。 程序一次执行一个段，从左到右，从上到下。一旦 CPU 执行到程序的结尾，又从上到下重新执行程序。

图 4–5 给出了一个梯形逻辑图的实例。

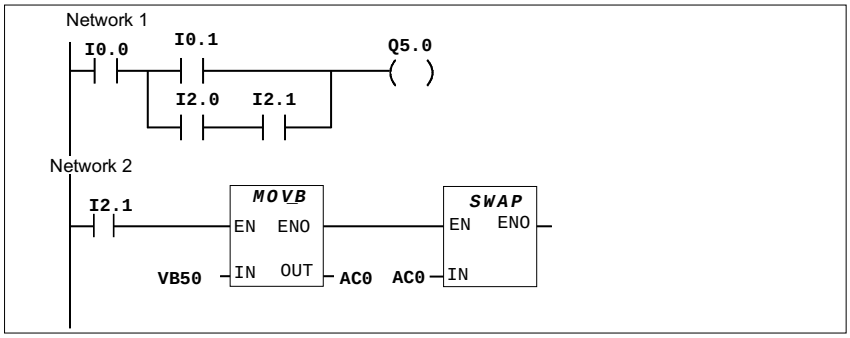


图 4-5 LAD 程序实例

用图形符号表示的指令包括三个基本形式。如图4-5所示，可以把几个指令盒串联起来。

- 触点 – 代表逻辑“输入”条件，例如，开关，按钮，内部条件等。
- 线圈 – 通常代表逻辑“输出”结果，例如，灯，电机启动器，中间继电器，内部输出条件等。
- 盒 – 代表附加指令，例如，定时器，计数器或数学运算指令。

当你选择 LAD 编辑器时主要考虑：

- 梯形逻辑易于初学者使用。
- 图形表示易于理解，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。

功能块图编辑器

利用 STEP 7-Micro/WIN 32 功能块图 (FBD) 编辑器可以查看到象普通逻辑门图形的逻辑盒指令。它没有梯形图编辑器中的触点和线圈，但是有与之等价的指令，这些指令是作为盒指令出现的。程序逻辑由这些盒指令之间的连接决定。也就是说，一个指令 (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。这样的连接思想可以使你解决范围广泛的逻辑问题。

图 4–6 是一个利用 FBD 编辑器建立的程序实例。

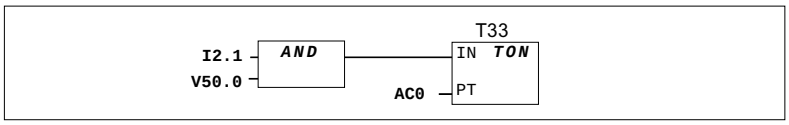


图 4-6 FBD 程序实例

当你选择 FBD 编辑器时主要考虑：

- 图形逻辑门表示格式有利于程序流的跟踪。
- FBD 编辑器可以使用 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。

4.4 理解 SIMATIC 和 IEC 1131-3 指令之间的区别

SIMATIC 指令集

大部分 PLC 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。SIMATIC 指令集是为 S7-200 PLC 设计的。和其它系列的 PLC 比较时，这些指令可能在表示和操作上有小的差别。当你选择SIMATIC 指令集时主要考虑：

- SIMATIC 指令通常执行时间短。
- 三个编程器 (LAD, STL, FBD) 都可以使用 SIMATIC 指令集。

IEC 1131-3 指令集

国际电工委员会 (IEC) 是一个世界性的组织，它制定电气技术领域的总的标准。在过去的几年中，委员会推出了一个有关 PLC 编程方面的轮廓性标准。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。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，程序就不再严格和 IEC 1131–3 兼容) .
- 一些指令可以接受多数据格式，这个概念通常指多重功能。例如，数学指令盒中不是具有独立的 ADD-I (整数加法) 和 ADD_R (实数加法) ，IEC1131-3 的加法指令检查被加数的格式，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，如果你给一个需要位值 (开/关) 的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。

选择 IEC 1131-3 指令时应该考虑如下的主要因素是：

- 易于学习如何创建不同品牌 PLC 的程序
- 除经常使用的许多 SIMATIC 指令，还有一些其它指令 (表示为标准指令)。
- 一些指令和 SIMATIC 指令不一样 (例如，定时器、计数器、乘法和除法等)。
- 这些指令可能执行时间长。
- 这些指令只能在 LAD 和 FBD 编辑器中使用。
- IEC 1131–3 规定变量必须用类型进行声明，而且支持系统数据类型检查。

SIMATIC 和 IEC 1131-3 变量数据类型

每个 SIMATIC 和 IEC1131-3 指令或带参数的子程序通过标识符进行精确定义。对所有的标准指令，每条指令允许的数据类型，可以通过标识符得到。对于带参数的子程序，子程序的这些标识符是由用户通过局部变量表建立。

STEP 7-Micro/WIN 32 对 SIMATIC 模式进行简单的数据类型检查，对 IEC 1131-3 模式进行完全的数据类型检查。当一个数据类型被指明为局部或全局变量时，STEP 7-Micro/WIN 32 保证操作数的数据类型和指令标识符相匹配。表4–2 定义定义基本的数据类型，表 4–3 给出了 STEP 7-Micro/WIN 32 中的复杂数据类型。

表 4-2 IEC 1131–3 基本数据类型

基本数据类型	内 容	数据范围
BOOL (1 位)	布尔型	0 to 1
BYTE (8 位)	无符号型	0 to 255
WORD (16 位)	无符号整数	0 to 65,535
INT (16 位)	有符号整数	-32768 to +32767
DWORD (32 位)	无符号双整数	0 to 2^{32} - 1
DINT (32 位)	有符号双整数	-2^{31} to $+2^{31}$ - 1
REAL (32 位)	IEEE 32 浮点数	-10^{38} to $+10^{38}$

表4-3 IEC 1131–3 复杂数据类型

复杂数据类型	内 容	地址范围
TON	接通延时定时器	1 ms T32, T96 10 ms T33 ~ T36, T97 ~ T100 100 ms T37 ~ T63, T101 ~ T255
TOF	关断延时定时器	1 ms T32, T96 10 ms T33 ~ T36, T97 ~ T100 100 ms T37 ~ T63, T101 ~ T255
TP	脉冲	1 ms T32, T96 10 ms T33 ~ T36, T97 ~ T100 100 ms T37 ~ T63, T101 ~ T255
CTU	加计数器	0 to 255
CTD	减计数器	0 to 255
CTUD	加/减计数器	0 to 255
SR	置位优先位触发器	--
RS	复位优先位触发器	--

数据类型检查 有三级数据类型检查：完全数据类型检查，简单数据类型检查和无数据类型检查。

完全数据类型检查 在该方式下，参数的数据类型必须同符号或变量数据类型匹配。每个有效参数只有一个数据类型 (多重指令除外)。例如，SRW (右移字) 指令的输入 (IN) 参数的数据类型是 WORD。只有给它分配 WORD 型的变量，才能编译成功。当设定为完全数据类型检查时，给 WORD 型指令分配整型 (INT) 变量是无效的。

完全数据类型检查只能在 IEC 1131-3 方式下执行。见表 4–4。

表 4-4 完全数据类型检查：用户选定和等价的数据类型

用户选定的数据类型	等价的数据类型
BOOL	BOOL
BYTE	BYTE
WORD	WORD
INT	INT
DWORD	DWORD
DINT	DINT
REAL	REAL

简单数据类型检查 在简单数据类型方式下，当给一个符号或变量一个数据类型时，也自动分配了和所选定数据类型相匹配的所有数据类型。例如，选择 DINT作为数据类型，局部变量也自动分配 DWORD 数据类型，因为两者都是 32 位的数据类型。虽然REAL也是 32 位数据类型，但是它不是自动分配的。由于 REAL 数据类型没有等价的数据类型，它总是单独定义的。简单数据类型检查只在 SIAMTIC 方式下使用局部变量时执行，见表 4–5。

表 4-5 简单数据类型检查：用户选定和等价的数据类型

用户选定的数据类型	等价的数据类型
BOOL	BOOL
BYTE	BYTE
WORD	WORD, INT
INT	WORD, INT
DWORD	DWORD, DINT
DINT	DWORD, DINT
REAL	REAL

无数据类型检查 无数据类型检查方式只在 SIMATIC 全局变量没有可选的数据类型时使用。在该方式下，所有相同大小的的数据类型自动分配给符号。例如，一个符号分配在地址 VD100 处，表 4-6 表示 STEP 7-Micro/WIN 32 自动为该符号分配了数据类型。

表 4-6 大小决定了 SIMATIC 全局符号的数据类型

用户选定的地址	分配的等价数据类型
V0.0	BOOL
VB0	BYTE
VW0	WORD, INT
VD0	DWORD, DINT, REAL

数据类型检查的优点

数据类型检查有助于避免常见的编程错误。如果一条指令支持有符号数，STEP 7-Micro/WIN 32 将对操作数标示出无符号数的使用。例如，关系比较 < I 是一个有符号指令。对于有符号操作数，-1 小于 0。但是，当 < I 指令允许支持无符号数据类型时，编程器必须保证不出现下面的情况。在程序执行期间，对于 < I 指令，无符号数 40,000 小于 0。

警告：

应该保证对有符号指令使用无符号数不能超过正负数的范围。

如果不能保证对有符号指令的无符号数不超过正负数的范围，在你的程序或控制器操作中回出现不可是预测的结果。

应该确保有符号指令的无符号操作数不超过正负数的范围。

总之，在 IEC 1131-3 编辑方式下，由于指令的数据类型的不匹配，会造成编译错误，完全数据类型检查有助于确定这些错误。SIMATIC 编辑器不具有该功能。

选择 SIMATIC 和 IEC 1131-3 编程方式

由于 IEC 1131-3 是完全数据类型化的，而 SIMATIC 是不完全数据类型化的，STEP 7-Micro/WIN 32 不允许在两个编程方式下传递程序。你必须选定一个自己喜欢的变成方式。

多重指令

多重指令支持一定范围的数据类型。完全数据类型检查仍然有效，由于在指令成功编译前所有的数据类型必须匹配。表 4–7 是 IEC 多重加法指令的一个例子。

表 4-7 IEC 多重加法指令实例

指令	允许数据类型 (完全数据类型检查)	允许数据类型 (数据类型检查)	编译后的指令
ADD	INT	WORD, INT	ADD_I (整数加法)
ADD	DINT	DWORD, DINT	ADD_D (双整数加法)
ADD	REAL	REAL	ADD_R (实数加法)

当所有的操作数都具有 DINT 数据类型时，编译器将产生一个双整数加法。如果多重指令的操作数的数据类型不一致，就会出现编译错误。根据数据类型检查的级别决定什么是非法操作数。下面的例子在完全数据类型检查下产生编译错误，但是在简单数据类型检查可以通过。

ADD IN1 = INT, IN2 = WORD, IN3 = INT

完全数据类型检查： 编译错误

数据类型检查：编译为 ADD_I (整数加法)

和关系比较例子一样，简单数据类型检查不能避免运行编程错误的发生。 使用简单数据类型检查，编译器不能发现下面的编程错误：ADD 40000, 1 将是一个负数，而不是无符号数 40,001。

在 IEC 中对多重指令使用直接寻址

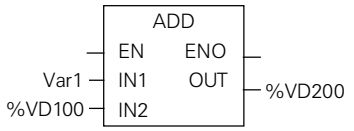
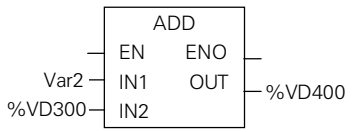
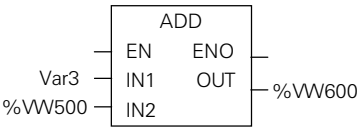
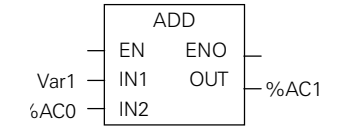
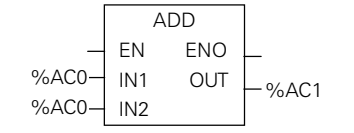
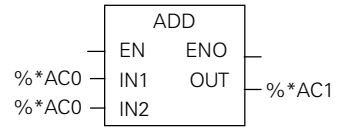
IEC 1131-3 编程方式允许指令使用直接访问内存作为指令参数。在参数中可以使用变量和内存。请记住直接表示的内存区不包含明显的类型信息。另外，任何多重 IEC 指令无法决定类型信息，因为这些指令接受变化的数据类型。

直接表示的参数数据类型通过检查指令中已经分配类型的其它参数决定，指令参数类型被配置成使用特定类型的变量，所有直接表示的参数就被分配成那个类型。表 4-8 和表 4–9 是直接表示的参数数据类型举例，%表示直接地址。

表 4-8　　直接寻址的数据类型举例

名 字	地 址	数据类型	注　释
Var1		REAL	这是浮点变量
Var2		DINT	这是双整数变量
Var3		INT	这是整数变量

表 4-9 直接寻址的数据类型举例

举　例	描　述
	由于 Var1 是实数型，VD100 和 VD200 将被分配成实数型。
	由于 Var2 是双整数型，VD300 和 VD400 将被分配成双整数型。
	由于 Var3 是整数型，VW500 和 VW600 将被分配成整数型。
	由于 Var1 是实数型，AC0 和 AC1 将被分配成实数型。
	这个配置是非法的，因为无法决定变量类型。累加器中的数据类型可以是任何类型。
	这个配置是非法的，因为无法决定变量类型。累加器中的数据类型可以是任何类型。
% 表示间接地址。	

使用转换指令

转换指令可以把一个字节型数据变成其它类型。STEP 7-Micro/WIN 32 支持表 4-10 中的转换指令，这些指令可以在简单的数据类型之间进行转换。

表 4-10 转换指令

转换指令	操作数的完全数据类型检查	操作数的数据类型检查
BYTE 到 INT	IN: OUT:	IN: BYTE OUT: WORD，INT
INT 到 BYTE	IN: OUT:	IN: WORD，INT OUT: BYTE
INT 到 DINT	IN: OUT	IN: WORD，INT OUT: DWORD，DINT
DINT 到 INT	IN: OUT:	IN: DWORD，DINT OUT: WORD，INT
DINT 到 REAL	IN: OUT:	IN: DWORD，DINT OUT: REAL
REAL 到 DINT (ROUND)	IN: OUT:	IN: REAL OUT: DWORD，DINT

在 IEC 1131-3 编程模式中，可以使用多重 Move 指令把一个数在 INT 和 WORD, DINT 和 DWORD 之间进行转换。当用 Move 指令转换相同大小的数据类型时,编译不会产生错误，见表 4-11。

表 4-11 多重 MOVE 指令的使用

IEC 1131-3 多重 Move 指令	IN	OUT
MOVE (INT 到 WORD)	INT	WORD
MOVE (WORD 到 INT)	WORD	INT
MOVE (DINT 到 DWORD)	DINT	DWORD
MOVE (DWORD 到 DINT)	DWORD	DINT

4.5 建立程序的基本元素

S7-200 CPU 连续地执行用户程序，控制一个任务或过程。利用 STEP 7-Micro/ WIN 32 可以建立用户程序并把它下装到 CPU。在主程序中，可以调用不同的子程序或中断程序。

程序的组织

S7-200 CPU 程序由三个基本元素构成：主程序、子程序 (可选) 和中断程序 (可选)。S7-200 程序由下面的元素来组织：

- 主程序：在程序的主体中放置控制应用指令，主程序中的指令按顺序在 CPU的每个扫描周期执行一次。
- 子程序：它们是程序的可选部分，只有当主程序调用它们时，才能够执行。
- 中断程序：它们是程序的可选部分，只有当中断事件发生时，才能够执行。

子程序和中断程序举例

下面是定时中断的举例，中断程序可以读模拟量输入。在本例中，模拟量的采样时间设定为 100 ms。

图 4-7 到图 4-11 是不同 S7-200 编程语言表示的子程序和中断程序。

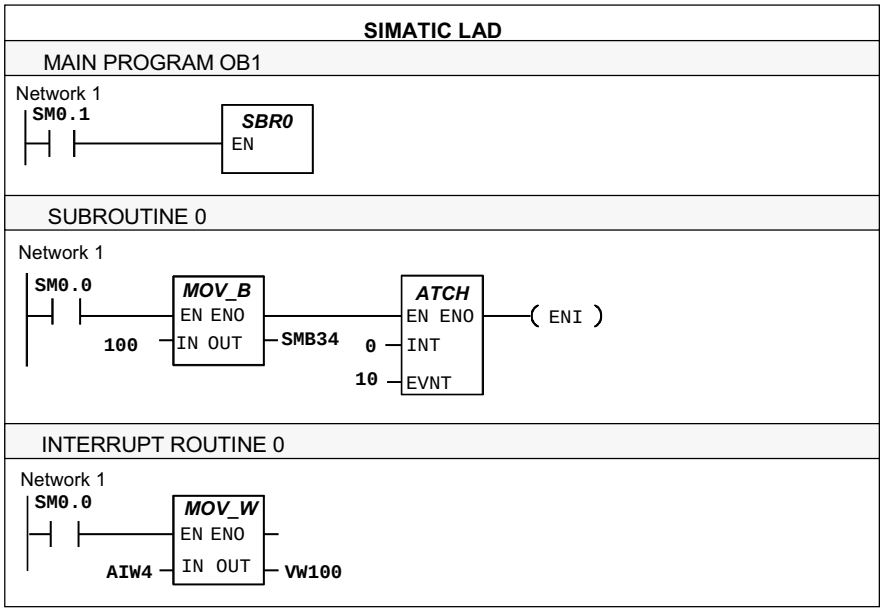


图 4-7 采用子程序和中断程序的 SIMATIC 梯形图程序

Statement List

Main Program OB1

```
      Network 1
      LD    SM0.1          //When first scan cycle
                           //bit comes
      CALL    0            //Call subroutine 0.
```

Subroutine 0

```
      Network 1
      LD   SM0.0           //Always on memory bit.
      MOVB   100, SMB34    //Set timed interrupt 0
                           //interval to 100 ms.
      AENO                 //If move is successful,
      ATCH   0, 10         //attach timed interrupt 0
                           //to interrupt routine 0.
      AENO                 //If attach is successful,
      ENI                  //then enable Global Interrupt.
```

Interrupt Routine 0

```
                           //Begin interrupt routine 0.
      Network 1
      LD SM0.0             //Always on memory bit.
      MOVW   AIW4,VW100    //Sample Analog Input 4.
```

图 4-8 采用子程序和中断程序的 SIMATIC STL 程序

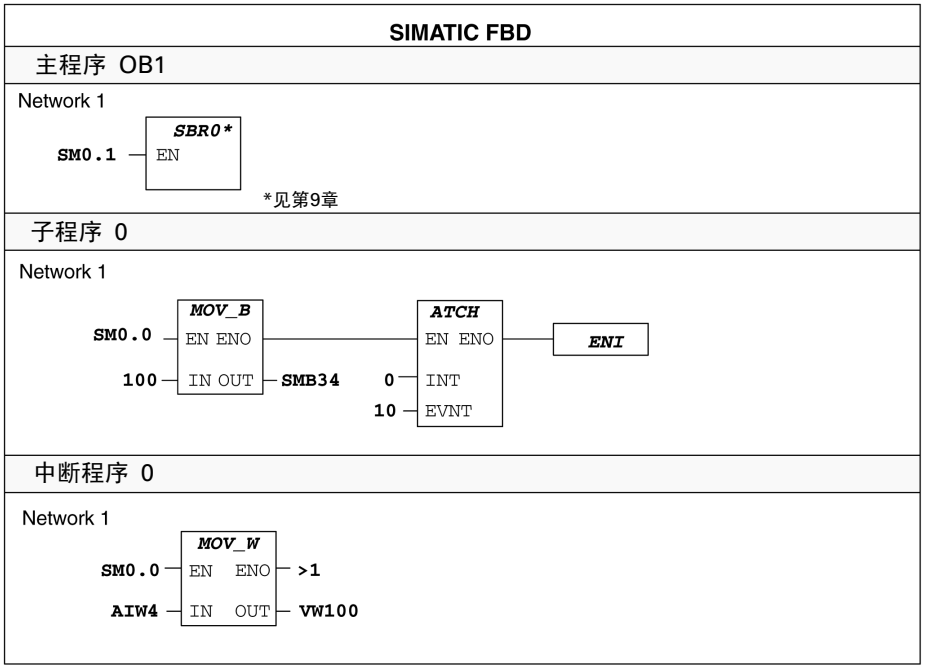


图 4-9 采用子程序和中断程序的 SIMATIC FBD 程序

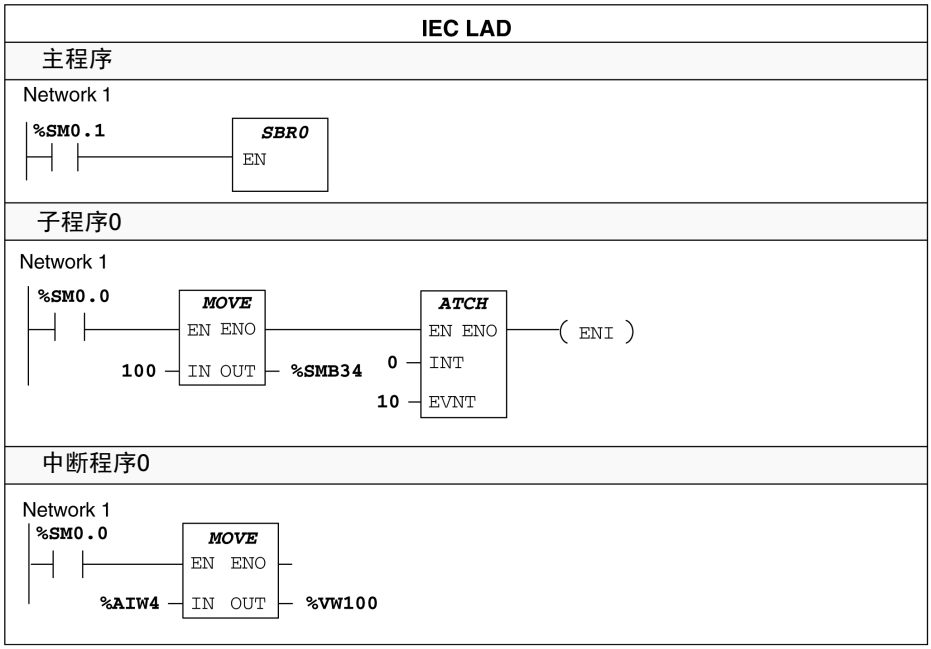


图 4-10 采用子程序和中断程序的 IEC 梯形图程序

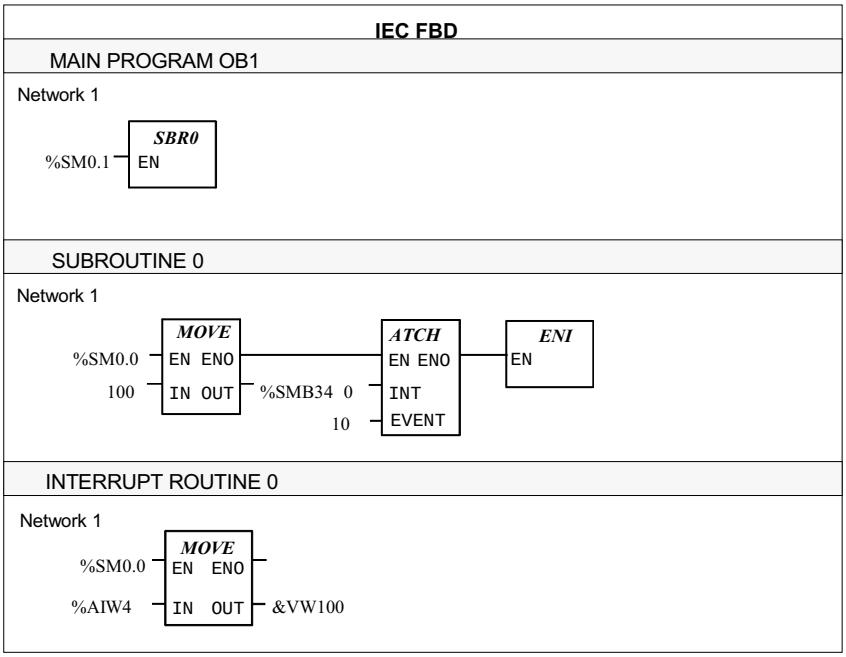


图 4-11　采用子程序和中断程序的 IEC FBD 程序

4.6　理解 CPU 扫描周期

S7–200 CPU 连续执行用户程序，任务的循环序列称为扫描。如图 4-12 所示，CPU 的扫描周期包含以下任务：

- 读输入
- 执行程序
- 处理通讯请求
- 执行 CPU 自诊断测试
- 写输出

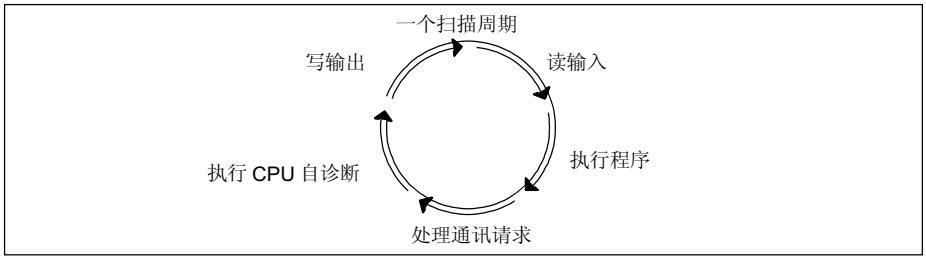


图 4-12　S7-200 CPU 的扫描周期

扫描周期中执行的任务依赖于 CPU 的操作模式。S7-200 CPU有两个操作模式：STOP 模式和 RUN 模式。对于扫描周期，STOP 模式与 RUN 模式的主要差别是在RUN模式下运行用户程序，而在 STOP 模式下不运行用户程序。

读数字输入

每次扫描周期开始时，先读数字输入点的当前值，然后把这些值写到输入映象寄存器中。

CPU 以 8 位 (1 个字节)为增量的方法来保留输入映象寄存器。如果 CPU 或扩展模块不给物理输入点提供保留字节的每一位，那么你就不能把这些位重新分配给 I/O 链中的后续模块，也不能在程序中使用它们。在每次扫描周期开始时，CPU 会将映象寄存器中未使用的输入位清零。然而，你的 CPU 可以连几个扩展模块，而且你并未使用这个 I/O 功能 (即未安装扩展模块) ，那么你可以用这些未使用的扩展输入位作为附加的内部存储器标志位来使用。

除非允许模拟量滤波，CPU 在扫描周期中是不能自动更新模拟量输入值。用户可以选择对每个模拟通道设置数字滤波。

数字滤波用于低成本的模拟量模块，这些模块不支持内部滤波。数字滤波应用于输入信号缓慢变化的场合。如果是高速信号，应该不选择数字滤波。

如果模拟输入选择输入滤波器，CPU 在每个扫描周期刷新模拟输入、执行滤波功能，并存储滤波值。当访问模拟输入时，使用滤波值。

如果模拟输入不选择输入滤波器，当访问模拟输入时，CPU每次从物理模块读取模拟值。

执行程序

在扫描周期的执行程序阶段里，CPU 执行程序是从第一条指令开始，直到最后一条指令结束。不论在程序或中断程序执行过程中，直接 I/O 指令允许你对输入和输出点直接存取。

如果在程序中使用了中断，与中断事件相关的中断程序就作为程序的一部分存储下来(见 4.5 节)。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(这可能会在扫描周期的任意点上)。

处理通讯请求

在扫描周期的信息处理阶段，CPU 处理从通讯端口接收到的任何信息。

执行 CPU 自诊断测试

在扫描周期的这个阶段里，CPU 检查其硬件，以及用户程序存储器(仅在 RUN 模式下) ，它也检查所有的 I/O 模块的状态。

写数字输出

在每个扫描周期的结尾，CPU 把存在输出映象寄存器中的数据输给数字输出点。

CPU 以 1 个字节 (8 位) 为增量来保留输出映象寄存器。如果 CPU 或扩展模块不给物理输出点提供保留字节的每一位，那么你就不能把这些位分配给 I/O 链中的后续模块。但是你可以象使用内部存储器标志位那样来使用输出映象寄存器的未用位。

当 CPU 操作模式从 RUN 切换到 STOP，数字输出设置为输出表中定义的值，或保持当前状态 (见 6.4 节) ，模拟输出保持最后写的值。缺省设置是关闭数字输出。

扫描周期中断

如果你使用中断，与每个中断事件相关的中断程序作为程序的一部分存储下来。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后才执行 (这可能会在扫描周期的任意点上)。当中断事件发生时，CPU 以异步扫描方式为用户中断服务。当中断发生时根据中断优先级来处理中断。

输入和输出映象寄存器

在程序的执行过程中，对于输入或输出的存取通常是通过映像寄存器，而不是实际的输入/输出 (I/O) 点，这主要有三个原因：

- 在同步扫描周期的开始采样所有输入，而在扫描周期的执行阶段就有了固定的输入值。而当程序执行完后，那就更新输出映像寄存器。这样使系统有稳定效果。
- 用户程序存取映像寄存器要比存取 I/O 点快得多，因此允许快速执行程序。
- I/O 点必须按位来存取，而映象寄存器可按位、字节、字或双字来存取，因此具有灵活性。

立即 I/O

立即 I/O 指令允许对实际输入输出点直接存取。尽管通常用映象寄存器作为 I/O存取的源或目的操作数。当使用立即 I/O 指令来存取输入点的值时，输入映像寄存器的值尚未更新。当使用立即 I/O 指令来存取输出点的同时，相应的输出映象寄存器被更新了。

除非给模拟输入设置数字滤波，CPU 把模拟 I/O 作为立即数据，见 6.5 节。当给模拟输出写一个值时，输出立即更新。

4.7 选择 CPU 的工作方式

S7–200 CPU 有两种工作方式：

- STOP：当 CPU 处于 STOP (停止) 方式时，CPU 不执行程序，你可以向CPU 装载程序或配置 CPU。
- RUN: CPU 运行程序。

在 CPU 前面板上用 LED 显示当前的工作方式。若要往程序存储器中装载程序，则必须把 CPU 置于 STOP 方式。

用如下的方法改变工作方式：

- 使用 PLC 上的方式开关来手动改变工作方式
- 使用 STEP 7-Micro/WIN 32编程软件，并把CPU模式开关放在TERM或RUN位置。
- 在程序中插入一个 STOP 指令

使用方式开关来改变工作方式

你可以使用方式开关 (位于 CPU 模块的出入口下面) 来手工选择CPU工作方式：

- 把模式开关切换到 STOP，可以停止程序的执行。
- 把模式开关切换到RUN，可以启动程序的执行。
- 设置方式开关为TERM (terminal) 方式，允许程序 (STEP 7 Micro/Win) 来控制 CPU 工作方式。

如果方式开关设为 TERM 或 STOP，且电源发生变化，则电源恢复时，CPU 会自动进入 STOP 方式。如果方式开关设为 RUN 方式，则电源恢复时，CPU 会自动进入 RUN 方式。

使用 STEP 7-Micro/WIN 32 来改变工作方式

如图 4–13 所示，你可以使用 STEP 7 Micro/Win 来改变 CPU 的工作方式，为此你必须设方式开关为 TERM 或 RUN。

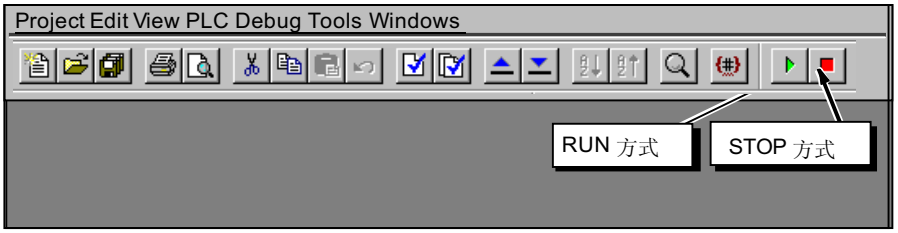


图 4-13 使用 STEP 7 Micro/Win 来改变 CPU 的工作方式

在程序中改变工作方式

你可以在用户程序中加入 Stop 指令，将 CPU 从 RUN 变为 STOP 方式。这就要基于程序逻辑允许中断程序的执行。关于 Stop 的 SIMATIC 指令和 IEC 1131-3 指令详细说明，详参阅第 9 章和第 10 章。

4.8 创建 CPU 密码

所有的 S7–200 CPU 都提供了密码保护，来限制对某些特定的 CPU 功能的使用。对 CPU 功能及存储器的存取授权是通过密码实现的。没有密码，CPU 对任何存取都限制。当你有密码保护时，根据安装密码时的设置，CPU 禁止所有的受限操作。

对 CPU 的存取限制

如表 4-12 所示，S7–200 CPU 提供了限制 CPU 功能存取的三个等级。每个等级允许特定的无需密码的存取功能。对于所有三个等级，输入正确的密码后，可使用 CPU的所有功能。S7 200 CPU 的缺省状态为等级 1 (没有限制)。

在网络上输入密码不会危及 CPU 的密码保护。允许一个用户使用授权的 CPU 功能就会禁止其它用户使用这些功能。在同一时刻，只允许一个用户不受限制地存取。

注：

当输入密码后，在编程设备同 CPU 断开后，该密码等级在一分钟内仍然有效。如果其它用户在此期间内连接 CPU，则它可以进入编程设备。

表 4-12 S7–200 CPU 的存取限制

任 务	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			
上装 CPU 中程序、数据和配置			需要密码
下装到 CPU		需要密码	
STL 状态			
删除用户程序、数据及配置			
强制数据或单次/多次扫描			
拷贝到存储器卡			
在 STOP 模式下写输出			

设置 CPU 密码

使用STEP 7-Micro/WIN 32 给 CPU 创建密码。选择菜单命令 View > System Block，并点击 Password 项，见图 4-14。首先输入合适的 CPU 限制级别，然后输入并确认 CPU 密码。

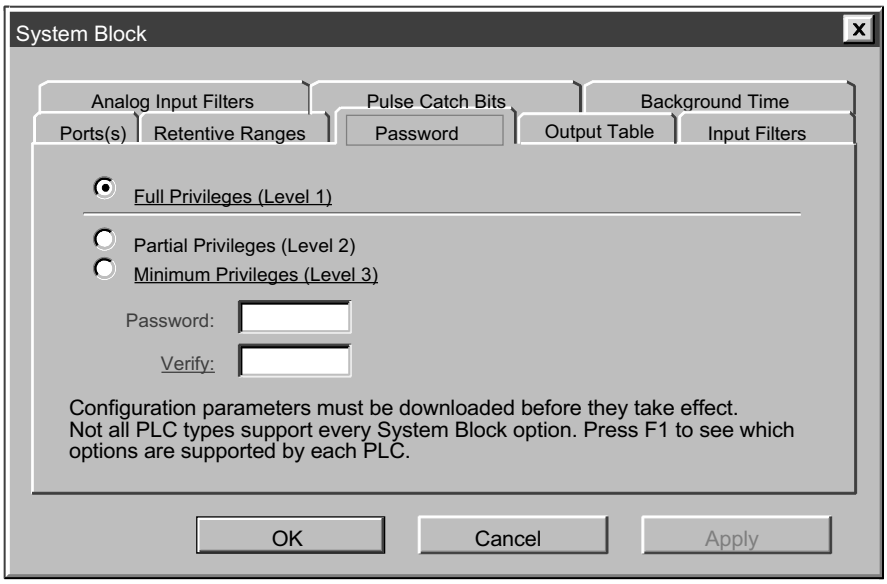


图 4-14 设置 CPU 密码

如果你忘记了密码怎么办?

如果你忘记了密码，那么你必须清除 CPU 存储器，并重新装入你的程序。清除CPU 存储器会使 CPU 处于 STOP方式，并重新设置 CPU 为厂家所设的缺省状态 (CPU 地址、波特率和时钟除外)。

清除 CPU 中的程序，先选择 **PLC>Clear**...菜单命令，显示出清除对话框。再选择“All”选项，点击“OK”按钮。如果配置了密码，就会显示密码授权对话框，输入清除密码 (Clear) ，就可以执行连续清除全部 (Clear All) 操作。

清除全部 (Clear All) 操作并不把程序从存储器卡中去掉。因为密码同程序一起保存在存储器卡中，你必须重新写存储器卡，才能从程序中去掉遗忘的密码。

警告:

清除 CPU 存储器可是导致输出关闭 (Off) (或模拟量输出处于某个固定值)。当清除 CPU 存储器时，如果 S7-200 模块和其它设备相连，那么输出状态的变化会传给该设备。如果你设备的输出“安全状态”同厂家设定不同，输出的变化可能会引起设备产生不可预料的动作，也可能引起人身伤亡，以及/或者损害设备。在清除 CPU 存储器之前，必须有合适的安全预防措施，并确保你的处理是安全的。

4.9 调试及监视程序

STEP 7-Micro/WIN 32 提供了一套工具来调试并监视用户程序。

用单次 / 多次扫描来监视用户程序

可以指定 CPU 以有限扫描次数 (从 1 到 65,535) 执行用户程序。通过选择CPU 扫描次数，当过程变量改变时，你可以监视用户程序的执行。使用菜单命令 **Debug>Multiple Scans** 来指定执行的扫描次数。图 4-15 即为输入 CPU执行扫描次数的对话框。

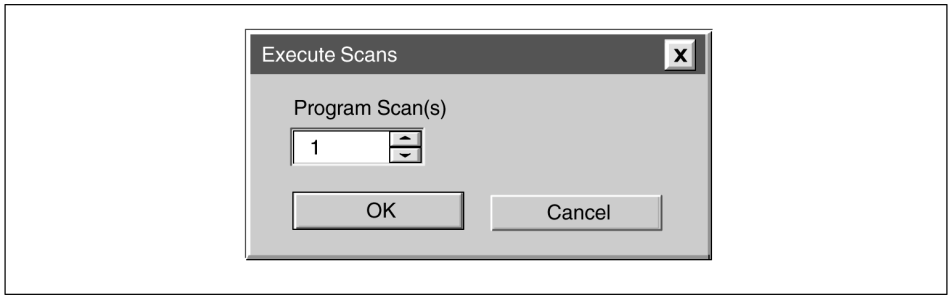


图 4-15 以指定扫描次数执行用户程序

用状态 / 制表来监视并修改用户程序

如图 4-16 所示，利用菜单命令 **View>Status Chart**，当程序运行时，你可以使用状态表来读、写、强制和监视变量。

- 状态表工具图标在STEP 7-Micro/WIN32的工具条区，当选择状态表时，可以激活这些工具图标 (前项排序、后项排序、单次读、全部写、强置、解除强置和读所有强置。
- 可以建立多个状态表。
- 选择单元和单元格式，按鼠标右键弹出下拉表 (图 4-16)。

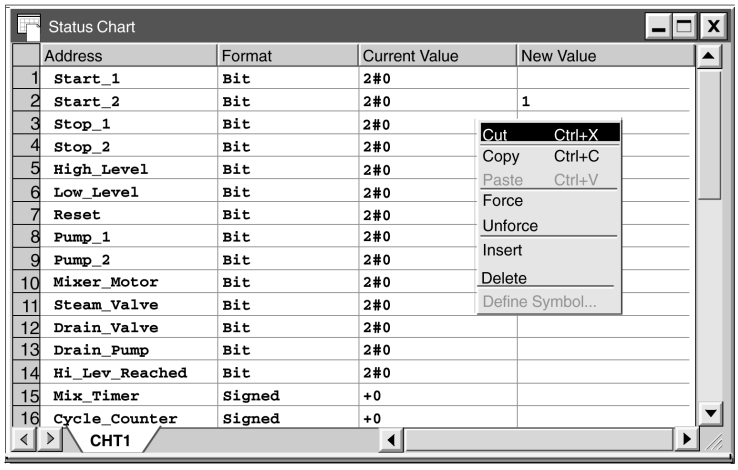


图 4-16 用状态表监视和修改变量

显示梯形图中的程序状态

利用 STEP 7-Micro/WIN 的程序编辑器可以监视在线程序状态 (程序必须为梯形图)。梯形图的状态显示所有操作数的值，如图 4–17 所示。所有这些操作数状态都是PLC扫描周期完成时的结果。STEP 7-Micro/WIN 32 经过多个 PLC 扫描周期采集状态值，然后刷新梯形图的屏幕显示。通常情况下，梯形图的状态显示不反映程序执行时的每个梯形图元素的实际状态。

您可以使用选项工具来组态状态屏幕。选择 **Tools>Options**，然后选择 LAD 状态表。表 4-13 所示为 LAD 状态显示选项。

打开 LAD 状态窗口，从工具条中选择状态图标 (图 4-17)。

表 4-13　　LAD 状态选择显示

显示选项	LAD 状态显示
显示指令内部的地址和指令外部的值	ADD EN ENO +777 VW0 VW4 +800 +23 VW2
显示指令外部的地址和值	ADD EN ENO +777=VW0 IN1 OUT +800=VW4 +23=VW2 IN2
只显示状态值	ADD EN ENO +777 IN1 OUT +800 +23 IN2

打开 LAD 状态窗口，从工具条中选择状态图标 (图 4-17)。

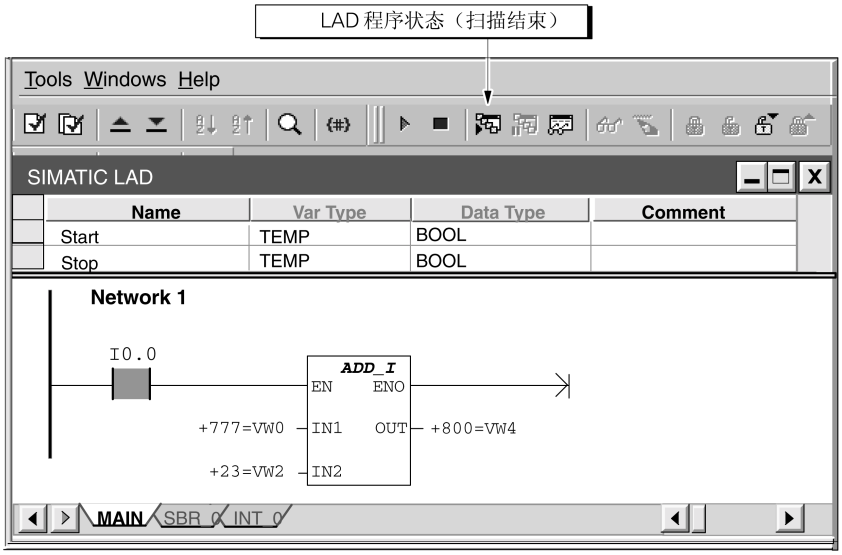


图 4-17　　梯形图中程序状态显示

显示功能块图 (FBD) 中的程序状态

用 STEP 7-Micro/WIN 32 可以监视功能块图程序的状态，STEP 7-Micro/WIN 32 必须选择FBD显示方式。FBD 状态显示所有指令操作数的值，所有这些操作数状态都是 PLC 扫描周期完成时的结果。 STEP 7-Micro/WIN 32 通过多个 PLC 扫描周期才能获得状态显示的值，然后刷新 FBD 状态屏显示。因此，FBD 状态显示并不反映程序执行时每个 FBD 元件的实际状态。

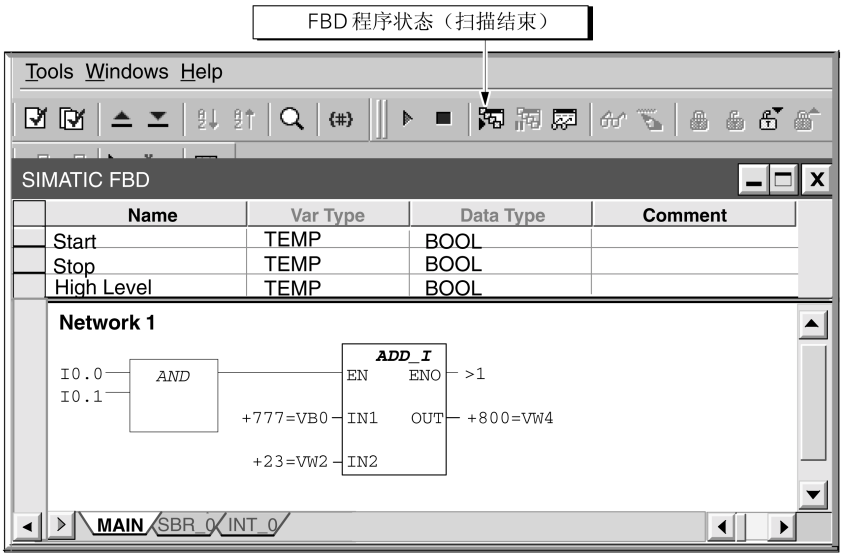
您可以使用选项工具来组态状态屏幕。选择 **Tools>Options**，然后选择 FBD 状态表。表 4-14 所示为 FBD 状态显示选项。

打开 FBD 状态窗口，从工具条中选择状态图标 (图 4-18)。

表 4-14　　FBD 状态选择显示

显示选项	FBD 状态显示
显示指令内部的地址和指令外部的值	ADD EN ENO +777 VWO VW4 +800 +23 VW2
显示指令外部的地址和值	ADD EN ENO +777=VW0 IN1 OUT +800=VW4 +23=VW2 IN2
只显示状态值	ADD EN ENO +777 IN1 OUT +800 +23 IN2

图 4-18　功能块图 (FBD) 程序的状态显示



语句表中程序状态显示

如果您使用 STL 编辑器来观察程序，STEP 7-Micro/WIN 32 提供了一种基于指令 – 指令的监控程序运行状态的方法。这种状态监控方法叫作STL状态图，使用STL状态图进行监控的程序窗口叫作STL状态窗口。该窗口的大小大约是 STEP 7-Micro/WIN32 屏幕的大小，从CPU获取的信息限于200个字节或25行在屏幕上的STL状态行。如果超过了这个限制，将会在状态窗口中显示“-”，状态信息是从位于编辑窗口顶端的第一句STL语句开始显示的。当向下滚动编辑窗口时，将从CPU获取新的信息。典型的STL状态图如图4-19所示。

请选择工具栏上的程序状态按钮来打开STL状态窗口（图4-19），调整STL编程页面的右边框来显示STL状态窗口。

当打开STL状态窗时，STL代码出现在左侧的STL状态窗口里，含有操作数的状态区域显示在右侧。间接寻址的操作数将同时显示指针地址和指针所指的存储单元中的数值。指针地址显示在括号内。

STL程序状态按钮连续不停地更新屏幕上的数值，如果您需要暂停更新屏幕上的数据，请选择已触发的暂停键（图4-19），当前的数据将保留在屏幕上，直到暂停键被重新按下。

操作数将会按顺序显示在屏幕上，其顺序与它们出现在指令中的顺序一样，当指令被执行时，这些数值将被捕捉，因此可以反映出指令的实际运行状态。

状态数值的颜色指示出指令执行的状态：

- 黑色表示指令正确执行，那些没有在SCR块中使用的非条件指令执行时，与逻辑堆栈无关。
- 红色表示指令执行有错误，参考第9章的章节，“设置ENO=0的错误条件”。
- 灰色表示指令没有执行。这是因为堆栈顶的值为“0”，或者是因为该指令在一个SCR块中，并且该块未被使能。
- 空白表示指令没有执行。

导致指令不被执行的条件包括：

- 堆栈顶为0
- 其它诸如跳转等指令将该指令跳过。
- PLC没有处于运行状态

为了使您需要的变量数值出现在STL状态窗口里，请选择 **Tools >Options**，然后选择STL状态标签页，您可以选择监控三类数值。

- 操作数（每条指令最多三个操作数）
- 逻辑堆栈（最多4个来自堆栈的最新数值）
- 指令状态位（最多12个指令状态位）

为了得到第一个扫描周期的信息，将PLC置于停止状态，使能STL状态表，然后选择**DEBUG>First Scan**。

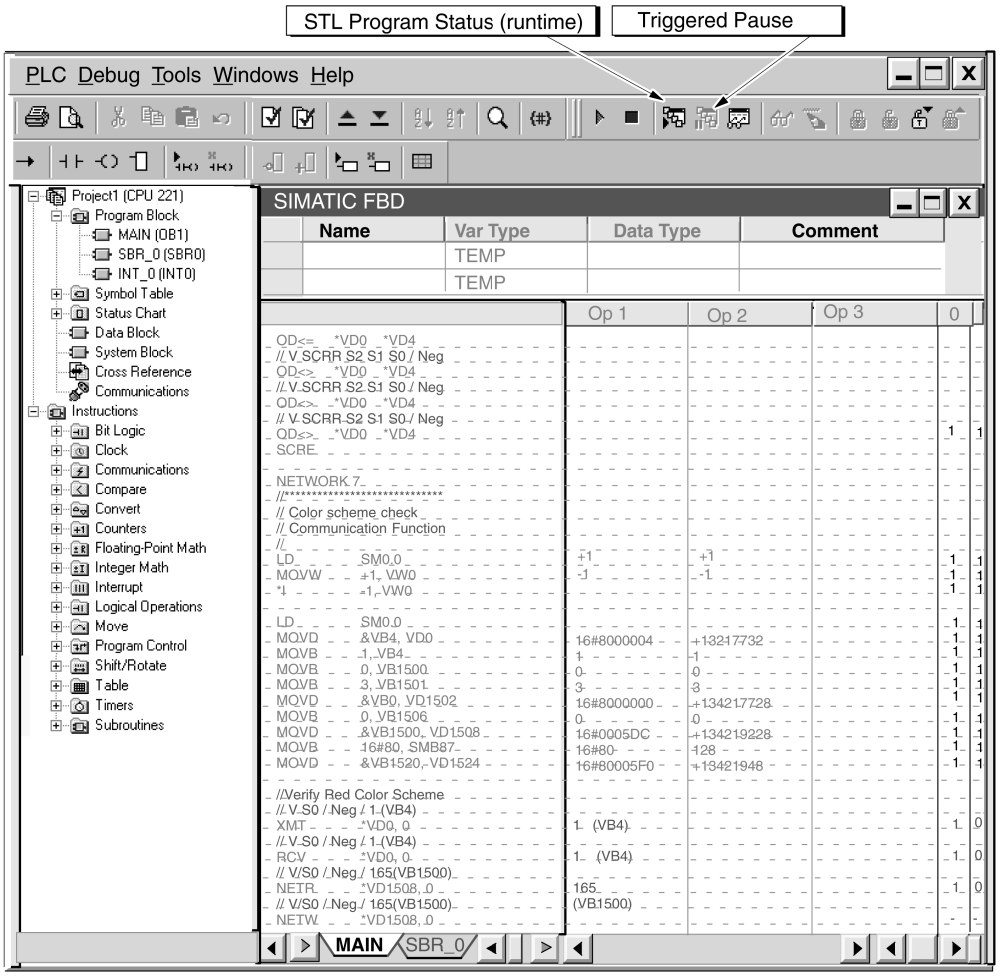


图 4-19

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)。另外你也可以强行改变最多 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4-20 显示了扫描周期中，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。

如图 4–21 所示，可以利用状态表强制数值。要强制一个新值，在状态表的新值栏输入一个值，然后按工具条中的强制钮。强制一个已经存在的值，点亮当前值栏中的值，然后按强制钮。

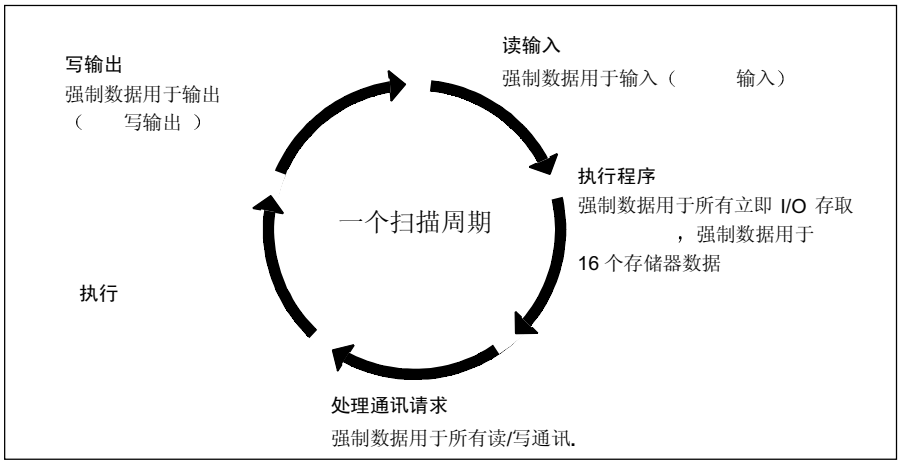


图 4-20 S7-200 CPU 的扫描周期

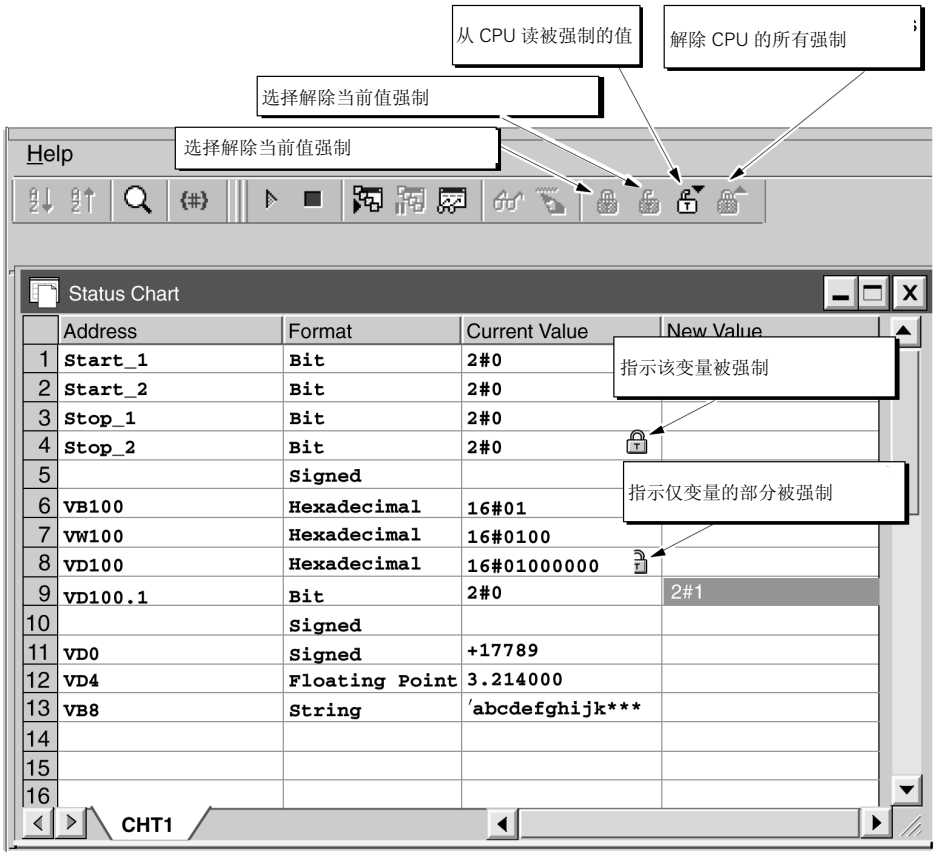


图 4-21 用状态表强制变量

4.10 在 RUN 模式下编辑

CPU 224 1.10 版以上及 CPU 226 1.00 版以上模块均可在 RUN 模式下进行编辑。RUN 模式下编辑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。

警告

当在 RUN 模式下向 CPU 下载修改程序时，修改的程序将立即影响过程操作。在 RUN 模式下修改程序可以导致不可预见的系统运行，可能导致严重的人身伤害和/或财产损失。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授权人员才可以执行 RUN 模式下的编辑。

在 RUN 模式下编辑程序必须满足以下条件：

- 所连接的 CPU 必须支持 RUN 模式编辑功能。
- 所连接的 CPU 必须处于 RUN 模式。

按以下步骤执行：

1. 选择 **Debug>Program Edit in RUN**。（ 见图 4-22 ）
2. 如果项目与CPU中的程序不同，将提示您存盘。RUN模式只能编辑 CPU 中的程序。

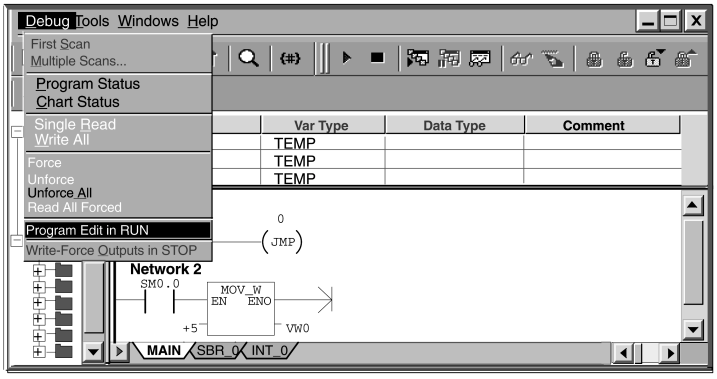


图 4-22 用 FBD 显示程序状态

3. 图 4-23 所示警告。当选择“Continue”后，所连接的CPU中的程序被上载，CPU处于RUN模式下编辑状态。

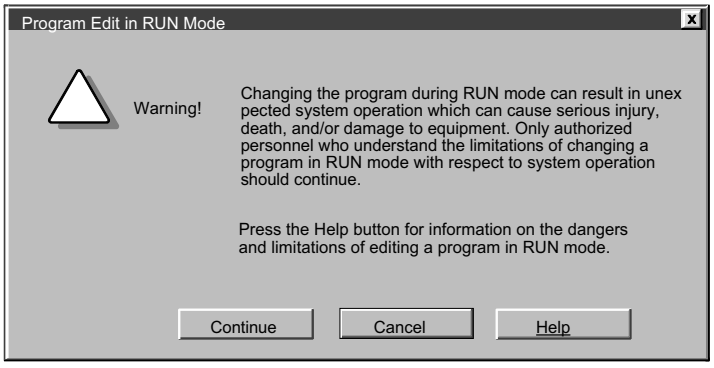


图 4-23 RUN 模式编辑对话框

注意

上升沿和下降沿指令带一个操作数，若需要查看有关沿指令的前一状态的信息，请在屏幕上的示窗部分选择交插参考图标，沿指令使用标签列出了您程序中所使用的沿指令个数，在您编辑程序时请注意不要使用重复的沿号码。

在 RUN 模式下载程序之前

RUN 模式编辑允许在 CPU 处于运行状态时下载您所编辑的程序。决定在 RUN 模式下或在 STOP 模式下下载程序时，应考虑下列因素：

- 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 模式前中断将继续运行。
- 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。

在 RUN 模式下载

在 RUN 模式下载程序，可选择工具条中的下载按钮或从菜单中选择 **File>Download**。程序编译成功后将下载到 CPU。注意，只能将程序块下载到 CPU。下载时应满足以下条件：

- CPU 支持 RUN 模式编辑功能
- 编译过程不许有错误
- 用 STEP 7 Micro/WIN 32 与计算机进行通讯，CPU 必须运行正常。

退出 RUN 模式编辑

退出 RUN 模式编辑，选择 **Debug>Program Edit in RUN**，点击checkmark 即可。如果您修改完后没有存盘，您可选择是否继续编辑、下载和退出 RUN 模式编辑，或不下载退出。

4.11 背景时间

您可以设定扫描循环时间的百分比，这段时间是用来处理通讯请求的，该通讯请求是由运行模式编译或STL状态监控所产生的。当您增加了用于处理通讯请求时间的百分比后，您同时也增加了扫描时间，这使得您的控制过程变慢。

选择背景时间标签页来设置用于背景通讯的扫描循环时间片（图4-24），编辑背景通讯时间的属性，然后将您所做的修改下载到 CPU 中。

缺省的用于处理通讯请求的时间百分比为10%，这个默认值是为了提供一个合理的时间，在处理编译或状态监控的同时，减小对您的控制过程的影响，您可以调节这个数到最大值，最大值是50%，最小增量为5%。

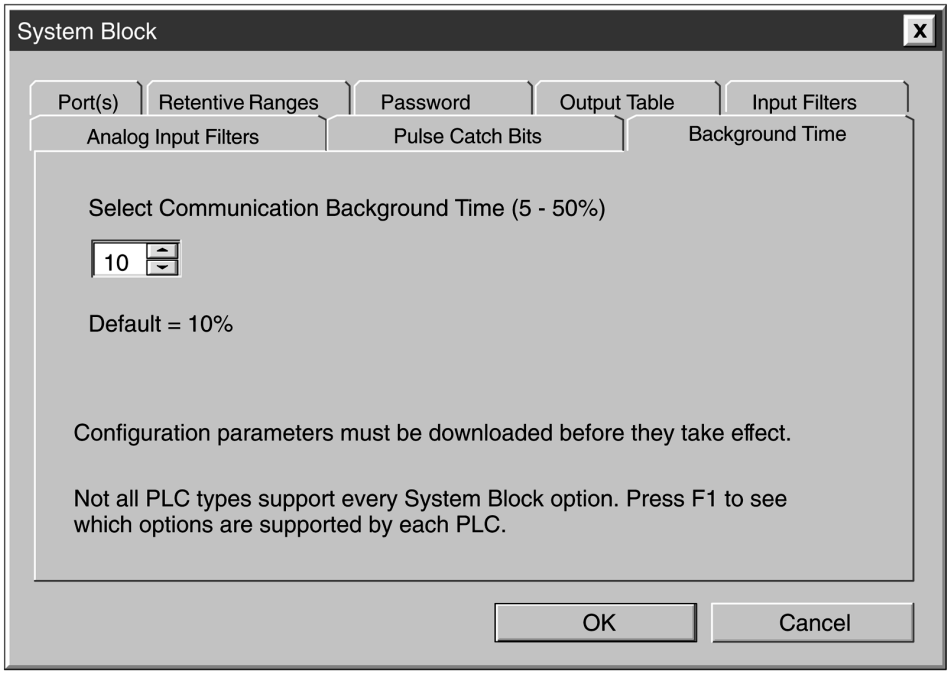


图4-24 背景时间设定画面

4.12 S7–200 CPU 的出错处理

把错误分成致命错误和非致命错误。通过选择主菜单中的 PLC>Information，你可以使用 STEP7-Micro/WIN 32 来查看错误产生的错误代码。图 4-25 是包含错误代码和错误描述的对话框。可参阅附录 B 中完整的错误代码列表。

在图 4–25 中， 最后的致命 (Last Fatal) 区显示 CPU 产生的致命错误代码，如果 RAM 区是保持的，该代码值就一直保持。当 CPU 的所有存储器区清除时或如果 RAM 区非保持时，该区也被清除。

总的致命 (Total Fatal) 区是前一次 CPU 清除所有存储器区后产生的致命错误的次数和。如果 RAM 区是保持的，该值就一直保持。当 CPU 的所有存储器区清除时或如果 RAM 区非保持时，该区也被清除。

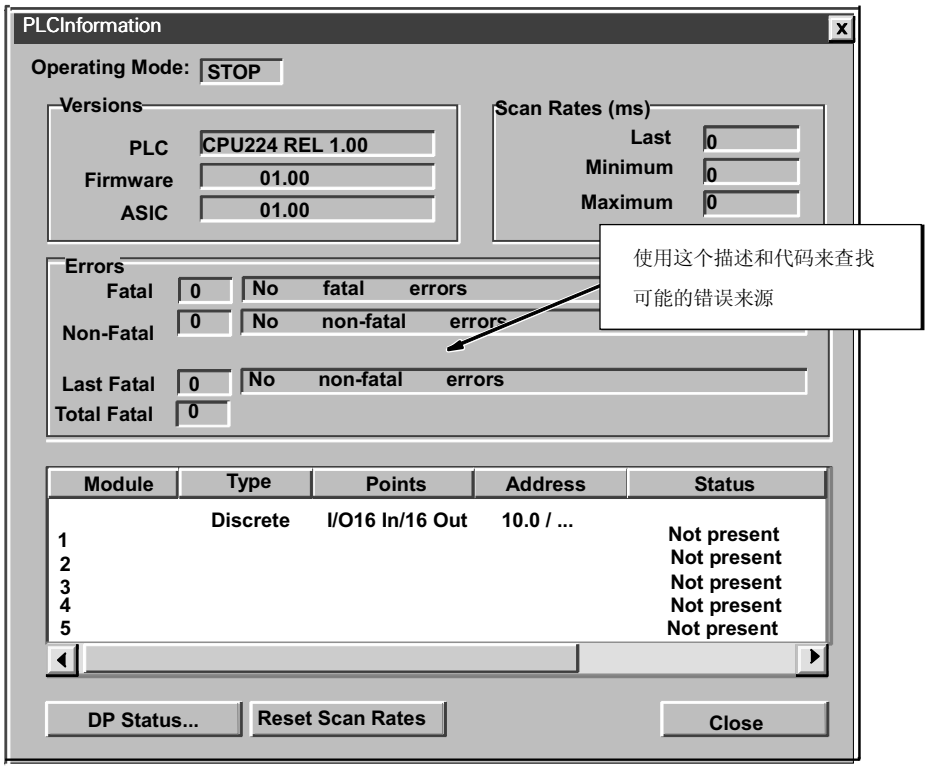


图 4-25 CPU 信息对话框：错误状态表

致命错误响应

致命错误导致 CPU 停止执行程序。根据错误严重程度，它会导致 CPU 无法执行某一或全部功能。处理致命错误的目的是把 CPU 引向安全状态，CPU 可对所存在的错误条件作出询问响应。当 CPU 检测出一个致命错误时，CPU 会变为 STOP 方式，点亮系统错误 LED 和 STOP (停止) LED 指示灯，并关闭 (off) 输出。CPU 会一直保持这种状态，直至消除致命错误条件。

一旦消除了致命错误条件后，必须重新启动 CPU。可以用如下的方法重新启动CPU：

- 通过重启电源
- 将方式开关由 RUN 或 TERM 变为 STOP
- 使用 STEP 7-Micro/WIN 启动 CPU。 在 STEP 7-Micro/WIN 32 的主菜单中选择 **PLC>Powerup Reset**，可以强迫 PLC 重启动并清除所有的致命错误。

重启 CPU 会清除这个致命错误条件，并执行上电诊断测试来确认已改正错误。如果又发现其它致命错误，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。这种情况下你不能看到来自 CPU 的错误代码。这种错误表示硬件出错，CPU 模块需要修理，而修改程序或清除 CPU内存是无法清除这种错误的。

非致命错误响应

非致命错误会导致 CPU 运行的某些方面效率降低，但它们不会使 CPU 无法执行用户程序或更新 I/O。如图 4-25 所示，可以使用 STEP 7 Micro/WIN 32 来查看它们的错误代码。有以下三种基本类型的非致命错误：

- 运行错误：在 RUN 方式下发现的非致命错误会反映在特殊存储器标志位 (SM) 上。用户程序可以监视和鉴定这些位。请参阅附录 C 关于 SM 位用于报告运行时非致命错误的详细信息。

  启动时， CPU 读取 I/O 配置，并存于系统数据存储器和 SM 存储器中，在正常运行过程中，I/O 状态定时更新并存于 SM 存储器中。如果 CPU 发现 I/O 置的差别，CPU 就会在模块错误字节中设置配置改变位。直至这一位复位，I/O 模块不会更新。要使 CPU 对该位复位，I/O 模块必须重新同存于系统数据存储器中的 I/O 的配置相匹配。

- 程序编译错误：CPU 编译程序后方可下装程序。如果 CPU 发现程序违反了编译规则，那么下装会停止，并生成一个错误代码 (已装入 CPU 的用户程序会依然存在于 EEPROM 存储器中，不会丢失)。当修改完程序后，可以重新下装程序。
- 运行时编程错误：当程序运行时，用户程序会产生错误条件。例如，一个编译时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出界地址。这被认为是运行时刻程序错误。使用图 4-25 所示对话框来判断错误类型。

当 CPU 发现了一个非致命错误时，它不会变为 STOP 方式，只是在 SM 存储器中记录该事件，并继续执行程序。然而，你可以设计使程序发现非致命错误时，强制 CPU 进入 STOP 方式。图 4-26 是一个监视 SM 位的程序网络。无论何时发现 I/O 错误，这条指令都会使 CPU 变为 STOP 方式。

图 4-26　设计程序检测非致命性错误

CPU 存储器的数据类型及寻址方式

5

S7-200 CPU 提供了存储器的特定区域，使控制数据的运行更快、更有效。

本章概述

5.1 CPU 存储器区域的直接寻址

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，这样就允许用户程序直接存取这个信息。

使用存储器地址来存取数据

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、字节地址及位号。图 5-1 是一个位寻址的例子 (也称为“字节.位”寻址)。在这个例子中，存储器区以及字节地址 (I= 输入，3= 字节 3) 和位地址 (第 4 位) 之间用点号“ • ”相隔开。

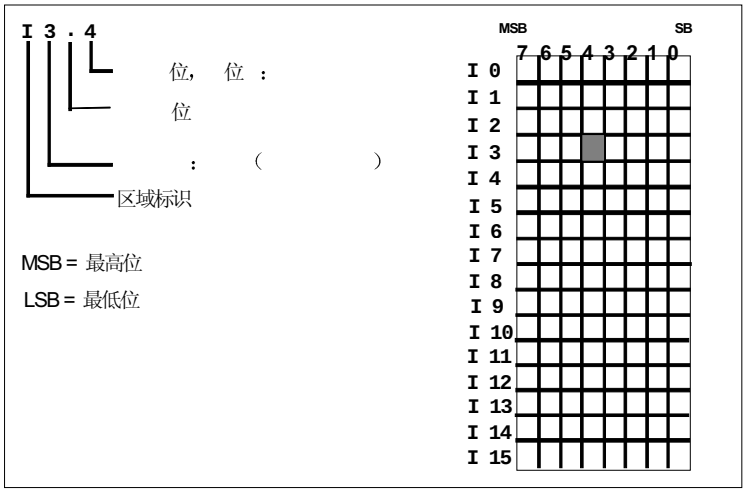


图 5-1 存取 CPU 存储器中的位数据 (字节.位寻址)

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 存储器中的一个字节、字或双字数据，则必须以类似位寻址的方式给出地址，包括区域标志符、数据大小以及该字节、字或双字的起始字节地址，如图 5-2 所示。其它 CPU 存储器区域 (如T，C，HC 以及累加器) 中存取数据使用的地址格式为：区域标识符和设备号。

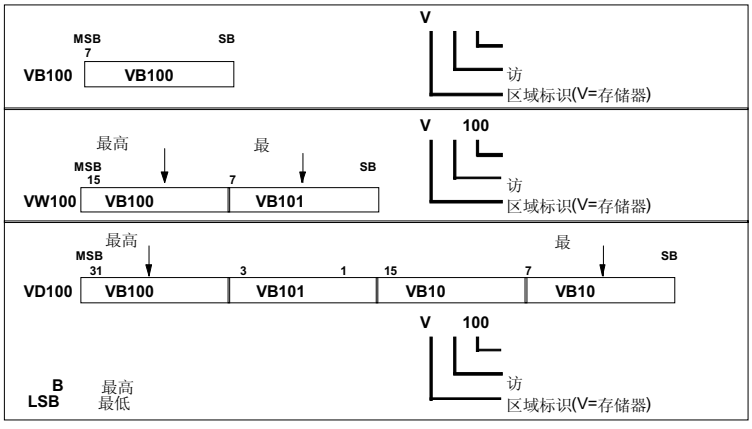


图 5-2 字节、字和双字对同一地址存取操作的比较

数值表示

表 5-1 给出了不同长度的数值所能表示的整数范围。

实数（或浮点数）采用 32 位单精度数来表示，其格式是正数：+1.175495E-38 到 +3.402823E+38；负数：-1.175495E-38 到 -3.402823E+38。按照ANSI/IEEE 754 1985 标准格式，以双字长度来存取。

表 5-1 数据大小规定及相关整数范围

数据大小	无符号整数		有符号整数	
	制	十六进制	十进制	十六进制
B (字节) : 8位值	0 to 255	0 to FF	-128 to 127	80 to 7F
W (字) : 16位值	0 to 65,535	0 to FFFF	-32,768 to 32,767	8000 to 7FFF
D (双字) : 32位值	0 to 4,294,967,295	0 to FFFF FFFF	-2,147,483,648 to 2,147,483,647	8000 0000 to 7FFF FFFF

输入映象寄存器 (I) 寻址

如 4.6 节所述，在每次扫描周期的开始，CPU 对输入点进行采样，并将采样值存于输入映象寄存器中。可以按位、字节、字或双字来存取输入映象寄存器。

格式:

位	I*[字节地址].[位地址]*	I0.1
字节，字，双字	I*[长度][起始字节地址]*	IB4

输出映象寄存器 (Q) 寻址

在扫描周期的结尾，CPU 将输出映象寄存器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映象寄存器。

格式：

位	Q	*[字节地址].[位地址]*	Q1.1
字节，字，双字	Q	*[长度][起始字节地址]*	QB5

变量 (V) 存储器区寻址

程序执行过程中控制逻辑操作的中间结果，也可以使用 V 存储器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。

格式：

位	V	*[字节地址].[位地址]*	V10.2
字节，字，双字	V	*[长度][起始字节地址]*	VW100

位存储器 (M) 区寻址

可以使用内部存储器标志位 (M) 作为控制继电器存储中间操作状态或其它的控制信息。尽管名为“位存储器区”，表示按位存储，但不仅可以按位，也可以按字节、字或双字来存取位存储器区。

格式：

位	M	*[字节地址].[位地址]*	M26.7
字节，字，双字	M	*[长度][起始字节地址]*	MD20

顺序控制继电器 (S) 存储器区寻址

顺序控制继电器位 (S) 用于组织机器操作或进入等效程序段的步。SCR 提供控制程序的逻辑分段，可以按位、字节、字或双字来存取 S 位。

格式：

位	S *[字节地址].[位地址]*	S3.1
字节，字，双字	S *[长度][起始字节地址]*	SB4

特殊存储器 (SM) 标志位

SM 位提供了 CPU 和用户程序之间传递信息的方法。可以使用这些位选择和控制 S7-200 CPU 的一些特殊功能，例如：

- 第一次扫描的 ON 位
- 以固定速度触发位
- 数学运算或操作指令状态位

关于 SM 的详细信息，请参阅附录 C。尽管 SM 区基于位存取，但可以按位、字节、字或 双字来存取。.

格式：

位	SM *[字节地址].[位地址]*	SM0.1
字节，字，双字	SM *[长度][起始字节地址]*	SMB86

局部存储器 (L) 区寻址

S7–200 PLC 有 64 个字节的局部存储器，其中 60 个可以用作暂时存储器或者给子程序传递参数。如果用梯形图或功能块图编程, STEP 7-Micro /WIN 32 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，但是不要使用局部存储器的最后 4 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，而局部存储器是局部有效的。全局是指同一个存储器可以被任何程序存取 (例如，主程序、子程序或中断程序)。局部是指存储器区和特定的程序相关联。S7–200 PLC 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；给中断程序分配 64 个字节。

子程序或中断子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、中断程序或其它子程序的局部存储器。同样地，中断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程序的局部存储器是不存在的。当出现中断或调用一个子程序时，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不同子程序或中断程序的相同局部存储器。

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 代替被传递的参数的值。局部存储器在参数传递过程中不接收值，在分配时不被初始化，也没有任何值。

可以按位、字节、字或双字访问局部存储器。可以把局部存储器作为间接寻址的指针，但是不能作为间接寻址的存储器区。

格式：

位	*L [字节地址].[位地址]*	L0.0
字节，字，双字	*L [长度] [起始字节地址]*	LB33

定时器 (T) 存储器区寻址

S7-200 CPU 中，定时器是累计时间增量的设备。S7-200 定时器精度 (时基增量) 有 1ms，10ms，100ms 三种。有两个相关的变量：

- 当前值：16 位符号整数，存储定时器所累计的时间。
- 定时器位：定时器当前值大于预设值时，该位置为“1”。(预设值作为定时器指令的一部分输入)

可以使用定时器地址 (T＋定时器号) 来存取这些变量。对定时器位或当前值的存取依赖于所用的指令：带位操作数的指令存取定时器位，而带字操作数的指令存取当前值。如图 5-3 所示，常开节点 (T3) 指令存取定时器位，而 MOV_W 指令存取定时器的当前值。关于 S7-200 指令系统的详细信息，请参阅第 9 章的 SIMATIC 指令和第十章的 IEC1131-3 指令。

格式: T *[定时器号]* T24

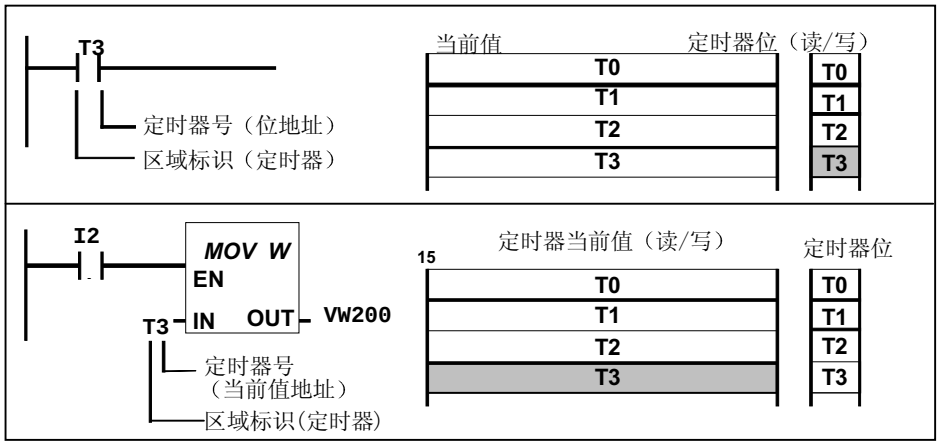


图 5-3 存取 SIMATIC 定时器数据

计数器 (C) 存储器区寻址

在 S7-200 CPU 中，计数器是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；一种是减计数；另一种既可增计数，又可减计数。与计数器相关的变量有两个：

- 当前值：16 位符号整数，存储累计脉冲数。
- 计数器位：当计数器的当前值大于或等于预设值时，此位置为“1”。(预设值作为计数器指令的一部分输入)

可以使用计数器地址 (C＋计数数号) 来存取这些变量。对计数器位或当前值的存取依赖于所用的指令：带位操作数的指令存取计数器位，而带字操作数的指令存取当前值。如图 5-4 所示，常开接点 (C3) 指令存取计数器位，而 MOV-W 指令存取计数器的当前值。关于 S7-200 指令系统的详细信息，请参阅第9章 SIMATIC 指令和第十章的 IEC1131-3 指令。

格式： C*[计数器号]* C20

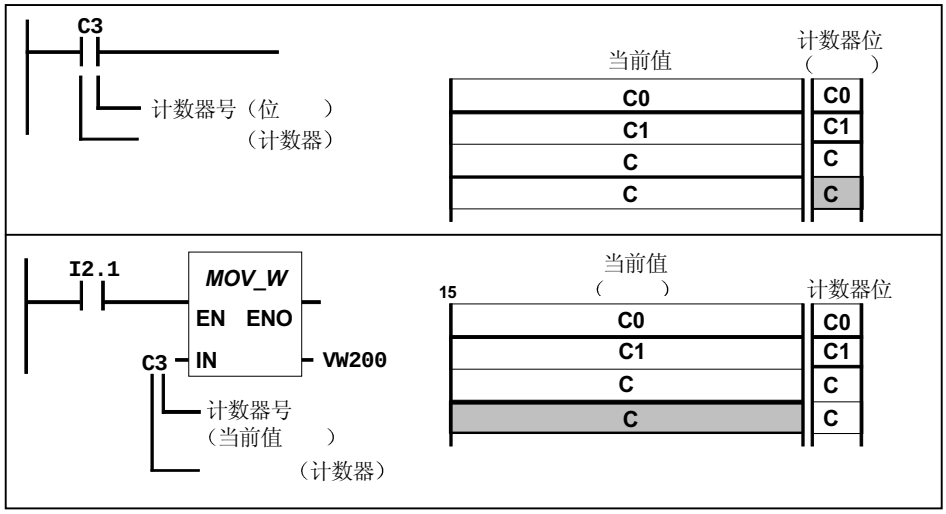


图 5-4　存取计数器数据

模拟量输入 (AI) 寻址

S7-200 将现实世界的模拟值 (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。如图 5-5 所示，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

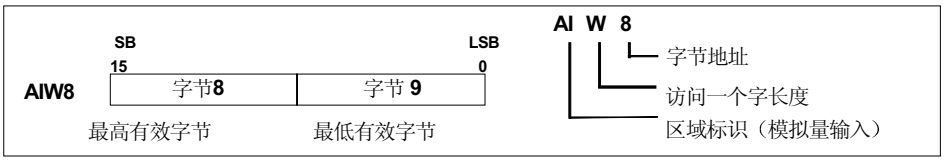


图 5-5　存取模拟量输入值

模拟量输出 (AQ) 寻址

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) 及起始字节地址来置这些值。如图 5-6 所示，因为模拟输出量为一个字长，且从偶数字节 (如 0.2，4) 开始，必须使用偶数字节地址 (如 AQW0，AQW2，AQW4) 来设置这些值。用户程序无法读取这个模拟输出值。

格式：　　AQW *[起始字节地址]*　　AQW4

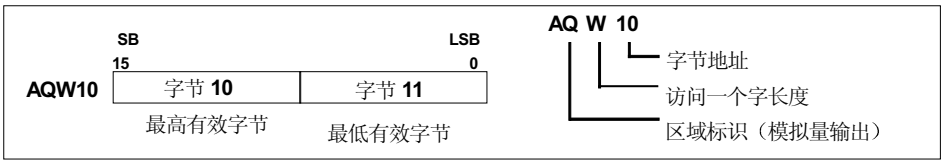


图 5-6　存取模拟量输出值

累加器 (AC) 寻址

累加器是可象存储器那样使用的读 / 写设备。例如，可以用它来向子程序传递参 数，或从子程序返回参数，以及用来存储计算的中间值。CPU 提供了 4 个 32位累加器(AC0，AC1，AC2，AC3)。可以按字节、字或双字来存取累加器中的数值。如图 5-7 所示，按字节、字来存取累加器只使用存于存储器中数据的低8位或低 16 位，以双字来存取要使用全部 32 位。存取数据的长度由所用指令决定。

格式： AC *[累加器号]* AC0

注意：

第九章有关中断程序中累加器的使用信息，请参阅 9.15,SIMATIC 通讯指令

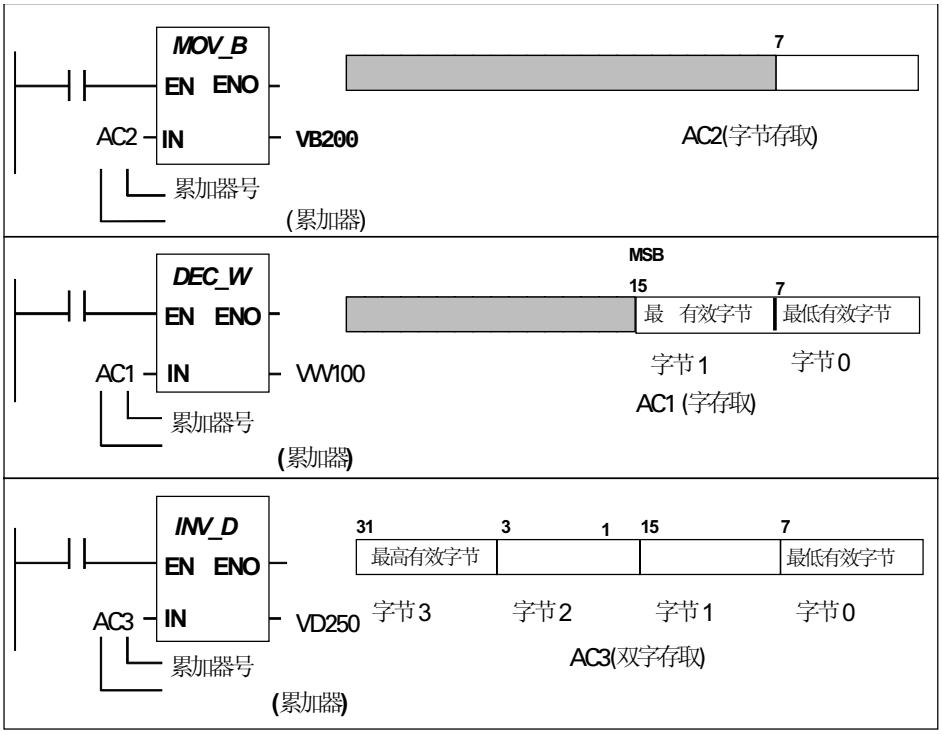


图 5-7 累加器寻址

高速计数器 (HC) 寻址

高速计数器用来累计比 CPU 扫描速率更快的事件。高速计数器有 32 位符号整数累计值(或当前值)。若要存取高速计数器中的值，则必须给出高速计数器的地址，即存储器类型 (HC) 及计数器号 (如 HC0)。如图 5-8 所示，高速计数器的当前值为只读值， 可作为双字 (32 位) 来寻址。

格式: HC *[高速计数器号]* HC1

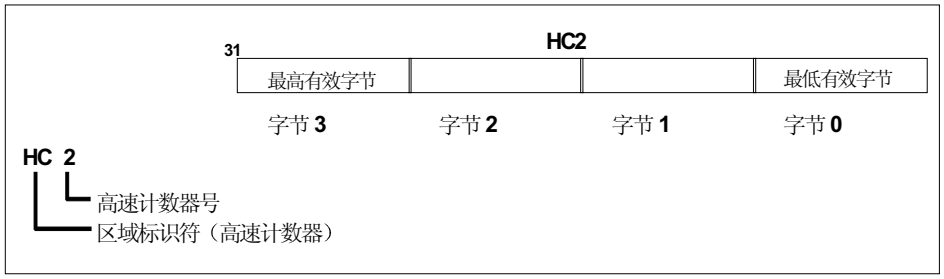


图 5-8 存取高速计数器当前值

使用常数

可以在许多 S7-200 指令中使用常数。常数值可为字节、字或双字。CPU 以二进制方式存储所有常数，也可用十进制、十六进制、ASCII 码或浮点数形式来表示。

十进制格式： [十进制值]
十六进制格式： 16#[十六进制值]
ASCII 格式： '[ASCII码文本]'
实数或浮点格式： ANSI/IEEE 754-1985
二进制格式： 2#1010_0101_1010_0101

S7-200 CPU 不支持“数据类型”或数据的检查 (例如指定常数作为整数、符号整数或双整数来存储)，且不检查某个数据的类型。举例来说，Add 指令可以把 VW100 的值作为一个符号整数来使用，而一条异或指令也可把 VW100 中的值作为一个符号二进制数来使用。

下面是十进制、十六进制、ASCII 及浮点格式常数例子：

- 十进制常数： 20047
- 十六进制常数： 16#4E4F
- ASCII 常数： 'Text goes between single quotes.'
- 实数或浮点格式： +1.175495E-38 (正数)
-1.175495E-38 (负数)
- 二进制格式 2#1010_0101_1010_0101

5.2 CPU 存储器区域的 SIMATIC 间接寻址

间接寻址使用指针来存取存储器中的数据。S7-200 CPU 允许使用指针对下述存储器区域进行间接寻址： I，Q，V，M，S，T (仅当前值) 以及 C (仅当前值)。但不可以对独立的位 (BIT) 值或模拟量进行间接寻址。

建立指针

为了对存储器的某一地址进行间接寻址，需要先为该地址建立指针。指针为双字值，存放另一个存储器的地址。只能使用变量存储区 (V) 、局部存储区 (L) 或累加器 (AC1、AC2、AC3) 作为指针。为了生成指针，必须使用双字传送指令 (MOVD)，将存储器某个位置的地址移入存储器另一位置或累加器作为指针。指令的输入操作数必须使用“&”符号表示某一位置的地址，而不是它的值。把从指针处取出的数值传送到指令输出操作数标识的位置。

例:
```
MOVD    &VB100, VD204
MOVD    &MB4, AC2
MOVD    &C4, LD6
```

使用指针来存取数据

在操作数前面加“*”号表示该操作数为一个指针。如图 5-9 所示，AC1 表示AC1 为 MOVW 指令确定的一个字长的指针。在这个例子中，存于 V200 和V201 中的值被移至累加器 AC0。

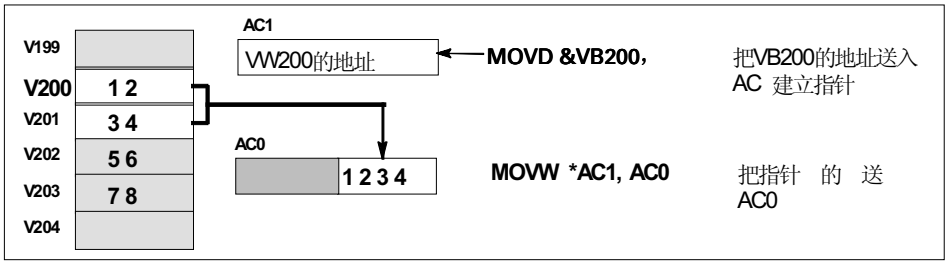


图 5-9 使用指针来间接寻址

修改指针

可以改变一个指针的值。因为指针为 32 位的值，所以使用双字指令来修改指针值。简单的数学运算指令，如加法或自增指令，可用于修改指针值。记住要调整存取的数据的长度：

- 当存取字节时，指针值加1。
- 当存取一个字、定时器或计数器的当前值时，指针值加 2。
- 当存取双字时，指针值加 4。

图 5–10 的例子说明了如何建立间接寻址的指针；如何间接存取数据和如何增加指针。

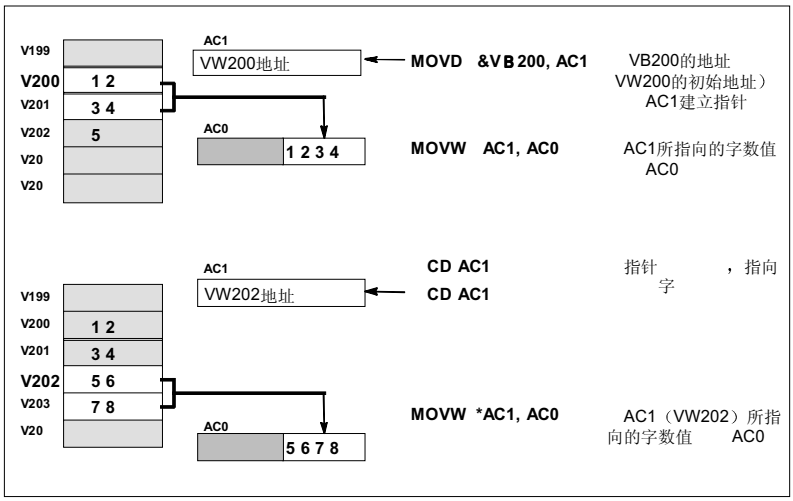


图 5-10 存取字数值时指针的修改

5.3 S7-200 CPU 的存储器保持

S7–200 CPU 提供了几种方法来确保用户程序、程序数据、以及 CPU 的组态数据不丢失，如图 5-11。

- CPU提供了一个EEPROM来永久保持用户程序选择的数据区，以及CPU的组态数据。
- CPU提供一个超级电容器，在 CPU 掉电时保存完整的 RAM 存储器。根据 CPU 模块类型，超级电容器可保存 RAM 存储器达几天之久。
- CPU提供一个可选的电池卡，当 CPU 掉电后，可延长 RAM 存储器保持的时间。电池卡只有在超级电容器耗尽后才提供电源。

这一节讨论在各种情况下，RAM 中数据的永久保存和驻留。

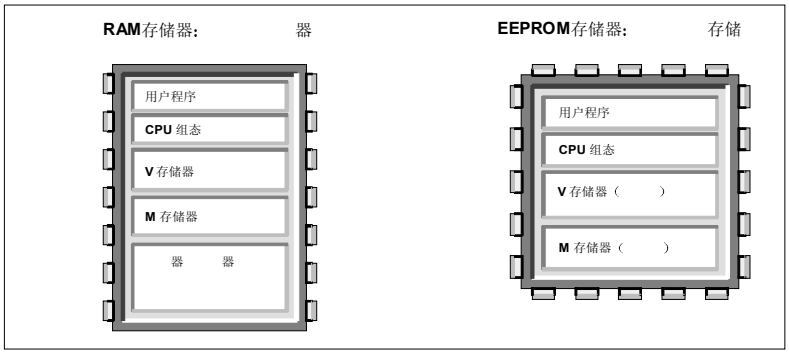


图 5-11 S7-200 CPU 的存储区域

下装和上装用户程序

程序包含三部分：用户程序，数据块 (可选)，CPU 组态 (可选)。如图 5-12 所示，下装的程序存于 CPU 存储器的RAM区。为了永久保存，CPU 会同时自动地把这些用户程序、数据块 (DB1) 以及 CPU 组态拷贝到 EEPROM 中。

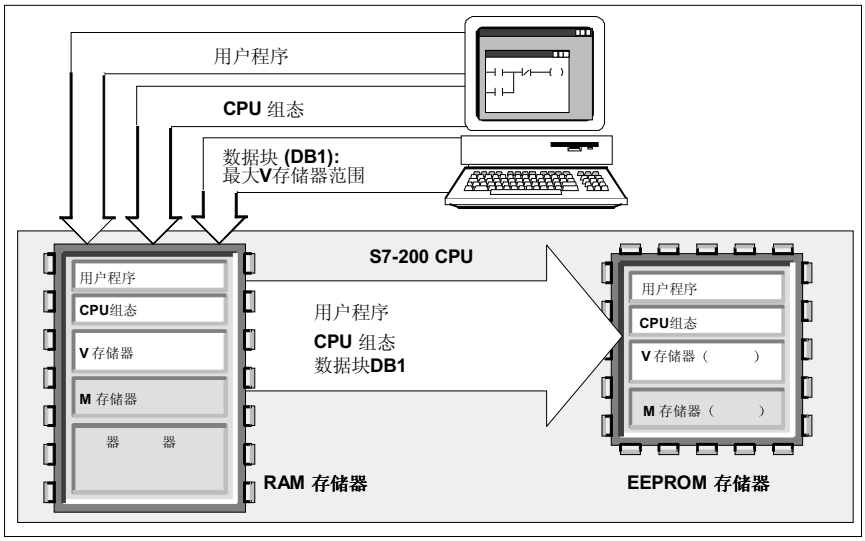


图 5-12　下装程序元素

当从 CPU 上装一个程序时，如图 5-13 所示，用户程序及 CPU 配置从 RAM中上装到个人计算机 (PC)。当上装数据块时，存于 EEPROM 中的永久数据块将同存于 RAM 中剩下的数据块 (如果有的话) 合并，然后把完整的数据块传到个人计算机 (PC) 上。

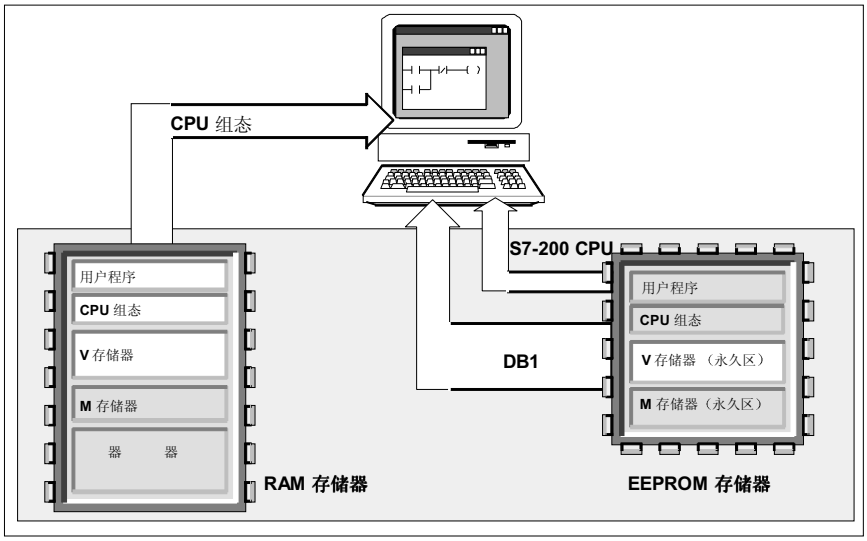


图 5-13　上装程序元素

CPU 掉电时自动保持位存储器 (M) 区域的数据

如果设为保持，则当 CPU 模块掉电时，M 存储器前 14 个字节 (MB0 到 MB13) 会完整保存到 EEPROM 中。如图 5-14 所示，CPU 将这些要保存的 M 存储器区移到 EEPROM 中。

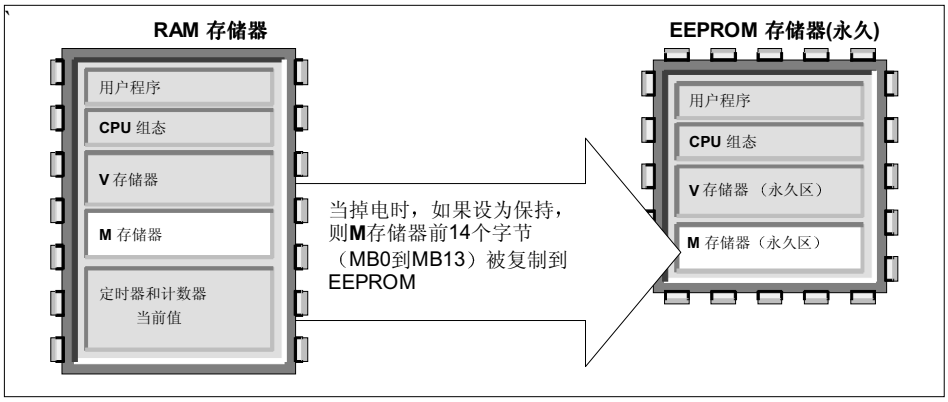


图 5-14　　掉电时把位存储器 (M) 部分存到 EEPROM 中

开机后恢复存储器

开机后，CPU 会从 EEPROM 向 RAM 中恢复用户程序和 CPU 配置，如图 5-15 所示。

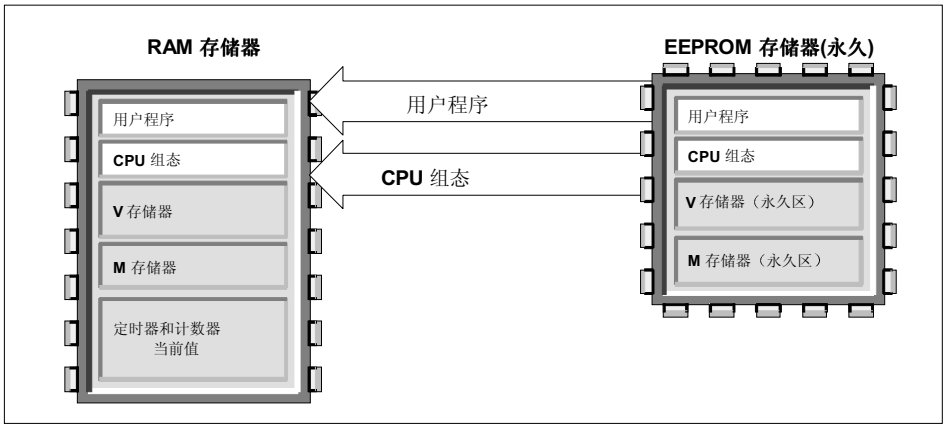


图 5-15　　开机时恢复用户程序和 CPU 配置

开机后，CPU 检查 RAM 存储器，确认超级电容器是否已成功保存了 RAM 存储器中的数据。如果成功保存，那么 RAM 存储器的保持区域将保持不变。如图 5-16 所示，V 存储器中的未保持区域，将从相应的 EEPROM 中的V存储器永久区域处恢复回来。

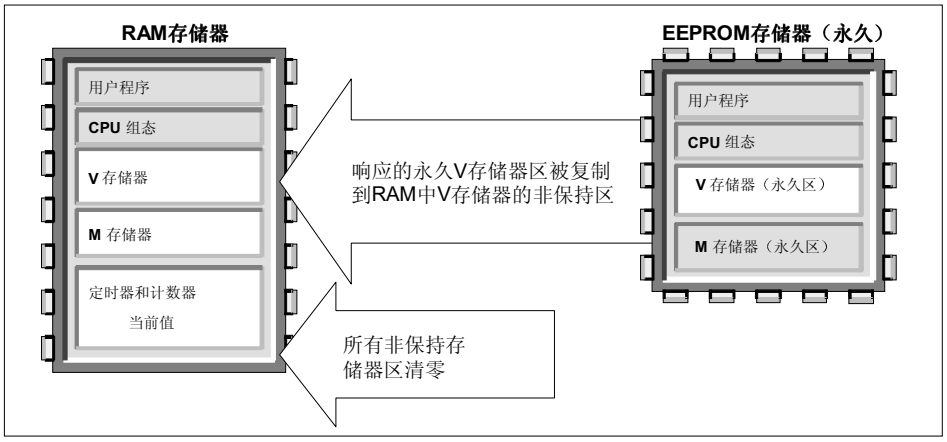


图 5-16　　开机时恢复程序数据 (数据成功地保留在 RAM中)

如果 RAM 存储器的内容没有保持下来 (如在意外掉电后)，CPU 会清除 RAM存储器 (包括保持和非保持区) 并置保持数据丢失存储器标志位 (SM0.2) 为“1”。如图 5-17 所示，当开机后，存于 EEPROM 永久区域中的数据会复制到 RAM 存储器中。

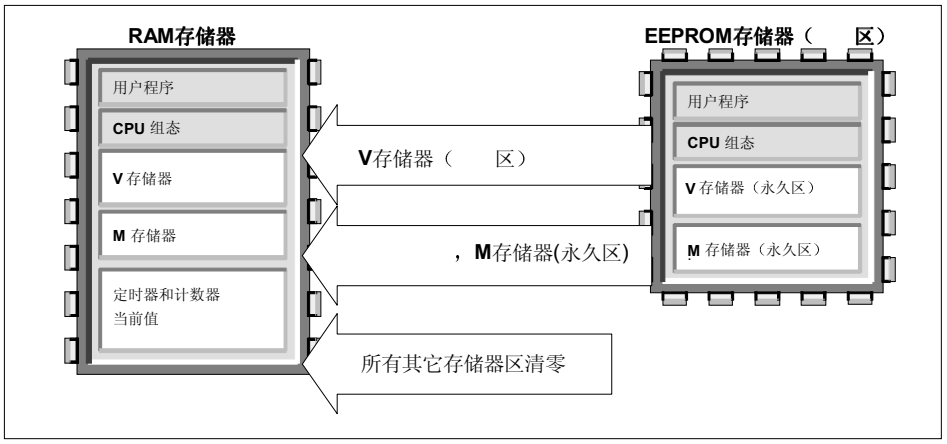


图 5-17　　开机时恢复程序数据 (数据不保留在 RAM 中)

定义存储器保持范围

如图 5-18 所示，当电源掉电时，最多可以定义六个可选的要保持的存储器区。你可以定义以下存储器区的保持地址范围：V，M，C和T。对于定时器，只有TONR 可以保持。在 STEP 7-Micro/WIN 32 中，缺省设置是 M 存储器的最开始的14个字节是不保持的。缺省情况是不允许 CPU 断电保存。

注意：

定时器和计数器只有当前值可被保持，而定时器位和计数器位是不能保持的。

为了定义存储器保持范围，选择菜单命令 **View>System Block,** 点击 Retentive Ranges 块。图 5-18 为定义保持范围的对话框。如果选择 CPU 的缺省保持范围，则按 **Defaults** 按钮。

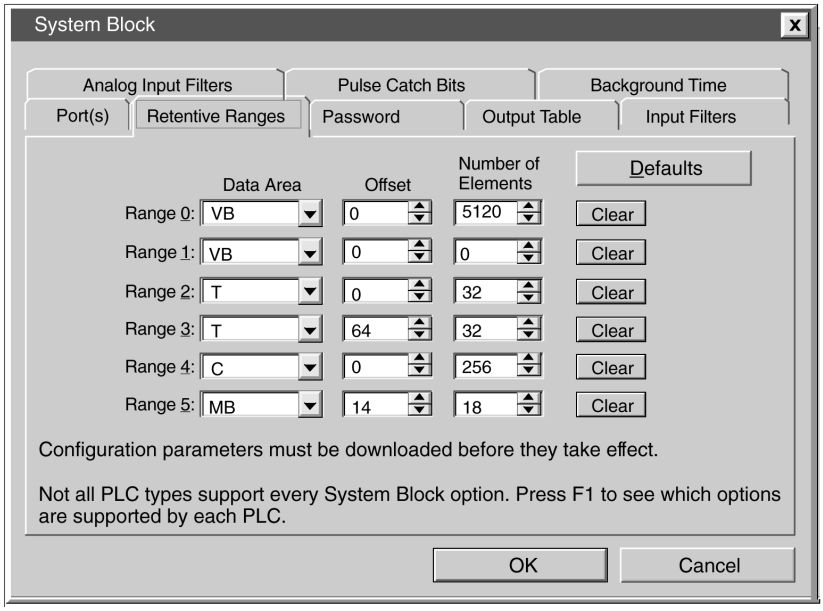


图 5-18　　设置 CPU 存储器的保持范围

5.4 由用户程序来永久保存数据

可以将存于存储器上的数据 (字节、字或双字) 备份到 EEPROM 存储器。这项功能可用于保存 V 存储器区任意位置的数据。

存一次 EEPROM 操作会把扫描时间增加 15 到 20ms。保存操作所写的数据会覆盖先前 EEPROM 中 V 存储器区的数据。

注意：

存 EEPROM 操作并不更新存储器卡中的数据

复制V存储器到 EEPROM

特殊存储器字节 31 (SMB31) 和特殊存储器字 32 (SMB32) 命令 CPU 复制 V 存储器中的一个数据到 EEPROM 中的 V 存储器区。图 5-19 为 SMB31 和SMB32 的格式。采用下述步骤来保存或写 V 存储器中的一个指定值。

1. 将要保持的 V 存储器地址置于 SMW32。
2. 将数据长度写入 SM31.0 和 SM31.1。 (见图 5–19)
3. 设置 SM31.7 = 1。

在每次扫描的末尾，CPU 自动检查 SM31.7，如果 SM31.7等于 1，则将指定的数据存于 EEPROM。当 CPU 将 SM31.7 置为 0 时，操作结束。在保存操作完成之前，不要改变 V 存储器中的值。

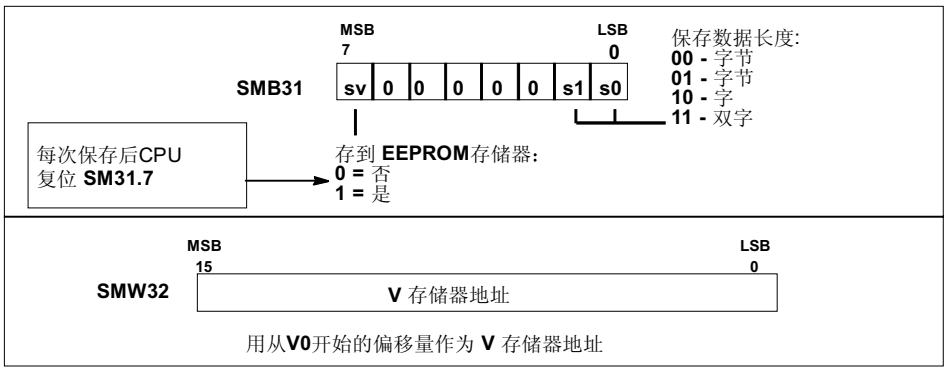


图 5-19 SMB31 和 SMB32 的格式

限制存 EEPROM 次数

因为存 EEPROM 操作的次数是有限制的 (最少 10 万次，典型为 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。原则上，当特殊事件发生时，才执行存操作，而这种事件又是不很频繁发生的。

例如，如果 S7-200 扫描时间为 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 至少可工作 11 年。

5.5 使用存储器卡来保存用户程序

CPU 支持可选的存储器卡，为用户程序提供一个便携式 EEPROM 存储器。你可以象磁盘那样来使用存储器卡。CPU 在存储器卡上存储下列元素：

- 用户程序
- 存于 EEPROM 的 V 存储器永久区的数据
- CPU 组态

有关存储器卡的详细内容，请参阅附录 A。

复制到存储器卡

只有当 CPU 在停机方式下通电且安装存储器卡时，你才可以从 RAM 存储器中复制程序到存储器卡。

警告：

静电放电可以损坏存储器卡或 CPU 接口。

当你拿存储器卡时，你应当使用接地垫或戴接地手套，还应当把存储器卡存于导电容器里。

当 CPU 通电时，你可以安装或卸下存储器卡。为了安装存储器卡，应当去掉存储器卡接口的保护带，把存储器卡插入 CPU 模块存取盖下面的接口(存储器卡的正确安装是关键)。当安装完存储器卡后，使用下列步骤来复制程序。

1. 将 CPU 置于停机状态。
2. 如果程序未下装入 CPU，那么应下装程序。
3. 使用菜单命令 **PLC>Program Memory Cartridge** 来向存储器卡中复制序。 图 5-20 显示了存于存储器卡中的 CPU 存储器元素。
4. 卸下存储器卡 (可选)。

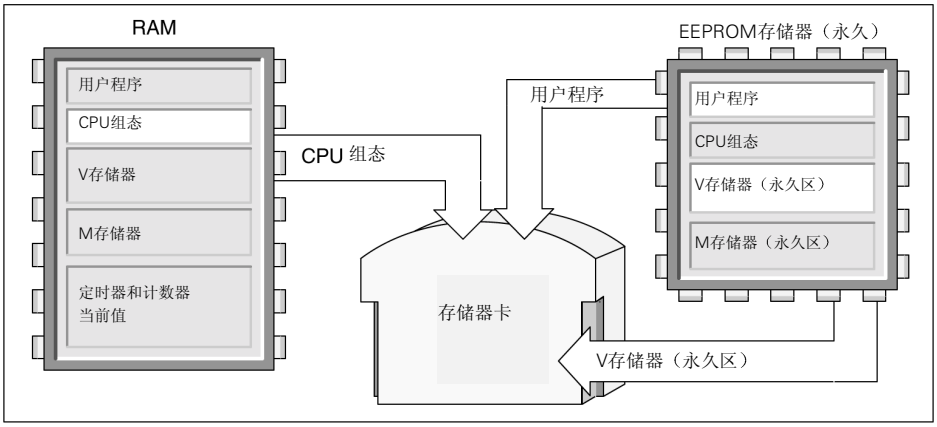


图 5-20 复制 CPU 存储器到存储器卡

用存储器卡恢复程序和存储器

为了从存储器卡向 CPU 传送程序，你必须打开装有存储器卡的 CPU 电源。如图 5-21 所示，电源接通后，CPU 执行下列任务 (当安装存储器卡后)：

- 清除 RAM 存储器
- 复制存储器卡内容到 RAM 存储器。
- 用户程序编程，CPU 组态和拷贝到 EEPROM。

注意：

在 CPU 通电时，若存储器卡为空，或对不同型号的 CPU 进行编程，则会出现错误。高型号的 CPU 可以读出用低型号 CPU 编程的存储器卡，反之则不能读出，例如 CPU 224 可以读出 CPU 221 或 CPU 222 所编写的存储器卡程序，但 CPU 224 在存储器卡中所编写的程序，CPU 221 或 CPU 222 不能读取。卸下存储器卡，再次通电，存储器卡可以插入并编程。

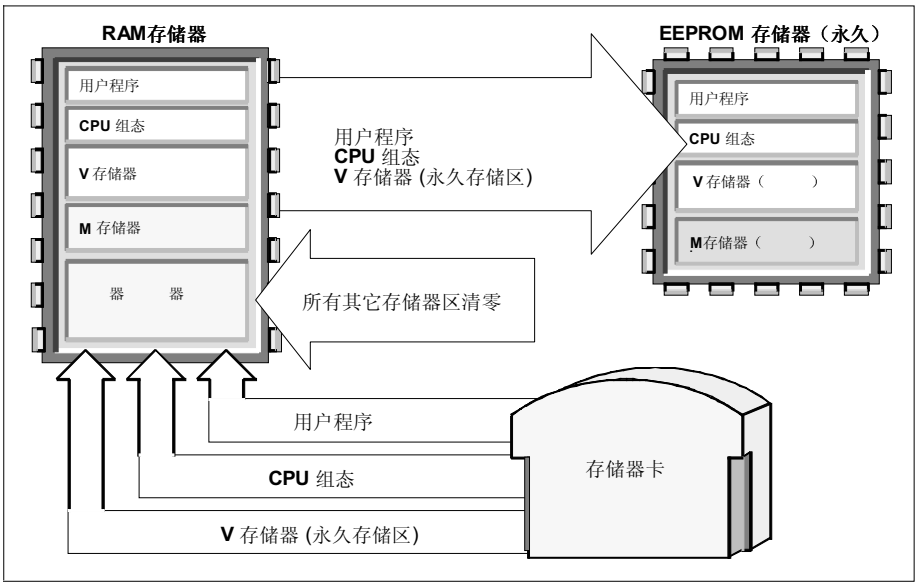


图 5-21 (用安装的存储器卡) 当 CPU 通电时恢复存储器

CPU 和输入/输出控制组态

6

输入和输出是系统控制点，输入信号来自现场设备 (如传感器和开关)，而输出则控制生产过程中的泵、电动机或其它设备。你可以使用本机 I/O (由 CPU 模块提供) 或扩展 I/O (由扩展 I/O 模块提供)。S7-200 CPU 也提供高速 I/O。

本章概述

6.1 本机 I/O 和扩展 I/O

输入和输出是系统控制点：输入信号来自现场设备 (如传感器和开关)，而输出则控制生产过程中的泵、电动机或其它设备。你可以使用本机 I/O (由 CPU 提供) 或扩展 I/O (由扩展 I/O 模块提供)：

- S7-200 CPU 具有一定数量的本机 I/O 数字点。关于 CPU 模块的本机 I/O 数量，请参阅附录 A 中的数据表。
- S7-200 CPU 222、CPU 224 和 CPU 226 支持外加的数字和模拟扩展 I/O。关于各种扩展 I/O 的详细信息，请参阅附录 A 中的数据表。

本机 I/O 及扩展 I/O 寻址

CPU 模块提供的本机 I/O 具有固定的 I/O 地址。你可以把扩展 I/O 模块接至CPU 右边来增加 I/O 点。该模块点的地址由 I/O 类型及模块在 I/O 链中的位置决定 (对于同种类型的输入输出模块而言)。举例来说，输出模块不会影响输入模块的点地址，反之亦然。类似的，模拟模块不会影响数字模块，反之亦然。

CPU 给离散或数字扩展模块总是保留以 8 位 (1个字节) 递增的映象寄存器空间。如果模块不给物理点提供每个保留字节的每个位，那么这些未用位就不会分配给 I/O 链中的后续模块。对于输出模块而言，保留字节中的未用位可象内部存储器标志位那样来使用。对于输入模块，每次输入更新时都把保留字节的未用位清零，因此不能用作内部存储器标志位。

模拟量扩展模块总是以 2 字节递增方式来分配空间。如果模块不给物理点提供这些空间，那么这些 I/O 点的空间就会掉失，且不能分配给 I/O 链中的后续模块。

本机 I/O 和扩展 I/O 例子

图 6-1 和 6-2 为不同硬件配置如何影响 I/O 数目的例子。注意，有些配置含有用户程序无法寻址的空位，而其它 I/O 地址可象内部存储器 (M) 标志位那样来使用。

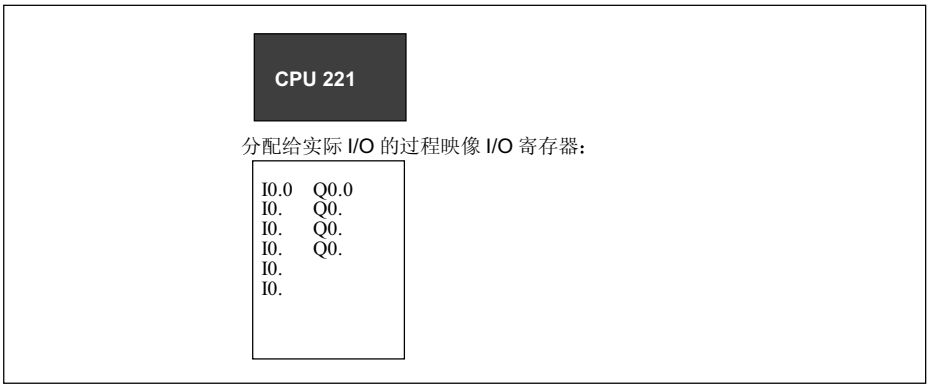


图 6-1 CPU 221 的 I/O 点数举例

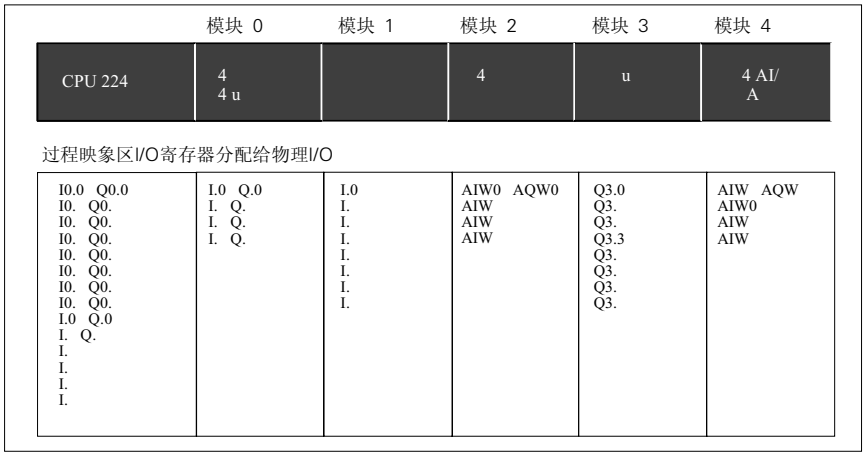


图 6-2 CPU 224 的 I/O 点数举例

6.2 使用可选的输入滤波器来抑制噪声干扰

S7-200 CPU 允许为某些或全部本机数字量输入点选择输入滤波器，并可定义延迟时间(从 0.2ms 到 12.8ms 可选)，对于特定的 CPU 的参数请参阅附录A。如图 6-3 所示，这个延迟时间给四个为一组的输入点加上标准响应时间。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 CPU 配置数据的一部分，可下装且存于 CPU 存储器中。为了设置输入滤波器延迟时间，应使用菜单命令 **View>System Block**，选择输入滤波器选项。

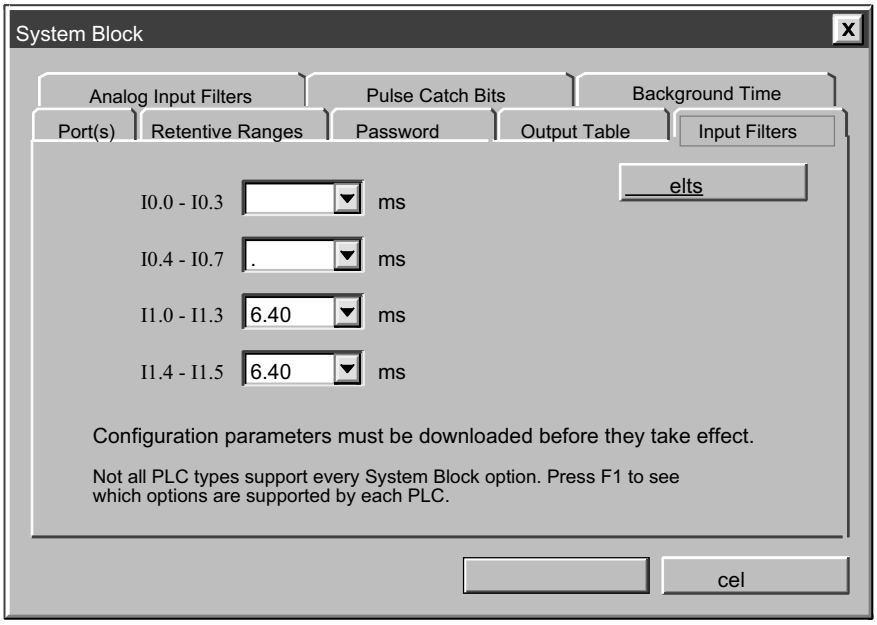


图6-3 为了抑制噪声而配置输入滤波器

6.3 脉冲捕捉

S7–200 CPU 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低电平脉冲。而在扫描周期的开始，CPU 不是总能读取这些数字量输入。

可以对每个本机数字量输入设置允许脉冲捕捉操作。当一个输入设置为脉冲捕捉时，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。用这样的方法，一个持续时间很短的脉冲信号被捕捉到，并保持到 CPU 读到该输入信号，确保不丢失脉冲。带有和不带有脉冲捕捉功能的 PLC 的基本操作如图 6-4 所示。

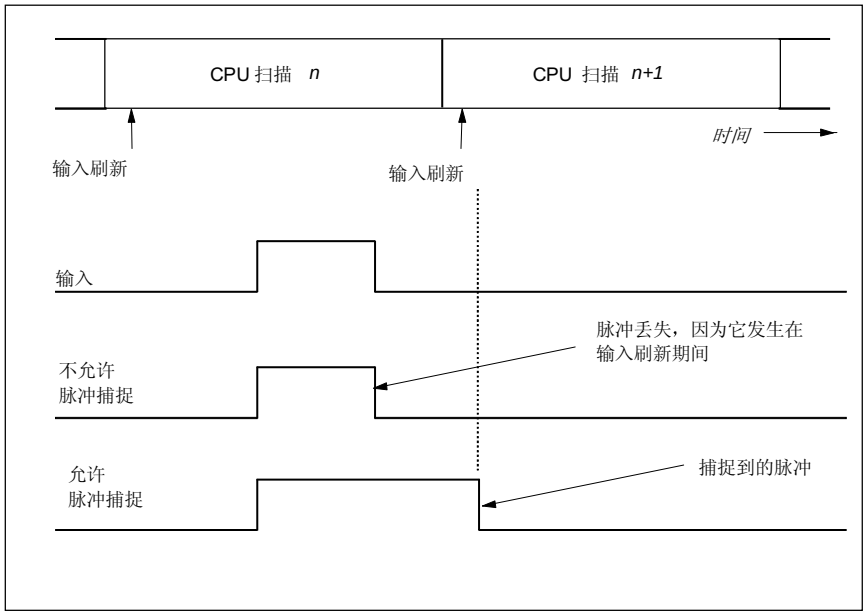


图6-4 带有和不带有脉冲捕捉功能的PLC操作

使用脉冲捕捉功能时，必须保证要把输入滤波器的时间调整到不会滤掉脉冲。 (在通过输入滤波器后，脉冲捕捉功能才能有效。)

数字输入电路的方块图如图6-5所示。

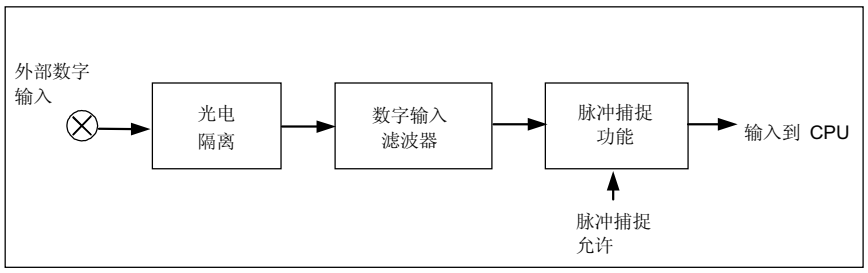


图 6-5 数字输入电路

对不同输入条件的脉冲捕捉电路的响应如图6-6所示。如果在一个给定的扫描周期内有不只一个脉冲时，则只有第一个脉冲可以读到。此时可用9.15节中的I/O中断处理多个脉冲。

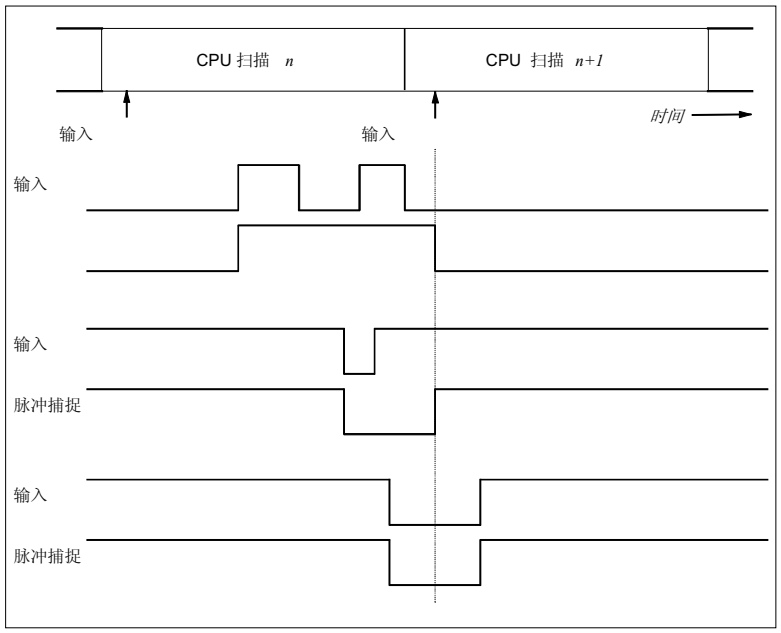


图 6-6 脉冲捕捉举例

从主菜单中选择菜单命令 **View>System Block** 并点击 Pulse Catch Bits 标签可以进入脉冲捕捉配置屏。图 6–7 是 Pulse Catch 配置屏。CPU 和 STEP 7-Micro/ WIN 32 的缺省设置是禁止所有的输入脉冲捕捉。

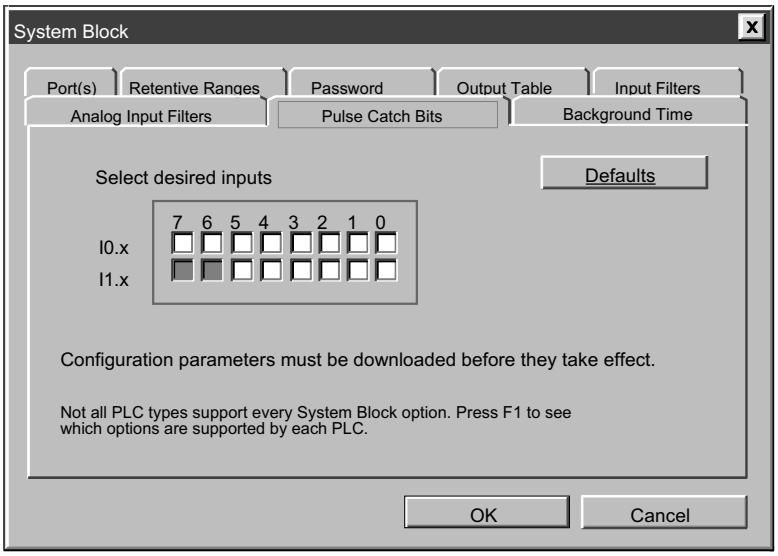


图6-7 脉冲捕捉配置屏

6.4 使用输出表来设置输出状态

S7–200 CPU 为输出点提供两种性能，一种是预置数字量输出点在 CPU 变为STOP 方式后为已知值，另一种是设置数字量输出保持 CPU 变为 STOP 方式之前的状态。

输出表是 CPU 配置数据的一部分，要下装且存入 CPU 存储器中。

输出值设置仅适用于数字量输出。模拟量输出值在 CPU 切换到 STOP 方式后即被锁定，因为用户程序负责刷新模拟量输出。CPU 并没有更新模拟量输入和输出的系统功能。CPU 没有为这些点提供内部映象存储器。

选择菜单命令 **View>System Block**，点击 Output Table 选择块来使用输出表设置对话框，见图 6–8。设置输出时，你有以下两种选择：

- 如果你想保持上一次的输出，那就选择 Freeze Output 框，并点击“OK”。
- 如果你想把输出表中的值复制到输出点上，则填写输出表值。点击你想要的相应位，则从 RUN 转到 STOP 方式后，该位便置为 1 (On)。点击“OK”来保存你的选择。

STEP 7-Micro/WIN 32 的缺省组态和 CPU 缺省设置是把输出表的值复制到输出点上。而输出表的缺省值全为 0 (Off)。

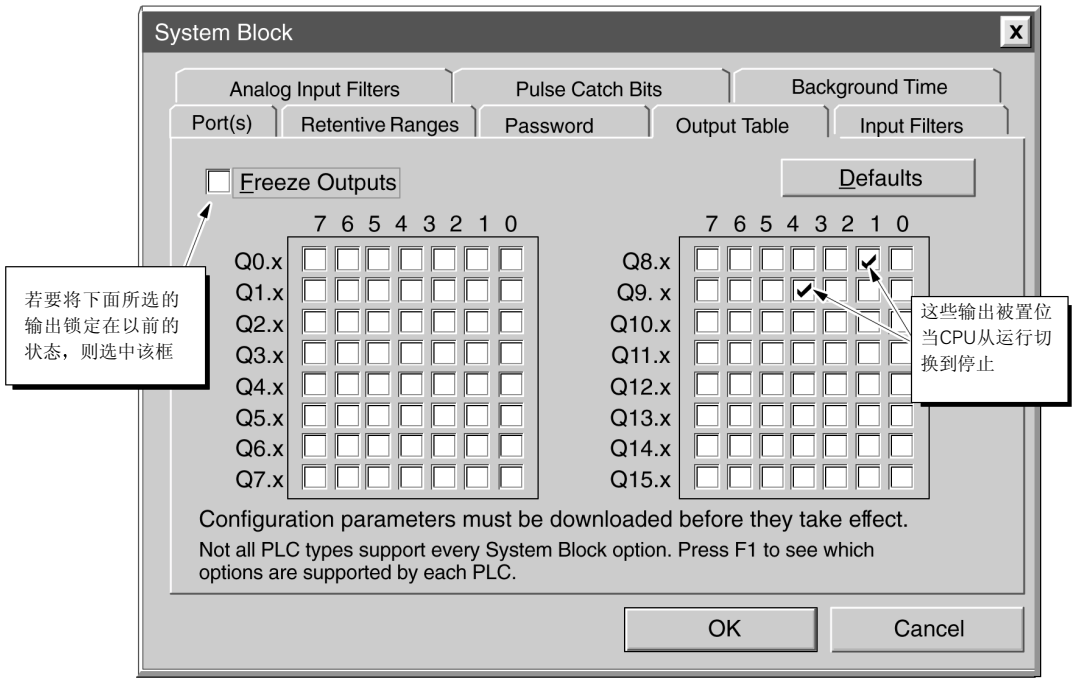


图 6-8 设置输出状态

6.5 模拟输入滤波器

对 CPU 222、CPU 224 和 CPU 226，可以对不同的模拟输入选择软件滤波器。滤波值是模拟量输入设定个数的采样值和的平均值。滤波器参数（采样次数和死区）对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的变化是一个阶跃函数。这个差称为死区，并用模拟量输入的数字信号设定。

注意：

那此需要利用模拟量控制字传递数字量信息或报警信息的模块不能使用模拟量滤波功能。使用图6-9所示屏幕禁止对RTD、热电耦和AS-i主站模块进行模拟量滤波

选择菜单命令 View > System Block 并点击 Analog Input Filters 选择块，可以使用模拟量输入滤波器。选择需要滤波的模拟量输入并点击“OK”，见图 6–9。STEP 7-Micro/WIN 32 的缺省配置是允许所有的输入滤波。

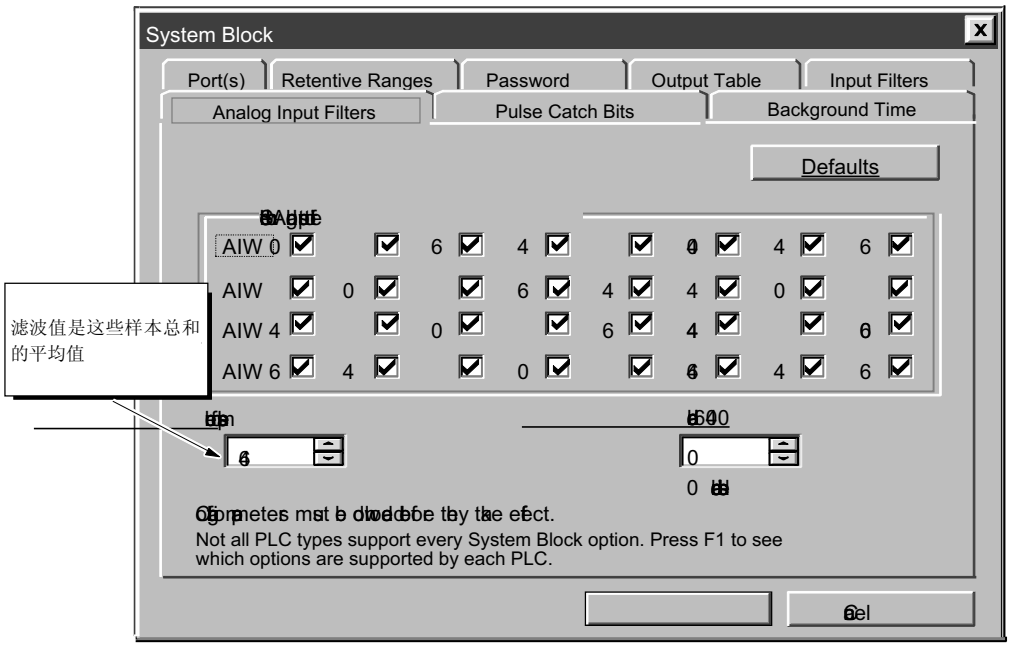


图 6-9 模拟输入滤波器

6.6 高速 I/O

S7–200 CPU 为控制高速事件提供了高速 I/O。关于每种 CPU 模块的高速 I/O的详细信息，请参阅附录 A。

高速计数器

S7–200 CPU 具有集成高速计数器功能，这些计数器可以记录 20 kHz 的事件，而不影响 CPU 的性能。下面说明每个高速计数器：

- HSC0 和 HSC4 是多用途计数器，可以配置成八种不同操作模式中的任何一种，包括单相和两相时钟输入。
- HSC1 和 HSC2 是多用途计数器，可以配置成十二种不同操作模式中的任何一种，包括单相和两相时钟输入。
- HSC3 和 HSC5 是简单计数器，只有一种操作模式 (单相时钟输入)。

表 6–1 定义了高速计数器 HSC0、HSC3、HSC4 和 HSC5 支持的操作模式。所有的S7–200 CPU支持这些高速计数器。

表 6-1 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	时钟	-	-	时钟	时钟	-	-	时钟
1	时钟	-	复位	-	时钟	-	复位	-
2	-	-	-	-	-	-	-	-
3	时钟	方向	-	-	时钟	方向	-	-
4	时钟	方向	复位	-	时钟	方向	复位	-
5	-	-	-	-	-	-	-	-
6	增时钟	减时钟	-	-	增时钟	减时钟	-	-
7	增时钟	减时钟	复位	-	增时钟	减时钟	复位	-
8	-	-	-	-	-	-	-	-
9	A相时钟	B相时钟	-	-	A相时钟	B相时钟	-	-
10	A相时钟	B相时钟	复位	-	A相时钟	B相时钟	复位	-
11	-	-	-	-	-	-	-	-

从这个表可以看出如果以模式 3 到模式 10 (时钟和方向或任何两相时钟模式) 使用HSC0，就不能使用 HSC3，因为 HSC0 和 HSC3 都使用 I0.1。对于HSC4 和 HSC5 也如此，它们使用 I0.4。

可以使用 I0.0 到 I0.3 作为高速计数输入，或把这些输入配置成边沿中断输入。 不能同时对这些输入使用边沿中断输入和高速计数输入。

同一个输入不能用于两个不同的功能，但是，高速计数器没有使用的输入可以用于其他用途。例如，如果 HSC0 工作在模式2，该模式使用 I0.0 和 I0.2，I0.1可以用于边沿中断或 HSC3。

表 6–2 定义了高速计数器 HSC1 和 HSC2 支持的操作模式。只有 CPU 224 和 CPU 226 支持这些高速计数器。

表 6-2 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	时钟	-	-	-	时钟	-	-	-
1	时钟	-	复位	-	时钟	-	复位	-
2	时钟	-	复位	启动	时钟	-	复位	启动
3	时钟	方向	-	-	时钟	方向	-	-
4	时钟	方向	复位	-	时钟	方向	复位	-
5	时钟	方向	复位	启动	时钟	方向	复位	启动
6	增时钟	减时钟	-	-	增时钟	减时钟	-	-
7	增时钟	减时钟	复位	-	增时钟	减时钟	复位	-
8	增时钟	减时钟	复位	启动	增时钟	减时钟	复位	启动
9	A相时钟	B相时钟	-	-	A相时钟	B相时钟	-	-
10	A相时钟	B相时钟	复位	-	A相时钟	B相时钟	复位	-
11	A相时钟	B相时钟	复位	启动	A相时钟	B相时钟	复位	启动

每个计数器都有专用的输入时钟、方向控制、复位、以及启动功能。在正交方式下，可以选择 1X 或 4X 最高计数频率。HSC 1 和 HSC 2 互相独立，且不影响其它的高速功能，二者都可互不影响地以最高频率运行。

关于高速计数器的详细信息，请参阅第 9 章 SIMATIC 高速计数器指令中 9.4节。

高速脉冲输出

S7–200 CPU 支持高速脉冲输出，Q0.0 和 Q0.1可以产生高速脉冲序列输出 (PTO) 或产生脉冲宽度调制 (PWM) 控制。

- PTO 输出方波 (占空比 50%)，并可指定所输出的脉冲数量和周期时间。脉冲数可指定为 1 到 4,294,967,295。周期既可以以微秒 (s)，也可以以毫秒 (ms) 为单位，设为 50 到 65,535 微秒或 2 到 65,535 毫秒。使用任何奇数微秒或毫秒 (如 75ms) 会引起脉冲周期占空比变形。脉冲序列输出 (PTO) 功能可以编程为产生一列脉冲或编程为产生由多个序列脉冲组成的脉冲包络。在脉冲包络操作方式中，PTO 功能被编程控制一个步进电机运行一个简单的斜坡上升，运行和斜坡下降顺序或更复杂的顺序。脉冲包络由 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 2 到 65,535 毫秒。脉宽可从 0 到 65,535 微秒或 0 到 65,535 毫秒。当脉宽等于周期时，占空比为 100%，输出恒定。当脉宽等于0时，占空比为 0，没有输出。

关于高速输出的详细信息，请参阅第9章 SIMATIC 高速计数器指令中 9.4 节。

6.7 模拟量电位器

提供了一个或两个模拟电位器 (位于模块盖下方的电位器)。你可以调节这些电位器来增加或降低存于特殊存储器 (SMB28 和 SMB29 字节) 中的值。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，输入或修改预设值，或设置限值。

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

你可以使用一个小螺丝刀来进行调节：将电位器顺时针 (向右) 旋转来升高值，逆时针 (向左) 旋转来减小值。图 6-10 为使用模拟电位器的示例程序。

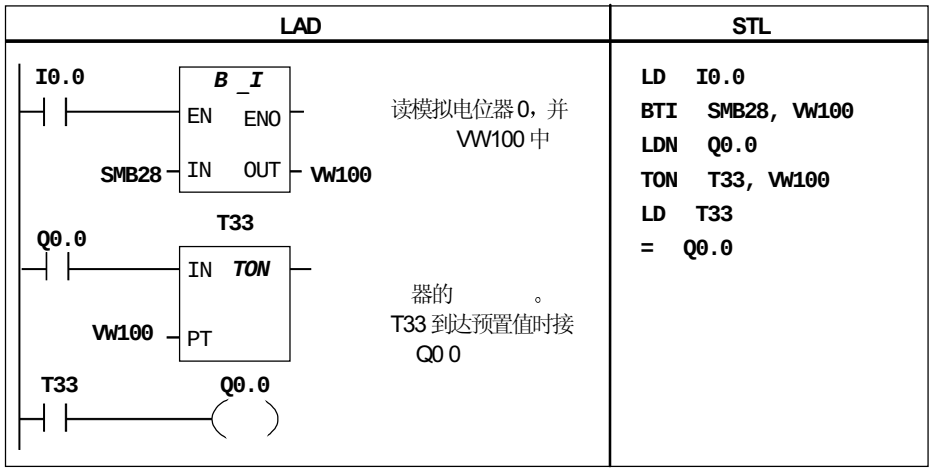


图 6-10 模拟电位器举例

设置通讯硬件和网络通讯

7

本章描述利用 STEP 7-Micro/WIN 32 3.1 版的通讯，该软件的以前版本可能操作不一样。本章也讲述如何设置通讯硬件和如何设置 S7–200 通讯网络。

本章概述

7.1 我的通讯选择是什么?

为了支持网络通讯，可以用不同的组态安排 S7–200 CPU。可以在有 Windows 95，Windows 98 或 Windows NT 操作系统的个人计算机上安装 STEP 7-Micro/WIN 32 软件，或者在 SIMATIC 编程器 (例如 PG740) 上安装。在下面的通讯组态中你可以把 PC 或编程器作为主站：

- 单主站：单主站连到一个或多个从站。见图 7-1。
- 多主站：单主站连到一个或多个从站和一个或多个主站。见图 7-2。
- 对11–位调制解调器用户：单主站连到一个或多个从站。该主站通过11–位调制解调器连到一个作为从站的 S7-200 CPU 或 S7-200 CPU 作为从站的网络。
- 对 10– 位调制解调器用户：单主站连到仅仅一个通过 10– 位调制解调器作为从站的 S7-200 CPU。

图 7–1 和图 7–2 给出了个人计算机连到几个 S7-200 CPU 的组态。STEP 7-Micro/WIN 32 设计成每次和一个 S7-200 CPU 通讯。但是，你可以访问网络上的任何 CPU。CPU 可以是从站或主站。TD 200 是一个主站。关于网络通讯的详细信息，请参阅 7.5 节。

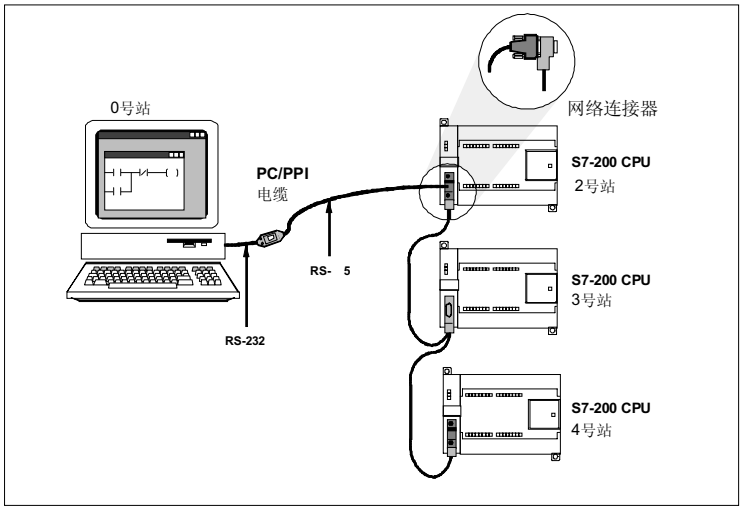


图 7-1 利用 PC/PPI 电缆和几个 S7-200 CPU 通讯

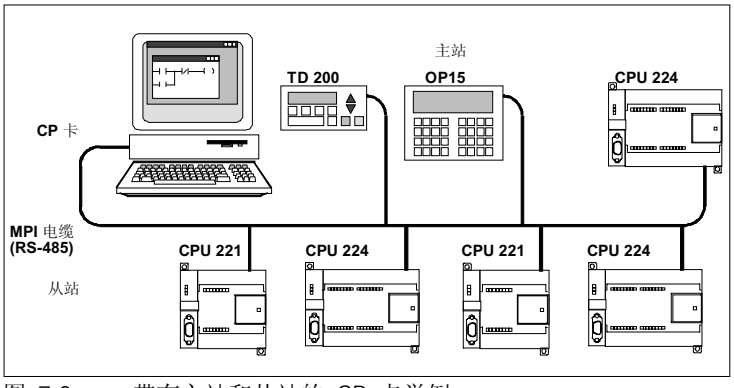


图 7-2 带有主站和从站的 CP 卡举例

如何选择通讯组态?

表 7-1 给出了 STEP 7-Micro/WIN 32 支持的可能的硬件组态和波特率。表 7-2 给出了 S7-200 CPU 和 EM 277 PROFIBUS-DP 模块的性能。

表 7-1 STEP 7-Micro/WIN 32 支持的硬件组态

支持的硬件	型 号	支持的波特率	说 明
PC/PPI 电缆	到 PC 通讯口的电缆连接器	9.6 k 波特 19.2 k 波特	支持 PPI 协议
CP 5511	II 型，PCMCIA–卡	9.6 k 波特 19.2 k 波特 187.5 k 波特	支持用于笔记本 PC 的 PPI、MPI 和 PROFIBUS 协议
CP 5611	PCI– 卡 (版本3 或以上版本)		支持用于PC的 PPI、MPI 和 PROFIBUS 协议
MPI	PG 中集成的 PC ISA– 卡		

表 7-2 S7-200 和 EM 277 模块的性能

连接口	支持的波特率	逻辑连接数	支持的协议
S7–200 CPU			
0 口	9.6 kbps	每个口 4个	PPI，MPI，和 PROFIBUS
1 口	19.2 kbps 187.5 kbps 187.5 kbps		
EM 277 PROFIBUS-DP 模块			
每个 CPU 多达2个	9.6 kbps 到 12 Mbps	每个模块 6个	MPI 和 PROFIBUS

利用 CP 或 MPI 卡进行数据通讯

西门子提供几种可以插入个人计算机或 SIMATIC 编程器的网络接口卡。这些卡可以使计算机或 SIMATIC 编程器作为网络主站。这些卡具有专门的硬件来帮助个人计算机或编程器管理多主站网络，而且可以在几个波特率上支持不同的协议。见表 7–1。

在 STEP 7-Micro/WIN 32 中利用 PG/PC 接口可以设定特定的卡和协议。请参阅 7.3 节。当使用 Windows 95，Windows 98 或 Windows NT 时，可以选择网络卡支持的任何协议 (PPI、MPI 或 PROFIBUS)。

每个卡提供一个连到 PROFIBUS 网络的 RS–485 接口。 CP 5511 PCMCIA 卡有一个提供 9 针 D 型插头的适配器。可以把MPI电缆的一端连到 CP 卡的 RS–485 接口，另一端连到网络上的一个编程口，见图 7–2。 有关通讯处理器卡的详细信息请参阅*全集成 ST 70 中的 SIMATIC 部件*。

在何处设置通讯?

可以在 Windows 95、Windows 98 或 Windows NT 4.0 下设置通讯：

- 安装 STEP 7-Micro/WIN 32 软件的最后一步。
- 在 STEP 7-Micro/WIN 32 内部

如何在 STEP 7-Micro/WIN 32 中设置通讯?

在 STEP 7-Micro/WIN 32 中有一个设置通讯的对话框，在其中可以配置有关的通讯设置。利用下面的一种方法可以找到这个对话框：

- 选择菜单命令 View>Communications。
- 在 STEP 7-Micro/WIN 32 窗口内单击通讯图标 (见图 7–3)。

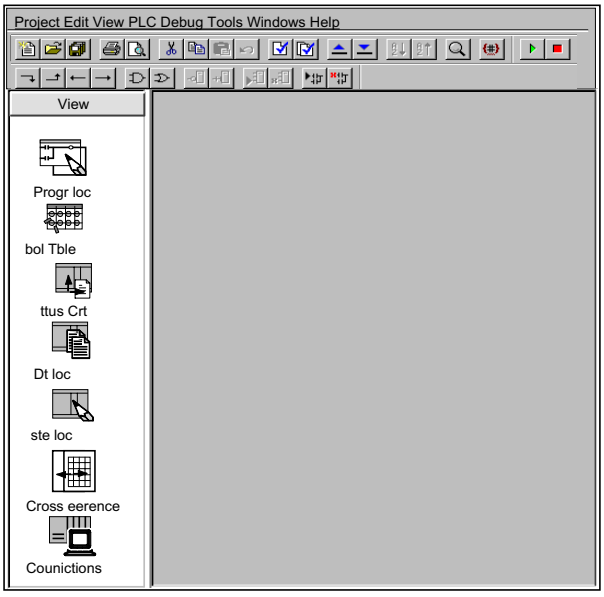


图 7-3　STEP 7–Micro/WIN 32 的菜单显示

双击通讯设置对话框中右上方的图标，将出现一个设置PG/PC接口的对话框，见图 7-4。

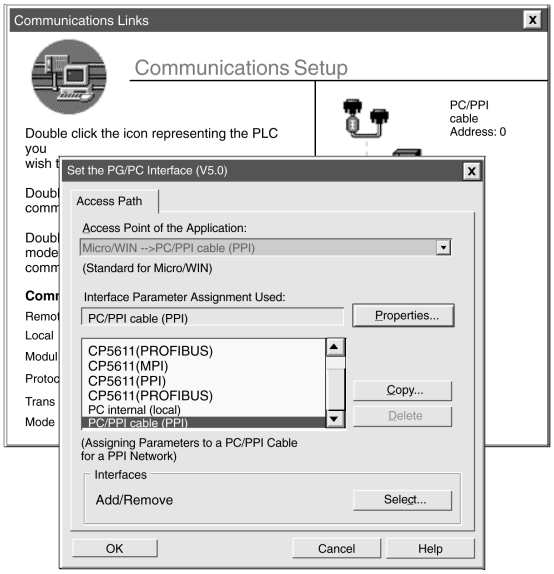


图 7-4　设置 PG/PC 接口的对话框

7.2 通讯接口的安装和删除

利用图 7-5 的安装/拆卸接口对话框可以安装和删除通讯硬件。在该对话框的左侧是一个还没有安装的硬件型号表；在该对话框的右侧是一个已经安装的硬件型号表。如果你使用的是 Windows NT 4.0 操作系统，在已经安装的设备表下面有一个 “Resources” 按钮。

安装硬件：

按照下面的步骤安装硬件：

1. 在设置 PG/PC 接口对话框 (见图 7–4) 中，单击“Select”按钮可以打开图7-5 所示的安装 / 删除接口对话框。
2. 从选择列表框中选一个你使用的硬件型号。在下面的窗口中给出了所选项目的 描述。
3. 单击 “Install - ->” 按钮。
4. 当完成安装硬件后，单击 “Close” 按钮，就出现设置 PG/PC 接口对话框，在已经采用的接口参数列表中可以看到刚才选择的硬件。(见图 7-4)。

删除硬件：

按照下面的步：

1. 从右边的已经安装设备列表中选择要删除的硬件。
2. 单击 “< - - Uninstall” 按钮。
3. 当完成删除硬件后，单击 “Close” 按钮。就出现设置 PG/PC 接口对话框，在已经采用的接口参数列表中可以看到刚才选择的硬件。(见图 7-4)。

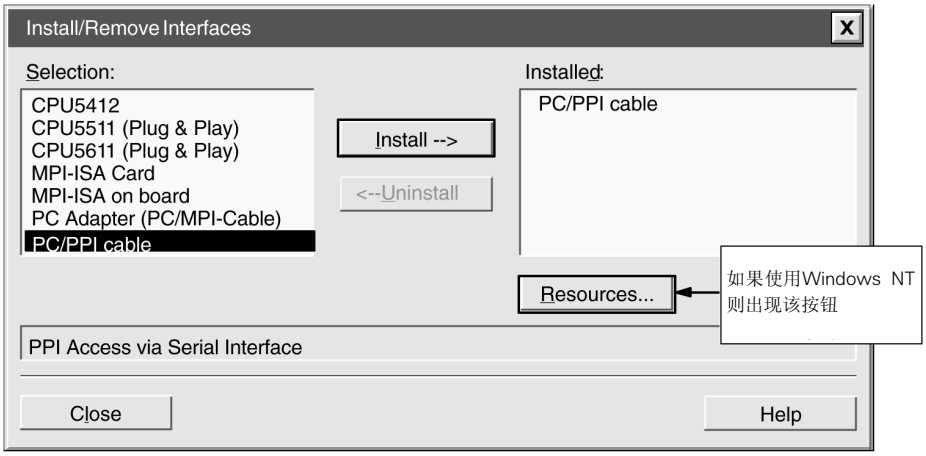


图 7-5 安装 / 删除接口对话框

Windows NT 用户的特殊硬件安装信息

在 Windows NT 操作系统安装硬件模块与在 Windows 95 下安装硬件模块有细微的差别。尽管对操作系统来说硬件模块是一样的，在 Windows NT 下安装需要更多的要安装的硬件知识。Windows 95 自动地为你设置系统资源，而 Windows NT 则不能。Windows NT 只提供缺省值，这些值与硬件配置可能匹配或可能不匹配。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（图 7–5 ），就出现资源 (Resources) 对话框（见图 7-6 ）。 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)，说明你不需要再做任何修改。

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。

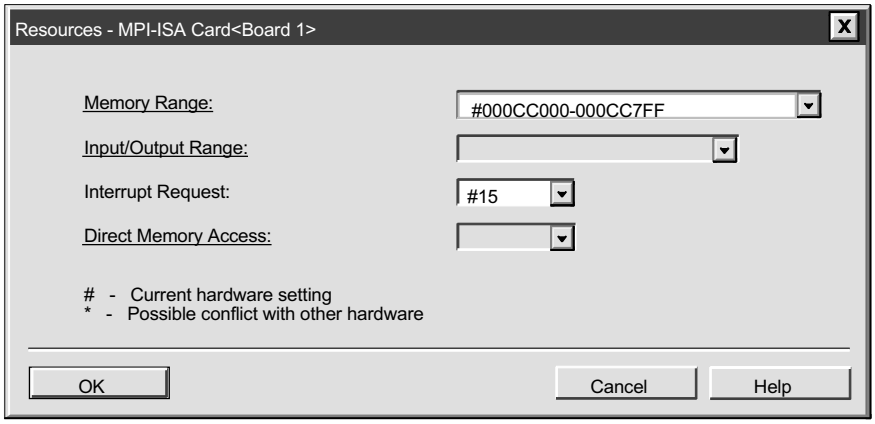


图 7-6 Windows NT 的资源对话框

注意：

如果你使用 Windows NT 和 PC/PPI 电缆，在网络中不能有其它的主站。

7.3 参数选择与修改

选择正确的接口参数并设置它

当打开设置 PG/PC 接口对话框时，要确保“Micro/WIN”出现在应用列表框 (见图 7-4) 中的访问接点中。对于几个不同的应用程序 (例如 STEP 7 和WinCC)，设置 PG/PC 是一样的，所以你可能需要选择为哪个应用程序设定参数。

当已经选择“Micro/WIN”并已经安装硬件时，需要为硬件设置通讯实际属性。第一步是决定网络中要采用的协议，应该为所有的 CPU 选用 PPI 协议。

当已经决定要采用的协议后，可以从设置 PG/PC 接口对话框中列出的接口参数设定中选择正确的设置。这个框列出了已经安装的每个硬件以及于括号中的协议类型。例如，一个简单的配置要求利用 PC/PPI 电缆与一个 CPU 222 通讯，在该例中选择“PC/PPI 电缆 (PPI) ”。

在选择正确的接口参数设置后，必须为当前组态设置个别参数。单击设置 PG/PC 接口对话框中的“Properties...”按钮。根据你所选择的参数设置（见图 7-7 ），这个操作可能引出几个可能的对话窗。

下面的几节详细说明每个对话框。

总之，按照下面的步骤选择接口参数设置：

1. 在设置 PG/PC 接口对话框（见图 7-4 ）中，从访问路径标示签的访问接点应用列表中选择“Micro/WIN”。
2. 确保硬件已经安装，见 7.2 节。
3. 决定要采用的协议，应该为所有 CPU 选择 PPI 协议。
4. 在设置 PG/PC 接口对话框下，从接口参数设置列表框中选择正确的设置。
5. 单击设置 PG/PC 接口对话框中的“Properties...”按钮。

现在，根据所选的参数设置进行选择。

设定 PC/PPI 电缆（PPI）参数

本节说明如何为 Windows 95、Windows 98 或 Windows NT 4.0 操作系统和为 PC/PPI 电缆设置 PPI 参数。

如果使用 PC/PPI 电缆，在设置 PG/PC 接口对话框中单击“Properties...”按钮，就出现一个 PC/PPI 电缆的属性窗口，见图 7–7。

当与 S7-200 CPU 通讯时，STEP 7-Micro/WIN 32 缺省设置为多主站 PPI 协议。 这个协议允许 STEP 7-Micro/WIN 32 与网络中的其它主站设备（ TD 200 和操作面板）共存。通过检查 PG/PC 接口中 PC/PPI 电缆属性对话框的 “Multiple Master Network”可以允许该工作方式。 Windows NT 4.0 不支持多主站选项。

STEP 7-Micro/WIN 32 也支持单主站 PPI 协议。当使用单主站协议时， STEP 7-Micro/WIN 32 假设它是网络中的唯一主站，并且不能与其它主站共享网络。 当通过调制解调器或躁声严重的网络通讯时，应该采用单主站协议。在 PG/PC 接口的 PC/PPI 电缆属性对话框中，通过清除“Multiple Master Network”选项可以选择单主站方式。

按照下面的步骤设置 PPI 参数：

1. 在 PPI 标示签的站参数区的地址框中，选择一个号。这个号标明在可编程控制器网络中 STEP 7-Micro/WIN 32 位于何处。站号 0 是运行 STEP 7-Micro/WIN 32 的个人计算机的缺省站地址。在网络中，第一个 PLC 的缺省地址是站号 2 。网络中的每个设备（PC、PLC 等）必须具有唯一的站地址，不要给几个设备分配同一个地址。
2. 在超时框中选择一个值。这个值代表使通讯处理器建立连接需要花费的时间长 度。缺省值应该足够长。
3. 决定是否需要 STEP 7-Micro/WIN 32 加入一个多主站网络。应该选中在多主网络（ Multiple Master Network ）检查盒，除非使用调制解调器或 Windows NT 4.0 。在那种情况下，由于 STEP 7-Micro/WIN 32 不支持这个功能，应该不选中多主网络（Multiple Master Network ）检查盒。
4. 设定 STEP 7-Micro/WIN 32 在网络中进行通讯的传输速率。 PPI 电缆支持9.6 k 波特和 19.2 k 波特。
5. 选择最高的站地址。这是 STEP 7-Micro/WIN 32 停止检查 PPI 网络中其它主站的地址。

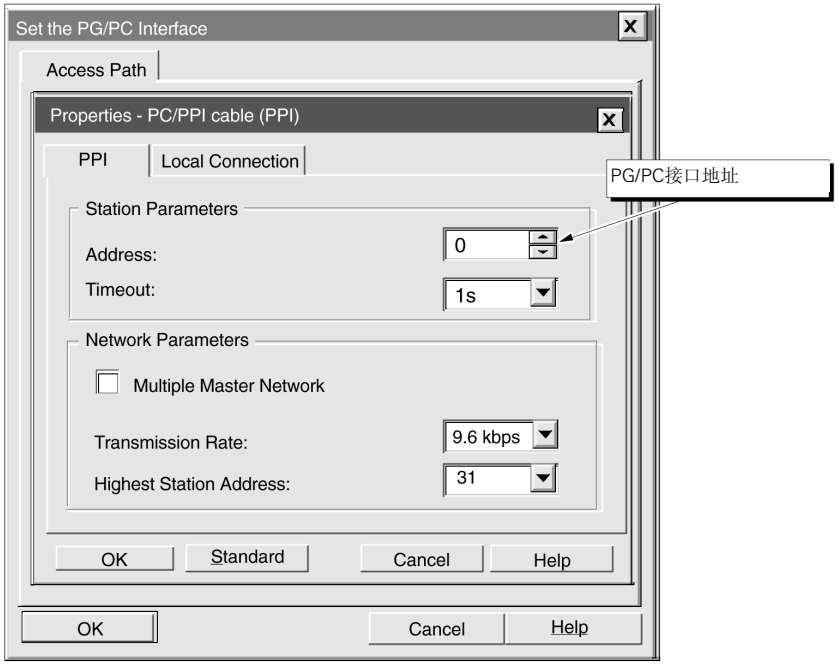


图 7-7　　属性 --PC/PPI 电缆对话，PPI 标签

6. 单击本机连接标志签，见图 7–8 。
7. 在本机连接标志签中，选择 PC/PPI 所连接的通讯口。如果使用调制解调器，选择调制解调器所连接的通讯口，并用调制解调器检查盒选择使用调制解调器。
8. 单击“OK”按钮，退出设定 PG/PC 接口 (Setting the PG/PC Interface) 对话。

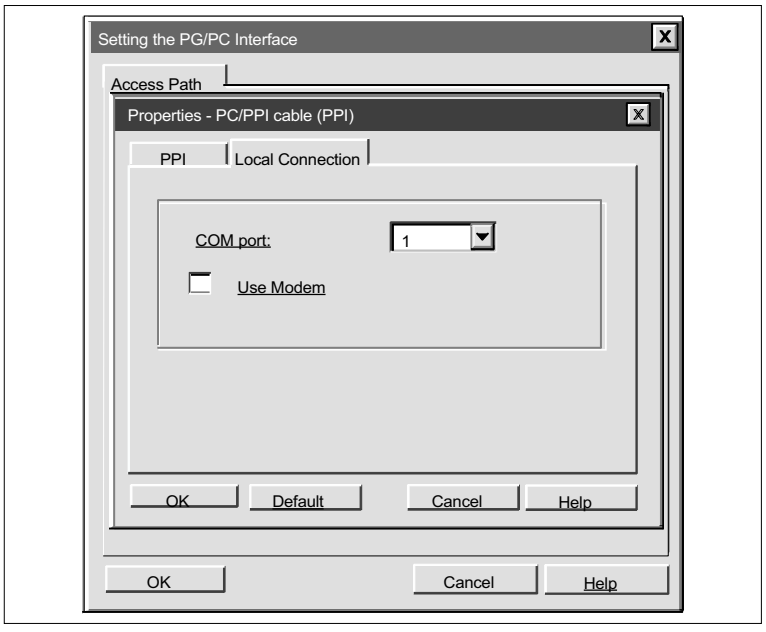


图 7-8　　属性 --PC/PPI 电缆对话，本机连接标志签

用 MPI 或 CP 卡组态：多主网络

当使用多主接口卡或通讯处理器卡时，可以有多种组态。用 MPI 电缆可以把卡提供的单一 RS-485 接口连接到网络。在包含多个主站的网络中，可以选择一个站运行 STEP 7-Micro/WIN 32 编程软件。(带 MPI 或 CP 卡的计算机，或SIMATIC 编程器) (如果允许多主站， PC/PPI 电缆也如此)。 这些主站包括操作员面板和文本显示器 (TD 200)。图 7–9 是一个网络中带有两个 TD 200 的组态。

注意：

如果使用 PPI 参数设置，STEP 7-Micro/WIN 32 不支持两个同时运行在相同MPI 或 CP 卡的不同应用。在通过 MPI 或 CP 卡把 STEP 7-Micro/WIN 32 连接到网络前要关闭另一个应用。

在这个组态中，下面给出了可能的通讯：

- STEP 7-Micro/WIN 32 (在 0 号站) 可以监视 2 号站的状态，同时 TD 200 (5 号和 1 号站) 和 CPU 224 模块 (3 号站和 4 号站) 通讯 。
- 通过网络指令 (NETR 和 NETW) 两个 CPU224 模块可以发送信息。
- 3 号站可以从 2 号站 (CPU 222) 和 4 号站 (CPU 224) 读写数据。
- 4 号站可以从 2 号站 (CPU 222) 和 3 号站 (CPU 224) 读写数据。

可以把多个主站和从站连到同一个网络。但是，当加入多个站时网络的性能会受到不利的影响。

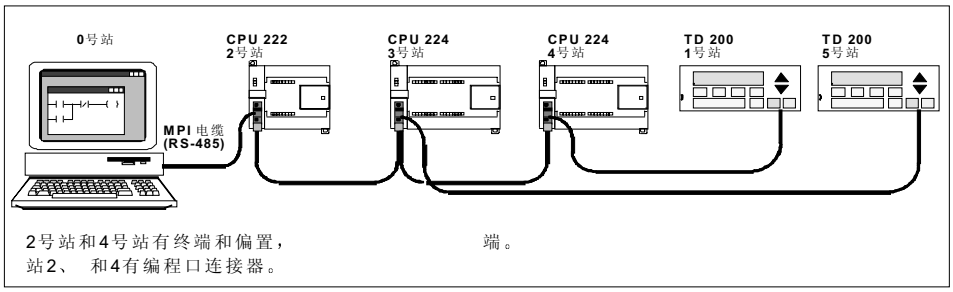


图 7-9 利用 MPI 或 CP 卡和 S7-200 CPU 通讯

设置 CP 或 MPI 卡 (PPI) 参数

本节说明如何为 Windows 95、Windows 98 或 Windows NT 4.0 操作系统设置 PPI 参数和下面的硬件：

- CP 5511
- CP 5611
- MPI

在设置 PG/PC 接口 (Setting the PG/PC Interface) 对话框中，如果使用上面所列的 MPI 或 CP 卡和 PPI 协议，单击“Properties...”按钮，将出现 XXX 卡 (PPI) 的属 性，这里“XXX”代表所安装的卡型号。例如， MPI–ISA ，请参阅图 7-10。

注意：

当与 S7-200 CPU215 的口 1 通讯时，使用 MPI 协议。有关 CPU 215 和 MPI 协议的详细内容，请参阅以前的 S7-200 可编程控制器系统手册 (定货号：6ES7-298-8FA01-8BH0)

按照下面的步骤设置 PPI 参数：

1. 在 PPI 标识签的地址框中选择一个号。这个号标明在可编程控制器网络中 STEP 7-Micro/WIN 32 位于何处。
2. 在超时框中选择一个值。这个值代表使通讯处理器建立连接需要花费的时间长 度。缺省值应该足够长。
3. 设定 STEP 7-Micro/WIN 32 在网络中进行通讯的传输速率。
4. 选择最高的站地址。这是 STEP 7-Micro/WIN 32 停止检查网络中其它主站的地址。
5. 单击“OK”按钮，退出设定 PG/PC 接口 (Setting the PG/PC Interface) 对话。

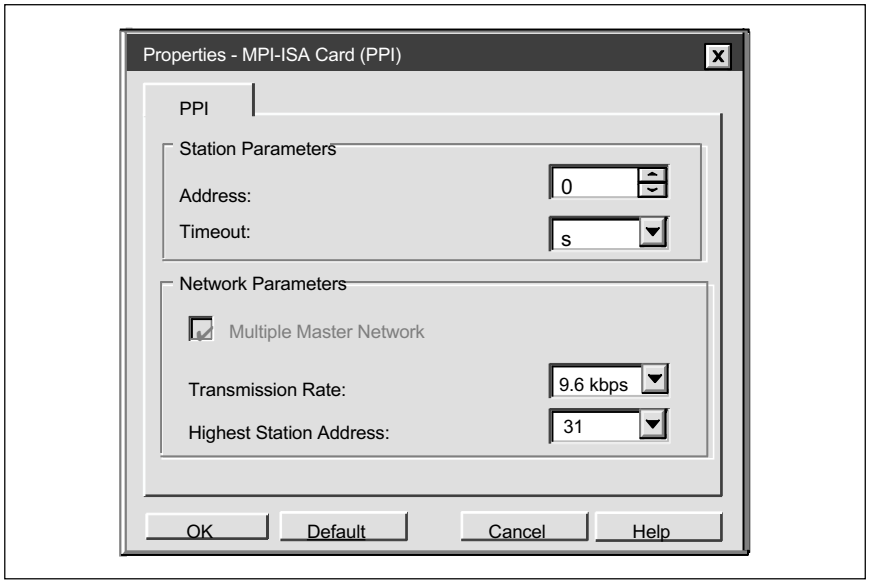


图 7-10 MPI-ISA 卡 (PPI) 属性窗口

7.4 利用调制解调器通讯

采用调制解调器时设置通讯参数

采用调制解调器设置编程器或计算机与 CPU 之间的通讯参数时，必须使用PC/PPI 电缆的模块参数集。否则，不能使用组态调制解调器功能。确保允许组态调制解调器(Configure Modems) 功能，然后用下面的步骤设定组态参数：

注意：

STEP 7-Micro/WIN 32 显示调制解调器设置对话框中的预定义调制解调器。这些调制解调器型号已经测试过而且确保工作。

设置本地调制解调器：

1. 选择菜单命令 "View>Communications" (或在通讯图标上单击)。

   双击通讯设置对话框中的 PC/PPI 电缆图标，出现设置 PG/PC 接口 (Setting the PG/PC Interface) 对话框，进入第三步。如果通讯设置对话框中没有PC/PPI 电缆图标，双击 PC 卡图标或右边的上方图标。

2. 在设置 PG/PC 接口 (Setting the PG/PC Interface) 对话框中，选择 PC/PPI 电缆(PPI)。如果列表框中没有该选择，必须进行安装。见 7.2 节。

3. 单击 “Properties” 按钮，出现 CPU 和调制解调器的 PC/PPI 电缆 (PPI) 属性窗口。见图 7–8。

4. 在PC/PPI 电缆 (PPI) 属性窗口中，单击本机连接标识签。

5. 在通讯口 (COM) 区，确保选择 “Use Modem” 盒。如果没有选择，请选择该功能。见图 7–8。

6. 单击 “OK” 按钮，出现设置 PG/PC 接口 (Setting the PG/PC Interface) 对话框。

7. 单击 “OK” 按纽，出现通讯设置 Communications Setup) 对话窗口，现在有两个调制解调器图标和一个连接调制解器图标 (见图 7-11)。

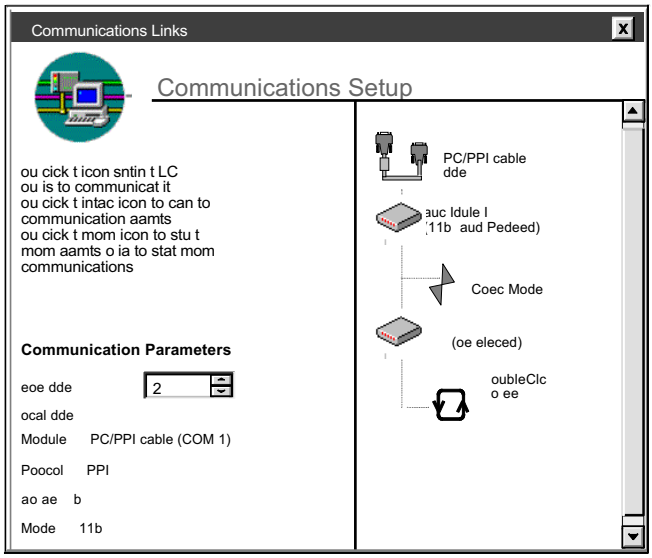


图 7-11 通讯设置对话窗口

8. 双击通讯设置对话框中的第一个调制解调器图标，出现本机调制解调器设置 (Modem Setup) 对话框 (图 7–12)。
9. 在本机调制解调器区，选择所用的调制解调器型号。如果没有列出你的调制解 调器，选择 Add 按钮配置你的调制解调器。要这样做你必须知道你的调制解调器的 AT 命令。请参考你的调制解调器说明书。
10. 在通讯方式区，选择通讯方式 (10– 位或 11– 位)。选择的通讯方式有赖于你的调制解调器的能力。(10– 位和 11– 位通讯方式在本节后面描述)。本地和远程调制解调器必须是相同的通讯方式。单击“Configure”按钮。

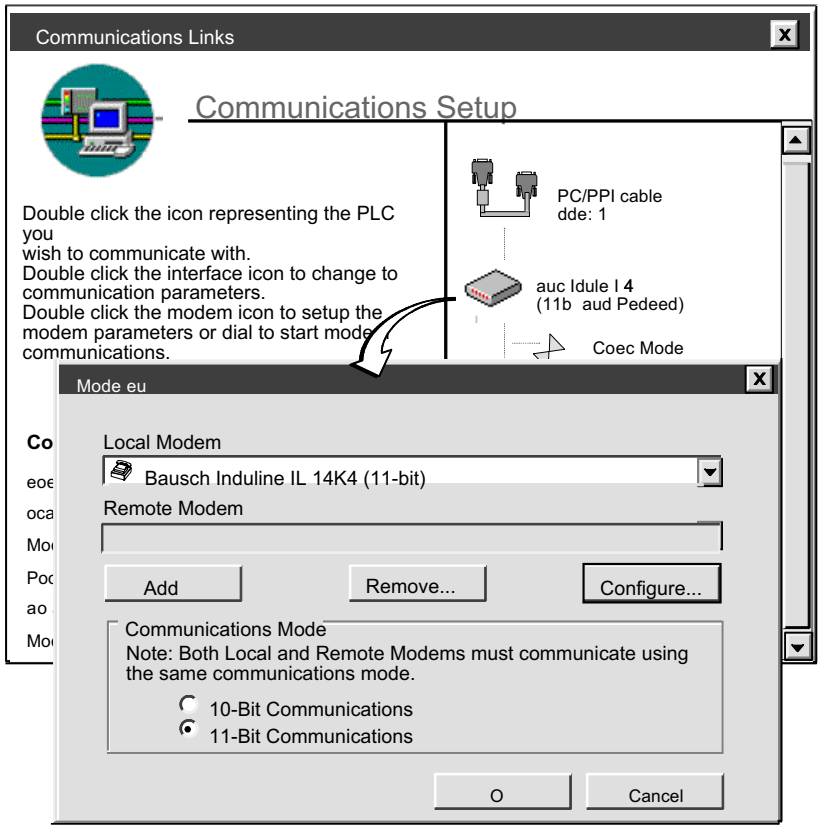


图 7-12　　本地调制解调器的设置对话窗口

11. 出现配置 (Configure) 对话窗口 (图 7–13)。如果正使用预定义的调制解调器， 在该对话窗口中唯一可以修改的是超时框。超时是本地和远程调制解调器建立连接的时间长度。如果连接建立前，用秒表示的超时区的时间已经用完，就会出现连接失败。如果不是使用预定义的调制解调器，必须输入 AT 命令串启动你的调制解调器。请参考你的调制解调器说明书。
12. 如果要测试本地调制解调器的配置，当调制解调器连到本地计算机 (编程器或 PC) 时，单击“Program/Test”按钮。这样可以把调制解调器配置成当前的协议和设置，并验证调制解调器接受配置的设定。单击‘‘OK’’按钮，返回通讯设置对话窗口。
13. 断开本地调制解调器并把远程调制解调器连到本地计算机 (编程器或个人计算机)。

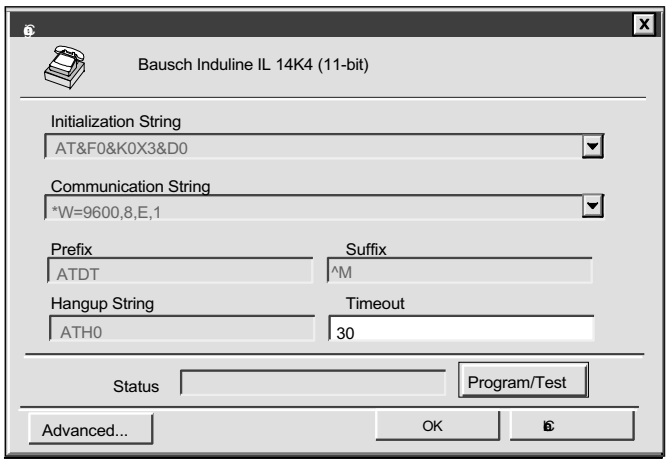


图 7-13　本地调制解调器配置

设置远程调制解调器：

1. 在通讯设置对话窗口中，双击第二个调制解调器图标 (图 7–11)，出现远程调制解调器的“Modem Setup” 对话窗口 (图 7-14)。
2. 在远程调制解调器区，选择调制解调器型号。如果你的调制解调器没有列出， 选择“Add” 按钮配置你的调制解调器。要这样做你必须知道你的调制解调器的 AT 命令。请参考你的调制解调器说明书。
3. 在通讯方式区，选择通讯方式 (10– 位或 11– 位)。选择的通讯方式有赖于你的调制解调器的能力。(10– 位和 11– 位通讯方式在本节后面描述)。 本地和远程调制解调器必须是相同的通讯方式。单击“Configure” 按钮。
4. 出现配置 (Configure) 对话窗口 (图 7–15)。如果正使用预定义的调制解调器， 在该对话窗口中没有可以修改的区域。如果不是使用预定义的调制解调器，必须输入 AT 命令串启动你的调制解调器。请参考你的调制解调器说明书。
5. 如果要测试远程调制解调器，当调制解调器连到本地计算机 (编程器或 PC) 时， 单击“Program/Test” 按钮。这样可以把参数传到远程调制解调器的存储器芯片。
6. 单击“OK”按钮，出现通讯设置对话窗口。

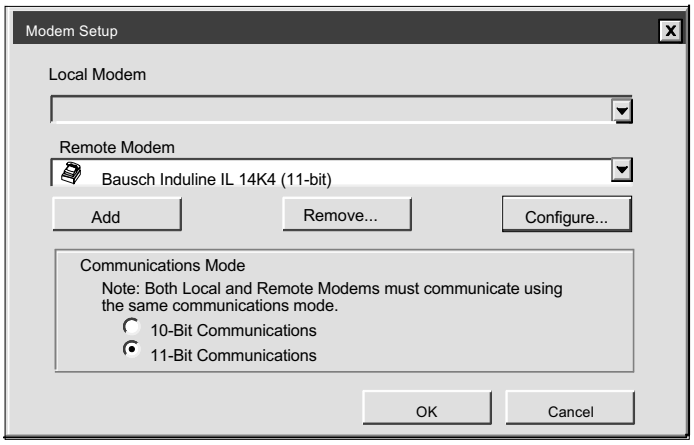


图 7-14 远程调制解调器的设置对话窗口

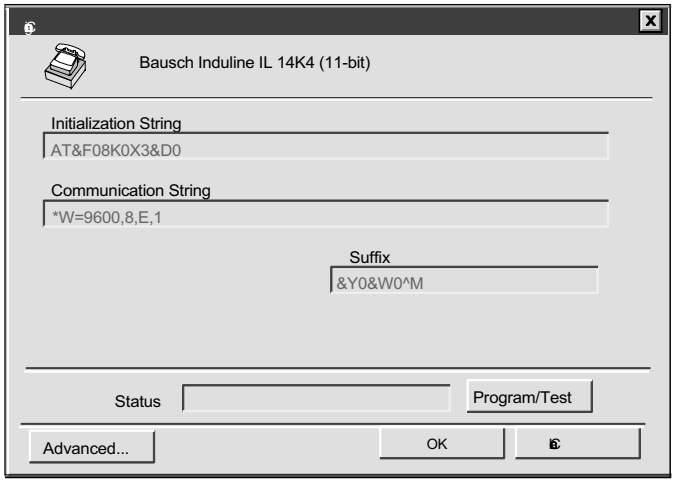


图 7-15　　远程调制解调器配置

7. 断开远程调制解调器与本地计算机 (编程器或 PC) 的连接。
8. 把远程调制解调器连到 S7–200 可编程控制器上。
9. 把本地调制解调器连到编程器或 PC 上。

连接调制解调器：

1. 双击 Communications Setup 对话窗口中的 Connect Modem 图标连接调制解调器，出现拨号窗口，见图 7-16。
2. 在 Dial 对话窗口中的 Phone Number 区输入电话号。
3. 单击 “Connect”按钮把本地和远程调制解调器连接起来。

调制解调器设置完成。

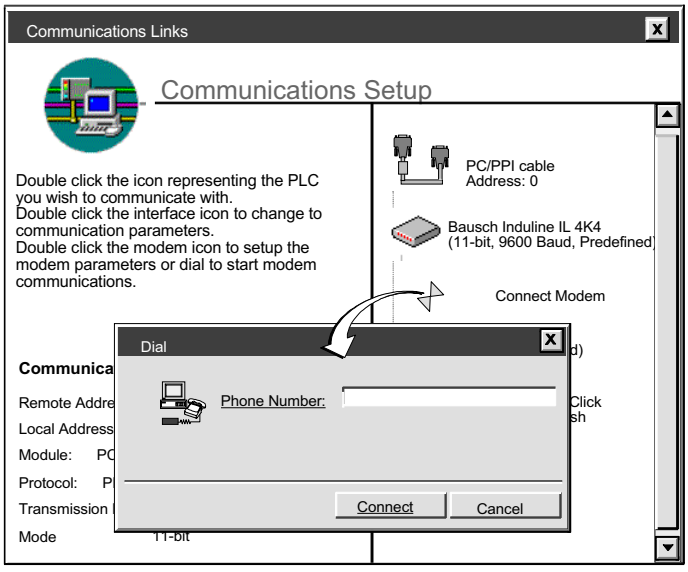


图 7-16　连接调制解调器

采用 10- 位调制解调器把 S7–200 CPU 连到 STEP 7-Micro/WIN 32 主站

用装有 Windows 95、Windows 98或 Windows NT 操作系统的计算机或SIMATIC 编程器 (例如 PG 740) 上的 STEP 7-Micro/WIN 32 作为一个单主站，仅可以连接一个 S7-200 CPU。采用贺氏 (Hayes) 兼容的 10 位调制解调器可以与一台远程 S7–200 CPU 通讯。图 7-17 所示用带有 5-开关 PC/PPI 电缆的 10 位调制解调器进行数据通讯。

需要如下的设备：

- 作为从站的单个 S7–200 CPU。CPU 221、CPU 222、CPU 224 和 CPU 226 支持10位格式。以前的 S7–200 CPU 不支持 10位格式。
- 一个把PC或SIMATIC编程器连接到全双工 10 位本地调制解调器的 RS–232 电缆。
- 一个 5– 开关 PC/PPI 电缆 (用来设定合适的波特率、10 位数据通讯方式和DTE 方式) 把远程调制解调器连到 CPU。
- 一个可选的 9– 针到 25– 针适配器 (如果你的调制解调器连接器需要)。

注意：

4–开关 PC/PPI 电缆不支持 10 位格式。

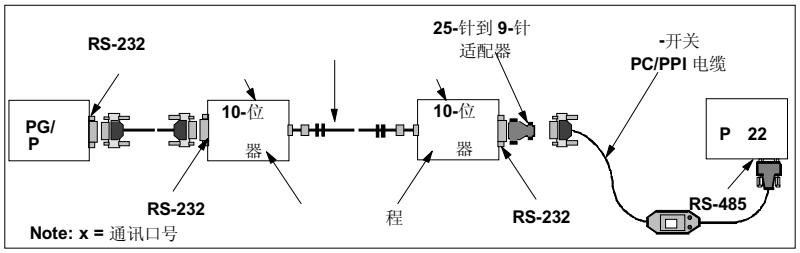


图 7-17 用 5- 开关 PC/PPI 电缆和 10 位调制解调器的 S7-200 数据通讯

这个配置只允许一个主站和一个从站，在配置中，S7–200 控制器要求 1 个起始位、8 个数据位、无校验位、一个停止位、异步通讯并且传输速度为 9600 波特。调制解调器要求表 7-3 中的设置。图 7-18 给出了 25 针到 9 针适配器的引脚分配。

表 7-3 需要 10– 位调制解调器的设置

调制解调器	用位表示的数据格式	在调制解调器和计算机间的传输速率	线上的传输速度	它特性
10位	8 数据	9600 波特，19200波特	9600 波特，19200波特	忽略 DTR 信号
	1 起始			无硬件流控制
	1 停止			无软件流控制
	无校验			

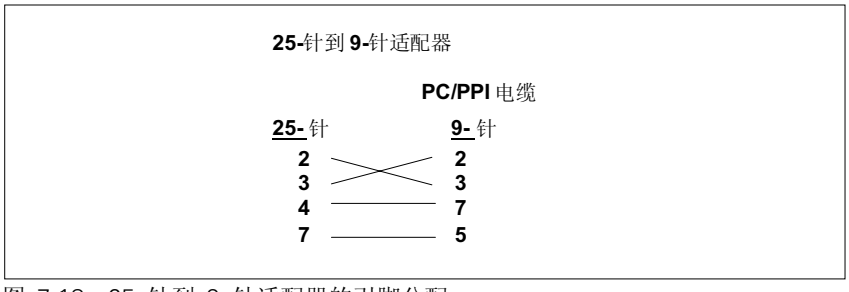


图 7-18 25 针到 9 针适配器的引脚分配

采用 11- 位调制解调器把 S7–200 CPU 连到 STEP 7-Micro/WIN 32 主站

在安装 Windows 95、Windows 98 或Windows NT 操作系统的计算机或SIMATIC 编程器上以主站的形式使用 STEP 7-Micro/WIN 32，可以连接到一个或多个 S7-200 CPU。大部分的调制解调器不支持 11位协议。

根据要连到单个的 S7-200 CPU 或它们的网络 (见图 7-19)，需要如下的硬件：

- 一个把PC或SIMATIC编程器连接到全双工11位本地调制解调器的标准RS–232电缆。
- 一个如下的 PC/PPI 电缆：
  - 把远程调制解调器连接到 CPU 的 5- 针 PC/PPI 电缆 (设置合适的波特率、11位数据通讯模式和数据终端设备 DTE 模式)。
  - 4针 PC/PPI 电缆 (设置合适的波特率) 和一个把远程调制解调器连接到 CPU 的调制解调器适配器。
- 如果远程调制解调器上连有多个 CPU，就需要在 PROFIBUS 网络上设置一个西门子编程口连接器 (见图 7-24 的内部连接电缆的偏置和终端匹配)。

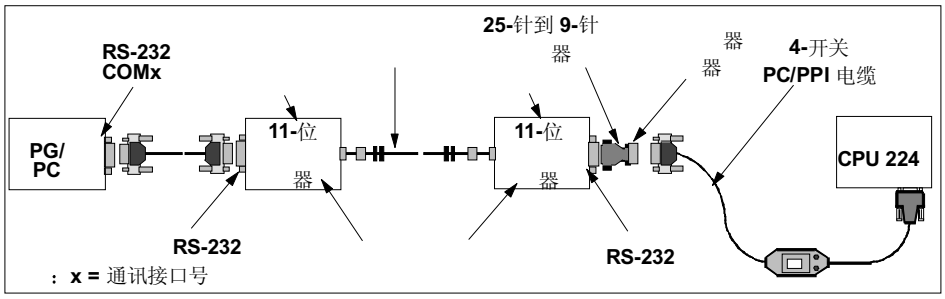


图 7-19 用 4-开关 PC/PPI 电缆采用 11-位调制解调器的 S7-200 数据通讯

这个组态只允许一个主站，而且只支持 PPI 协议。为了通过 PPI 接口通讯，S7–200 PLC 要求调制解调器采用 11 位数据串。在这个模式下， S7–200 控制器要求 1 个起始位、8 个数据位、1 个校验位 (偶校验) 、1 个停止位和异步通讯方式，通讯的速率为 9600/19200 波特。许多调制解调器不支持这个数据格式，调制解调器的设置要求在表 7-4 中列出。

图 7-20 给出了调制解调器和 25 针到 9 针适配器的引针分配。

表 7-4 11 位调制解调器要求的设置

调制解调器	位表示的数据格式	调制解调器与计算机之间的传输速率	线上的传输速率	它特性
11-位	8 位数据	9600 波特，19200 波特	9600 波特，19200 波特	忽略 DTR 信号
	1 位起始			无硬件流控制
	1 位结束			无软件流控制
	1 位校验 (偶)			

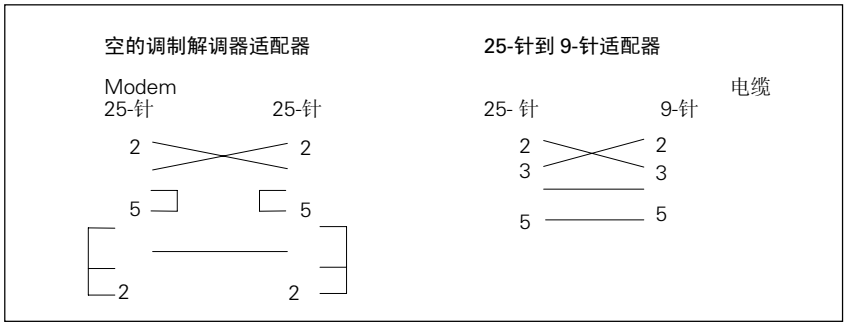


图 7-20　空的调制解调器和 25 针到 9 针适配器的引针分配

7.5　网络概述

网络主站

图 7-21 是一台个人计算机连接几个 S7–200 CPU的组态。STEP 7-Micro/WIN 32 设计一次可以和一个 S7–200 CPU 通讯，但是可以访问网络中的任何 CPU。图 7-21 中的 CPU 可以是从站或主站。 TD 200 是主站。

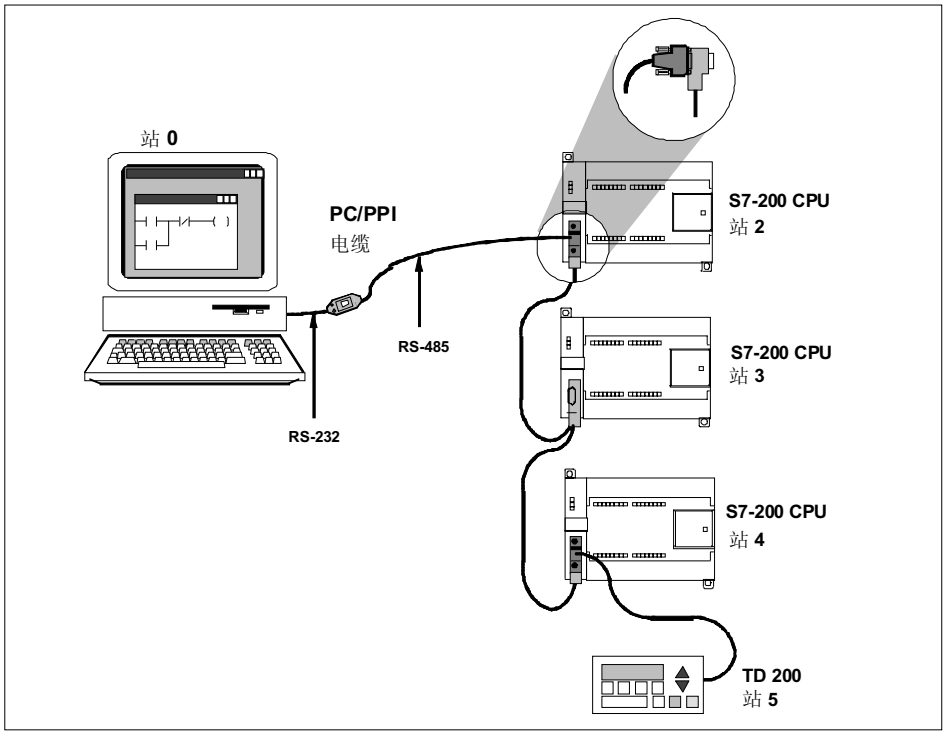


图 7-21　　利用 PC/PPI 电缆和几个 S7-200 CPU 主站通讯

图 7-22 所示为多主站设备构成的通用网络。使用 EM 277 PROFIBUS-DP 模块可以提高通讯速率和连接数量。

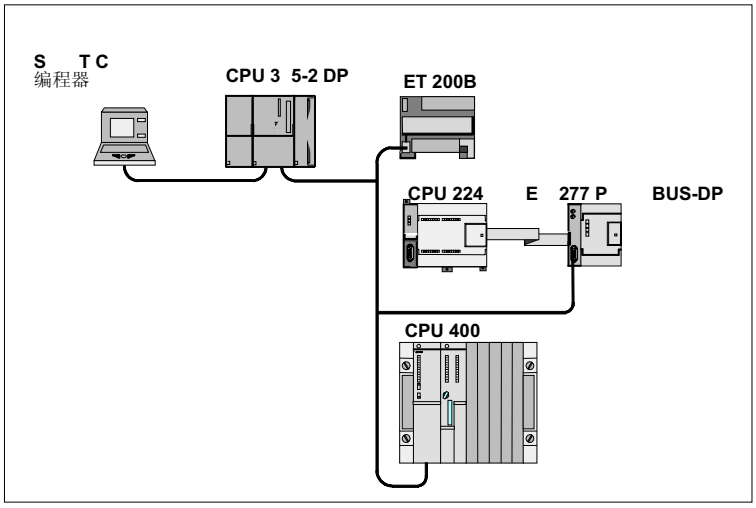


图 7-22 在 PROFIBUS-DP 网络中 EM 277 PROFIBUS-DP 模块和 CPU 224

网络通讯协议

S7–200 CPU 支持多样的通讯能力。根据所使用 S7–200 CPU，你的网络可以支持一个或多个以下协议：

- 点到点 (Point-to-Point) 接口 (PPI)
- 多点接口 (Multi-Point) (MPI)
- PROFIBUS

这些基于开放系统内连接 (OSI) 7 层通讯结构模型。PPI 和 MPI 协议通过令牌环网实现，令牌环网遵守欧洲标准 EN 50170 中的过程现场总线 (PROFIBUS)。

这些都是异步、基于字符的协议，带有起始位、8 位数据、偶校验和 1 个停止位。 通讯帧由特殊的起始和结束字符、源和目的站地址、帧长度和数据完整性检查和组成。只要相互的波特率相同，三个协议可以在一个网络中同时运行，而不会相互影响。

PROFIBUS 网络使用 RS–485 标准双绞线。它允许在一个网络段上最多连接 32 台设备。根据波特率不同，网络段的长度可以达到 1,200 米 (3,936 英尺)。采用中继器连接网络段可以在网络上连接更多的设备，延长网络的长度。 根据不同的波特率，采用中继器可以把网络延长到 9,600 米 (31,488 英尺)，见表 7-7。

协议定义了两类网络设备：主站和从站。主站可以对网络上另一个设备初始化申请。从站只响应来自主站的申请。从站不初始化本身的申请。

协议支持一个网络上的 127 个地址 (从 0 到 126)，网络上最多由 32 个主站。为了通讯，网络上的所有设备必须具有不同的地址。运行 STEP 7-Micro/WIN 32 的 SIMATIC 编程器和计算机的缺省地址是 0。操作面板 (如TD200、OP3和OP7) 的缺省地址是 1。可编程控制器的缺省地址是 2。

PPI 协议

PPI 是一个主/从协议。在这个协议中，主站 (其它 CPU、SIMATIC 编程器或 TD 200) 给从站发送申请，从站进行响应。从站不初始化信息，但是当主站发出申请或查询时，从站才响应。网络上的所有 S7–200 CPU 都作为从站。

如果在用户程序中允许 PPI 主站模式，一些 S7–200 CPU 在 RUN 模式下可以作为主站。(见附录 C 中SMB30 的描述)。一旦允许 PPI 主站模式，就可以利用网络读 (NETR) 和网络写 (NETW) 指令读写其它 CPU。有关这些指令的详细描述， 请参阅第九章的9.15节通讯指令。 当S7-200 CPU作为 PPI 主站时，它还可以作为从站响应来自其它主站的申请。

对于任何一个从站有多少个主站和它通讯，PPI 没有限制，但是在网络中最多只能有32个主站。

MPI 协议

MPI 可以是主/主协议或主/从协议。协议如何操作有赖于设备类型。如果设备是S7–300 CPU，那么就建立主/主，因为所有的 S7–300 CPU 都是网络主站。如果设备是 S7–200 CPU，那么就建立主/从连接，因为 S7–200 CPU 是从站。

MPI 总是在两个相互通讯的设备之间建立连接。一个连接可能是两个设备之间的非公用连接。另一个主站不能干涉两个设备之间已经建立的连接。主站为了应用可以短时间建立一个连接，或无限地保持连接断开。

由于设备之间S7-200的连接是非公用的，并且需要 CPU 中的资源，每个S7-200 CPU 只能支持一定数目的连接，每个 CPU 支持四个连接，每个EM 277模块支持6个连接。每个S7-200 CPU和EM 277模块保留两个连接，其中一个给 SIMATIC 编程器或计算机，另一个给操作面板。所保留的连接可用于连接至少一台编程器或PC机以及至少一个操作面板。这些保留的连接不能由其它类型的主站 (如 CPU) 使用。

通过与 S7-200 CPU 建立一个非保留的连接，S7-300 和 S7-400 CPU 可以和S7-200 CPU或EM 277模块 进行通讯。利用 XGET 和 XPUT 指令，S7-300 和 S7-400 可以读写 S7-200。(请参考 S7-300 或 S7-400 编程手册)

PROFIBUS 协议

PROFIBUS 协议设计用于分布式 I/O 设备 (远程 I/O) 的高速通讯。许多厂家生产类型众多的 PROFIBUS 设备。这些设备包括从简单的输入或输出模块到电机控制器和可编程控制器。

PROFIBUS 网络通常有一个主站和几个 I/O 从站。主站配置成知道所连接的 I/O 从站的型号和地址。主站初始化网络并核对网络上的从站设备和配置中的是否匹配。主站连续地把输出数据写到从站并从它们读取输入数据。当 DP 主站成功地组态一个从站时，它就拥有该从站。如果网络中有第二个主站，它只能很有限地访问第一个主站的从站。

有关 EM 277 PROFIBUS-DP 模块的信息及使用方法，请参见附录 A 中的产品规范。

用户定义协议 (自由口)

自由口通讯是通过用户程序可以控制 S7-200 CPU 通讯口的操作模式。利用自由口模式，可以实现用户定义的通讯协议连接多种智能设备。

通过使用接收中断、发送中断、发送指令 (XMT) 和接收指令 (RCV)，用户程序控制通讯口的操作。在自由口模式下，通讯协议完全由用户程序控制。通过SMB30 (口0) 允许自由口模式，而且只有在 CPU 处于 RUN 模式时才能允许。当 CPU 处于 STOP 模式时，自由口通讯停止，通讯口转换成正常的 PPI 协议操作。有关发送和接收指令的描述请参阅第 9 章的 9.15 节通讯指令。

7.6 网络部件

可以把每个 S7–200 上的通讯口连到网络总线。下面描述通讯口、网络总线连接器、网络电缆和用于扩展网络的中继器。

通讯口

S7–200 CPU 上的通讯口是符合欧洲标准 EN 50170 中 PROFIBUS 标准的 RS–485 兼容 9-针 D 型连接器。图 7-23 是通讯接口的物理连接口，表 7-5 给出了通讯口插针的分配。

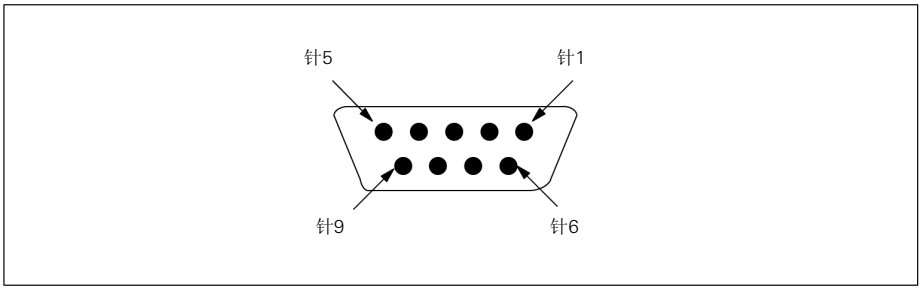


图 7-23 S7-200 CPU 通讯口引脚分配

表 7-5 S7–200 通讯口引脚分配

针	PROFIBUS 名称	端口 0 / 端口 1
1	屏蔽	逻辑地
2	24 V 返回	逻辑地
3	RS–485 信号 B	RS–485 信号 B
4	发送申请	RTS (TTL)
5	5 V 返回	逻辑地
6	+5 V	+5 V, 100 Ω 串联电阻
7	+24 V	+24 V
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

网络连接器

利用西门子提供的两种网络连接器可以把多个设备很容易地连到网络中。两种连接器都有两组螺丝端子，可以连接网络的输入和输出。两种网络连接器还有网络偏置和终端匹配的选择开关。一个连接器仅提供连接到 CPU 的接口，而另一个连接器增加了一个编程接口(见图 7-24)。有关定货信息请参阅附录E。

带有编程接口的连接器可以把 SIMATIC 编程器或操作面板增加到网络中，而不用改动现有的网络连接。编程口连接器把 CPU 来的信号传到编程口。这个连接器对于连接从 CPU 取电源的设备 (例如 TD 200 或 OP3) 很有用。编程口连接器上的电源引针连到编程口。

警告:

连接具有不同参考电位的设备会在连接电缆中产生不必要的电流。这些不必要的电流会造成通讯故障或损坏设备。

确保需要通讯电缆连接的所有设备或者共享一个共同的参考点，或者进行隔离以防止不必要的电流。请参阅 2.3“隔离电器的接地和电器的接地和电路参考点”。

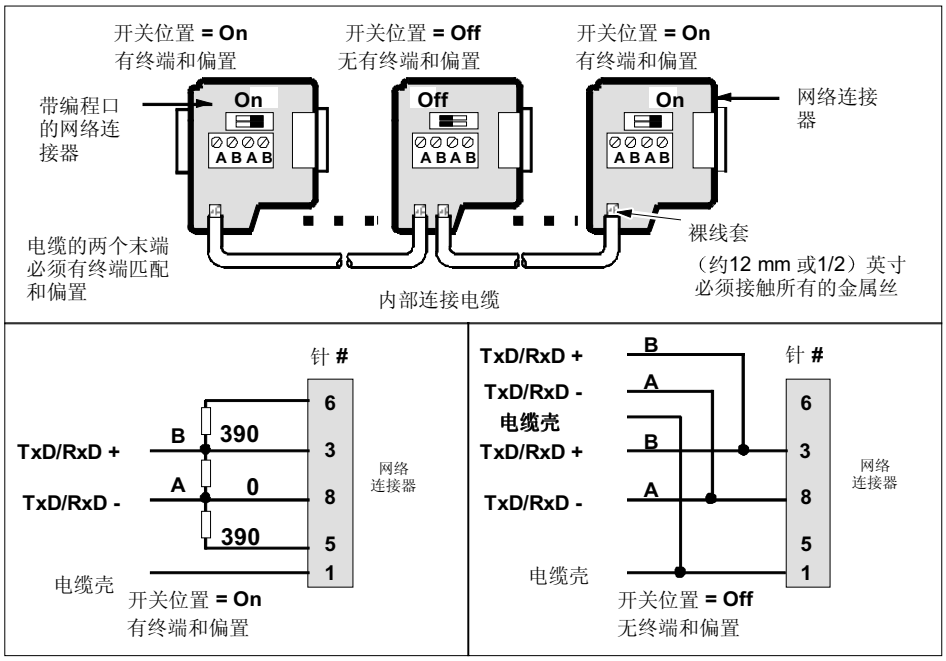


图 7-24　　内部连接电缆的偏置和终端

PROFIBUS 网络电缆

表 7-6 列出了 PROFIBUS 网络电缆的总规范。有关满足这些要求的 PROFIBUS 电缆的西门子定货号请参阅附录E。

表 7-6　　PROFIBUS 网络电缆的总的规范

通用特性	规　　范
类型	屏蔽双绞线
导体截面积	24 AWG (0.22 mm^2)或更粗
电缆电容	< 60 pF/m
阻抗	100 Ω　～120　Ω

PROFIBUS 网络的最大长度有赖于波特率和所用电缆的类型。表 7-7 列出了采用满足表 7-6 中列出规范的电缆时网络段的最大长度。

表 7-7　　PROFIBUS 网络中段的最大电缆长度

传输速率	网络　的最大电缆长度
9.6 k 波特 ～93.75 k 波特	1,200 米 (3,936 英尺)
187.5 k 波特	1,000 米 (3,280 英尺)
500 k 波特	400 米 (1,312 英尺)
1M 波特 ～1.5M 波特	200 米 (656 英尺)
3M 波特 ～12M 波特	100 米 (328 英尺)

网络中继器

西门子提供连接到 PROFIBUS 网络段的网络中继器，见图 7-25。利用中继器可以延长网络距离；允许给网络加入设备；并且提供了一个隔离不同网络段的方法。在波特率是 9,600 时，PROFIBUS 允许在一个网络环上最多有 32 个设备，最长距离是 1,200 m (3,936 英尺)，每个中继器允许给网络增加另外 32 个设备，而且可以把网络再延长 1,200 m (3,936 英尺)。网络中最多可以使用 9 个中继器,网络总长度可增加至9600米。每个中继器为网络段提供偏置和终端匹配。有关定货信息请参阅附录 E。

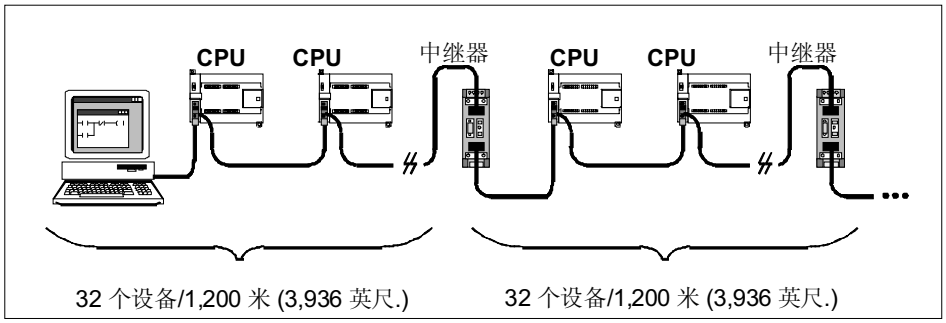


图 7-25　带有中继器的网络

7.7 用其它设备和自由口使用 PC/PPI 电缆

可以利用 PC/PPI 电缆和自由口通讯功能把 S7–200 CPU 连接到许多和 RS-232标准兼容的设备。

有两种不同型号的 PC/PPI 电缆：

- 带有 RS-232 口的隔离型 PC/PPI 电缆，用 5 个 DIP 开关设置波特率和其它配置项(见图 7–27)。有关隔离型 PC/PPI 电缆的技术规范，请参阅附录 A。
- 带有 RS-232 口的非隔离型 PC/PPI 电缆，用 4 个 DIP 开关设置波特率。 有关非隔离型 PC/PPI 电缆的技术规范，请参阅以前的 *S7–200 可编程控制器系统手册* (定货号：6ES7-298-8FA01-8BH0)。

两个 PC/PPI 电缆支持波特率设置：600 波特 ~ 38,400 波特。利用 PC/PPI 电缆盒上的 DIP 开关配置正确的波特率。表 7-8 是波特率和开关位置的列表。

表 7-8 PC/PPI 电缆上的波特率开关

波 特 率	开 关 (1 = 上)
38400	000
19200	001
9600	010
4800	011
2400	100
1200	101
600	110

当数据从 RS-232 传送到 RS-485 口时，PC/PPI 电缆是发送模式。当数据从RS-485 传送到 RS-232 口时，PC/PPI 电缆是接收模式。当检测到 RS-232 的发送线有字符时，电缆立即从接收模式转换到发送模式。当 RS-232 发送线处于闲置的时间超过电缆切换时间时，电缆又切换到接收模式。这个时间与电缆上的 DIP 开关设定的波特率选择有关，(见表 7-9)。

如果在已经使用自由口的系统中使用 PC/PPI 电缆，对于下面的情况，必须在S7–200 CPU 的用户程序中包含转换时间：

- S7–200 CPU 对 RS-232 设备发送的信息的响应。
  在接收到 RS-232 设备的申请信息后，S7-200 CPU 的发送信息响应必须延迟超过或等于电缆的切换时间。
- RS–232 设备对 S7-200 CPU 发送的信息的响应。
  在接收到 RS-232 设备的申请信息后，S7-200 CPU 的下一次申请信息的发出必须延迟超过或等于电缆的切换时间。

在两种情况，延迟使 PC/PPI 电缆有足够的时间从发送模式切换到接收模式，以便于数据从 RS-485 口传到 RS-232 口。

表 7-9　　PC/PPI 电缆转换时间 (发送模式到接收模式)

波特率	转换时间 (ms)
38400	0.5
19200	1
9600	2
4800	4
2400	7
1200	14
600	28

通过 5-开关 PC/PPI 电缆使用调制解调器

可以用 5–开关 PC/PPI 电缆把 S7 200 CPU 连到调制解调器的 RS–232 通讯口。 允许计算机使用 RS–232 控制信号 (例如 RTS、CTS和 DTR) 控制调制解调器。 这个 PC/PPI 电缆不监视这些信号，但是在数据终端设备 (DTE) 模式下提供 RTS 信号。 所以当利用 5- 开关 PC/PPI 电缆使用调制解调器时，调制解调器必须配置成不带这些信号的操作。最小要求也是必须把调制解调器配置成忽略 DTR。参照调制解调器的操作手册，决定需要组态调制解调器的命令。

5– 开关 PC/PPI 电缆的 RS-232 口可以设置为数据通讯设备 (DCE) 模式或数据终端设备 (DTE) 模式。这个接口上提供的信号有发送数据、申请发送、接收数据和地。5– 开关 PC/PPI 电缆不使用或不提供清除发送信号 (CTS)。表 7–9 和表 7–10 是 PC/PPI 电缆的引针。

调制解调器是数据通讯设备 (DCE) 类。当把 PC/PPI 电缆连到调制解调器时，应该通过 DPI 开关 5 把 PC/PPI 电缆的 RS232 口设置成数据终端设备 (DTE)。这样可以在 PC/PPI 电缆与调制解调器之间省去一个调制解调器适配器。根据所连接的调制解调器，你可能还需要一个 9-针到 25-针的适配器。图7-26 是 9-针到 25-针适配器的典型设置和引针分配。

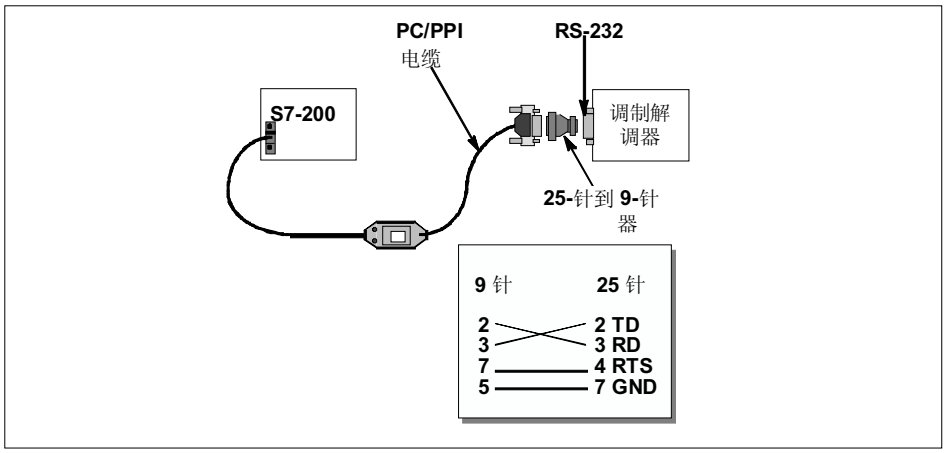


图 7-26　　带有调制解调器的 5- 开关 PC/PPI 电缆引脚分配

为了设置数据通讯设备 (DCE) 模式，应该把开关 5 设置为 0 或向下位置 (见图 7-27)。为了设置数据终端设备 (DTE) 模式，应该把开关5设置为 1 或向上位置。表 7-10 是 DTE 模式下 PC/PPI 电缆的引针号和 RS–485 到 RS–232口 的功能。表 7-11 DCE 模式下 PC/PPI 电缆的引针号和 RS–485 到 RS–232口 的功能。应该注意只有在 DTE 模式下 PC/PPI 电缆才提供 RTS。

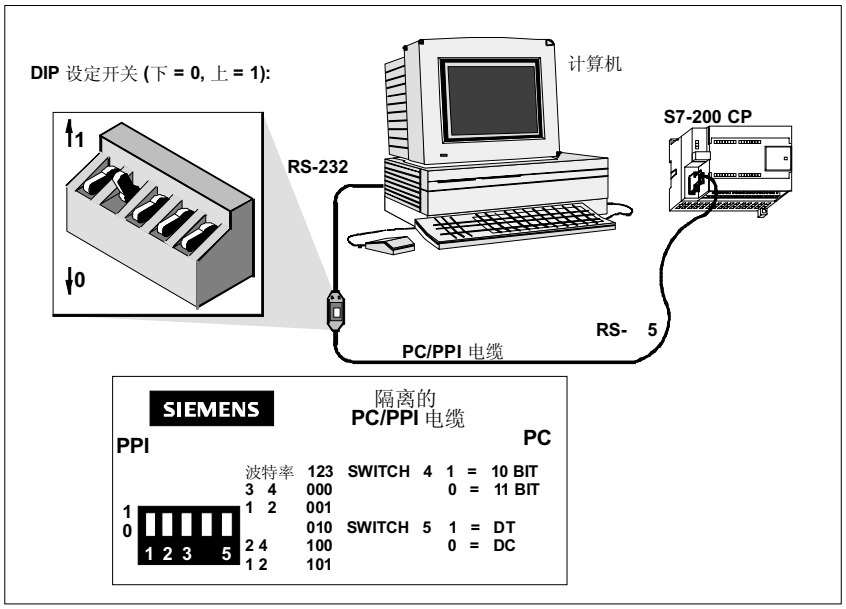


图 7-27　　PPI 方式的 CPU 通讯

PC/PPI 电缆的开关 4 告诉 S7–200 CPU 是使用 10 位协议或常规的 11 位 PPI 协议。如果 CPU 不连接到 STEP 7-Micro/WIN 32，这个开关不使用，应该放置在 11- 位设置的位置，便于其它设备的操作。

表 7-10　RS–485 到 RS-232 DTE 连接器引针

RS–485 连接器引针		RS–232 DTE 连接器引针[1]	
引针号	信号说明	引针号	信号说明
1	地 (RS–485 逻辑地)	1	数据载波检测 (DCD) (不用)
2	24 V 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	+5 V (带 100 □Ω串联电阻)	6	数据设置就绪 (DSR) (不用)
7	24 V 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　　RS–485 到 RS–232 DCE 连接器引针

RS–485 连接器引针		RS–232 DCE 连接器引针	
RS–485 连接器引针		RS–232 DTE 连接器引针[1]	
引针号	信号说明	引针号	信号说明
1	地 (RS–485 逻辑地)	1	数据载波检测 (DCD) (不用)
2	24 V 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 电缆)
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	+5 V (带 100 Ω 串联电阻)	6	数据设置就绪 (DSR) (不用)
7	24 V 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)

通过 4- 开关 PC/PPI 电缆使用调制解调器

可以用 4– 开关 PC/PPI 电缆把 S7 200 CPU 连到调制解调器的 RS–232 通讯口。允许计算机使用 RS–232 控制信号 (例如 RTS、CTS和 DTR) 控制调制解调器。这个 PC/PPI 电缆不使用这些信号，所以当利用 4- 开关 PC/PPI 电缆使用调制解调器时，调制解调器必须配置成不带这些信号的操作。 最小要求也是必须把调制解调器配置成忽略 RTS 和 DTR。参照调制解调器的操作手册，决定需要组态调制解调器的命令。

调制解调器是数据通讯设备 (DCE) 类。4- 开关 PC/PPI 电缆的 RS–232 口也是数据通讯设备 (DCE) 类。当连接相同类 (两个都是 DCE) 的两个设备时，数据发送和数据接收引针必须交换。调制解调器适配器交换这些引针。典型设置和调制解调器适配器的引针分配如图 7-28 所示。

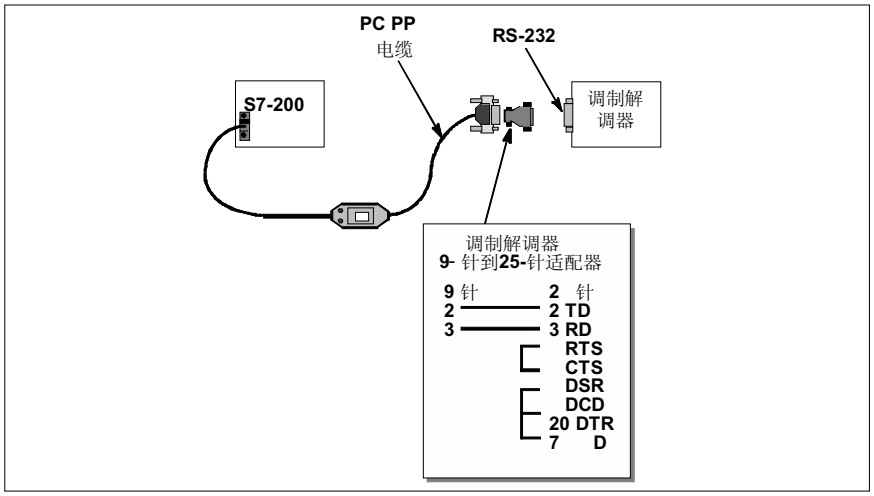


图 7-28　　带有 9- 针到 25- 针适配器的 11 位调制解调器

7.8 网络性能

优化网络性能

波特率和主站数是影响网络性能的两个主要因素。以所有设备支持的最高波特率进行网络操作会得到最佳的通讯效果。减少网络中的主站数目可以提高网络性能。网络中的每个主站会增加网络的负载要求。主站少可以减轻网络负载。

下面的因素也影响网络性能：

- 主站和从站地址选择
- 间隙刷新因子 (GUF)
- 最高站地址

所有主站的地址按照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。

S7–200 CPU可以配置成每隔一定时间检查地址间隙。利用 STEP 7-Micro/WIN32， 通过设置 CPU 配置中的间隙刷新因子 (GUF) 完成检查配置。GUF 告诉 CPU 多长时间检查地址间隙寻找其它主站。GUF1 每次占有令牌时告诉 CPU 检查地址间隙。GUF 2 每两次占有令牌时告诉 CPU 检查地址间隙。如果在主站之间有间隙，设置高的 GUF 可以降低网络负载。如果主站之间没有地址间隙，GUF 不影响网络性能。由于不频繁检查地址，设置大的 GUF 会造成长的延时。只有当把 CPU 作为 PPI 主站时才使用 GUF。

最高站地址 (HSA) 定义了一个主站寻找其它主站的最高地址。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。最高的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。HSA 只有当 CPU 作为PPI主站时 HSA 才有用。可以用 STEP 7-Micro/WIN 32 在 CPU 配置中为一个 CPU口设定 HSA 。

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。

令牌循环

在令牌传送网络中，只有拥有令牌的站有初始化通讯的权力。因此，对于象 PPI这样的令牌传送网络的重要品质因素为令牌循环时间。这是将令牌传到逻辑环中所有主机 (令牌拥有者) 所需的时间。为了描述 PPI 网络的操作，采用图 7-29所示的例子。

图 7-29 中的网络有 4 个 S7 200 CPU 模块，每一个都有自己的 TD 200。两个 CPU224 模块收集来自其它所有 CPU 模块的数据。

注意:

图 7-29 给出的例子是这样的网络。配置包括 TD 200，CPU 224 使用 NETR和 NETW 指令。如图 7-30 令牌占有时间和令牌循环时间的计算公式也是基于这个配置。

COM PROFIBUS 提供一个决定网络性能的分析器。

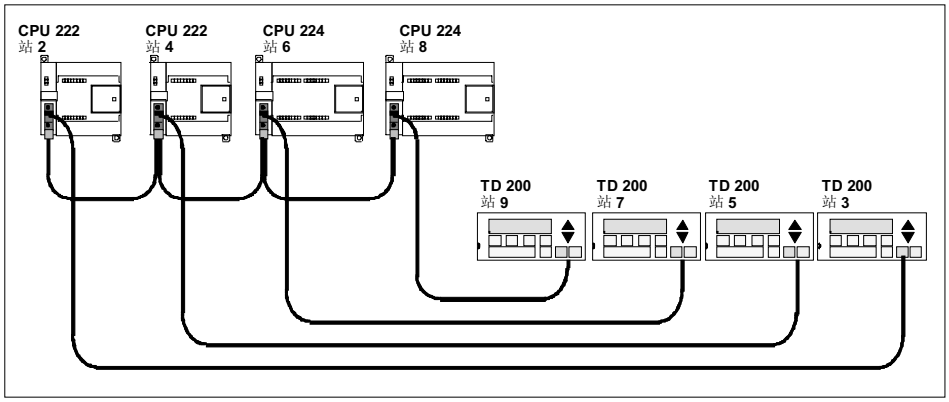


图 7-29　令牌传递网络举例

在这个配置中，TD 200 (站 3) 和 CPU 222 (站 2) 通讯，TD 200 (站 5) 和 CPU 222 (站 4) 通讯，等等。同时 CPU 224 (站 6) 向站 2、4、8 发送信息，而CPU 224 (站 8) 向站 2、4、6 发送信息。在这个网络里，有 6 个主站 (4 个TD 200 单元和两个 CPU 224 模块) 和两个从站 (两个 CPU 222 模块)。

发送信息

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

令牌循环时间由各站占有令牌的时间决定。对于 S7 200 网络，令牌循环时间可由各主机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)，CPU 利用网络读 (NETR) 和网络写 (NETW) 指令可以向其它 CPU 发送信息。这些指令的描述请见第九章的 9.15 节 SIMATIC 通讯指令。如果采用这些指令发送信息，并且下述假设成立，令牌循环时间可由图 7-29 所示公式近似得出：

- 各站占有令牌时发送一个请求。
- 请求为连续数据位置的读或写请求。
- CPU 的通讯缓冲区使用没有冲突。
- CPU 的扫描时间都不超过 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/秒= 151.25 ms/主机

T (令牌循环时间) = 151.25 ms/主机 * 6 主机=907.5ms

(一位时间等于一个信号的持续时间)

图 7-30 用NETR和 NETW 指令时令牌占用时间和令牌循环时间的公式

令牌循环比较

表 7-12、表 7-13 和表 7-14 给出了 9.6 k、19.2 k 和 187.5 k 波特率下的通讯站数和循环时间的比较。时间考虑 CPU 或其它主站采用网络读 (NETR) 和网络写 (NETW) 指令的情况。

表 7-12 9.6 k 波特率时通讯站数和数据量与令牌循环时间的关系

9.6 k波特率时每个站传输的字节数	站数及以秒表示的时间								
	2 个站	3 个站	4 个站	5 个站	6 个站	7 个站	8 个站	9 个站	10 个站
1	0.30	0.44	0.59	0.74	0.89	1.03	1.18	1.33	1.48
2	0.30	0.45	0.60	0.74	0.89	1.04	1.19	1.34	1.49
3	0.30	0.45	0.60	0.75	0.90	1.05	1.20	1.35	1.50
4	0.30	0.45	0.61	0.76	0.91	1.06	1.21	1.36	1.51
5	0.30	0.46	0.61	0.76	0.91	1.07	1.22	1.37	1.52
6	0.31	0.46	0.61	0.77	0.92	1.07	1.23	1.38	1.54
7	0.31	0.46	0.62	0.77	0.93	1.08	1.24	1.39	1.55
8	0.31	0.47	0.62	0.78	0.94	1.09	1.25	1.40	1.56
9	0.31	0.47	0.63	0.78	0.94	1.10	1.26	1.41	1.57
10	0.32	0.47	0.63	0.79	0.95	1.11	1.27	1.42	1.58
11	0.32	0.48	0.64	0.80	0.96	1.11	1.27	1.43	1.59
12	0.32	0.48	0.64	0.80	0.96	1.12	1.28	1.44	1.60
13	0.32	0.48	0.65	0.81	0.97	1.13	1.29	1.45	1.62
14	0.33	0.49	0.65	0.81	0.98	1.14	1.30	1.46	1.63
15	0.33	0.49	0.66	0.82	0.98	1.15	1.31	1.47	1.64
16	0.33	0.50	0.66	0.83	0.99	1.16	1.32	1.49	1.65

表 7-13　19.2 k 波特率时通讯站数和数据量与令牌循环时间的关系

19.2 k波特率时每个站传输的字节数	站数及以秒表示的时间								
	2 个站	3 个站	4 个站	5 个站	6 个站	7 个站	8 个站	9 个站	10 个站
1	0.15	0.22	0.30	0.37	0.44	0.52	0.59	0.67	0.74
2	0.15	0.22	0.30	0.37	0.45	0.52	0.60	0.67	0.74
3	0.15	0.23	0.30	0.38	0.45	0.53	0.60	0.68	0.75
4	0.15	0.23	0.30	0.38	0.45	0.53	0.61	0.68	0.76
5	0.15	0.23	0.30	0.38	0.46	0.53	0.61	0.69	0.76
6	0.15	0.23	0.31	0.38	0.46	0.54	0.61	0.69	0.77
7	0.15	0.23	0.31	0.39	0.46	0.54	0.62	0.70	0.77
8	0.16	0.23	0.31	0.39	0.47	0.55	0.62	0.70	0.78
9	0.16	0.24	0.31	0.39	0.47	0.55	0.63	0.71	0.78
10	0.16	0.24	0.32	0.40	0.47	0.55	0.63	0.71	0.79
11	0.16	0.24	0.32	0.40	0.48	0.56	0.64	0.72	0.80
12	0.16	0.24	0.32	0.40	0.48	0.56	0.64	0.72	0.80
13	0.16	0.24	0.32	0.40	0.48	0.57	0.65	0.73	0.81
14	0.16	0.24	0.33	0.41	0.49	0.57	0.65	0.73	0.81
15	0.16	0.25	0.33	0.41	0.49	0.57	0.66	0.74	0.82
16	0.17	0.25	0.33	0.41	0.50	0.58	0.66	0.74	0.83

表 7-14　187.5 k 波特率时通讯站数和数据量与令牌循环时间的关系

187.5 k波特率时 每个站传输的字节数	站数及以毫秒表示的时间								
	2 个站	3 个站	4 个站	5 个站	6 个站	7 个站	8 个站	9 个站	10 个站
1	8.68	13.02	17.37	21.71	26.05	30.39	34.73	39.07	43.41
2	8.80	13.20	17.60	22.00	26.40	30.80	35.20	39.60	44.00
3	8.92	13.38	17.83	22.29	26.75	31.21	35.67	40.13	44.59
4	9.03	13.55	18.07	22.59	27.10	31.62	36.14	40.66	45.17
5	9.15	13.73	18.30	22.88	27.46	32.03	36.61	41.18	45.76
6	9.27	13.90	18.54	23.17	27.81	32.44	37.08	41.71	46.35
7	9.39	14.08	18.77	23.47	28.16	32.85	37.55	42.24	46.93
8	9.50	14.26	19.01	23.76	28.51	33.26	38.02	42.77	47.52
9	9.62	14.43	19.24	24.05	28.86	33.67	38.49	43.30	48.11
10	9.74	14.61	19.48	24.35	29.22	34.09	38.95	43.82	48.69
11	9.86	14.78	19.71	24.64	29.57	34.50	39.42	44.35	49.28
12	9.97	14.96	19.95	24.93	29.92	34.91	39.89	44.88	49.87
13	10.09	15.14	20.18	25.23	30.27	35.32	40.36	45.41	50.45
14	10.21	15.31	20.42	25.52	30.62	35.73	40.83	45.84	51.04
15	10.33	15.49	20.65	25.81	30.98	36.14	41.30	46.46	51.63

S7–200 指令规约

8

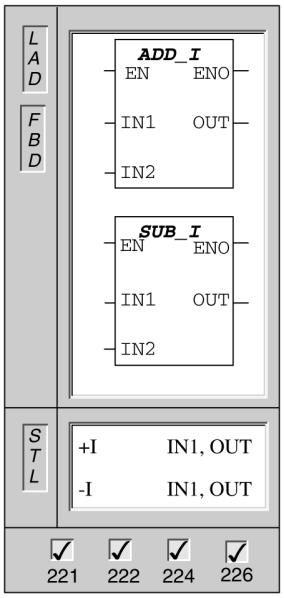
本章用到的如下规约解释等价的梯形图、功能块图和语句表指令和使用这些指令的 CPU 型号。

本章概述

8.1 STEP 7-Micro/WIN 32 编程的概念和规约

下图说明了本章使用的 Micro/WIN 32 指令格式。下面给出图中的指令格式的解释。

整数加法和整数减法



整数的加减指令把两个 16 位整数相加或相减，得到一个 16 位的结果 (OUT) 。

在梯形图和功能块图中：

IN1 + IN2 = OUT

IN1 - IN2= OUT

在语句表中：

IN1 + OUT = OU T

OUT - IN1=OUT

使 ENO = 0 的错误条件：

SM1.1 (溢出) ，SM4.3 (运行) ， 0006 (间接寻址)

这些指令下面特殊存储器位：

SM1.0 (零位) ；SM1.1 (溢出) ； SM1.2 (负号位)

输入 / 输出	操作数	数据类型
IN1, IN2	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, Constant, *VD, *AC, *LD	INT
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *AC, *LD	INT

指令或指令组的标题：在本例中，**整数加法和整数减法**是标题。

说明 Micro/WIN 32 指令的图形：在指令标题下的图形是梯形图 (LAD) 指令的元件图、功能块图 (FBD) 的元件图和 SIMATIC 指令方式下的语句表 (STL) 。在一些情况下，梯形图和功能块图的指令是一样的，只给出包含梯形图和功能块图的盒 (本例中的情况) 。SIMATIC 语句表 指令的助记符盒操作数在另一个盒中给出。

在本例中，梯形图/功能块图图形有三个输入 (输入总是在图形的左边) 和两个输出 (输出总是在图形的右边) 。在梯形图中，有两种基本的输入和输出类型。第一类输入/输出是输入或输出的能量流。

梯形图基于继电器梯形逻辑电气图。在梯形图中，有一个提供能量的左母线。触点闭合可以使能量流过该器件，到下一个器件，触点打开将阻止能量流断过。任何可以连到左/右母线或触点的梯形图元件都有输入/输出能量流。

在 SIMATIC 功能块图中，不使用左/右母线的概念。“能量流”的术语用来表示流过功能块图逻辑模块的控制流概念。通过功能块图元件的逻辑 “1” 称为能量流。

在梯形图中，一个能量流输入或输出总是限于能量的流动，而不能分配给一个操作数。在功能块图中，能量流的起点和能量流的终点可以直接分配给一个操作数。

除具有能量流外，许多 (不是所有的) 指令具有一个或多个输入/输出操作数。输入和输出操作数允许的参数在 LAD/FBD/STL 图下面的输入/输出表中给出。

CPU型号: 指令图给出了支持该指令的 CPU 型号。在本例中，指令支持 CPU 221、CPU 222、CPU 224 和 CPU 226.

指令描述: 在 8-2 页的指令图右边的文字描述了指令的操作。在一些情况下，每种语言有一个指令操作的描述，在其它情况下，给出一个适用于三种编程语言的简单描述。注意 IEC 的术语与 SIMATIC 术语只有细微的差别。例如，SIMATIC 的加计数 (CTU) 称为指令，而 IEC 的 CTU 称为功能块。

使 ENO = 0 的错误条件: 如果 LAD/FBD 指令有 ENO 输出，该处给出导致 ENO变为 0 的错误条件。

受影响的 SM 位: 如果在指令执行器件影响 SM 位，该处给出被影响的位和它们如何被影响的。

操作数表: 在 LAD/FBD/STL 图下面是每个输入和输出的允许操作数列表，还有每个操作数的数据类型。表8-3给出了每个 CPU 操作数的存储器范围。

由于所有 LAD和FBD 指令的操作数是一样的，在指令操作数表中没有给出EN/ENO 操作数和数据类型。表 8–1 列出了这些 LAD和FBD 的操作数和数据类型。这些操作数应用于本手册的所有 LAD 和 FBD 指令。

表 8-1 LAD 和 FBD 的 EN/ENO 操作数和数据类型

语言编辑器	输入 / 输出	操作数	数据类型
LAD	EN	能量流	BOOL
	ENO	能量流	BOOL
FBD	EN	I, Q, M, S, SM, T, C, V, L, 能量流	BOOL
	ENO	I, Q, M, S, SM, T, C, V, L, 能量流	BOOL

编程的一般规约

网络: 在梯形图中，程序被分成称为网络的一些段。一个网络是触点、线圈和功能框的有序排列。这些元件连接在一起组成一个左母线和右母线之间的完整电路。 (不存在短路、开路和反向能量流) 。STEP 7-Micro/WIN 32 允许以网络为单位给 LAD 程序建立注释。

FBD 编程使用网络概念给程序分段和加注释。STL 程序不使用网络，但是，可以使用 NETWORK 这个关键词对程序分段。 如果这样，程序可以转换成 LAD 或 FBD。

执行分区: 在 LAD、FBD 或 STL 中，一个程序包含至少一个命令部分和其它可选部分。命令部分是主程序。可选部分包括一个或多个子程序或者中断程序。通过选择或点击 STEP 7-Micro/WIN 32 中的分区选项，可以容易地切换程序的分区。

EN/ENO 定义: EN (允许输入) 是 LAD 和 FBD 中功能框的布尔量输入。对要执行的功能框，这个输入必须存在能量流。 在 STL 中，指令没有EN输入，但是对于要执行的 STL 语句，栈顶的值必须是“1”。

ENO (允许输出) 是 LAD 和 FBD 中功能框的布尔量输出。如果功能框的 EN 输入存在能量流，功能框准确无误地执行了其功能，那么 ENO 输出将把能量流传到下一个单元。如果在执行过程中存在错误，那么能量流就在出现错误的功能框终止。

在 SIMATIC STL 中没有 ENO 输出，但是， 和带有 ENO 输出的 LAD 和FBD 指令相对应的 STL 指令设置一个特殊的 ENO 位。这个位可以用 STL 指令的 AENO (AND ENO) 访问，可以用来产生与功能框的 ENO 位相同的效果。

条件/无条件输入: 在 LAD 和 FBD 中， 与能量流有关的功能框或线圈用不是到左母线的连接表示。与能量流无关的线圈或功能框用一个直接到左母线的连接表示。图 8–1 给出了条件和无条件输入。

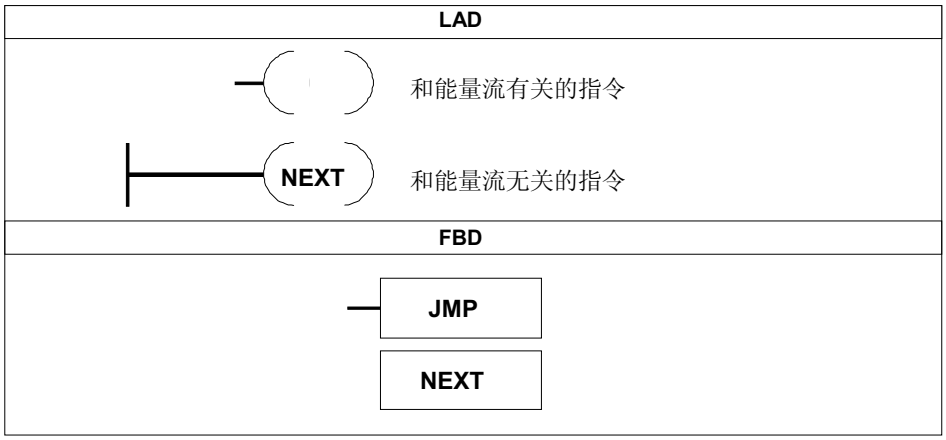


图 8-1 条件和无条件输入的 LAD 图

无输出的指令: 不能级联的指令盒用不带布尔输出来表示。它们是子程序调用、 JMP、CRET 等。也有只能放在左母线的梯形图线圈，它们包括是 LBL、NEXT、 SCR 和 SCRE 等。在 FBD 中，它们表示为指令盒，并把它们与不带标号的能量输入相区别。

比较指令: SIMATIC FBD、IEC 梯形图和 IEC FBD 的比较指令用指令盒表示，尽管比较操作是一个触点。

比较指令的执行和能量流的状态无关。如果能量流不存在，比较的输出就是“0”。如果能量流存在，比较的输出就和比较的结果有关。

STEP 7-Micro/WIN 32 规约: 在 STEP 7-Micro/WIN 32中，使用如下的规约：

- 一个符合所有的大写字母(ABC)表示该符号是全局符号。
- 带着重字符的符号名 #var1 表示该符号是局部符号。
- 符号 % 指示一个直接地址
- 操作数符号 “?” 或 “????” 指示需要一个值。

LAD的规约：在梯形图编辑器中，可以用 F4、F6 和 F9 键输入触点、输出盒和线圈指令。

- 符号 “- - ->” 是一个开路符号，或需要能量流连接。
- 符号 →| 表示输出是一个可选的能量流，用于指令的级联。
- 符号 “<<” 或 “>>” 指示有一个值或一个能量流可以使用。
- 一个连接到能量线的节点表示该指令独立于能量流(见表8-1)。
- 在 FBD 中，操作数或驱动输入的能量流状态的逻辑非条件用输入端的小圆圈表示。在图 8–2 中，Q0.0 等于 I0.0 的非和 I0.1 的 “与”。

FBD 中的规约：在FDB编辑器中，键盘中的 F4、F6 和 F9 键对应着与、或和输出指令：

- EN操作数上的"- - ->>"符号是能量流或操作数指示器。它表示开路或需要能量流连接。
- 符号 →| 表示输出是一个可选的能量流，用于指令的级联。
- 取非圆圈： 在 FBD 中，操作数或驱动输入的能量流状态的逻辑非条件用输入端的小圆圈表示。在图 8–2 中，Q0.0 等于 I0.0 的非和 I0.1 的“与”。

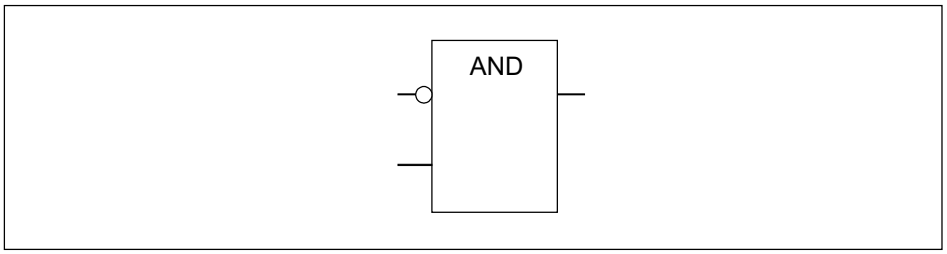


图 8-2 逻辑非条件的 FBD 图

- 立即指示： 在 FBD 指令中，布尔操作数的立即条件用输入端的垂线表示(图 8–3)。立即指示将立即读取特殊物理输入。立即操作只对物理输入有效。

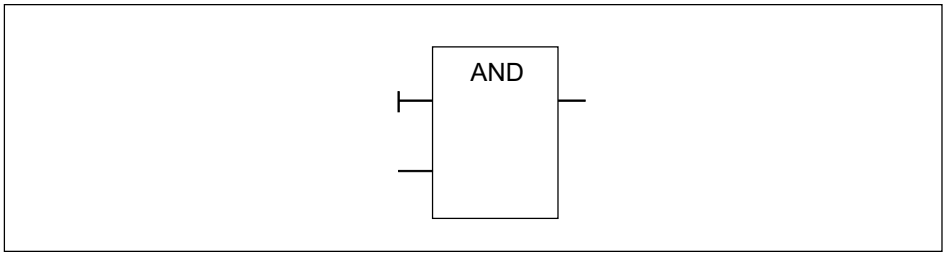


图 8-3 立即条件的 FBD 图

- Tab 键: Tab 键把光标从一个输入移动到另一个输入。选中的输入变为红色。光标从第一个输入到输出循环移动。
- 无输入的BOX：没有输入或输出的BOX表示该指令独立于能量流(见表8-1)。
- 操作数 tics：对于与、或指令，操作数可扩展到 32 个输入。使用“＋”、“－”键可增、减操作数 tics。

8.2 S7–200 CPU 的有效范围

表 8-2 S7–200 CPU 存储器范围和特性一览表

描 述	CPU 221	CPU 222	CPU 224	CPU 226
用户程序大小	2 K 字	2 K 字	4 K 字	4 K 字
用户数据大小	1 K 字	1 K 字	2.5 字	2.5 字
输入映像寄存器	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
模拟量输入（只读）	--	AIW0 - AIW30	AIW0 – AIW62	AIW0 - AIW62
模拟量输出（只写）	--	AQW0 - AQW30	AQW0 – AQW62	AQW0 - AQW62
变量存储器（V）[1]	VB0.0 - VB2047.7	VB0.0 - VB2047.7	VB0.0 - VB5119.7	VB0.0 — VB5119.7
局部存储器（L）[2]	LB0.0 - LB63.7	LB0.0 - LB63.7	LB0.0 - LB63.7	LB0.0—LB63.7
位存储器（M）	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 - SM179.7 SM0.0 - SM29.7	SM0.0 - SM179.7 SM0.0 - SM29.7	SM0.0 - SM179.7 SM0.0 - SM29.7
定时器 有记忆接通延迟1ms 有记忆接通延迟10ms 有记忆接通延迟100ms 接通/关断 延迟1ms 接通/关断 延迟10ms 接通/关断 延迟100ms	256 (T0 - T255) T0, T64 T1 - T4, T65 - T68 T5 - T31, T69 - T95 T32, T96 T33 - T36, T97 - T100 T37 - T63, T101 - T255	256 (T0 - T255) T0, T64 T1 - T4, T65 - T68 T5 - T31, T69 - T95 T32, T96 T33 - T36, T97 - T100, T37 - T63, T101 - T255	256 (T0 - T255) T0, T64 T1 - T4, T65 - T68 T5 - T31, T69 - T95 T32, T96 T33 - T36, T97 - T100, T37 - T63, T101 - T255	256 (T0 - T255) T0, T64 T1 - T4, T65 - T68 T5 - T31, T69 - T95 T32, T96 T33 - T36, T97 - T100, T37 - T63, T101 - T255
计数器	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
顺序控制继电器（S）	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
跳转/标号	0 - 255	0 - 255	0 - 255	0—255
调用/子程序	0 - 63	0 - 63	0 - 63	0—63
中断时间	0 - 127	0 - 127	0 - 127	0—127
PID 回路	0 - 7	0 - 7	0 - 7	0—7
端口	0	0	0	0, 1

[1]所有的 V 存储器都可以存储在永久存储器区。
[2]LB60 ~ LB63 为 STEP 7-Micro/WIN 32 的 3.0 版本或以后的版本软件保留

表 8-3　S7–200 CPU 操作数范围

存取方式	CPU 221	CPU 222	CPU 224，CPU 226
位存取 (字节.位)	V 0.0 - 2047.7 I 0.0 - 15.7 Q 0.0 - 15.7 M 0.0 - 31.7 SM 0.0 - 179.7 S 0.0 - 31.7 T 0 - 255 C 0 - 255 L 0.0 - 63.7	V 0.0 - 2047.7 I 0.0 - 15.7 Q 0.0 - 15.7 M 0.0 - 31.7 SM 0.0 - 179.7 S 0.0 - 31.7 T 0 - 255 C 0 - 255 L 0.0 - 63.7	V 0.0 - 5119.7 I 0.0 - 15.7 Q 0.0 - 15.7 M 0.0 - 31.7 SM 0.0 - 179.7 S 0.0 - 31.7 T 0 - 255 C 0 - 255 L 0.0 - 63.7
字节存取	VB 0 - 2047 IB 0 - 15 QB 0 - 15 MB 0 - 31 SMB 0 - 179 SB 0 - 31 LB 0 - 63 AC 0 - 3 常数	VB 0 - 2047 IB 0 - 15 QB 0 - 15 MB 0 - 31 SMB 0 - 179 SB 0 - 31 LB 0 - 63 AC 0 - 3 常数	VB 0 - 5119 IB 0 - 15 QB 0 - 15 MB 0 - 31 SMB 0 - 179 SB 0 - 31 LB 0 - 63 AC 0 - 3 常数
字存取	VW 0 - 2046 IW 0 - 14 QW 0 - 14 MW 0 - 30 SMW 0 - 178 SW 0 - 30 T 0 - 255 C 0 - 255 LW 0 - 62 AC 0 - 3 常数	VW 0 - 2046 IW 0 - 14 QW 0 - 14 MW 0 - 30 SMW 0 - 178 SW 0 - 30 T 0 - 255 C 0 - 255 LW 0 - 62 AC 0 - 3 AIW 0 - 30 AQW 0 - 30 常数	VW 0 - 5118 IW 0 -14 QW 0 - 14 MW 0 - 30 SMW 0 - 178 SW 0 - 30 T 0 - 255 C 0 - 255 LW 0 - 62 AC 0 - 3 AIW 0 - 30 AQW 0 - 30 常数
双字存取	VD 0 - 2044 ID 0 - 12 QD 0 - 12 MD 0 - 28 SMD 0 - 176 SD 0 - 28 LD 0 - 60 AC 0 - 3 HC 0, 3, 4, 5 常数	VD 0 - 2044 ID 0 - 12 QD 0 - 12 MD 0 - 28 SMD 0 - 176 SD 0 - 28 LD 0 - 60 AC 0 - 3 HC 0, 3, 4, 5 常数	VD 0 - 5116 ID 0 - 12 QD 0 - 12 MD 0 - 28 SMD 0 - 176 SD 0 - 28 LD 0 - 60 AC 0 - 3 HC 0 - 5 常数

SIMATIC 指令

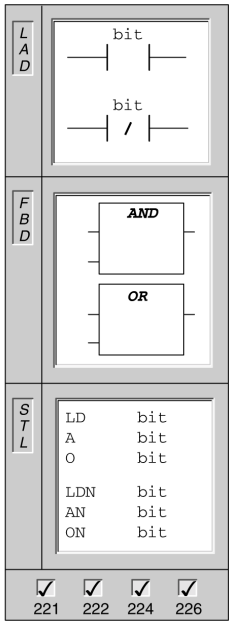
9

本章讲解 S7-200 SIMATIC 指令集。

本章概述

9.1 SIMATIC 位逻辑指令

标准触点



如果数据类型是 I 或 Q，这些指令从存储器或映像寄存器存取数值。对于 AND 和 OR 指令盒最多可以使用 7个输入。

当**常开** (NO) 触点对应的存储器地址位 (bit) 为 1 时,表示该触点闭合。

当**常闭** (NC) 触点对应的存储器地址位 (bit) 为 0 时，表示该触点闭合。

在梯形图 (LAD) 中，常开和常闭指令用触点表示。

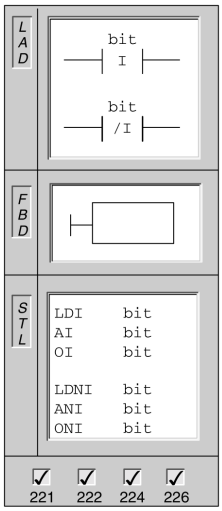
在功能块图 (FBD) 中，常开指令用 AND/OR 盒表示。和梯形图中的触点一样，这些指令用来处理布尔信号。常闭指令也用盒表示，用输入信号上加一个取非的圆圈来表示常闭指令。

在语句表 (STL) 中，常开触点由 LD (装载)，A (与) 及 O (或) 指令描述，LD 将位 bit 值装入栈顶，A，O 分别将位 bit 值与、或栈顶值，运算结果仍存入栈顶。

在语句表中，常闭触点由 LDN (非装载)，AN (非与) 和 ON (非或) 指令描述，LDN 将位 bit 值取反后再装入栈顶，AN、ON 先将位 bit 值取反，再分别与、或栈顶值，其运算结果仍存入栈顶。

输入/输出	操 作 数	数据类型
位 (LAD, STL)	I, Q, M, SM, T, C, V, S, L	BOOL
输入 (FBD)	I, Q, M, SM, T, C, V, S, L, 能流	BOOL
输出 (FBD)	I, Q, M, SM, T, C, V, S, L, 能流	BOOL

立即触点



当立即指令执行时，读取物理输入的值，但是不更新映像寄存器。

当常开立即触点的物理输入点 bit 的位值为 1 时，表示该触点闭合。

当常闭立即触点的物理输入点 bit 的位值为 0 时，表示该触点闭合。

在梯形图 (LAD) 中，常开和常闭指令用触点表示。

在功能块图 (FBD) 中，常开立即指令用操作数前加立即标示符表示。当使用能流时，可能没有立即标示符。和梯形图中的触点一样，这些指令用来处理布尔信号。

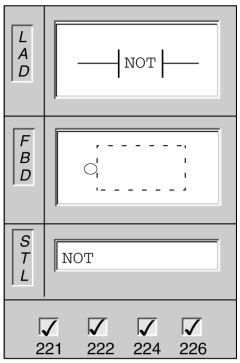
在功能块图 (FBD) 中，常闭立即指令也用操作数前加立即标示符和取负圆圈表示。当使用能流时，可能没有立即标示符，用输入信号上加一个取非的圆圈来表示常闭指令。

在语句表中，常开立即触点，由 LDI (立即装载)，AI (立即与) 及 OI (立即或) 指令描述。LDI指令把物理输入点 bit 的位值立即装入栈顶，AI，OI 分别将物理输入点 bit 的位值与、或栈顶值，运算结果仍存入栈顶。

在语句表中，常闭立即触点由 LDNI (立即非装载)，ANI (立即非与)，ONI (立即非或) 指令描述。LDNI 把物理输入点 bit 的位值取反后立即装入栈顶。ANI，ONI 先将物理输入点 bit 的位值取反，再分别与、或栈顶值，运算结果仍存入栈顶。

输入/输出	操 作 数	数据类型
位 (LAD, STL)	I	BOOL
输入 (FBD)	I	BOOL

取非



取非触点改变能流的状态。能流到达取非触点时，就停止；能流未到达取非触点，就通过。

在梯形图 (LAD) 中，取非指令用触点表示。

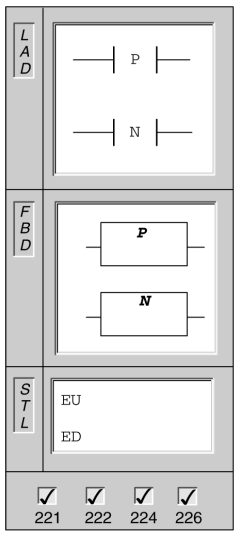
在功能块图 (FBD) 中，取非指令用带有非号的布尔盒输入表示。

语句表 (STL) 中，取非指令改变栈顶值，由 0 变到1，或者由 1 变到 0。

操作数：无

数据类型：无

正、负跳变



正跳变触点在检测到每一次正跳变 (从 off 到 on) 之后，让能流接通一个扫描周期。

负跳变触点在检测到每一次负跳变(从 on 到 off) 后，让能流接通一个扫描周期。

在梯形图 (LAD) 中，正、负跳变用触点表示。

在功能块图 (FBD) 中，正、负跳变用P和N指令盒表示。

在语句表 (STL) 中，正跳变触点由 EU 指令来描述，一旦发现栈顶的值出现正跳变 (由 0 到 1)，该栈顶值被置为 1，否则置 0。

在语句表 (STL) 中，负跳变触点由 ED 指令来描述。一旦发现栈顶的值出现负跳变 (由 1 到 0)，该栈顶值被置 1，否则置 0。

输入/输出	操 作 数	数据类型
IN (FBD)	I, Q, M, SM, T, C, V, S, L, 能流	BOOL
OUT (FBD)	I, Q, M, SM, T, C, V, S, L, 能流	BOOL

触点举例

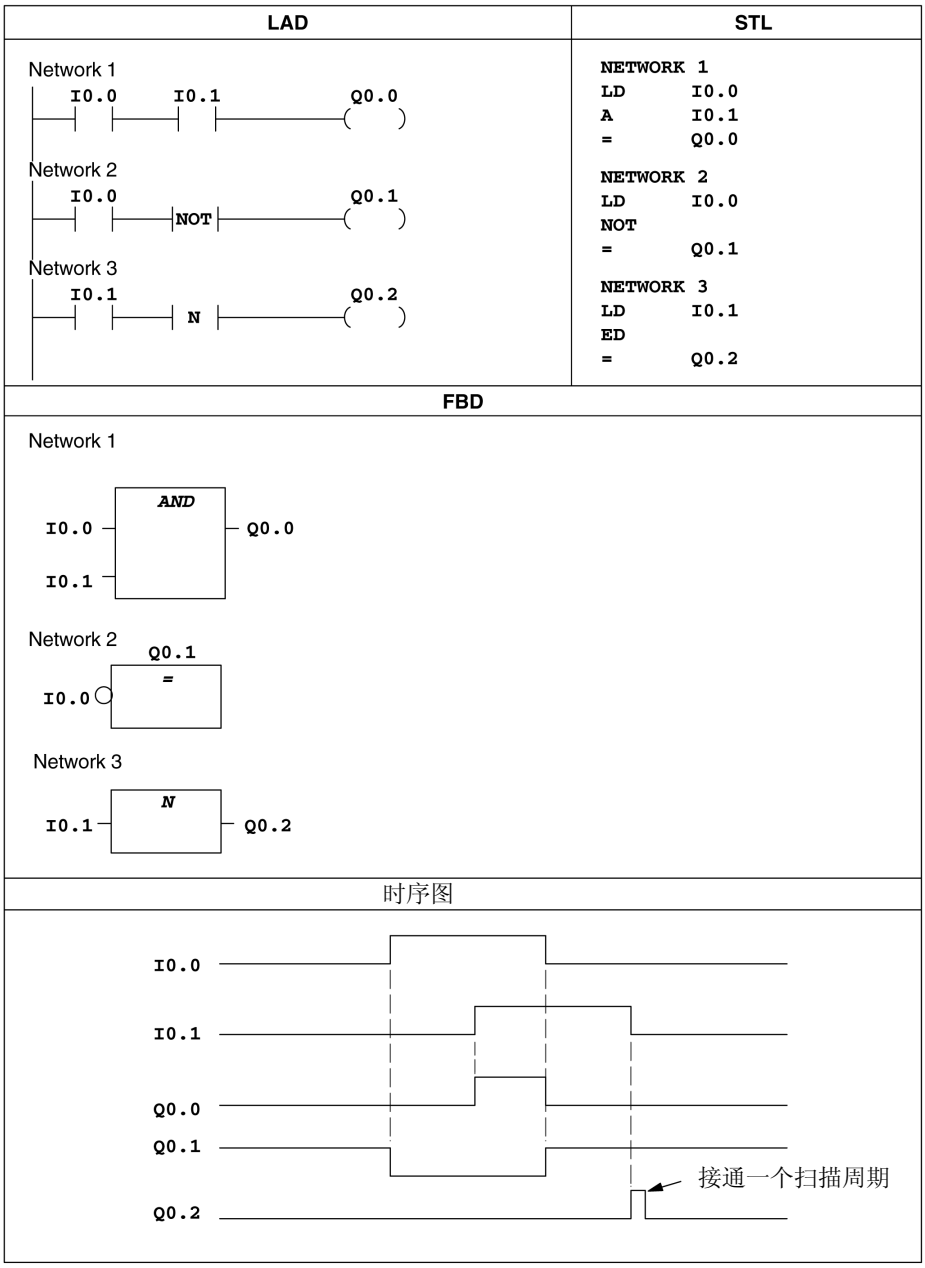
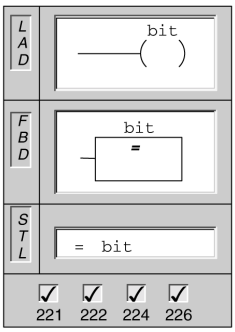


图 9-1　　SIMATIC LAD、STL、FBD 的0布尔触点指令举例

输出



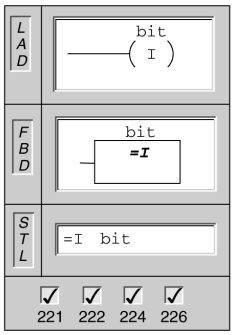
当执行输出指令时，映像寄存器中的指定参数位 (bit) 被接通。

在梯形图 (LAD) 和功能块图 (FBD) 中，当执行输出指令时，指定的位设为等于能流。

在语句表 (STL) 中，输出指令把栈顶值复制到指定参数位 (bit)。

输入/输出	操 作 数	数据类型
位	I, Q, M, SM, T, C, V, S, L	BOOL
输入 (LAD)	能流	BOOL
输入 (FBD)	I, Q, M, SM, T, C, V, S, L, 能流	BOOL

立即输出



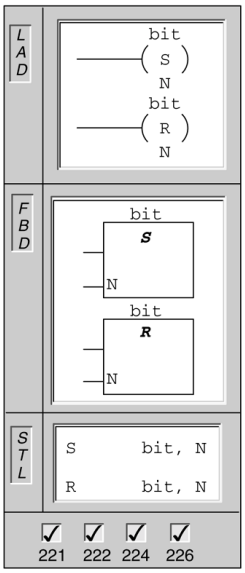
当执行立即输出指令时，该物理输出点 (bit或 OUT) 被设为等于能流。

指令中的“I”表示立即之意。当执行指令时，新值被同时写到物理输出点和相应的映象寄存器。这就不同于非立即输出，非立即输出只是把新值写到映象寄存器。

在语句表中，立即输出指令把栈顶值复制到指定物理输出点 (bit)。

输入/输出	操 作 数	数据类型
位	Q	BOOL
输入 (LAD)	能流	BOOL
输入 (FBD)	I, Q, M, SM, T, C, V, S, L, 能流	BOOL

置位和复位 (N 位)



执行置位 (置 1) 、复位 (置 0) 指令时，从 bit 或 OUT 指定的地址参数开始的N个点都被置位或复位。

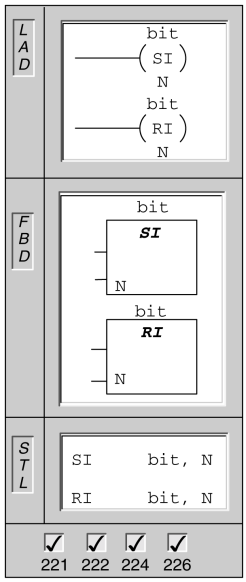
复位、置位的点数N可以是 1 - 255。当用复位指令时，如果 bit 或 OUT 指定的是 T 位或 C 位，那么定时器或计数器被复位，同时定时器或计数器当前值将被清零。

使 ENO = 0 的出错条件：

SM4.3 (运行时间)，0006 (间接寻址)，0091 (操作数超界)

输入/输出	操 作 数	数据类型
位	I, Q, M, SM, T, C, V, S, L	BOOL
N	VB, IB, QB, MB, SMB, SB, LB, AC, 常数,*VD, *AC, *LD	BYTE

立即置位和立即复位 (N 位)



当执行立即置位或复位指令时，从 bit 或 OUT 开始的 N 个物理 输出点将被立即置位或复位。

置位、复位的点数 N 可以是 1 - 128。

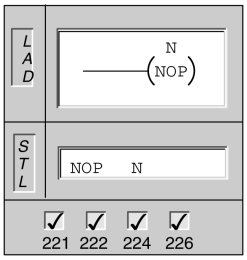
指令中的“I”表示立即之意。执行该指令时，新值被同时写到物理输出点和相应的映象寄存器。这是与非立即指令的区别，非立即指令只把新值写到映象寄存器。

使 ENO = 0 的出错条件：

SM4.3 (运行时间)，0006 (间接寻址)，0091 (操作数超界)

输入/输出	操 作 数	数据类型
位	Q	BOOL
N	VB, IB, QB, MB, SMB, SB, LB, AC, 常数, *VD, *AC, *LD	BYTE

空操作



空操作指令不影响程序的执行，操作数 N 是一个 0 到 255 之间的数。

操作数：N：常数 (0 - 255)

数据类型：BYTE

输出举例

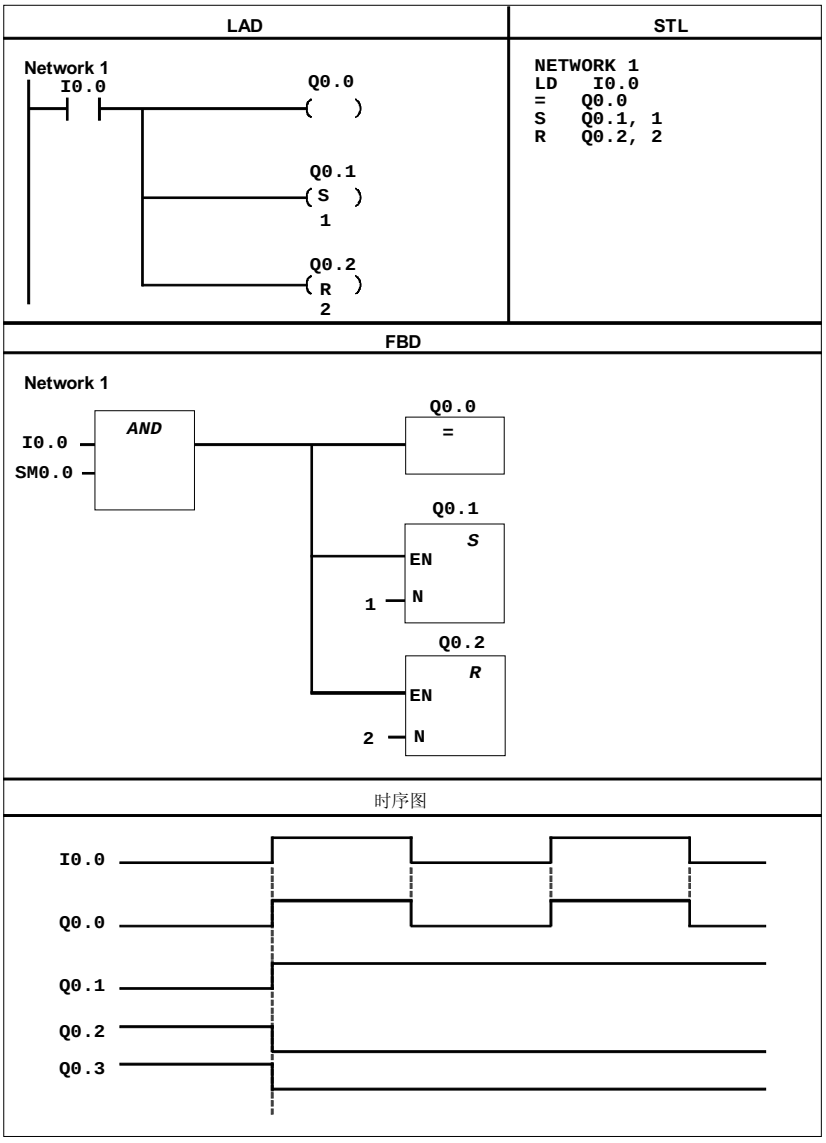
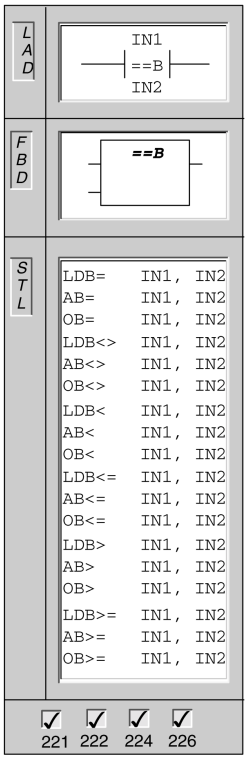


图 9-2　SIMATIC LAD、STL 和 FBD 输出指令举例

9.2 SIMATIC 比较指令

字节比较



字节比较指令用来比较两个值 IN1和 IN2 的大小，比较式可以是 IN1=IN2，IN1 >= IN2，IN1 <= IN2，IN1 > IN2，IN1 < IN2, 或IN1 <> IN2。

字节比较是无符号的。

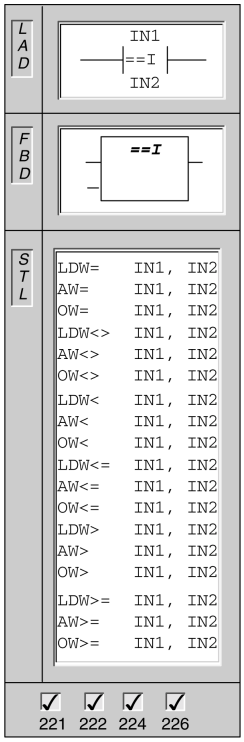
在 LAD 中，当比较式为真时，该触点闭合。

在 FBD 中，当比较式为真时，输出接通。

在语句表中，使用 LD、A 或 O 指令，当比较式为真时，将栈顶置 1。

输入/输出	操 作 数	数据类型
输入	IB, QB, MB, SMB, VB, SB, LB, AC, 常数, *VD, *AC,*LD	BYTE
输出 (FBD)	I, Q, M, SM, T, C, V, S, L, 能流	BOOL

整数比较



整数比较指令用来比较两个值 IN1 和 IN2 的大小，比较式可以是：IN1 = IN2，IN1 >= IN2，IN1 <= IN2，IN1 > IN2， IN1 < IN2 或 IN1 <> IN2。

整数比较是有符号的： (16#7FFF > 16#8000)。

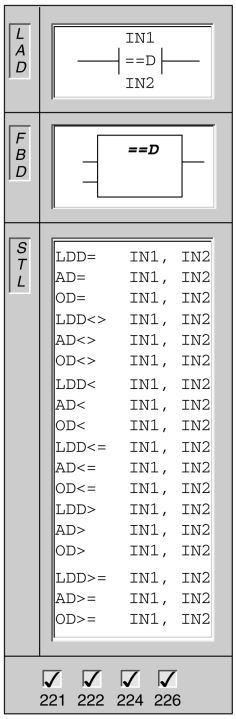
在 LAD 中，当比较式为真时，该触点闭合。

在 FBD 中，当比较式为真时，输出接通。

在 STL 中，使用 LD，A 或 O 指令，当比较式为真时，将栈顶置 1。

输入/输出	操 作 数	数据类型
输入	IW, QW, MW, SW, SMW, T, C, VW, LW, AIW, AC, 常数, *VD, *AC,*LD	INT
输出 (FBD)	I, Q, M, SM, T, C, V, S, L, 能流	BOOL

双字整型数比较



双字整型数比较指令用来比较两个双字整型数值 IN1和 IN2 的大小。其比较式可以是：IN1 = IN2，IN1 >= IN2，IN1 <= IN2，IN1 > IN2，IN1 < IN2，或 IN1 <> IN2。

双字比较是有符号的

(16#7FFFFFFF > 16#80000000)。

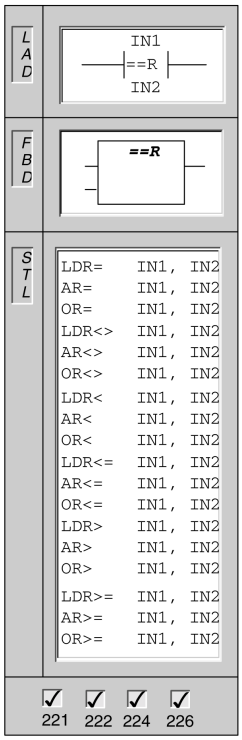
在 LAD 中，当比较式为真时，该触点闭合。

在 FBD 中，当比较式为真时，输出接通。

在 STL中，使用 LD，A，或 O 指令，当比较式为真时，将栈顶置 1。

输入/输出	操 作 数	数据类型
输入	ID, QD, MD, SD, SMD, VD, LD, HC, AC, 常数, *VD, *AC, *LD	DINT
输出 (FBD)	I, Q, M, SM, T, C, V, S, L, 能流	BOOL

实数比较



实数比较指令用来比较两个实数 IN1和 IN2 的大小。其比较式可以是：IN1 = IN2，IN1 >= IN2，IN1 <= IN2，IN1 > IN2，IN1 < IN2 或 IN1 <> IN2。

实数比较是有符号的。

在 LAD 中，当比较式为真时，该触点闭合。

在 FBD 中，当比较式为真时，输出接通。

在 STL 中，使用 LD，A 或 O 指令，当比较式为真时，将栈顶置1。

输入/输出	操 作 数	数据类型
输入	ID, QD, MD,SD, SMD, VD, LD, AC, 常数, *VD, *AC, *LD	REAL
输出 (FBD)	I, Q, M, SM, T, C, V, S, L, 能流	BOOL

比较触点举例

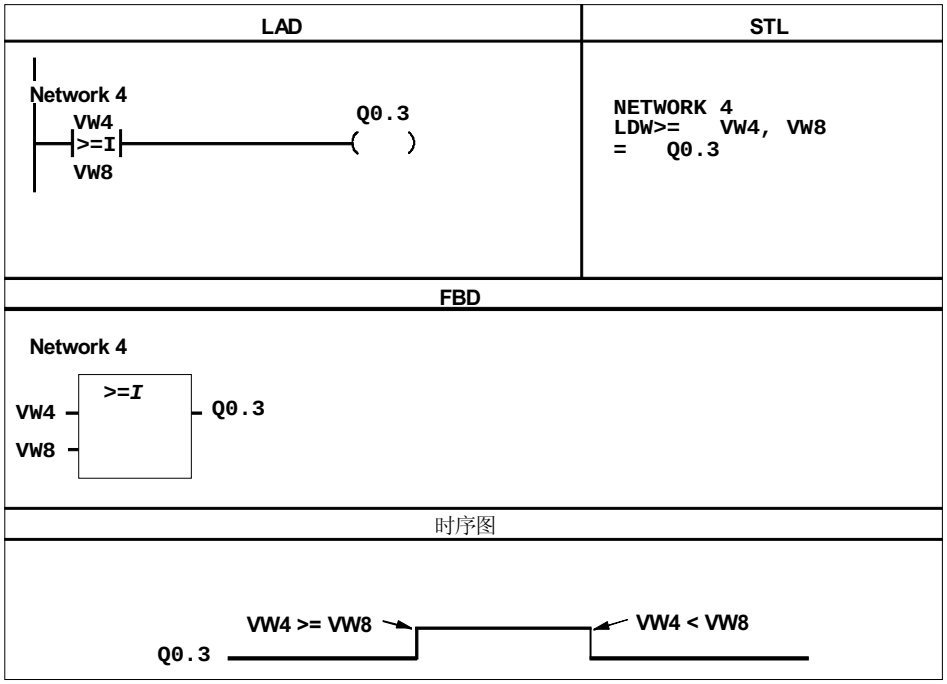
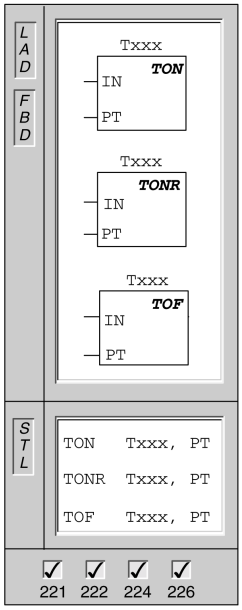


图 9-3 比较触点指令的 LAD、STL 和 FBD 举例

9.3 SIMATIC 定时器指令

接通延时定时器，有记忆接通延时定时器，断开延时定时器



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。

输入/输出	操 作 数	数据类型
Txxx	常数	字
IN (LAD)	能流	BOOL
IN (FBD)	I, Q, M, SM, T, C, V, S, L, 能流	BOOL
PT	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, 常数, *VD, *AC, *LD	INT

TON，TONR 和 TOF 定时器有三个分辨率。这些分辨率由表 9-1 中的定时器号决定。每个当前值的计数是多重时基，例如，一个以 10 ms 为时基的数 50 代表 500ms。

表 9-1 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	1 ms	32.767 s	T0, T64
	10 ms	327.67 s	T1 - T4, T65 - T68
	100 ms	3276.7 s	T5 - T31, T69 - T95
TON, TOF	1 ms	32.767 s	T32, T96
	10 ms	327.67 s	T33 - T36, T97 - T100
	100 ms	3276.7 s	T37 - T63, T101 - T255

注意：

不能把一个定时器号同时用作 TOF 和 TON，例如，不能既有 TON T32 又有 TOF T32。

理解 S7-200 定时器指令

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。 表 9–2 是不同定时器的功能。

- 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)用于故障事件后的时间延时(例如：在电机停后，需要冷却电机)

表 9-2 定时器功能

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 5.3 节。				

注意：

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。

下面给出定时器在不同分辨率时功能的解释。

1ms 分辨率

1ms 定时器对定时器启动后的 1ms 间隔进行计数，执行定时器指令启动定时，但是 1ms 定时器每隔 1ms 刷新一次 (定时器位和定时器当前值)，不和扫描周期同步。也就是说，定时器位和定时器当前值在扫描周期大于 1 ms 的一个周期中要刷新几次。

定时器指令用来启动、复位定时器，或在 TONR 中关闭定时器。

由于定时器在 1ms 内可以在任何地方启动，预设值必须大于最小需要时间间隔。例如，使用 1ms 定时器要确保至少 56ms 的时间间隔，预设值应该设为 57。

10 ms 分辨率

10ms 定时器对定时器启动后的 10ms 间隔进行计数，执行定时器指令启动定时，但是 10ms 定时器在每次扫描周期的开始刷新 (也就是说，在一个扫描周期内定时器位和定时器当前值保持)，把累计的 10ms 的间隔数加到启动的定时器的当前值。

由于定时器在 10ms 内可以在任何地方启动，预设值必须大于最小需要时间间隔。例如，使用 10ms 定时器要确保至少 140ms 的时间间隔，预设值应该设为 15。

100 ms 分辨率

100ms 定时器对定时器启动后的 100ms 间隔进行计数，执行定时器指令启动定时，但是 100ms 定时器在每次扫描周期的开始刷新 (也就是说，在一个扫描周期内定时器位和定时器当前值保持)，把累计的 100ms 间隔数加到启动的定时器的当前值。

只有定时器指令执行时，100ms 定时器的当前值才刷新。因此，如果 100ms 定时器激活，但是每个扫描周期没有执行定时器指令，定时器的当前值不刷新，造成时间丢失。同样地，如果在一个扫描周期内多次相同的 100ms 定时器指令，就会造成多计时间。定时器仅用在定时器指令每个扫描周期精确执行一次的地方。

由于定时器在 100ms 内可以在任何地方启动，预设值必须大于最小需要时间间隔。例如，使用 100ms 定时器时，为了保证至少 2100ms 的时间间隔，预设时间值应该设为 22。

定时器当前时间值的更新

如何使用定时器决定了各种更新当前时间值所造成的效果。图 9-4 所示的操作就是一例：

- 1ms 定时器的使用方法在修改之前，只有当该定时器的当前值更新发生在常闭触点 T32 执行以后及常开触点 T32 执行以前 Q0.0 才能被置位一个扫描周期，其他情况不能置位。
- 10ms 定时器的使用方法在修改以前，Q0.0 将永不会被置位 (ON)，因为定时位 T33 只能在每次扫描开始被置位。往后，执行定时器指令时，定时器将被复位 (当前值和T位将被置 0)。当常开触点 T33 被执行时，因 T33 为 off，所以 Q0.0 也会 off，即 Q0.0 永不会被置位 (ON)。
- 100ms 定时器的使用方法在修改以前，只要当定时器当前值达到预置值时，Q0.0 会被置位一个扫描周期。

用常闭触点 Q0.0 代替常闭触点 T32 作为定时器的允许计时输入，这就保证当定时器达到预置值时，Q0.0 会置位 (ON) 一个扫描周期。

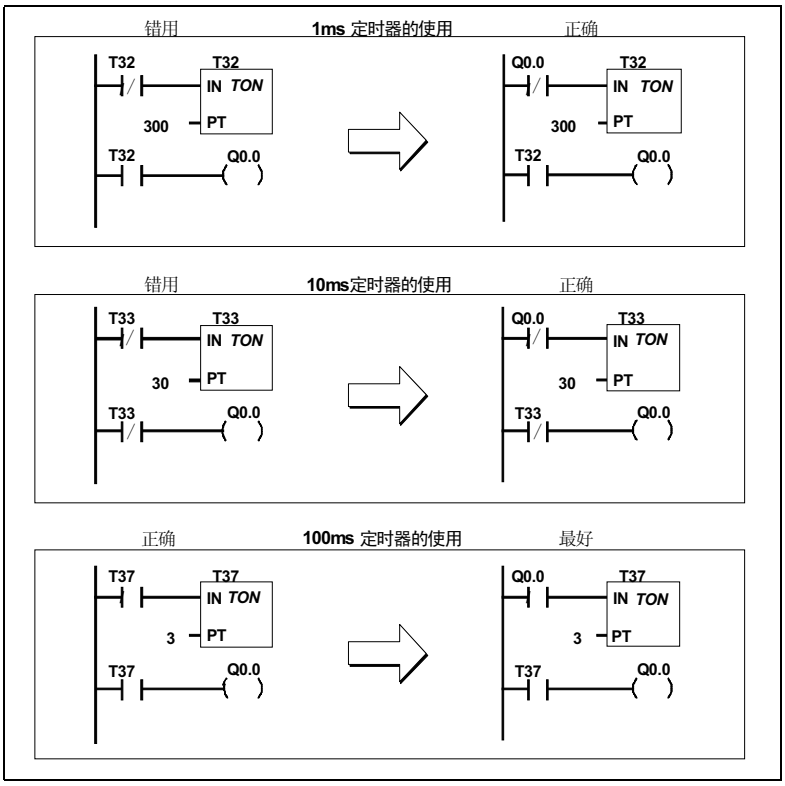


图 9-4　　自动触发一次定时器举例

接通延时定时器举例

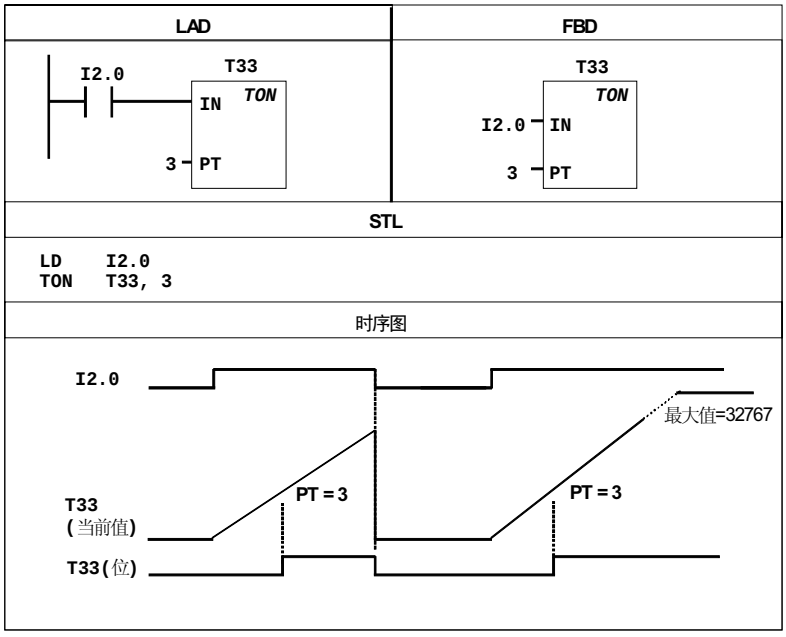


图 9-5　　接通延时定时器指令在 LAD、FBD 和 STL中应用举例

有记忆接通延时定时器举例

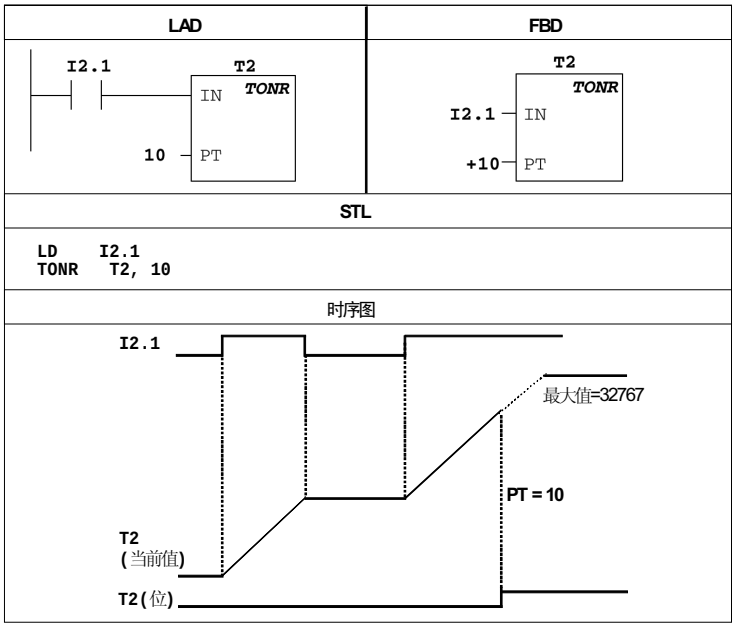


图 9-6　有记忆通电延时定时器在 LAD、FBD 和 STL 中应用举例

断开延时定时器举例

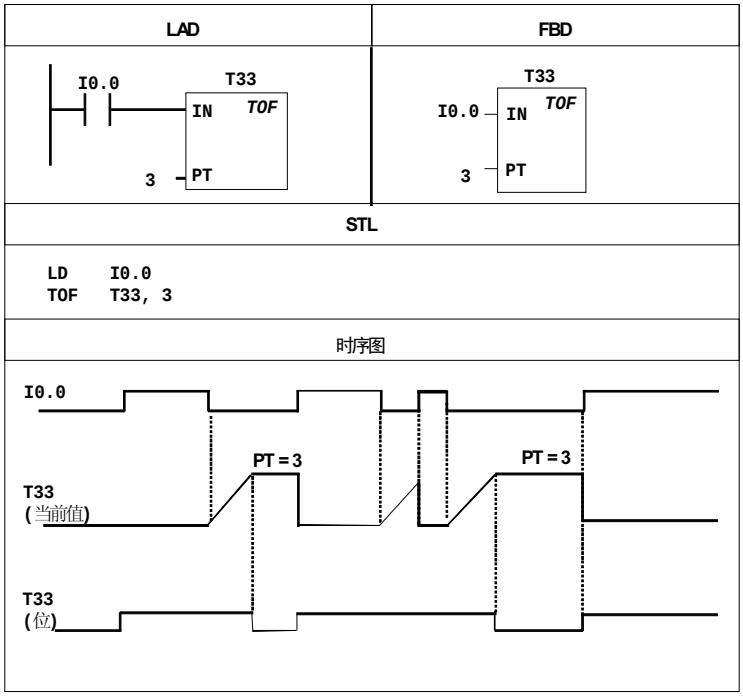
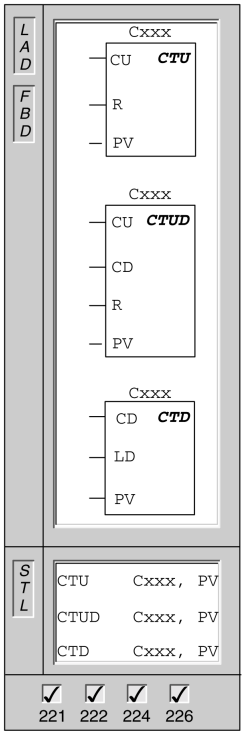


图 9-7　断开延时定时器在 LAD、FBD 和 STL 中应用举例

9.4 SIMATIC 计数器指令

增计数，增/减计数，减计数



增计数器指令 (CTU)，使该计数器在每一个 CU 输入的上升沿 (从 off 到 on) 递增计数，直至计数最大值。当当前计数值 (Cxxx) 大于或等于预置计数值 (PV) 时，该计数器位被置位。当复位输入 (R) 置位时，计数器被复位。

增/减计数器指令 (CTUD)，使该计数器在每一个 CU输入的上升沿递增计数；在每一个 CD 输入的上升沿递减计数。当当前值 (Cxxx) 大于或等于预置计数值 (PV) 时，该计数器位被置位。当复位输入 (R) 置位时，计数器被复位。

减增计数器指令 (CTD)，使该计数器在每一个 CD输入的上升沿 (从 off 到 on) 从预设值开始递减计数。当当前计数值 (Cxxx) 等于 0 时，该计数器位被置位。当复位输入 (R) 置位时，计数器把预设值 (PV) 装入当前值 (CV)。当计数值达到 0时，停止计数。

计数器范围：Cxxx=C0 到 C255

在语句表 (STL) 中，栈顶第一个值是 CTU 复位输入，第二个值是 CU 输入。

在语句表中，栈顶的第一个值是复位输入，第二个值是 CD 输入，第三个值是 CU 输入。

在语句表 (STL) 中，栈顶第一个值是 CTD 装载输入，第二个值是 CD 输入。

输入/输出	操 作 数	数据类型
Cxxx	常数	WORD
CU, CD , LD, R (LAD)	能流	BOOL
CU, CD , LD, R (FBD)	I, Q, M, SM, T, C, V, S, L, 能流	BOOL
PV	VW, IW, QW, MW, SMW, LW, AIW, AC, T, C, 常数, *VD, *AC, *LD, SW	INT

理解 S7-200 计数器指令

递增计数器指令 (CTU) 在每一个CU输入的上升沿 (从 off 到 on)，从当前计数值开始递增计数。当复位输入 (R) 置位或者执行复位指令时，计数器复位。计数器在达到最大计数值 (32767) 时停止计数。

递增 / 递减计数器指令 (CTUD) 在每个 CU 输入的上升沿，从当前计数值开始递增计数，在每个 CD 输入的上升沿，递减计数。当复位输入 (R) 置位或执行复位指令时，计数器复位。在达到计数器最大值 32767 后，下一个 CU 输入上升沿将使计数值变为最小值 (-32768)。 同样在达到最小计数值 (-32768) 后，下一个 CD 输入上升沿将使计数值变为最大值 (32767)。

增和增/减计数器的当前值记录当前的计数值。该种计数器的预置值在计数器指令执行期间用来与当前值作比较，当当前值大于等于预置值时，该计数器位被置位 (on)，否则，计数器位被复位 (off)。

当减计数输入端有上升沿时，减计数器每次从计数器的当前值减计数。当装载输入端接通时，计数器复位并把预设值装入当前值。当计数器达到 0 时，计数器位接通。

当用复位指令复位计数器时，计数器位被复位并且当前值清零。参照计数器的当前值和计数器位使用计数器号。

注意：

由于每个计数器只有一个当前值，请不要把一个计数器号分配给几个类型的计数器 (增计数器、增/减计数器和减计数器都使用同一个当前值)。

计数器举例

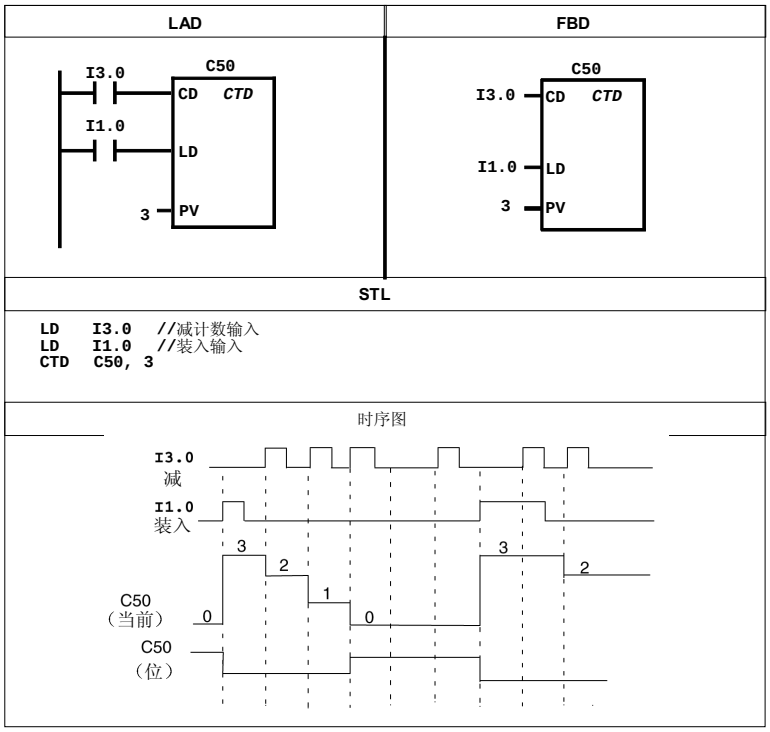


图 9-8 减计数器 (CTD) 指令的 LAD、FBD 和 STL 应用举例

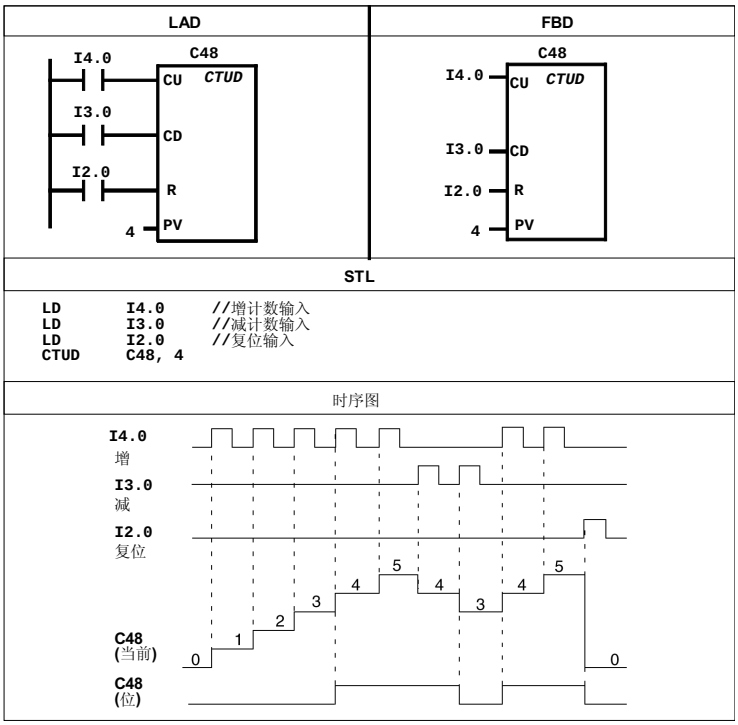
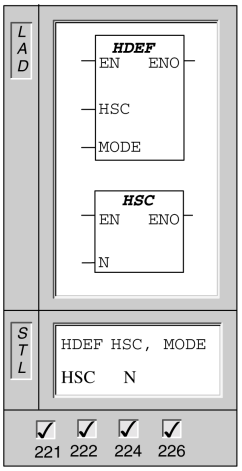


图 9-9 增/减计数器 (CTUD) 指令的 LAD、FBD 和 STL 应用举例

高速计数器定义，高速计数器



定义高速计数器指令为指定的高速计数器分配一种工作模式，见表 9-5。

高速计数器指令 (HSC) 执行时，根据 HSC 特殊存储器位的状态，设置和控制高速计数器的工作模式。参数N指定了高速计数器号。

CPU 221 和 CPU 222 不支持 HSC1 和 HSC2。

每个高速计数器只能用 1 个 HDEF。

使 ENO=0 的 HDEF 出错条件：

SM4.3 (运行时间)，0003 (输入冲突)，0004 (中断中的非法指令)，000A (HSC 重定义)

使 ENO=0 的 HSC 出错条件：

SM4.3 (运行时间) 0001 (在 HDEF 前使用 HSCHDEF)， 0005 (同时操作 HSC/PLS)

输入/输出	操 作 数	数据类型
HSC	常数	BYTE
MODE	常数	BYTE
N	常数	WORD

理解高速计数器指令

高速计数器累计 CPU 扫描速率不能控制的高速事件，可以配置最多 12 种不同的操作模式，这些操作模式在表 9-5 中列出。高速计数器的最高计数频率有赖于 CPU 的型号，有关 CPU 的详细信息参阅附录A。

每个计数器对它所支持的时钟，方向控制，复位和启动都有专用的输入。对于两相计数器，两个时钟可以同时以最大速率工作。对正交模式，可以选择以单倍 (1X) 或 4 倍 (4X) 最大计数速率工作。HSC1 和 HSC2 互相完全独立，并且不影响其它的高速功能。所有高速计数器可同时以最高速率工作而互不干扰。

使用高速计数器

一般来说，高速计数器被用作驱动鼓形计时器设备，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做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，这样使得程序简单直接，并容易读懂。当然，也可以在一个中断程序中处理所有的中断事件。若要更详细地了解中断，请参阅 9.15 节。

理解高速计数器的详细时序

下面的时序图 (图 9-10 到图 9-16) 按模式给出了每个计数器是如何工作的。复位和启动输入的操作用独立的时序图表示，并且对所有用到复位和启动输入的种类都给出了时序图。在复位和启动输入图中，复位和启动都编程为高电平有效。

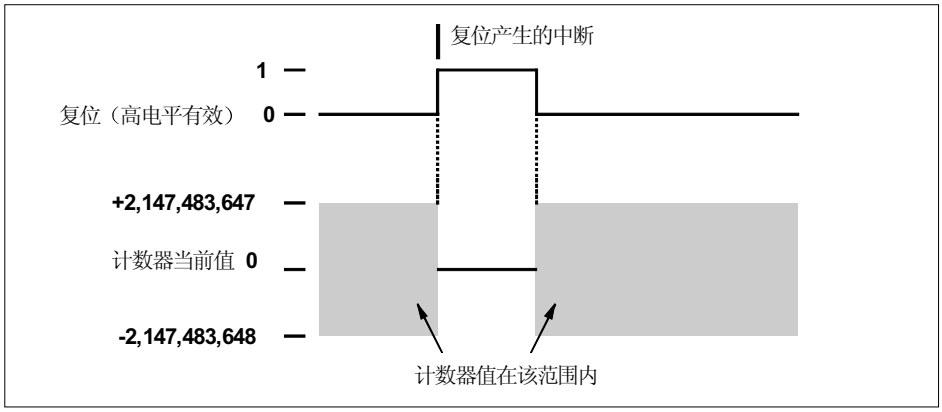


图 9-10　有复位无启动的操作举例

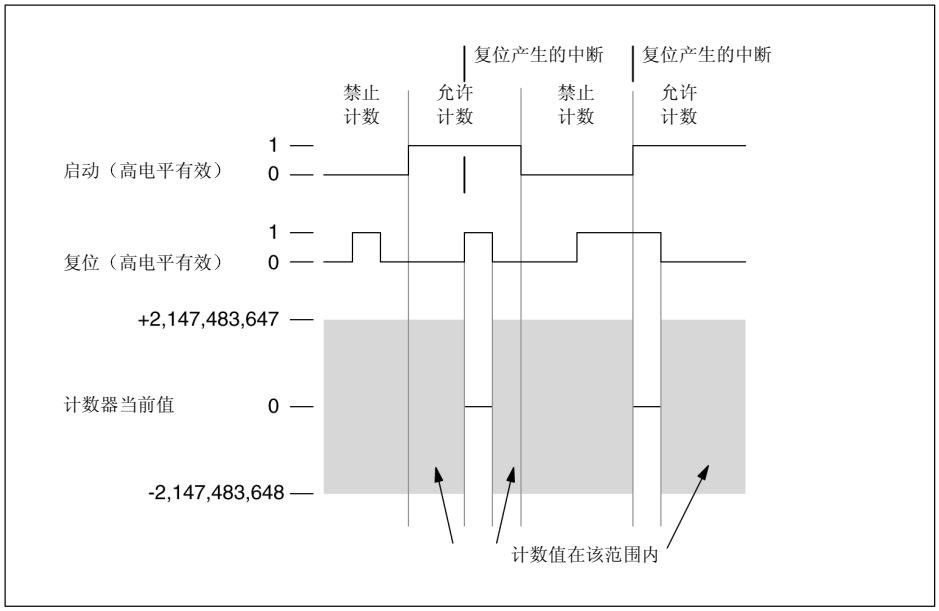


图 9-11　有复位和启动的操作举例

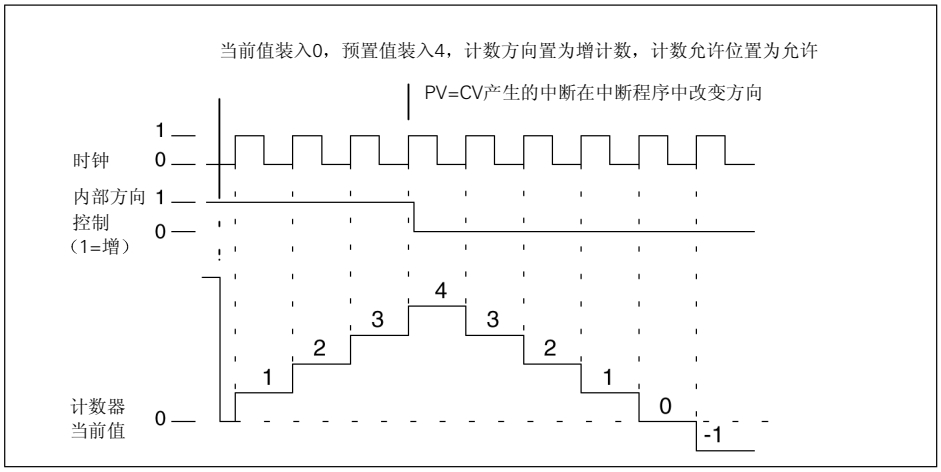


图 9-12　模式 0，1 或 2 的操作举例

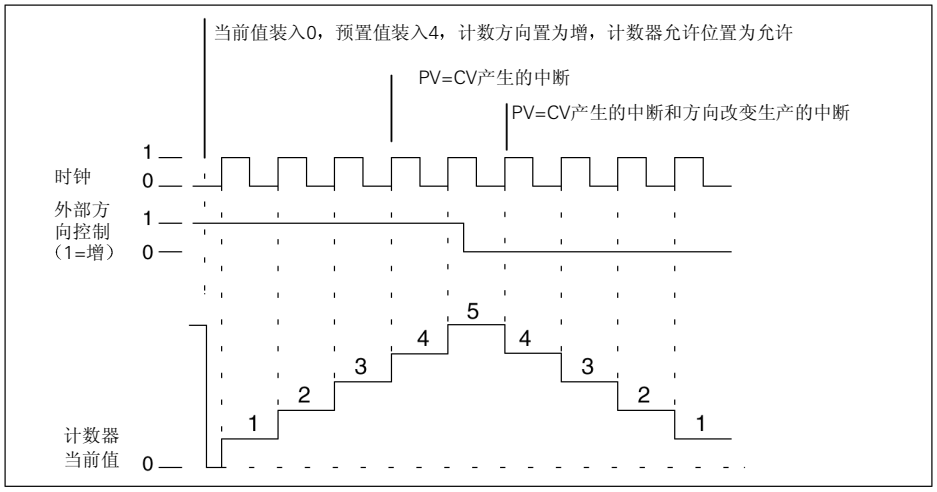


图 9-13　　模式 3，4 或 5 的操作举例

当采用计数模式 6，7 或 8 时，若增时钟和减时钟的上升沿出现彼此相差不到0.3 ms，高速计数器会认为这些事件是同时发生的。如果出现这种情况，当前值不会发生变化，也不会有计数方向变化的指示。当增时钟和减时钟的上升沿距离大于这个时间段 (0.3 ms) 时，高速计数器可以分别捕获到每一个独立事件。在任一情况下，都不会有错误产生，计数器会保持正确的计数值。请见图 9-14,图 9-15 和图 9-16。

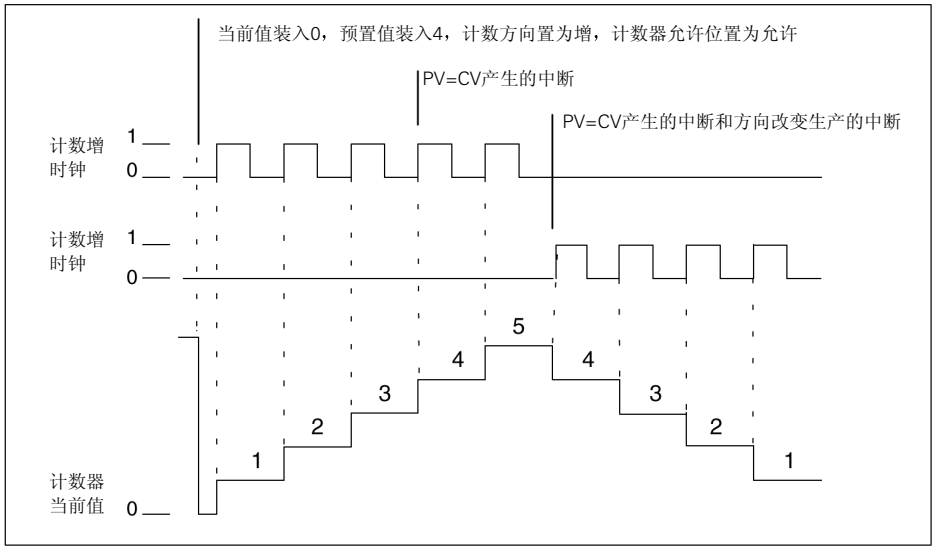


图 9-14　　模式 7，8 或 9 的操作举例

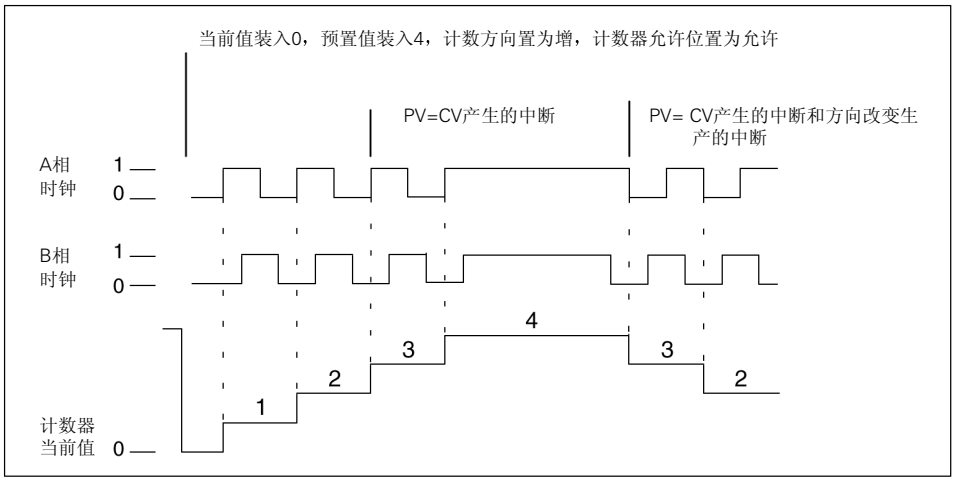


图 9-15　模式 9，10 或 11 (正交 1X模式) 的操作举例

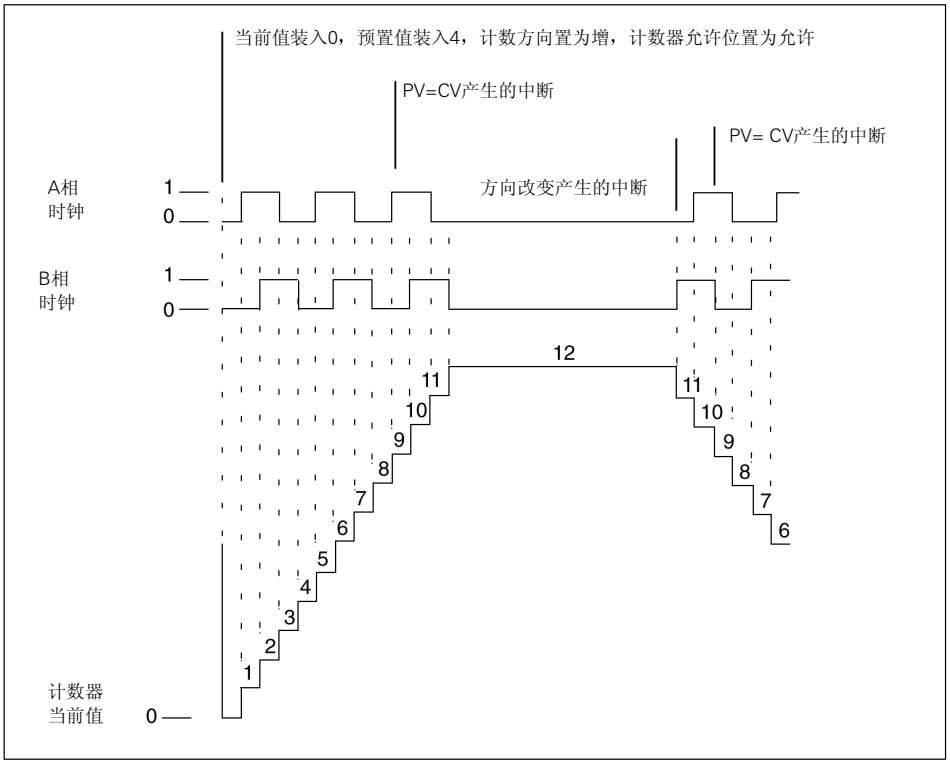


图 9-16　模式 9，10 或 11 (正交 4X模式) 的操作举例

高速计数器输入线的连接

表 9-3 给出了高速计数器的时钟、方向控制、复位和启动所使用的输入。这些输入功能描述见表 9-5 到 9-10。

表 9-3 高速计数器的指定输入

高速计数器	使用的输入
HSC0	I0.0, I0.1, 0.2
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

如表 9-4 所示，高速计数器和边沿中断的输入点分配存在一些重叠。同一个输入不能用于两个不同的功能，但是，不使用高速计数的输入端可以作它用。例如，如果 HSC0 工作于模式2，它使用 I0.0 和 I0.2，于是，I0.1 可以用于HSC3 的边沿中断。

如果 HSC0 的模式不使用输入 I0.1，那么该输入端可以用作 HSC3 或边沿中断。同样地，如果在选择的 HSC0 模式中不使用 I0.2，该输入端可以作边沿中断；如果在选择的 HSC4 模式中不使用 I0.4，该输入端可以用为 HSC5 所用。注意：HSC0 的所有模式都使用I0.0，HSC4的所有模式都使用I0.3。所以，当使用这些计数器时这些点不能作它用。

表 9-4 高速计数器和边沿中断的输入点分配

	输入点 (I)													
高速计数器	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	1.0	1.1	1.2	1.3	1.4	1.5
HSC0	x	x	x											
HSC1							x	x	x	x				
HSC2											x	x	x	x
HSC3		x												
HSC4				x	x	x								
HSC5					x									
边沿中断	x	x	x	x										

表 9-5 HSC0 操作模式 (CPU 221、CPU 222、CPU 224 和 CPU 226)

HSC0					
模式	描　　述	I0.0	I0.1	I0.2	
0	带内部方向控制的单相增/减计数器 SM37.3 = 0，减计数	时钟			
1	SM37.3 = 1，增计数			复位	
3	带外部方向控制的单相增/减计数器 I0.1 = 0，减计数	时钟	方向		
4	I0.1 = 1，增计数			复位	
6	带增减计数时钟输入的双相计数器	时钟 (增)	时钟 (减)		
7				复位	
9	A/B 相正交计数器 A 相超前 B 相 90 度，顺时针转动	时钟 A 相	时钟 B 相		
10	B 相超前 A 相 90 度，逆时针转动			复位	

表 9-6 HSC1 操作模式 (CPU 224 和 CPU 226)

HSC1					
模式	描　　述	I0.6	I0.7	I1.0	I1.1
0	带内部方向控制的单相增/减计数器	时钟			
1	SM47.3 = 0，减计数			复位	
2	SM47.3 = 1，增计数				启动
3	带外部方向控制的单相增/减计数器	时钟	方向		
4	I0.7 = 0，减计数			复位	
5	I0.7 = 1，增计数				启动
6	带增减计数时钟输入的双相计数器	时钟 (增)	时钟 (减)		
7				复位	
8					启动
9	A/B 相正交计数器	时钟 A 相	时钟 B 相		
10	A 相超前 B 相 90 度，顺时针转动			复位	
11	B 相超前 A 相 90 度，逆时针转动				启动

表 9-7 HSC2 操作模式 (CPU 224 和 CPU 226)

HSC2					
模式	描　　述	I1.2	I1.3	I1.4	I1.5
0	带内部方向控制的单相增/减计数器	时钟			
1	SM57.3 = 0，减计数			复位	
2	SM57.3 = 1，增计数				启动
3	带外部方向控制的单相增/减计数器	时钟	方向		
4	I1.3 = 0，减计数			复位	
5	I1.3 = 1，增计数				启动
6	带增减计数时钟输入的双相计数器	时钟 (增)	时钟 (减)		
7				复位	
8					启动
9	A/B 相正交计数器	时钟 A 相	时钟 B 相		
10	A 相超前 B 相 90 度，顺时针转动			复位	
11	B 相超前 A 相 90 度，逆时针转动				启动

表 9-8 HSC3 操作模式 (CPU 221、CPU 222、CPU 224 和 CPU 226)

HSC3					
模式	描　　述	I0.1			
0	带内部方向控制的单相增/减计数器 SM137.3 = 0，减计数 SM137.3 = 1，增计数	时钟			

表 9-9 HSC4 操作模式 (CPU 221、CPU 222、CPU 224 和 CPU 226)

HSC4					
模式	描　　述	I0.3	I0.4	I0.5	
0	带内部方向控制的单相增/减计数器 SM147.3 = 0，减计数	时钟			
1	SM147.3 = 1，增计数			复位	
3	带外部方向控制的单相增/减计数器 I0.4 = 0，减计数	时钟	方向		
4	I0.4 = 1，增计数			复位	
6	带增减计数时钟输入的双相计数器	时钟 (Up)	时钟 (Dn)		
7				复位	
9	A/B 相正交计数器 A 相超前 B 相 90 度，顺时针转动	时钟 A相	时钟 B 相		
10	B 相超前 A 相 90 度，逆时针转动			复位	

表 9-10 HSC5 操作模式 (CPU 221、CPU 222、CPU 224 和 CPU 226)

HSC5					
模式	描　　述	I0.4			
0	带内部方向控制的单相增/减计数器 SM157.3 = 0，减计数 SM157.3 = 1，增计数	时钟			

访问高速计数器 (HC)

存取高速计数器的计数值，必须指明高速计数器的地址，并采用 HC 类型和计数器号 (例如 HCS0)。高速计数器的当前值是只读的，并且只能用双字 (32位) 来寻址，如图 9–17 所示。

格式：　　HC *[高速计数器号]*　　HC2

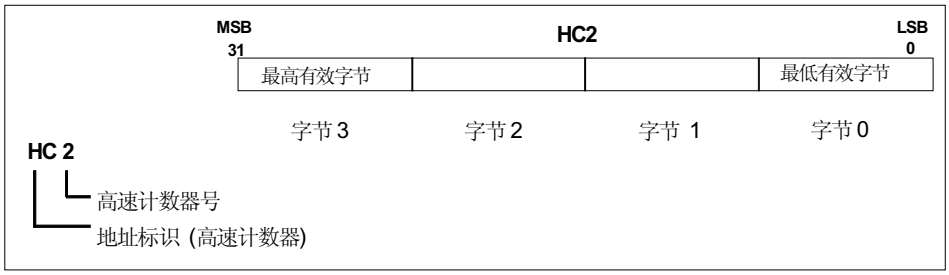


图 9-17 存取高速计数器的当前值

对高速计数器差异的理解

所有计数器在相同的工作模式下有相同的功能。如表 9-5 所示共有4种基本的计数模式。你可使用下列类型：无复位或启动输入，有复位无启动输入，或同时有复位和启动输入。

当激活复位输入，就清除当前计数值并保持到复位无效。当激活启动输入，就允许计数器计数。当启动输入无效时，计数器的当前值保持不变，时钟事件被忽略。如果在启动输入保持无效时，复位有效，则复位被忽略，当前值不变。如果在复位保持有效时，启动变为有效， 则计数器的当前值被清除。

使用高速计数器前，必须选定一种工作模式，你可以用 HDEF 指令 (定义高速计数器) 做到这件事。HDEF 给出了高速计数器 (HSCx) 和计数模式之间的联系。对每个高速计数器只能使用一条 HDEF 指令。可利用初次扫描存储器位SM0.1 (此位仅在第一次扫描周期时接通，然后断开) 调用一个包含 HDEF 指令的子程序来定义高速计数器。

选择有效状态和 1x /4x 模式

四个高速计数器有 3 个控制位，用来设置复位与启动输入的有效状态，以及选择1x 或 4x 计数方式 (只能是正交计数器)。这些位在每个计数器的控制字节中，只有在执行 HDEF 指令时才有用。这些位的定义见表 9-11。

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。缺省的设置为：复位和启动输入高电平有效，正交计数速率是 4x (4倍输入时钟频率)。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 9-11 复位、启动和 1x /4x 控制位的有效电平

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

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，包括下列几项：允许或禁止计数，计数方向控制 (只能是模式 0,1,2) 或对所有其它模式的初始化计数方向，要装入的计数器当前值和要装入的预置值。执行 HSC 指令时，要检验控制字节和有关的当前值及预置值。表 9-12 对这些控制位逐一做了说明。

表 9-12 HSC0、HSC1 和 HSC2 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中写入新的当前值： 0=不更新；1 =更新当前值
SM37.7	SM47.7	SM57.7	SM137.7	SM147.7	SM157.7	HSC 允许：0 = 禁止 HSC；1 = 允许 HSC

设定当前值和预置值

每个高速计数器都有一个 32 位的当前值和一个 32 位的预置值。当前值和预置值都是符号整数。为了向高速计数器装入新的当前值和预置值，必须先设置控制字节，并把当前值和 / 或预置值存入特殊存储器字节中，然后必须执行 HSC指令，从而将新的值送给高速计数器。表 9-13 对保存新的当前值和预置值的特殊存储器字节作了说明。

除了控制字节和新的预置值与当前值保存字节外，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) 后跟计数器号 (0，1，2，3，4 或 5) 的格式读出。因此，可用读操作直接访问当前值，但写操作只能用上述的 HSC 指令来实现。

表9-13 HSC0、HSC1、HSC2、HSC3、HSC4和 HSC5 的当前值和预置值

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

状态字节

每个高速计数器都有一个状态字节，其中某些位指出了当前计数方向，当前值是否等于预置值，当前值是否大于预置值。表 9-14 对每个高速计数器的状态位作了定义 。

表 9-14 HSC0、HSC1、HSC2、HSC3、HSC4 和 HSC5 的的状态位

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	描 述
SM36.0	SM46.0	SM56.0	SM136.0	SM146.0	SM156.0	不用
SM36.1	SM46.1	SM56.1	SM136.1	SM146.1	SM156.1	不用
SM36.2	SM46.2	SM56.2	SM136.2	SM146.2	SM156.2	不用
SM36.3	SM46.3	SM56.3	SM136.3	SM146.3	SM156.3	不用
SM36.4	SM46.4	SM56.4	SM136.4	SM146.4	SM156.4	不用
SM36.5	SM46.5	SM56.5	SM136.5	SM146.5	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

注:

只有执行高速计数器的中断程序时，状态位才有效。监视高速计数器的状态的目的是使外部事件可产生中断,以完成重要的操作。

HSC 中断

所有高速计数器支持中断条件：当前值等于预置时产生中断。 使用外部复位输入的计数器模式支持外部复位有效时产生的中断。除模式 0、1 和 2 外，所有的计数器模式支持计数方向改变的中断，每个中断条件可分别地被允许或禁止。为全面了解中断的用法，请参看 9.15 节。

注:

当使用外部复位中断时，不要写入一个新当前值或者在与那个事件有关的中断程序内禁止然后再允许高速计数器,否则,会造成一个致命错误。

为帮助你理解高速计数器的操作，提供了如下的初始化和操作顺序的描述，并以HSC1作为这些描述的计数器模型，即以 HSC1 为例。初始化描述假定 S7-200已置成 RUN 模式，由于这个原因，初次扫描存储器位为真 (SM0.1=1)。如果不是这种情况，记住在进入 RUN 模式后，对每个高速计数器的 HDEF 指令只能执行一次。对一个高速计数器执行第二个 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。

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
   写入新的当前值
   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. 用所希望的当前值装入 SMD48 (双字) 中，若装入 0，则清除 SMD48。
5. 用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 9.15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
   写入新的当前值
   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. 用所希望的当前值装入 SMD48 (双字) 中，若装入 0，则清除 SMD48。
5. 用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 9.15 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
   写入新的当前值
   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. 用所希望的当前值装入 SMD48 (双字) 中，若装入 0，则清除 SMD48。
5. 用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 9.15 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
   写入新的当前值
   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
   写入新的当前值
   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. 用所希望的当前值装入 SMD48 (双字) 中，若装入 0，则清除 SMD48。
5. 用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 9.15 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。

写入新的当前值 (任何模式下)

以下步骤描述了如何改变 HSC1 的当前值 (任何模式下) ：

在改变当前值时，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 写入新的当前值的控制位：

   SMB47 = 16#C0　　允许计数
   写入新的当前值

2. 向 SMD48 (双字) 写入所希望的当前值 (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。

虽然上面依次给出了如何单独改变计数方向、当前值和预置值，但实际上你可以在同一步中，通过对 SMB47 设置适当的值去改变所有的或其中的任意几个，然后执行 HSC 指令。

高速计数器举例

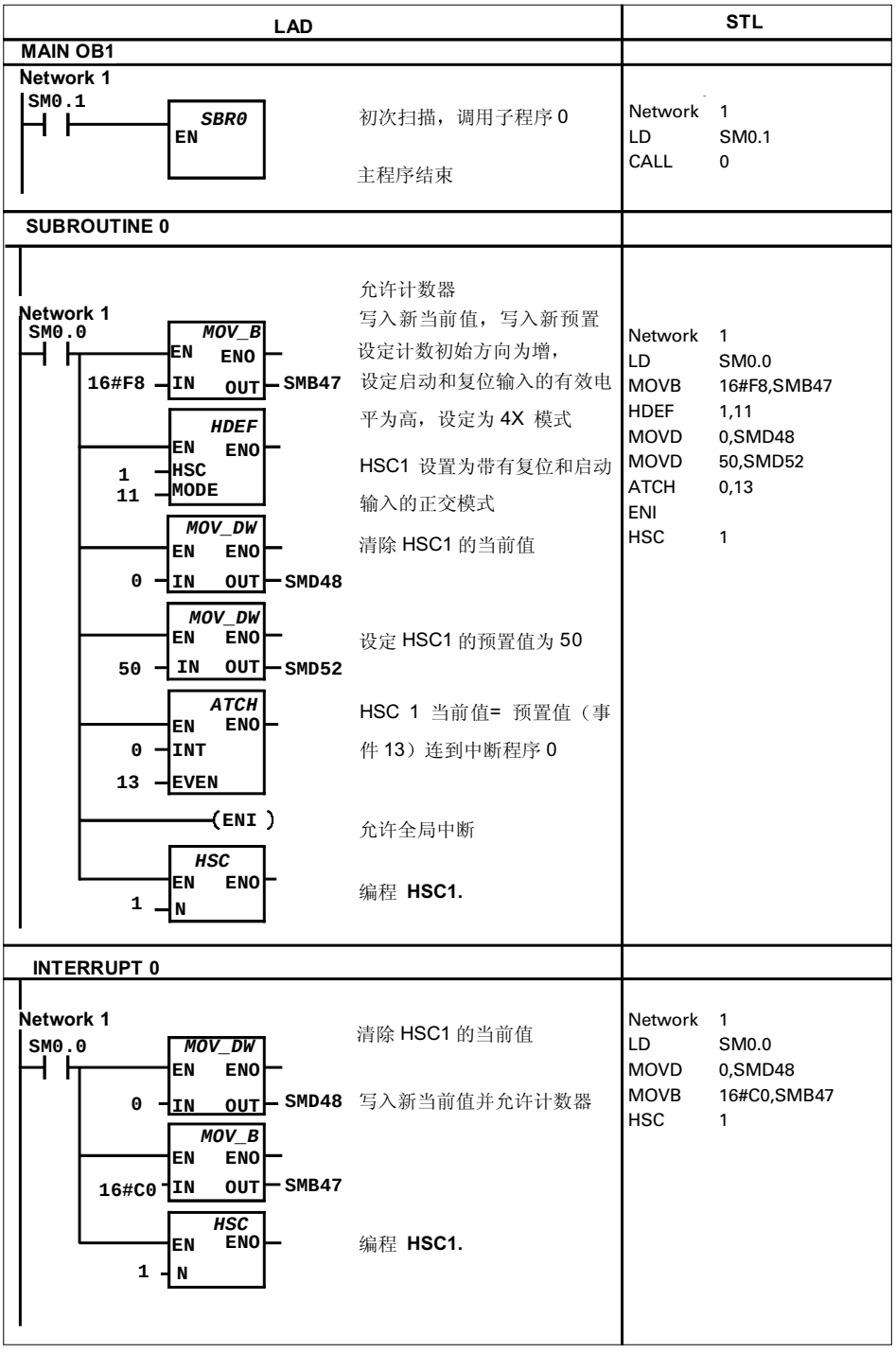


图 9-18 初始化 HSC1 的举例 (LAD 和 STL)

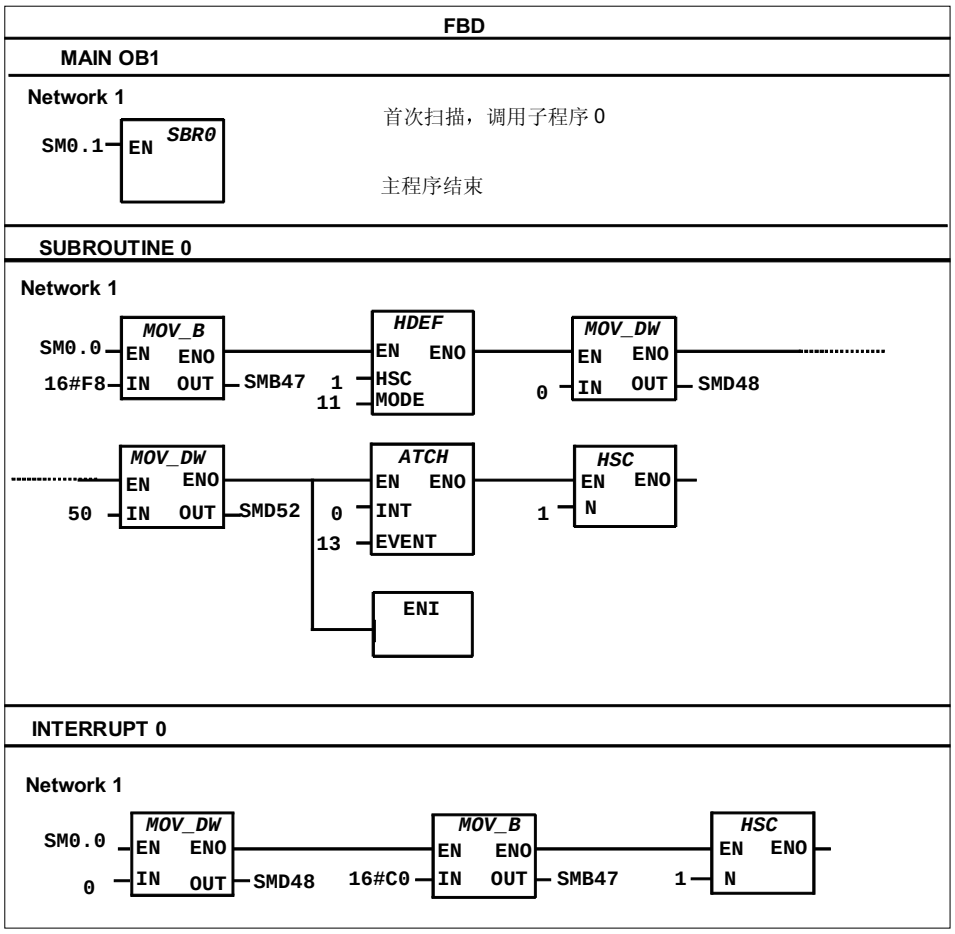
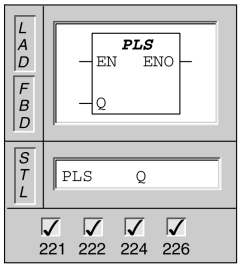


图 9-19　初始化 HSC1 的举例 (FBD)

脉冲输出



脉冲输出指令 (PLS) 检测为脉冲输出 (Q0.0 或Q0.1) 设置的特殊存储器位，然后激活由特殊存储器位定义的脉冲操作。

操作数：Q 常数 (0 或 1)

数据类型：字

脉冲输出范围：Q0.0 到 Q0.1

理解 S7-200 高速输出指令

每个 CPU 有两个 PTO/PWM 发生器产生高速脉冲串和脉冲宽度可调的波形。一个发生器分配在数字输出 Q0.0，另一个分配在数字输出 Q0.1。

PTO/PWM 发生器和映像寄存器共同使用 Q0.0 和 Q0.1。当 Q0.0 或 Q0.1设定为 PTO 或 PWM 功能时，PTO/PWM 发生器控制输出，在输出点禁止使用通用功能。映像寄存器的状态、输出强置或立即输出指令的执行都不影响输出波形。当不使用 PTO/PWM 发生器时，输出由映像寄存器控制。映像寄存器决定输出波形的初始和结束状态，以高电平或低电平产生波形的起始和结束。

注:

建议在允许 PTO 或 PWM 操作前把 Q0.0 和 Q0.1 的映像寄存器设定为 0。

脉冲串 (PTO) 功能提供方波 (50% 占空比) 输出，用户控制周期和脉冲数。脉冲宽度调制 (PWM) 功能提供连续、变占空比输出，用户控制周期和脉冲宽度。

每个 PTO/PWM 发生器有一个控制字节 (8位)，16 位无符号的周期时间值和脉宽值各一个，还有一个 32 位无符号的脉冲计数值。这些值全部存储在指定的特殊存储器中，一旦这些特殊存储器的位被置成所需操作，可通过执行脉冲指令 (PLS) 来调用这些操作。这条指令使 S7-200 读取特殊存储器中的位，并对相应的 PTO/PWM 发生器进行编程。

修改特殊寄存器(SM)区(包括控制字节)，然后执行PLS指令，可以改变PTO或PWM特性。

把 PTO/PWM 控制字节 (SM66.7 或 SM77.7) 的允许位置为 0，并执行 PLS指令，可以在任何时候禁止 PTO 或 PWM 波形的产生。

注:

所有控制字节、周期、脉冲宽度和脉冲数的缺省值都是 0。

注:

在 PTO/PWM 功能中，输出从 off 到 on 和从 on 到 off 的切换时间不一样。这种切换时间的差异表明了占空比的畸变。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。

PWM 操作

PWM 功能提供占空比可调的脉冲输出。周期和脉宽的增量单位为微秒 (μs) 或毫秒 (ms)。周期变化范围分别为 50～65,535 微秒或 2～65,535 毫秒。脉宽变化范围分别为 0～65,535 微秒或 0～65,535 毫秒。当脉宽大于等于周期时，占空比为 100%，即输出连续接通。当脉宽为 0 时，占空比为 0%，即输出断开。如果周期小于 2 个时间单位，那么周期时间被缺省地设定为 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。

PTO 操作

PTO 提供指定脉冲个数的方波 (50% 占空比) 脉冲串发生功能。周期可以用微秒或毫秒为单位指定。周期的范围是 50 到 65,535 微秒，或 2 到 65,535 毫秒。如果设定的周期是奇数，会引起占空比的一些失真。脉冲数的范围是：1 到 4,294,967,295。

如果周期时间少于 2 个时间单位，就把周期缺省地设定为 2 个时间单位。如果指定脉冲数为 0，就把脉冲数缺省地设定为 1 个脉冲。

状态字节中的 PTO 空闲位 (SM66.7 或 SM76.7) 用来指示可编程脉冲串完成。另外，根据脉冲串的完成调用中断程序 (有关中断和通讯指令的细节请见9.15 节)。如果使用多段操作，根据包络表的完成调用中断程序。请见下面的多段管线。

PTO 功能允许脉冲串的排队。当激活的脉冲串完成时，立即开始新脉冲的输出。这保证了顺序输出脉冲串的连续性。

有两种方法完成管线：单段管线或多段管线。

单段管线 在单段管线中，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，就必须立即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一直保持到第一个脉冲串发送完成。在管线中一次只能存一个入口，一旦第一个脉冲串发送完成，接着输出第二个波形，管线可以用于新的脉冲串。重复这个过程设定下一个脉冲串的特性：

除下面的情况外，脉冲串之间进行平滑转换：

- 如果发生了时间基准的改变。
- 如果在利用 PLS 指令捕捉到新脉冲串前启动的脉冲串已经完成。

当管线满时，如果试图装入管线，状态寄存器中的 PTO 溢出位 (SM66.6 或 SM76.6) 将置位。当 PLC 进入 RUN 状态时，这个位初始化为 0。如果要检测序列的溢出，必须在检测到溢出后手动清除这个位。

多段管线 在多段管线中，CPU 自动从 V 存储器区的包络表中读出每个脉冲串段的特性。在该模式下，仅使用特殊寄存器区的控制字节和状态字节。选择多段操作，必须装入包络表的起始 V 存储器区的偏移地址 (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一个基准，而且当包络执行时，不能改变。多段操作可以用 PLS 指令启动。

每段的长度是8个字节，由 16 位周期值、16 位周期增量值和 32 位脉冲计数值组成。

包络表的格式如表 9–15 所示。多段 PTO 操作的另一个特点是按照每个脉冲的个数自动增减周期的能力。在周期增量区输入一个正值将增加周期；输入一个负值将减小周期；输入 0 值将不改变周期。

如果在许多脉冲后指定的周期增量值导致非法周期值，会产生一个算术溢出错误，同时停止 PTO 功能，PLC 的输出变为由映像寄存器控制。另外，在状态字节中的增量计算错误位 (SM66.4 或 SM76.4) 被置为 1。

如果要人为地终止一个正进行中的 PTO 包络，只需要把状态字节中的用户终止位 (SM66.5 或 SM76.5) 置为1。

当 PTO 包络执行时，当前启动的段数目保存在 SMB166 (或 SMB176) 中。

表 9-15 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)； 数 0 产生一个非致命性错误，将不产生 PTO 输出。
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)
: :	: :	: :

计算包络表值

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

图 9-20 的例子说明了如何生成包络表值，按要求产生输出波形加速电机、恒速运行，然后减速电机。

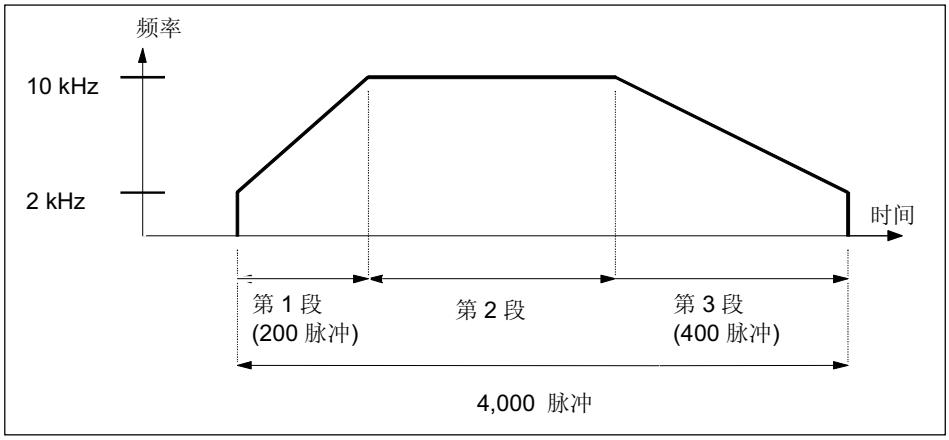


图 9-20 步进电机应用的频率-时间图举例

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 10 kHz。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，启动和结束的周期是500us，最大频率对应的周期是100us。

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

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 9-16 给出了产生所要求波形的值。

表 9-16 包络表值

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。图 9-23 是采用多段 PTO 操作的程序指令举例。

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 (除非周期增量是0)。 如果需要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

这两项的影响是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

PTO/PWM 控制寄存器

表 9–17 是控制 PTO/PWM 操作的寄存器，利用表 9-18 可以作为快速参考，确定放入 PTO/PWM 控制寄存器中的值，启动要求的操作。对 PTO/PWM 0 使用 SMB67，对 PTO/PWM 1 使用 SMB77。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 9-17 PTO /PWM 控制寄存器

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 = 无上溢； 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 = 1 祇/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 PTO/PWM; 1 = 允许 PTO/PWM
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 9-18 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		1us/ 周期			装入
16#84	Yes	PTO	单段		1us/ 周期	装入		
16#85	Yes	PTO	单段		1us/ 周期	装入		装入
16#89	Yes	PTO	单段		1 ms/ 周期			装入
16#8C	Yes	PTO	单段		1 ms/ 周期	装入		
16#8D	Yes	PTO	单段		1 ms/ 周期	装入		装入
16#A0	Yes	PTO	多段		1us/ 周期			
16#A8	Yes	PTO	多段		1 ms/ 周期			
16#D1	Yes	PWM		同步	1us/ 周期			装入
16#D2	Yes	PWM		同步	1us/ 周期		装入	
16#D3	Yes	PWM		同步	1us/ 周期		装入	装入
16#D9	Yes	PWM		同步	1 ms/ 周期			装入
16#DA	Yes	PWM		同步	1 ms/ 周期		装入	
16#DB	Yes	PWM		同步	1 ms/ 周期		装入	装入

PTO/PWM 初始化和操作顺序

PTO/PWM 的初始化和操作步骤说明如下。它可帮助你更好地理解 PTO 和PWM 功能的操作，这些步骤的说明使用了输出 Q0.0。初始化操作假定 S7-200 已置成 RUN 模式，因此初次扫描存储器位为真 (SM0.1=1)。如果不是这种情况，或 PTO/PWM 必须重新初始化，你可以用一个条件 (不一定是初次扫描存储器位)来调用初始化程序。

PWM 初始化

把 Q0.0 初始化成 PWM，应遵循以下步骤：

1. 用初次扫描存储器位 (SM0.1) 设置输出为1，并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，后续扫描就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，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。
3. 向 SMW68 (字) 写入所希望的周期值。
4. 向 SMW70 (字) 写入所希望的脉宽。
5. 执行 PLS 指令，以使 S7-200 对 PTO/PWM 发生器 编程。
6. 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。
7. 退出子程序。

修改 PWM 输出的脉冲宽度

为了在子程序中改变 PWM 输出的脉宽，请遵循如下步骤：(假定SMB67中装入16#D2 或 16#DA.)

1. 调用一个子程序，以把所需脉宽装入 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。

PTO 初始化- 单段操作

为了初始化 PTO，请遵循如下步骤：

1. 用初次扫描存储器位 (SM0.1) 复位输出为 0，并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，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。
3. 向 SMW68 (字) 写入所希望的周期值。
4. 向 SMD72 (双字) 写入所希望的脉冲计数。
5. 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 9.16 节中断指令，以了解中断处理的详细内容。
6. 执行 PLS 指令，使 S7-200 对 PTO/PWM 发生器编程。
7. 退出子程序。

修改 PTO 周期 - 单段操作

当使用单段PTO操作时，为了在中断程序中或子程序中改变PTO周期，请遵循如下步骤：

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。

修改 PTO 脉冲数 - 单段操作

当使用单段 PTO 操作时，为了在中断程序中或子程序中改变 PTO 脉冲计数，请遵循如下步骤：

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，和设置更新脉冲计数。
2. 向 SMD72 (双字) 写入所希望的脉冲计数。
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。

修改 PTO 周期和脉冲数 - 单段操作

当使用单段 PTO 操作时，为了在中断程序中或子程序中改变 PTO 的周期和脉冲计数，请遵循如下步骤：

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，设置更新周期和脉冲计数。
2. 向 SMW68 (字) 写入所希望的周期值。
3. 向 SMD72 (双字) 写入所希望的脉冲计数。
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。

PTO 初始化 - 多段操作

为了初始化 PTO，请遵循如下步骤：

1. 用初次扫描存储器位 (SM0.1) 复位输出为 0，并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，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，设置更新脉冲计数和周期值。
3. 向 SMW168 (字) 写入包络表的起始V存储器偏移值。
4. 在包络表中设定段数，确保段数区 (表的第一个字节) 正确。

5. 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 9.16 节中断指令，以了解中断处理的详细内容。
6. 执行 PLS 指令，使 S7–200 对 PTO/PWM 发生器编程。
7. 退出子程序。

PWM 举例

图 9-21 是脉冲宽度调制 (PWM) 的实例。

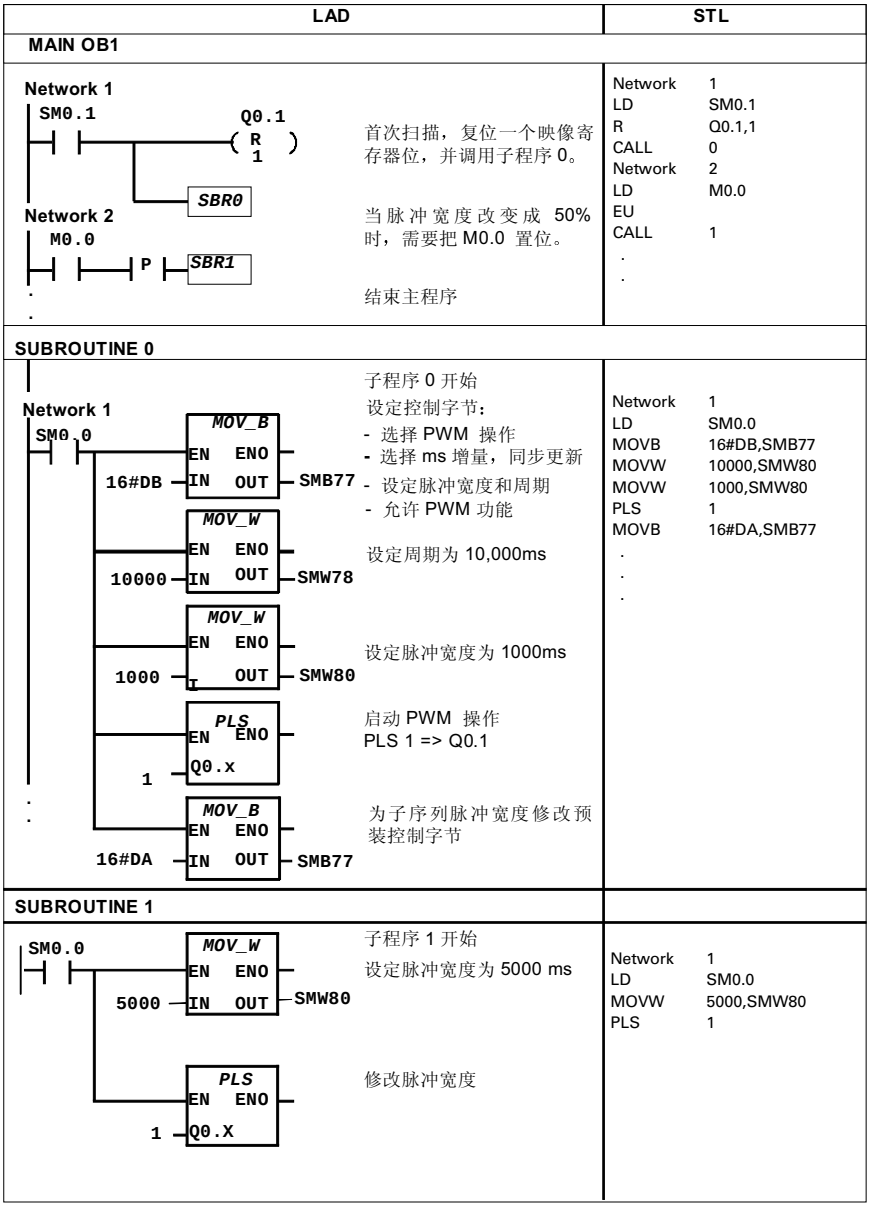


图 9-21 使用 PWM 的高速输出实例

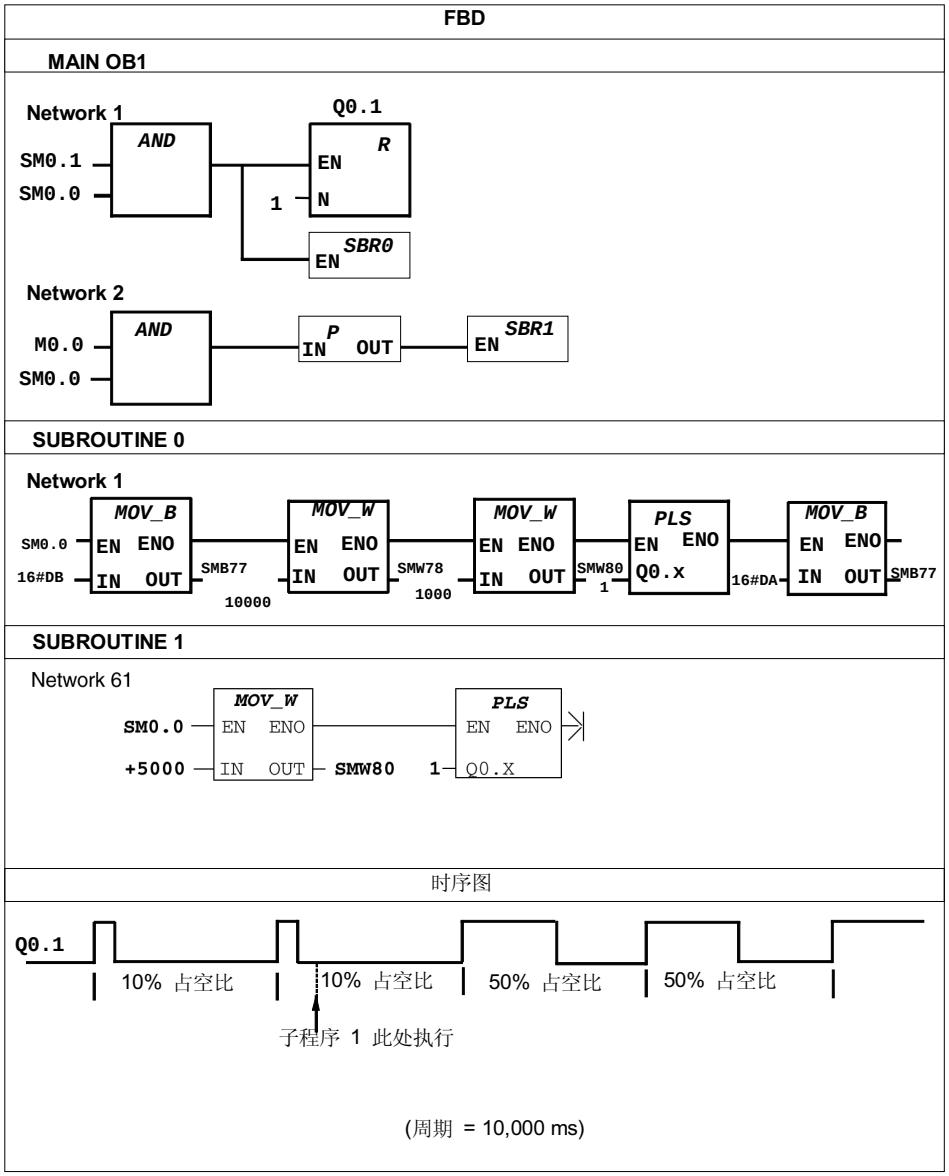


图 9-21 使用 PWM 的高速输出实例(续)

单段操作脉冲串实例

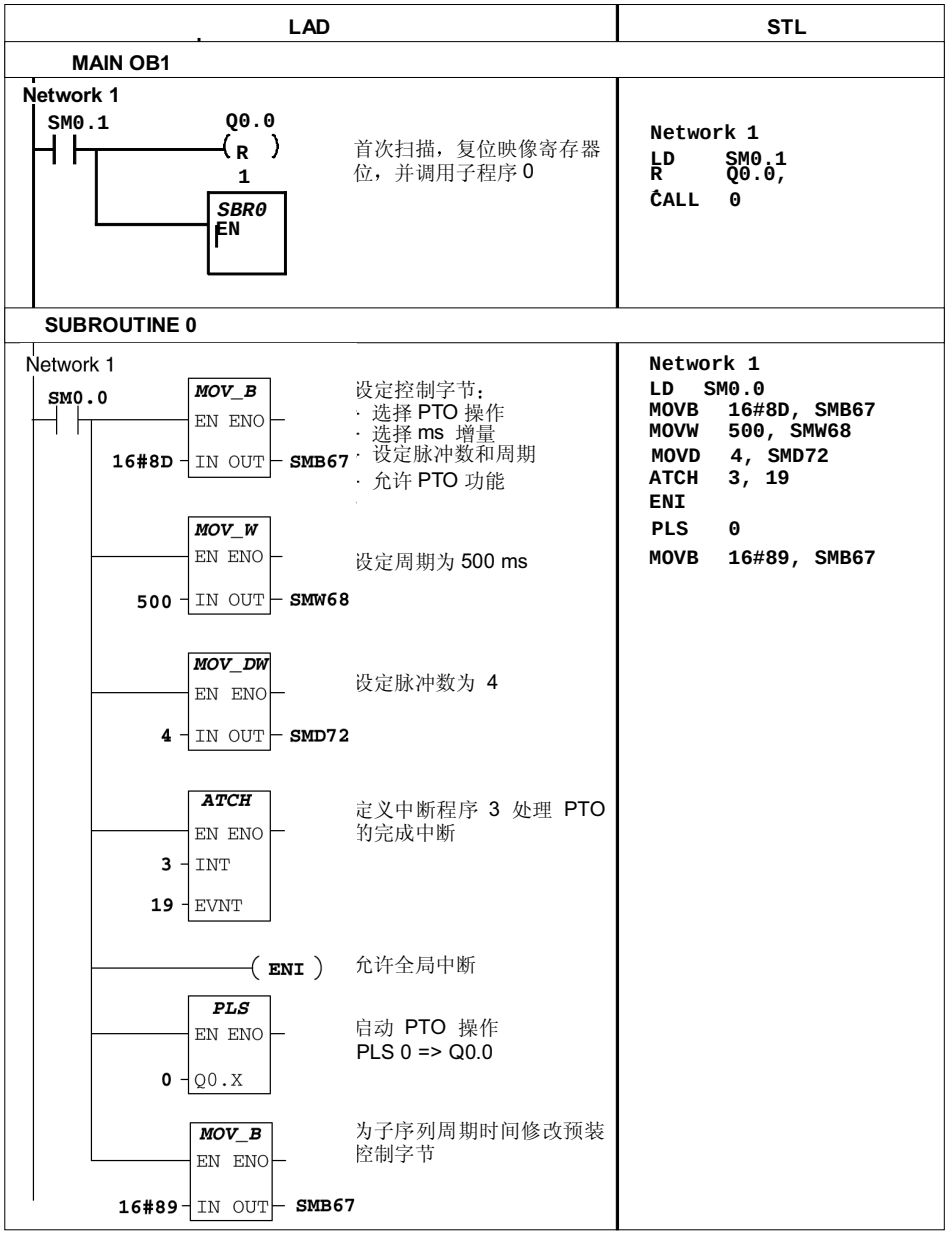


图 9-22　在 SM 存储器中使用单段操作的脉冲串输出实例

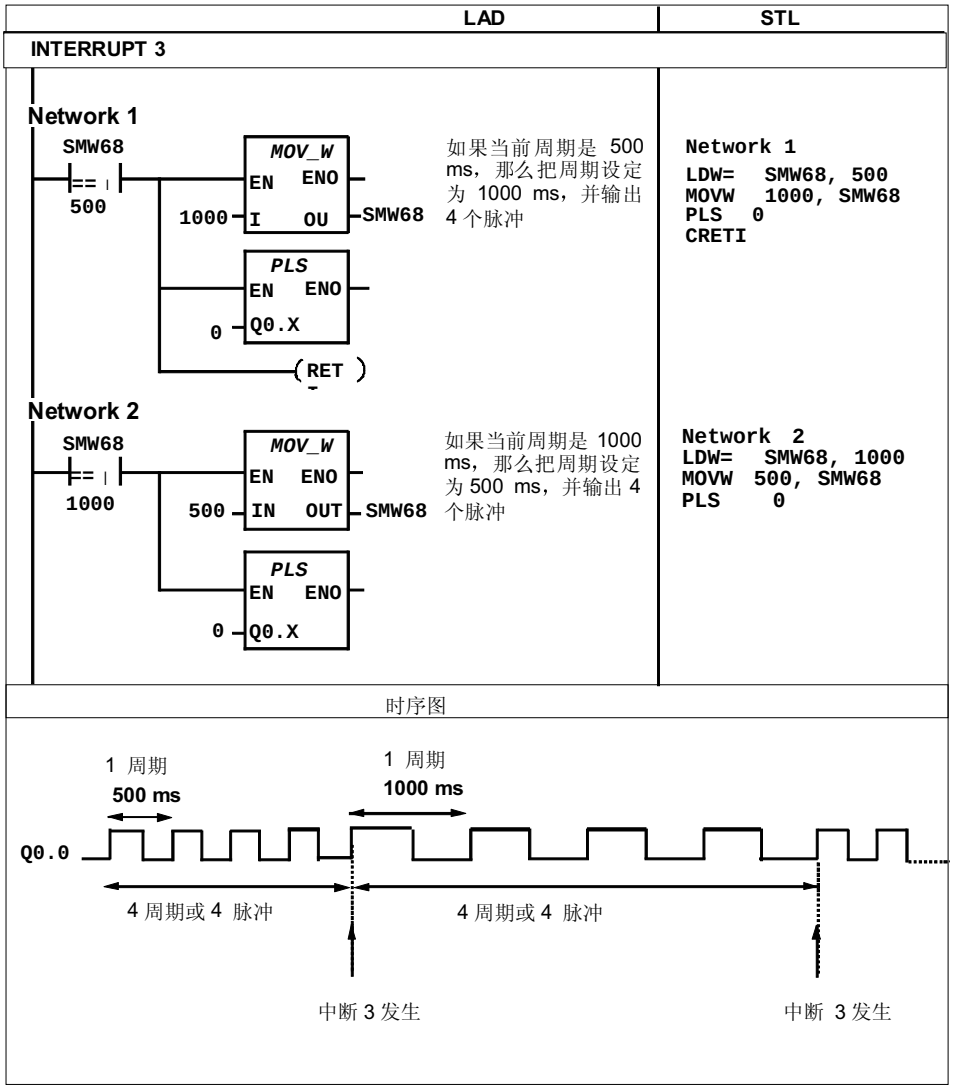


图 9-22 在 SM 存储器中使用单段操作的脉冲串输出实例 (续)

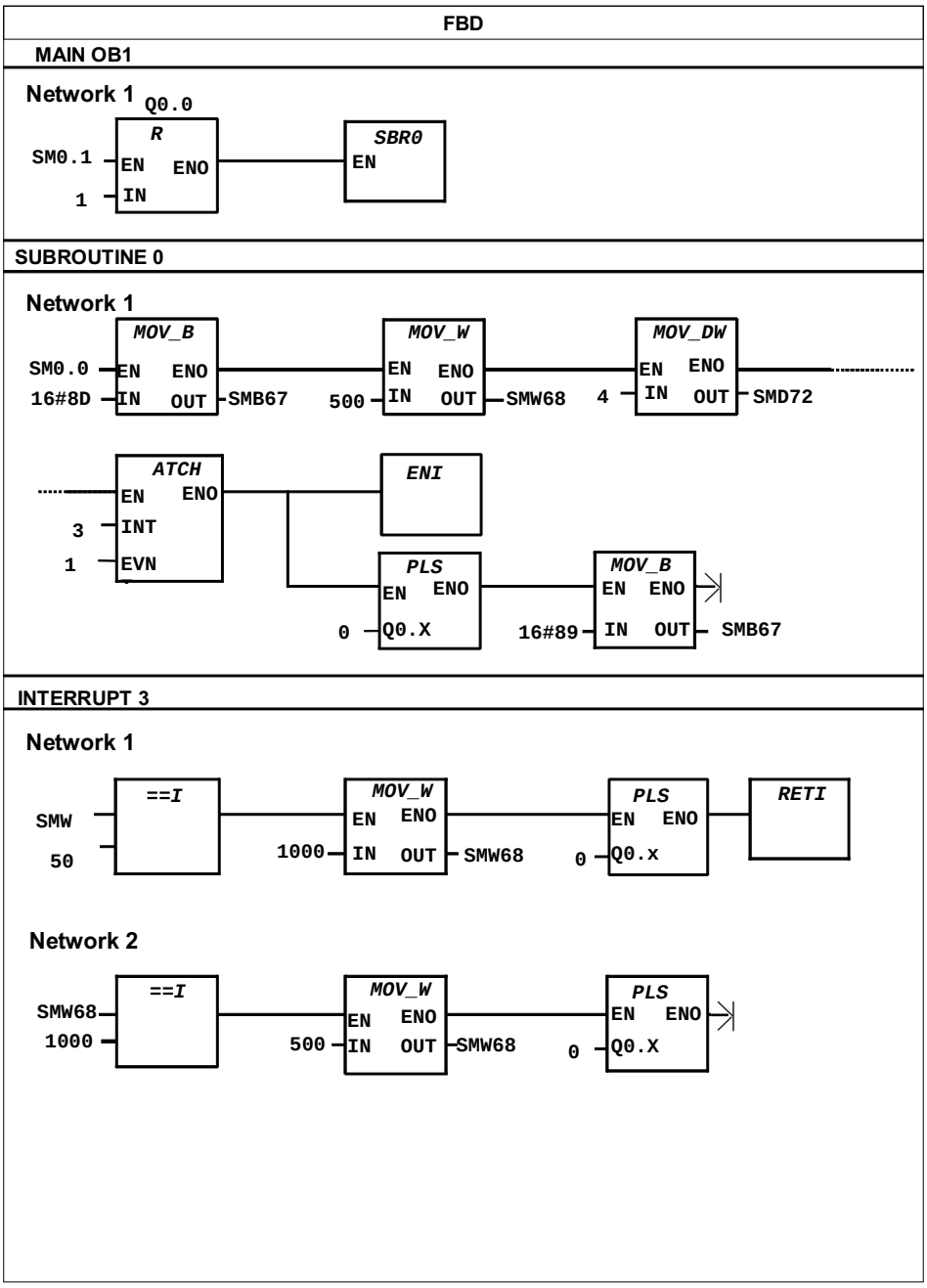


图 9-22　在 SM 存储器中使用单段操作的脉冲串输出实例 (续)

多段操作脉冲串输出实例

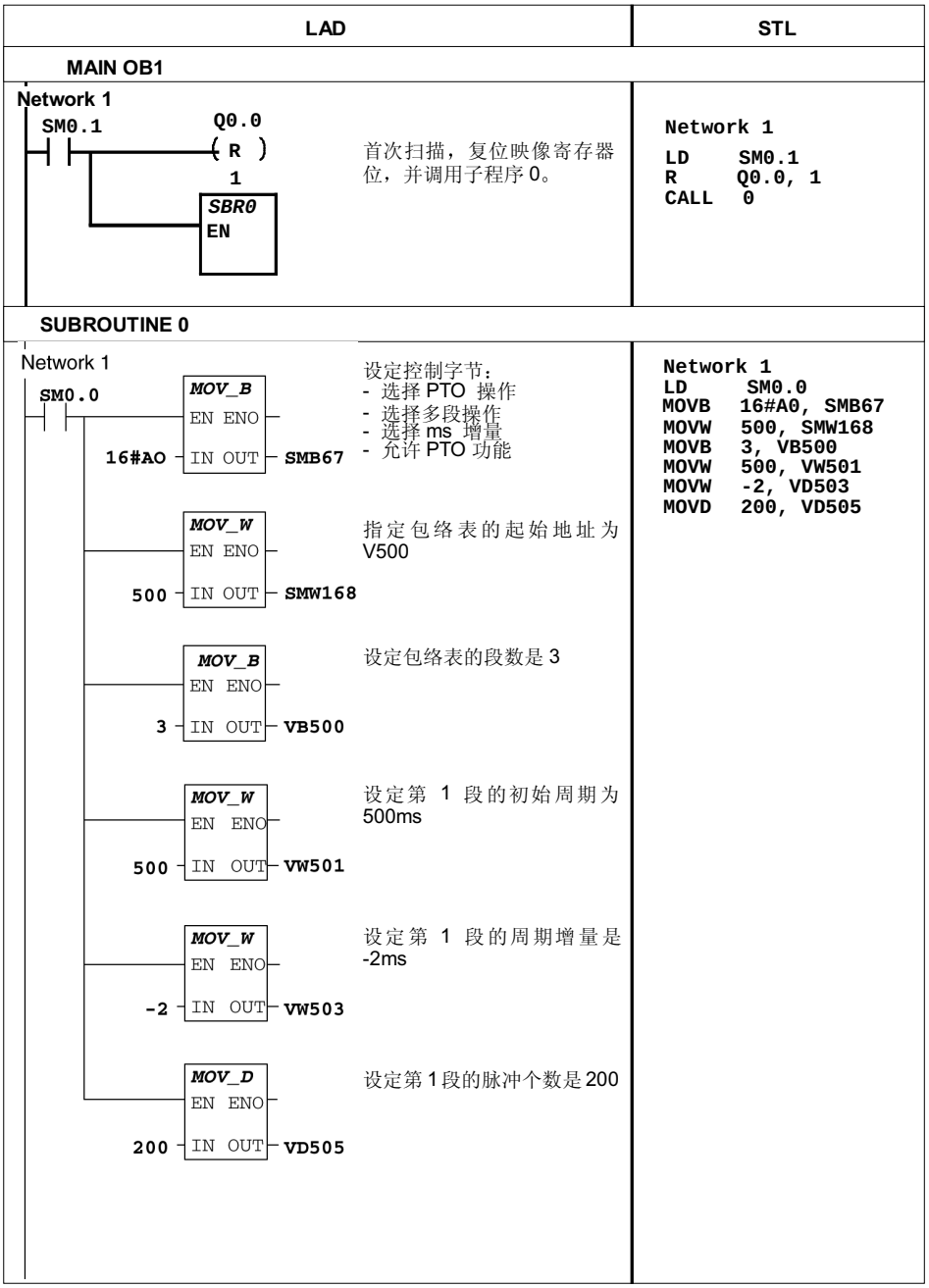


图 9-23　使用多段操作的脉冲串输出实例

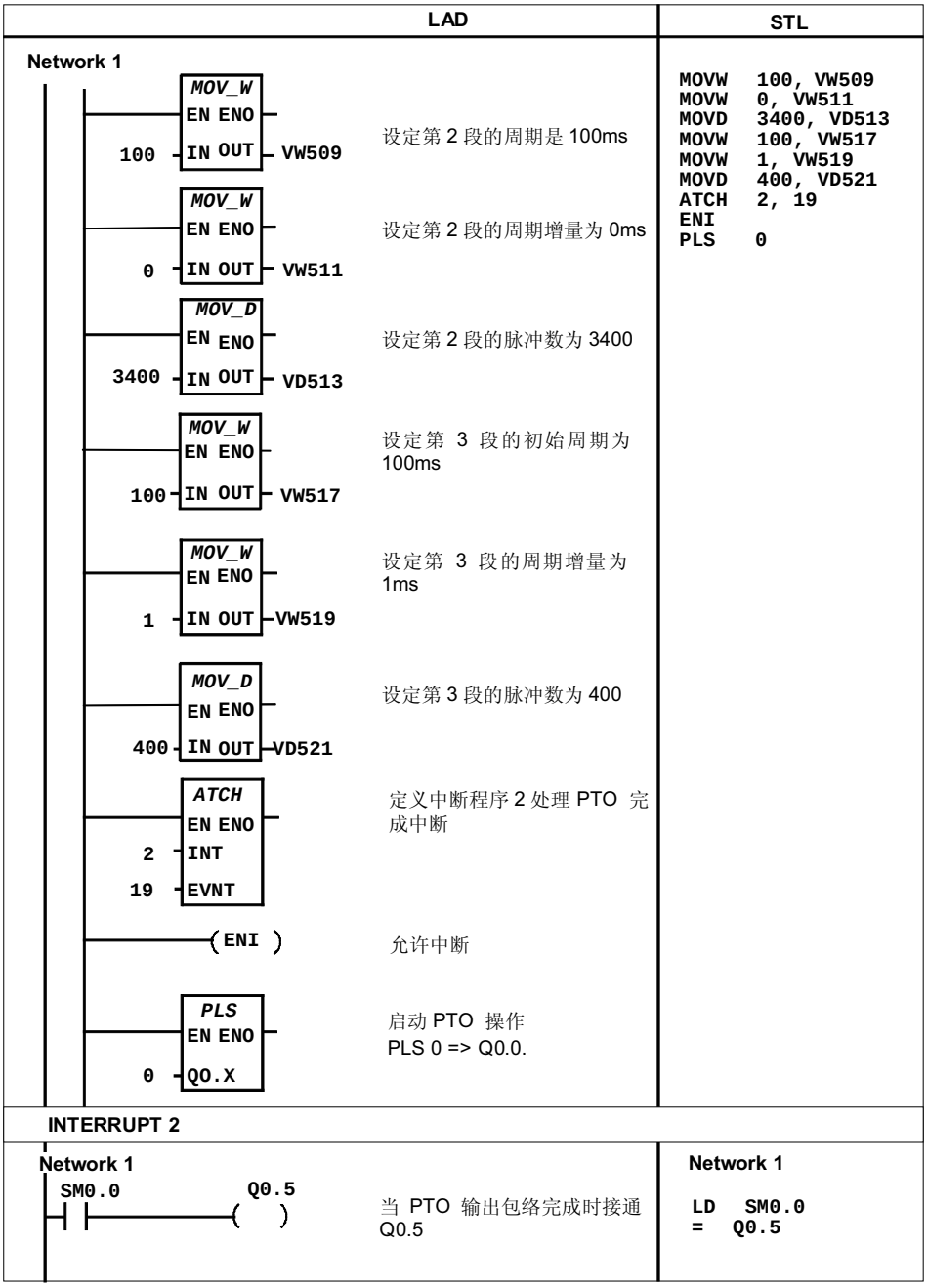


图 9-23　使用多段操作的脉冲串输出实例 (续)

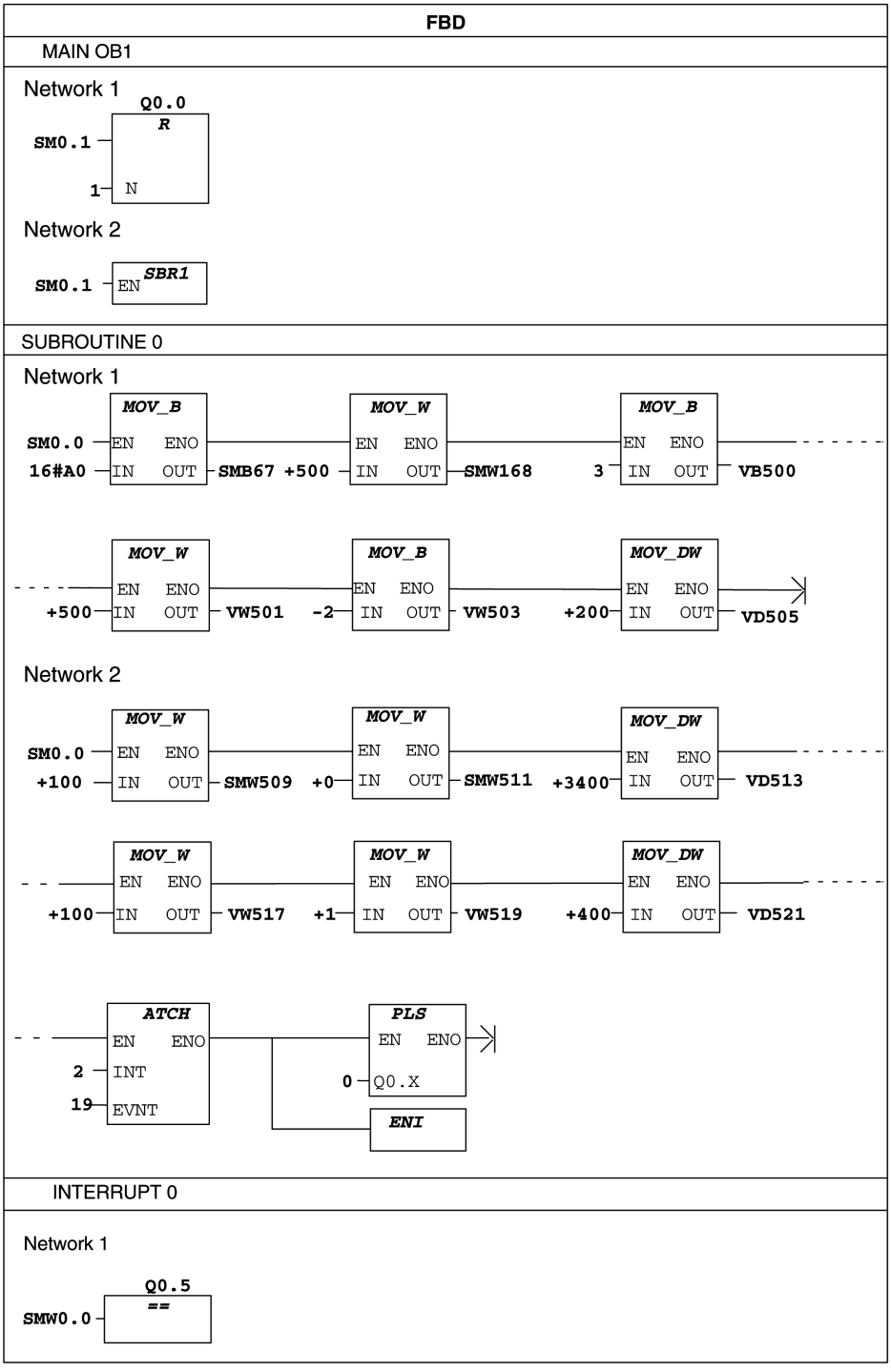
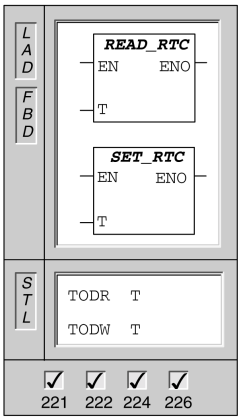


图 9-23　使用多段操作的脉冲串输出实例 (续)

9.5 SIMATIC 时钟指令

读实时时钟，设定实时时钟



读实时时钟指令读当前时间和日期并把它装入一个8 字节的缓冲区 (起始地址是T)。

设定实时时钟指令写当前时间和日期并把8字节缓冲区 (起始地址是T) 装入时钟。

在语句表中，TODR 和 TODW 指令表示读时钟 (TODR) 和写时钟 (TODW)。

TODR: 使 ENO = 0 的出错条件：
SM4.3 (运行时间)， 0006 (间接寻址)，
000C (时钟模块不存在)

TODW: 使 ENO = 0 的出错条件：
SM 4.3 (运行时间)， 0006 (间接寻址)，
0007 (TOD 数据错误)， 000C (时钟模块不存在)

输入/输出	操 作 数	数据类型
T	VB, IB, QB, MB, SMB, SB, LB, *VD, *AC, *LD	BYTE

9–24 是时钟缓冲器(T)的格式。

T	T+1	T+2	T+3	T+4	T+5	T+6	T+7
年	月	日	小时	分钟	秒	0	星期几

图 9-24 定时缓冲器的格式

当扩展电源停电或存储器丢失时，实时时钟初始化下面的数据和时间：

日期： 01-Jan-90 (90年1月1日)
时间： 00:00:00
星期 Sunday (星期日)

S7-200 中的日期、时间时钟只用年份的最低两位表示年份，因此 2000 年会表示成 00 (它将从 99 变为 00)。

必须用 BCD 码表示所有的日期和时间值 (例 16#97 表示 1997 年)。数据格式如下：

年/月	yymm	yy -	0 to 99	mm -	1 to 12
日/时	ddhh	dd -	1 to 31	hh -	0 to 23
分/秒	mmss	mm -	0 to 59	ss -	0 to 59
星期	d	d -	0 to 7	1 =	Sunday (星期日)
				0 =	禁用星期 (保持0)

注意：

S7-200 CPU 不执行检查和核实日期与星期待几是否符合。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。

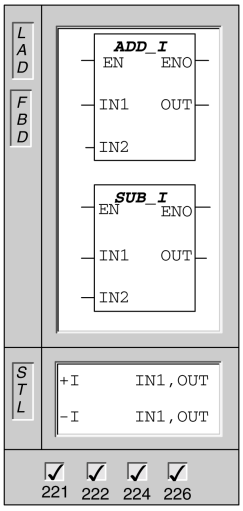
不要同时在主程序和中断程序中使用 TODR/TODW 指令。如果这样做，而在执行TOD 指令时出现了执行 TOD 指令的中断，则中断程序中的 TOD 指令不会被子执行。SM4.3 批示了试图对时钟进行两个同时的访问（非致命错误 0007）。

S7-200PLC 不使用任何形式的年信息，不会受到世纪跨越的影响。但是，用到年份进行算术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。

通过2096 年进行闰年的正确处理。

9.6 SIMATIC 整数数学运算指令

整数加法和整数减法



整数的加法和减法指令把两个 16 位整数相加或相减，产生一个 16 位结果 (OUT)。

在 LAD 和 FBD 中：IN1 + IN2 = OUT

IN1 - IN2 = OUT

在 STL 中：IN1 + OUT = OUT

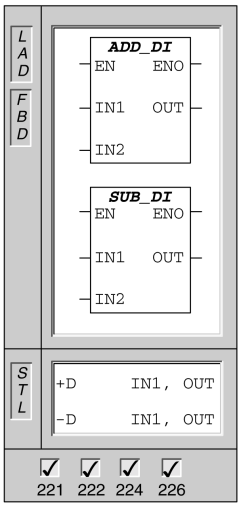
OUT - IN1 = OUT

使 ENO = 0 的错误条件是：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位：SM1.0 (零)；SM1.1 (溢出)；SM1.2 (负)

输入/输出	操 作 数	数据类型
IN1, IN2	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, 常数, *VD, *AC, *LD	INT
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *AC, *LD	INT

双整数加法和双整数减法



双整数的加法和减法指令把两个 32 位双整数相加或相减，产生一个 32 位结果 (OUT)。

在 LAD 和 FBD 中：IN1 + IN2 = OUT

IN1 - IN2 = OUT

在 STL 中： IN1 + OUT = OUT

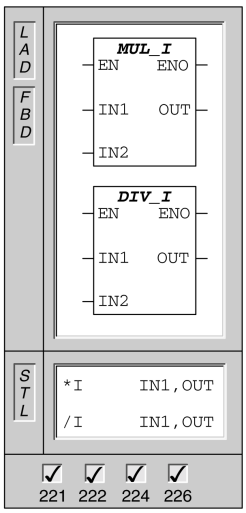
OUT - IN1 = OUT

使 ENO = 0 的错误条件是：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位：SM1.0 (零) ；SM1.1 (溢出) ；SM1.2 (负)

输入/输出	操 作 数	数据类型
IN1, IN2	VD, ID, QD, MD, SMD, SD, LD, AC, HC, 常数, *VD, *AC, *LD	DINT
OUT	VD, ID, QD, MD, SM, SD, LD, AC, *VD, *AC, *LD	DINT

整数乘法和整数除法



整数乘法指令把两个16位整数相乘，产生一个16位乘积。

整数除法指令把两个16位整数相除，产生一个16位商，不保留余数。

如果结果大于一个字，就置位溢出位。

在 LAD 和 FBD 中： IN1 * IN2 = OUT

IN1 / IN2 = OUT

在 STL 中： IN1 * OUT = OUT

OUT / IN1 = OUT

使 ENO = 0 的错误条件是：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)

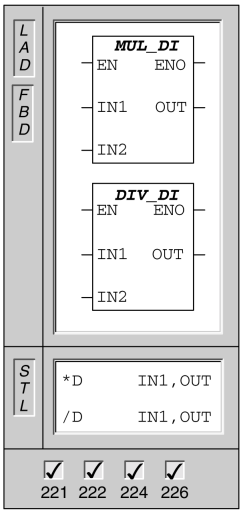
这些指令影响下面的特殊存储器位：SM1.0 (零) ；SM1.1 (溢出) ；SM1.2 (负) ；SM1.3 (被 0 除)

如果在乘或除的操作过程中 SM1.1 (溢出) 被置位，就不写到输出，并且所有其它的算术状态位被置为 0。

如果在除法操作的时候SM1.3(被0除)被置位，其它的算术状态位保留不变，原始输入操作数不变化。否则，所有有关的算术状态位是算术操作的有效状态。

输入/输出	操 作 数	数据类型
IN1, IN2	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, 常数, *VD, *AC, *LD	INT
OUT	VW, QW, IW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC	INT

双整数乘法和双整数除法



整数乘法指令把两个32位双整数相乘，产生一个32位乘积。

双整数除法指令把两个32位双整数相除，产生一个32位商，不保留余数。

在 LAD 和 FBD 中：IN1 * IN2 = OUT

IN1 / IN2 = OUT

在 STL 中：IN1 * OUT = OUT

OUT / IN1 = OUT

使 ENO = 0 的错误条件是：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)

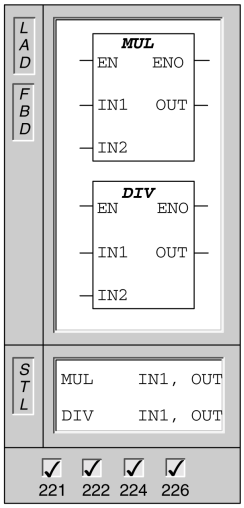
这些指令影响下面的特殊存储器位：SM1.0 (零)；SM1.1 (溢出)；SM1.2 (负)；SM1.3 (被 0 除)

如果在乘或除的操作过程中 SM1.1 (溢出) 被置位，就不写到输出，并且所有其它的算术状态位被置为 0。

如果在除法操作的时候 SM1.3 (被 0 除) 被置位，其它的算术状态位保留不变，原始输入操作数不变化。否则，所有有关的算术状态位是算术操作的有效状态。

输入/输出	操 作 数	数据类型
IN1, IN2	VD, ID, QD, MD, SMD, SD, LD, HC, AC, 常数, *VD, *AC, *LD	DINT
OUT	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC	DINT

整数乘法产生双整数和整数除法产生双整数



整数乘法产生双整数指令把两个16位整数相乘，产生一个32位积。

整数除法产生双整数指令把两个16位整数相除，产生一个32位结果，其中16位是余数(最高有效位)，16位是商(最低有效位)。

在STL乘法指令中，32位结果的最低有效位(16位)被用作乘数。

在STL乘法指令中，32位结果的最低有效位(16位)被用作被除数。

在 LAD 和 FBD 中：IN1 * IN2 = OUT

IN1 / IN2 = OUT

在 STL 中：IN1 * OUT = OUT

OUT / IN1 = OUT

使 ENO = 0 的错误条件是：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位：SM1.0 (零)；SM1.1 (溢出)；SM1.2 (负)；SM1.3 (被 0 除)

如果在除法操作的时候 SM1.3 (被 0 除) 被置位，其它的算术状态位保留不变，原始输入操作数不变化。否则，所有有关的算术状态位是算术操作的有效状态。

输入/输出	操 作 数	数据类型
IN1, IN2	VW, IW, QW, MW, SW, SMW, LW, AC, AIW, T, C, 常数, *VD, *AC, *LD	INT
OUT	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC	DINT

算术运算举例

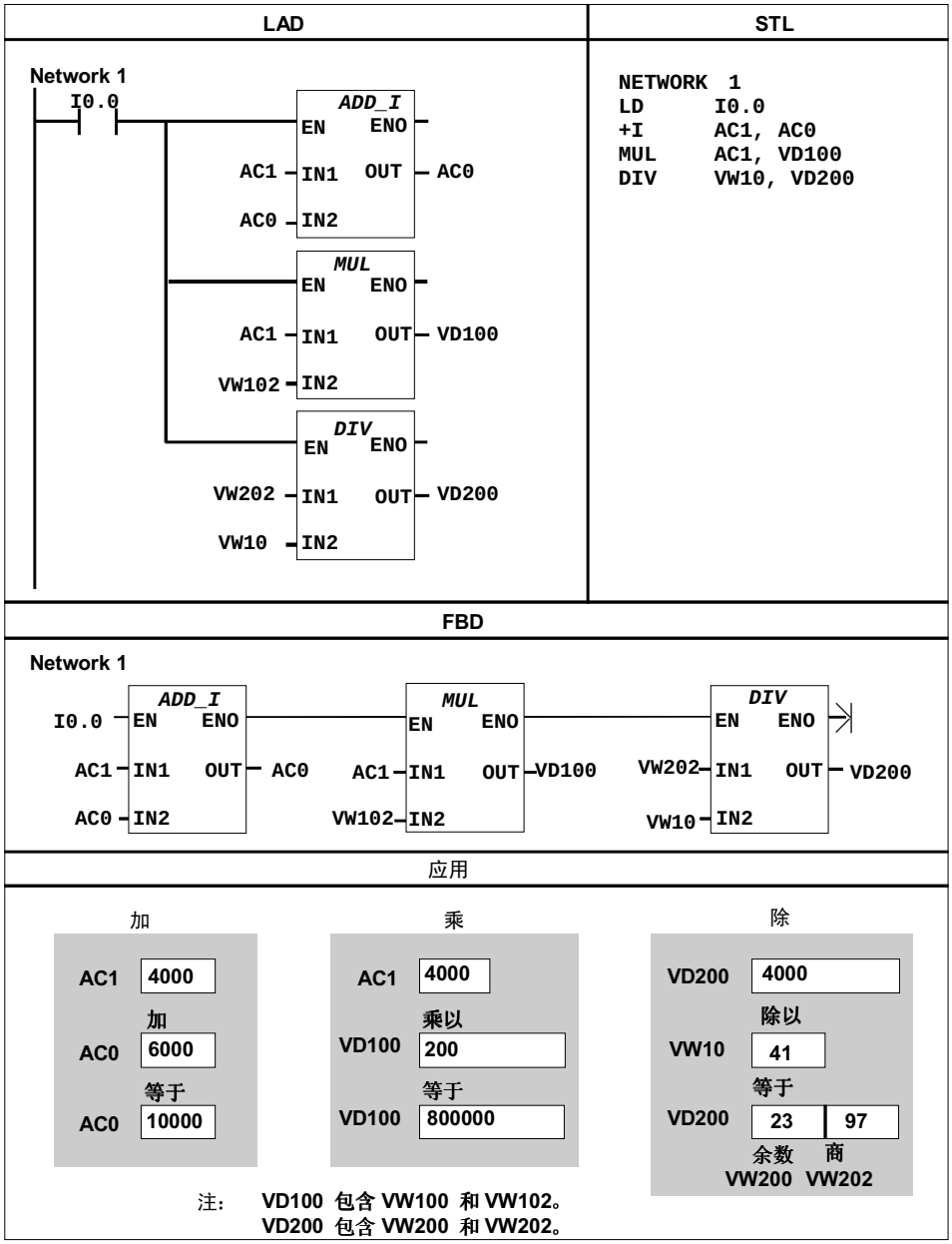
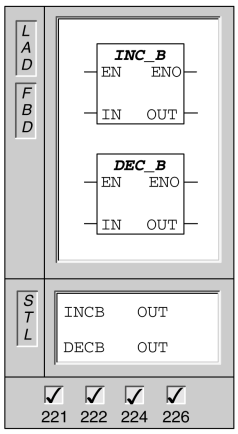


图 9-25 整数算术运算指令的 LAD，STL 和 FBD 举例

字节增和字节减



字节增 (INCB) 或**字节减** (DECB) 指令把输入字节 (IN) 加 1 或减 1，并把结果存放到输出单元 (OUT)。

字节增减指令是无符号的。

在 LAD 和 FBD 中： IN + 1 = OUT

IN - 1 = OUT

在STL中： OUT+ 1 = OUT

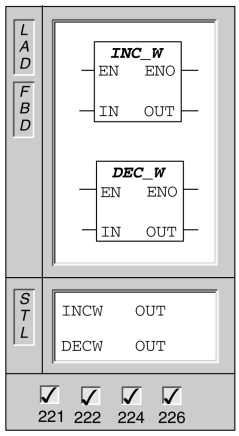
OUT - 1 = OUT

使 ENO = 0 的错误条件是：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位：SM1.0 (零)；SM1.1 (溢出)

输入/输出	操 作 数	数据类型
IN	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD	BYTE
OUT	VB, IB, QB, MB, SB, SMB, LB, AC,*VD, *AC, *LD	BYTE

字增和字减



字增 (INCW) 或**字减** (DECW) 指令把输入字 (IN) 加1或减1，并把结果存放到输出单元 (OUT)。

字增减指令是有符号的 (16#7FFF > 16#8000)。

在 LAD和 FBD 中： IN + 1 = OUT

IN - 1 = OUT

在STL中： OUT+ 1 = OUT

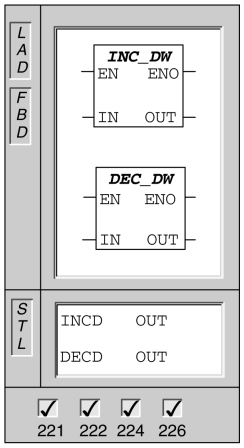
OUT - 1 = OUT

使 ENO = 0 的错误条件是：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位：SM1.0 (零)；SM1.1 (溢出)；SM1.2 (负)

输入/输出	操 作 数	数据类型
IN	V W, IW, QW, MW, SW, SMW, AC, AIW, LW, T, C, 常数, *VD, *AC,*LD	INT
OUT	VW, IW, QW, MW, SW, SMW, LW, AC, T, C, *VD, *AC, *LD	INT

双字增和双字减



字增或双字减指令把输入字 (IN) 加1或减1，并把结果存放到输出单元 (OUT)。

在 LAD 和 FBD 中： IN + 1 = OUT

IN - 1 = OUT

双字增减指令是有符号的 (16#7FFFFFFF > 16#80000000)。

在 STL中： OUT + 1 = OUT

OUT - 1 = OUT

使 ENO = 0 的错误条件是：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位SM1.0(零)；SM1.1 (溢出)；SM1.2 (负)

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SD, SMD, LD, AC, HC, 常数, *VD, *AC, *LD	DINT
OUT	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD	DINT

增减举例

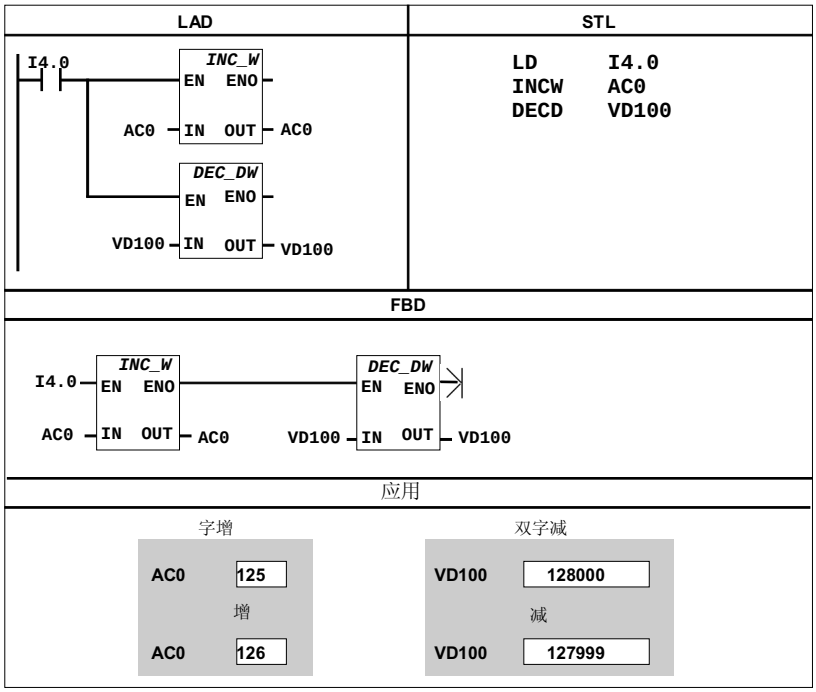
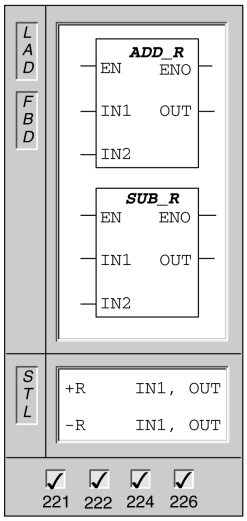


图 9-26 增/减指令的 LAD，STL 和 FBD 举例

9.7 SIMATIC 实数数学运算指令

实数的加减



实数的加减指令把两个 32 位实数相加或相减，得到32 位实数结果 (OUT)。

在 LAD 和 FBD 中： IN1 + IN2 = OUT

IN1 - IN2 = OUT

在 STL 中： IN1 + OUT = OUT

OUT - IN1 = OUT

使 ENO = 0 的错误条件是：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位：SM1.0 (零) ；SM1.1 (溢出) ；SM1.2 (负)

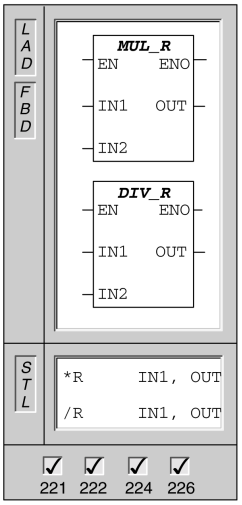
SM1.1 用来指示溢出错误和非法值。如果 SM1.1 置位，SM1.0 和 SM1.2 的状态就无效，原始操作数不改变。如果 SM1.1 不置位，SM1.0 和 SM1.2 的状态反映算术操作的结果。

输入/输出	操 作 数	数据类型
IN1, IN2	VD, ID, QD, MD, SD, SMD, AC, LD, 常数, *VD, *AC, *LD	REAL
OUT	VD, ID, QD, MD, SD, SMD, AC, LD, *VD, *AC, *LD	REAL

注意：

实数或浮点数在 ANSI/IEEE 754 1958 标准（单精度）中有描述，若要得到更多信息，请参阅该标准。

实数的乘、除



实数的乘法指令把两个 32 位实数相乘，产生 32 位实数结果 (OUT)。

实数的除法指令把两个32位实数相除，得到32位实数商。

在 LAD 和 FBD 中： IN1*IN2 = OUT

IN1/ IN2 = OUT

在 STL 中： IN1*OUT = OUT

OUT / IN1 = OUT

使 ENO = 0 的错误条件是：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位：SM1.0 (零) ；SM1.1 (溢出，或在运算中产生非法值，或发现输入参数非法) ；SM1.2 (负) ；SM1.3 (被 0 除)

如果在除法操作过程中 SM1.3 被置位，其它的算术状态位保持不变，原始输入操作数也不变。SM1.1 用来指示溢出错误和非法值。如果 SM1.1 置位，SM1.0 和 SM1.2 的状态就无效，原始操作数不改变。如果 SM1.1 和 SM1.3 (在除法操作中) 不置位，SM1.0 和 SM1.2 的状态反映算术操作的结果。

输入/输出	操 作 数	数据类型
IN1, IN2	VD, ID, QD, MD, SMD, SD, LD, AC, 常数, *VD, *AC, *LD	REAL
OUT	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *AC, *LD	REAL

注意：

实数或浮点数在 ANSI/IEEE 754 1958 标准（单精度）中有描述，若要得到更多信息，请参阅该标准。

算术运算举例

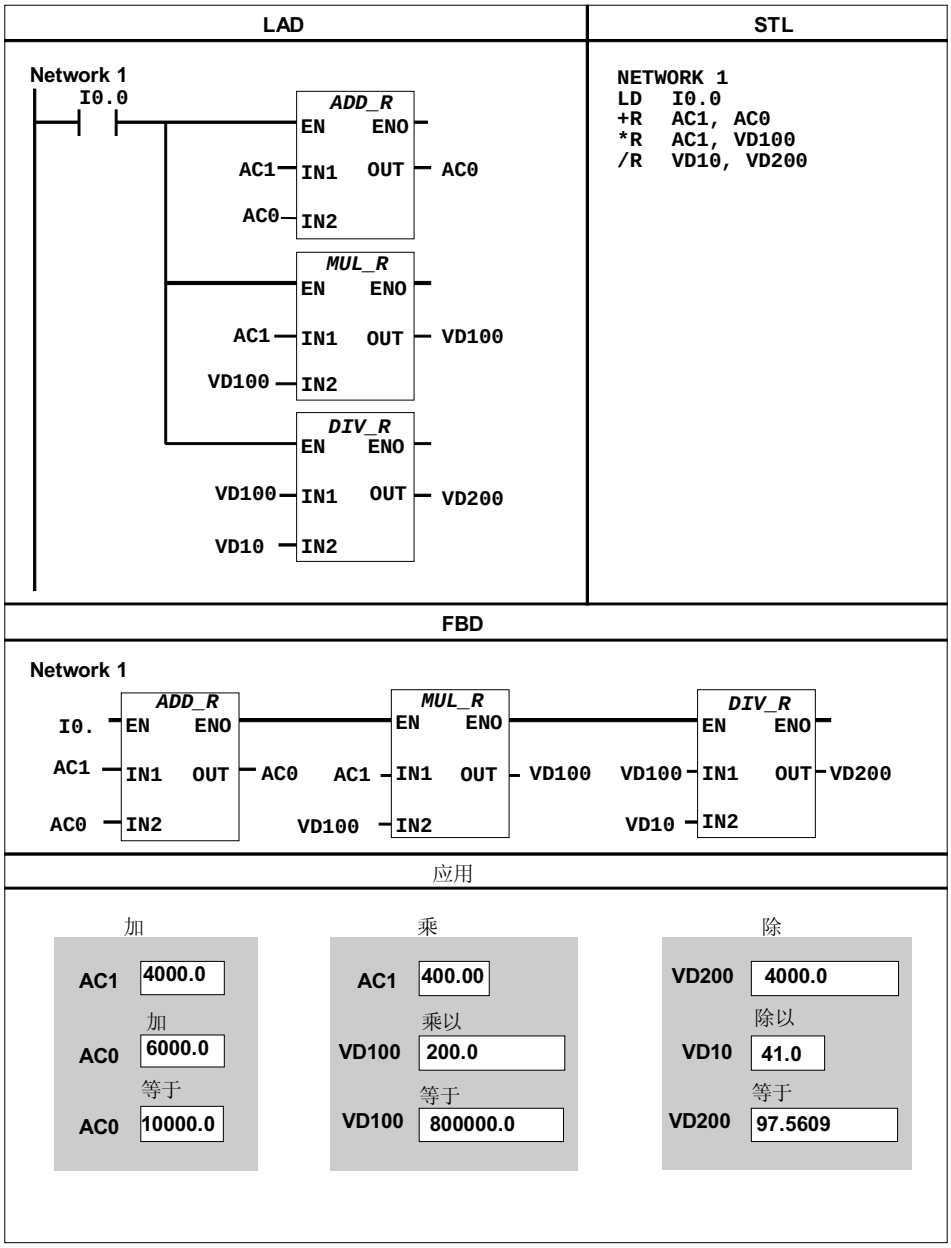
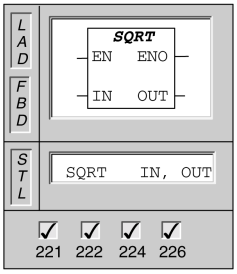


图 9-27 实数算术运算指令的 LAD，STL 和 FBD 举例

9.8 SIMATIC 数学功能指令

平方根



实数的开方指令 (SQRT) 把一个 32 位的实数 (IN) 开方，得到32位实数结果 (OUT)。如下面等式所示：

$\sqrt{IN} = OUT$

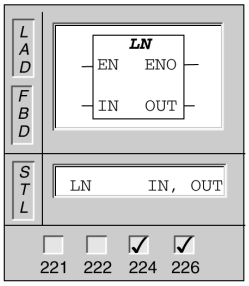
使 ENO = 0 的错误条件是：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位：SM1.0 (零)；SM1.1 (溢出)；SM1.2 (负)

SM1.1 用来指示溢出错误和非法值。如果 SM1.1 置位，SM1.0 和 SM1.2 的状态就无效，原始操作数不改变。如果 SM1.1不置位，SM1.0 和 SM1.2的状态反映算术操作的结果。

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SMD, SD, LD, AC, 常数, *VD, *AC, *LD	REAL
OUT	VD, ID, QD, MD, SMD, SD, LD,AC, *VD, *AC, *LD	REAL

自然对数



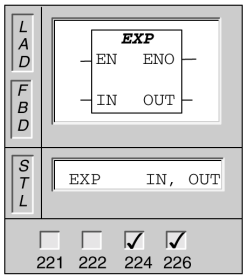
自然对数指令将输入 IN 的值取自然对数，结果放入输出 OUT。当求以 10 为底的自然对数时，用 DIV_R (/R) 指令将自然对数除以 2.302585 即可 (其值近似于以 10 为底的对数值)。

使 ENO=0 的出错条件：SM1.1 (溢出)，0006 (间接寻址)。

该指令影响下列的特殊存储器位：SM1.0 (零位)，SM1.1 (溢出)，SM1.2 (负数)，SM4.3 (运行时间)

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SMD, SD, LD, AC, 常数, *VD, *AC, *LD	REAL
OUT	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *AC, *LD	REAL

指数



指数指令将输入 IN 的值取以e为底的指数，结果放入输出 OUT。

使 ENO=0 的出错条件：SM1.1 (溢出)，0006 (间接寻址)。

这个功能影响下列的特殊存储器位：SM1.0 (零位)， SM1.1 (溢出)，SM1.2 (负数)，SM4.3 (运行时间)。

该指令可与前面的自然对数指令相配合，完成以任意数为底，任意数为指数计算。

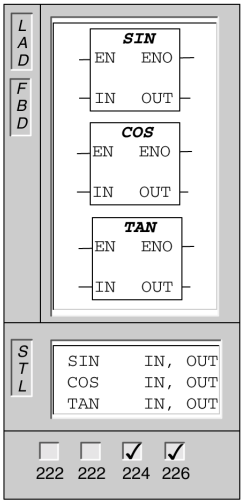
如：

5的3次方=5^3=EXP（3*LN(5)）=125

125的3次方根=125^(1/3)=EXP（1/3* V(125)）=5

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SMD, SD, LD, AC, 常数, *VD, *AC, *LD	REAL
OUT	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *AC, *LD	REAL

正弦、余弦、正切



正弦指令将输入 IN 的弧度值取正弦，结果放入输出 OUT。输入值为弧度值。如果输入为角度值，使用 MUL_R (*R) 将该角度值乘以 π/180 以将其转换为弧度值。

余弦指令将输入 IN 的弧度值取余弦，结果放入输出 OUT。输入值为弧度值。如果输入为角度值，则将该角度值乘以 π /180 以将其转换为弧度值。

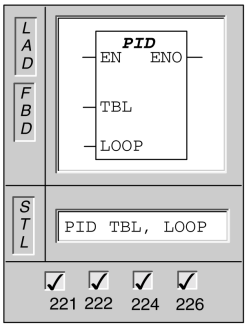
正切指令将输入 IN 的弧度值取正切，结果放入输出 OUT。输入值为弧度值。如果输入为角度值，则将该角度值乘以 π /180 以将其转换为弧度值。

使 ENO=0 的出错条件：SM1.1 (溢出)，0006 (间接寻址)。

这个功能影响下列的特殊存储器位：SM1.0 (零位)，SM1.1 (溢出)，SM1.2 (负数)，SM4.3 (运行时间) 。

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SMD, SD, LD, AC, 常数, *VD, *AC, *LD	REAL
OUT	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *AC, *LD	REAL

PID 回路



PID 回路指令运用以回路表中的输入和组态信息，进 行 PID 运算。

使 ENO = 0 的错误条件是：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)

该指令影响下列特殊存储器标志位：SM1.1 (溢出)

输入/输出	操 作 数	数据类型
TBL	VB	BYTE
LOOP	常数 (0 到 7)	BYTE

PID 回路指令 (包含比例、积分、微分回路) 是用来进行 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：TABLE 和 LOOP。其中 TABLE 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 (PVn-1)，给定值 (SPn)，输出值 (Mn)， 增益 (Kc)，采样时间 (Ts)，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。

使用 STEP 7-Micro/WIN 32 中的 PID 向导

STEP 7–Micro/WIN 32 提供了 PID 向导指导你定义一个闭环控制过程的 PID 算法。选择菜单命令 **Tools>Instruction Wizard**，然后从指令向导窗口中选择 PID 指令。

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：

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

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，其实也不必要。

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：

$$_n = \ _C * e_n + \ _I * e_n + \ + \ _D * (e_n - e_{n-1})$$

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

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
Kc	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

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：

$$MI_n = Kc * T_S / T_I * (SP_n - PV_n) + MX$$

其中：

Mi_n	第 n 采样时刻的积分项值
Kc	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。

微分项

微分项值 MD 与偏差的变化成正比。其计算等式为：

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

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
Kc	回路增益
Ts	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，只需要一种或二种回路控制类型。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，比如是设定的汽车速度。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：

```
XORD      AC0，AC0              //清空累加器。
MOVW      AIW0，AC0             //把待变换的模拟量存入累加器。
LDW>=     AC0，0                //如果模拟量为正
JMP       0                     //则直接转成实数
NOT                             //否则
ORD       16#FFFF0000，AC0      //先对 AC0 中值进行 符号扩展
LBL       0
DTR       AC0，AC0              //把 32 位整数转成实数
```

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，可能最大值减去可能最小值 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，使其落在 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反过程。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，可能最大值减去可能最小值 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
MOVW    AC0, 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 0.0～1.0 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)
- 置过量变量前值 (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 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 9-19 所示。

表 9-19 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。

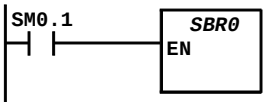
当工作在手动控制方式下，可以把水泵速度 (0.0～1.0 之间的实数) 写到 VD108 (VD108 是回路表中保存输出的寄存器)。

图 9-28 是本控制实例的程序。

LAD

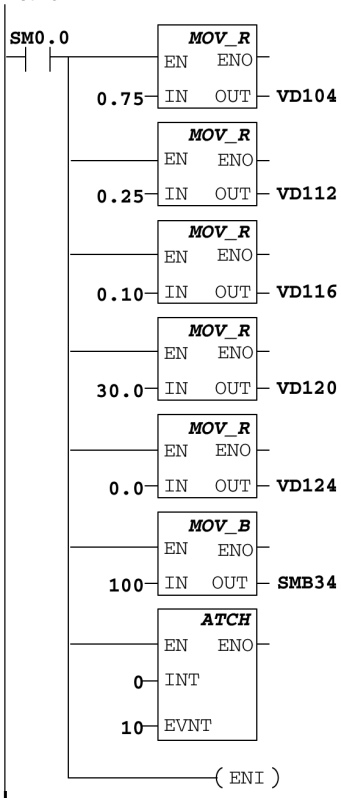
MAIN OB1

Network 1



SUBROUTINE 0

Network 1



STL

```
Network 1
LD   SM0.1   //在首次扫描调用初始化子程序 0
CALL  0
```

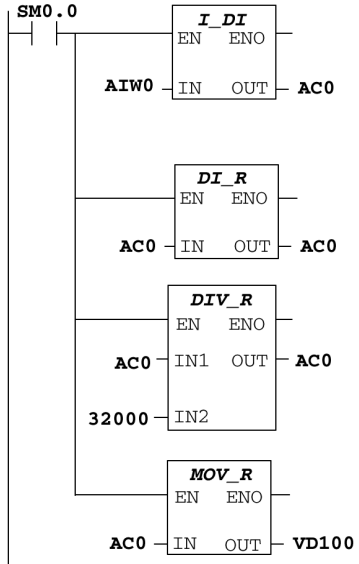
```
Network 1
LD   SM0.0
MOVR   0.75, VD104    //装入设定值75%
MOVR   0.25, VD112    //装入回路增益=0.25
MOVR   0.10, VD116    //装入采样时间=0.1秒
MOVR   30.0, VD120    //装入积分时间=30分
MOVR   0.0, VD124     //关闭微分作用
MOVB   100, SMB34     //设定定时中断0发生的时间
                      //间隔为100ms
ATCH   0, 10          //设置定时中断以执行PID计算
ENI                   //允许中断
                      //子程序0结束
```

图 9-28 PID 回路控制实例

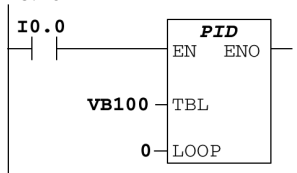
LAD

INTERRUPT 0

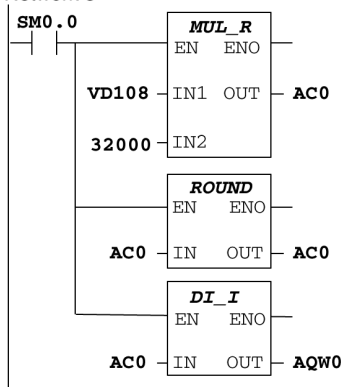
Network 1



Network 2



Network 3



STL

```
NETWORK 1

                                  // 把PV转换成一个标准的实数
                                  // PV是一个单极
LD    SM0.0
ITD   AIW0, AC0                   // 把单极性模拟量存入累加器
DTR   AC0, AC0                    // 把32位整数变换成实数
/R    32000.0, AC0                // 标准化累加器中的值
MOVR  AC0, VD100                  // 把标准化的PV值存入TABLE表

NETWORK 2

LD    I0.0                        // 在自动方式下执行PID控制
PID   VB100, 0                    // 一旦进入自动方式，就执行PID

NETWORK 3

                                  // 把Mn转换成16位整数，Mn为单
LD    SM0.0                           极性且非负
MOVR  VD108, AC0
                                  // 把输出值传送到累加器
*R    32000.0, AC0
                                  // 累加器中为刻度值
ROUND AC0, AC0
                                  // 把实数转换成32位整数
DTI   AC0, AQW0
                                  // 把16位整数写到模拟输出
```

图 9-28　PID 回路控制实例 (续)

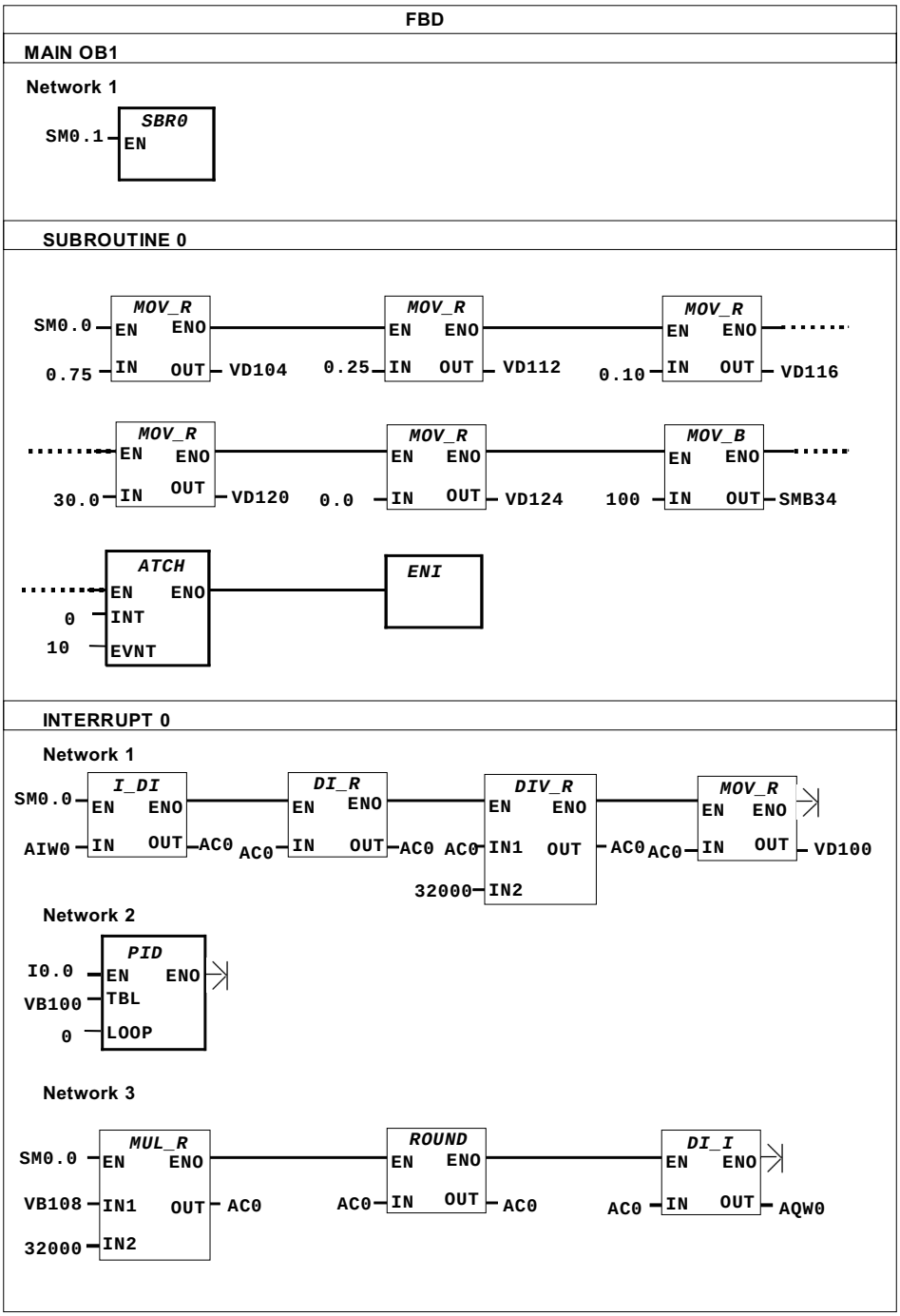
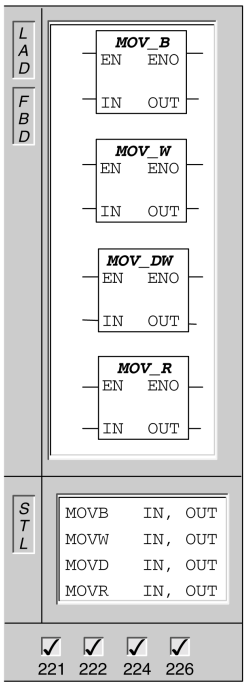


图9-28 PID回路控制实例 (续)

9.9 SIMATIC 传送指令

节，字，双字和实数的传送



传送字节指令 (MOVB) 把输入字节 (IN) 传送到输出字节 (OUT)，在传送过程中不改变字节的大小。

传送字指令 (MOVW) 把输入字节 (IN) 传送到输出字节 (OUT)，在传送过程中不改变字的大小。

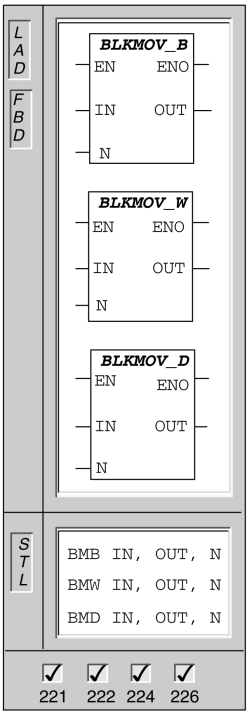
传送双字指令 (MOVDW) 把输入字节 (IN) 传送到输出字节 (OUT)，在传送过程中不改变双字的大小。

传送实数指令 (MOVR) 把输入字节 (IN) 传送到输出字节 (OUT)，在传送过程中不改变实数的大小。

使 ENO = 0 的错误条件是：SM4.3 (运行时间) ；0006 (间接寻址)

传送	输入/输出	操 作 数	数据类型
字节	IN	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD	BYTE
	OUT	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	BYTE
字	IN	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, 常数, AC *VD, *AC, *LD	WORD, INT
	OUT	VW, T, C, IW, QW, SW, MW, SMW, LW, AC, AQW, *VD, *AC, *LD	WORD, INT
双字	IN	VD, ID, QD, MD, SD, SMD, LD, HC, &VB, &IB, &QB, &MB, &SB, &T, &C, AC, 常数, *VD, *AC, *LD	DWORD, DINT
	OUT	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD	DWORD, DINT
实数	IN	VD, ID, QD, MD, SD, SMD, LD, AC, 常数, *VD, *AC, *LD	REAL
	OUT	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD	REAL

字节，字和双字的块传送



传送字节块指令 (BMB) 把从输入字节 (IN) 开始的N个字节值传送到从输出字节 (OUT) 开始的N个字节，N可取1 ～ 255。

传送字块指令 (BMW) 把从输入字 (IN) 开始的N个字值传送到从输出字 (OUT) 开始的N个字，N可取1 ～ 255。

传送双字块指令 (BMDW) 把从输入地址 (IN) 开始的N个双字值传送到从输出地址 (OUT) 开始的N个双字，N可取1 ～ 255。

使ENO = 0的错误条件是：SM4.3 (运行时间) ；0006 (间接寻址) ；0091 (操作数超界)。

块传送	输入/输出	操 作 数	数据类型
字节	IN, OUT	VB, IB, QB, MB,SB, SMB, LB, *VD, *AC, *LD	BYTE
	N	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD	BYTE
字	IN	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, *VD, *AC, *LD	WORD
	N	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD	BYTE
	OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AQW, *VD, *LD, *AC	WORD
双字	IN, OUT	VD, ID, QD, MD, SD, SMD, LD, *VD, *AC, *LD	DWORD
	N	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD	BYTE

块传送实例

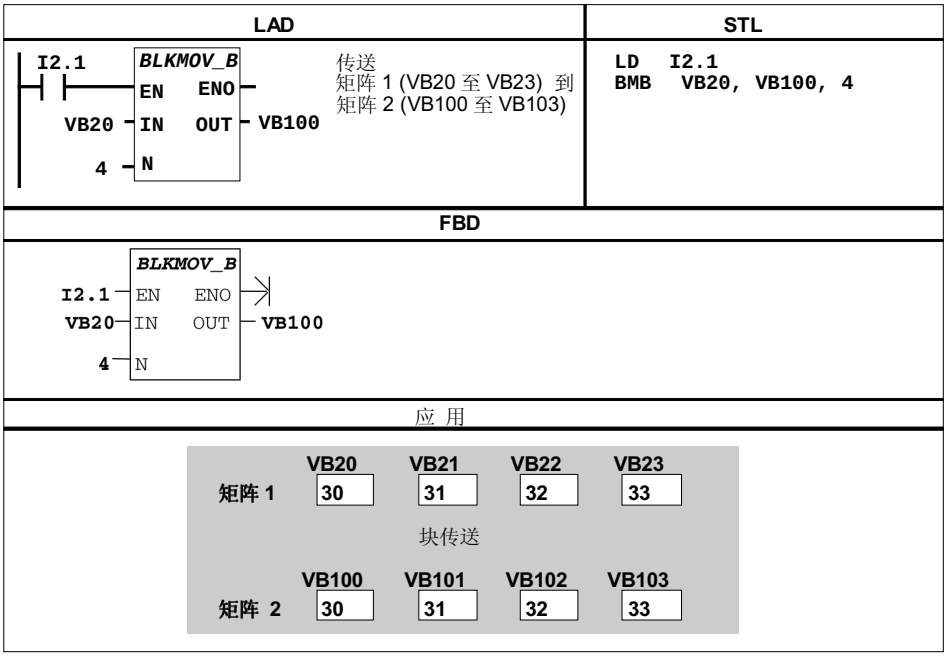
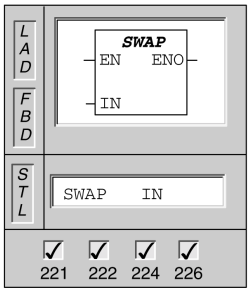


图 9-29　块传送指令的 LAD，STL 和 FBD 实例

交换字节



换字节指令 (SWAP) 用来交换输入字 (IN) 的高字节与低字节。使 ENO = 0 的错误条件是：SM4.3 (运行时间) ；0006 (间接寻址)。

输入/输出	操　作　数	数据类型
IN	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *AC, *LD	WORD

传送与交换指令举例

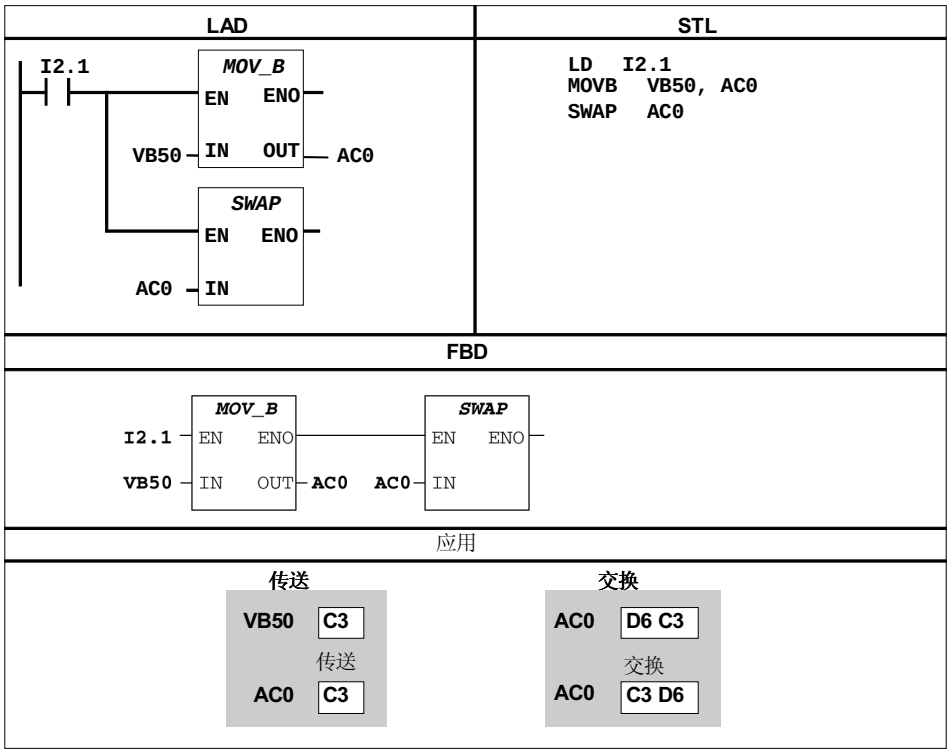
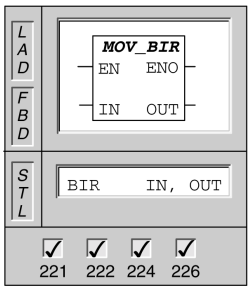


图 9-30　传送和交换指令的 LAD，STL 和 FBD 应用举例

传送字节立即读

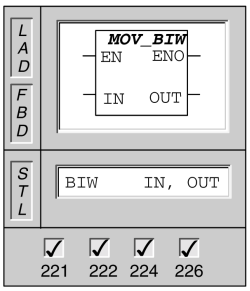


送字节立即读指令读取输入 IN 的物理值，将结果写入输出 OUT。

使 ENO = 0 的错误条件是：SM4.3 (运行时间)，0006 (间接寻址)。

输入/输出	操 作 数	数据类型
IN	IB	BYTE
OUT	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC	BYTE

传送字节立即写



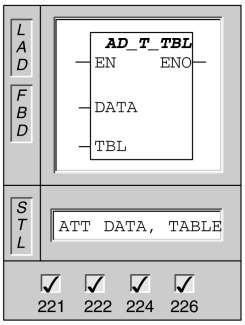
传送字节立即写指令将从输入 IN 读取的值写入输出 OUT 物理影响区。

使 ENO = 0 的错误条件是：SM4.3 (运行时间)，0006 (间接寻址)。

输入/输出	操 作 数	数据类型
IN	VB, IB, QB, MB, SB, SMB, LB, AC, 常数,*VD, *LD, *AC	BYTE
OUT	QB	BYTE

9.10 SIMATIC 表功能指令

填表 (ATT)



ATT **指令**向表 (TBL) 中增加一个字值 (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。一张表除了 TL 和 EC 这二个参数外，还可以有最多 100 个填表数据。

使 ENO = 0 的错误条件是：SM4.3 (运行时间) ；0006 (间接寻址) ；0091 (操作数超界)。

本指令影响下列特殊存储器标志位：如果表溢出，SM1.4 置1。

输入/输出	操 作 数	数据类型
DATA	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, 常数, *VD, *AC, *LD	INT
TBL	VW, IW, QW, MW, SW, SMW, LW, T, C, *VD, *AC, *LD	WORD

填表举例

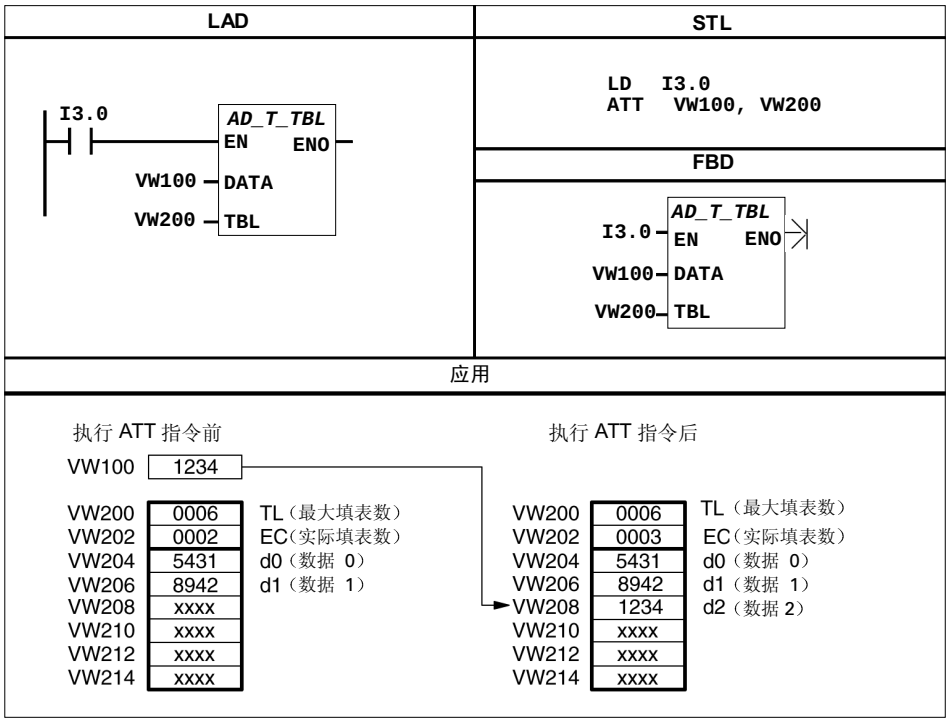
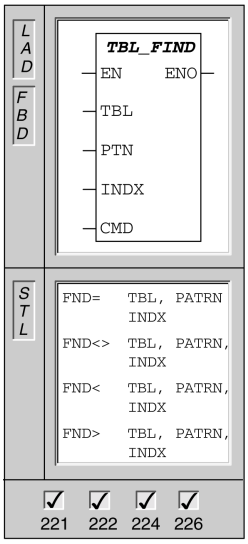


图 9-31　填表指令举例

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。

填表数据的个数 (搜索区域)为 0～99。一张表除了最大填表数 (TL) 和实际填表数 (EC) 这两个参数外，还可有最多 100 个填表数据。

使 ENO = 0 的错误条件是：SM4.3 (运行时间) ；0006 (间接寻址) ；0091 (操作数超界)。

输入/输出	操 作 数	数据类型
SRC	VW, IW, QW, MW, SMW, LW, T, C, *VD, *AC, *LD	WORD
PTN	VW, IW, QW, MW, SW, SMW, AIW, LW, T, C, AC, 常数, *VD, *AC, *LD	INT
INDX	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *AC, *LD	WORD
CMD	常数	BYTE

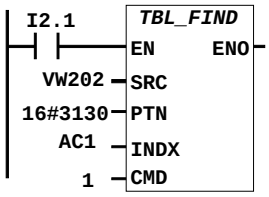
注意：

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 9-33 所示。

图 9-32 FND 查表指令和 ATT、LIFO、FIFO 指令所用表格式的差异

查表举例

LAD

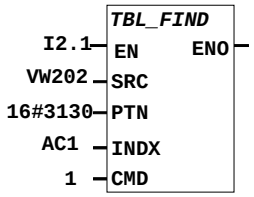


当 I2.1 接通时，从表中寻找等于 3130H 的数

STL

```
LD    I2.1
FND=  VW202, 16#3130, AC1
```

FBD



应 用

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。

VW202	0006	EC (实际填表数)
VW204	3133	d0 (数据 0)
VW206	4142	d1 (数据 1)
VW208	3130	d2 (数据 2)
VW210	3030	d3 (数据 3)
VW212	3130	d4 (数据 4)
VW214	4541	d5 (数据 5)

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 个符合查找条件的数据编号（d4）

AC1 5 查表中剩余数据前，INDEX 加 1

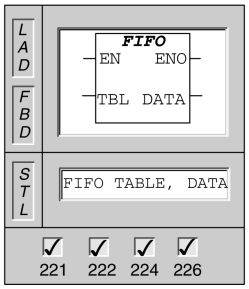
执行查表

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 值须置为 0

图 9-33 查表指令的 LAD、STL 和 FBD 应用实例

先进先出 (FIFO)



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。

使 ENO = 0 的错误条件是：SM1.5 (表空) ；SM4.3 (运行时间) ；0006 (间接寻址) ；0091 (操作数超界)。

本指令影响下列特殊存储器标志位：

如果试图从空表中移走数据，那么 SM1.5 置 1。

输入/输出	操 作 数	数据类型
TABLE	VW, IW, QW, MW, SW, SMW, LW, T, C,*VD, *AC, *LD	INT
DATA	VW, IW, QW, MW, SW, SMW, LW, AC, AQW, T, C, *VD, *AC, *LD	WORD

FIFO 指令举例

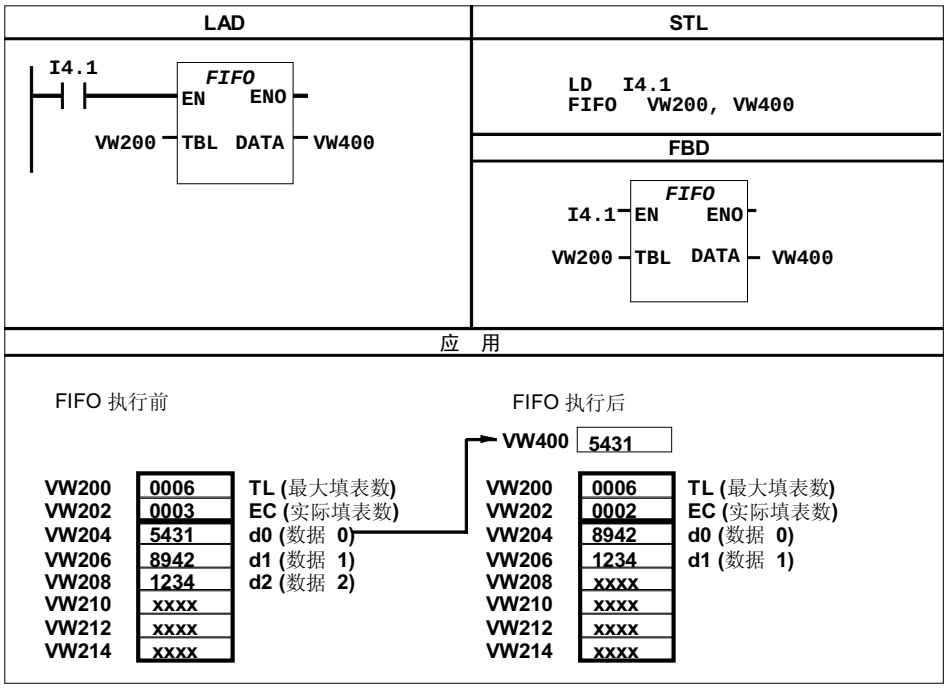
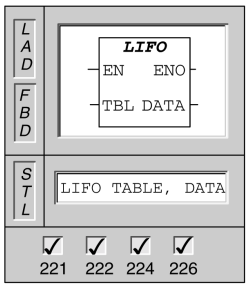


图 9-34 FIFO 指令应用举例

后进先出 (LIFO)



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 的错误条件是：SM1.5 (表空) ；SM4.3 (运行时间) ；0006 (间接寻址) ；0091 (操作数超界)。

本指令影响下列特殊存储器标志位：

如果试图从空表中移走数据，那么 SM1.5 置1。

输入/输出	操 作 数	数据类型
TABLE	VW, IW, QW, MW, SW, SMW, LW, T, C,*VD, *AC, *LD	INT
DATA	VW, IW, QW, MW, SW, SMW, LW, AQW, T, C, AC, *VD, *AC, *LD	WORD

LIFO 指令举例

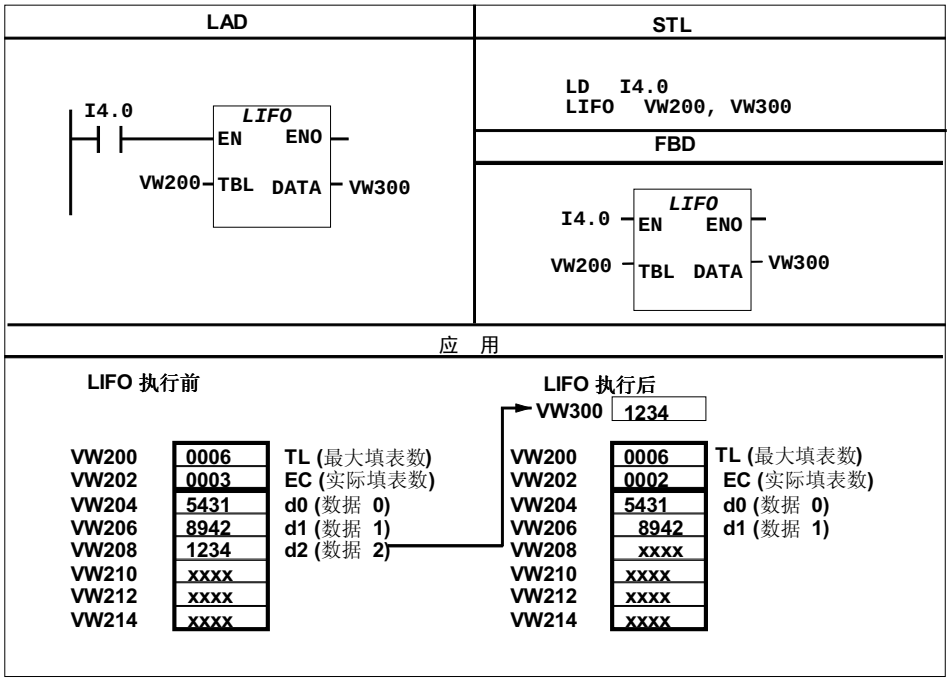
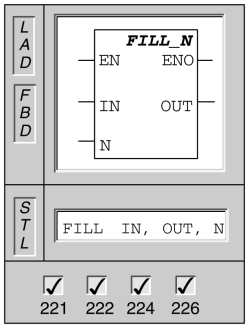


图 9-35 LIFO 指令应用举例

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 的错误条件是：SM4.3 (运行时间) ；0006 (间接寻址) ；0091 (操作数超界)。

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

填充举例

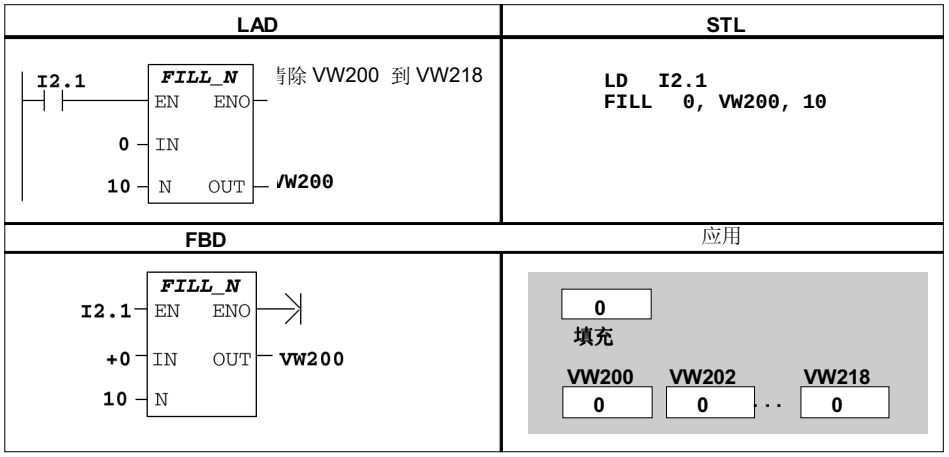
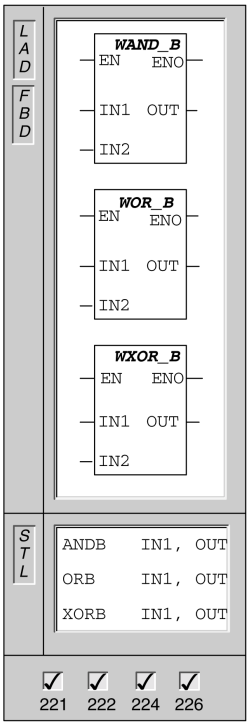


图 9-36 填充指令应用举例

9.11 SIMATIC 逻辑运算指令

字节与，字节或，字节异或



ANDB (字节与) 指令对两个输入字节按位与，得到一个字节结果 (OUT)。

ORB (字节或) 指令对两个输入字节按位或，得到一个字节结果 (OUT)。

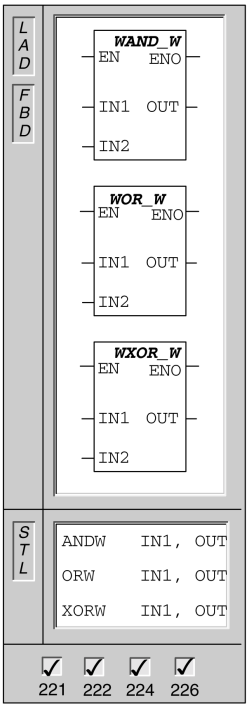
XORB (字节异或) 指令对两个输入字节按位异或，得到一个字节结果 (OUT)。

使 ENO = 0 的错误条件是：SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位： SM1.0 (零)

输入/输出	操 作 数	数据类型
IN1, IN2	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD	BYTE
OUT	VB, IB, QB, MB,SB, SMB, LB, AC, *VD, *AC, *LD	BYTE

字与，字或，字异或



ANDW（字与）指令对两个输入字按位与，得到一个字结果(OUT)。

ORW（字或）指令对两个输入字按位或，得到一个字结果(OUT)。

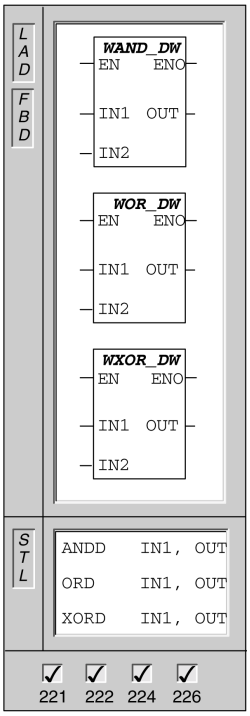
XORW（字异或）指令对两个输入字按位异或，得到一个字结果(OUT)。

使 ENO = 0 的错误条件是：SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位： SM1.0 (零)

输入/输出	操 作 数	数据类型
IN1, IN2	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, 常数, *VD, *AC, *LD	WORD
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *AC, *LD	WORD

双字与，双字或，双字异或



ANDD （双字与）指令对两个输入双字按位与，得到一个双字结果 (OUT)。

ORD （双字或）指令对两个输入双字按位或，得到一个双字结果 (OUT)。

XORD（ 双字异或）指令对两个输入双字按位异或，得到一个双字结果 (OUT)。

使 ENO = 0 的错误条件是：SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位： SM1.0 (零)

输入/输出	操 作 数	数据类型
IN1, IN2	VD, ID, QD, MD, SMD, AC, LD, HC, 常数, *VD, *AC, SD, *LD	DWORD
OUT	VD, ID, QD, MD, SMD, LD, AC, *VD, *AC, SD, *LD	DWORD

与，或，异或指令举例

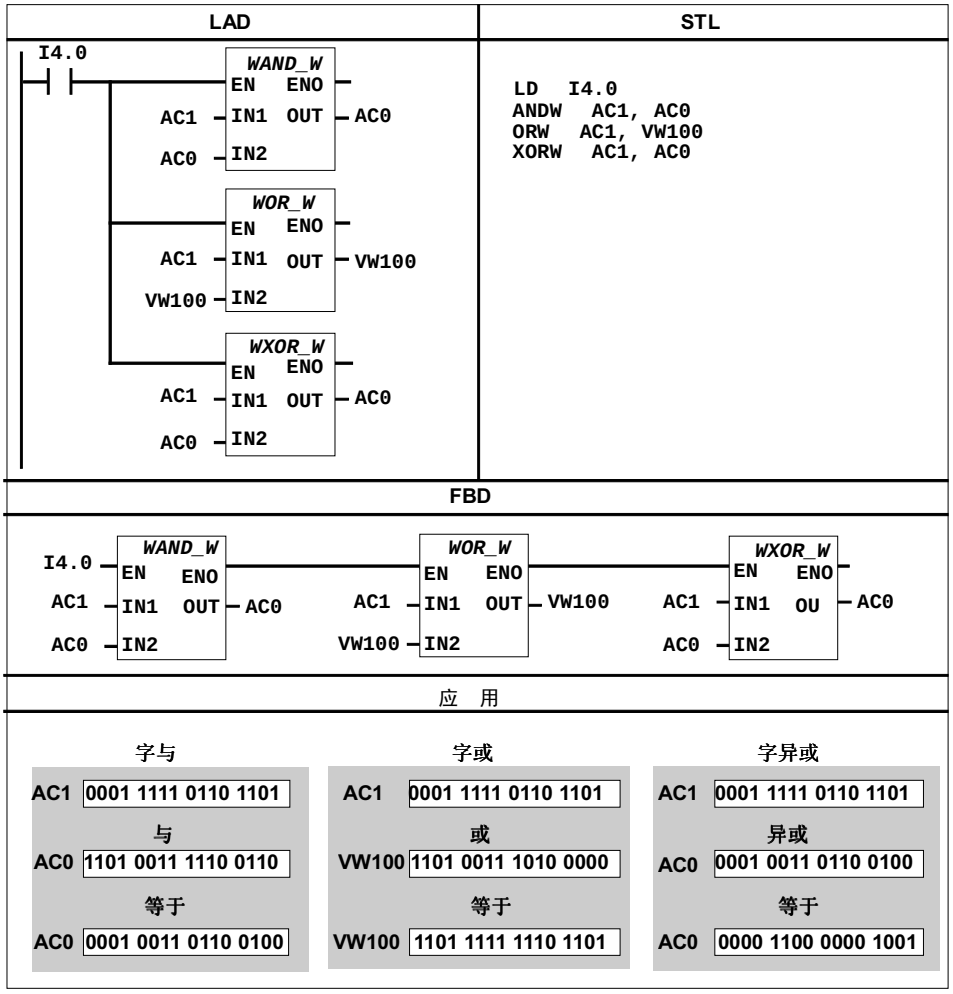
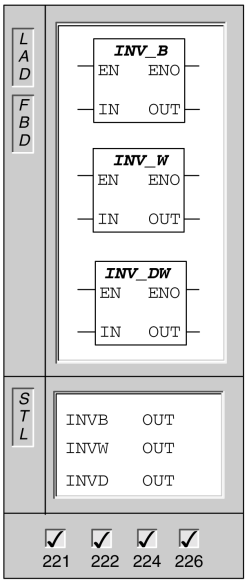


图 9-37　逻辑操作指令举例

字节取反，字取反，双字取反指令



INVB（字节取反）指令求出输入字节 (IN) 的反码，得到一个字结果 (OUT)。

INVW（字取反）指令求出输入字 (IN) 的反码，得到一个字结果 (OUT)。

INVDW（双字取反）指令求出输入双字 (IN) 的反码，得到一个字结果 (OUT)。

使 ENO = 0 的错误条件是：SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位： SM1.0 (零)

Invert...	输入/输出	操 作 数	数据类型
字节	IN	VB, IB, QB, MB,SB, SMB, LB, AC, 常数, *VD, *AC, *LD	BYTE
	OUT	VB, IB, QB, MB, SB, SMB, LB, AC,*VD, *AC, *LD	BYTE
字	IN	VW, IW, QW, MW, SW, SMW, T, C, AIW, LW, AC, 常数, *VD, *AC, *LD	WORD
	OUT	VW, IW, QW, MW,SW, SMW, T, C, LW, AC, *VD, *AC, *LD	WORD
双字	IN	VD, ID, QD, MD, SD, SMD, LD, HC, AC, 常数, *VD, *AC, *LD	DWORD
	OUT	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD	DWORD

取反举例

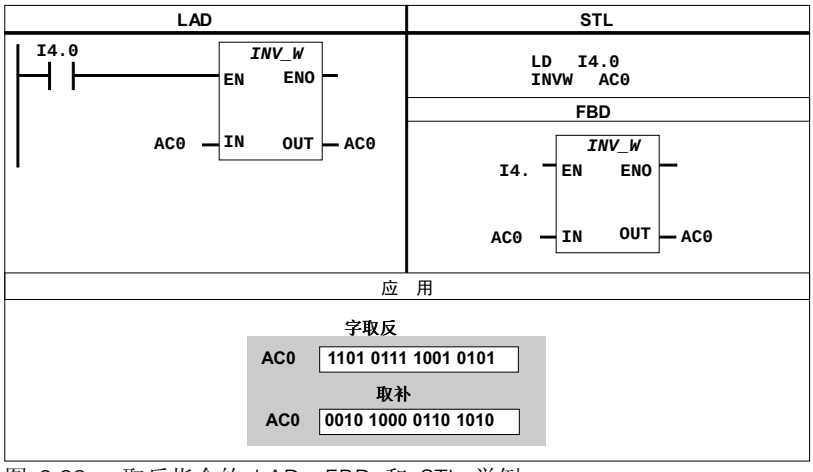
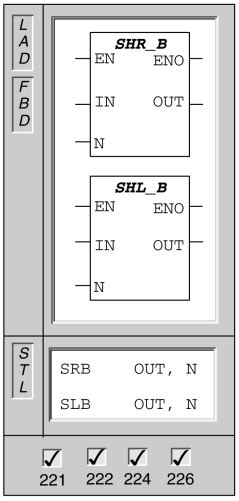


图 9-38 取反指令的 LAD、FBD 和 STL 举例

9.12 SIMATIC 移位和循环指令

字节右移位和左移位



字节左移位指令 (SLB) 或**右移位**指令 (SRB) 把输入字节 (IN) 左移或右移N位后，再把结果输出到 OUT 字节。

移位指令对移出位自动补零。如果所需移位次数N大于或等于8，那么实际最大可移位数为 8。

如果所需移位次数大于零，那么溢出位 (SM1.1) 上就是最近移出的位值。如果移位操作的结果是 0，零存储器位 (SM1.0) 就置位。

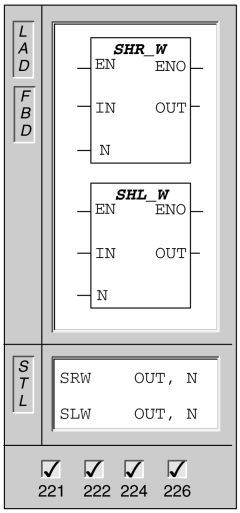
字节左移位或右移位操作是无符号的。

使 ENO = 0 的错误条件是：SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位： SM1.0 (零) ；SM1.1 (溢出)

输入/输出	操 作 数	数据类型
IN	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD，常数	BYTE
OUT	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	BYTE
N	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD	BYTE

字左移位或右移位



字左移位指令 (SLW) 或**右移位**指令 (SRW) 把输入字 (IN) 左移或右移 N 位后，再把结果输出到字 OUT。

移位指令会对移出位自动补零。如果所需移位次数N大于或等于16，那么实际最大可移位数为 16。如果所需移位数大于零，那么溢出位 (SM1.1) 上就是最近一次移出的位值。如果移位操作的结果是0，零存储器位 (SM1.0) 就置位。

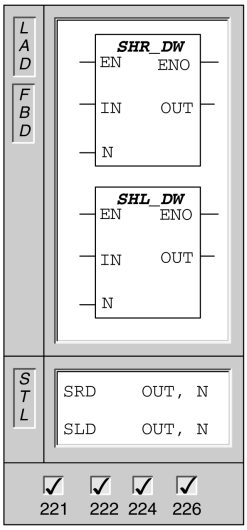
字左移位或右移位操作是无符号的。

使 ENO = 0 的错误条件是：SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位： SM1.0 (零) ；SM1.1 (溢出)

输入/输出	操 作 数	数据类型
IN	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, 常数, *VD, *AC, *LD	WORD
N	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD	BYTE
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *AC, *LD	WORD

双字左移位或右移位



双字左移位指令 (SLDW) 或**右移位**指令 (SRDW) 把输入双字 (IN) 左移或右移N位后，再把结果输出到双字 (OUT)。

移位指令会对移出位自动补零。如果所需移位次数N大于或等于 32，那么实际最大可移位数为 32。如果所需移位次数大于零，那么溢出位 (SM1.1) 上就是最近一次移出的位值。如果移位操作的结果是 0，零存储器位 (SM1.0) 就置位。

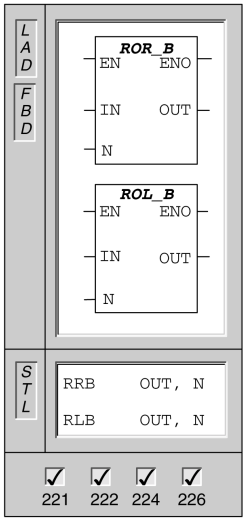
字左移位或右移位操作是无符号的。

使 ENO = 0 的错误条件是：SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位： SM1.0 (零) ；SM1.1 (溢出)

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SD, SMD, LD, AC, HC, 常数, *VD, *AC, *LD	DWORD
N	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD	BYTE
OUT	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD	DWORD

字节循环左移或循环右移



字节循环左移指令 (RLB) 或**循环右移**指令 (RRB) 把输入字节 (IN) 循环左移或循环右移N位后，再把结果输出到字节 (OUT)。

如果所需移位次数大于或等于 8，那么在执行循环移位前，先对N取以 8 为底的模，其结果 0～7 为实际移动位数。如果所需移位数为零，那就不执行循环移位。如果执行循环移位的话，那么溢出位 (SM1.1) 值就是最近一次循环移动位的值。

如果移位次数不是 8 的整数倍，最后被移出的位就存放到溢出存储器位 (SM1.1)。如果移位操作的结果是 0，零存储器位 (SM1.0) 就置位。

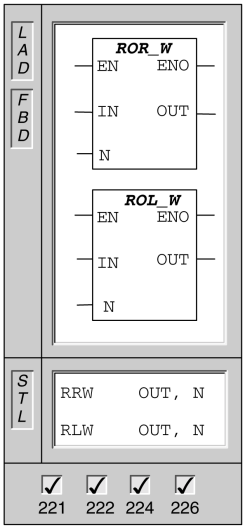
字节循环移位操作是无符号的。

使 ENO = 0 的错误条件是：SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位： SM1.0 (零) ；SM1.1 (溢出)

输入/输出	操 作 数	数据类型
IN	VB, IB, QB, MB, SMB, SB, LB, AC, *VD, *AC, *LD	BYTE
N	VB, IB, QB, MB, SMB, SB, LB, AC, 常数, *VD, *AC, *LD	BYTE
OUT	VB, IB, QB, MB, SMB, SB, LB, AC, *VD, *AC, *LD	BYTE

字循环左移或循环右移



字循环左移指令 (RLW) 或**右移**指令 (RRW) 把输入字 (IN) 循环左移或循环右移 N 位后，再把结果输出到字 (OUT)。

如果所需移位次数大于或等于 16，那么在执行循环移位前，先对N取以 16 为底的模，其结果 0～15 为实际所移位数。如果所需移位数为零，那就不执行循环移位。如果执行循环移位的话，那么溢出位 (SM1.1) 上的值就是最近一次循环移动位的值。

如果移位次数不是 16 的整数倍，最后被移出的位就存放到溢出存储器位 (SM1.1)。如果移位操作的结果是 0，零存储器位 (SM1.0) 就置位。

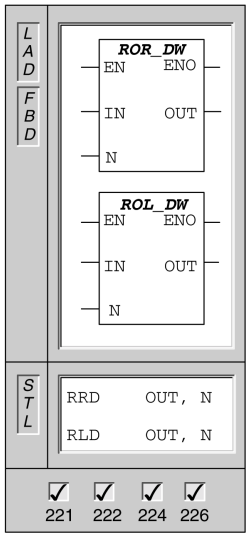
字循环移位操作是无符号的。

使 ENO = 0 的错误条件是：SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位： SM1.0 (零) ；SM1.1 (溢出)

输入/输出	操 作 数	数据类型
IN	VW, T, C, IW, MW, SMW, AC, QW, LW, AIW, 常数, *VD, *AC, SW, *LD	WORD
N	VB, IB, QB, MB, SMB, LB, AC, 常数, *VD, *AC, SB, *LD	BYTE
OUT	VW, T, C, IW, QW, MW, SMW, LW, AC, *VD, *AC, SW, *LD	WORD

双字循环左移或循环右移



双字循环左移指令 (RLD) 或**循环右移**指令 (RRD) 把输入双字 (IN) 循环左移或循环右移 N 位，再把结果输出到双字 (OUT)。

如果所需移位次数 N 大于或等于 32，那么在执行循环移位前，先对 N 取以 32 为底的模，其结果 0～31 为实际所移位数。如果所需移位数为零，那就不执行循环移位。如果执行循环移位的话，那么溢出位 (SM1.1) 上的值就是最近一次循环移动的值。

如果移位次数不是 32 的整数倍，最后被移出的位就存放到溢出存储器位 (SM1.1)。如果移位操作的结果是 0，零存储器位 (SM1.0) 就置位。

字循环移位操作是无符号的。

使ENO = 0的错误条件是：SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下面的特殊存储器位：SM1.0 (零)；SM1.1 (溢出)

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SMD, LD, AC, HC, 常数, *VD, *AC, SD, *LD	DWORD
N	VB, IB, QB, MB, SMB, LB, AC, 常数, *VD, *AC, SB, *LD	BYTE
OUT	VD, ID, QD, MD, SMD, LD, AC, *VD, *AC, SD, *LD	DWORD

移位和循环举例

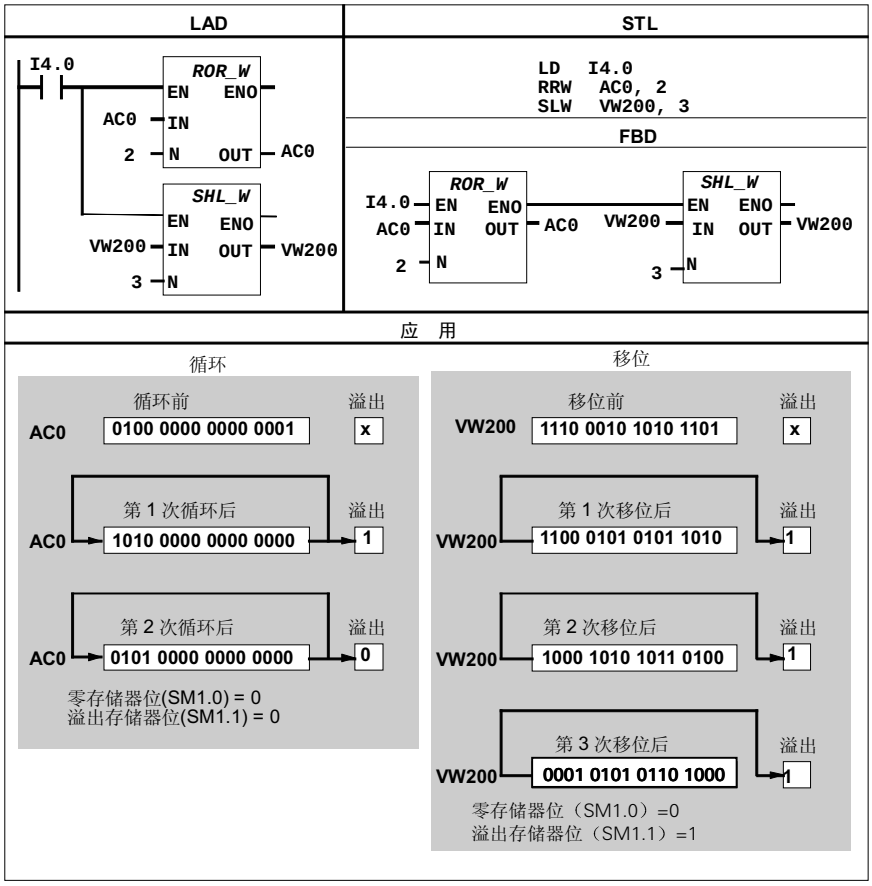
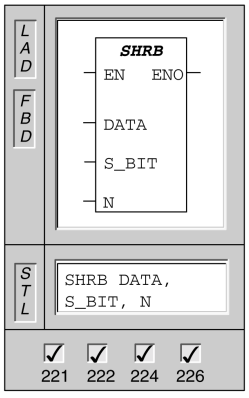


图 9-39　移位和循环指令的 LAD，STL 和 FBD 举例

位移位寄存器指令



位移位寄存器指令 (SHRB) 把输入的 DATA 数值移入移位寄存器，而该移位寄存器是由 S-BIT 和 N 决定的。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。(正向移位= N，反向移位= -N)。

SHRB 指令的位被放再溢出位 (SM1.1)。

使 ENO = 0 的错误条件是：SM4.3 (运行时间) ；0006 (间接寻址) ；0091 (操作数超界) ；0092 (计数区错误)

这些指令影响下面的特殊存储器位：SM1.1 (溢出)

输入/输出	操　作　数	数据类型
DATA, S_BIT	I, Q, M, SM, T, C, V, S, L	BOOL
N	VB, IB, QB, MB, SMB, LB, AC, 常数, *VD, *AC, SB, *LD	BYTE

对位移位寄存器指令的理解

位移位寄存器指令提供了一种排列和控制产品流或数据流的简单方法。在每个扫描周期，整个移位寄存器移动一位。此移位寄存器由 S-BIT 和 N 决定。图 9-41 为位移位寄存器应用举例。

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]

因为 S-BIT 也是移位寄存器中的一位，故必须减 1。

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

位移位寄存器移位方向由 N 的正负决定。反移时，N 为负，输入数据从最高位移入，最低位 (S-BIT) 移出；正移时，N 为正，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存储器位 (SM1.1)。移位寄存器的最大长度是 64 位，可正可负。图 9–40 为两种不同方向的移位。

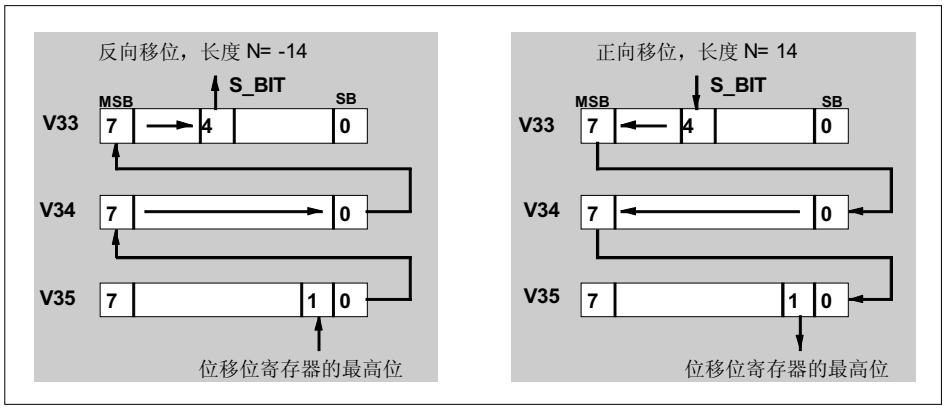


图 9-40 正移、反移时位移位寄存器的入口与出口

位移位寄存器指令举例

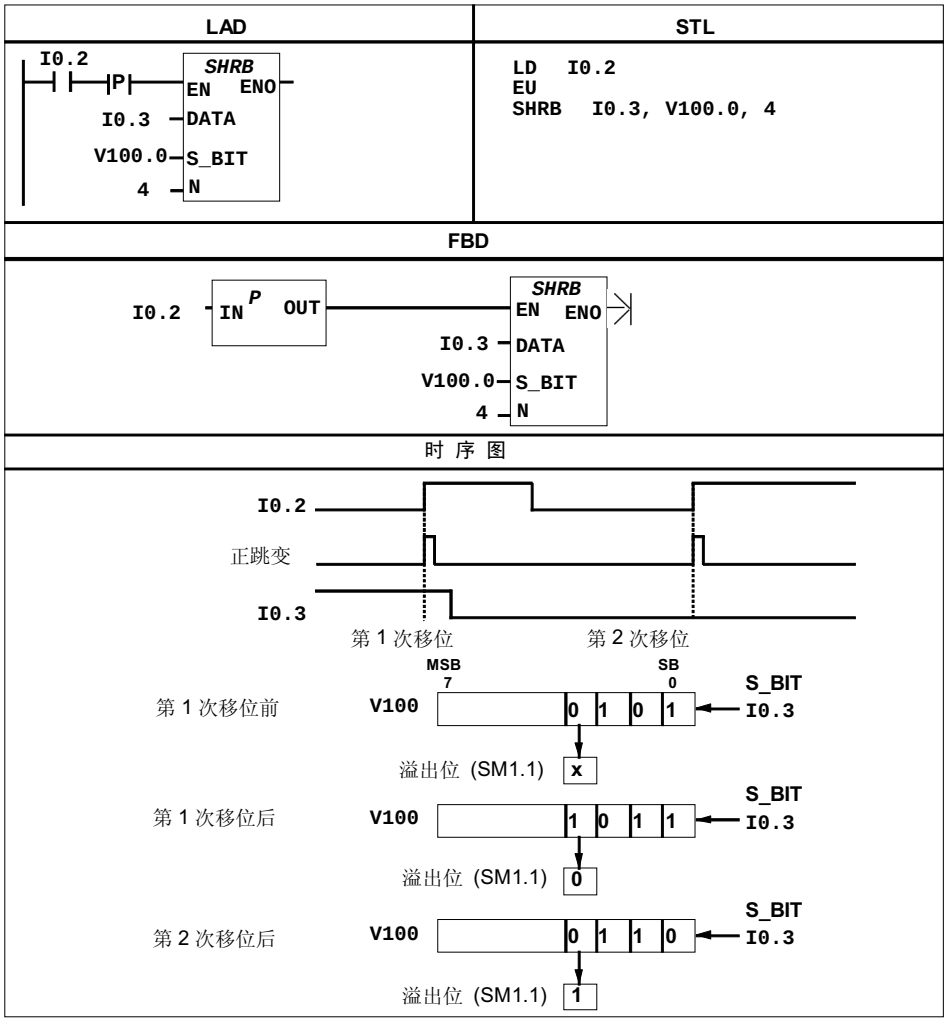
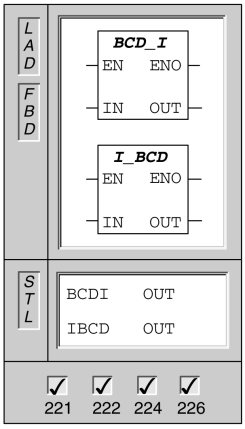


图 9-41　位移位寄存器的 LAD，STL 和 FBD 应用举例

9.13 SIMATIC 转换指令

BCD 码转为整数 (BCDI)，整数转为 BCD码 (IBCD)



CDI 指令将输入的 **BCD 码 (IN) 转换成整数 (OUT)**，即将结果送入 OUT，输入 IN 的范围是 0到9999。

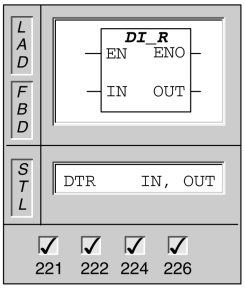
IBCD 指令将输入的**整数 (IN) 转换成 BCD 码 (OUT)**，即将结果送入 OUT，输入IN的范围是 0 到9999。

使 ENO = 0 的错误条件是：SM1.6 (BCD 错误) ；SM4.3 (运行时间) ；0006 (间接寻址)

这些指令影响下面的特殊存储器位： SM1.6 (非法BCD)

输入/输出	操 作 数	数据类型
IN	VW, T, C, IW, QW, MW, SMW, LW, AC, AIW, 常数, *VD, *AC, SW, *LD	WORD
OUT	VW, T, C, IW, QW, MW, SMW, LW, AC, *VD, *AC, SW, *LD	WORD

双字整数转为实数 (DTR)

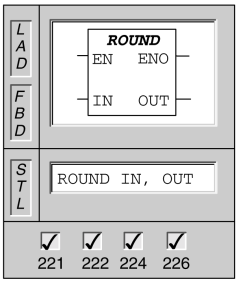


DTR 指令将 32 位有符号整数 (IN) 转换成 32 位实数 (OUT)。

使 ENO = 0 的错误条件是：SM4.3 (运行时间)；0006 (间接寻址)。

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SMD, AC, LD, HC, 常数, *VD, *AC, SD, *LD	DINT
OUT	VD, ID, QD, MD, SMD, LD, AC, *VD, *AC, SD, *LD	REAL

取整 (ROUND)



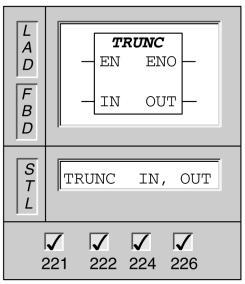
取整指令 (ROUND) 将实数 (IN) 转换成双整数值(OUT)。如果小数部分大于 0.5，就仅一位。

使 ENO = 0 的错误条件是：SM1.1 (溢出) ；SM4.3 (运行时间) ；0006 (间接寻址)

这些指令影响下面的特殊存储器位： SM1.1 (溢出)

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SMD, AC, LD, HC, 常数, *VD, *AC, SD, *LD	REAL
OUT	VD, ID, QD, MD, SMD, LD, AC, *VD, *AC, SD, *LD	DINT

取整 (TRUNC)



取整指令 (TRUNC) 将 32 位实数 (IN) 转换成 32 位有符号整数 (OUT)，只有实数的整数部分被转换 (舍去小数部分)。

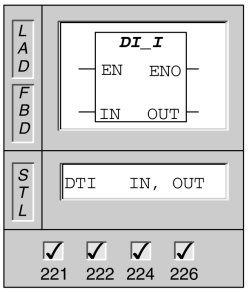
如果要转换的值是无效的实数，或太大而输出无法表示，溢出位被置位，输出不变化。

使 ENO = 0 的错误条件是：SM1.1 (溢出) ；SM4.3 (运行时间) ；0006 (间接寻址)

这些指令影响下面的特殊存储器位： SM1.1 (溢出)

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SMD, LD, AC, 常数, *VD, *AC, SD, *LD	REAL
OUT	VD, ID, QD, MD, SMD, LD, AC, *VD, *AC, SD, *LD	DINT

双整数到整数



双整数到整数转换指令把输入端 (IN) 的双整数转换成一个整数 (OUT)。

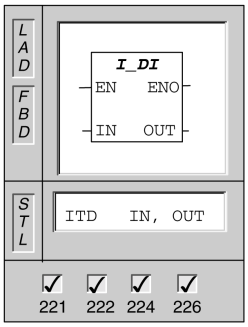
如果要转换的数太大，溢出位被置位，并且输出保持不变。

使 ENO = 0 的错误条件是：SM1.1 (溢出) ；SM4.3 (运行时间) ；0006 (间接寻址)

这些指令影响下面的特殊存储器位： SM1.1 (溢出)

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SMD, AC, LD, HC, 常数, *VD, *AC, SD, *LD	DINT
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC	INT

整数到双整数



整数到双整数转换指令把把输入端 (IN) 的整数转换成一个双整数 (OUT)，符号进行扩展。

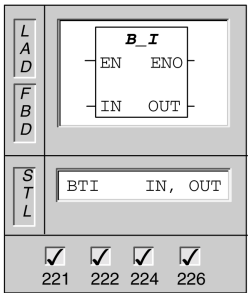
使 ENO = 0 的错误条件是： SM4.3 (运行时间) ；0006 (间接寻址)

输入/输出	操 作 数	数据类型
IN	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, 常数, *AC, *VD, *LD	INT
OUT	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC	DINT

整数到实数

整数转换到实数时，使用整数到双整数指令(9-98页)，然后再使用双整数到实数指令。参见 9-33 页。

字节到整数

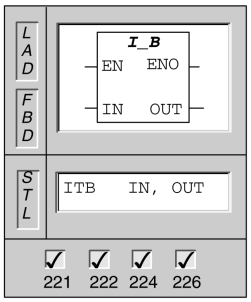


字节到整数转换指令把输入端 (IN) 的字节值转换成一个整数 (OUT)。由于字节是无符号的，所以，没有符号扩展。

使 ENO = 0 的错误条件是： SM4.3 (运行时间) ；0006 (间接寻址)

输入/输出	操 作 数	数据类型
IN	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *AC, *VD, *LD	BYTE
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC	INT

整数到字节



整数到字节转换指令把输入端 (IN) 的字转换成一个字节 (OUT)。

值的范围是 0 到255，所有其它的值会造成溢出，输出不变化。

使 ENO = 0 的错误条件是：SM1.1 (溢出) ；SM4.3 (运行时间) ；0006 (间接寻址)

这些指令影响下面的特殊存储器位： SM1.1 (溢出)

输入/输出	操 作 数	数据类型
IN	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, 常数, *VD, *LD, *AC	INT
OUT	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	BYTE

转换指令举例

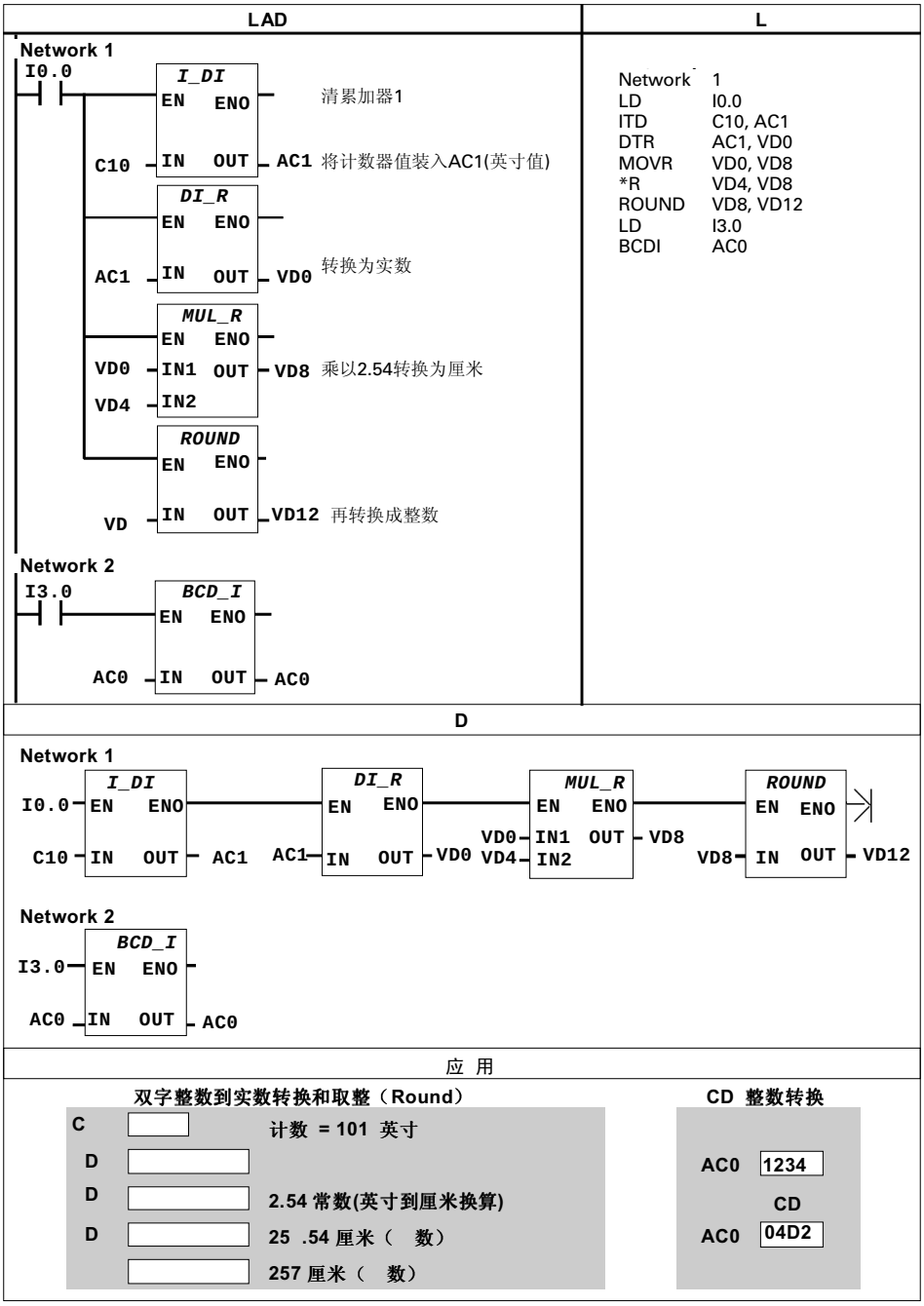
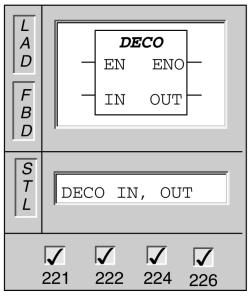


图 9-42　转换指令实例

译码

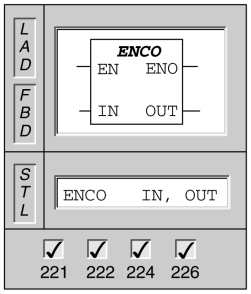


译码指令 (DECO) 根据输入字节 (IN) 的低四位 (半个字节) 所表示的位号置输出字 (OUT) 的相应位为 1，其他位置 0。

使ENO = 0的错误条件是： SM4.3 (运行时间) ；0006 (间接寻址)。

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

编码



编码指令 (ENCO) 将输入字 (IN) 的最低有效位 (值为 1) 的位号写入输出字节 (OUT) 的低四位 (半个字节)。

使 ENO = 0 的错误条件是：SM4.3 (运行时间)；0006 (间接寻址)。

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

译码和编码举例

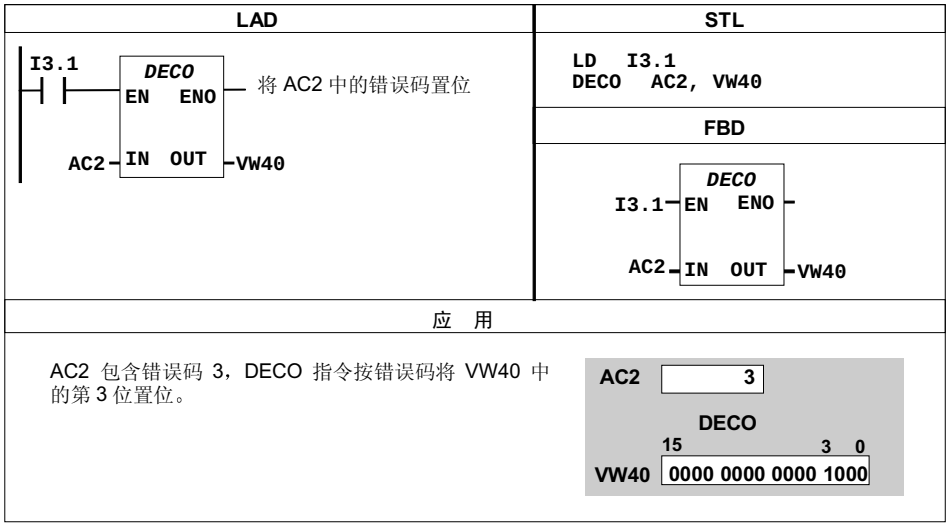


图 9-43　用译码指令设定错误位举例

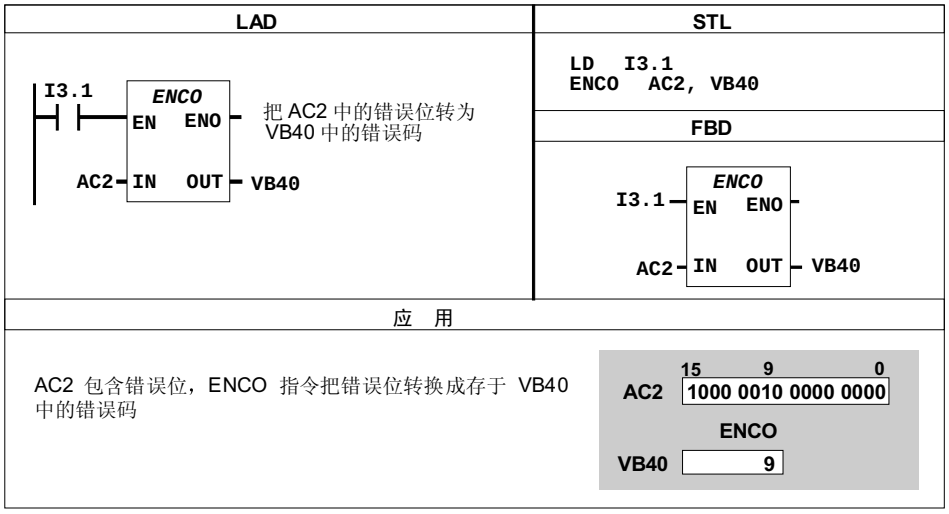
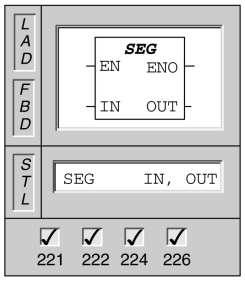


图 9-44　用编码指令 (ENCO) 把错误位转换成错误码举例

段码 (SEG)



段码指令 (SEG) 产生点亮七段码显示器的位模式段码值 (OUT)。它是根据输入字节 (IN) 的低 4 位的有效数字值产生相应点亮段码。

使 ENO = 0 的错误条件是： SM4.3 (运行时间) ； 0006 (间接寻址)。

图 9–45 给出了用段码指令 (SEG) 编码的七段码显示。

输入/输出	操 作 数	数据类型
IN	VB, IB, QB, MB, SMB, LB, AC, 常数, *VD, *AC, SB, *LD	BYTE
OUT	VB, IB, QB, MB, SMB, LB, AC, *VD, *AC, SB, *LD	BYTE

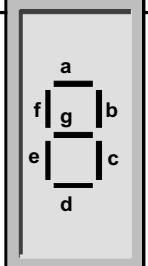
(IN) LSD	段显示	(OUT) -gfe dcba		(IN) LSD	段显示	(OUT) -gfe dcba
0	0	0011 1111		8	8	0111 1111
1	1	0000 0110		9	9	0110 0111
2	2	0101 1011		A	A	0111 0111
3	3	0100 1111		B	b	0111 1100
4	4	0110 0110		C	C	0011 1001
5	5	0110 1101		D	d	0101 1110
6	6	0111 1101		E	E	0111 1001
7	7	0000 0111		F	F	0111 0001

图 9-45 七段显示编码

段码举例

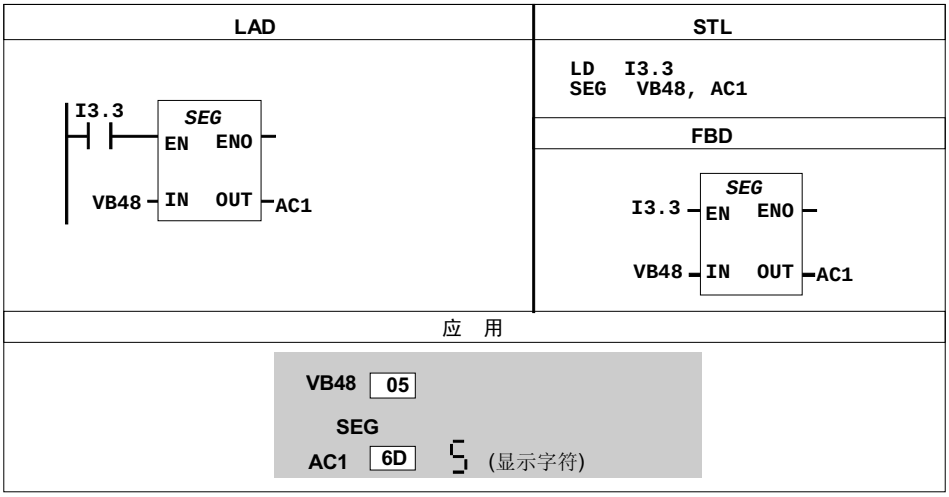
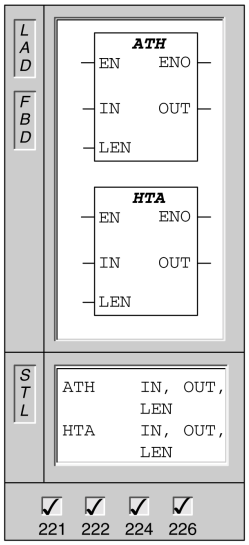


图 9-46 段指令举例

ASCII 码转为 16 进制 (ATH)，16 进制转为 ASCII 码 (HTA)



ATH 指令把从 IN 字符开始，长度为 LEN 的 ASCII 码字符串转换成从 OUT 开始的 16 进制数。ASCII 码字符串的最大长度为 255 个字符。

HTA 指令把从 IN 字符开始，长度为 LEN 的 16 进制数转换成从OUT 开始的ASCII 码字符串。可转换的 16 进制数的最大个数为 255。

合法的 ASCII 码字符的 16 进制值在 30～39 和 41～46 之间。

ASCII 码到 16 进值：使 ENO = 0 的错误条件是：SM1.7 (非法ASCII 码)， SM4.3 (运行时间) ；0006 (间接寻址) ；0091 (操作数超界)

16 进值到 ASCII码：使ENO = 0的错误条件是： SM4.3 (运行时间) ；0006 (间接寻址) ；0091 (操作数超界)

这些指令影响下列特殊存储器标志位：SM1.7 (非法 ASCII 码)

输入/输出	操 作 数	数据类型
IN, OUT	VB, IB, QB, MB, SMB, LB, *VD, *AC, SB, *LD	BYTE
LEN	VB, IB, QB, MB, SMB, LB, AC, 常数, *VD, *AC, SB, *LD	BYTE

ASCII 码到 16 进值转换举例

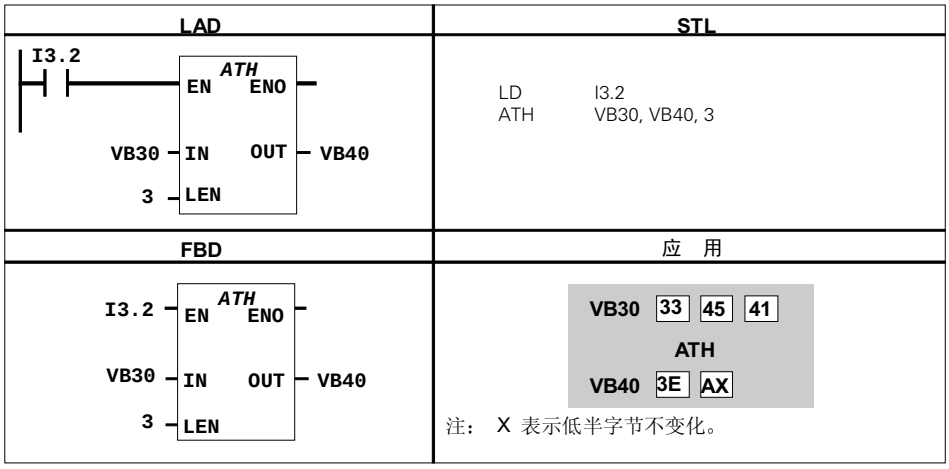
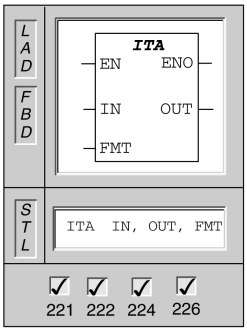


图 9-47 ASCII 码到 16 进值转换指令举例

整数到 ASCII 码



整数到 ASCII 指令把输入端 (IN) 的整数转换成一个 ASCII串。格式 (FMT) 指定右对位的转换精度，十进制对位是逗号或间隔。转换的结果放在 OUT 指定的连续 8 个字节中。ASCII 码串始终是 8 个字符。

使 ENO = 0 的出错条件是： SM4.3 (运行时间) ；0006 (间接寻址) ；无输出 (非法格式)

输入/输出	操 作 数	数据类型
IN	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, 常数, *VD, *AC, *LD	INT
FMT	VB, IB, QB, MB, SMB, LB, AC, 常数, *VD, *AC, SB, *LD	BYTE
OUT	VB, IB, QB, MB, SMB, LB, *VD, *AC, SB, *LD	BYTE

ITA (整数到 ASCII 码转换) 指令的格式操作数 (FMT) 定义如图 9-48 所示。输出缓冲区的大小始终是 8 个字节。nnn 区指定输出缓冲区中的十进制对位右边的位数。nnn 区的有效范围是 0 到 5。指定十进制右对位为 0，表示显示的值没有小数位。对于大于 5 的 nnn，输出缓冲区用 ASCII 空格填充。位 c 指定是用逗号 (c=1) 或小数点 (c=0) 作为整数和小数的分割符。高 4 位必须为 0。

输出缓冲区按照下面的规则进行格式化：

1. 正值不带符号写入输出缓冲区。
2. 负值带负号写入输出缓冲区。
3. 小数点左边前的 0 进行删除处理 (除非临近小数点的数字 0)。
4. 在缓冲区中数值采用右对齐。

图9–48是采用小数点(c=0)进行格式化的数的格式，在小数点右边有三位数字(nnn=011)。

FMT

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

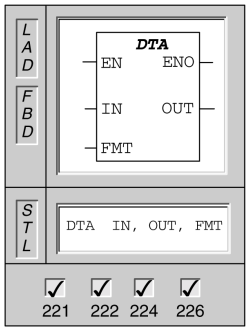
例如:

c = 逗号(1) 或小数点(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 9-48 ITA 指令的 FMT 操作数

双整数到 ASCII 码



双整数到 ASCII 指令把输入端 (IN) 的整数转换成一个 ASCII 串。格式 (FMT) 指定右对位的转换精度，十进制对位是逗号或间隔。转换的结果放在 OUT 指定的连续 12 个字节中。

使 ENO = 0 的出错条件是： SM4.3 (运行时间) ； 0006 (间接寻址) ； 无输出 (非法格式)

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SD, SMD, LD, HC, 常数, AC, *VD, *AC, *LD	DINT
FMT	VB, IB, QB, MB, SMB, LB, AC, 常数, *VD, *AC, SB, *LD	BYTE
OUT	VB, IB, QB, MB, SMB, LB, *VD, *AC, SB, *LD	BYTE

DTA (双整数到 ASCII 码转换) 指令的格式操作数 (FMT) 定义如图 9-49 所示。输出缓冲区的大小始终是 12 个字节。nnn 区指定输出缓冲区中的十进制对位右边的位数。nnn 区的有效范围是 0 到 5。指定十进制右对位为 0，表示显示的值没有小数位。对于大于 5 的 nnn，输出缓冲区用 ASCII 空格填充。位 c 指定是用逗号 (c=1) 或小数点 (c=0) 作为整数和小数的分割符。高 4 位必须为 0。输出缓冲区按照下面的规则进行格式化：

1. 正值不带符号写入输出缓冲区。
2. 负值带负号写入输出缓冲区。
3. 小数点左边前的 0 进行删除处理 (除非临近小数点的数字 0)。
4. 在缓冲区中数值采用右对齐。

图 9–49 是采用小数点 (c = 0) 进行格式化的数的格式，在小数点右边有三位数字 (nnn =100)。

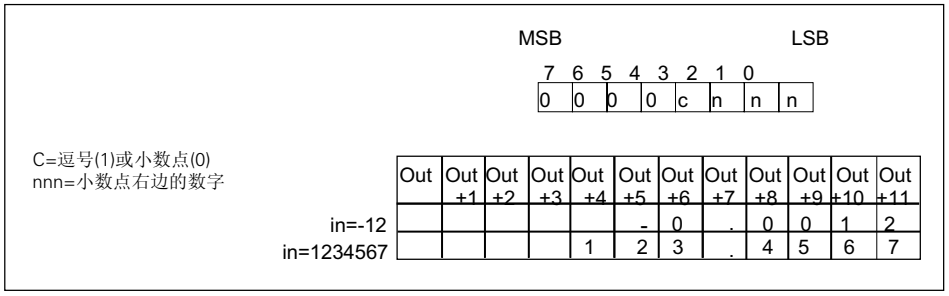
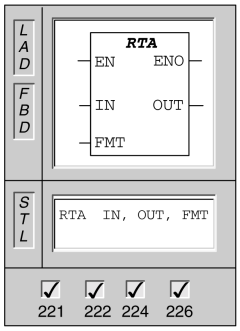


图 9-49 DTA 指令的 FMT 操作数

实数到 ASCII 码



实数到 ASCII 指令把输入端 (IN) 的整数转换成一个 ASCII 串。格式 (FMT) 指定右对位的转换精度，十进制对位是小数点或间隔。转换的结果放在 OUT 指定的连续 3 到 15 个字节中。

使 ENO = 0 的出错条件是： SM4.3 (运行时间) ；0006 (间接寻址) ；无输出 (非法格式)

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD	REAL
FMT	VB, IB, QB, MB, SMB, LB, AC, 常数, *VD, *AC, SB, *LD	BYTE
OUT	VB, IB, QB, MB, SMB, LB, *VD, *AC, SB, *LD	BYTE

RTA (双整数到 ASCII 码转换) 指令的格式操作数 (FMT) 定义如图 9-50 所示。输出缓冲区的大小由 ssss 区的值指定，0、1 或 2 个字节是不允许的。nnn 区指定输出缓冲区中的十进制对位右边的位数。nnn 区的有效范围是 0 到 5。指定十进制右对位为 0，表示显示的值没有小数位。对于大于 5 的 nnn，输出缓冲区用 ASCII 空格填充。位 c 指定是用逗号 (c=1) 或小数点 (c=0) 作为整数和小数的分割符。高 4 位必须为 0。输出缓冲区按照下面的规则进行格式化：

1. 正值不带符号写入输出缓冲区。
2. 负值带负号写入输出缓冲区。
3. 小数点左边前的 0 进行删除处理 (除非临近小数点的数字 0)。
4. 小数点右边的值是小数点右边的数的位数。
5. 输出缓冲区的大小必须不小于 3 个字节，还要大于小时点右边的位数。
6. 在缓冲区中数值采用右对齐。

图 9–50 是采用小数点 (c = 0) 进行格式化的数的格式，在小数点右边有三位数字 (nnn =001)，缓冲区的大小是 6 个字节 (ssss=0110)。

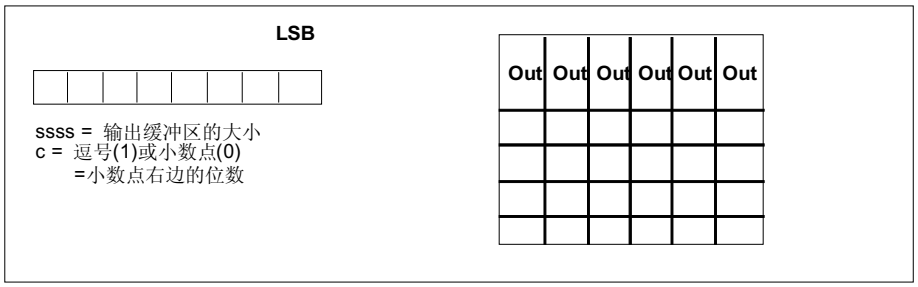


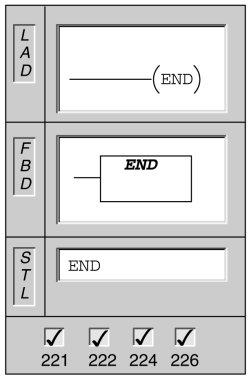
图 9-50 RTA 指令的 FMT 操作数

注意：

S7-200 CPU 采用的浮点数格式最大支持 7 位有符号数，试图显示大于 7 位有符号数将产生一个错误。

9.14 SIMATIC 程序控制指令

有条件结束 (END)



有条件结束指令 (END) 可以根据前面的逻辑关系，终止用户主程序。

操作数：无

数据类型：无

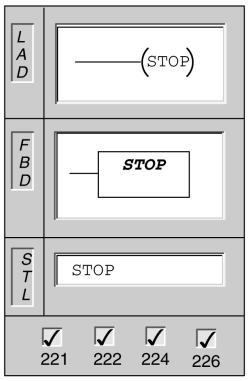
注意：

可以用在主程序中使用有条件结束语句，但是不能在子程序或中断程序中使用。

注意：

Micro/WIN 32 自动在主程序结束加上一个无条件结束。

暂停 (STOP)

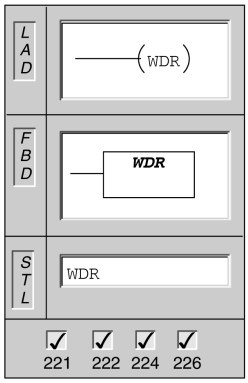


暂停指令 (STOP) 能够引起 CPU 方式发生变化，从 RUN到 STOP，从而可以立即终止程序的执行。

操作数：无

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 (WDR)



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，这就可以增加一次扫描所允许的时间。

操作数：无

有关用 WDR 指令复位看门狗定时器的几点考虑：

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，10 ms 和 100 ms 定时器将不会正确累计时间。
- 在中断程序中的 STOP 指令

注意：

如果希望扫描超过 300ms，或者希望中断事件而该中断事件能使扫描时间大于300ms，那么样最好 WDR 指令来重新触发看门狗定时器。

如果将 S7-200 CPU方式开关切到 STOP 位置，则在 1.4 秒里 CPU 转到STOP方式。

Stop, End, and WDR Example

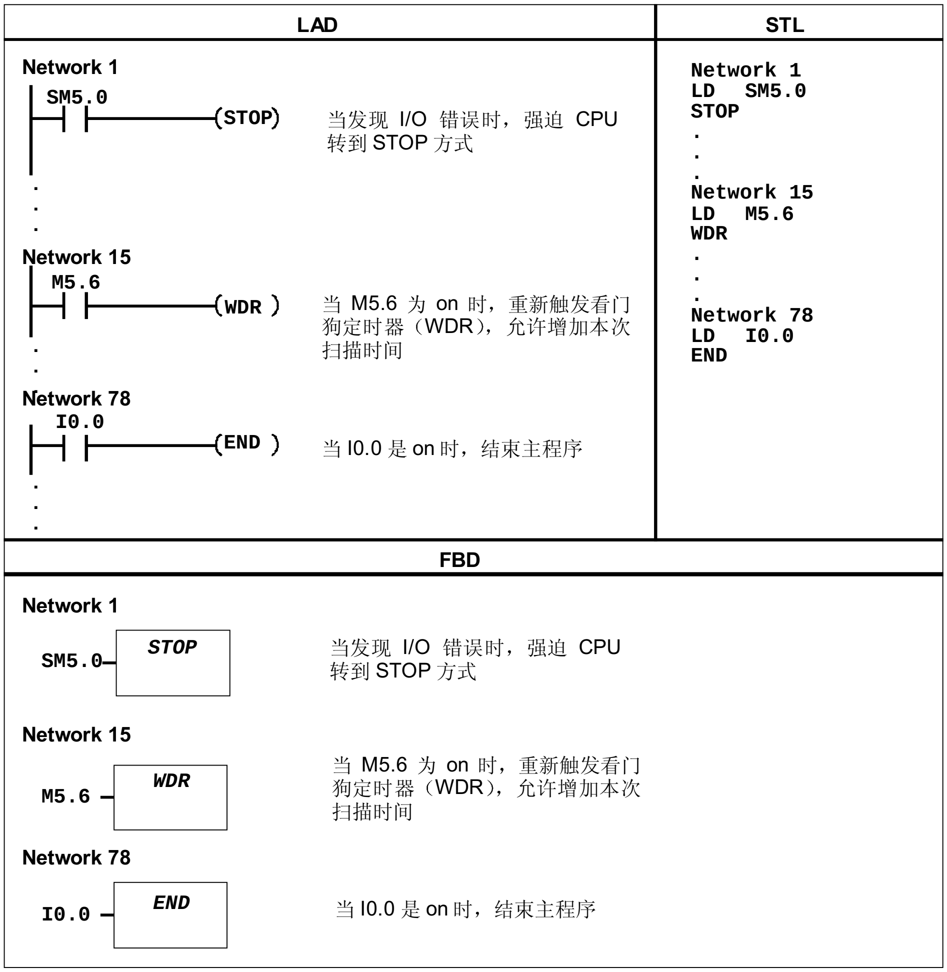
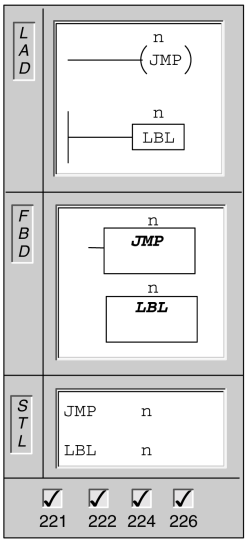


图 9-51 STOP，END 和 WDR 指令在 LAD、FBD 和 STL 中应用举例

跳转及标号指令



跳转指令 (JMP) 可使程序流程转到同一程序中的具体标号 (n) 处，当这种跳转执行时，栈顶的值总是逻辑 1。

标号指令 (LBL) 标记跳转目的地的位置 (n)。

操作数：n: 常数 0 到 255

数据类型：WORD

跳转和标号指令必须用在主程序，子程序或中断程序中。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。

跳转及标号指令举例

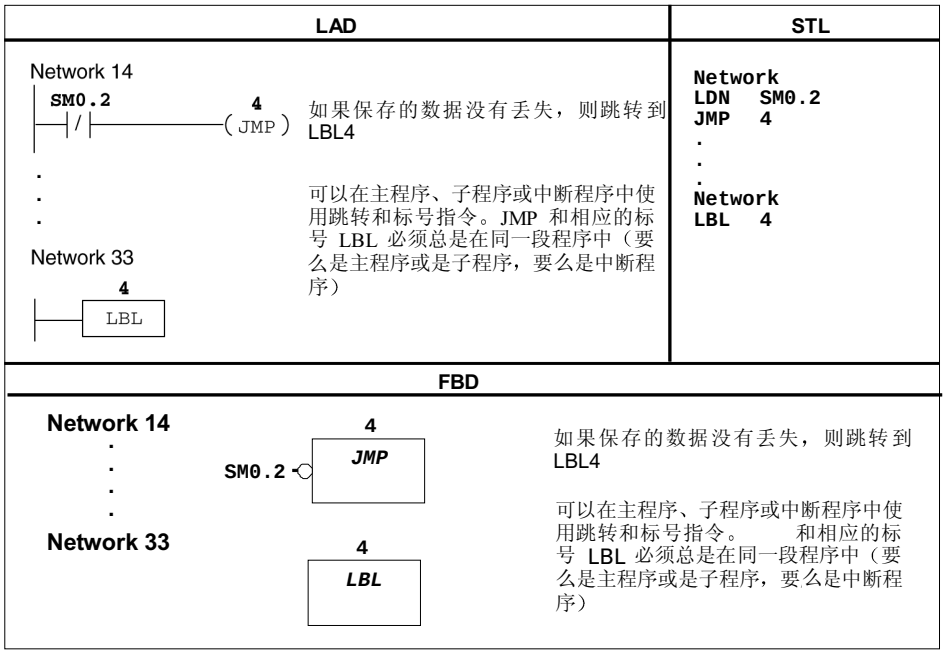
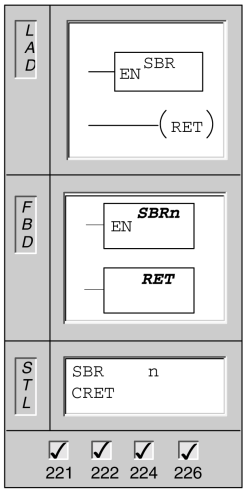


图 9-52 JMP 和 LBL 指令在 LAD、STL 和 FBD 中应用举例

子程序，子程序返回指令



子程序调用指令 (CALL) 把程序控制权交给子程序 (n)。可以带参数或不带参数调用子程序。通过选择 **Edit>Insert> Subroutine** 加入一个子程序。

有条件**子程序返回指令** (CRET) 根据该指令前面的逻辑关系，决定是否终止子程序 (n)。

操作数：无

数据类型：无

执行完子程序以后，控制程序回到子程序调用指令的下一条指令。

图 9–55 是调用子程序并从子程序返回的举例。

使 ENO = 0 的出错条件是： SM4.3 (运行时间) ；0008 (最多子程序嵌套超界)

注意：

Micro/WIN32 为每个子程序自动加入返回指令。

子程序的嵌套深度最多是 8 层，尽管子程序不禁止递归调用 (自己调用自己)，但使用时要慎重。

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

子程序可能包含要传递的参数。参数在子程序的局部变量表中定义 (见图 9-53)。参数必须有一个符号名 (最多 8 个字符) 、变量类型和数据类型。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到子程序 (IN) 、传入和传出子程序 (IN_OUT) 或者传出子程序 (OUT)。参数类型的特征如下所述：

- 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 (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。

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。

	Name	ae	ae	mmet
	EN	N		
L0.0	N1	N		
LB1	N	N	E	
LB2.0	N	N		
LD3	N	N		
LW7	N1	N		
LD9	1			
		E		

图 9-53　STEP 7-Micro/WIN 32 局部变量表

局部变量表中的数据类型区定义了参数的大小和格式。参数类型是：

- 能流：布尔能流仅允许对位输入操作。该变量声明告述 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。图 9-54 中的允许输入 (EN) 和 IN1 输入都使用布尔逻辑。
- 布尔 - 该数据类型用于单独位输入和输出。图 9-54 中的 IN2 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 4 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一个 2 或 4 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。

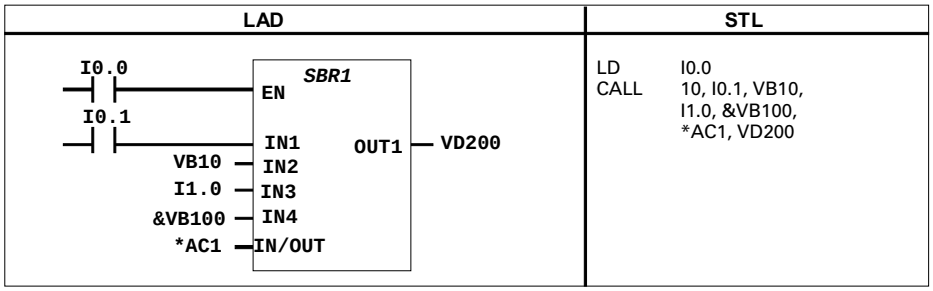


图 9-54　用 LAD 和 STL 表示的子程序调用

图 9-54 中的地址参数 (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，如果局部变量表声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 (见图 9–53) 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，参数值分配给局部存储器，起始地址是 L.0。
- 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

带参数调用子程序使 EN0=0 的错误条件是：

SM4.3 (运行时间)，0008 (子程序嵌套超界)

子程序和从子程序返回举例

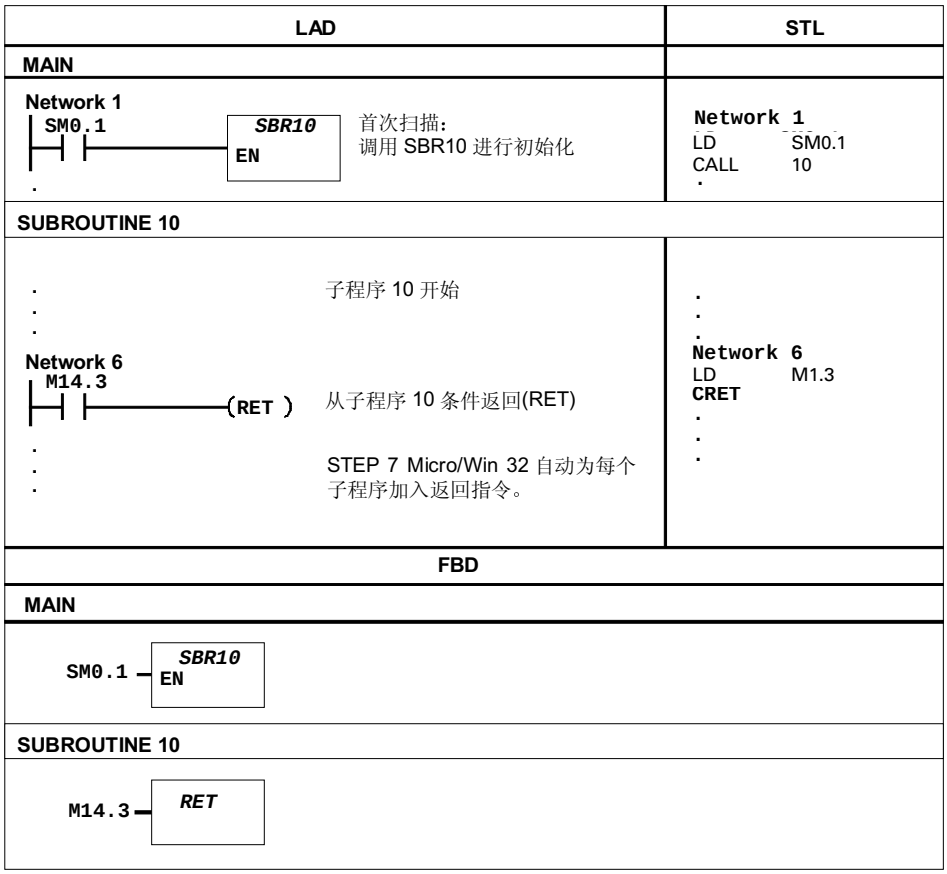
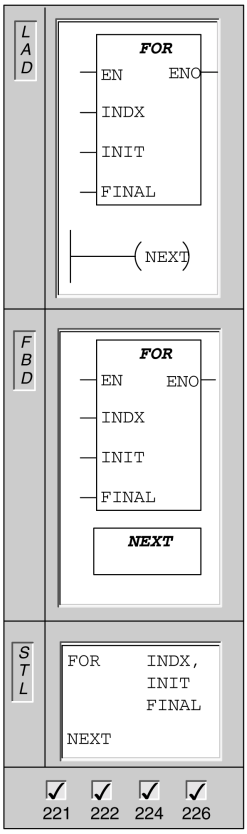


图 9-55 子程序指令的 LAD，STL 和 FBD 举例

循环指令 (For, Next)



FOR 指令和NEXT指令必须成对使用，FOR 标记循环的开始，NEXT标记循环的结束。FOR 标记在 FOR 和 NEXT 之间执行指令，必须给 FOR 指令指定当 前循环计数 (INDX) 、初值 (INIT) 和终值 (FINAL)。

NEXT 指令标记循环的结束，并且置栈顶值为 1。

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。

For: 使 ENO = 0 的出错条件是： SM4.3 (运行时间) ；0006 (间接寻址)

输入/输出	操 作 数	数据类型
INDX	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *AC, *LD	INT
INIT	VW, IW, QW, MW, SW, SMW, T, C, AC, LW, AIW, 常数, *VD, *AC, *LD	INT
FINAL	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, AIW, 常数, *VD, *AC, *LD	INT

这里是一些使用 FOR/NEXT 循环指令的规则：

- 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，它把初始值拷贝到指针值中 (当前循环次数)。当下一次允许时，FOR/NEXT 指令复位它自己。

FOR 和 NEXT 指令可以描述需重复执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 举例

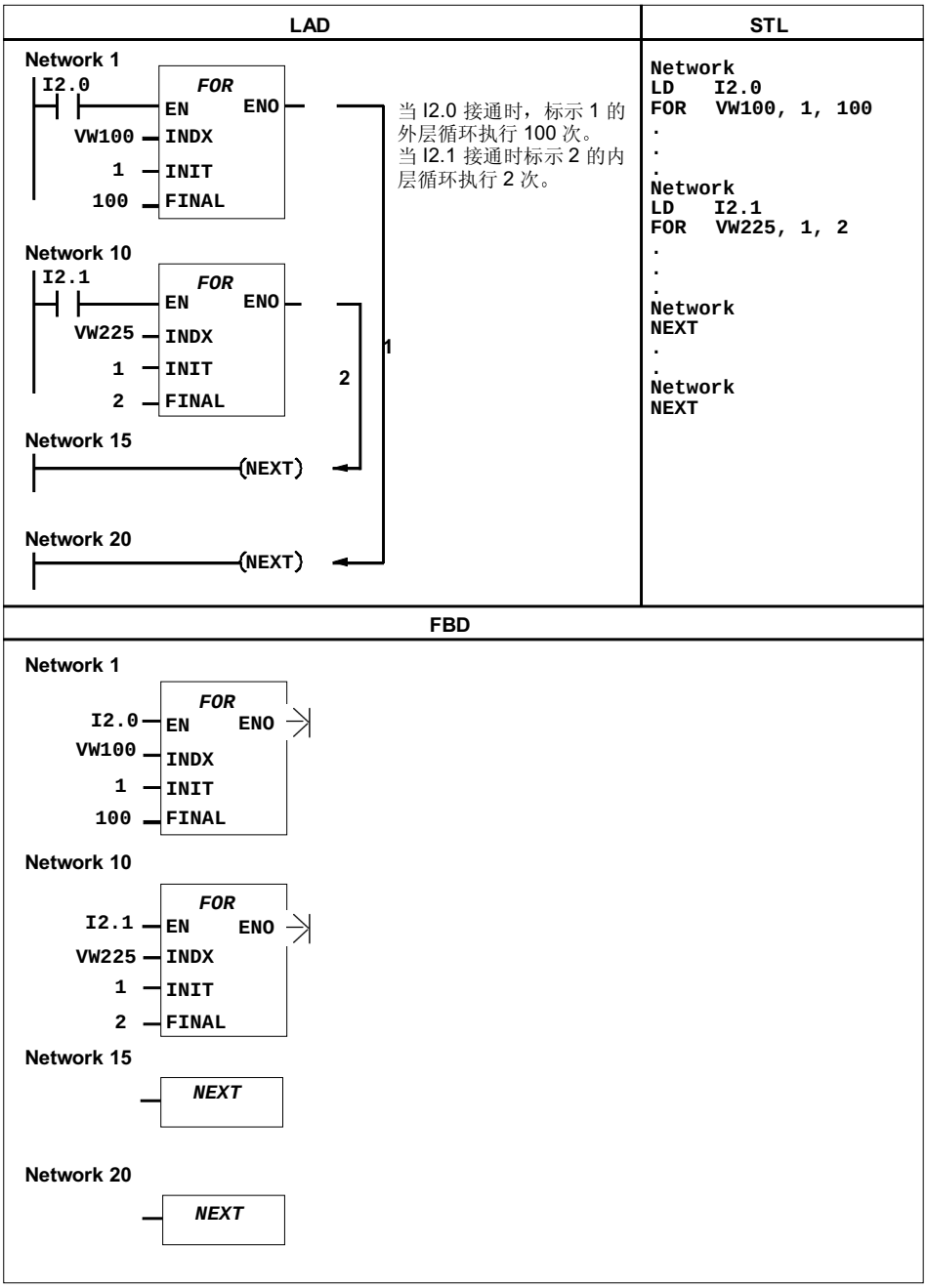
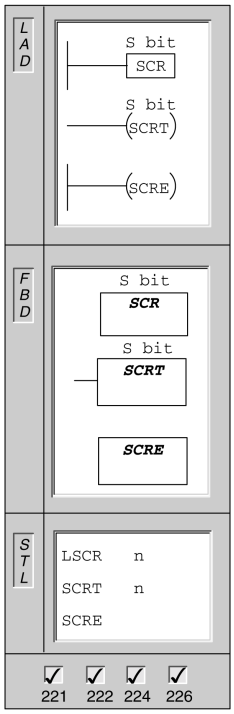


图 9-56　FOR/NEXT 指令的 LAD、FBD 和 STL 举例

顺序控制继电器指令



LSCR 指令标记一个顺序控制继电器 (SCR) 段的开始。当 n=1 时，允许该 SCR 段工作。SCR 段必须用 SCRE 指令结束。

SCRT 指令执行 SCR 段的转移。当 n=1 时，一方面对下一个 SCR 使能位 (S 位) 置位，以便下一个SCR 段工作；另一方面又同时对本 SCR 使能位 (S位)复位，以使本 SCR 段停止工作。

SCRE 指令标示一个 SCR 段的结束。

输入/输出	操 作 数	数据类型
n	S	BOOL

对 SCR 指令的理解

在梯形图和语句表中，SCR 如同程序分段一样组织机器操作步骤。SCR 指令允许对控制程序进行逻辑分段。

LSCR 指令把 S 位的引用放到 SCR 堆栈和逻辑堆栈中。SCR 堆栈的值决定该 SCR 段是否执行。由于逻辑堆栈顶的值是 S 位的引用值，所以能在无触点干涉下，把 LAD 左电流线直接连到 SCR 及其后的输出线圈或方框上。图 9-57 揭示了 LSCR 指令执行后对 S 堆栈和逻辑堆栈的影响。

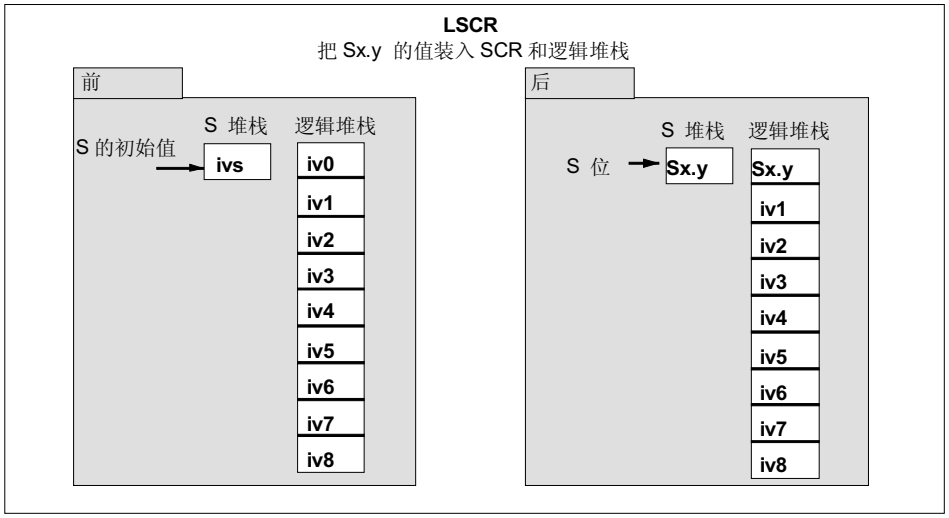


图 9-57　LSCR 对逻辑堆栈的影响

对 SCR 段的正确理解：

- 从 LSCR 开始到 SCRE 结束的所有指令组成 SCR 段，它能否执行取决于 SCR 堆栈的值。SCRE和 下一个 LSCR 之间的指令逻辑与 SCR 堆栈的值无关。
- SCRT 指令一方面对下一个 SCR 使能位置位，以使下一个 SCR 段工作；另一方面又同时对本 SCR 使能位复位，以使本 SCR 段停止工作。

限制

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 段中不能使用 JMP 和 LBL 指令，就是说不允许跳入、跳出或在内部跳转。可以在 SCR 段附近使用跳转和标号指令。
- 在 SCR 段中不能使用 FOR，NEXT 和 END 指令。

SCR 举例

图 9–58 是 SCR 指令的实例。

- 在本例中，首次扫描标志位 SM0.1 用来设置 S0.1，而该位可以在第 一次扫描期间激活状态 1 (SCR 段)。.
- 经过 2 秒的延时，T37 导致转移到状态 2，并使状态 2 的 SCR 段 (S0.2) 工作，同时使状态 1 的 SCR 段 (S0.1) 停止工作。

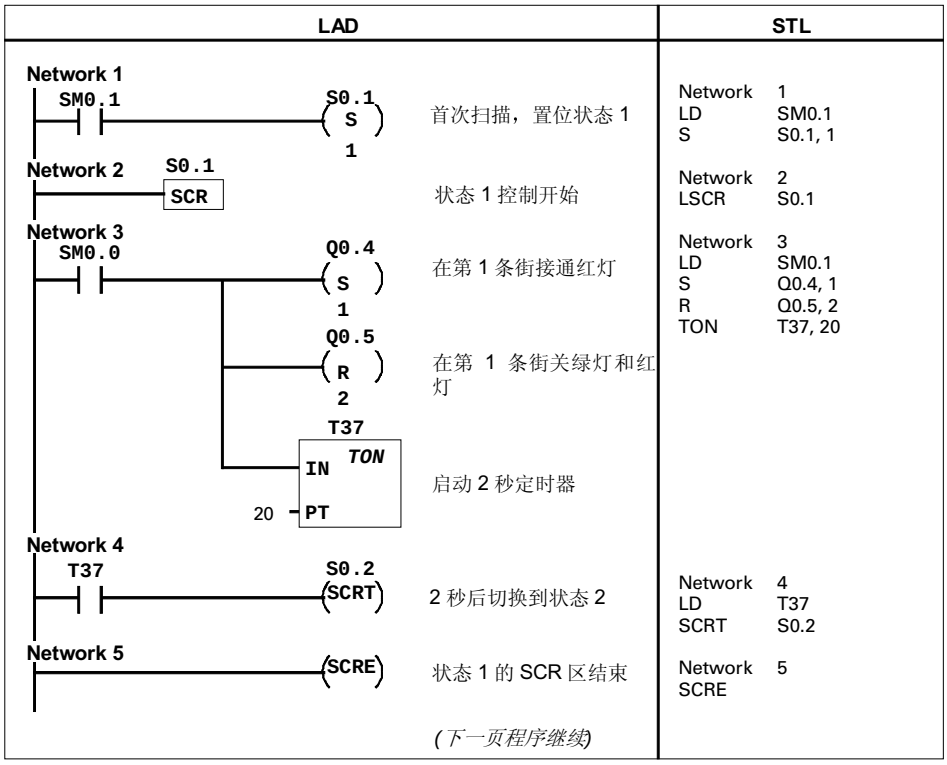


图 9-58 顺序控制继电器 (SCR) 举例

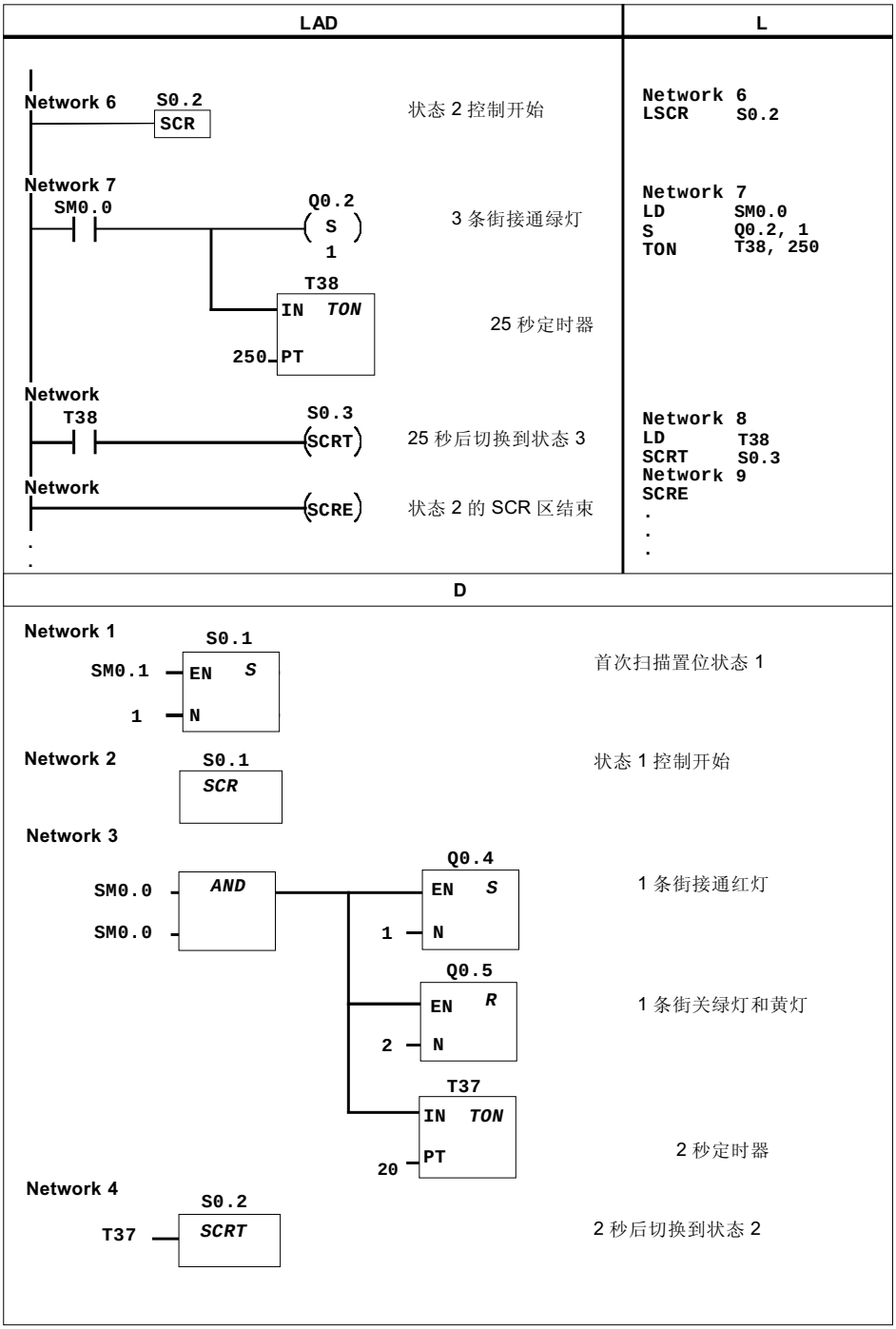


图 9-58　顺序控制继电器 (SCR) 举例 (续)

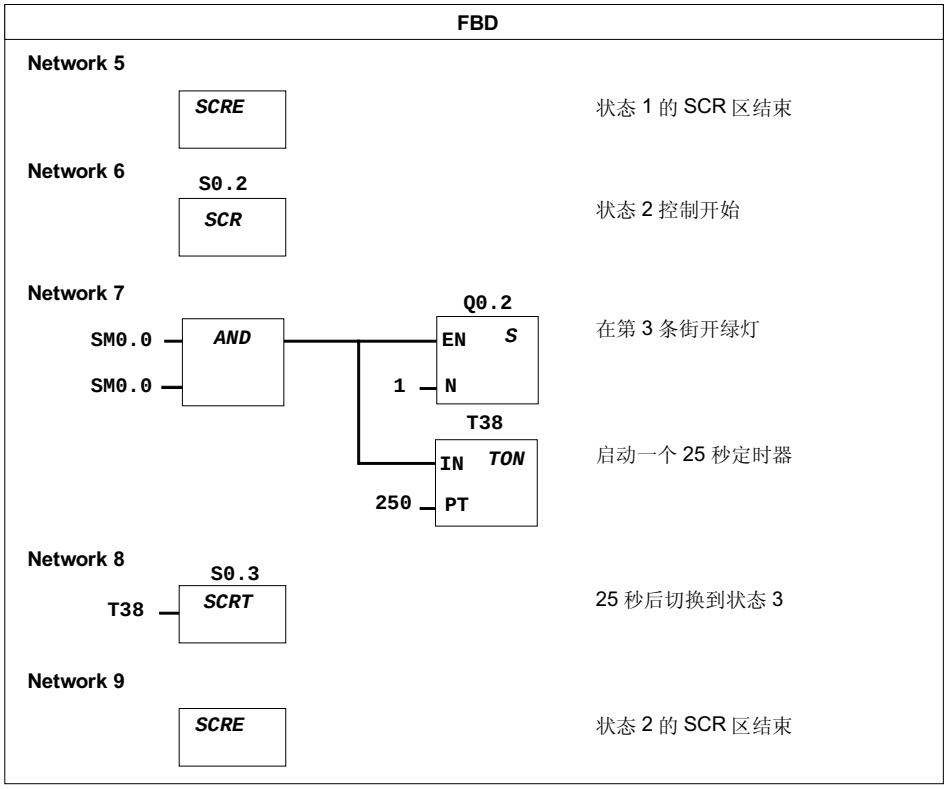


图 9-58　顺序控制继电器 (SCR) 举例 (续)

分支控制

在许多实例中，一个顺序控制状态流必须分成 2 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 9-59 所示。

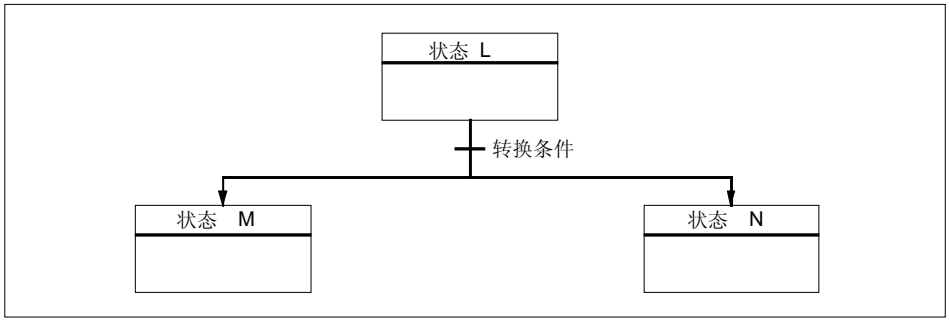


图 9-59　控制流的分支

在同一个转移条件的允许下，使用多条 SCRT 指令可以在一段 SCR 程序中实现控制流的分支，如图 9-60 所示。

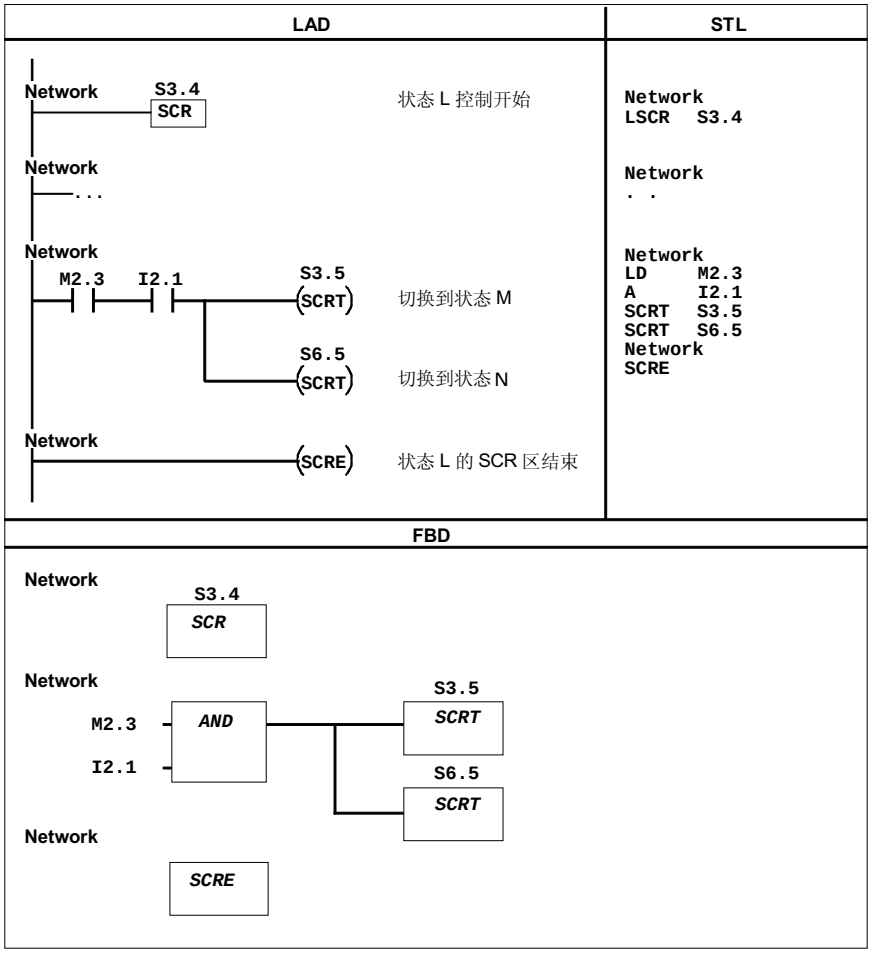


图 9-60　控制流分支举例

合并控制

当多个控制流产生类似结果时，可以把这些控制流合并成一个控制流，我们称之为控制状态流的合并。在合并控制流时，所有的控制流必须都是完成了的，才能执行下一个状态，如图 9-61 所示。

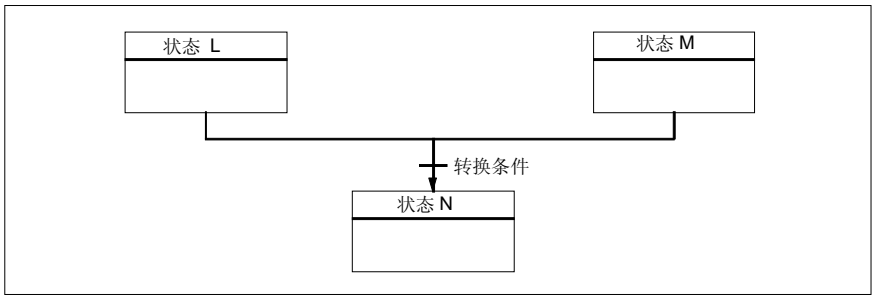


图 9-61　控制流的合并

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，如图 9-62 所示。

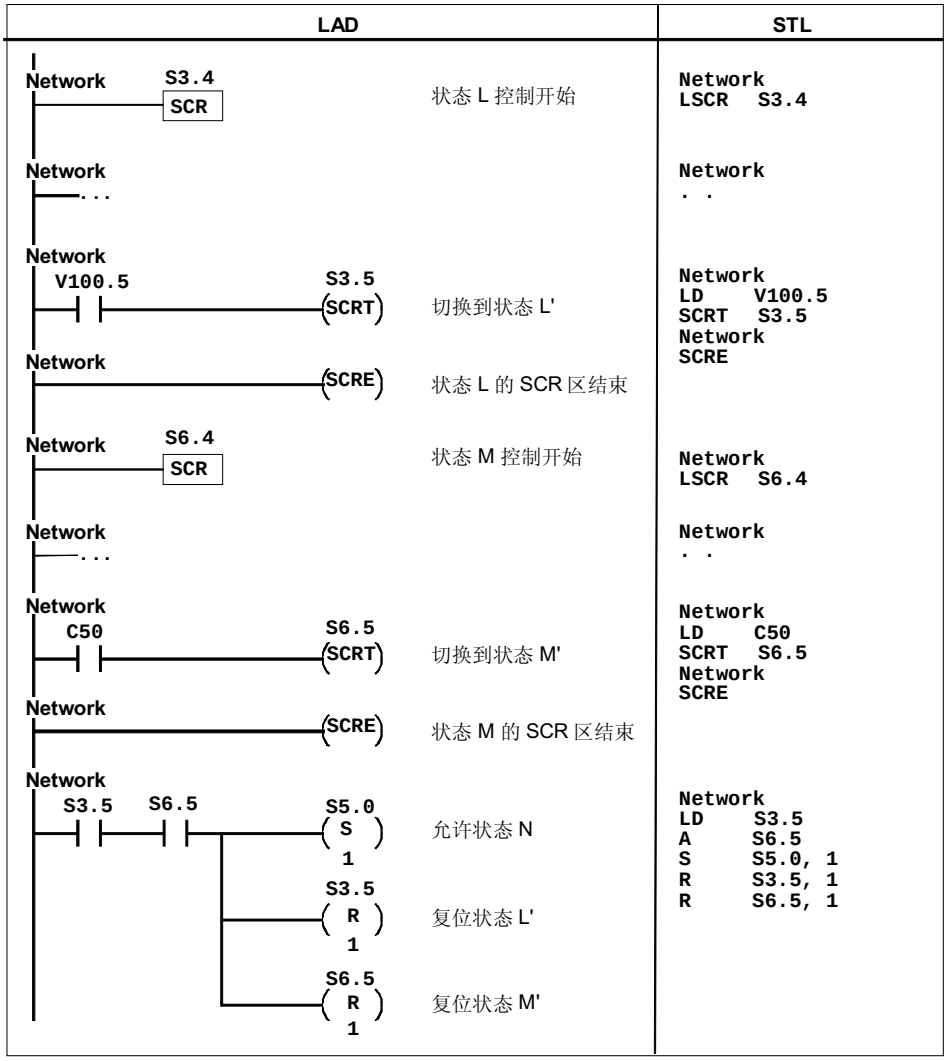


图 9-62　控制流合并举例

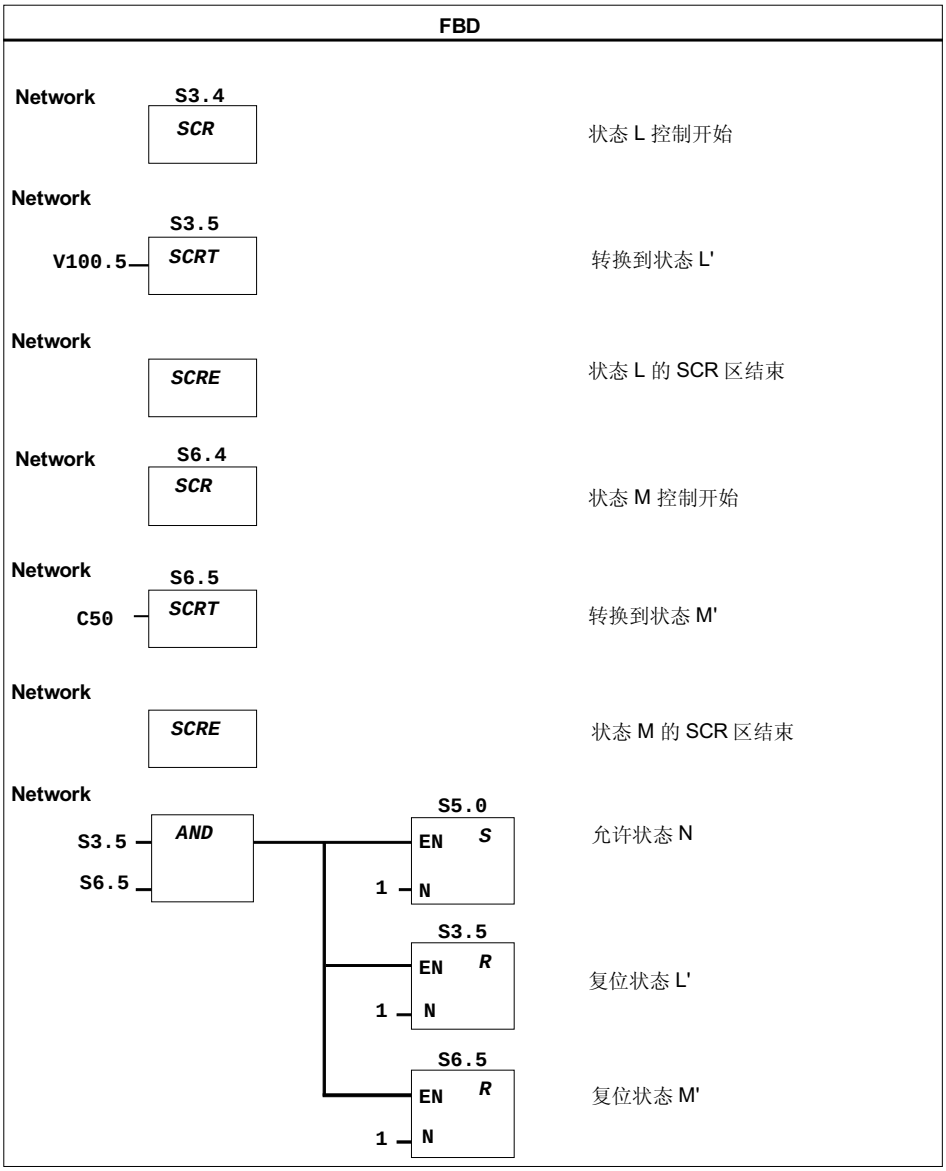


图 9-62　控制流合并举例 (续)

在有些情况下，一个控制流可能转入多个可能的控制流中某一个。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 9-63 所示。

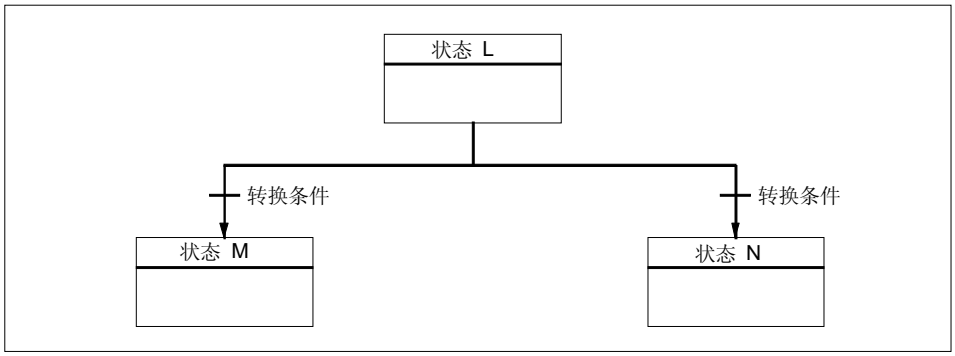


图 9-63　基于转移条件的控制流分支

对应的 SCR 程序如图 9–64 所示。

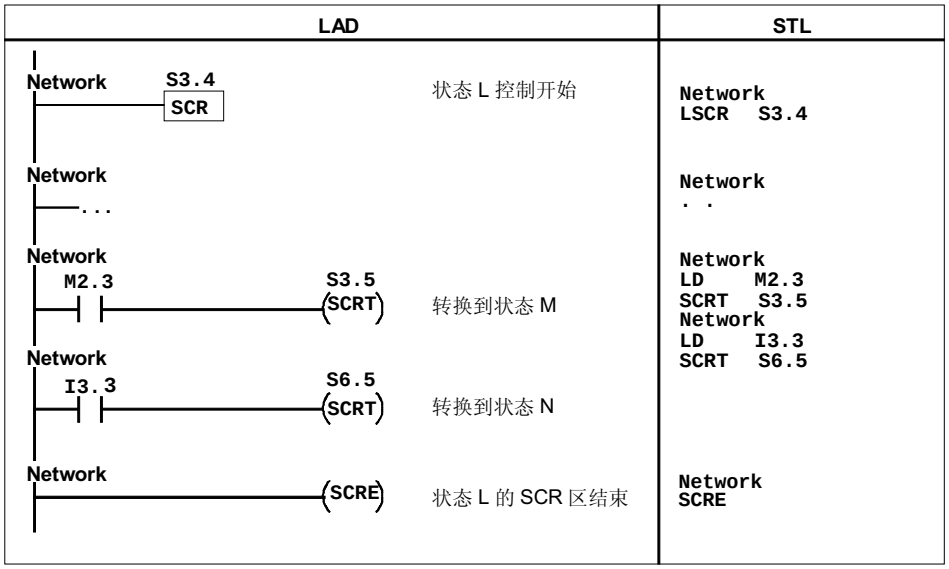


图 9-64　条件转换举例

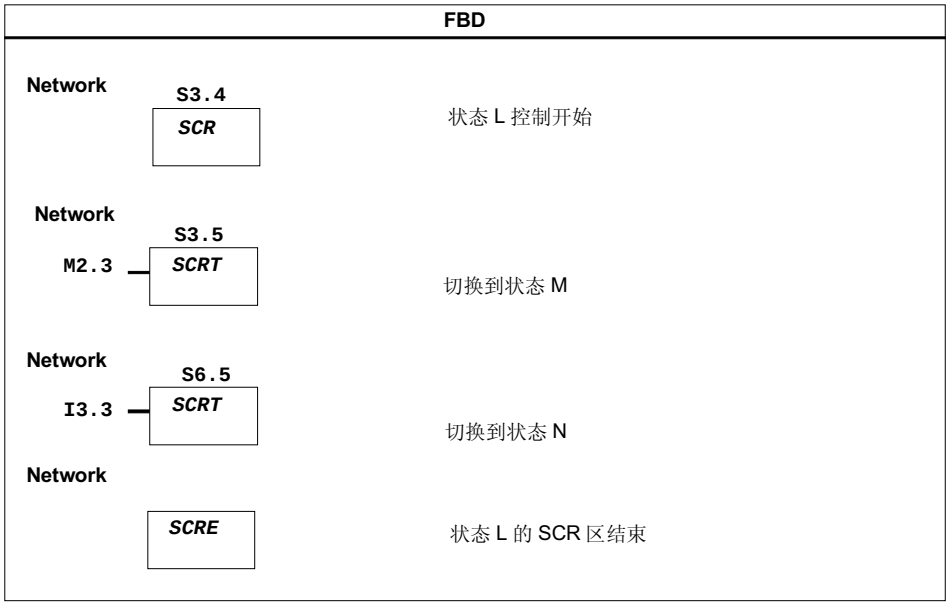
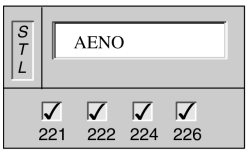


图 9-64　条件转换举例 (续)

ENO



ENO 是 LAD 和 FBD 中指令盒的布尔量输出。如果指令盒的输入有能流，而且执行没有错误，ENO 输出就把能流传到下一个指令盒。ENO 可以作为允许位表示指令成功执行。

借助栈顶，ENO 位用来影响其后指令执行的能流。

STL 指令没有 EN 输入；对于要执行的指令栈顶必须是1。

在 STL 中没有 ENO 输出，但是，带有 ENO 的 LAD 和FBD 指令对应的 STL 指令置一个特殊的 ENO 位。该位用与 **And ENO** (AENO) 指令访问。AENO 可以用来产生和指令盒的 ENO 位同样的效果。AENO 只能在 STL 中使用。

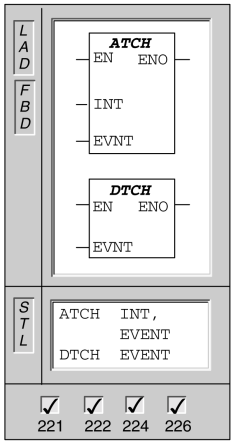
AENO 执行栈顶和 ENO 位的逻辑与，与操作的结果保存在栈顶。

操作数：无

数据类型：无

9.15 SIMATIC 中断和通讯指令

中断连接，中断分离



中断连接指令 (ATCH) 把一个中断事件 (EVNT) 和一个中断程序 (INT) 联系起来，并允许这个中断事件。

中断分离指令 (DTCH) 截断一个中断事件 (EVNT) 和所有的中断程序的联系，并禁止了该中断事件。

中断连接指令：使 ENO = 0 的出错条件是： SM4.3 (运行时间) ；0006 (间接寻址)

输入/输出	操 作 数	数据类型
INT	常数	BYTE
EVNT	常数 (CPU 221/222: 0-12, 19-23, 27-33; CPU 224: 0-23, 27-33; CPU 226: 0-33)	BYTE

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。

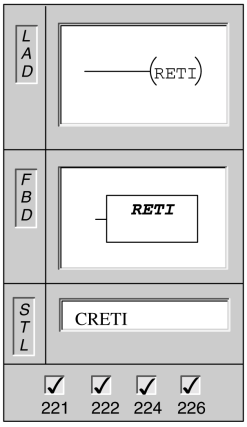
可以用中断分离指令 (DTCH) 截断中断事件和中断程序之间的联系，以单独禁止中断事件。中断分离指令 (DTCH) 使中断回到不激活或无效状态。

表 9–20 列出了中断时间的不同类型。

表 9-20 中断事件

事件号	中断描述	CPU 221	CPU 222	CPU 224	CPU 226
0	上升沿，I0.0	Y	Y	Y	Y
1	下降沿，I0.0	Y	Y	Y	Y
2	上升沿，I0.1	Y	Y	Y	Y
3	下降沿，I0.1	Y	Y	Y	Y
4	上升沿，I0.2	Y	Y	Y	Y
5	下降沿，I0.2	Y	Y	Y	Y
6	上升沿，I0.3	Y	Y	Y	Y
7	下降沿，I0.3	Y	Y	Y	Y
8	端口0：接收字符	Y	Y	Y	Y
9	端口0：发送字符	Y	Y	Y	Y
10	定时中断0，SMB34	Y	Y	Y	Y
11	定时中断1，SMB35	Y	Y	Y	Y
12	HSC0 CV=PV (当前值=预置值)	Y	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	HSC2 CV=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	PLS0 脉冲数完成中断	Y	Y	Y	Y
20	PLS1 脉冲数完成中断	Y	Y	Y	Y
21	定时器 T32 CT=PT 中断	Y	Y	Y	Y
22	定时器 T96 CT=PT 中断	Y	Y	Y	Y
23	端口0：接收信息完成	Y	Y	Y	Y
24	端口1：接收信息完成			Y	Y
25	端口1：接收字符			Y	Y
26	端口1：发送字符			Y	Y
27	HSC0 输入方向改变	Y	Y	Y	Y
28	HSC0 外部复位	Y	Y	Y	Y
29	HSC4 CV=PV (当前值=预置值)	Y	Y	Y	Y
30	HSC4 输入方向改变	Y	Y	Y	Y
31	HSC4 外部复位	Y	Y	Y	Y
32	HSC3 CV=PV (当前值=预置值)	Y	Y	Y	Y
33	HSC5 CV=PV (当前值=预置值)	Y	Y	Y	Y

中断返回指令



有条件**中断返回指令**可以用来根据逻辑操作的条件从中断程序中返回。从菜单 **Edit>Insert>Interrupt** 中加入一个中断。

操作数：无

数据类型：无

从中断程序的返回可以从 STEP 7-Micro/WIN 32 中的指令表中选择。

中断程序

你可以用中断程序入口点处的中断程序标号来识别每个中断程序。中断程序由位于中断程序标号和无条件中断返回指令间的所有指令组成。中断程序在响应与之关联的内部或外部中断事件时执行。你可以用无条件中断返回指令 (RETI) 或条件中断返回指令 (CRETI) 退出中断程序 (从而将控制还给主程序)，而无条件中断返回指令是必需的。

中断使用指南

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”。

限制

在中断程序总不能使用 DISI、ENI、HDEF、LSCR 和 END 指令。

系统对中断的支持

由于中断指令影响接点、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。

在中断程序中调用子程序

从中断程序中可以调用一个嵌套子程序。累加器和逻辑堆栈在中断程序和被调用的子程序中是共用的。

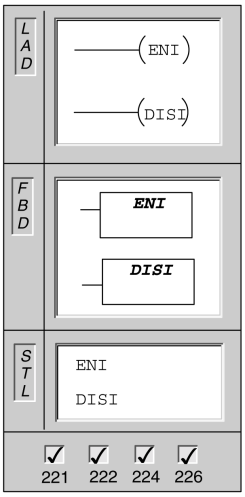
在主程序和中断程序间共享数据

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。

中断允许，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 (DISI) 全局地禁止处理所有中断事件。

操作数：无

数据类型：无

当进入 RUN 模式时，就禁止了中断。而在 RUN 模式，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) 允许中断事件排队等候，但不允许激活中断子程序。

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。

表 9-21 支持的上升/下降沿中断

I/O 中断	S7–200 CPU
I/O 点	I0.0 到 I0.3

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 5 ms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)。请参见图 9-66 使用定时中断的举例。

定时器 T32/T96 中断允许及时地响应一个给定时间间隔到。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。

对中断优先级和排队的理解

中断按以下固定的优先级顺序优先执行：

- 通讯 (最高优先级)
- I/O 中断
- 时基中断 (最低优先级)

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。三个中断队列及其能保存的最大中断个数如表 9-22 所示。

表 9-22　　中断队列和每个队列的最大中断数

队　列	CPU 221	CPU 222	CPU 224	CPU 226
通讯中断队列	4	4	4	8
I/O 中断队列	16	16	16	16
定时中断队列	8	8	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 9-23 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 9-23　　用于中断队列溢出的特殊存储器位的定义

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 9–24 给出了中断事件、优先级和分配的事件号。

表 9-24 按优先级排列的中断事件

事件号	中断描述	优先组	优先组中的优先
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

中断指令举例

图 9–65 给出了中断程序指令的实例

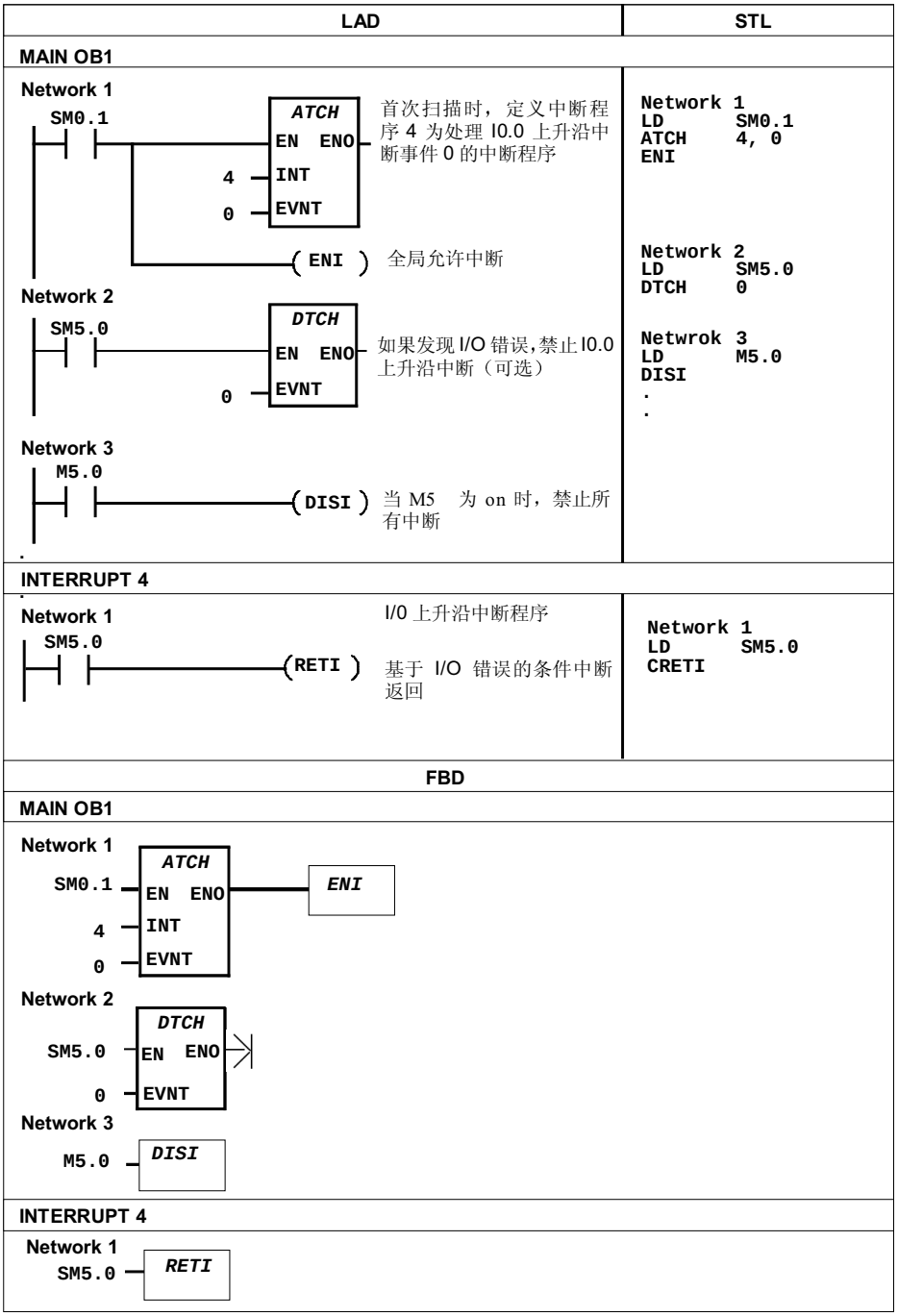


图 9-65 中断指令举例

图 9–66 给出了如何设置定时中断去读取模拟量输入值。

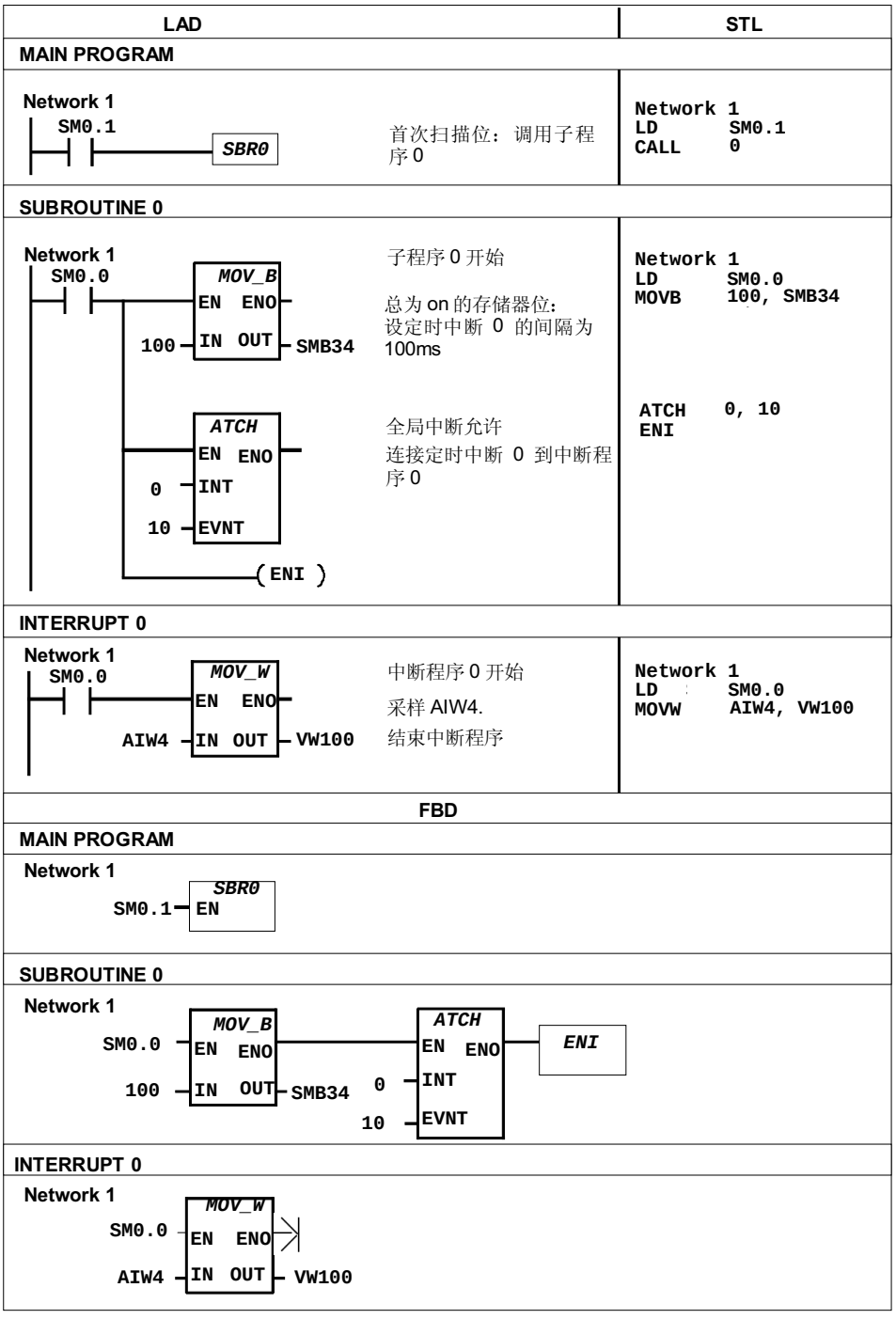
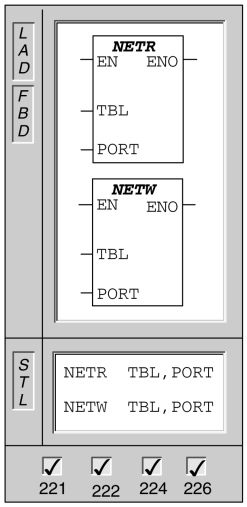


图 9-66　如何设置一个定时中断去读取一个模拟量输入值举例

网络读，网络写



网络读指令 (NETR) 初始化通讯操作，通过指令端口 (PORT) 从远程设备上接收数据并形成表 (TBL)。

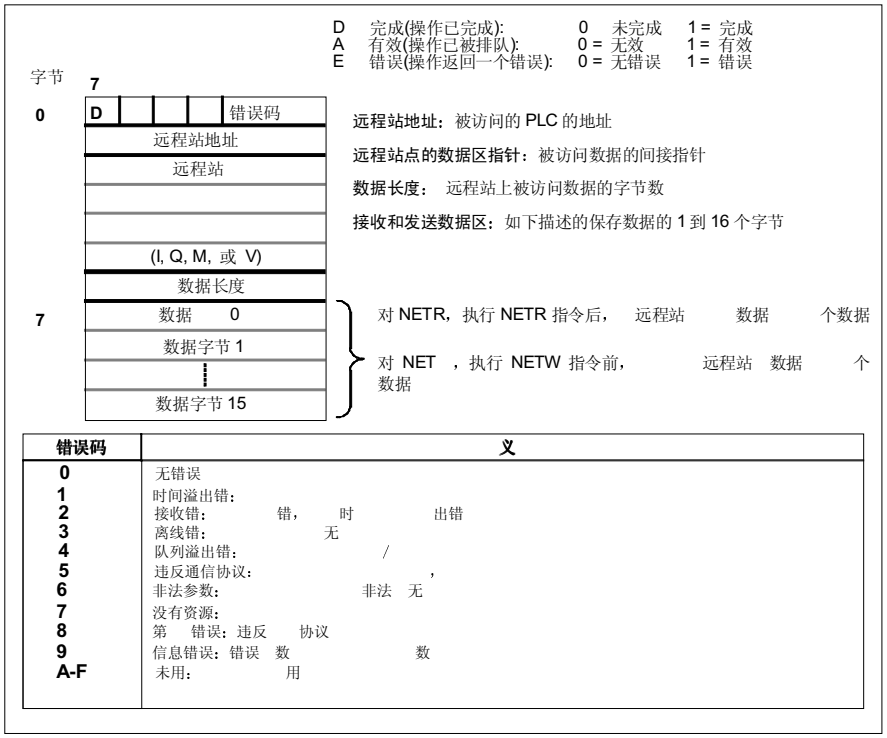
网络写指令 (NETW) 初始化通讯操作，通过指定端口 (PORT) 向远程设备写表 (TBL) 中的数据。

NETR 指令可以从远程站点上读最多 16个 字节的信息，NETW 指令则可以向远程站点写最多 16 个字节的信息。任何同一时间，只能有最多为 8 条 NETR 和 NETW 指令有效。例如，在所给的 S7–200 PLC中，可以有 4 条 NETR 指令和 4 条 NETW 指令，或 2 条 NETR 指令和 6 条 NETW 指令。图 9-67 定义了在 NETR 和 NETW 指令中的TBL 参数参照表。

NETR: 使 ENO = 0 的错误条件为：
SM4.3 (运行时间)，0006 (间接寻址)

NETW: 使 ENO = 0 的错误条件为：
SM4.3 (运行时间)，0006 (间接寻址)

输入/输出	操 作 数	数据类型
TBL	VB, MB, *VD, *AC, *LD	BYTE
PORT	常数	BYTE



错误码	义
0	无错误
1	时间溢出错：
2	接收错： 错， 时 出错
3	离线错： 无
4	队列溢出错： /
5	违反通信协议： ,
6	非法参数： 非法 无
7	没有资源：
8	第 错误：违反 协议
9	信息错误：错误 数 数
A-F	未用： 用

图 9-67 NETR 和 NETW 指令所用 TABLE 的定义

网络读和网络写举例

图 9–68 给出一个例子以解释 NETR 和 NETW 指令的使用。本示例中，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 CPU 221 模块用于控制打包机 ，一个 CPU 222 模块安装了 TD 200 操作器接口，被用来控制分流机。图 9–68 给出了该网络配置。

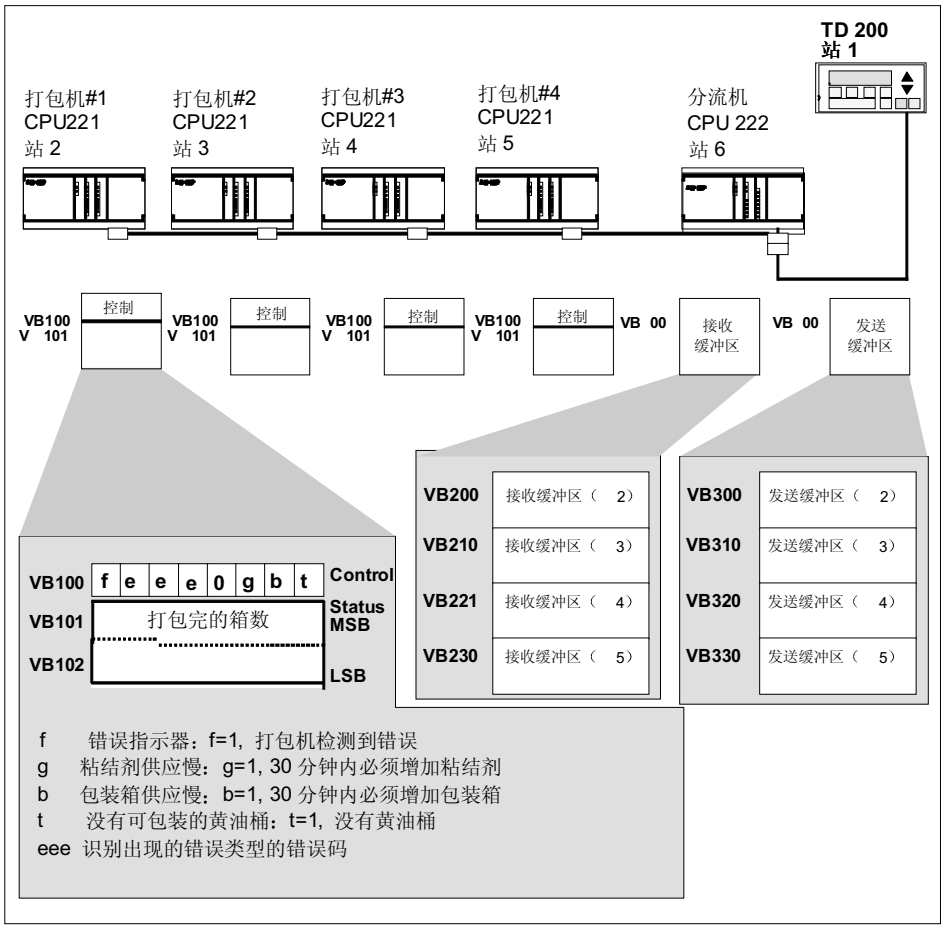


图 9-68 NETR 和 NETW 指令举例

图 9-69 详细地给出了访问站 #2 数据 (分别在 VB200 和 VB300 中) 的接收和发送缓冲区。

CPU 224 用NETR指令连续地读取每个打包机的控制和状态信息。每当某个打包机包装完 100 箱，分流机会注意到这件事，并用NETW指令发送一条信息清除状态字。

程序对单个打包机 (打包机 #1) 需要读取控制字节、包装完的箱数和复位包装完的箱数。请见图 9-70 的图示。

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 9-69　　用于 NETR 和 NETW 指令的 TABLE 数据举例

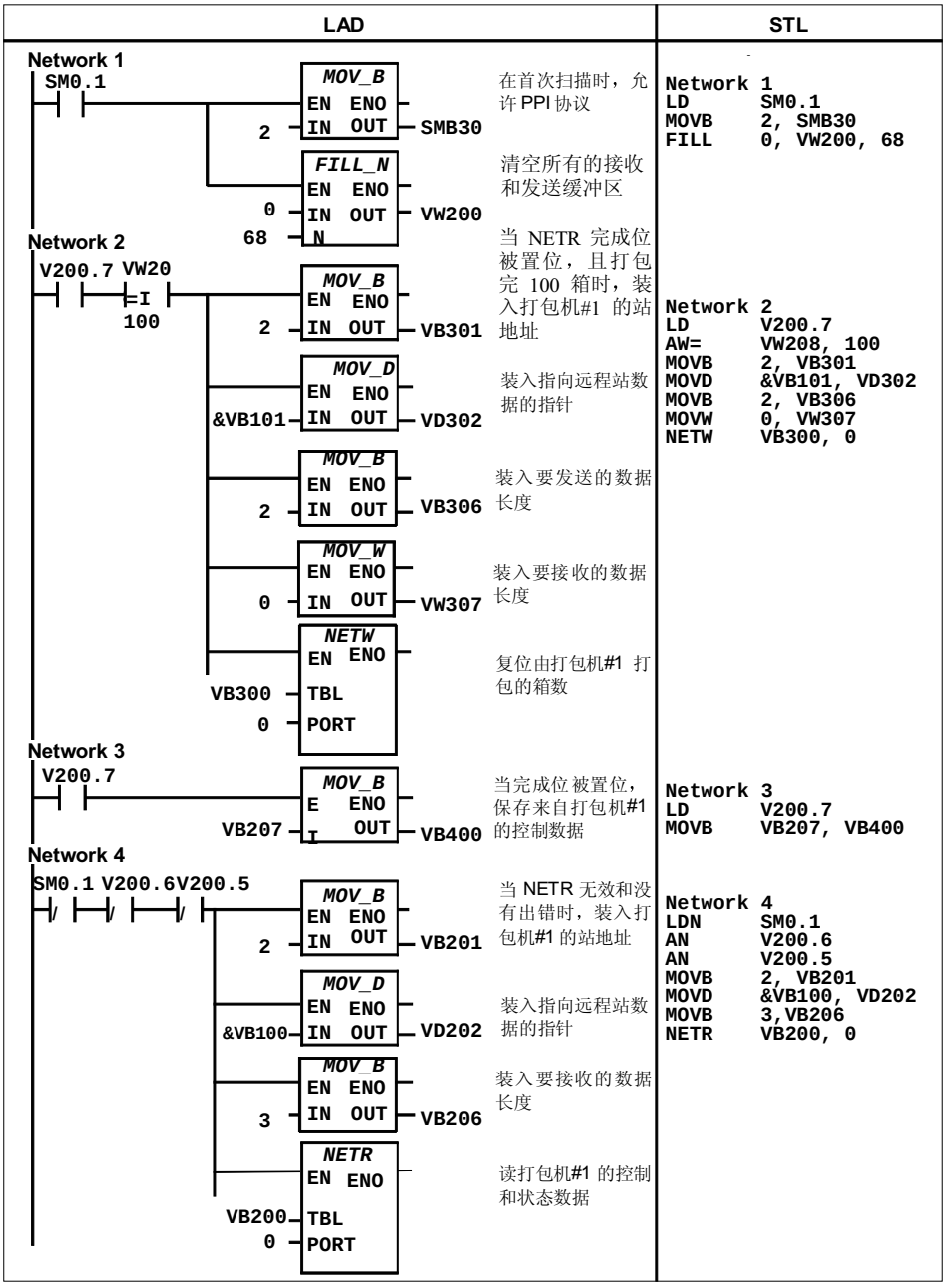


图 9-70　　NETR 和 NETW 指令的 LAD 和 STL 举例

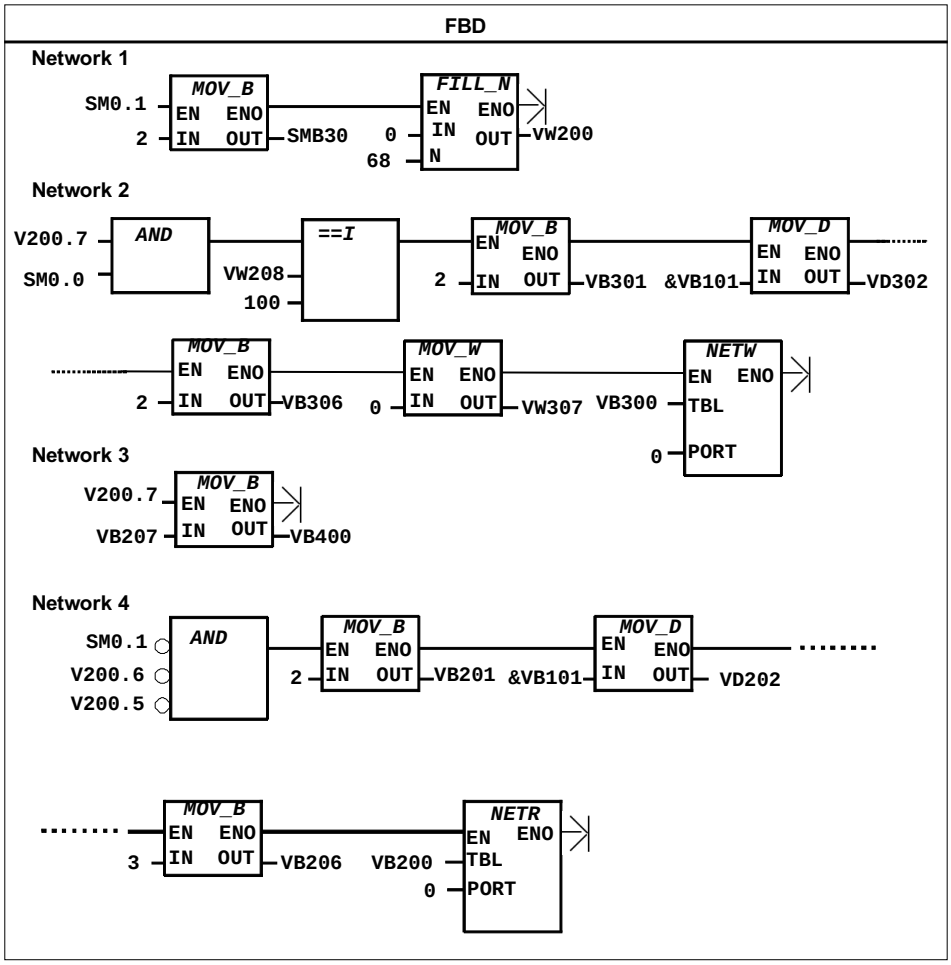
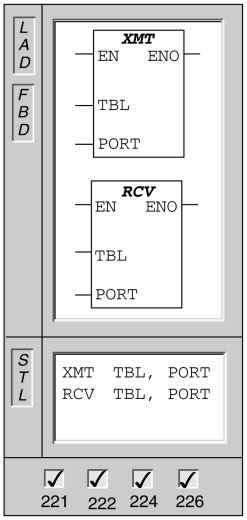


图9-71　　NETR 和 NETW 指令的 FBD 举例

发送，接收



发送指令 (XMT) 激活发送数据缓冲区 (TBL) 中的数据。数据缓冲区的第一个数据指明了要发送的字节数。PORT 指定了用于发送的端口。

XMT指令用于自由端口模式，由通讯端口发送数据。

接收指令 (RCV) 激活初始化或结束接收信息的服务。通过指定端口 (PORT) 接收的信息存储于数据缓冲区 (TBL)。数据缓冲区的第一个数据指明了接收的字节数。

Transmit: 使 ENO = 0 的错误条件是：
SM4.3 (运行时间)； 0006 (间接寻址)；
0009 (在端口 0 同时 XMT/RCV)；
000B (在端口 1 同时 XMT/RCV)

Receive: 使 ENO = 0 的错误条件是：SM86.6 和 SM186.6 (RCV 参数错误)；SM4.3 (运行时间)；0006 (间接寻址)；0009 (在端口 0 同时 XMT/RCV)； 000B (在端口 1 同时 XMT/RCV)

输入/输出	操 作 数	数据类型
TABLE	VB, IB, QB, MB, SB, SMB, *VD, *AC, *LD	BYTE
PORT	常数 (CPU 221、CPU 222 和 CPU 224 为 0；CPU 226 为 0或1)	BYTE

对自由端口模式的理解

CPU 的串行通讯口可由用户程序控制，这种操作模式称为自由端口模式。当选择了自由端口模式，梯形图程序可以使用接收中断、发送中断、发送指令 (XMT) 和接收指令 (RCV) 来控制通讯操作。在自由端口模式下，通讯协议完全由梯形图程序控制。SMB30 (用于端口 0) 和 SMB31 (如果 CPU 有两个端口，则用于端口 1) 用于选择波特率和奇偶校验。

当 CPU 处于 STOP 模式，自由端口模式被禁止，重新建立正常的通讯(如可编程设备的访问)。

简单的情况下，可以只用发送指令 (XMT) 向打印机或显示器发送信息。其他例子包括同条码阅读器、重量计和焊机连接。每种情况下，都必须编写程序，以支持自由端口模式下设备同 CPU 通讯的协议。

只有 CPU 处于 RUN 模式时，才能进行自由端口通讯。通过向 SMB30 (端口 0) 或 SMB130 (端口 1) 的协议选择区置 1，可以允许自由端口模式。处于自由端口模式时，不能与可编程设备通讯。

注意：

可以用反映 CPU 工作方式的模式开关当前位置的特殊存储器位 SM0.7 控制自由端口模式的进入。当 SM0.7为 0 时，模式开关处于TREM位置；当 SM0.7 为 1 时，模式开关处于 RUN 位置。只有模式开关位于 RUN 位置，才允许自由端口模式，为使用可编程设备监视或控制 CPU 操作，可以把模式开关改变到任何其它位置。

自由端口的初始化

SMB30 和 SMB130 分别配置通讯端口 0 和 1，为自由端口通讯选择波特率，奇偶校验和数据位数。自由端口的控制字节描述如表 9-25 所示。

表 9-25 特殊存储器位 SMB30 和 SMB130

端口 0	端口 1	描 述
SMB30 格式	SMB130 格式	MSB LSB p p d b b b m m 自由口模式控制字节
SM30.6 和 SM30.7	SM130.6 和 SM130.7	pp 奇偶选择 00 = 无奇偶校验 01 = 偶校验 10 = 无奇偶校验 11 = 奇校验
SM30.5	SM130.5	d 每个字符的数据位 0 = 每个字符8位 1 = 每个字符7位
SM30.2 到 SM30.4	SM130.2 到 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 = 600 波特 111 = 300 波特
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 (缺省设置为 PPI/ 从站模式)
注：每个配置都有一个停止位。		

用XMT指令发送数据

可以用 XMT 指令方便地发送数据。XMT 指令发送一个或多个字符，最多有 255个字符的缓冲区。如果有一个中断程序连接到发送结束事件上，在发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)。可以监视发送完成状态位 SM4.5 或 SM4.6 的变化，而不是用中断进行发送 (如向打印机发送信息)。

通过把字符数设置为 0，然后执行 XMT 指令，可以产生一个 BREAK 条件。以当前的波特率在线上产生一个 16 位的 BREAK 条件。发送任何其他信息时，和发送其他信息一样发送 BREAK。当 BREAK 完成时，产生一个XMT中断。SM4.5 或 SM4.6 反映XMT的当前状态。

XMT 缓冲区的格式如图 9–72 所示

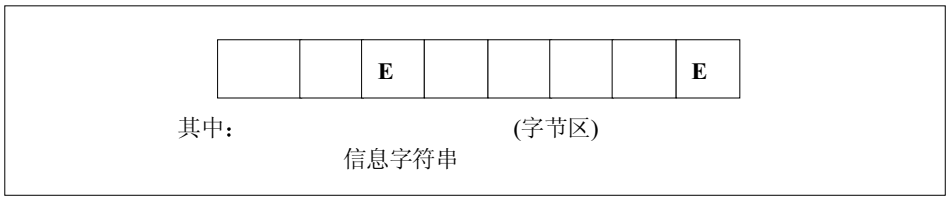


图 9-72 XMT 缓冲区格式

用 RCV 指令接收数据

可以用 RCV 指令方便地接收信息。RCV 指令可以接收一个或多个字符，最多有255 个字符，这些字符存储在缓冲区中。如果有一个中断程序连接到接收完成事件上，在接收到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 23，对端口 1 为中断事件 24)。

可以监视 SMB86 或 SMB186 状态的变化，而不是用中断进行信息接收。当 RCV 邮箱不启动或已经接收结束时，SMB86 或 SMB186 不为 0；当正在接收时，它们为 0。

RCV 指令允许选择信息开始和信息结束条件。信息开始和结束条件的描述如表 9-26 所示 (SM86 至 SM94 用于端口 0，SM186 至 SM194 用于端口1)。RCV 缓冲区的格式如图9–73所示。

注意：

当超界或奇偶校验错时，接收信息功能自动终止。必须为接收信息功能操作定义一个启动条件 (x 或 z) 和一个结束条件 (y，t 或最大字符数)。

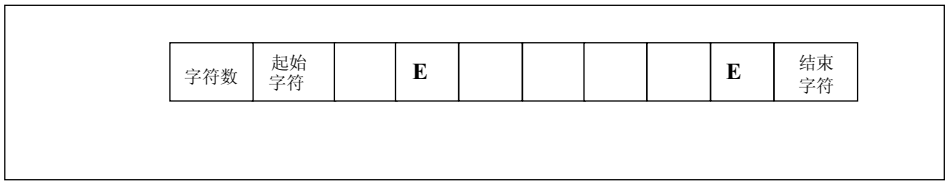


图 9-73 RCV 缓冲区格式

表 9-26 特殊存储器字节SMB86到SMB94，SMB186到SMB194

端口 0	端口 1	描 述
SMB86	SMB186	MSB LSB 接收信息状态字节 n r e 0 0 t c p 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

表 9-26　特殊存储器字节SMB86到SMB94，SMB186到SMB194 (续)

端口 0	端口 1	描　述
SMB87	SMB187	MSB 7　LSB 0 en | sc | ec | il | c/m | tmr | bk | 0　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 SMB90 或 SMB190 1=使用 SMB90 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 is a don't care 3.中断检测:　il=0,sc=1,bk=1,SMW90 is a don't care 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 is a don't care 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
端口 0	**端口 1**	**描　述**
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。SM90 (或SM190) 是最高有效字节，SM91 (或 SM191) 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。SM92 (或SM192) 是最高有效字节， SM93 (或SM193) 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，即使信息的字符数终止用不到。

使用字符中断控制接收数据

由协议支持所允许的完全柔性，可以使用字符中断控制来接收数据。每接收一个字符会产生一个中断。在执行连接到接收字符中断事件上的中断程序前，接收到的字符存储在 SMB2 中，奇偶状态 (如果允许) 存储在 SM3.0 中。

- SMB2 是自由端口接收数据的缓冲区。自由端口模式下接收到的每个字符被存储在这个单元，以方便用户程序访问。.
- SMB3 用于自由端口模式，并包含一个在检测到接收字符奇偶校验错时接通的奇偶校验错误位。该字节的所有其他位被保留。用该位丢弃本信息或产生对本信息的否定确认。

注意：

端口 0 和端口 1 共用 SMB2 和 SMB3，当在端口 0 接收字符时，端口 0 的接收启动和该事件 (中断事件 8) 相连接的中断程序，SMB2 中包含端口 0 接收的字符，SMB3 中包含字符的校验状态。当在端口 1 接收字符时，端口 1的接收启动和该事件 (中断事件 25) 相连接的中断程序。SMB2 中包含端口1接收的字符，SMB3 中包含字符的校验状态。

接收和发送举例

本程序展示了接收和发送的使用，它将接收一串字符，直到接收到回车符。信息又发回到发送方。

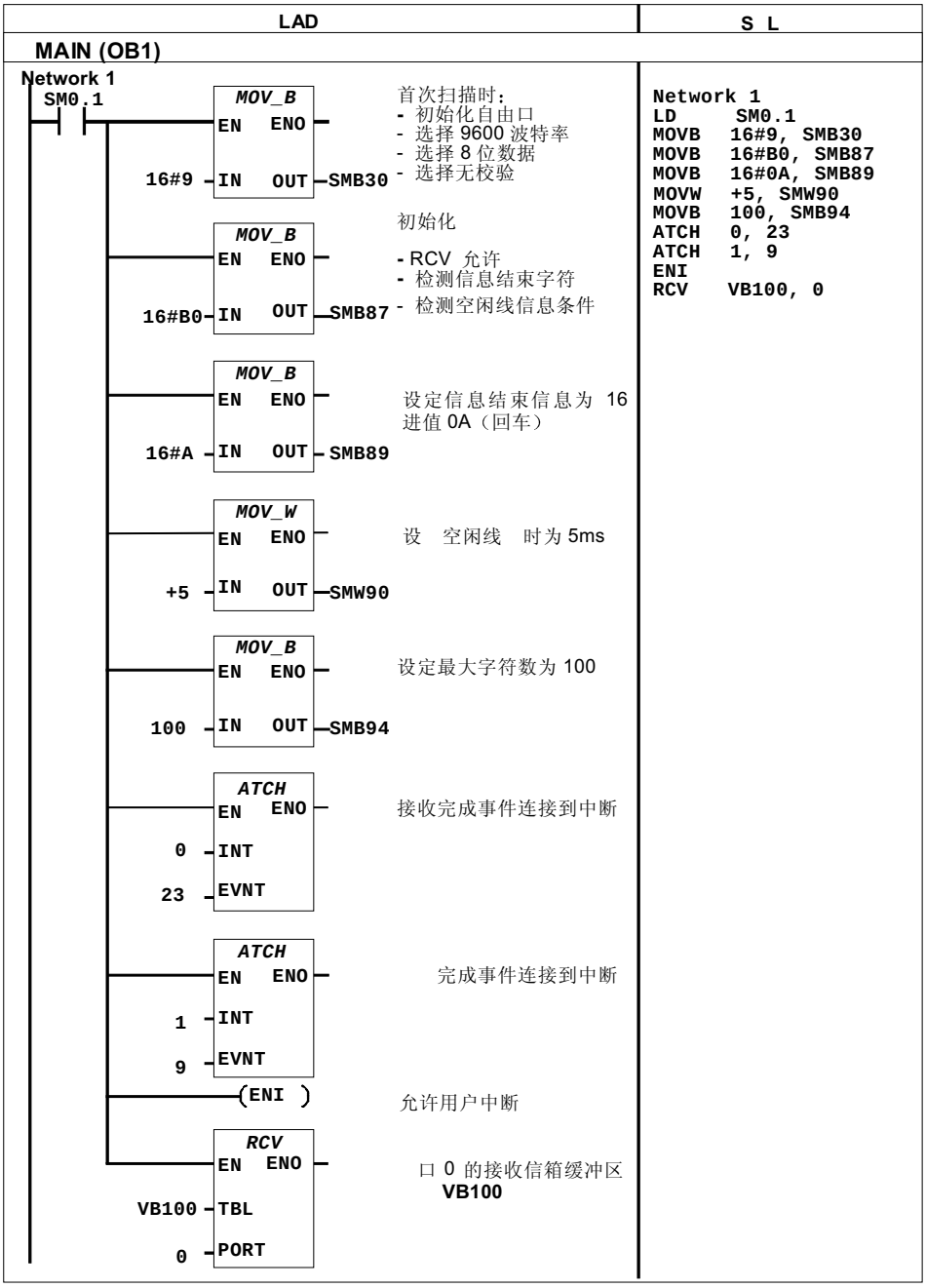


图 9-74　　传送指令举例

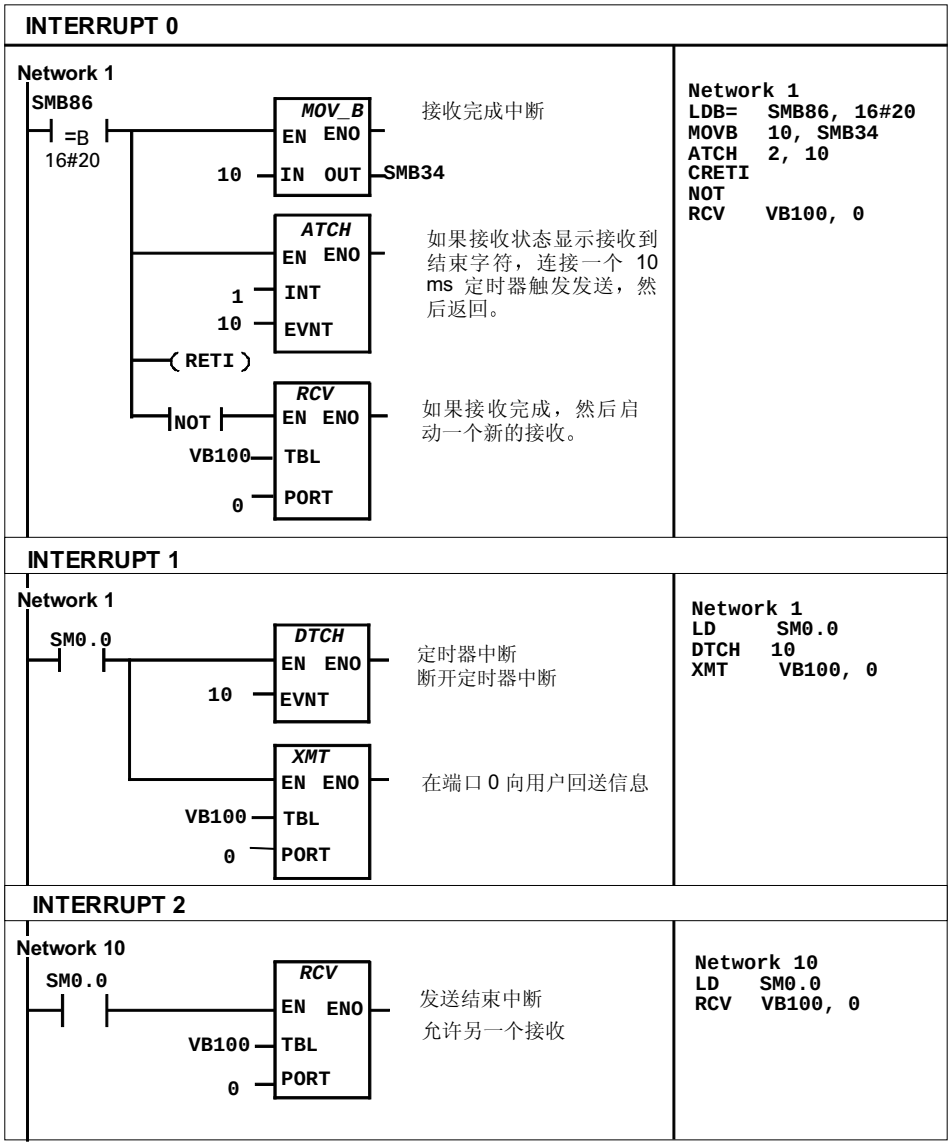


图 9-74　　发送指令举例 (续)

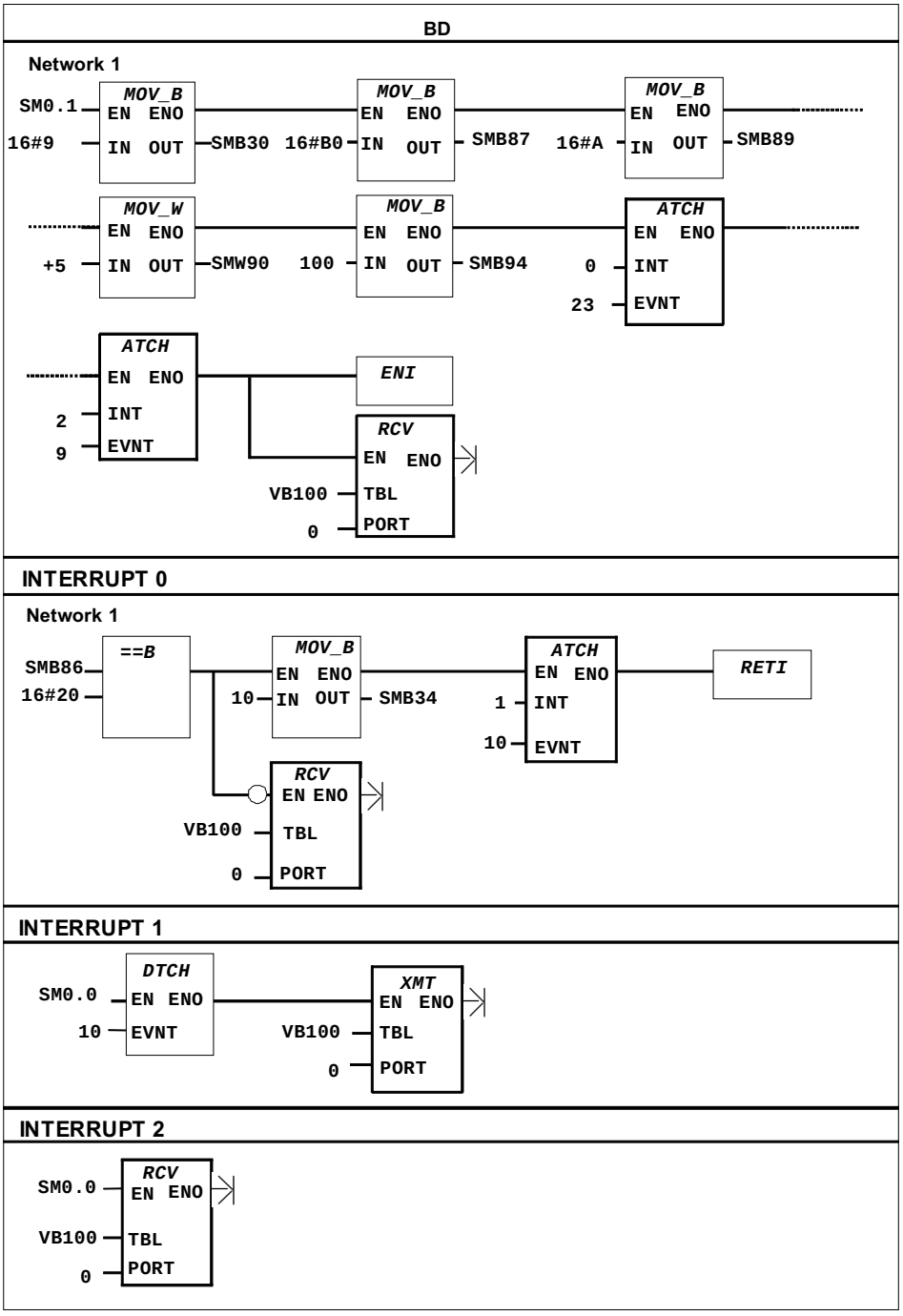
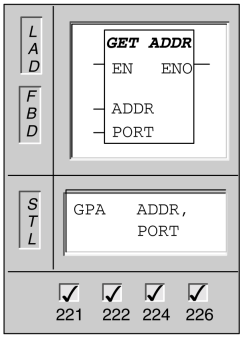


图 9-74　　发送指令举例 (续)

获取口地址

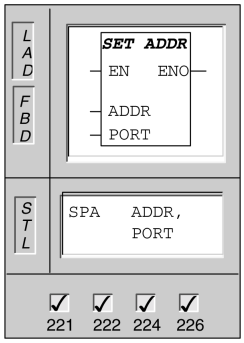


获取口地址 (GPA) 指令读取 PORT 指定的 CPU 口的站地址，将数值放入 ADDR 指定的地址中。

GPA: 使 ENO = 0 的错误条件: SM4.3 (运行时间), 0006 (间接寻址)

输入/输出	操 作 数	数据类型
ADDR	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	BYTE
PORT	常数	BYTE

设定口地址



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

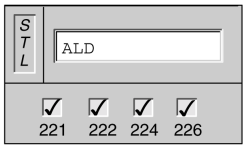
SPA: 使 ENO = 0 的错误条件: SM4.3 (运行时间), 0006 (间接寻址)

新地址不能永久保存。上电后，口地址将返回到上次的地址值 (用系统块下载的地址)。

输入/输出	操 作 数	数据类型
ADDR	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD	BYTE
PORT	常数	BYTE

9.16 SIMATIC 逻辑堆栈指令

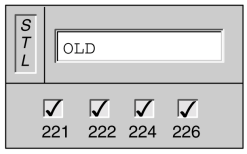
栈装载与 (ALD)



ALD 指令对堆栈中的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 ALD 指令后堆栈深度减 1

操作数：无

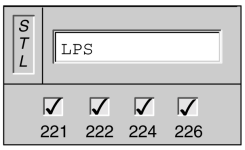
栈装载或 (OLD)



OLD 指令对堆栈中的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 OLD 指令后，堆栈深度减 1。

操作数：无

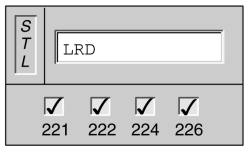
逻辑推入栈



LPS 指令复制栈顶的值并将这个值推入栈。栈底的值被推出并丢失。

操作数：无

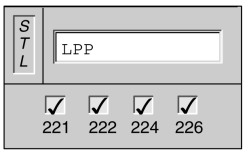
逻辑读栈



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

操作数：无

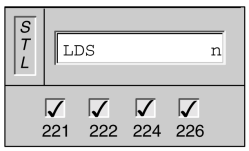
逻辑弹出栈



LPP指令弹出栈顶的值，堆栈的第二个值成为新的栈顶值。

操作数：无

装入堆栈



装入堆栈指令复制堆栈中的第 n 个值到栈顶。栈底丢失。

操作数：n (1 到 8)

逻辑堆栈操作

图 9–75 揭示了 ALD 指令和 OLD 指令的操作过程。

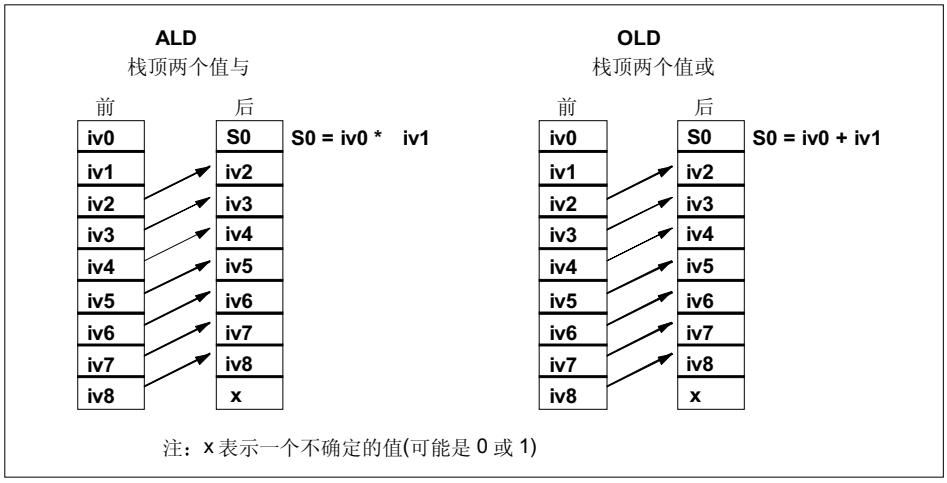


图 9-75 栈装载与和栈装载或指令

图 9–76 揭示了 LPS，LRD 及 LPP 指令的操作过程。

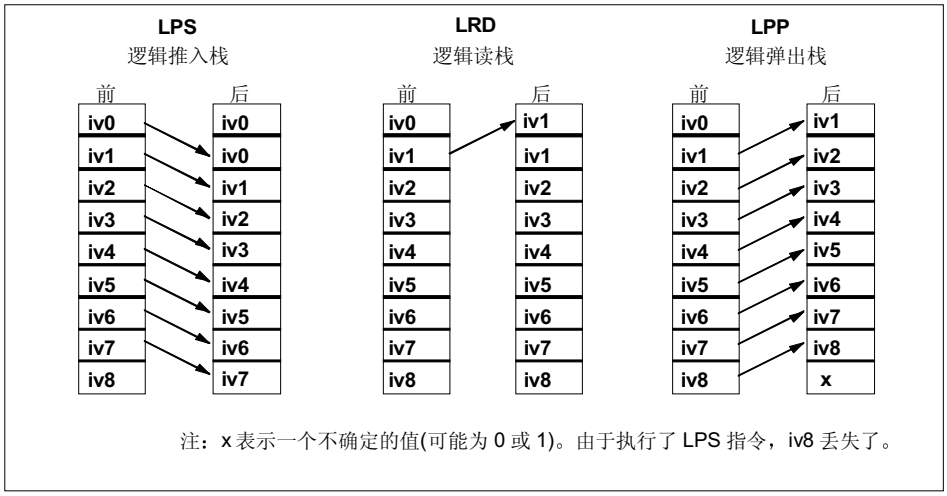


图 9-76 LPS，LRD 和 LPP 指令的操作过程

图 9–77 揭示了装入堆栈指令的操作过程。

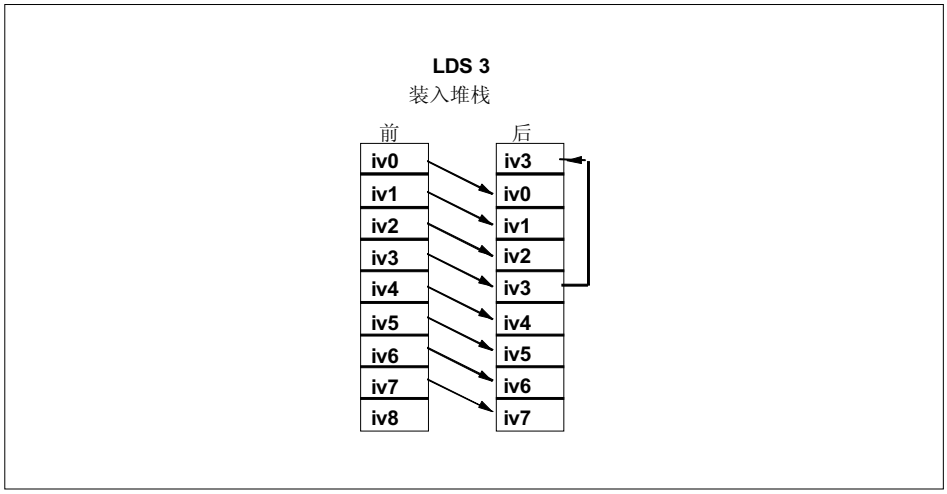


图 9–77 装入堆栈指令的操作过程。

逻辑堆栈举例

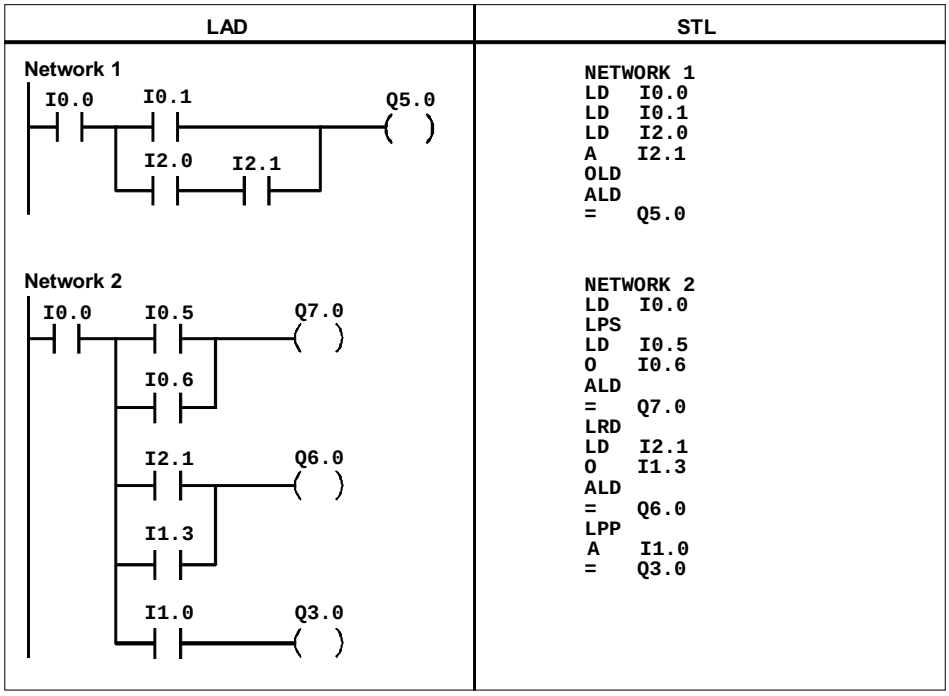


图 9-78 逻辑堆栈指令的 LAD 和 STL 举例

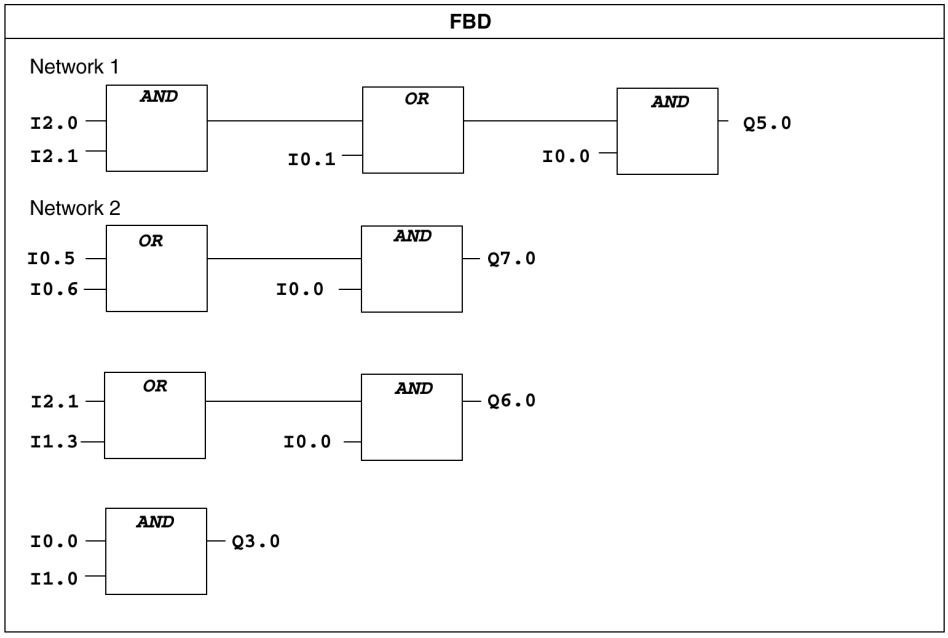


图 9-79　　逻辑堆栈指令的 FBD 举例

IEC 1131-3 指令

10

本章讲解标准 IEC 1131-3 指令。一些 SIMATIC 指令可以用于 IEC 程序，这些指令称为非标准 IEC 指令，在每节的开始讲解这些指令。

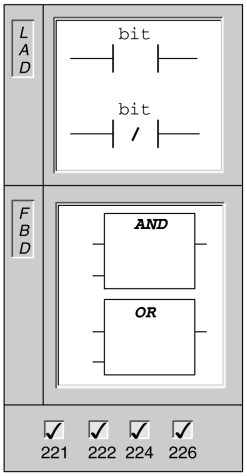
本章概述

10.1 IEC 位逻辑指令

表 10-1 给出非标准 IEC 位逻辑指令的页参考。

表 10-1 非标准 IEC 位逻辑指令

标准触点 (非标准 IEC 1131-3)



常开 (NO) 触点：当位值是 1 时，该触点闭合。

常闭 (NC) 触点：当位值是 0 时，该触点闭合。

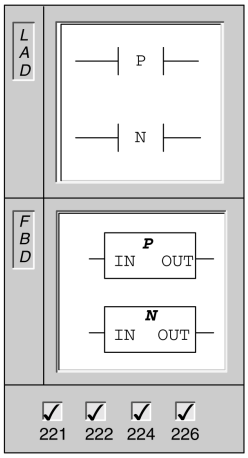
如果存储器类型是 I 或 Q，这些指令从存储器或映像寄存器取值。

在梯形图 (LAD) 中，常开和常闭指令用触点表示。

在功能块图 (FBD) 中，常开指令用 AND/OR 指令盒表示。这些指令象梯形图中的触点一样用来处理布尔信号。常闭指令也可以用指令盒表示，只不过要在输入信号上用一个圆圈表示取反。与和或指令盒的输入可最多扩展到 7 个输入。

输入/输出	操 作 数	数据类型
位 (LAD，STL)	I, Q, M, SM, T, C, V, S, L	BOOL
输入 (LAD)	能量流	BOOL
输入 (FBD)	I, Q, M, SM, T, C, V, S, L, 能量流	BOOL
输出 (LAD, FBD)	I, Q, M, SM, T, C, V, S, L, 能量流	BOOL

上升沿、下降沿转换



上升沿触点对于从断开到接通的信号变化产生一个扫描周期的能量流信号。

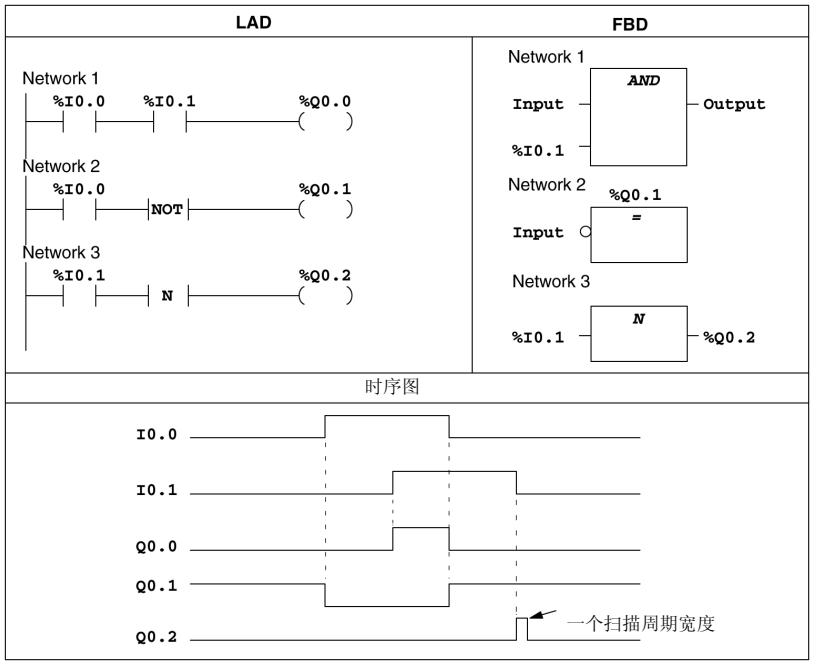
下降沿触点对于从接通到断开的信号变化产生一个扫描周期的能量流信号。

在 LAD 中，上升沿和下降沿转换指令用触点表示。

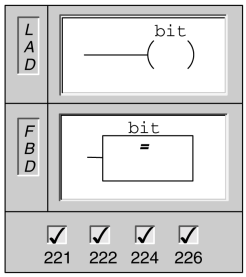
在 FBD 中，上升沿和下降沿转换指令用 P 和 N 指令盒表示。

输入/输出	操 作 数	数据类型
IN (LAD)	能量流	BOOL
IN (FBD)	I, Q, M, SM, T, C, V, S, L, 能量流	BOOL
OUT (LAD)	能量流	BOOL
OUT (FBD)	I, Q, M, SM, T, C, V, S, L, 能量流	BOOL

触点举例



输出



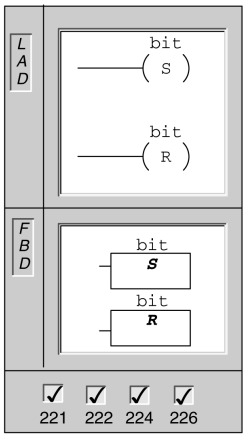
当执行输出时，输出点接通。

在 LAD 中，输出指令用线圈表示。

在 FBD 中，输出指令用“ = ”指令盒表示。

输入/输出	操 作 数	数据类型
位 (LAD/FBD)	I, Q, M, SM, T, C, V, S, L	BOOL
输入 (LAD)	能量流	BOOL
输入 (FBD)	I, Q, M, SM, T, C, V, S, L, 能量流	BOOL

置位和复位



当执行**置位和复位**指令时，位或OUT 指定的值被置位或复位。

输入/输出	操 作 数	数据类型
位 (LAD, FBD)	I, Q, M, SM, T, C, V, S, L	BOOL
输入 (LAD)	能量流	BOOL
输入 (FBD)	I, Q, M, SM, T, C, V, S, L, 能量流	BOOL

输出举例

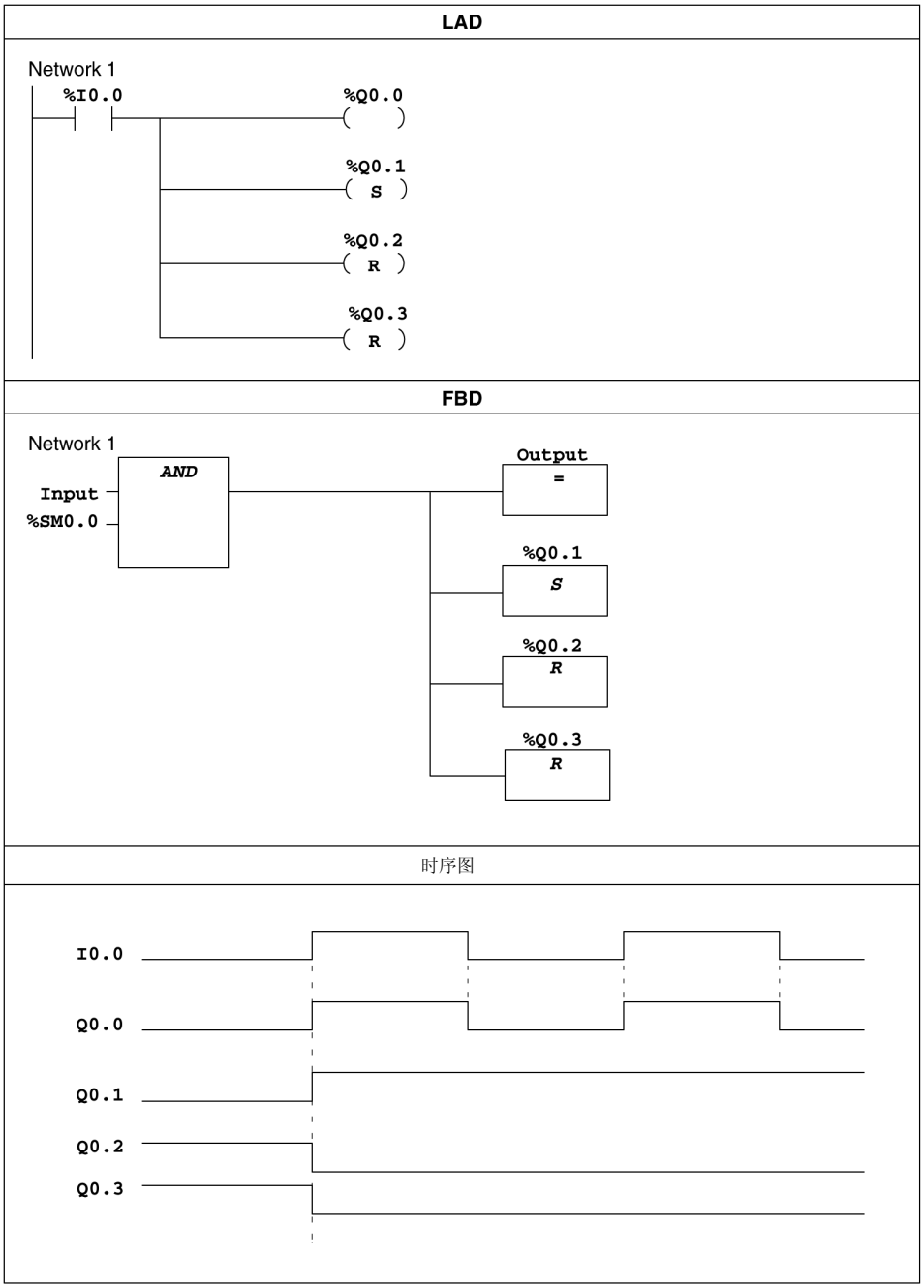
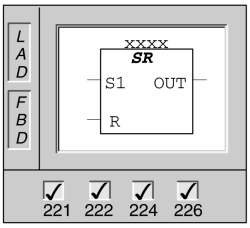


图 10-2　　输出指令的 LAD 和 FBD 举例

置位优先双稳态触发器

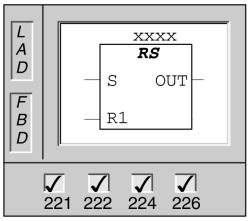


置位优先双稳态触发器是一个置位优先的锁存器。如果置位 (S1) 和复位 (R) 信号都为真，输出 (OUT) 就为真。

功能块参数 xxxx 指定要置位和复位的布尔参数，可选的 输出反映了 xxxx 的信号状态。

输入/输出	操 作 数	数据类型
S1, R (LAD)	能量流	BOOL
S1, R (FBD)	I, Q, M, SM, T, C,V, S, 能量流	BOOL
OUT (LAD)	能量流	BOOL
OUT (FBD)	I, Q, M, SM,T, C, V, S, L, 能量流	BOOL
xxxx	I, Q, M, V, S	BOOL

复位优先双稳态触发器



复位优先双稳态触发器是一个复位优先的锁存器。如果置位 (S) 和复位 (R1) 信号都为真，输出 (OUT) 将为假。

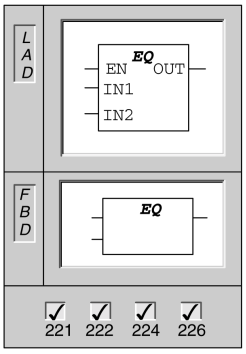
功能块参数 xxxx 指定要置位和复位的布尔参数，可选的输出反映了 xxxx 的信号状态

输入/输出	操 作 数	数据类型
S, R1 (LAD)	能量流	BOOL
S, R1 (FBD)	I, Q, M, SM, T, C, V, S, L, 能量流	BOOL
OUT (LAD)	Power Flow	BOOL
OUT (FBD)	I, Q, M, SM, T, C,V, S, L, 能量流	BOOL
xxxx	I, Q, M, V, S	BOOL

10.2 IEC 比较指令

没有非标准 IEC比较指令。

相等比较

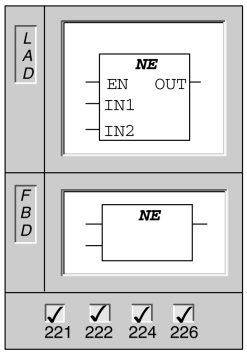


相等比较把 IN1 和 IN2 端的变量进行比较，在OUT 端输出一个布尔量。输入和输出的数据类型可以变化，但必须保持一致。

字节比较是无符号的，整数、双整数和实数比较是有符号的，时间比较是有符号的整数。

输入/输出	操 作 数	数据类型
输入 (LAD 和 FBD)	IB, QB, MB, SB, SMB, VB, LB, IW, QW, MW, SW, SMW, VW, LW, T, C, AIW, ID, QD, MD, SD, SMD, VD, LD, HC, AC, 常数, *VD, *AC, *LD	BYTE, INT, DINT, REAL
OUT (仅LAD)	能量流	BOOL
OUT (仅FBD)	I, Q, M, SM, T, C, V, S, L, 能量流	BOOL

不等比较

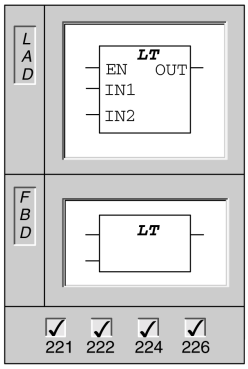


不等比较把 IN1 和 IN2 端的变量进行比较，在 OUT 端输出一个布尔量。输入和输出的数据类型可以变化，但必须保持一致。

字节比较是无符号的，整数、双整数和实数比较是有符号的，时间比较是有符号的整数。

输入/输出	操 作 数	数据类型
输入 (LAD 和 FBD)	IB, QB, MB, SB, SMB, VB, LB, IW, QW, MW, SW, SMW, VW, LW, T, C, AIW, ID, QD, MD, SD, SMD, VD, LD, HC, AC, 常数, *VD, *AC, *LD	BYTE, INT, DINT, REAL
OUT (仅 LAD)	能量流	BOOL
OUT (仅 FBD)	I, Q, M, SM, T, C, V, S, L, 能量流	BOOL

小于比较

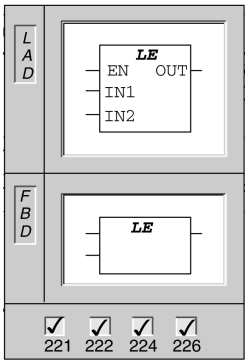


小于比较把 IN1 和 IN2 端的变量进行比较，在 OUT 端输出一个布尔量。输入和输出的数据类型可以变化，但必须保持一致。

字节比较是无符号的，整数、双整数和实数比较是有符号的，时间比较是有符号的整数。

输入/输出	操 作 数	数据类型
输入 (LAD 和 FBD)	IB, QB, MB, SB, SMB, VB, LB, IW, QW, MW, SW, SMW, VW, LW, T, C, AIW, ID, QD, MD, SD, SMD, VD, LD, HC, AC, 常数, *VD, *AC, *LD	BYTE, INT, DINT, REAL
OUT (仅LAD)	能量流	BOOL
OUT (仅FBD)	I, Q, M, SM, V, S, L, 能量流	BOOL

小于等于比较

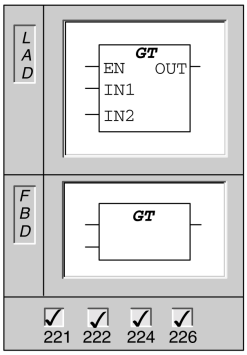


小于等于比较把 IN1 和 IN2 端的变量进行比较，在OUT 端输出一个布尔量。输入和输出的数据类型可以变化，但必须保持一致。

字节比较是无符号的，整数、双整数和实数比较是有符号的，时间比较是有符号的整数。

输入/输出	操 作 数	数据类型
输入 (LAD 和 FBD)	IB, QB, MB, SB, SMB, VB, LB, IW, QW, MW, SW, SMW, VW, LW, T, C, AIW, ID, QD, MD, SD, SMD, VD, LD, HC, AC, 常数, *VD, *AC, *LD	BYTE, INT, DINT, REAL
OUT (仅LAD)	能量流	BOOL
OUT (仅FBD)	I, Q, M, SM, V, S, L, 能量流	BOOL

大于比较

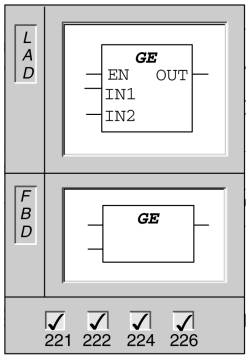


大于比较把 IN1 和 IN2 端的变量进行比较，在OUT 端输出一个布尔量。输入和输出的数据类型可以变化，但必须保持一致。

字节比较是无符号的，整数、双整数和实数比较是有符号的，时间比较是有符号的整数。

输入/输出	操 作 数	数据类型
输入 (LAD 和 FBD)	IB, QB, MB, SB, SMB, VB, LB, IW, QW, MW, SW, SMW, VW, LW, T, C, AIW, ID, QD, MD, SD, SMD, VD, LD, HC, AC, 常数, *VD, *AC, *LD	BYTE, INT, DINT, REAL
OUT (仅LAD)	能量流	BOOL
OUT (仅FBD)	I, Q, M, SM, V, S, L, 能量流	BOOL

大于等于比较



大于等于比较把 IN1 和 IN2 端的变量进行比较，在 OUT 端输出一个布尔量。输入和输出的数据类型可以变化，但必须保持一致。

字节比较是无符号的，整数、双整数和实数比较是有符号的。

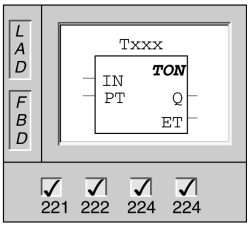
输入/输出	操 作 数	数据类型
输入 (LAD 和 FBD)	IB, QB, MB, SB, SMB, VB, LB, IW, QW, MW, SW, SMW, VW, LW, T, C, AIW, ID, QD, MD, SD, SMD, VD, LD, HC, AC, 常数, *VD, *AC, *LD	BYTE, INT, DINT, REAL
OUT (仅LAD)	能量流	BOOL
OUT (仅FBD)	I, Q, M, SM, V, S, L, 能量流	BOOL

10.3 IEC 定时器指令

表 10–2 是非标准IEC定时器指令的页参考。

表 10-2 非标准IEC定时器指令

接通延时定时器

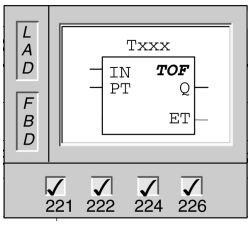


当使能输入端 (IN) 为 1 时，接通延时定时器功能块开始启动定时，一直到预设值。当经过时间 (ET) 大于等于预设 (PT) 时，定时器输出位 (Q) 变为 1。

当使能输入端 (IN) 为 0 时，定时器输出复位。当预设时间 (PT) 到达时，定时停止并且定时器不工作。

输入/输出	操　作　数	数据类型
IN (LAD)	能量流	BOOL
IN (FBD)	I, Q, M, SM, T, C, V, S, L, 能量流	BOOL
PT (LAD 和 FBD)	VW, IW, QW, MW, SMW, LW, SW, AIW, AC, 常数, *VD, *AC, *LD	INT
Q (LAD 和 FBD)	I, Q, M, SM, V, S, L	BOOL
ET (LAD 和 FBD)	VW, IW, QW, MW, SMW, LW, SW, AQW, AC, *VD, *AC, *LD	INT
Txxx	参考表 10–3	TON

断开延时定时器



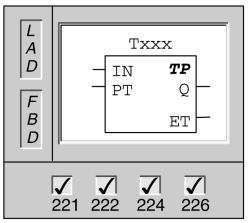
当输入断开时，**断开延时定时器**功能块把输出断开的时间延迟一个固定的时间。当使能输入 (IN) 变为 0 时，定时器的值又变为预设值。 当经过时间 (ET) 大于等于预设时间 (PT) 时，定时器输出位 (Q) 接通。

一旦达到预设值，定时器的输出位变为 0，经过时间一直保持到使能输入 (IN) 再变为1。如果使能输入 (IN) 变为0的持续时间小于预设值(PT)，定时器的输出位一直保持1。

有关定时器号和分辨率的信息，请参考表 10–3。

输入/输出	操　作　数	数据类型
IN (LAD)	能量流	BOOL
IN (FBD)	I, Q, M, SM,T, C, V, S, L, 能量流	BOOL
PT (LAD 和 FBD)	VW, IW, QW, MW, SMW, LW, SW, AIW, AC, 常数, *VD, *AC, *LD	INT
Q (LAD 和 FBD)	I, Q, M, SM, V, S, L	BOOL
ET (LAD 和 FBD)	VW, IW, QW, MW, SMW, LW, SW, AQW, AC,*VD, *AC, *LD	INT
Txxx	参考表10–3	TOF

脉冲定时器



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，在预设时间内输出位保持接通。一旦经过时间 (ET) 达到预设时间 (PT)，输出位变为 0。
有关定时器号和分辨率的信息，请参考表 10–3。

输入/输出	操 作 数	数据类型
IN (LAD)	能量流	BOOL
IN (FBD)	I, Q, M, SM, T, C, V, S, L, 能量流	BOOL
PT (LAD 和 FBD)	VW, IW, QW, MW, SMW, LW, SW, AIW, AC, 常数, *VD, *AC, *LD	INT
Q (LAD 和 FBD)	I, Q, M, SM, S, V, L	BOOL
ET (LAD 和 FBD)	VW, IW, QW, MW, SW, LW, AQW, AC, *VD, *AC, *LD	INT
Txxx	参考表 10–3	TP

理解 IEC 1131-3 定时器指令

TON、TOF和TP 定时器有三种分辨率。分辨率由表 10-3 中的定时器号决定。每个当前值的数有多个时基，例如，以 10 ms 为时基的数 50 代表 500 ms。

表 10-3 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) 表示的最大值	定时器号
TON, TOF, TP	1 ms	32.767 s	T32, T96
	10 ms	327.67 s	T33 ~ T36, T97 ~ T100
	100 ms	3276.7 s	T37 ~ T63, T101 ~ T255

注意：

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，TOF T32。

接通延时定时器举例

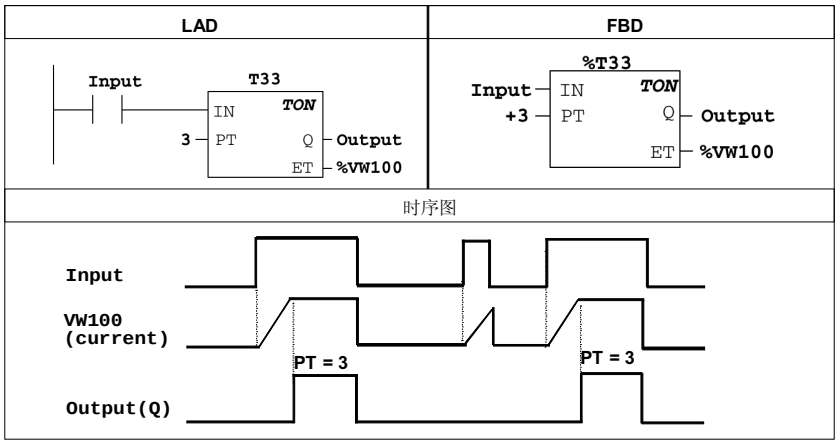


图 10-3 接通延时定时器的 LAD 和 FBD 举例

断开延时定时器举例

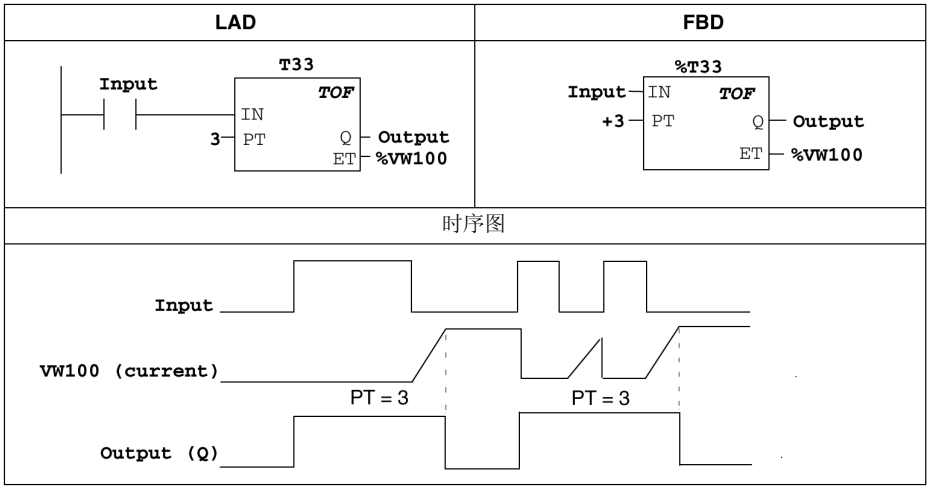


图 10-4　　断开延时定时器 LAD 和 FBD 举例

脉冲定时器举例

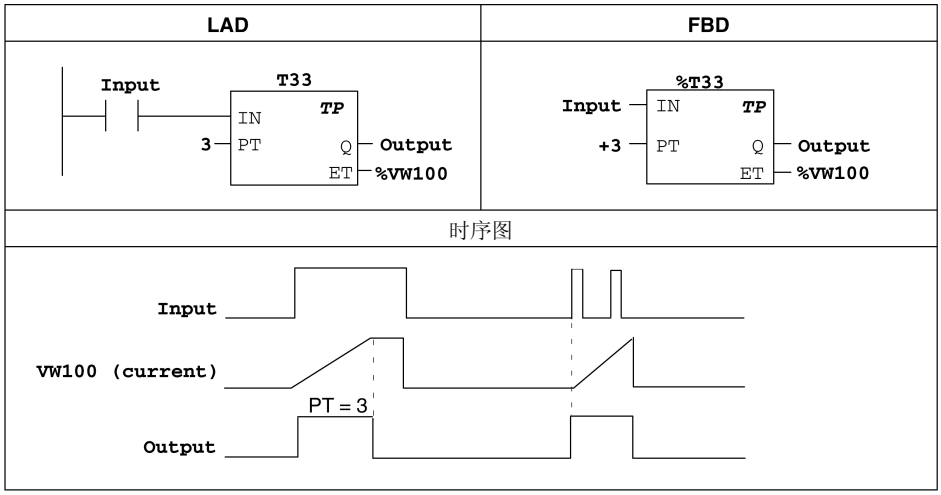


图 10-5　　脉冲定时器的 LAD 和 FBD 举例

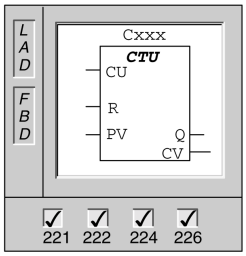
10.4 IEC 计数器指令

表 10-4 是非标准 IEC 计数器指令的页参考。

表 10-4 非标准 IEC 计数器指令

描 述	页
高速计数器指令	9–27
高速计数器定义指令	9–27
脉冲输出指令	9–49

增计数器



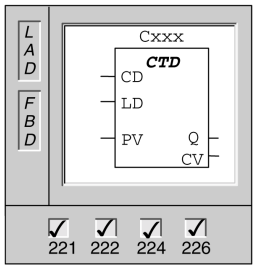
增计数器功能块对增计数 (CU) 输入端的上升沿从当前值到预设值进行计数。当计数器当前值 (CV) 大于等于预设值 (PV) 时，输出 (Q) 接通。当复位输入 (R) 端有使能信号时，计数器复位。当增计数器达到预设值时，计数器停止计数。

注意：

由于每个计数器只有一个当前值，请不要把一个计数器号分配给几个计数器 (增计数器，减计数器和增/减计数器存取同一个当前值)。

输入/输出	操 作 数	数据类型
CU，R (仅LAD)	能量流	BOOL
CU，R (仅FBD)	I, Q, M, SM, V, S, L, T, C, 能量流	BOOL
PV (LAD 和 FBD)	VW, IW, QW, MW, SMW, LW, SW, AIW, AC, 常数, *VD, *AC, *LD	INT
Q (LAD 和 FBD)	I, Q, M, SM, V, S, L	BOOL
CV (LAD 和 FBD)	VW, IW, QW, MW, SW, SMW, LW, AC, *VD, *AC, *LD	INT
Cxxx(LAD和FBD)	常数	CTU

减计数器



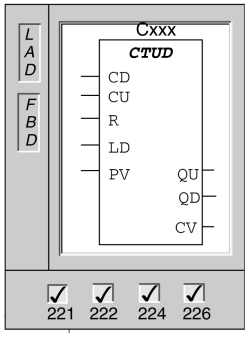
减计数器功能块对减计数 (CD) 输入端的上升沿从预设值进行计数。当计数器当前值 (CV) 等于 0 时，输出 (Q) 接通。当装载输入 (LD) 端有使能信号时，计数器复位并把预设值 (PV) 装入当前值 (CV)。当减计数器达到 0 时，计数器停止计数。

注意：

由于每个计数器只有一个当前值，请不要把一个计数器号分配给几个计数器 (增计数器，减计数器和增/减计数器存取同一个当前值)。

输入/输出	操 作 数	数据类型
CD, LD (LAD)	能量流	BOOL
CD, LD (FBD)	I, Q, M, SM, V, S, L, T, C, 能量流	BOOL
PV (LAD 和 FBD)	VW, IW, QW, MW, SMW, LW, SW, AIW, AC, 常数, *VD, *AC, *LD	INT
Q (LAD 和 FBD)	I, Q, M, SM, V, S, L	BOOL
CV (LAD 和 FBD)	VW, IW, QW, MW, SW, LW, AC, *VD, *AC, *LD	INT
Cxxx	常数	CTD

加/减计数器



加/减计数功能块在预置值的基础上对加计数端 (CU) 或减计数端 (CD) 的上升边沿进行加或减操作。当目前值 (CV) 等于预设值时，输出 (QU) 接通。当目前值 (CV) 等于0时，输出 (QD) 接通。当装载 (LD) 输入端有允许脉冲时，计数器把预置值装入当前值区域。相同地，当复位 (R) 输入端有允许脉冲时，计数器把 0 装入当前值区域。当计数器达到预置值或 0 时，计数器停止计数。

注意：

由于每个计数器只有一个当前值，请不要把一个计数器号分配给几个计数器 (增计数器，减计数器和增/减计数器存取同一个当前值)。

输入/输出	操 作 数	数据类型
CD, CD, R, LD (仅LAD)	能量流	BOOL
CU, CD, R, LD (仅FBD)	I, Q, M, SM, V, S, L, T, C, 能量流	BOOL
PV (LAD 和 FBD)	VW, IW, QW, MW, SMW, LW, SW, AIW, AC, 常数, *VD, *AC, *LD	INT
QU (LAD 和 FBD)	I, Q, M, SM, V, S, L	BOOL
QD (LAD 和 FBD)	I, Q, M, SM, V, S, L	BOOL
CV (LAD 和 FBD)	VW, T, C, IW, QW, MW, SW, LW, AC, *VD, *AC, *LD	INT
Cxxx	常数	CTUD

计数器举例

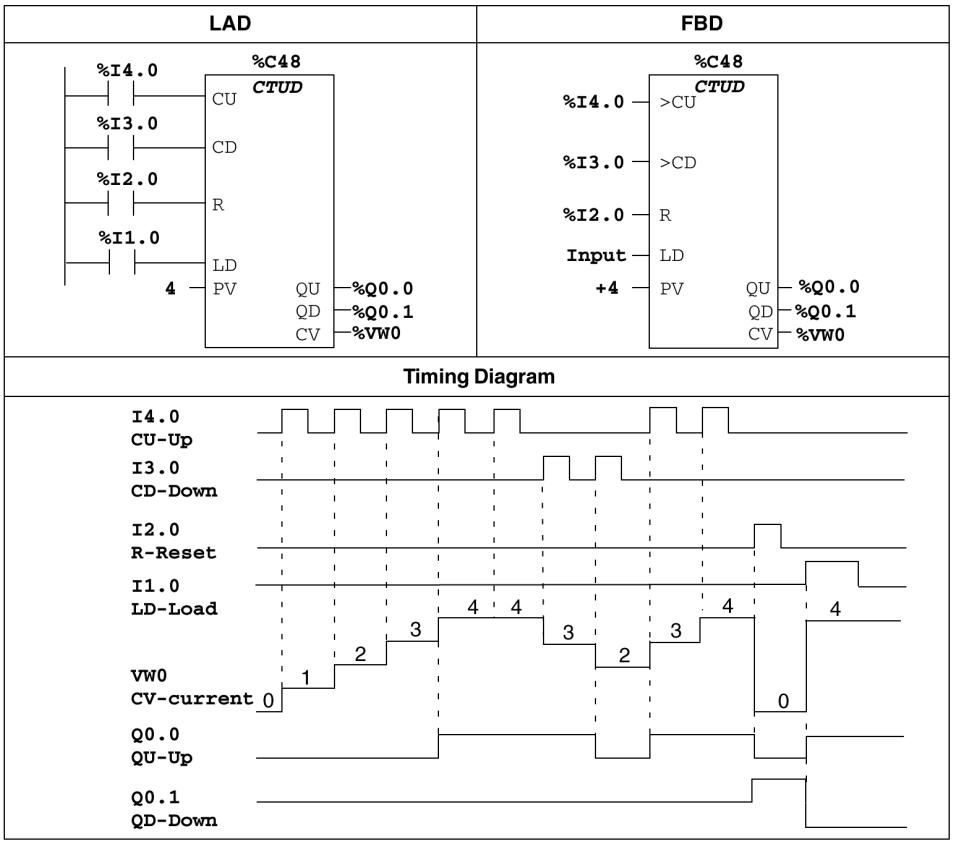
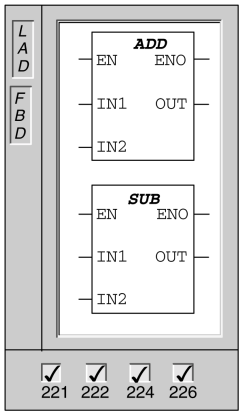


图 10-6　　计数器指令的 LAD 和 FBD 指令举例

10.5 IEC 数学运算指令

加法和减法



加法和减法功能把 IN1 和 IN2 端的值相加或减，并把结果放入 OUT 中。输入和输出数据类型可以变化，但必须相同。例如，两个 16 位变量可以相加或相减，但结果一定要放到一个 16 位变量中；两个 32 位变量的加法或减法结果一定要放到 32 位变量中。

在 LAD 中：IN1 + IN2 = OUT

IN1 - IN2 = OUT

使 ENO=0 的出错条件：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)

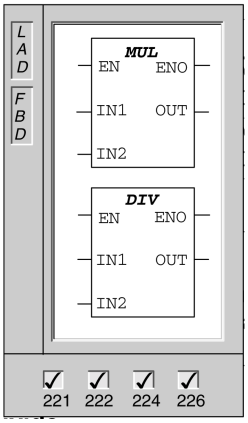
这些功能影响下列特殊存储器位：SM1.0 (零)，SM1.1 (溢出)，SM1.2 (负)

输入/输出	操 作 数	数据类型
IN1, IN2	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, VD, ID, QD, MD, SMD, SD, LD, HC, AC, 常数, *VD, *AC, *LD	INT, DINT, REAL
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *AC, *LD	INT, DINT, REAL

注意：

实数或浮点数用 ANSI/IEEE751-1985 标准 (单精度) 的格式表示。有关详细信息请参考这些标准。

乘法和除法



乘法指令把 IN1 与 IN2 相乘，把结果存到 OUT 指定的变量。

除法指令把 IN1 与 IN2 相除，把结果存到 OUT 指定的变量.

输入和输出数据类型可以变化，但必须相同。例如，两个 16 位变量的乘积一定要放到 16 位变量中，两个 32 位变量的乘积一定要放到 32 位变量中。

在 LAD 中：IN1×IN2 = OUT

IN1 / IN2 = OUT

使 ENO=0 的出错条件：SM1.1 (溢出)，SM1.3 (被0 除)，SM4.3 (运行时间)，0006 (间接寻址)

这些功能影响下列特殊存储器位：SM1.0 (零)，SM1.1 (溢出)，SM1.2 (负)，SM1.3 (被 0 除)。

如果 SM1.1 (溢出位) 置 1，那么清除算术运算状态位， 而且不改动输出操作数。对于整数操作，如果在除法指令执行的过程中 SM1.3 置位，那么其它的数学运算状态位不改变，原始输入操作数也不改变。否则，当运算操作完成时，所有有关的数学运算状态位包含有效状态 。

输入/输出	操 作 数	数据类型
IN1, IN2	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, VD, ID, QD, MD, SMD, SD, LD, HC, AC, 常数, *VD, *AC, *LD	INT, DINT, REAL
OUT	VW, IW, QW, MW, SW, SMW, T, C, LW, VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *AC, *LD	INT, DINT, REAL

注意：

实数或浮点数用 ANSI/IEEE751-1985 标准(单精度)的格式表示。有关详细信息请参考这些标准。

数学运算举例

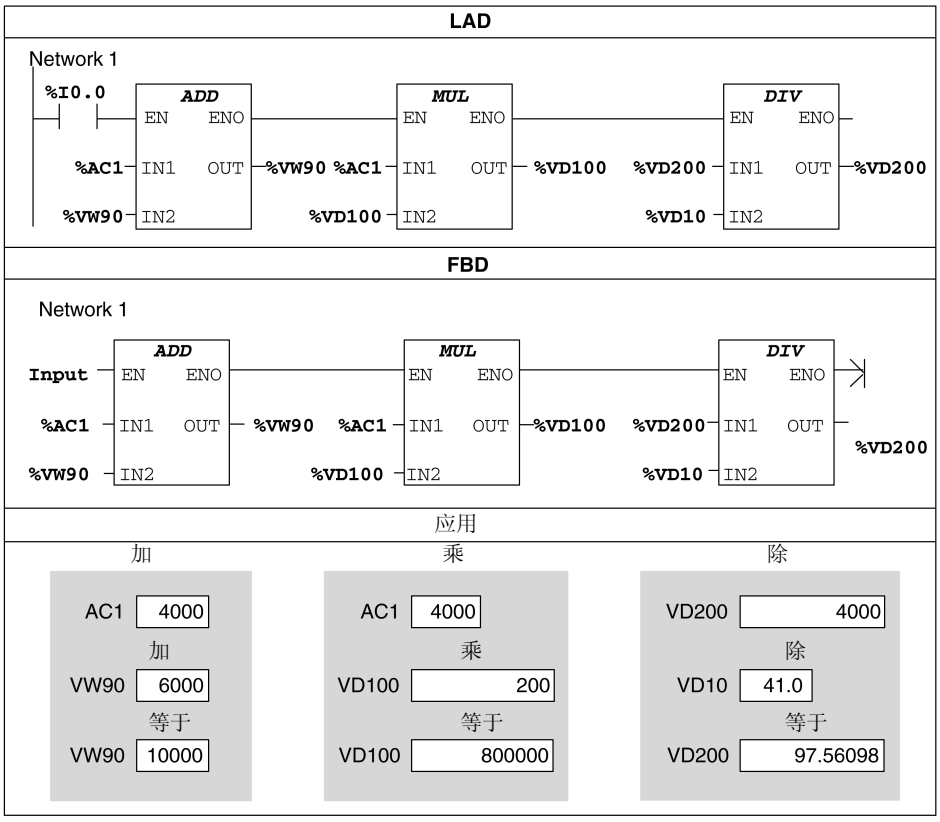
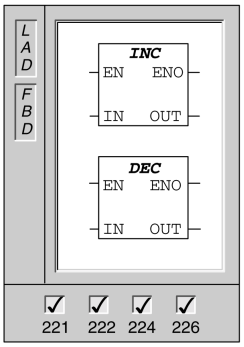


图 10-7　　数学运算功能的 LAD和FBD 举例

增量，减量



增量和**减量**功能把IN端的值加 1 或减 1，把结果送入 OUT。

字节的增量和减量操作是无符号。

使 ENO=0 的出错条件：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)。

这些功能影响下列的特殊存储器位：SM1.0 (零位)， SM1.1 (溢出)，SM1.2 (负数)

输入/输出	操 作 数	数据类型
IN	VB, IB, QB, MB, SB, SMB, LB, VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, VD, ID, QD, MD, SD, SMD, LD, HC, AC, 常数, *VD, *AC, *LD	BYTE, INT, DINT
OUT	VB, IB, QB, MB, SB, SMB, LB, VW, IW, QW, MW, SW, SMW, T, C, LW, VD, ID, QD, MD, SMD, SD, LD, AC,*VD, *AC, *LD	BYTE, INT, DINT

增量，减量指令举例

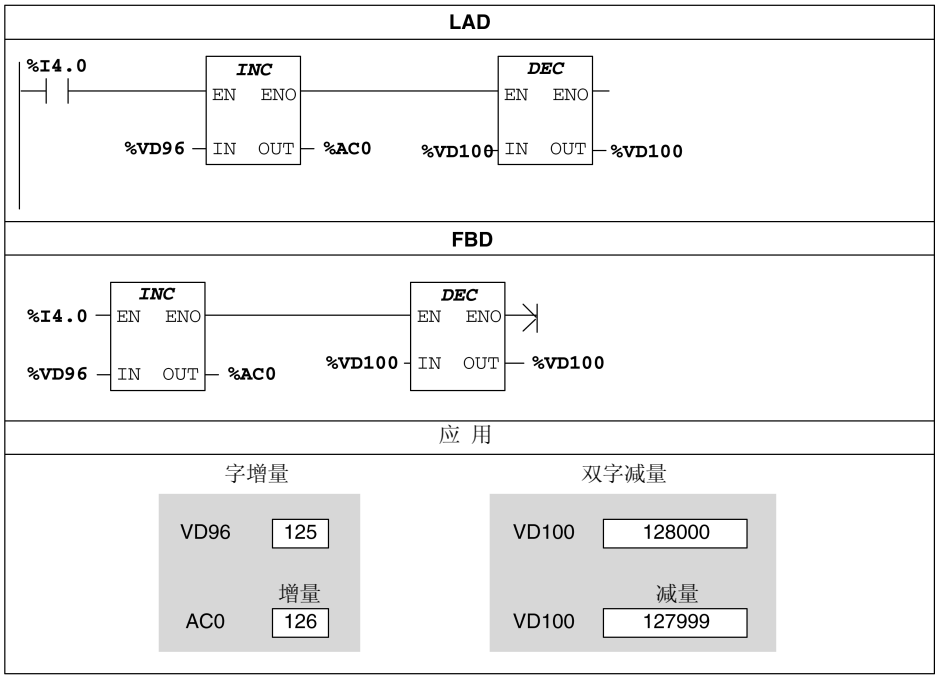


图 10-8　　增量/减量功能的 LAD 和 FBD 举例

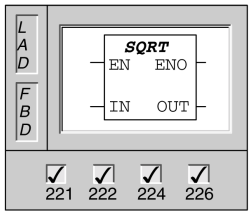
10.6 IEC 数学功能指令

表 10-5 给出非标准数学功能指令的参考页。

表 10-5 非标准数学参考指令

描 述	页
PID 指令	9–85

平方根



平方根功能把 IN 端的值取平方根，并把结果送入 OUT。

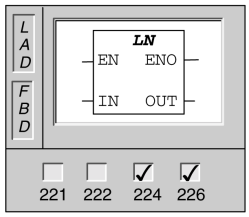
使 ENO=0 的出错条件：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)。

这个功能影响下列的特殊存储器位：SM1.0 (零位)， SM1.1 (溢出)，SM1.2 (负数)

如果 SM1.1 (溢出位) 置 1，那么清除算术运算状态位，而且不改动输出操作数。

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SMD, SD, LD, AC, 常数, *VD, *AC, *LD	REAL
OUT	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *AC, *LD	REAL

自然对数



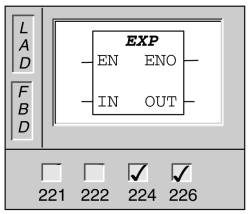
自然对数指令将输入 IN 的值取自然对数，结果放入输出 OUT。

使 ENO=0 的出错条件：SM1.1 (溢出)，0006 (间接寻址)。

这个功能影响下列的特殊存储器位：SM1.0 (零位)， SM1.1 (溢出)，SM1.2 (负数)

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SMD, SD, LD, AC, 常数, *VD, *AC, *LD	REAL
OUT	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *AC, *LD	REAL

指数



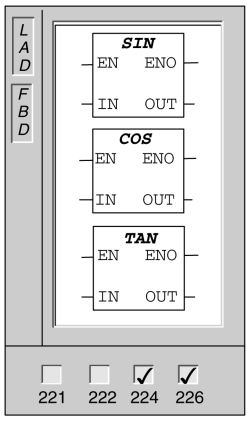
指数指令将输入 IN 的值取指数，结果放入输出 OUT。

使 ENO=0 的出错条件：SM1.1 (溢出)，0006 (间接寻址)。

这个功能影响下列的特殊存储器位：SM1.0 (零位)， SM1.1 (溢出)，SM1.2 (负数)

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SMD, SD, LD, AC, 常数, *VD, *AC, *LD	REAL
OUT	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *AC, *LD	REAL

正弦、余弦、正切



正弦指令将输入 IN 的弧度值取正弦，结果放入输出 OUT。输入值为弧度值。如果输入为角度值，则将该角度值乘以 π /180 以将其转换为弧度值。

余弦指令将输入 IN 的弧度值取余弦，结果放入输出 OUT。输入值为弧度值。如果输入为角度值，则将该角度值乘以 π /180 以将其转换为弧度值。

正切指令将输入 IN 的弧度值取正切，结果放入输出 OUT。输入值为弧度值。如果输入为角度值，则将该角度值乘以 π /180 以将其转换为弧度值。

使 ENO=0 的出错条件：SM1.1 (溢出)，0006 (间接寻址)。

这个功能影响下列的特殊存储器位：SM1.0 (零位)， SM1.1 (溢出)，SM1.2 (负数) 。

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SMD, SD, LD, AC, 常数, *VD, *AC, *LD	REAL
OUT	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *AC, *LD	REAL

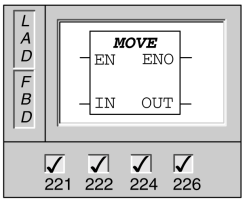
10.7 IEC 传送指令

表 10-6 是非标准 IEC 传送指令的页参考。

表 10-6 非标准 IEC 传送指令

描　　述	页
交换指令	9–76
传送字节立即读	9-77
传送字节立即写	9-78

传送



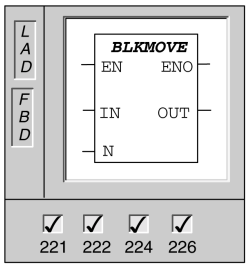
传送和分配值功能把 IN 端的值送到地址 OUT。该指令执行一个分配操作，在执行时不许改变。

输入和输出数据类型可以变化，但必须相同。

使 ENO=0 的出错条件：SM4.3 (运行时间)，0006 (间接寻址)

输入/输出	操　作　数	数据类型
IN	VB, IB, QB, MB, SB, SMB, LB, VW, IW, QW, MW, SM, SMW, LW, T, C, AIW, VD, ID, QD, MD, SMD, SD, LD, HC, &VB, &IB, &QB, &MB, &SB, AC, 常数, *VD, *AC, *LD	BYTE, WORD, INT, DWORD, DINT, REAL
OUT	VB, IB, QB, MB, SB, SMB, LB, VW, IW, QW, MW, SW, SMW, LW, T, C, AQW, VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *AC, *LD	BYTE, WORD, INT, DWORD, DINT, REAL

块传送



块传送功能可以把从 IN 开始的 N 个字传送到OUT。N 的范围是：1 到 255。

输入和输出数据类型可以变化，但必须相同。

块传送是非标准的 只有 IEC 具有的功能。

使 ENO=0 的出错条件：SM4.3 (运行时间)，0006 (间接寻址)，0091 (操作数超界)

输入/输出	操　作　数	数据类型
IN	VB, IB, QB, MB, SB, SMB, LB, VW, IW, QW, MW, SM, SMW, LW, T, C, AIW, VD, ID, QD, MD, SMD, SD, LD, HC, &VB, &IB, &QB, &MB, &SB, AC, 常数, *VD, *AC, *LD	BYTE, WORD, DWORD
OUT	VB, IB, QB, MB, SB, SMB, LB, VW, IW, QW, MW, SW, SMW, LW, T, C, AQW, VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *AC, *LD	BYTE, WORD, DWORD
N	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD	BYTE

传送举例

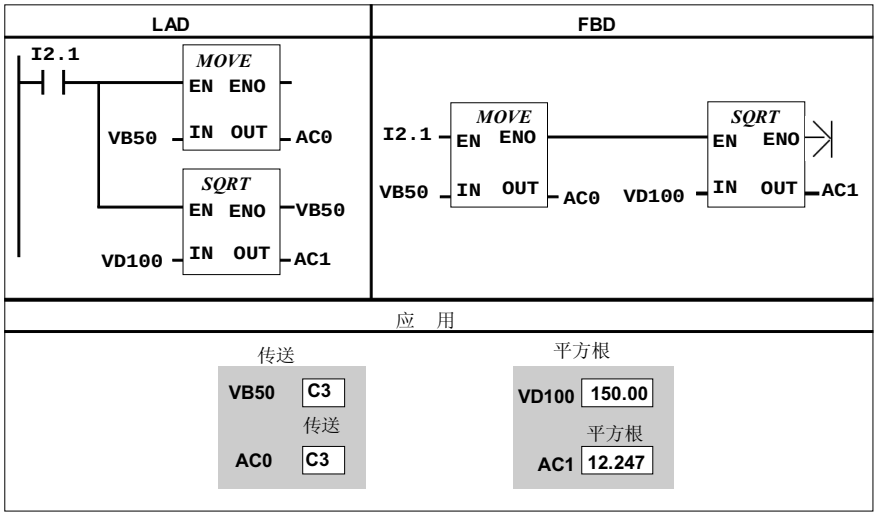
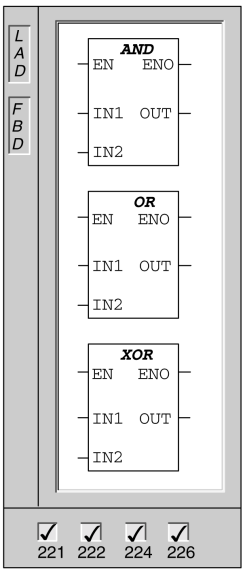


图 10-9　　传送功能的 LAD 和 FBD 举例

10.8 IEC 逻辑指令

没有非标准 IEC 逻辑指令。

And, Or, 异或



与 (AND) 功能对 IN1 和 IN2 的对应位进行与运算，并把结果存入 OUT。

或 (OR) 功能对 IN1 和 IN2 的对应位进行或运算，并把结果存入 OUT。

异或 (XOR) 功能对 IN1 和 IN2 的对应位进行异或运算，并把结果存入 OUT。

输入和输出数据类型可以变化，但必须相同。

使 ENO=0 的出错条件：SM4.3 (运行时间)，0006 (间接寻址)

这些指令影响下列的特殊存储器位：SM1.0 (零位)

输入/输出	操 作 数	数据类型
IN1, IN2	VB, IB, QB, MB, SB, SMB, LB, VW, IW, QW, MW, SW, SMW, AIW, T, C, LW, VD, ID, QD, MD, SD, SMD, LD, HC, AC, 常数, *VD, *AC, *LD	BYTE, WORD DWORD
OUT	VB, IB, QB, MB, SB, SMB, LB, VW, IW, QW, MW, SW, SMW, T, C, LW, VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD	BYTE, WORD DWORD

AND，OR，异或指令举例

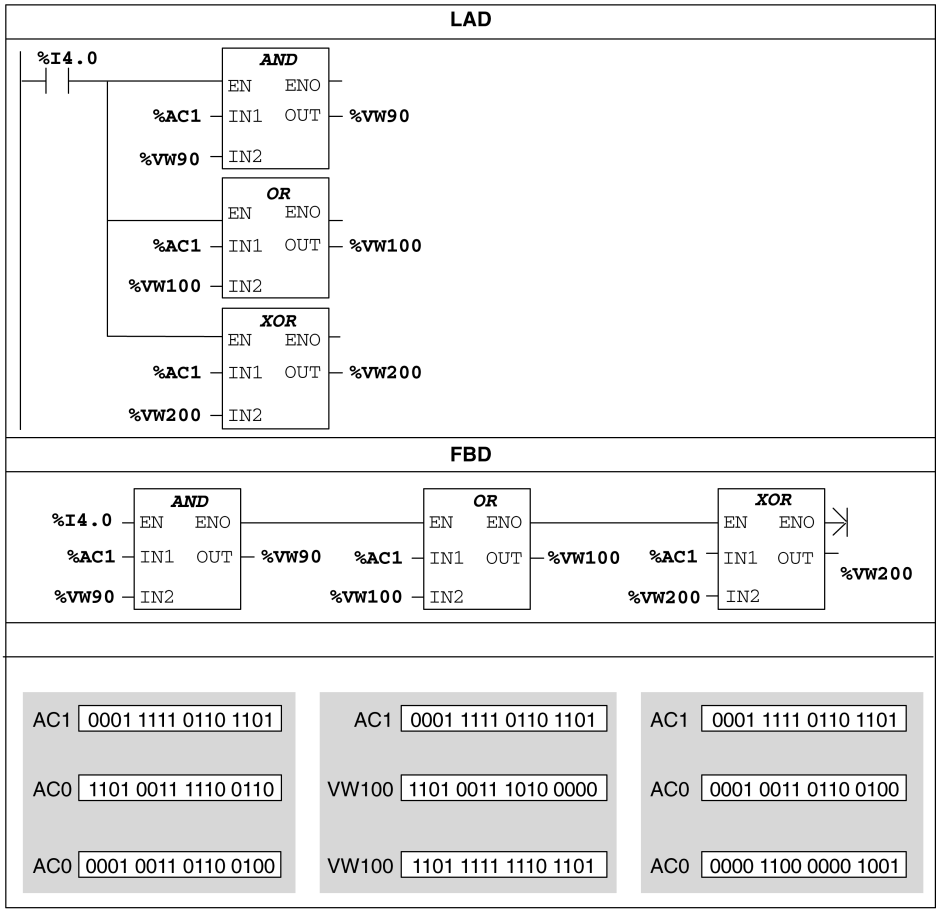
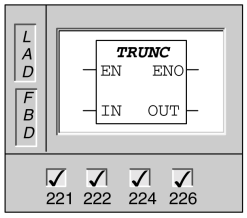


图 10-10　　AND，OR，异或功能举例

取反



反功能把 IN 端的输入值进行取反运算，并把结果送到 OUT 端。

输入和输出数据类型可以变化，但必须相同。

使 ENO=0 的出错条件：SM4.3 (运行时间)，0006 (间接寻址)。

这条指令影响下列特殊存储器位：SM1.0 (零位)

输入/输出	操　作　数	数据类型
IN	VB, IB, QB, MB, SB, SMB, LB, VW, IW, QW, MW, SW, SMW, AIW, T, C, LW, VD, ID, QD, MD, SD, SMD, LD, HC, AC, 常数 , *VD, *AC, *LD	BYTE, WORD DWORD
OUT	VB, IB, QB, MB, SB, SMB, LB, VW, IW, QW, MW, SW, SMW, T, C, LW, VD, ID, QD, MD, SD, SMD, LD, AC,*VD, *AC, *LD	BYTE, WORD DWORD

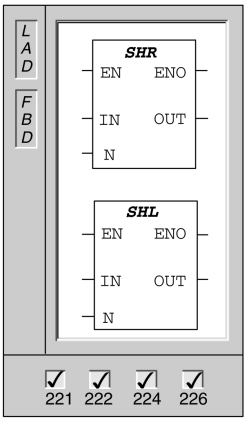
10.9 IEC 移位和循环指令

表 10-7 是非标准 IEC 移位指令的页参考。

表 10-7 非标准 IEC 移位指令

描　　述	页
移位寄存器位指令	9–90

逻辑右移，逻辑左移



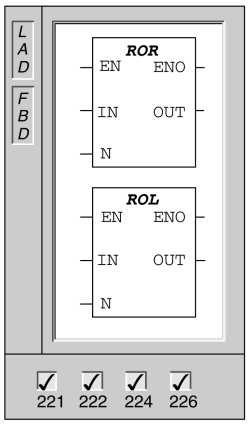
右移功能把输入端 (IN) 的值向右移动N端指定的位数，结果存入 OUT 指定的变量中。当右移时，用 0 填充被右移出的位。

左移功能把输入端 (IN) 的值向左移动N端指定的位数，结果存入 OUT 指定的变量中。当左移时，用 0 填充被左移出的位。

使 ENO=0 的出错条件：SM4.3 (运行时间)，0006 (间接寻址)。

输入/输出	操　作　数	数据类型
IN	VB, IB, QB, MB, SB, SMB, LB, VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, VD, ID, QD, MD, SD, SMD, LD, HC, AC, 常数, *VD, *LD, *AC	BYTE, WORD DWORD
N	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *LD, *AC	BYTE
OUT	VB, IB, QB, MB, SB, SMB, LB, VW, IW, QW, MW, SW, SMW, LW,T, C, VD, ID, QD, MD, SD, SMD, LD, AC *VD, *LD, *AC	BYTE, WORD DWORD

逻辑循环右移，逻辑循环左移



循环右移和循环左移功能把输入值 (IN) 向右或向左循环移动某个计数位 (N)，并把结果保存到输出 (OUT)。

循环是环型的，在 ROR 中，位 0 被循环移动到最高有效位；在 ROL 中，最高有效位被移动到位 0。

使ENO = 0的错误条件：SM4.3 (运行时间)，0006 (间接寻址)。

输入/输出	操 作 数	数据类型
IN	VB, IB, QB, MB, SB, SMB, LB, VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, VD, ID, QD, MD, SD, SMD, LD, HC, AC, 常数, *VD, *LD, *AC	BYTE, WORD DWORD
N	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *LD, *AC	BYTE
OUT	VB, IB, QB, MB, SB, SMB, LB, VW, IW, QW, MW, SW, SMW, LW, T, C, VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC	BYTE, WORD DWORD

移位和循环指令举例

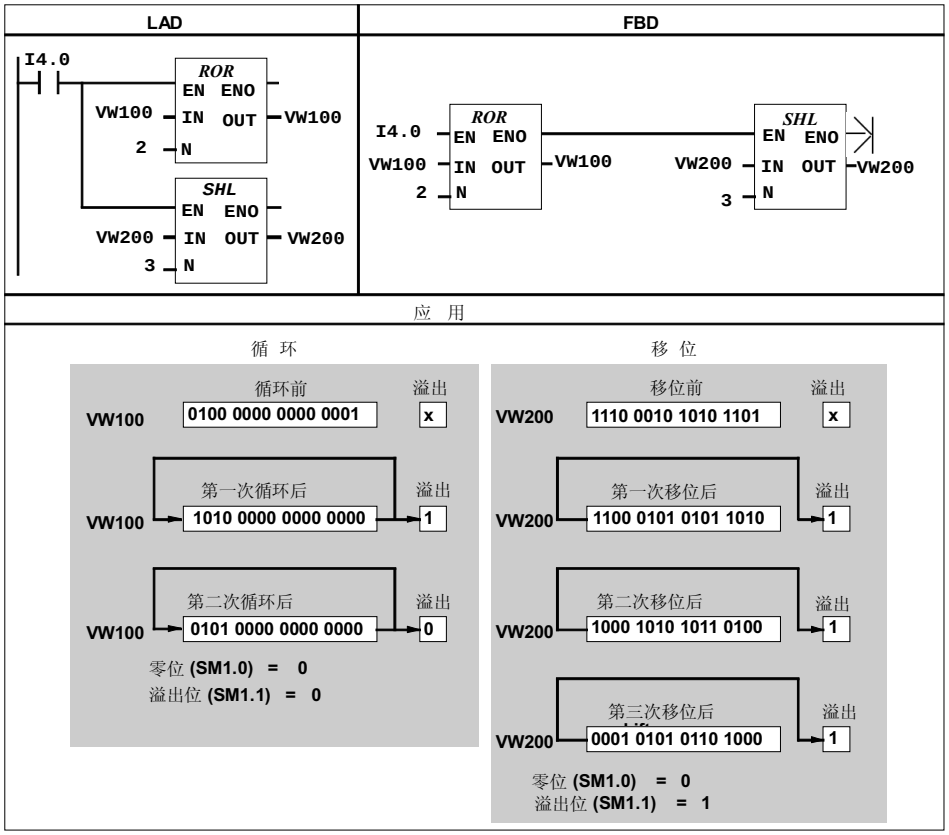


图 10-11 LAD 和 FBD 的移位和循环指令举例

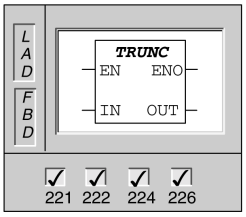
10.10 IEC 转换指令

表 10-8 是非标准IEC转换指令的页参考。

表 10-8 非标准 IEC 转换指令

描　　述	页
解码指令	9–100
编码指令	9–100
段码指令	9–102
ASCII 到 Hex，Hex 到 ASCII 指令	9–103
整数到 ASCII 指令	9–104
双整数到 ASCII 指令	9–105
实数到 ASCII 指令	9–106

取整



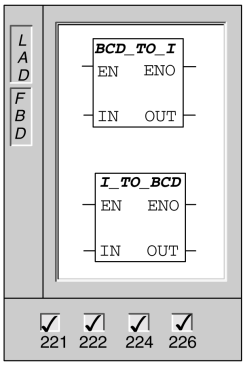
取整功能把一个实数 (IN) 转换成一个整数，并把结果放入 OUT中。不执行舍入。

使 ENO = 0 的错误条件：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)。

这些功能影响下列特殊存储器位：SM1.1 (溢出)

输入/输出	操　作　数	数据类型
IN	VD, ID, QD, MD, SD, SMD, LD, AC, 常数, *VD, *AC,* LD	REAL
OUT	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD	DINT

BCD 码到整数，整数到 BCD 码



BCD 码到整数转换功能把一个 BCD 数值 (IN) 转换成一个整数值，并把结果装入 OUT 指定的变量中。

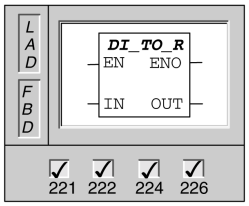
整数到 BCD 码转换功能把一个整数值 (IN) 转换成一个 BCD 值，并把结果装入 OUT 指定的变量中。

使 ENO = 0 的错误条件：SM1.6 (BCD)，SM4.3 (运行时间)，0006 (间接寻址)。

这些功能影响下列特殊存储器位：SM1.6 (无效的 BCD 码)

输入/输出	操　作　数	数据类型
IN	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, AC, 常数, *VD, *LD, *AC	WORD
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *AC, *LD	WORD

双整数到实数转换

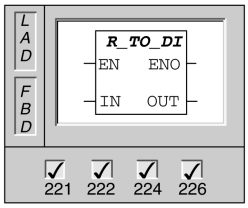


双整数到实数的转换功能把一个双整数值 (IN) 转换成一个实数值，并把结果装入 OUT 指定的变量中。

使 ENO = 0 的错误条件： SM4.3 (运行时间)，0006 (间接寻址)。

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD,SD, SMD, LD, HC, AC, 常数,*VD,*LD, *AC	DINT
OUT	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC	REAL

实数到双整数转换

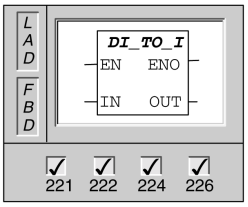


实数到双整数的转换功能把一个实数值 (IN) 转换成一个双整数值，并把结果装入 OUT 指定的变量中。

使 ENO = 0 的错误条件： SM4.3 (运行时间)，0006 (间接寻址)。

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SD, SMD, LD, AC, 常数,*VD,*LD, *AC	REAL
OUT	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC	DINT

双整数到整数转换



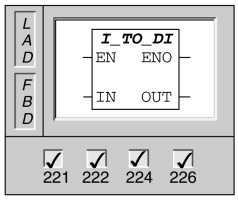
双整数到整数的转换功能把一个双整数值 (IN) 转换成一个整数值，并把结果装入 OUT 指定的变量中。

使 ENO = 0 的错误条件： SM4.3 (运行时间)，0006 (间接寻址)。

该功能影响如下的特殊存储器位：SM1.1 (溢出)

输入/输出	操 作 数	数据类型
IN	VD, ID, QD, MD, SD, SMD, LD, HC, AC, 常数,*VD,*LD, *AC	DINT
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC	INT

整数到双整数转换

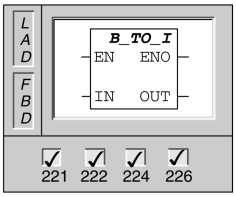


整数到双整数的转换功能把一个整数值 (IN) 转换成一个双整数值，并把结果装入 OUT 指定的变量中。

使 ENO = 0 的错误条件： SM4.3 (运行时间)，0006 (间接寻址)。

输入/输出	操 作 数	数据类型
IN	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, 常数, AC, *VD,*LD, *AC	INT
OUT	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC	DINT

字节到整数

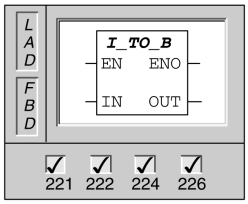


字节到整数的转换功能把一个字节值 (IN) 转换成一个整数值，并把结果装入 OUT 指定的变量中。

使 ENO = 0 的错误条件：SM4.3 (运行时间)，0006 (间接寻址)。

输入/输出	操 作 数	数据类型
IN	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *LD, *AC	BYTE
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC	INT

整数到字节转换



整数到字节的转换功能把一个整数值 (IN) 转换成一个字节值，并把结果装入 OUT 指定的变量中。

使 ENO = 0 的错误条件：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)

该功能影响如下的特殊存储器位：SM1.1 (溢出)。

输入/输出	操 作 数	数据类型
IN	VW, IW, QW, MW, SW, SMW, LW, T ,C, AIW, AC, 常数, *VD, *LD, *AC	INT
OUT	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC	BYTE

转换程序举例

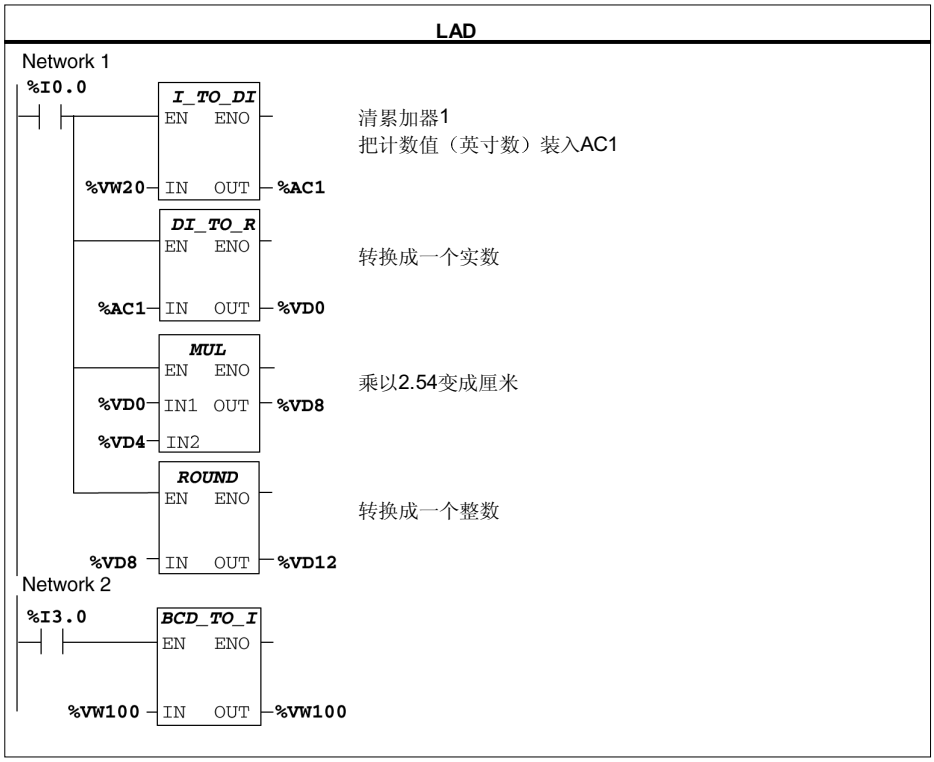


图 10-12　　实数转换指令的 LAD 程序举例

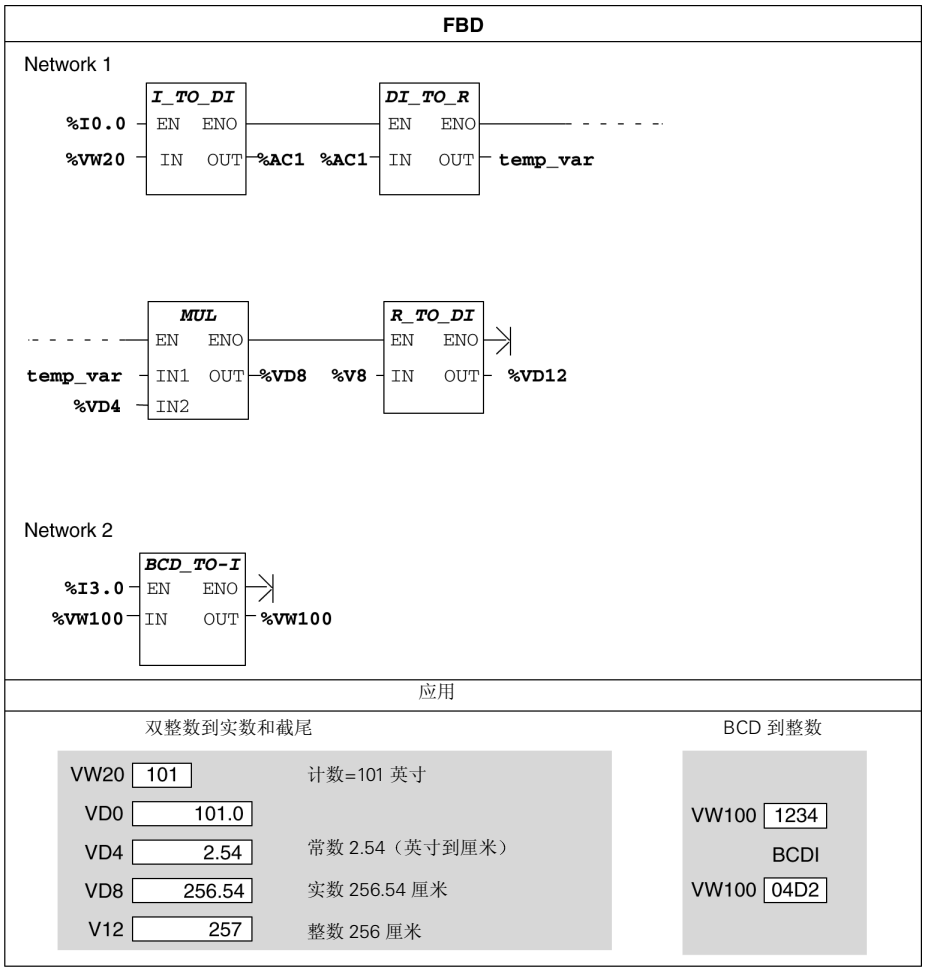


图 10-13　实数转换指令的 FBD 程序举例

使用 USS 协议指令和变频器通信 11

本章描述能使 S7-200 控制 MicroMaster 变频器的标准 USS 协议指令。USS 协议指令是 STEP 7-Micro/WIN 32 软件工具包一个组成部分。

STEP 7-Micro/WIN 32 软件工具包通过专为 USS 协议通信而设计的预配置子程序和中断程序，使 MicroMaster 变频器的控制更为方便。这些程序在STEP 7-Micro/WIN 指令树的库文件夹中作为指令出现的。使用这些新指令可控制变频器和读/写变频器参数。

当你选择 USS 协议指令时，会自动添加一个或几个有关的子程序 (USS 1 至 USS 7)，而不需编程者的参与。

本章概况

11.1 USS 协议指令的要求

USS 协议指令需要能提供以下资源的 CPU：

- 容量1250 字节至 1750 字节 (取决于所使用的指令)
- 端口 0 (仅一个)
- 8 个子程序和 3 个中断
- 400字节容量的 V 存储器和 一个用于某些指令的16 字节缓冲区

资源利用

USS 协议指令应用以下资源：

- 端口 0: 当端口 0 用于 USS 协议通信时，它不能再用于其它目的，包括与 STEP 7-Micro/WIN 的通信。USS_INIT 指令控制端口 0 分配给USS 协议或 PPI 。一旦端口 0 分配给 USS 使用，只有通过其它 USS_INIT 指令的程序再分配，或将MODE 开关置于在 STOP，才能重新用于 STEP 7-Micro/WIN 通信。停止与变频器的通信会导致变频器停止工作，建议在 USS 协议应用程序的程序开发过程中，开发者应用一个CPU 226 或EM 277 PROFIBUS –DP与 PC 中的一个 PROFIBUS CP 卡一起使用。当USS协议运行时，这样做能提供第二个通信端口，用于监视 STEP 7-Micro/WIN 应用程序。
- 在端口0上，与自由端口通信有关的所有SM位置，都会受USS协议指令的影响。
- 用户程序空间：除了被每个指令占用的空间外，尚有由USS协议程序所占用的最多为1750字节的用户程序空间开销。
- V存储器：一个从用户分配的存储单元开始的V存储器的400字节程序块是为USS变量保存的。一个从用户分配的存储单元开始的16字节通信，缓冲区是某些指令所需要的，建议为USS协议指令的每个实例分配一个唯一的缓冲区。
- 9个子例行程序和3个中断都取自USS协议。

全局符号表配置

当为第一个符号输入一个地址后，全局符号表自动计算和分配表中的剩余符号。图11-1表示符号表的USS标记。需要为符号表分配一个占用400字节的起始V区位置。

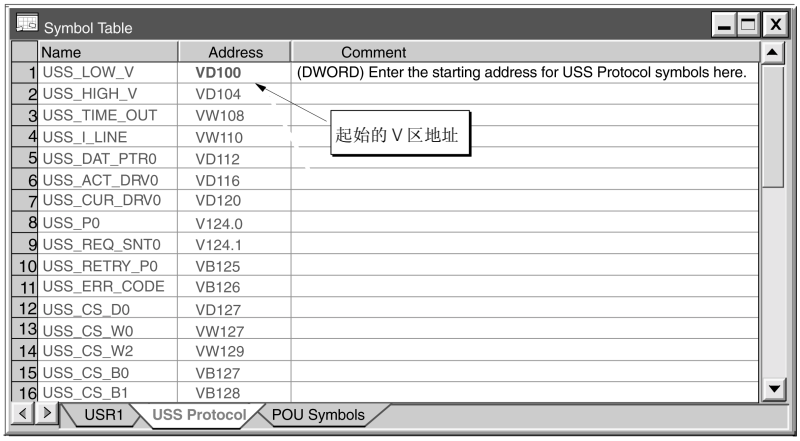


图11-1 符号表配置

变频器通信时间

与变频器的通信，对CPU扫描是异步的。完成一个变频器通信事务通常需要几次 CPU 扫描。这取决于连接的变频器数目，波特率，以及 CPU 的扫描时间。表11-1表示通信处理时间。一旦USS_INIT指令将端口0分配给USS协议，CPU有规律地按表11-1中的时间间隔轮询所有有效的变频器。必须为每个变频器设定超时参数以适应这种轮询时间。

表11-1　变频器能信时间

波特率	轮询有效变频器的间隔时间（ms）
1200	（460最大/230典型）*变频器数
2400	（240最大/120典型）*变频器数
4800	（130最大/65典型）*变频器数
9600	（80最大/40典型）*变频器数
19200	（50最大/25典型）*变频器数

约束

一次只能启动一个READ_PM或WRITE_PM指令。在用户逻辑启动一个新指令之前，每个指令的Done输出应发出输出完成的信号。对每个变频器只能使用一个DRV_CTRL指令。

11.2　编程顺序

使用 USS 协议指令的编程顺序如下：

1. 将 USS_INIT 指令置于用户程序会自动将几个隐含的子程序和中断程序加到程序内。只能通过一次扫描调用USS_INIT以启动或改变 USS 通信参数。有关USS_INIT指令的更详细情况，见11-4页。
2. 将一个V存储器地址分配给 USS 全局符号表中的第一个存储单元。所有其它地址都是自动地分配的。总共需要 400 连续字节。选择图11-1中的USS标签即可看到相关的符号表。
3. 在用户程序内每一个激活变频器只能有一条 DRV_CTRL。可以任意添加 READ_PM 和 WRITE_PM 指令，但是每次只能激活其中的一个指令。
4. 配置变频器参数，以便与程序中所用的波特率和地址相匹配。参阅 11.5 节中的“变频器设置”。
5. 连接 CPU 和变频器之间的通信电缆。非常重要的是，连接到变频器的任何控制设备(如PLC)，均需用一根短、粗的电缆连接到和变频器相同的接地点或星形接线的中点。

注意

有不同参数电位的设备互连会在互连电缆中流通不希望有的电流。

这些不希望有的电流会引起通信故障或损坏设备。

要确实保证，通信电缆连接的所有设备，或是共用一个公共电路参考点，或是相互隔离的，以防止不希望有的电流流通。见 2.3 节中的“用隔离电路的接地和电路参考点”。

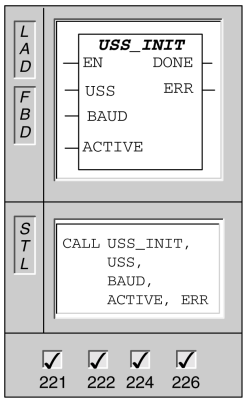
屏蔽线必须连接到机箱接地点或9针连接器的插针 1。建议将 MicroMoster 变频器上的端子 2-0V 连接到机箱接地点。

注

如不能读出 USS 指令块上的所有变量，从菜单上选择 **View►Zoom**，然后增加栏的宽度即可。

11.3 USS 协议指令

USS_INIT



USS_INIT 指令用于允许和初始化或禁止 MicroMaster变频器通信。在可以使用任何其它 USS 协议指令之前，必须先执行 USS_INIT 指令且没有错误返回。指令执行完后，完成位（Done bit）立即置位，然后才能继续执行下一条指令。

当 EN 输入为接通时，每一次扫描执行指令。每一次要改变通信状态，必须精确地执行一次 USS_INIT 指令。因此，应通过一个边沿跳变检测指令来检测 EN 的脉冲接通。一旦 USS 协议已启动，在改变初始化参数之前，必须通过执行一个新的 USS_INIT 指令以禁止 USS 协议。

USS 输入的值选择通信协议。1 将端口 0 分配给 USS 协议和允许该协议。 0将端口0分配给PPI并禁止USS协议。

BAUD 设定波特率在 1200，2400，4800，9600，或 19200。

ACTIVE 指示哪 一个变频器是激活的。有些变频器只支持地址 0 至 30。图 11-2 表示激活变频器输入的描述和格式。标记为 ACTIVE 的任何变频器都是自动地在后台进行轮询控制的，汇集状态，并防止变频器的串行链路超时。参考 11-3 页的表 11-1，计算状态轮询之间的时间。有关串行链路超时参数 P093 如何配置，参数11.5节。

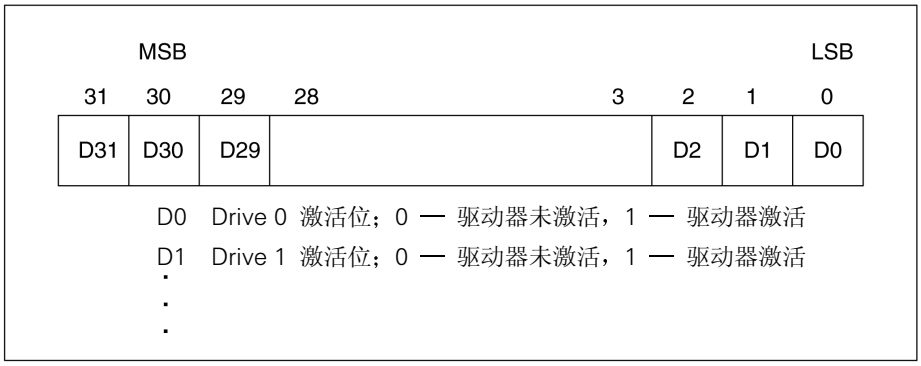


图 11-2 激活变频器的描述和格式

当 USS_INIT 指令完成时，DONE输出接通，ERR 输出字节包含指令执行的结果。11-6 页上的表 11-6 定义在指令执行中可能出现的错误类型，表11-2 表示 USS 子程序的操作数和数据类型。

表11-2 USS_INIT 子例行程序用的操作数和数据类型

输入/输出	操作数	数据类型
USS	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *AC, *LD	字节
BAUD	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, Constant, AC *VD, *AC, *LD	字
ACTIVE	VD, ID, QD, MD, SD, SMD, LD, AC, Constant, *VD, *AC, *LD	双字
DONE	I, Q, M, S, SM, T, C, V, L	布尔数
ERR	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	字节

图 11-3 表示如何使用 LAD，FBD 和 STL 语言编写 USS_INIT 子程序。

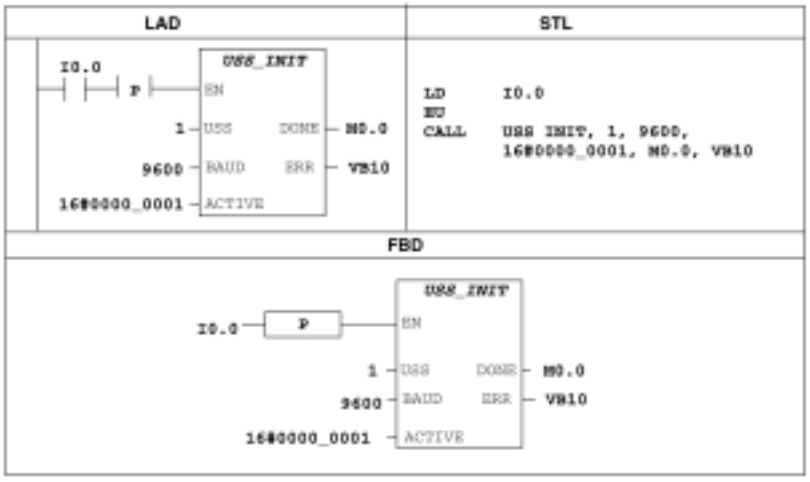
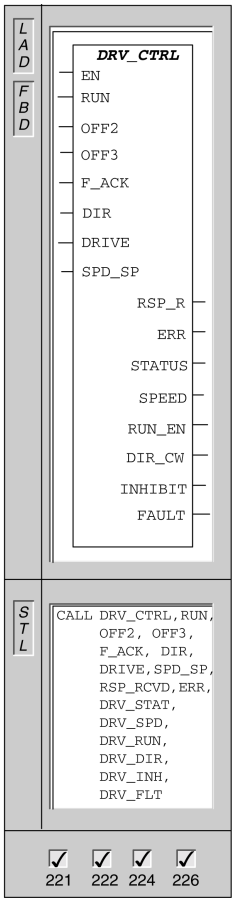


图 11-3　在 USS_INIT 子程序中使用 LAD，FBD和STL语言。

DRV_CTRL



DRV_CRTL 指令用于控制 ACTIVE MicroMaster 变频器。DRV_CRTL 指令将选择的命令放在一个通信缓冲区内。缓冲区中的命令发送到编址的变频器（DRIVE 参数），如果该变频器已由 USS_INIT指令的 ACTIVE 参数选 中的话。每个变频器只应有一个 DRV_ CTRL 指令。

EN 位必须接通，以启动 DRV_CRTL 指令。典型地，这个指令总是在允许状态。

RUN（RUN/STOP）指示变频器是接通（1）或是断开（0）。当RUN位是接通时，MicroMaster 变频器收到一个命令，以便开始以规定的速度和方向运动。为了使变频器运动，必须具备以下条件：

- 在USS_INIT 中将变频器激活。
- OFF1 和 OFF2必须设定为 0。
- FAULT 和 INHIBIT 必须为0

当 RUN 断开时，则发送 MicroMaster 变频器一个命令，电动机速度降低一直到停止。

OFF2 位用来使 MicroMaster 变频器减速到停止。OFF3 位用来命令 MicroMaster 变频器快速停止。

F_ACK（故障确认）位用来确认一个故障。当 F_ACK 从低变高时，变频器清除故障（FAULT）。

DIR（方向）位指示变频器应向那个方向运动（0-逆时针方向，1-顺时针方向）。

DRIVE（变频器地址）输入是 DRV_CTRL 命令发送给 MicroMaster 变频器的地址。有效地址为 0 至 31。

SPD_SP（速度设定点）是全速度百分值的变频速度（-200.0% 至 200.0%）。SPD_SP 负值使变频器反方向旋转。

注

每台变频器只能分配一个 DRV_CTR 指令。

RSP_R(收到响应)位确认从变频器来的响应。对所有激活的变频器轮询最新的变频状态信息。每当CPU从变频器收到一个响应，RSP_R位接通进行一次扫描，并更新以下所有的数值。

ERR 是一个错误状态字节，它包含与变频器通讯请求的最新结果。11-16 页上的表 11-6 定义指令执行中可能会出现的错误。

STATUS是由变频器返回的状态字的原始值。图11-4表示标准状态字和主反馈的状态位。

SPEED是全速度有分值的变频器速度（-200.0% 至 200.0%）。

注

有些变频器只报告正值的速度。如果速度是负值，变频器仍报告正值的速度，但是将 DIR_CW（方向）位反向。

RUN_EN（RUN允许）指示变频器正在运行（1）或已停止（0）。

DIR_CW 指示变频器的旋转方向（0-逆时针方向，1-顺时针方向）。

INHIBIT 指示变频器上的禁止位的状态（0-不禁止，1-被禁止）。要清除禁止位，FAULT 位必须断开，RUN，OFF2，以及 OFF3 输入也必须断开。

FAULT 指示故障位的状态（0-无故障，1-故障）。变频器显示故障代码。（参阅变频器使用手册）。要清除 FAULT位，需消除故障原因，并接通 F_ACK位。

表 11-3 表示 DRV_CTRL 子程序的操作数和数据类型。

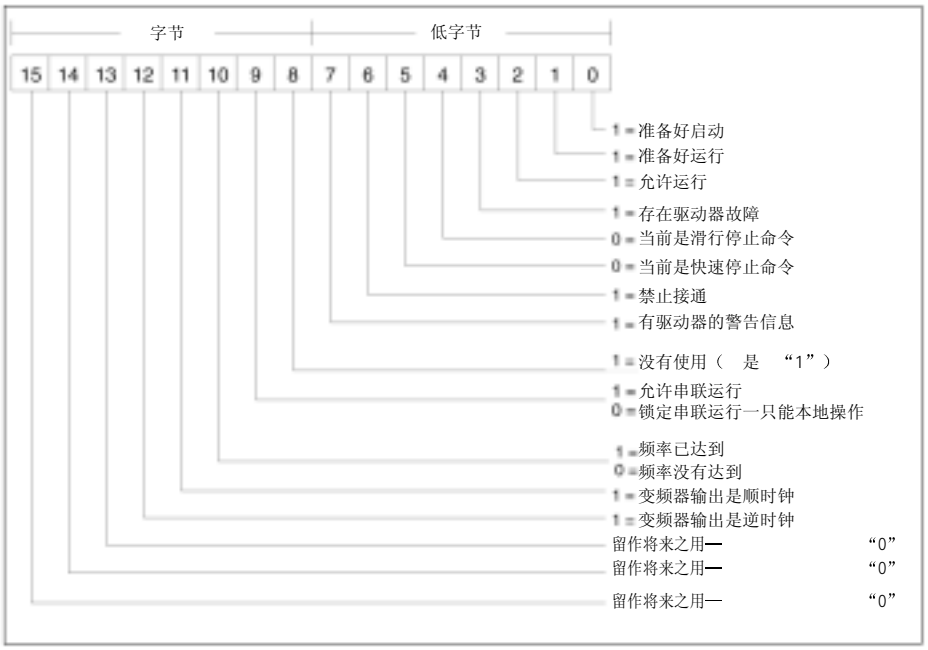


图 11-4 标准状态字和主要反馈的状态位

表 11-3　DRV_CTRL 子程序的操作数和数据类型

输入/输出	操作数	数据类型
RUN	I, Q, M, S, SM, T, C, V, L, 功率流	布尔数
OFF2	I, Q, M, S, SM, T, C, V, L, 功率流	布尔数
OFF3	I, Q, M, S, SM, T, C, V, L, 功率流	布尔数
F_ACK	I, Q, M, S, SM, T, C, V, L, 功率流	布尔数
DIR	I, Q, M, S, SM, T, C, V, L, 功率流	布尔数
DRIVE	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD	字节
SPD_SP	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD, 常数	实数
RSP_R	I, Q, M, S, SM, T, C, V, L	布尔数
ERR	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	字节
STATUS	VW, T, C, IW, QW, SW, MW, SMW, LW, AC, AQW, *VD, *AC, *LD	字
SPEED	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD	实数
RUN_EN	I, Q, M, S, SM, T, C, V, L	布尔数
DIR_CW	I, Q, M, S, SM, T, C, V, L	布尔数
INHIBIT	I, Q, M, S, SM, T, C, V, L	布尔数
FAULT	I, Q, M, S, SM, T, C, V, L	布尔数

图 11-5　说明如何使用 LAD，FBD和 STL 语言来编写 DRV_CTRL 子程序。

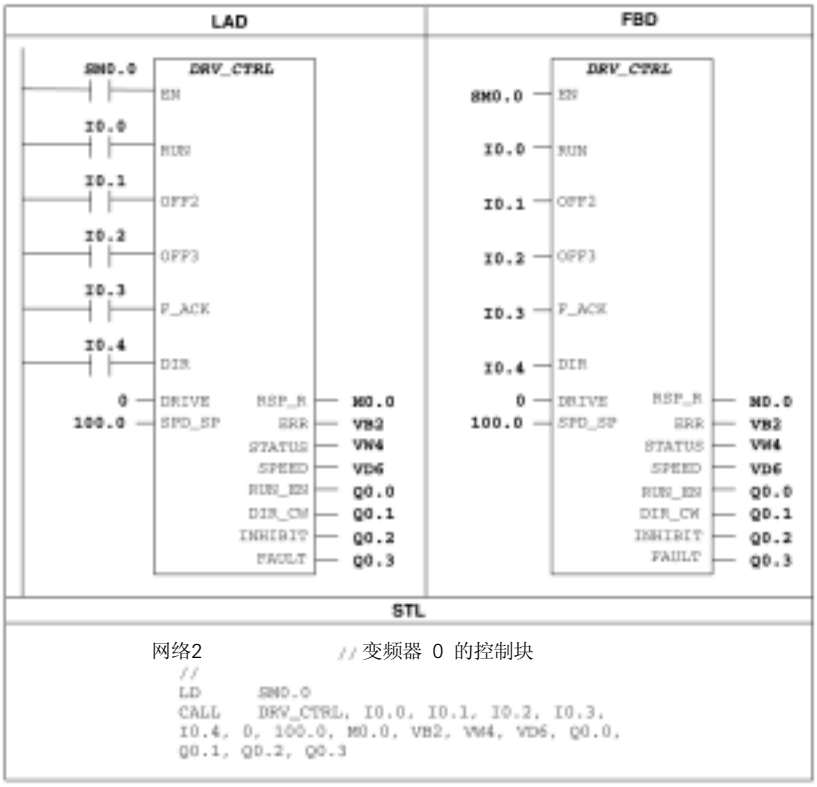
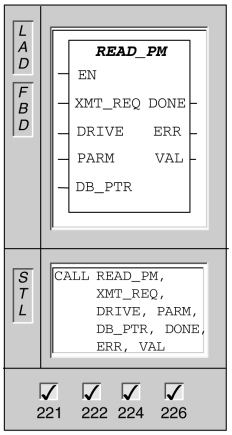


图 11-5　在 DRV_CTRL 子程序中使用LAD，FBD 和 STL语言。

READ_PM



READ_PM 指令读取变频器参数。当 MicroMaster 变频器确认接收到命令时或当发送一个出错状况时，则完成 READ_PM 指令的处理。在该处理等待响应时，逻辑扫描仍继续进行。EN 位必须接通以启动发送请求。这个位应保持接通一直到 DONE 位被置位才标志着整个处理结束。

当 XMT_REQ 输入接通时，每一次扫描READ_PM发送请求到变频器，因此，XMT_REQ的输入端必须与脉冲边缘检测语句相联接，使得EN输入端每次上升沿到来时，都会向变频器发出请求。

变频器输入（DRIVE input）是变频器的地址，READ_PM 命令将被发送到这个地址。每个变频器的有效地址为 0 到 31。

PARM 是参数号

必须将 16 字节缓冲区的地址提供给 DB_PTR输入。READ_PM 指令使用这个缓冲区以存储向变频器所发送命令的结果。

READ_PM 指令完成时，DONE输出接通和 ERR 输出字节包含执行这个指令的结果。第 11-16 页上的表 11-6定义在执行指令时有可能出现的错误类型。

VAL 是返回的参数值。

注

在同一时间内，只能激活一个 READ_PM 或一个 WRITE_PM 指令。

表 11-4　READ_PM 子指令的操作数和数据类型

表 11-4　READ_PM 子程序中的操作数和数据类型

输入/输出	操作数	数据类型
XMT_REQ	I, Q, M, S, SM, T, C, V, L, 通过上升沿跳变检测指令触发	布尔数
DRIVE	VB, IB, QB, MB, SB, SMB, LB, AC, 常数, *VD, *AC, *LD	字节
PARM	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, 常数, AC *VD, *AC, *LD	字
DB_PTR	&VB	双字
DONE	I, Q, M, S, SM, T, C, V, L	布尔数

表 11-14　用于 READ_PM 子程序的操作数和数据类型

输入/输出	操作数	数据类型
ERR	VB, IB, QB, MB, SB, SMB, LB, AC. *VD, *AC, *LD	字节
VAL	VW, T, C, IW, QW, SW, MW, SMW, LW, AC, AQW, *VD, *AC, *LD	字

图 11-6　说明如何使用 LAD，FBD 和 STL 语言来编写 READ_PM 子程序。

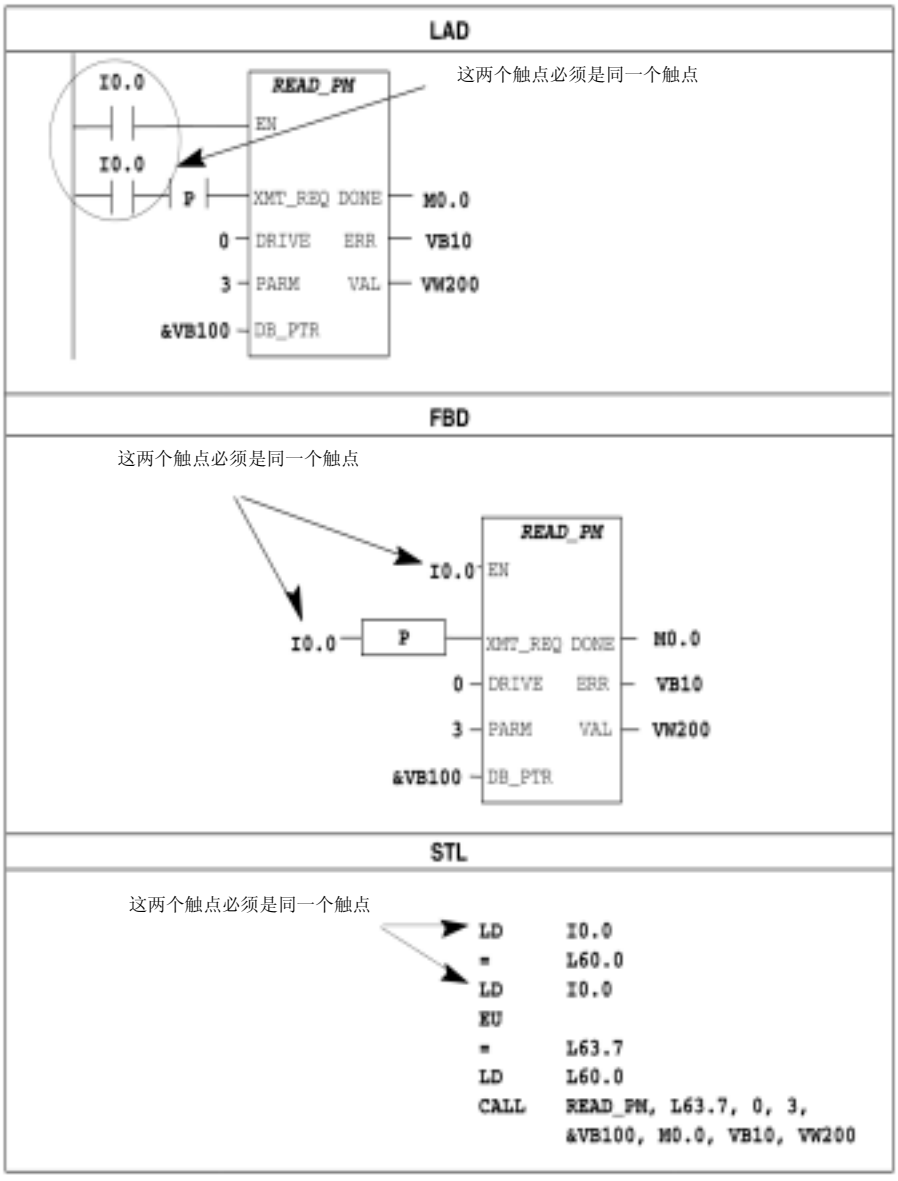
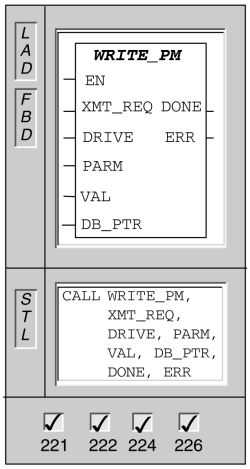


图 11-6　在 READ_PM 子程序中使用 LAD，FBD 和 STL 语言。

WRITE_PM



WRITE_PM 指令将变频器参数写入到指定的位置，当变频器确认接收到命令或发送一个出错状况时，则完成 WRITE_PM指令处理，在该处理等待响应时，逻辑扫描仍继续进行。

EN 位必须接通以发送请求。这个位应保持接通一直到 DONE 位被设置表明已完成了处理。当XMT_REQ 输入接通时，每一次扫描，WRITE_PM的请求被发送到变频器。因此，XMT_REQ 的输入端必须与脉冲边缘检测语句相联接，使得EN输入端每次上升沿到来时，都会向变频器发出请求。

变频器输入（DRIVE input）是MicroMaster变频器的地址，WRITE_PM命令发送到这个地址，每个变频器的有效地址为 0 到 31。

PARM是参数号，VAL 是要写入的参数值。

必须将16字节缓冲区的地址提供给 DB_PTR输入。WRITE_PM 指令使用这个缓冲区以存储向变频器所发出命令的结果。

WRITE_PM 指令完成时，DONE 输出接通和 ERR 的输出字节包含有执行这个指令的结果，第 11-16 页上的表 11-6 定义在执行指令时有可能出现的错误类型。

注

在一个时间内，只能激活一个READ_PM或一个WRITE_PM 指令。

注意

当使用 WRITE_PM 指令来更新保持在变频器 EEPROM 中的参数集时，必须保证不能超过到 EEPROM 的写周期的最大数（约为 50,000次）

超过写周期的最大数会造成所有存储的数据紊乱和随之而来的数据丢失，读周期的数量没有限制。

如需要频繁的写入变频器参数，则首先必须设置 P971（EPROM 存储控制）为 0。

表 11-5 WRITE_PM 子程序中的操作数和数据类型

输入/输出	操作数	数据类型
XMT_REQ	I, Q, M, S, SM, T, C, V, L, 由上升沿检测语句触发	布尔数
DRIVE	VB, IB, QB, MB, SB, SMB, LB, AC. 常数*VD, *AC, *LD	字节
PARM	VW, IW, QW, MW, SW, SMW, LW, T, C, 常数W, 常数, AC *VD, *AC, *LD	字
VAL	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, 常数, AC *VD, *AC, *LD	字
DB_PTR	&VB	双字
DONE	I, Q, M, S, SM, T, C, V, L	布尔数
ERR	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	字节

图 11-7 说明如何使用 LAD，FBD 和 STL 语言来编写WRITE_PM 子程序。

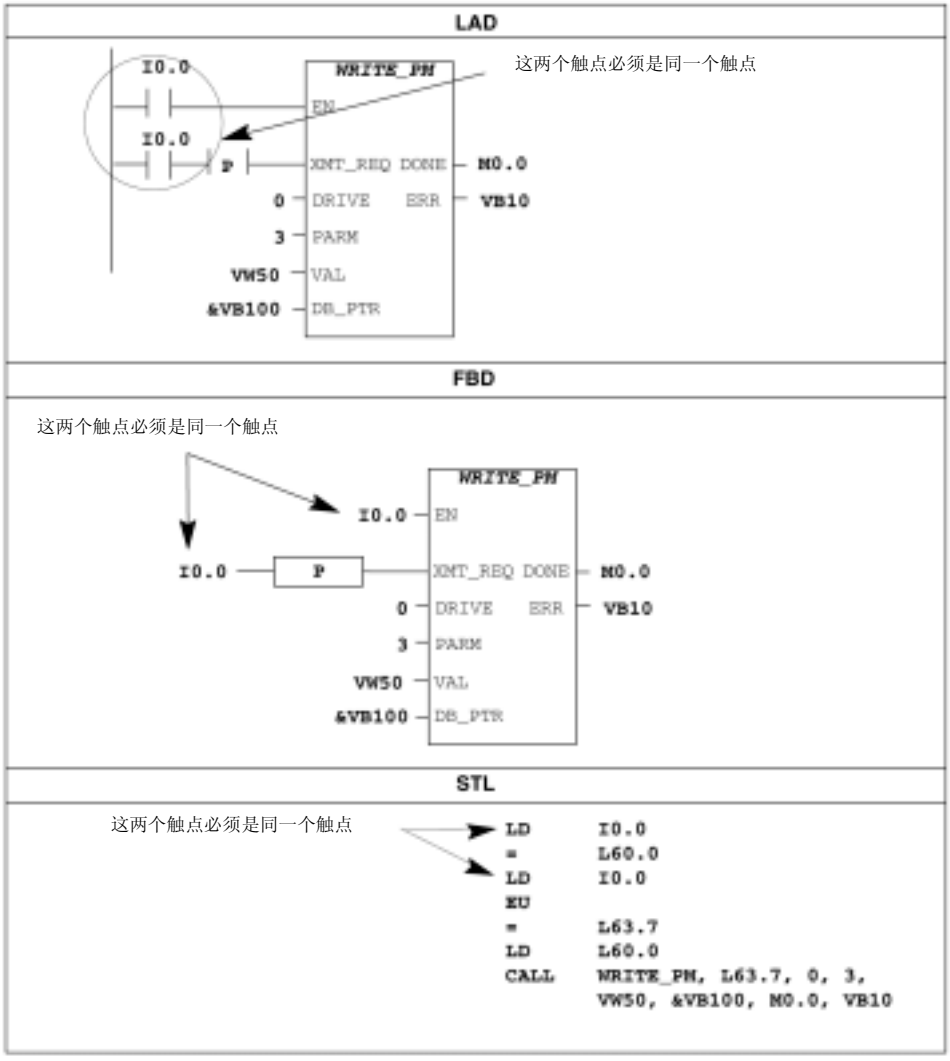


图 11-7 在WRITE_PM 子程序中使用 LAD，FBD 和 STL 语言。

表 11-6 在 USS 指令中出现的执行错误

出错号	说明
0	没有出错
1	变频器不能响应
2	检测到变频器响应中包含加和校验错误
3	检测到变频器响应中包含奇偶校验错误
4	由用户程序干扰引起的错误
5	企图执行非法命令
6	提供非法的变频器地址
7	没有为 USS 协议设置通信口
8	通信口正忙于处理指令
9	输入的变频器速率超出范围
10	变频器响应的长度不正确
11	变频器响应的第一个字符不正确
12	变频器响应的长度字符不正确
13	变频器错误响应
14	提供的 DB_PTR 地址不正确
15	提供的参数号不正确
16	所选择的协议无效
17	USS 激活；不允许更改
18	指定了非法的波特率
19	没有通信；变频器没有激活
20	在变频器中响应中的参数或数值有误

11.4 变频器连接

可用标准的 PROFIBUS 电缆和连接器将 CPU 连接到 MicroMaster 变频器。详细情况见第 7 章中的“网络连接器”。互连电缆正确的偏置和终接，见图 11-8。

注意

把具有不同电位参考点的设备互连会在互连电缆中产生不应有的电流。

这些不应有的电流会引起通信错误或损坏设备。

要确实保证通信电缆连接的所有设备，或是共用一个公共电路参考点，或是相互隔离的，以防止不应有的电流产生。见 2.3 节中的“用隔离电路的接地和电路参考点”。

屏蔽线必须连接到机箱接地点或 9 针连接的插针 1。建议将 MicroMaster 变频器上的端子 2-0V 连接到机箱接地点。

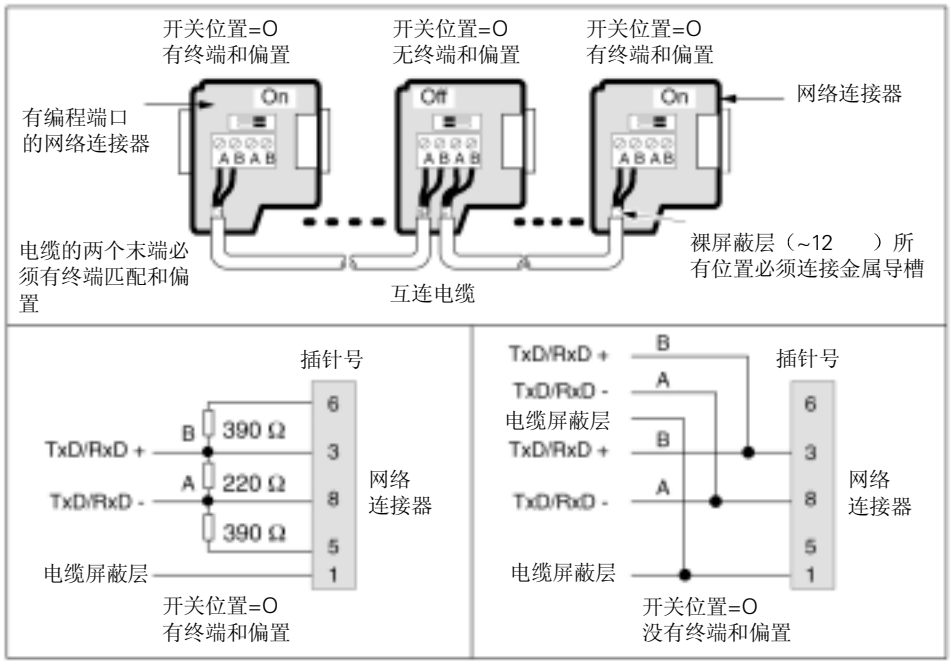


图 11-8 互连电缆的偏置和终端

11.5 变频器的设置

将变频器连接到 PLC 之前，必须确认，变频器已有以下的系统参数。使用变频器小键盘设定参数。

按以下步骤设定变频器参数：

1. 将变频器复位到工厂设定值（可选的）。按 P 键：显示 P000，按向上或向下箭头键，直到显示器显示 P944。按 P 键，输入参数。

   P944=1

2. 允许读/写所有参数。按 P 键。按向上或向下箭头键，直到显示器显示 P009。按 P 键，输入参数。

   P009=3

3. 检查变频器的电动机设定值。设定值随所用的电动机而变。按 P 键。按向上或向下箭头键，直到显示器显示电动机的设定值为止。按 P 键，输入参数。

   P081=电动机的额定频率（Hz）

   P082=电动机的额定速度（RPM）

   P083=电动机的额定电流（A）

   P084=电动机的额定电压（V）

   P085=电动机的额定功率（KW/HP）

4. 设定就地/远程控制方式。按 P 键。按向上或向下箭头键，直到显示器显示 P910。按 P 键，输入参数。

   P910=1远程控制方式

5. 设定 RS-485 串行接口的波特率。按 P 键。按向上或向下箭头键，直到显示出 P092。按 P 键，输入参数。按向上或向下箭头键，直到显示器显示出与你的 RS-485 串行接口相对应的波特率数字为止。按 P 键，输入参数。

P092	3	（1200波特）
	4	（2400波特）
	5	（4800波特）
	6	（9600波特缺省值）
	7	（19200波特）

6. 输入从站地址。每个变频器（最大31）可经过总线运行。按 P 键。按向上或向下箭头键，直到出现 P 091为止。按 P 键，输入参数。按向上或向下箭头键，直到显示器显示出所需要的从站地址。按 P 键输入。

   P091=0至31

7. 增速时间（可选的）。这是以秒表示的电动机加速到最大频率所需的时间。按 P 键。按向上或向下箭头键，直到出现 P002 为止。按 P 键，输入参数。按向上或向下箭头键直到显示器出现所需要的增速时间为止。按 P 键，输入参数。

   P002=0-650.00

8. 斜坡减速时间（任选）。这是以秒表示的电动机减速到完全停止的需要的时间。按 P 键，按向上或向下箭头键，直到出现 P003 为止。按 P 键，输入参数。按向上或向下箭头键直到显示器出现所需要的减速时间为止。按 P 键输入。

   P003=0-650.00

9. 串行链路超时。这是二个输入数据报文之间的最大允许时间间隔。这一特性用于通信发生故障时断开变频器。

   收到了有效的数据报文后，开始定时。如果在规定的时间间隔内没有收到其它的数据报文，变频器跳闸并显示故障代码 F008。将值设定为 0，断开控制回路。变频器进行状态轮询之间的时间可用表 11-1 计算。

   按 P 键。按向上或向下箭头键，直到出现 P093为止。按 P 键输入参数。按向上或向下箭头键直到显示器出现所需要的串行链路超时为止。按 P 键输入。

   P093=0-240（0是缺省值；时间单位为秒）

10. 串行链路额定系统设定点。这个值可以改变，但是典型情况是相当于50Hz或60Hz，它定义了相当于100%的 PVs 或 SPs 的数值。按 P 键。按向上或向下箭头键，直到出现 P094 为止。按 P 键输入参数。按向上或向下箭头键直到显示器出现所需要的串行链路额定系统设定点为止。按 P 键输入

    P094=0-400.00

11. USS 兼容性（可选的）。按P键。按向上或向下箭头键，直到出现 P 095。按 P 键输入参数。按向上或向上箭头键直到显示器出现所需要的，相应于 USS 兼容性的数值为止。按 P 键输入。

    P095=0　0.1 Hz 分辨率（缺省值）

    1　0.01 Hz 分辨率

12. EEPROM 存储器控制（任选）。按 P 键。按向上或向下箭头键，直到出现 P 971。按 P 键输入参数。按向上或向下箭头键直到显示器出现所需要的，相应于EEPROM存储器控制的数值为止。按 P 键输入。

    P971=0　当断电时，丢失更改的参数设定值（包括 P971）

    1　（缺省值）断电期间仍保持更改的参数设定值

13. 运行显示，按 P 键，退出参数方式 P。

USS 协议程序的实例

图 11-9 至 11-11 表示使用 LAD，FBD，STL语言的 USS 程序的实例。

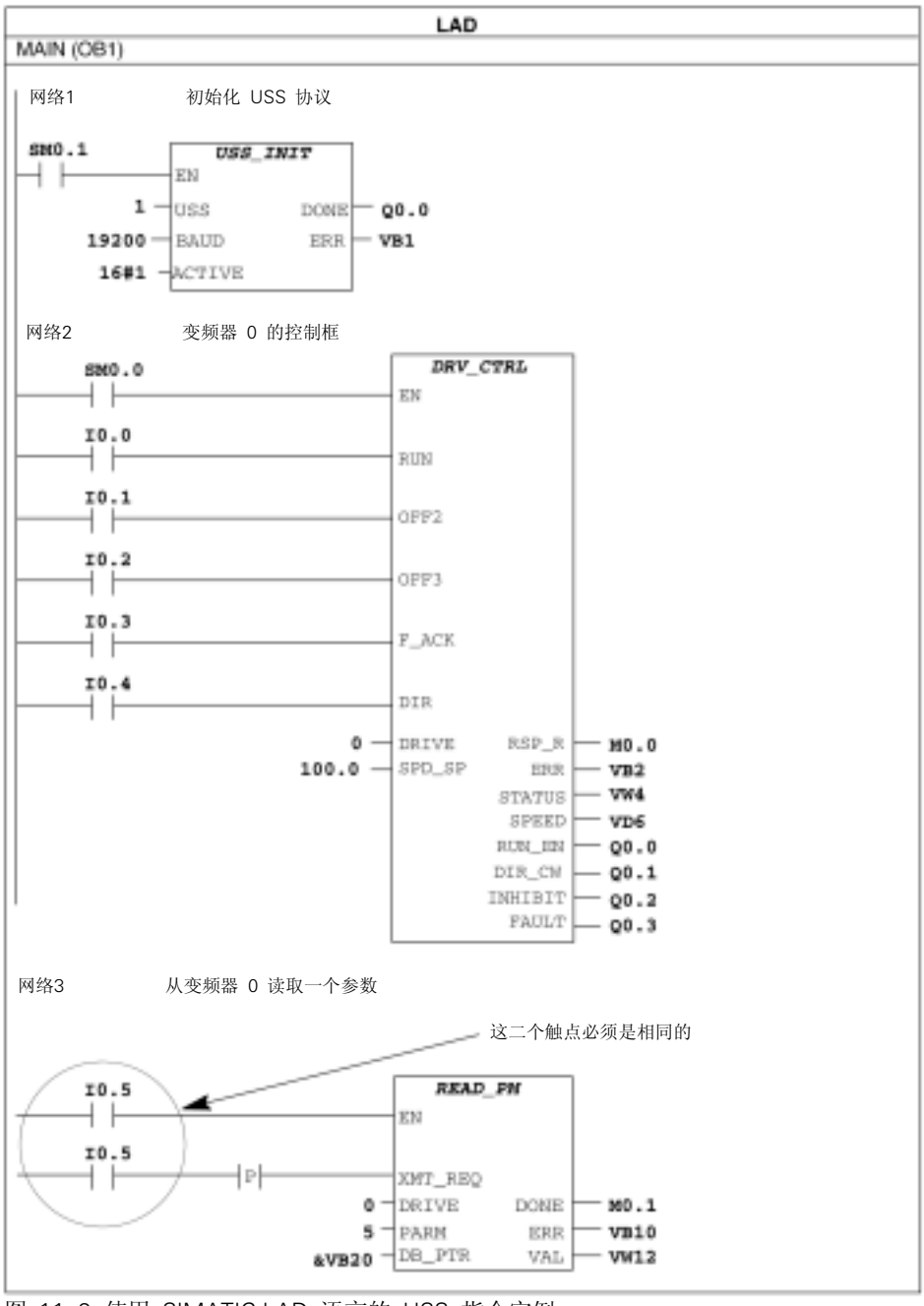


图 11-9 使用 SIMATIC LAD 语言的 USS 指令实例。

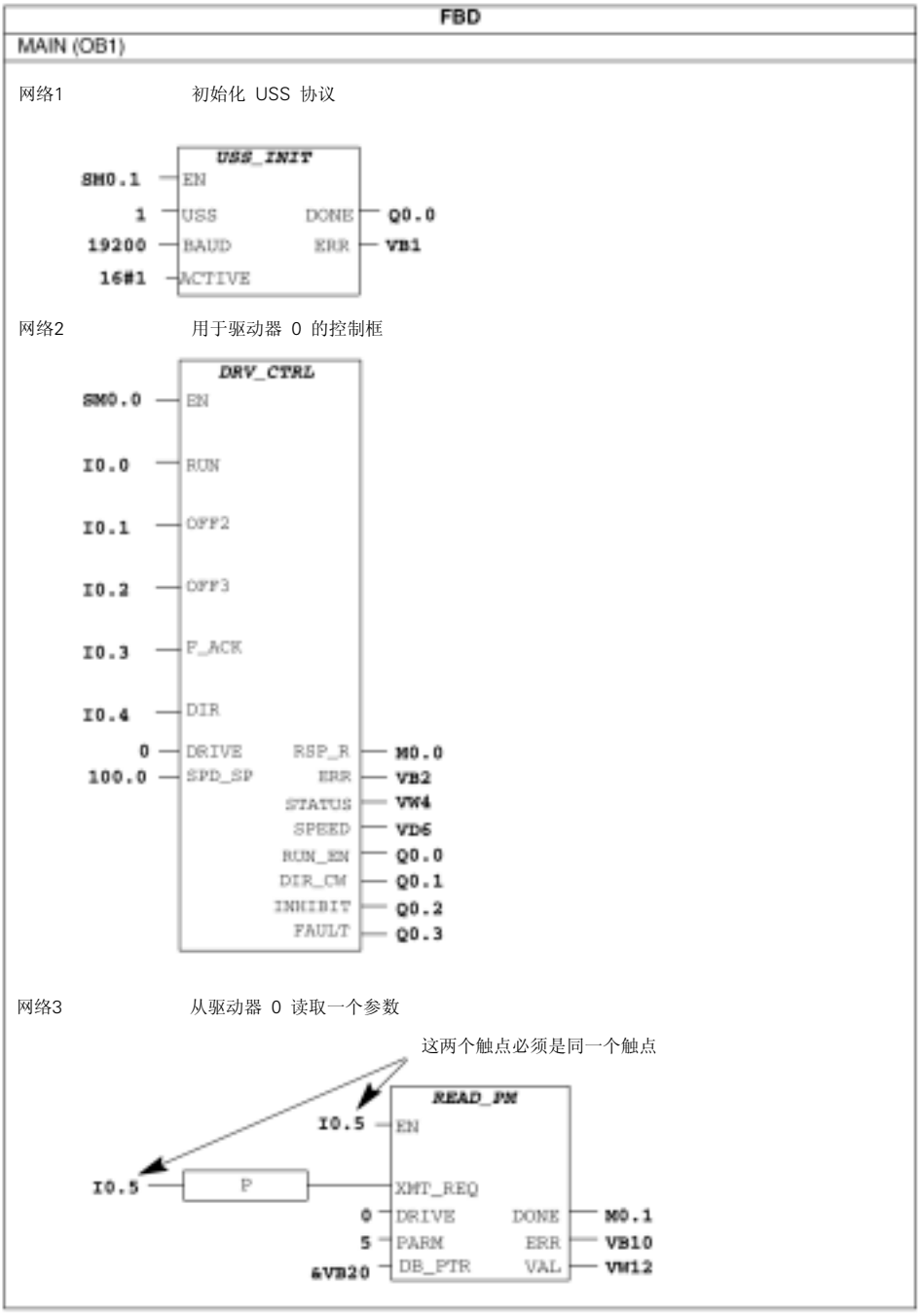


图 11-10　使用 SIMATIC FBD 语言的 USS 指令举例。

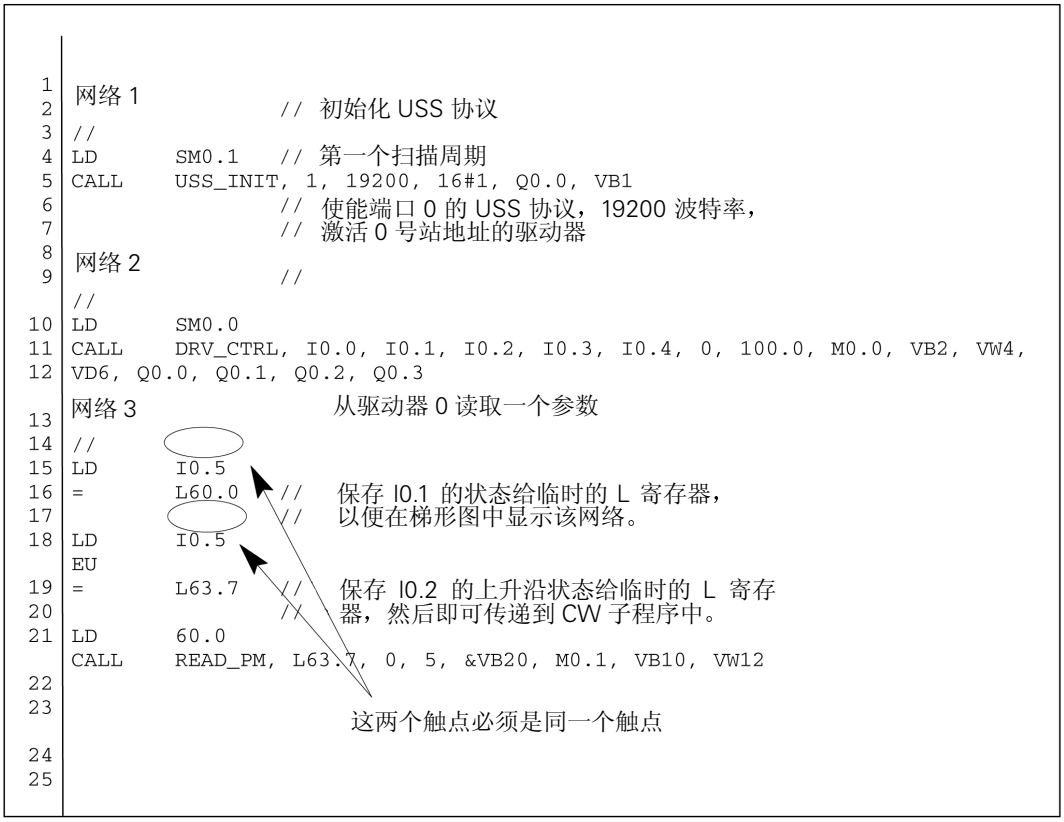


图 11-11 使用 SIMATIC STL 语言的 USS 指令举例

中文 TD 200 使用说明 12

概述

TD 200 文本显示器是所有 SIMATIC S7-200 系列操作员界面问题的最佳解决方法。TD 200 连接很简单，只需用它提供的连接电缆接到 S7-200 系列 PPI 接口上即可。不需要单独的电源。

TD200 具有下列用途：

- 显示信息。
- 在控制系统中起设定和修正参数的作用，例如：改变动作、报警等的设定值，设定实时时钟的时间等。
- 可以提供8个由用户自定义的功能键。
- 提供密码保护功能。

特点

TD 200 具有：

- 牢固的塑料壳，前面板IP65防护等级。
- 27mm的安装深度，无须附件即可安装在箱内或面板内，或用作手持设备。
- 背光LCD液晶显示：即使在逆光情况下也易看清。
- 人体工学设计的输入键位于可编程的功能键上部。
- TD 200中文版内置国标汉字库
- 内置连接电缆的接口。
- 如果TD 200与S7-200系列之间距离超过2.5 m，需接额外电源。这时用Profibus总线电缆连接。

功能

TD200 具有下列功能：

- 文本信息的显示：
- 用选择项确认方法可显示最多80条信息，每条信息最多可包含4个变量。五种系统语言
- 可设定实时时钟
- 提供强制I/O点诊断功能
- 提供密码保护功能
- 过程参数的显示和修改，参数在显示器中显示并可用输入键进行修改，例如，进行温度设定或速度改变。
- 可编程的8个功能键可以替代普通的控制按钮，作为控制键。这样还可以节省8个输入点。
- 可选择通讯的速率。
- 输入和输出的设定：
  8个可编程功能键的每一个都分配了一个存储器位。例如：这些功能键可在系统启动，测试时进行设置和诊断。又例如：可以不用其它的操作设备即可实现对电动机的控制。
- 可选择显示信息刷新时间。

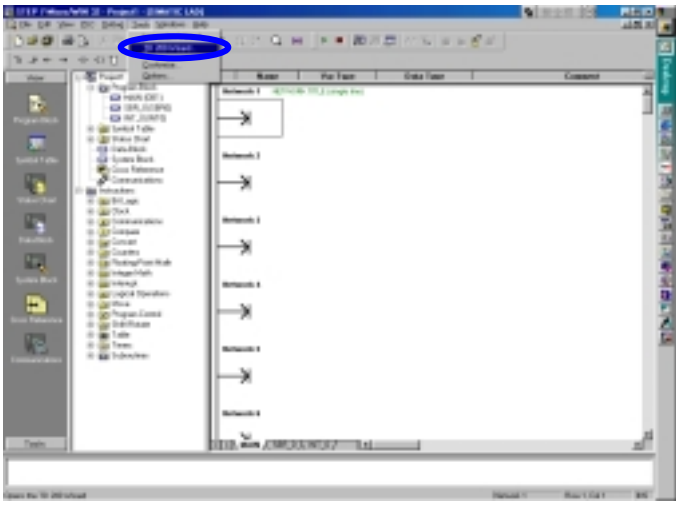
编程

TD200 用 STEP 7-Micro/WIN 软件进行编程。无需其它的参数赋值软件。在 S7-200 系列的 CPU 中保留了一个专用区域用于与 TD200 交换数据。TD200 直接通过这些数据区访问 CPU 的必要功能。

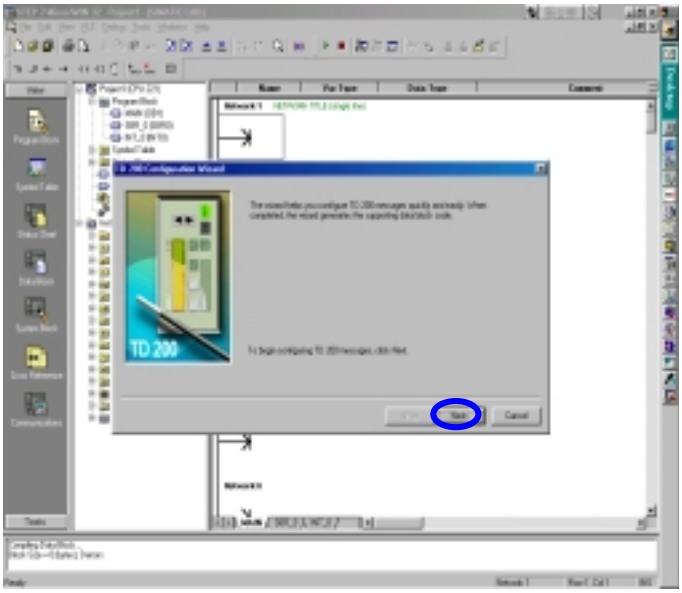
中文TD 200使用说明

支持多种亚洲文字的TD200目前已在国内广泛使用，该面板的组态和使用非常方便。为使广大用户更加容易地使用TD200，现将其组态步骤以图形方式介绍如下。

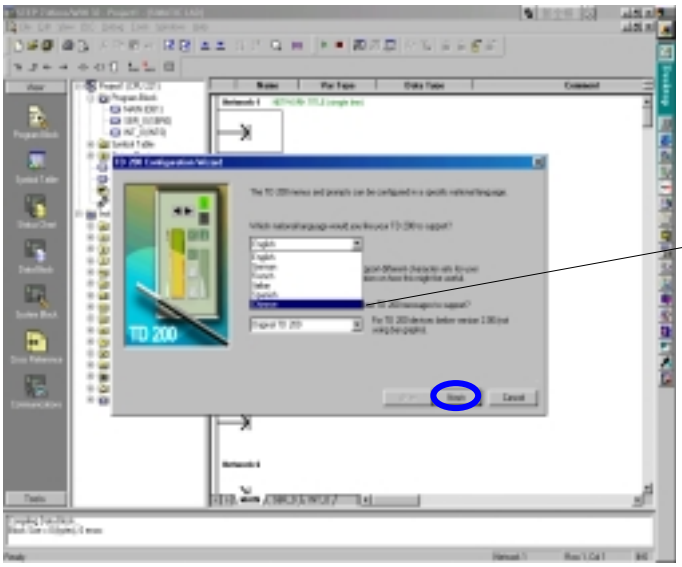
步骤一：在Tools菜单中选中TD 200 Wizard



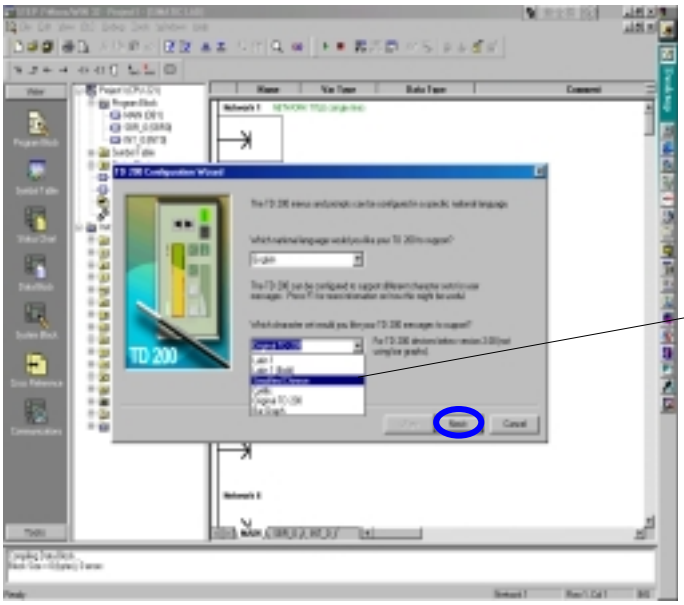
步骤二：点击Next键



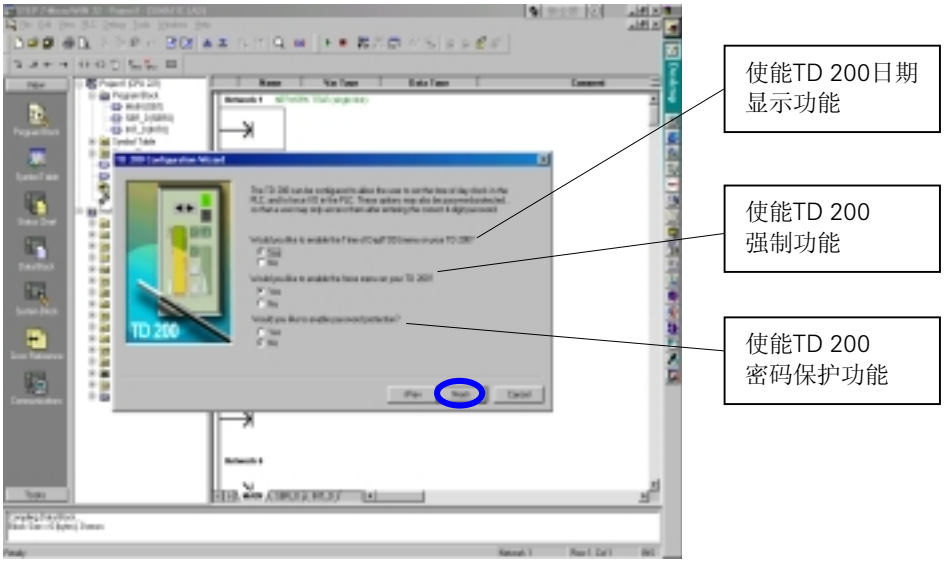
步骤三：选择Chinese项,然后点击Next键



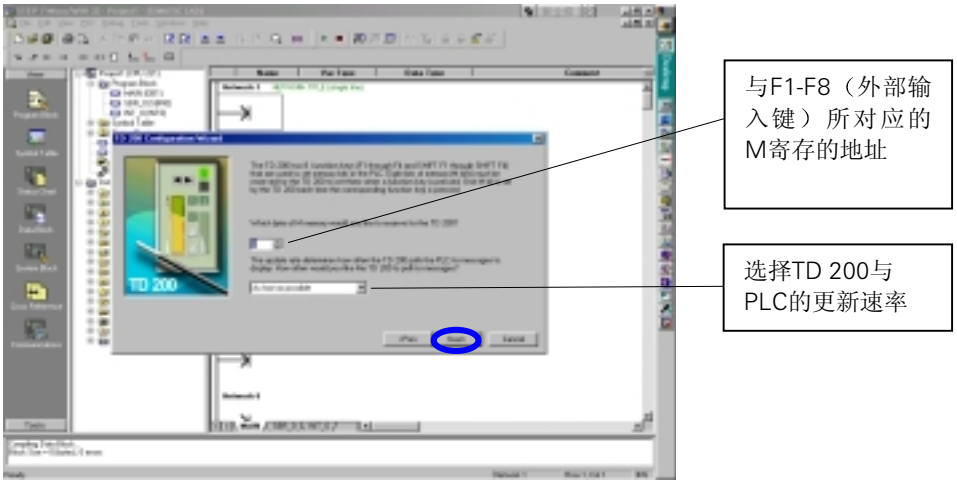
步骤四：选择Simplified Chinese项，然后点击Next键



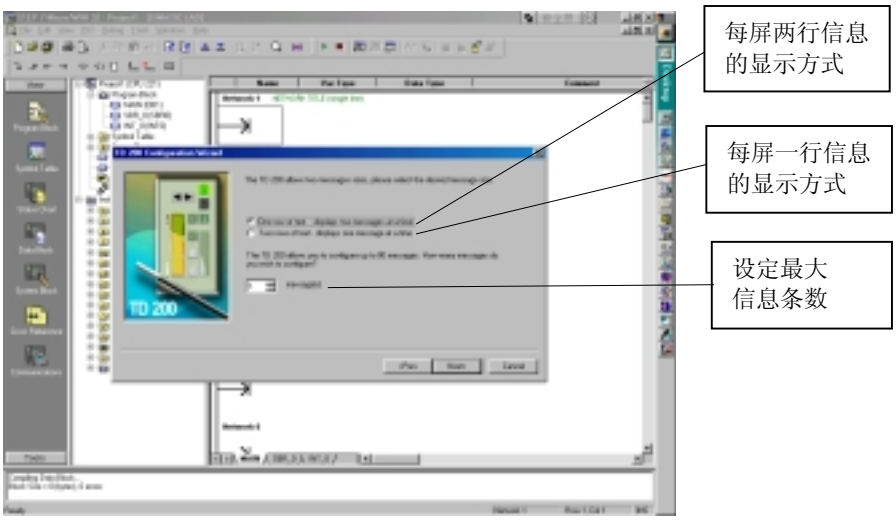
步骤五：接受默认选项，然后点击Next键



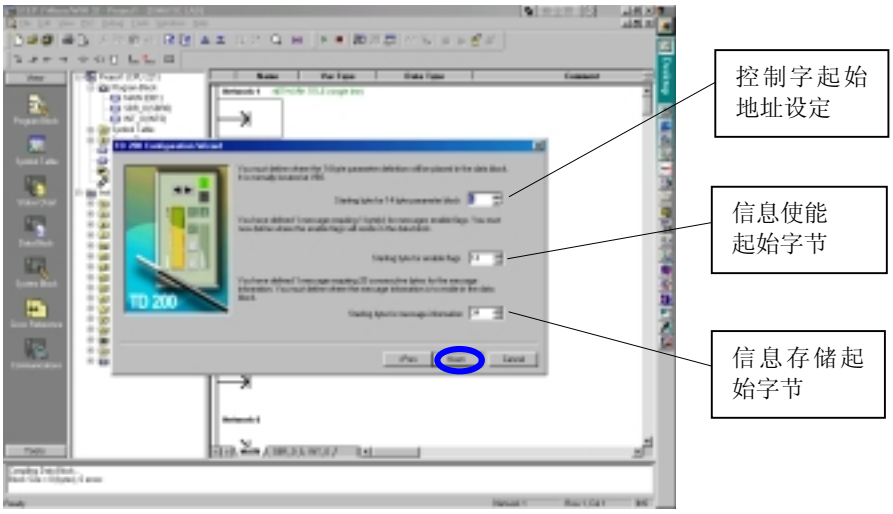
步骤六：接受默认选项，然后点击Next键



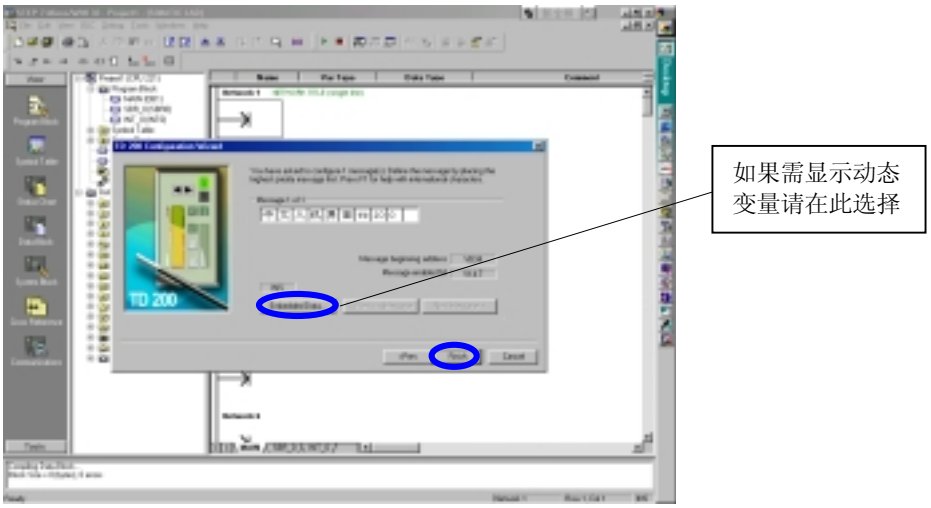
步骤7：接受默认选项，然后点击Next键



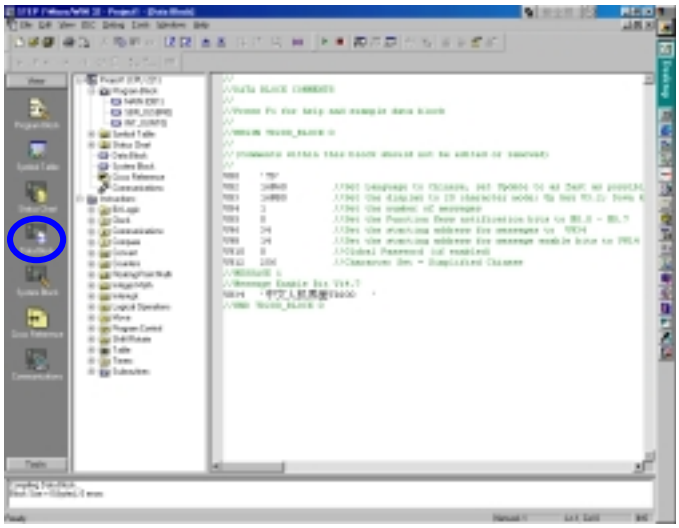
步骤8：接受默认选项，然后点击Next键



步骤九：键入所需的中文文字，然后点击Finish键



步骤十：TD200中的信息可以在DB块中看到



DB块中的控制字说明：

VB0	'TD'	//TD200数据块的标志
VB2	16#60	//设置为中文，更新速率尽可能快
VB3	16#B0	//设置20个字符模式;上箭头对应的位是V3.2；下箭头对应的位是V3.3 ;
VB4	1	//设置显示信息的条数
VB5	0	//M0.0~M0.7对应TD200面板上F1~F4和SHIFT＋F1~SHIFT+F4
VW6	34	//设置信息起始地址为 VW34
VW8	14	//设置信息使能位的地址为VW14
VW10	0	//用于设置全局密码保护（此处为无密码保护）
VW12	256	//设置字符集为简体中文

附录 A：S7-200 技术规范

A

本章概述

A.1 通用技术规范

国家和国际标准

下面列出了用于确定性能规格和测试 S7-200 系列产品的国家和国际标准。表 A -1定义了这些标准遵守的规范。

- Underwriters Laboratories, Inc.: UL 508 Listed (工业控制设备)
- 加拿大标准协会： CSA C22.2 142 许可号 (过程控制设备)
- 工厂互助研究会：FM级I，区2，组A,B,C,&D 危险场所，T4A
- VDE 0160: 电子设备的电源安装
- 欧共体 (CE) 低压指导 72/23/EEC EN 61131 2: 可编程控制器设备要求
- 欧共体 (CE) EMC 指导89/336/EEC
  电磁辐射标准： EN50081-1：民用、商用和轻工业
  EN50081-2：工业环境
  电磁防护标准： EN 50082-2：工业环境

技术规范

S7–200 CPU 系列和所有的 S7 200 扩展模块遵守表 A-1 中所列的技术规范。

表 A-1　　S7–200 系列的技术规范

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	100 mm， 4 次倒下，未包装
IEC 68-2-32，自由落下	1 m，5 次，运输包装
环境条件 —— 工作	
控制柜温度范围 （单元下部 25 mm 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	15 G，11 ms 脉冲，每轴向 (3轴) 震动 6 次
IEC 68-2-6 正弦波振动	峰-峰值 0.30 mm，频率 10 ~ 57 Hz；2 G / 面板安装，1 G / 导轨安装， 57 Hz ~ 150 Hz；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[1] 按照 EN50082-2[1]	
EN 61000-4-2 (IEC 801-2) 静电放电	对所有的面和通讯接口 8 kV 空气放电
EN 50140 (IEC 801-3) 辐射电磁场	80 MHz ~ 1 GHz 10 V/m，用 1 kHz 信号 80% 调制
EN 50141 传导干扰	0.15 ~ 80 MHz 10 V RMS 1kHz下80% 调幅
EN 50204 数字电话防护	900 MHz ± 5 MHz，10 V/m，50% 作用周期, 200 Hz 重复频率
EN 61000-4-4 (IEC 801-4) 瞬间冲击	对 AC 和 DC 电源系统的连接网络，2 kV, 5 kHz； 对数字量 I/O 和通讯口的连接端子，2 kV, 5 kHz
EN 61000-4-5 (IEC 801-5) 浪涌防护	2 kV 非对称， 1 kV 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 390 V，1.3 ms脉冲 对180 VAC线，90° 相角，允许峰值 750 V，1.3 ms脉冲

表 A-1　　S7–200 系列的技术规范

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 0.5 MHz ~ 5 MHz 5 MHz ~ 30 MHz	 < 79 dB (µV) 准峰值；< 66 dB (µV) 平均值 < 73 dB (µV) 准峰值；< 60 dB (µV) 平均值 < 73 dB (µV) 准峰值；< 60 dB (µV) 平均值
EN 55011, Class A, Group 1, 辐射[1] 30 MHz ~ 230 kHz 230 MHz ~ 1 GHz	 30 dB (µV/m) 准峰值；30 米测量 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 0.5 MHz ~ 5 MHz 5 MHz ~ 30 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) < 56 dB (µV) 准峰值< 46 dB (µV) 平均值 < 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 230 MHz ~ 1 GHz	 30 dB (µV/m) 准峰值；10 米测量 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 115/230 V 电路对地 115/230 V 电路对 115/230 V 电路 230 V 电路对 24 V/5 V 电路 115 V 电路对 24 V/5 V 电路	500 VAC (光电隔离限制) 1,500 VAC 1,500 VAC 1,500 VAC 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 SIEMENS 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 250cm。24VDC 供电线和传感器供电线必须屏蔽。

继电器电气寿命

图 A–1 是继电器厂商提供的典型性能数据。实际的性能可能根据特定的应用有所变化。

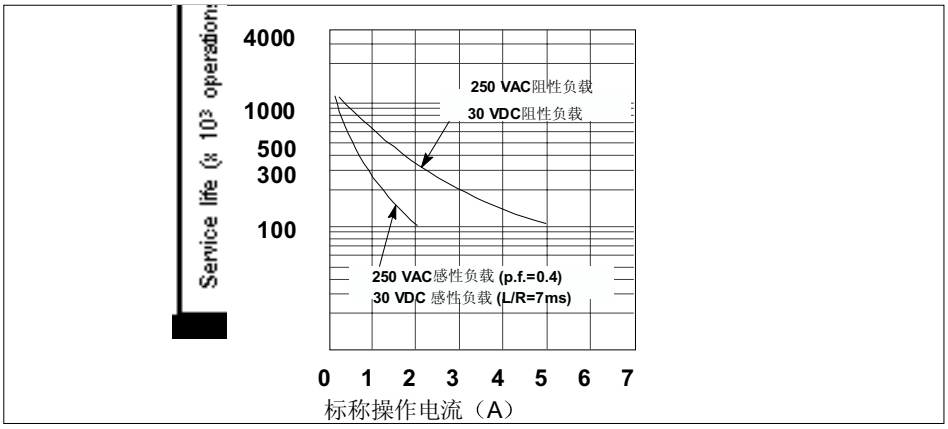


图 A-1　　继电器电气寿命

A.2 CPU 221 性能参数

表 A-2 CPU 221 DC/DC/DC 和 CPU 221 AC/DC/ 继电器的规范

型 号 定货号	CPU 221 DC/DC/DC 6ES7 211-0AA20-0XBO	CPU 221 AC/DC/继电器 6ES7 211-0BA20-0XB0
总体特性		
外形尺寸 (长 x宽H x 高) 重量 功耗	90 mm x 80 mm x 62 mm 270 g 4 W	90 mm x 80 mm x 62 mm 310 g 6 W
CPU 特性		
本机数字输入 本机数字输出	6 输入 4 输出	6 输入 4 输出
高速计数器 (32–位值) 总数 单相计数器个数 两相计数器个数 脉冲输出 模拟电位器 时间中断 边沿中断 可选择的输入滤波器时间 脉冲捕捉	 4 个高速计数器 4 个都是 20 kHz 时钟速率 2 个都是 20 kHz 时钟速率 2 个，20 kHz 脉冲速率 1 个，8 位分辨率 2 个，1 ms 分辨率 4个上升沿和/或 4 下降沿 7 个，范围 0.2 ms ~ 12.8 ms 6 个脉冲捕捉输入	 4 个高速计数器 4 个都是 20 kHz 时钟速率 2 个都是 20 kHz 时钟速率 2 个，20 kHz 脉冲速率 1 个，8 位分辨率 2 个 ，1 ms 分辨率 4 个上升沿和/或 4 下降沿 7 个，范围 0.2 ms ~ 12.8 ms 6 个脉冲捕捉输入
程序空间 (永久保存) 数据块空间: 永久保存 由超级电容或电池保存 最大的数字量 I/O	2048 字 1024 字 1024 字 1024 字 10 输入	2048 字 1024 字 1024 字 1024 字 10 输出
内部存储器位 掉电永久保存 由超级电容或电池保存	256 位 112 位 256 位	256 位 112 位 256 位
定时器总数 由超级电容或电池保存 1 ms 10 ms 100 ms	256 定时器 64 定时器 4 定时器 16 定时器 236 定时器	256 定时器 64 定时器 4 定时器 16 定时器 236 定时器
计数器总数 由超级电容或电池保存 布尔量运算执行速度 字传送的执行速度 定时器/计数器执行速度 单精度数学运算执行速度 实数运算执行速度 超级电容的数据保存时间	256 计数器 256 计数器 0.37 μs 每条指令 34 μs 每条指令 50 μs ~ 64 μs 每条指令 46 μs 每条指令 100 μs ~ 400 μs 每条指令 50 小时/典型，8 小时/最小， 40° C	256 计数器 256 计数器 0.37 μs 每条指令 34 μs 每条指令 50 μs ~ 64 μs 每条指令 46 μs 每条指令 100 μs ~ 400 μs 每条指令 50 小时/典型，8小时/最小，40° C
本机通讯 通讯口数 电气接口 隔离 (外部信号到逻辑电路) PPI/MPI 波特率 自由口波特率	 1 口 RS–485 不隔离 9.6, 19.2, 和187.5 k 波特 0.3, 0.6, 1.2, 2.4, 4.8, 9.6, 19.2, 和38.4 k 波特	 1 口 RS–485 不隔离 9.6, 19.2, 和 187.5 k 波特 0.3, 0.6, 1.2, 2.4, 4.8, 9.6, 19.2, 和38.4 k 波特

表 A-2 CPU 221 DC/DC/DC 和 CPU 221 AC/DC/ 继电器的规范

型 号 定货号	CPU 221 DC/DC/DC 6ES7 211-0AA20-0XBO	CPU 221 AC/DC/继电器 6ES7 211-0BA20-0XB0
每网络段最大电缆长度 直到 38.4 k 波特 187.5 k 波特 最大站数 每个网络段 每个网络 最大主站数 PPI 主站模式 (NETR/NETW) MPI 连接	 1200 m 1000 m 32 站 126 站 32 主站 是 共 4 个; 2 保留: 一个给 PG，另一个给 OP	 1200 m 1000 m 32 站 126 站 32 主站 是 共 4 个; 2 保留: 一个 给PG ，另一个给 OP
可选的卡 存储器卡 (永久保存) 电池卡 (数据保存时间) 时钟卡 (时钟精确度)	 程序、数据和组态 200 days, typical 25° C时，2 分钟/月 0~55° C 时，7 分钟/月	 程序、数据和组态 200 天/典型 25° C时，2 分钟/月 0~55° C 时，7 分钟/月
电源		
电源电压允许范围 输入电流 仅 CPU /最大负载 冲击电流 (最大) 隔离 (输入电源到逻辑电路) 保持时间 (从断开电源) 内部应用 (用户不能替换)	20.4~ 28.8 VDC 70/600 mA ，24 VDC 10 A ，28.8 VDC 不隔离 10 ms，24 VDC 2 A，250 V，慢速熔断	85 ~ 264 VAC 47 ~ 63 Hz 25/80 mA ，240 VAC 25/180 mA ，120 VAC 20 A ，264 VAC 1500 VAC 80 ms，240 VAC, 20 ms， 120 VAC 2 A，250 V，慢速熔断
24 VDC 传感器电源输出 电压范围 最大电流 纹波找声躁声 电流限制 隔离 (传感器到逻辑电路)	 15.4 ~ 28.8 VDC 180 mA 和电源相同 600 mA 不隔离	 20.4 ~ 28.8 VDC 180 mA 峰峰值小于1V (最大) 600 mA 不隔离
主机输入数 输入类型	6 输入 汇型/源型 (IEC Type 1 漏型)	6 输入 汇型/源型 (IEC Type 1 漏型)
输入电压 允许的最大连续值 浪涌 标称值 逻辑 1 信号 (最小) 逻辑 0 信号 (最大)	 30 VDC 35 VDC/ 0.5 s 24 VDC/ 4 mA, 标准值 15 VDC/ 2.5 mA, 最小 5 VDC/ 1 mA, 最大	 30 VDC 35 VDC/ 0.5 s 24 VDC/ 4 mA, 标准值 15 VDC/ 2.5 mA, 最小 5 VDC/ 1 mA, 最大
隔离 (现场侧到逻辑电路) 光电隔离 (galvanic) 隔离组	 500 VAC ，1 分钟 4 点/2 点	 500 VAC ，1 分钟 4 点/2 点
输入延时 滤波输入和中断输入	 0.2 ~ 12.8 ms, 用户可选	 0.2 to 12.8 ms, 用户可选
HSC 时钟输入速率 单相 逻辑 1 电平 = 15 ~ 30 VDC 逻辑 1 电平 = 15 ~ 26 VDC 两相 逻辑 1 电平 = 15 ~ 30 VDC 逻辑 1 电平 = 15 ~ 26 VDC	 20 kHz 30 kHz 10 kHz 20 kHz	 20 kHz 30 kHz 10 kHz 20 kHz

表 A-2 CPU 221 DC/DC/DC 和 CPU 221 AC/DC/ 继电器的规范

型 号 定货号	CPU 221 DC/DC/DC 6ES7 211-0AA20-0XBO	CPU 221 AC/DC/继电器 6ES7 211-0BA20-0XB0
输入特性		
连接 2 线接近开关传感器 (Bero) 允许漏电流	 最大 1 mA	 最大 1 mA
电缆长度 不屏蔽 (非 HSC) 屏蔽 屏蔽的 HSC 输入	 300 m 500 m 50 m	 300 m 500 m 50 m
输入同时接通的数目 40 ° C 55 ° C	 6 输入 6 输入	 6 输入 6 输入
输出特性		
主机输出数 输出类型	4 输出 固态 MOSFET (金属场效应管)	4 输出 继电器-干触点
输出电压 允许范围 标称值 最大电流时逻辑 1 信号 带10 KΩ 负载时逻辑 0 信号	 20.4 ~ 28.8 VDC 24 VDC 20 VDC, 最小 0.1 VDC, 最大	 5 ~ 30 VDC 或 5 ~ 250 VAC - - -
输出电流 逻辑 1 信号 输出组数 输出接通个数 (最多) 每组 - 水平安装 (最多) 每组 - 垂直安装 (最多) 每组的最大电流 灯负载 接通电阻 (触点电阻) 每点的漏电流 浪涌电流 过载保护	 0.75 A 1 4 输出 4 输出 4 输出 3.0 A 5.0 W 0.3 Ω 最大 10 µA 8 A，100 ms (最大) 无	 2.00 A 2 4 输出 3 和 1 输出 3 和 1 输出 6.0 A 30 W DC/200 W AC 当开始使用时，最大是 0.002 Ω - 触点闭合时 7 A 无
隔离 光电隔离 隔离电阻 线圈到触点的隔离 触点间的隔离 组	 500 VAC ，1 分钟 - - - 4 点	 - 当是新的时，最小 100 M Ω 1500 VAC，1 分钟 750 VAC， 1 分钟 3 点和1 点
感性负载嵌位 重复的能量吸收 < 0.5 LI^2 x 开关速率 嵌位电压限制	 1 W, 所有通道 正负 48 V	 - -
输出延时 Off 到 On (Q0.0 和 Q0.1) On 到 Off (Q0.0 和 Q0.1) Off 到 On (Q0.2 和 Q0.3) On 到 Off (Q0.2 和 Q0.3)	 2 µs, 最大 10 µs, 最大 15 µs, 最大 100 µs, 最大	 - - - -
开关频率 (脉冲串输出) Q0.0 和 Q0.1	 20 kHz，最大	 1 Hz，最大
继电器 开关延时 机械寿命 (无负载) 带标称负载时触点寿命	 - - -	 10 ms，最大 10,000,000 开/关循环 100,000 开/关循环
电缆长度 不屏蔽 屏蔽	 150 m 500 m	 150 m 500 m

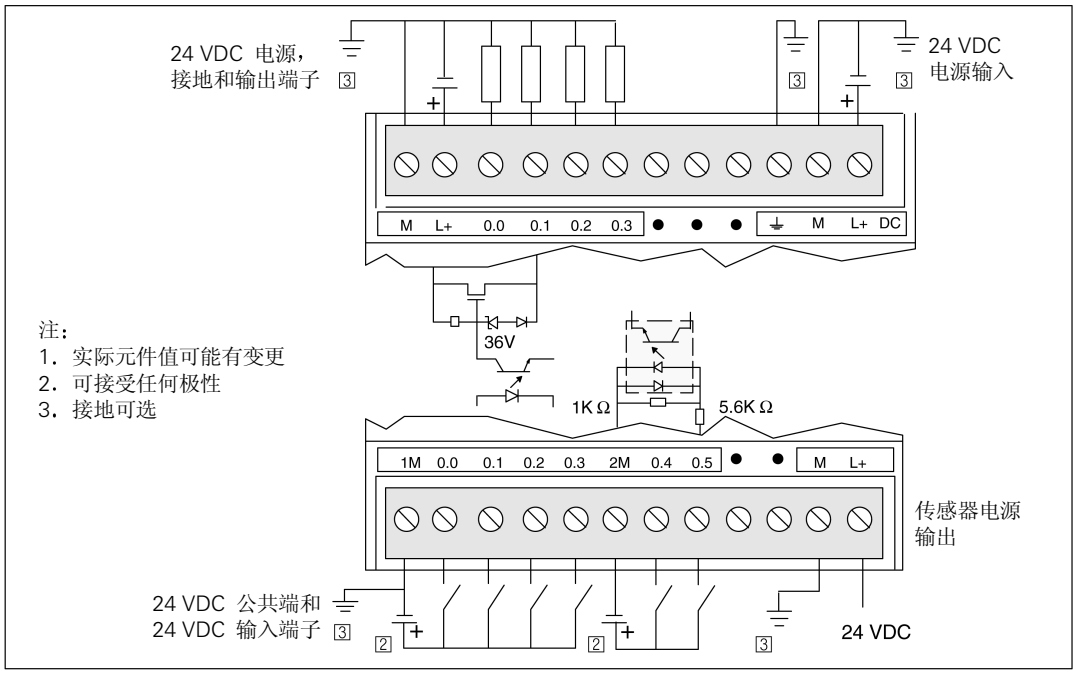


图 A-2　CPU 221 DC/DC/DC 连接器端子图

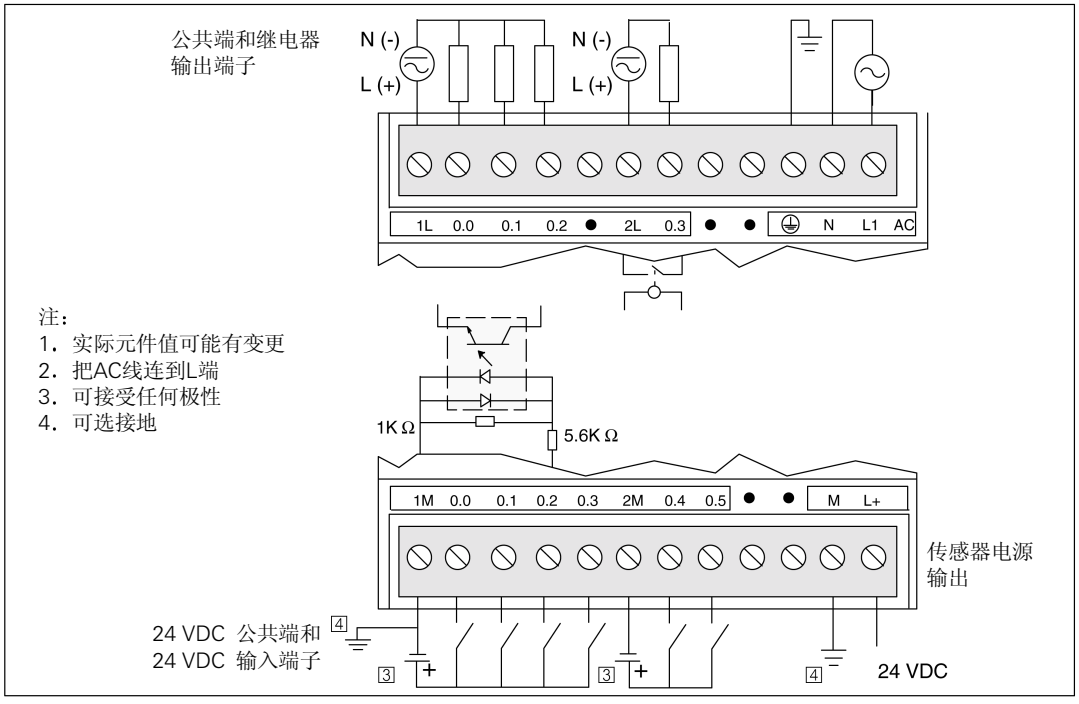


图 A-3　CPU 221 AC/DC/ 继电器连接器端子图

A.3 CPU 222 性能参数

表 A-3 CPU 222 DC/DC/DC 和 CPU 222 AC/DC/ 继电器规范

型 号 定货号	CPU 222 DC/DC/DC 6ES7 212-1AB20-0XB0	CPU 222 AC/DC/继电器 6ES7 212-1BB20-0XB0
总体特性		
外形尺寸 (长x宽 x 高) 重量 功耗	90 mm x 80 mm x 62 mm 270 g 4 W	90 mm x 80 mm x 62 mm 310 g 6 W
CPU 特性		
本机数字输入 本机数字输出	8 输入 6 输出	8 输入 6 输出
高速计数器 (32–位值) 总数 单相计数器个数 两相计数器个数 脉冲输出 模拟电位器 时间中断 边沿中断 可选择的输入滤波器时间 脉冲捕捉	 4 个高速计数器 4 个都是20 kHz 时钟速率 2 个都是20 kHz 时钟速率 2 个，20 kHz 脉冲速率 1 个，8 位分辨率 2 个，1 ms 分辨率 4 个上升沿和/或 4 下降沿 7 个，范围0.2 ms ~ 12.8 ms 8 个脉冲捕捉输入	 4 个高速计数器 4 个都是 20 kHz 时钟速率 2 个都是 20 kHz 时钟速率 2 个，20 kHz 脉冲速率 1个，8 位分辨率 2 个，1 ms 分辨率 4 个上升沿和/或 4 下降沿 7 个，范围 0.2 ms ~ 12.8 ms 8 个脉冲捕捉输入
程序空间 (永久保存) 数据块空间: 永久保存 由超级电容或电池保存 扩展模块的数量 最大的数字量 I/O 最大的模拟量 I/O	2048 字 1024 字 1024 字 1024 字 2 个模块 256 16 AI/16 AO	2048 字 1024 字 1024 字 1024 字 2 个模块 256 16 AI/16 AO
内部存储器位 掉电永久保存 由超级电容或电池保存	256 位 112 位 256 位	256 位 112 位 256 位
定时器总数 由超级电容或电池保存 1 ms 10 ms 100 ms	256 定时器 64 定时器 4 定时器 16 定时器 236 定时器	256 定时器 64 定时器 4 定时器 16 定时器 236 定时器
计数器总数 由超级电容或电池保存 布尔量运算执行速度 字传送的执行速度 定时器/计数器执行速度 单精度数学运算执行速度 实数运算执行速度 超级电容的数据保存时间	256 计数器 256 计数器 0.37 µs 每条指令 34 µs 每条指令 50 µs ~ 64 µs 每条指令 46 µs 每条指令 100 µs ~ 400 µs 每条指令 50 小时/典型，8 小时/最小，40° C	256 计数器 256 计数器 0.37 µs 每条指令 34 µs 每条指令 50 µs ~ 64 µs 每条指令 46 µs 每条指令 100 µs ~ 400 µs 每条指令 50 小时/典型，8 小时/最小，40° C
本机通讯 通讯口数 电气接口 隔离 (外部信号到逻辑电路) PPI/MPI 波特率 自由口波特率	 1 口 RS–485 不隔离 9.6, 19.2, 和 187.5 k 波特 0.3, 0.6, 1.2, 2.4, 4.8, 9.6, 19.2, 和38.4 k 波特	 1 口 RS–485 不隔离 9.6, 19.2, 和 187.5 k 波特 0.3, 0.6, 1.2, 2.4, 4.8, 9.6, 19.2, 和38.4 k 波特

表 A-3 CPU 222 DC/DC/DC 和 CPU 222 AC/DC/ 继电器规范

型 号 定货号	CPU 222 DC/DC/DC 6ES7 212-1AB20-0XB0	CPU 222 AC/DC/继电器 6ES7 212-1BB20-0XB0
每网络段最大电缆长度 直到 38.4 k 波特 187.5 k 波特 最大站数 每个网络段 / 每个网络 最大主站数 PPI 主站模式 (NETR/NETW) MPI 连接	 1200 m 1000 m 32 站 / 126 站 32 主站 是 共 4个； 2 保留： 一个 给 PG ，另一个给 OP	 1200 m 1000 m 32 站 / 126 站 32 主站 是 共 4 个； 2 保留： 一个 给 PG ，另一个给 OP
可选的卡 存储器卡 (永久保存) 电池卡 (数据保存时间) 时钟卡 (时钟精确度)	 程序、数据和组态 200 天/典型 25° C时，2 分钟/月 0~55° C 时，7 分钟/月	 程序、数据和组态 200 天/典型 25° C时，2 分钟/月 0~55° C 时，7 分钟/月
电源		
电源电压允许范围 输入电流 仅 CPU / 最大负载 冲击电流 (最大) 隔离 (输入电源到逻辑电路) 保持时间 (从断开电源) 内部应用 (用户不能替换)	20.4~ 28.8 VDC 70/600 mA ，24 VDC 10 A ，28.8 VDC 不隔离 10 ms，24 VDC 2 A，250 V，慢速熔断	85 ~ 264 VAC 47 ~ 63 Hz 25/80 mA ，240 VAC 25/180 mA ，120 VAC 20 A ，264 VAC 1500 VAC 80 ms ，240 VAC，20 ms，120 VAC 2 A，250 V，慢速熔断
+5 V 扩展I/O模块电源 (最大)	340 mA	340 mA
24 VDC 传感器电源输出 电压范围 最大电流 纹波找声躁声 电流限制 隔离 (传感器到逻辑电路)	 15.4 ~ 28.8 VDC 180 mA 和电源相同 600 mA 不隔离	 20.4 ~ 28.8 VDC 180 mA 峰峰值小于 1V (最大) 600 mA 不隔离
主机输入数 输入类型	8 输入 汇型/源型 (IEC Type 1 漏型)	8 输入 汇型/源型 (IEC Type 1 漏型)
输入电压 允许的最大连续值 浪涌 标称值 逻辑 1 信号 (最小) 逻辑 0 信号 (最大)	 30 VDC 35 VDC/ 0.5 s 24 VDC/ 4 mA, 标准值 15 VDC/ 2.5 mA, 最小 5 VDC/ 1 mA, 最大	 30 VDC 35 VDC/ 0.5 s 24 VDC/ 4 mA, 标准值 15 VDC/ 2.5 mA, 最小 5 VDC/ 1 mA, 最大
隔离 (现场侧到逻辑电路) 光电隔离 隔离组	 500 VAC ，1 分钟 4 输入	 500 VAC ，1 分钟 4 输入
输入延时 滤波输入和中断输入	 0.2 ~ 12.8 ms, 用户可选	 0.2 ~ 12.8 ms, 用户可选
HSC 时钟输入速率 单相 逻辑 1 电平 = 15 ~ 30 VDC 逻辑 1 电平 = 15 ~ 26 VDC 两相 逻辑 1 电平 = 15 ~ 30 VDC 逻辑 1 电平 = 15 ~ 26 VDC	 20 kHz 30 kHz 10 kHz 20 kHz	 20 kHz 30 kHz 10 kHz 20 kHz

表 A-3 CPU 222 DC/DC/DC 和 CPU 222 AC/DC/ 继电器规范

型 号 定货号	CPU 222 DC/DC/DC 6ES7 212-1AB20-0XB0	CPU 222 AC/DC/继电器 6ES7 212-1BB20-0XB0
输入特性		
连接 2 线接近开关传感器 (Bero) 允许漏电流	 最大1 mA	 最大1 mA
电缆长度 不屏蔽 (不是 HSC) 屏蔽 屏蔽的 HSC 输入	 300 m 500 m 50 m	 300 m 500 m 50 m
输入同时接通的数目 40 ° C 55 ° C	 8 输入 8 输入	 8 输入 8 输入
输出特性		
主机输出数 输出类型	6 输出 固态-MOSFET	6 输出 继电器-干触点
输出电压 允许范围 标称值 最大电流时逻辑 1 信号 带 10 K□ 负载时逻辑 0 信号	 20.4 ~ 28.8 VDC 24 VDC 20 VDC, 最小 0.1 VDC, 最大	 5 ~ 30 VDC 或 5 ~ 250 VAC - - -
输出电流 逻辑 1 信号 输出组数 输出接通个数 (最多) 每组 - 水平安装 (最多) 每组 - 垂直安装 (最多) 每组的最大电流 灯负载 接通电阻 (触点电阻) 每点的漏电流 浪涌电流 过载保护	 0.75 A 1 6 输出 6 输出 6 输出 4.5 A 5 W 0.3 Ω 最大 10 µA 8 A ，100 ms (最大) 无	 2.00 A 2 6 输出 3 输出 3 输出 6 A 30 W DC/ 200 W AC 当是新的时，最大 0.002 Ω - 触点闭合时 7 A 无
隔离 光电隔离 隔离电阻 线圈到触点的隔离 触点间的隔离 组	 500 VAC ，1 分钟 - - - 6 点	 - 当是新的时，最小 100 MΩ 1500 VAC，1 分钟 750 VAC， 1 分钟 3 点
感性负载嵌位 重复的能量吸收< 0.5 LI^2 x 开关速率 嵌位电压限制	 1 W, 所有通道 正负 48 V	 - -
输出延时 Off 到 On (Q0.0 和 Q0.1) On 到 Off (Q0.0 和 Q0.1) Off 到 On (Q0.2 和 Q0.5) On 到 Off (Q0.2 和 Q0.5)	 2 µs, 最大 10 µs, 最大 15 µs, 最大 100 µs, 最大	 - - - -
开关频率 (脉冲串输出) Q0.0 and Q0.1	 20 kHz，最大	 1 Hz，最大
继电器 开关延时 机械寿命 (无负载) 带标称负载时触点寿命	 - - -	 10 ms，最大 10,000,000 开/关循环 100,000 开/关循环
电缆长度 不屏蔽 屏蔽	 150 m 500 m	 150 m 500 m

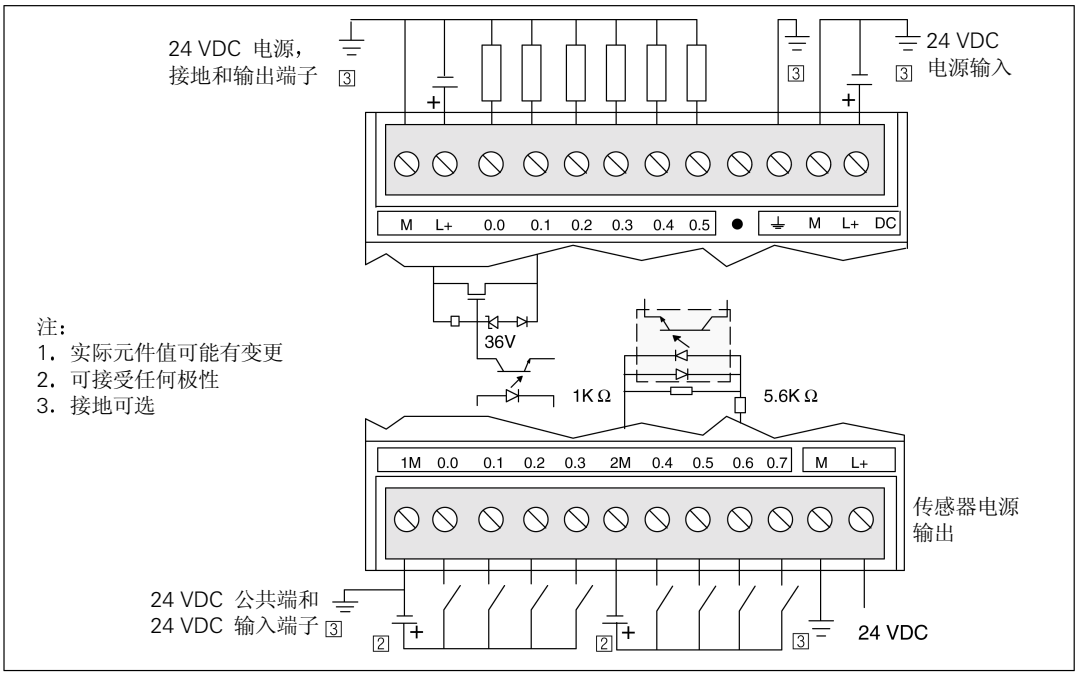


图 A-4　CPU 222 DC/DC/DC 接器端子图

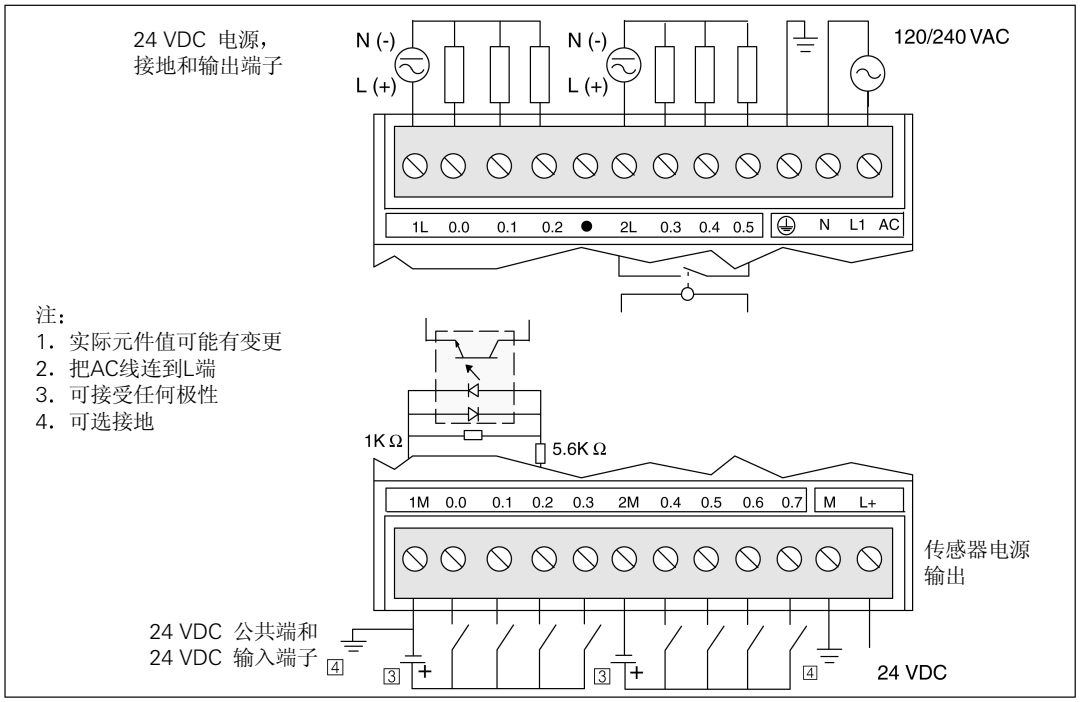


图 A-5　CPU 222 AC/DC/ 继电器连接器端子图

A.4 CPU 224 性能参数

表 A-4 CPU 224 DC/DC/DC 和 CPU 224 AC/DC/ 继电器规范

型 号 定货号	CPU 224 DC/DC/DC 6ES7 214-1AD20-0XB0	CPU 224 AC/DC/继电器 6ES7 214-1BD20-0XB0
总体特性		
外形尺寸 (长 x宽H x 高) 重量 功耗	120.5 mm x 80 mm x 62 mm 360 g 8 W	120.5 mm x 80 mm x 62 mm 410 g 9 W
CPU 特性		
本机数字输入 本机数字输出	14 路数字量输入 10 路数字量输出	14 路数字量输入 10 路数字量输出
高速计数器 (32–位值) 总数 单相计数器个数 两相计数器个数 脉冲输出 模拟电位器 时间中断 边沿中断 可选择的输入滤波器时间 脉冲捕捉 时钟 (时钟精度)	 6 个高速计数器 6 个都是20 kHz 时钟速率 4 个都是20 kHz 时钟速率 2 个，20 kHz 脉冲速率 1 个，8 位分辨率 2 个，1 ms 分辨率 4 个上升沿和/或 4 下降沿 7 个，范围 0.2 ms ~ 12.8 ms 14 个脉冲捕捉输入 25° C 时，2 分种/月 0° C ~ 55° C 时，7 分种/月	 6 个高速计数器 6 个都是 20 kHz 时钟速率 4 个都是 20 kHz 时钟速率 2 个，20 kHz 脉冲速率 1 个，8 位分辨率 2 个，1 ms 分辨率 4 个上升沿和/或 4 下降沿 7 个，范围 0.2 ms ~ 12.8 ms 14 个脉冲捕捉输入 25° C 时，2 分种/月 0° C ~ 55° C时，7 分种/月
程序空间 (永久保存) 数据块空间: (永久保存) 永久保存 由超级电容或电池保存 扩展模块的数量 最大的数字量 I/O 最大的模拟量 I/O	4096 字 2560 字 2560 字 2560 字 7 个模块 256 16 AI/16 AO	4096 字 2560 字 2560 字 2560 字 7 个模块 256 16 AI/16 AO
内部存储器位 掉电永久保存 由超级电容或电池保存	256 位 112 位 256 位	256 位 112 位 256 位
定时器总数 由超级电容或电池保存 1 ms 10 ms 100 ms	256 定时器 64 定时器 4 定时器 16 定时器 236 定时器	256 定时器 64 定时器 4 定时器 16 定时器 236 定时器
计数器总数 由超级电容或电池保存 布尔量执行速度 传送字的执行速度 定时器/计数器执行速度 单精度数学运算执行速度 实数运算执行速度 超级电容的数据保存时间	256 计数器 256 计数器 0.37 µs 每条指令 34 µs 每条指令 50 µs ~ 64 µs 每条指令 46 µs 每条指令 100 µs ~ 400 µs 每条指令 190 小时/典型，120 小时/最小， 40° C	256 计数器 256 计数器 0.37 µs 每条指令 34 µs 每条指令 50 µs ~ 64 µs 每条指令 46 µs 每条指令 100 µs ~ 400 µs 每条指令 190 小时/典型，120 小时/最小， 40° C
本机通讯 通讯口数 电气接口 隔离 (外部信号到逻辑电路) PPI/MPI 波特率 自由口波特率	 1 口 RS–485 不隔离 9.6, 19.2, 和 187.5 k 波特 0.3, 0.6, 1.2, 2.4, 4.8, 9.6, 19.2, 和38.4 k 波特	 1 口 RS–485 不隔离 9.6, 19.2, 和187.5 k 波特 0.3, 0.6, 1.2, 2.4, 4.8, 9.6, 19.2, 和 38.4 k 波特

表 A-4 CPU 224 DC/DC/DC 和 CPU 224 AC/DC/ 继电器规范

型 号 定货号	CPU 224 DC/DC/DC 6ES7 214-1AD20-0XB0	CPU 224 AC/DC/继电器 6ES7 214-1BD20-0XB0
每网络段最大电缆长度 直到 38.4 k 波特 187.5 k 波特 最大站数 每个网络段 / 每个网络 最大主站数 PPI 主站模式 (NETR/NETW) MPI 连接	 1200 m 1000 m 32 站 / 126 站 32 主站 是 共 4 个；2 保留：一个给 PG，另一个给 OP	 1200 m 1000 m 32 站 / 126 站 32 主站 是 共 4 个；2 保留：一个给 PG，另一个给 OP
可选的卡 存储器卡 (永久保存) 电池卡 (数据保存时间)	 程序、数据和组态 200 天/典型	 程序、数据和组态 200 天/典型
电源		
电源电压允许范围 输入电流 仅 CPU / 最大负载 冲击电流 (最大) 隔离 (输入电源到逻辑电路) 保持时间 (从断开电源) 内部应用 (用户不能替换)	20.4 ~ 28.8 VDC 120/900 mA，24 VDC 10 A，28.8 VDC 不隔离 10 ms，24 VDC 2 A, 250 V, 慢速熔断	85 ~ 264 VAC 47 ~ 63 Hz 35/100 mA，240 VAC 35/220 mA，120 VAC 20 A，264 VAC 1500 VAC 80 ms，240 VAC, 20 ms，120 VAC 2 A, 250 V, 慢速熔断
+5 V 扩展I/O模块电源 (最大)	660 mA	660 mA
24 VDC **传感器电源输出** 电压范围 最大电流 纹波找声躁声 电流限制 隔离 (传感器到逻辑电路)	 15.4 ~ 28.8 VDC 180 mA 和电源相同 600 mA 不隔离	 20.4 ~ 28.8 VDC 180 mA 峰峰值小于 1V (最大) 600 mA 不隔离
主机输入数 输入类型	14 输入 汇型/源型 (IEC Type 1)	14 输入 汇型/源型 (IEC Type 1)
输入电压 允许的最大连续值 浪涌 标称值 逻辑 1 信号 (最小) 逻辑 0 信号 (最大)	 30 VDC 35 VDC/ 0.5 s 24 VDC/ 4 mA, 标准值 15 VDC/ 2.5 mA, 最小 5 VDC/ 1 mA, 最大	 30 VDC 35 VDC/ 0.5 s 24 VDC/ 4 mA, 标准值 15 VDC/ 2.5 mA, 最小 5 VDC/ 1 mA, 最大
隔离 (现场侧到逻辑电路) 光电隔离 隔离组	 500 VAC，1 分钟 8 点和 6 点	 500 VAC，1 分钟 8 点和 6 点
输入延时 滤波输入和中断输入	 0.2 ~ 12.8 ms, 用户可选	 0.2 to 12.8 ms, 用户可选
HSC 时钟输入速率 单相 逻辑 1 电平 = 15 ~ 30 VDC 逻辑 1 电平 = 15 ~ 26 VDC 两相 逻辑 1 电平 = 15 ~ 30 VDC 逻辑 1 电平 = 15 ~ 26 VDC	 20 kHz 30 kHz 10 kHz 20 kHz	 20 kHz 30 kHz 10 kHz 20 kHz
连接线接近开关传感器 (Bero) 允许漏电流	 最大 1 mA	 最大 1 mA

表 A-4 CPU 224 DC/DC/DC 和 CPU 224 AC/DC/ 继电器规范

型 号 定货号	CPU 224 DC/DC/DC 6ES7 214-1AD20-0XB0	CPU 224 AC/DC/继电器 6ES7 214-1BD20-0XB0
输入特性		
电缆长度		
不屏蔽 (不是 HSC)	300 m	300 m
屏蔽	500 m	500 m
屏蔽的 HSC 输入	50 m	50 m
输入同时接通的数目		
40 ° C	14 输入	14 输入
55 ° C	14 输入	14 输入
输出特性		
主机输出数	10 输出	10 输出
输出类型	固态 -MOSFET	继电器-干触点
输出电压		
允许范围	20.4 ~ 28.8 VDC	5 ~ 30 VDC 或 5 ~ 250 VAC
标称值	24 VDC	-
最大电流时逻辑 1 信号	20 VDC, 最小	-
带 10 KΩ 负载时逻辑 0 信号	0.1 VDC, 最大	-
输出电流		
逻辑 1 信号	0.75 A	2.00 A
输出组数	2	3
输出接通个数 (最多)	10	10
每组 - 水平安装 (最多)	5	4/3/3
每组 - 垂直安装 (最多)	5	4/3/3
每组的最大电流	3.75 A	8 A
灯负载	5 W	30 W DC/200 W AC
接通电阻 (触点电阻)	0.3 Ω	当开始使用时，最大 0.002 Ω
每点的漏电流	最大 10 µA	-
浪涌电流	8 A ，100 ms (最大)	触点闭合时 7 A
过载保护	无	无
隔离		
光电隔离	500 VAC ，1 分钟	-
隔离电阻	-	当开始使用时，最小 100 MΩ
线圈到触点的隔离	-	1500 VAC，1 分钟
触点间的隔离	-	750 VAC， 1 分钟
组	5 点	4 点 / 3 点 / 3 点
感性负载嵌位		
重复的能量吸收 < 0.5 LI^2 x 开关速率	1 W, 所有通道	-
嵌位电压限制	正负 48 V	-
输出延时		
Off 到 On (Q0.0 和 Q0.1)	2 µs, 最大	-
On 到 Off (Q0.0 和 Q0.1)	10 µs, 最大	-
Off 到 On (Q0.2 和 Q1.1)	15 µs, 最大	-
On 到 Off (Q0.2 和 Q1.1)	100 µs, 最大	-
开关频率 (脉冲串输出)		
Q0.0 和 Q0.1	20 kHz，最大	1 Hz，最大
继电器		
开关延时	-	10 ms，最大
机械寿命 (无负载)	-	10,000,000 开/关 循环
带标称负载时触点寿命	-	100,000 开/关 循环
电缆长度		
不屏蔽	150 m	150 m
屏蔽	500 m	500 m

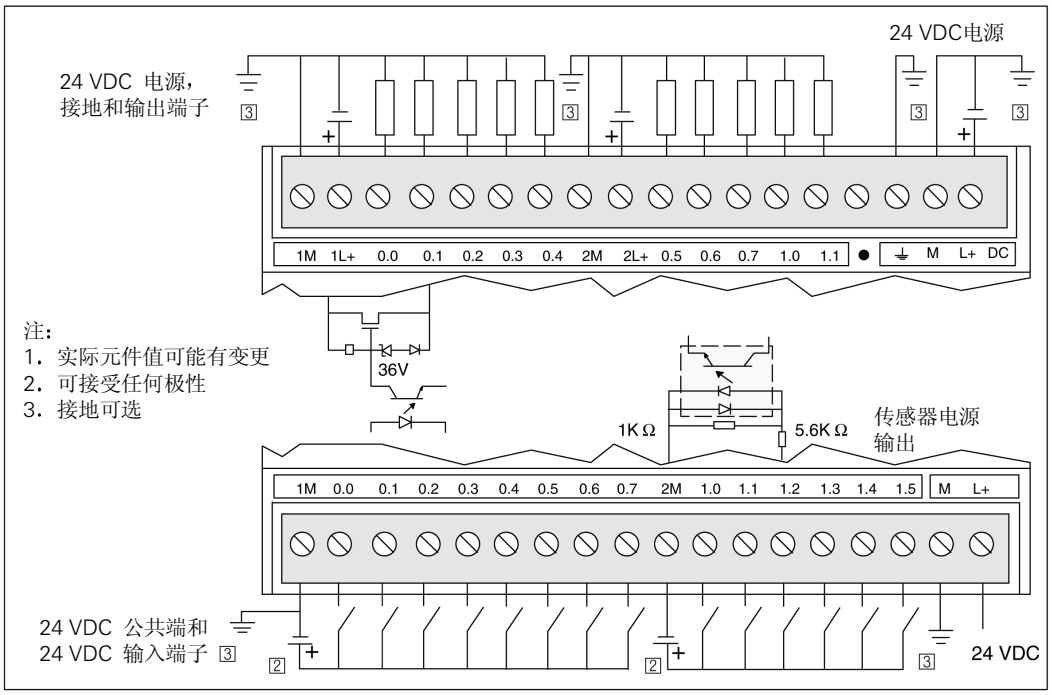


图 A-6　CPU 224 DC/DC/DC 连接器端子图

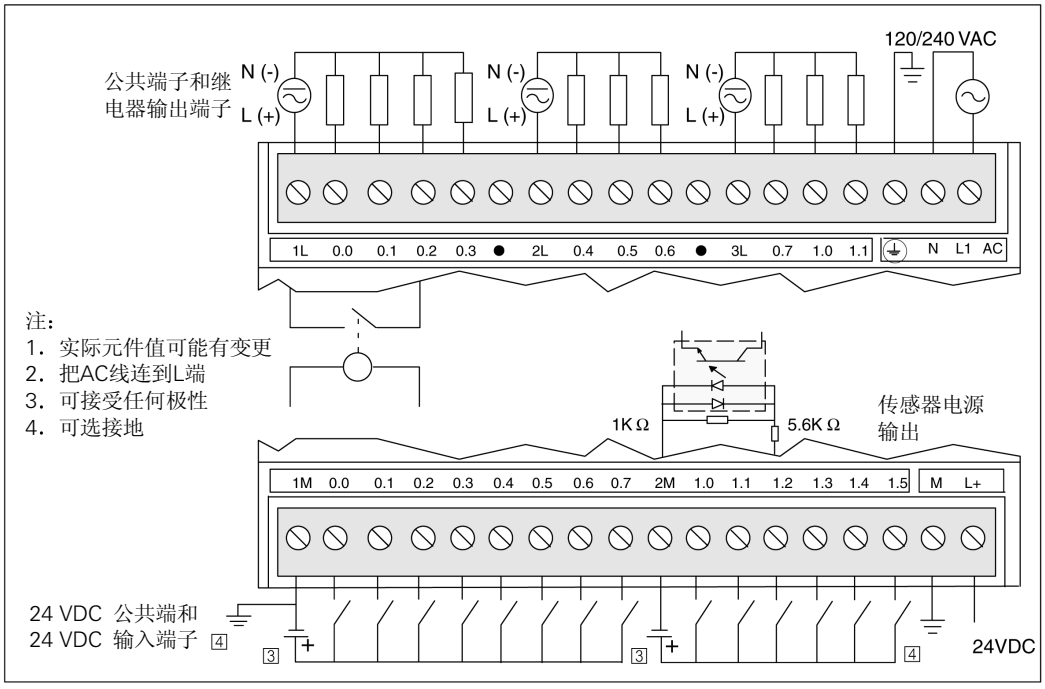


图 A-7　CPU 224 AC/DC/ 继电器连接器端子图

A.5 CPU 226 性能参数

表 A-5 CPU 226 DC/DC/DC 和 CPU 226 AC/DC/ 继电器规范

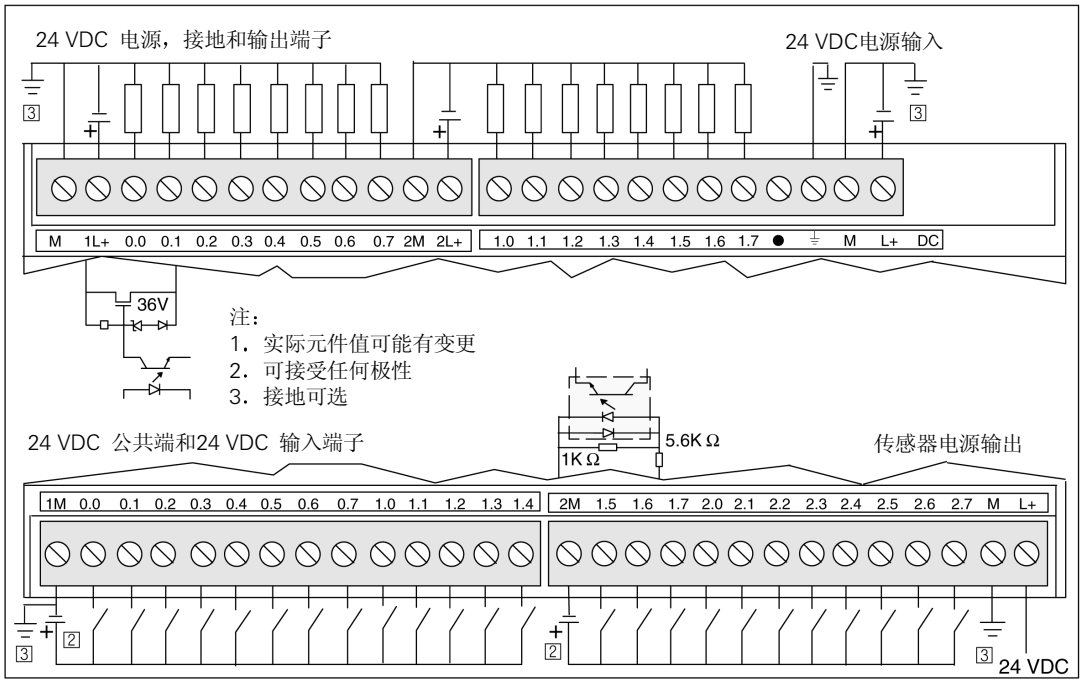
型 号 定货号	CPU 226 DC/DC/DC 6ES7 216-1AD21-0XB0	CPU 226 AC/DC/继电器 6ES7 216-1BD21-0XB0
总体特性		
外形尺寸 (长 x宽H x 高) 重量 功耗	196 mm x 80 mm x 62 mm 550 g 11 W	196 mm x 80 mm x 62 mm 660 g 17 W
CPU 特性		
本机数字输入 本机数字输出	24 路数字量输入 16 路数字量输出	24 路数字量输入 16 路数字量输出
高速计数器 (32–位值) 总数 单相计数器个数 两相计数器个数 脉冲输出 模拟电位器 时间中断 边沿中断 可选择的输入滤波器时间 脉冲捕捉 时钟 (时钟精度)	 6 个高速计数器 6 个都是30 kHz 时钟速率 4 个都是20 kHz 时钟速率 2 个，20 kHz 脉冲速率 2 个，8 位分辨率 2 个，1 ms 分辨率 4 个上升沿和/或 4 下降沿 7 个，范围 0.2 ms ~ 12.8 ms 14 个脉冲捕捉输入 25°C 时，2 分种/月 0°C ~ 55 °C 时，7 分种/月	 6 个高速计数器 6 个都是 30 kHz 时钟速率 4 个都是 20 kHz 时钟速率 2 个，20 kHz 脉冲速率 2 个，8 位分辨率 2 个，1 ms 分辨率 4 个上升沿和/或 4 下降沿 7 个，范围 0.2 ms ~ 12.8 ms 14 个脉冲捕捉输入 25°C 时，2 分种/月 0°C ~ 55°C 时，7 分种/月
程序空间 (永久保存) 数据块空间: (永久保存) 永久保存 由超级电容或电池保存 扩展模块的数量 最大的数字量 I/O 最大的模拟量 I/O	4096 字 2560 字 2560 字 2560 字 7 个模块 256 32 AI和32 AO	4096 字 2560 字 2560 字 2560 字 7 个模块 256 32 AI和32 AO
内部存储器位 掉电永久保存 由超级电容或电池保存	256 位 112 位 256 位	256 位 112 位 256 位
定时器总数 由超级电容或电池保存 1 ms 10 ms 100 ms	256 定时器 64 定时器 4 定时器 16 定时器 236 定时器	256 定时器 64 定时器 4 定时器 16 定时器 236 定时器
计数器总数 由超级电容或电池保存 布尔量执行速度 传送字的执行速度 定时器/计数器执行速度 单精度数学运算执行速度 实数运算执行速度 超级电容的数据保存时间	256 计数器 256 计数器 0.37 µs 每条指令 34 µs 每条指令 50 µs ~ 64 µs 每条指令 46 µs 每条指令 100 µs ~ 400 µs 每条指令 190 小时/典型， 40°C时最小 120 小时	256 计数器 256 计数器 0.37 µs 每条指令 34 µs 每条指令 50 µs ~ 64 µs 每条指令 46 µs 每条指令 100 µs ~ 400 µs 每条指令 190 小时/典型， 40°C时最小 120 小时
本机通讯 通讯口数 电气接口 隔离 (外部信号到逻辑电路) PPI/MPI 波特率 自由口波特率	 2 口 RS–485 不隔离 9.6, 19.2, 和 187.5 k 波特 0.3, 0.6, 1.2, 2.4, 4.8, 9.6, 19.2, 和 38.4 k 波特	 2 口 RS–485 不隔离 9.6, 19.2, 和187.5 k 波特 0.3, 0.6, 1.2, 2.4, 4.8, 9.6, 19.2, 和 38.4 k 波特

表 A-5 CPU 226 DC/DC/DC 和 CPU 226 AC/DC/ 继电器规范

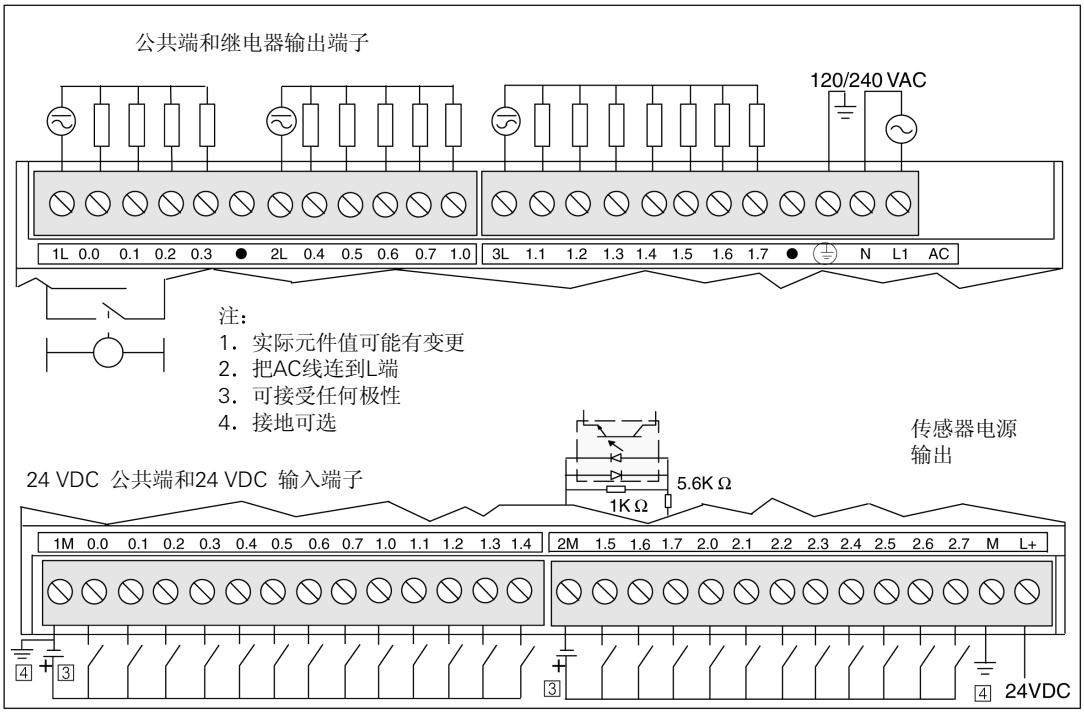
型 号 定货号	CPU 226 DC/DC/DC 6ES7 216-1AD21-0XB0	CPU 226 AC/DC/继电器 6ES7 216-1BD21-0XB0
每网络段最大电缆长度 直到 38.4 k 波特 187.5 k 波特 最大站数 每个网络段 / 每个网络 最大主站数 PPI 主站模式 (NETR/NETW) MPI 连接	 1200 m 1000 m 32 站 / 126 站 32 主站 是 共 4 个; 2 保留：一个给 PG，另一个给 OP	 1200 m 1000 m 32 站 / 126 站 32 主站 是 共 4 个; 2 保留：一个给 PG，另一个给 OP
可选的卡 存储器卡 (永久保存) 电池卡 (数据保存时间)	 程序、数据和组态 200 天/典型	 程序、数据和组态 200 天/典型
电源		
电源电压允许范围 输入电流 仅 CPU / 最大负载 冲击电流 (最大) 隔离 (输入电源到逻辑电路) 保持时间 (从断开电源) 内部熔断 (用户不能替换)	20.4 ~ 28.8 VDC 150/1050 mA，24 VDC 10 A，28.8 VDC 不隔离 10 ms，24 VDC 3 A, 250 V, 慢速熔断	85 ~ 264 VAC 47 ~ 63 Hz 40/160 mA，240 VAC 80/320 mA，120 VAC 20 A，264 VAC 1500 VAC 80ms，240VAC, 20ms，120 VAC 3 A, 250 V, 慢速熔断
+5 V 扩展I/O模块电源 (最大)	1000 mA	1000 mA
24 VDC **传感器电源输出** 电压范围 最大电流 纹波躁声 电流限制 隔离 (传感器到逻辑电路)	 15.4 ~ 28.8 VDC 400 mA 和电源相同 约1.5A 不隔离	 20.4 ~ 28.8 VDC 400 mA 峰峰值小于 1V (最大) 约1.5A 不隔离
输入特性		
主机输入数 输入类型	24 输入 汇型/源型 (IEC Type 1)	24 输入 汇型/源型 (IEC Type 1)
输入电压 允许的最大连续值 浪涌 标称值 逻辑 1 信号 (最小) 逻辑 0 信号 (最大)	 30 VDC 35 VDC/ 0.5 s 24 VDC/ 4 mA, 标准值 15 VDC/ 2.5 mA, 最小 5 VDC/ 1 mA, 最大	 30 VDC 35 VDC/ 0.5 s 24 VDC/ 4 mA, 标准值 15 VDC/ 2.5 mA, 最小 5 VDC/ 1 mA, 最大
隔离 (现场侧到逻辑电路) 光电隔离 隔离组	 500 VAC ，1 分钟 13 点和 11 点	 500 VAC ，1 分钟 13 点和 11 点
输入延时 滤波输入和中断输入	 0.2 ~ 12.8 ms, 用户可选	 0.2 ~ 12.8 ms, 用户可选
HSC 时钟输入速率 单相 逻辑 1 电平 = 15 ~ 30 VDC 逻辑 1 电平 = 15 ~ 26 VDC 两相 逻辑 1 电平 = 15 ~ 30 VDC 逻辑 1 电平 = 15 ~ 26 VDC	 20 kHz 30 kHz 10 kHz 20 kHz	 20 kHz 30 kHz 10 kHz 20 kHz
输入特性		
连接2线接近开关传感器 (Bero) 允许漏电流	 最大 1 mA	 最大 1 mA

表 A-5 CPU 226 DC/DC/DC 和 CPU 226 AC/DC/ 继电器规范

型 号 定货号	CPU 226 DC/DC/DC 6ES7 216-1AD21-0XB0	CPU 226 AC/DC/继电器 6ES7 216-1BD21-0XB0
电缆长度		
不屏蔽 (不是 HSC)	300 m	300 m
屏蔽	500 m	500 m
屏蔽的 HSC 输入	50 m	50 m
输入同时接通的数量		
40 ℃	24 输入	24 输入
55 ℃	24 输入	24 输入
输出特性		
主机输出数	16 输出	16 输出
输出类型	固态 -MOSFET	继电器-干触点
输出电压		
允许范围	20.4 ~ 28.8 VDC	5 ~ 30 VDC 或 5 ~ 250 VAC
标称值	24 VDC	-
最大电流时逻辑 1 信号	20 VDC, 最小	-
带 10 KΩ 负载时逻辑 0 信号	0.1 VDC, 最大	-
输出电流		
逻辑 1 信号	0.75 A	2.00 A
输出组数	2	3
输出接通个数 (最多)	16	16
每组 - 水平安装 (最多)	8	4/5/7
每组 - 垂直安装 (最多)	8	4/5/7
每组的最大电流	6 A	10 A
灯负载	5 W	30 W DC/200 W AC
接通电阻 (触点电阻)	0.3 Ω	当开始使用时，最大 0.2 Ω
每点的漏电流	最大 10 µA	-
浪涌电流	8 A ，100 ms (最大)	触点闭合时 7 A
过载保护	无	无
隔离 (现场侧到逻辑电路)		
光电隔离	500 VAC ，1 分钟	-
隔离电阻	-	当开始使用时，最小 100 MΩ
线圈到触点的隔离	-	1500 VAC，1 分钟
触点间的隔离	-	750 VAC， 1 分钟
组	8 点	4 点 / 5 点 / 7 点
感性负载嵌位		
重复的能量吸收 < 0.5 LI^2 x 开关速率	1 W, 所有通道	-
嵌位电压限制	L+— 48 V	-
输出延时		
Off 到 On (Q0.0 和 Q0.1)	2 µs, 最大	-
On 到 Off (Q0.0 和 Q0.1)	10 µs, 最大	-
Off 到 On (Q0.2 和 Q1.7)	15 µs, 最大	-
On 到 Off (Q0.2 和 Q1.7)	100 µs, 最大	-
开关频率 (脉冲串输出)		
Q0.0 和 Q0.1	20 kHz，最大	1 Hz，最大
继电器		
开关延时	-	10 ms，最大
机械寿命 (无负载)	-	10,000,000 开/关 循环
带标称负载时触点寿命	-	100,000 开/关 循环
电缆长度		
不屏蔽	150 m	150 m
屏蔽	500 m	500 m



图A-8　CPU 226 DC/DC/DC 端子连接图



图A-9　CPU 226 AC/DC/ 继电器端子连接图

A.6 EM221 数字量输入模块性能参数

表 A-6 EM221 24 VDC，8 数字量输入模块技术规范

型 号 定货号	EM221 24 VDC, 8 输入 6ES7 221-1BF20-0XA0
总体特性	
外形尺寸 (长× 宽×高) 重量 功耗	46 x 80 x 62 mm 150 g 2 W
输入特性	
本机输入点数 输入类型 输入电压 允许的最大连续值 浪涌 **标称值** 逻辑 1 信号 (最小) 逻辑 0 信号 (最大) **隔离** 光电隔离 隔离组 **输入延时** 最大 **连接 2 线接近开关传感器 (Bero)** 允许漏电流 **电缆长度** 不屏蔽 屏蔽 **同时接通的输入数** 40 ° C 55 ° C **电能消耗** +5 VDC (从 I/O 总线)	8 路数字量输入 漏型 / 源型 (IEC Type 1 漏型) 30 VDC 35 VDC/ 0.5 s 24 VDC/ 4 mA, 标准值 15 VDC/ 2.5 mA, 最小 5 VDC/ 1 mA, 最大 500 VAC ，1 分钟 4 点 4.5 ms 最大 1 mA 350 m 500 m 8 8 30 mA

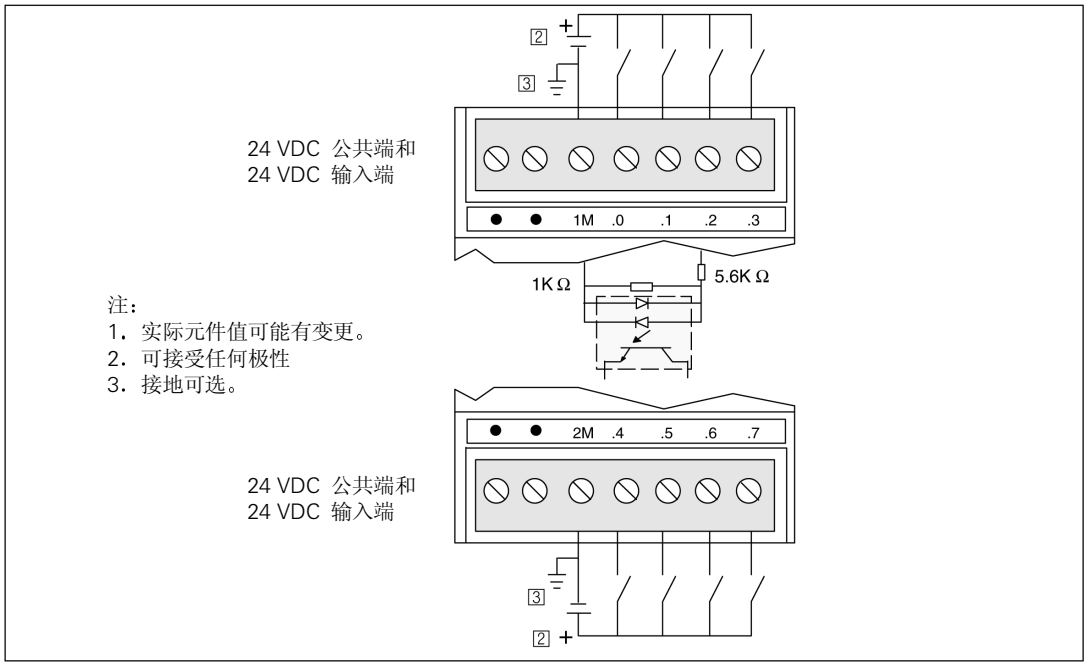


图 A-10　EM221 数字量输入 8 x 24 VDC 连接器端子图

A.7 EM222 数字量输出模块性能参数

表 A-7 EM222 24 V DC 输出和继电器输出模块技术规范

型 号 定货号	EM222 24 VDC 输出 6ES7 222-1BF20-0XA0	EM222 继电器输出 6ES7 222-1HF20-0XA0
总体特性		
外形尺寸 (长× 宽×高) 重量 功耗	46 x 80 x 62 mm 150 g 2 W	46 x 80 x 62 mm 170 g 2 W
输出特性		
输出点数 输出类型	8 路数字输出 固态-MOSFET	8 路数字输出 继电器，干触点
输出电压 允许范围 标称值 最大电流时逻辑 1信号 带 10 K□ 负载时，逻辑 0 信号	 20.4 ~ 28.8 VDC 24 VDC 20 VDC，最小 0.1 VDC，最大	 5 ~ 30 VDC，或 5 ~ 250 VAC - - -
输出电流 逻辑 1信号 输出组数 输出接通的最大个数 每组 - 水平安装 (最多) 每组 - 垂直安装 (最多) 每组最大电流 灯负载 接通状态电阻 (触点电阻) 每点的漏电流 浪涌电流 过流保护 隔离 光电隔离 隔离电阻 线圈到触点的隔离 触点之间的隔离 每组	 0.75 A 2 8 4 4 3 A 5 W 0.3 Ω 10 μA，最大 8 A，100 ms，最大 无 500 VAC，1 分钟 - - - 4点	 2.00 A 2 8 4 4 8 A 30 W DC/200 W AC 开始使用时最大 0.002 Ω - 7 A，触点闭合时 无 - 当开始使用时最小 100M Ω 1500 VAC ，1 分钟 750 VAC，1 分钟 4点
感性负载嵌位 重复的 能量吸收 <0.5 LI^2 x 开关速率 嵌位电压限制	 1 W, 所有通道 正负 48 V	 - -
输出延时 Off 到 On On 到 Off	 50 μs, 最大 200 μs, 最大	 - -
继电器 开关延时 机械寿命 (无负载) 带标称负载时触点寿命	 - - -	 10 ms，最大 10,000,000 开/关 循环 100,000 开/关 循环
电缆长度 不屏蔽 屏蔽 电能消耗 从+5 VDC (从 I/O 总线) 从 L+ L+ 线圈电源电压范围	 150 m 500 m 50 mA - -	 150 m 500 m 40 mA 当接通时每点为 9 mA 20.4 ~ 28.8 VDC

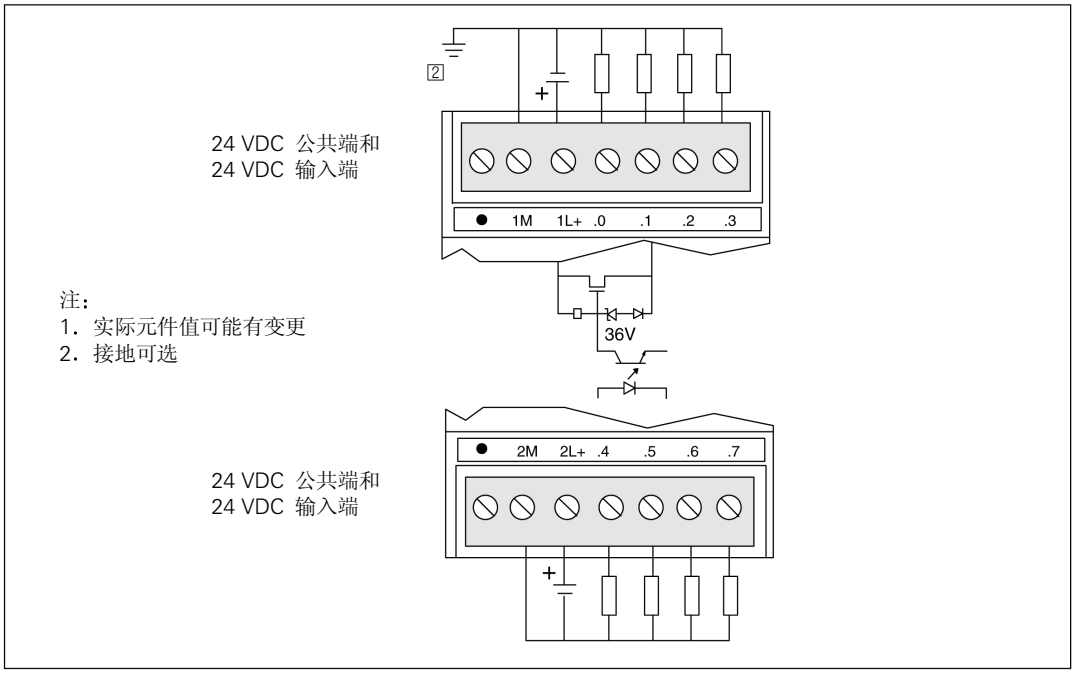


图 A-11　EM222 数字输出 8 x 24 VDC 连接器端子图

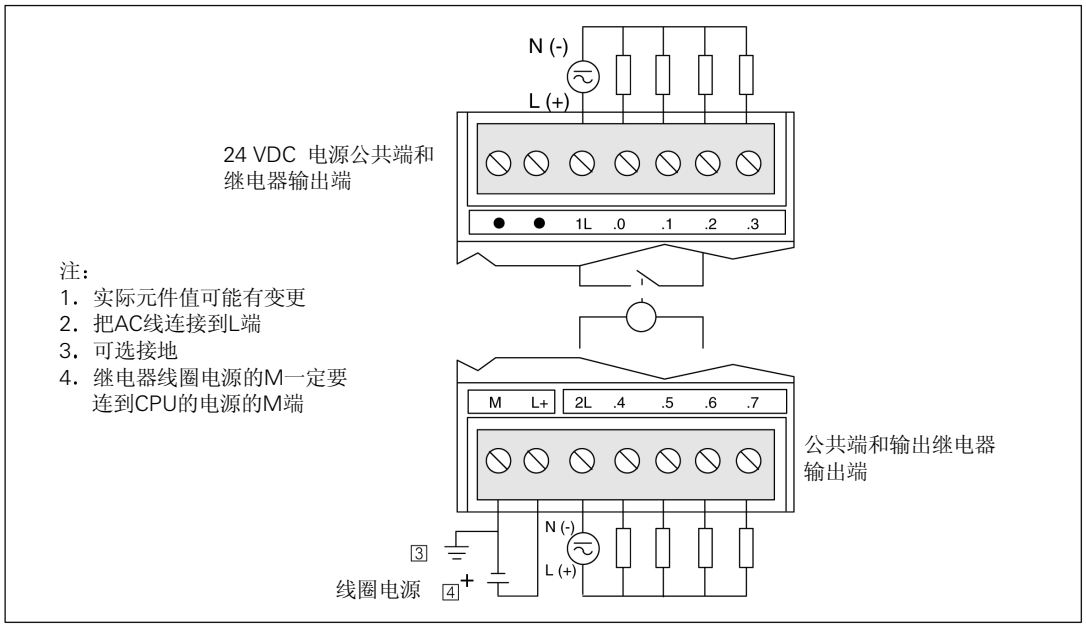


图 A-12　EM222 数字输出 8 x 继电器 连接器端子图

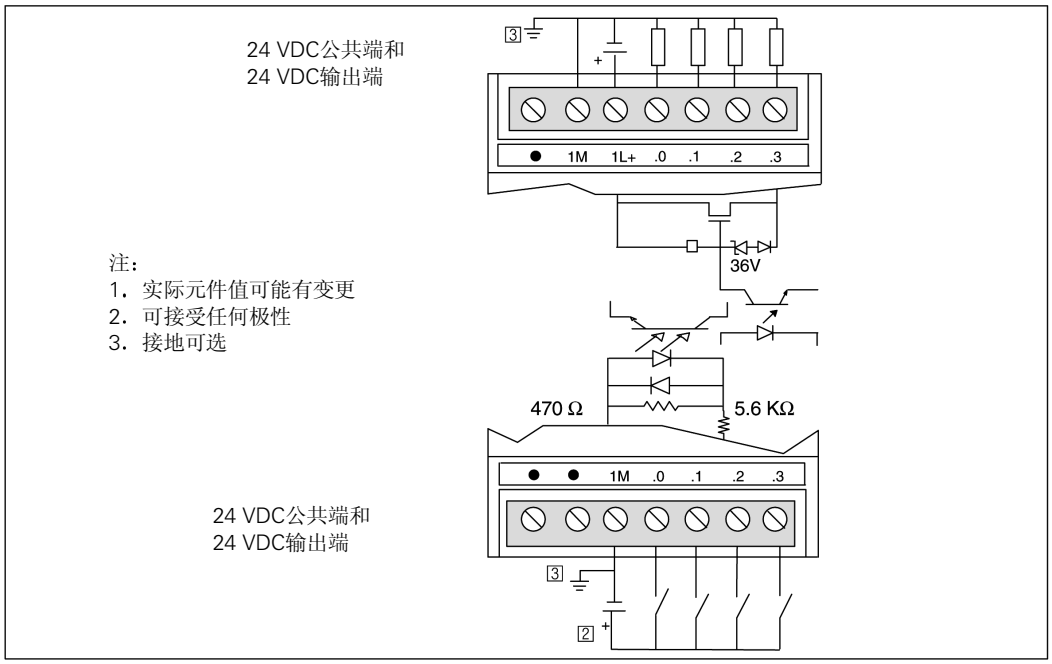
A.8 EM 223 数字量混合模块，4 输入 / 4 输出性能参数

表 A-8 EM 223 24 VDC 4输入/ 4输出，和 EM 223 24 VDC 4输入/ 4继电器输出技术规范

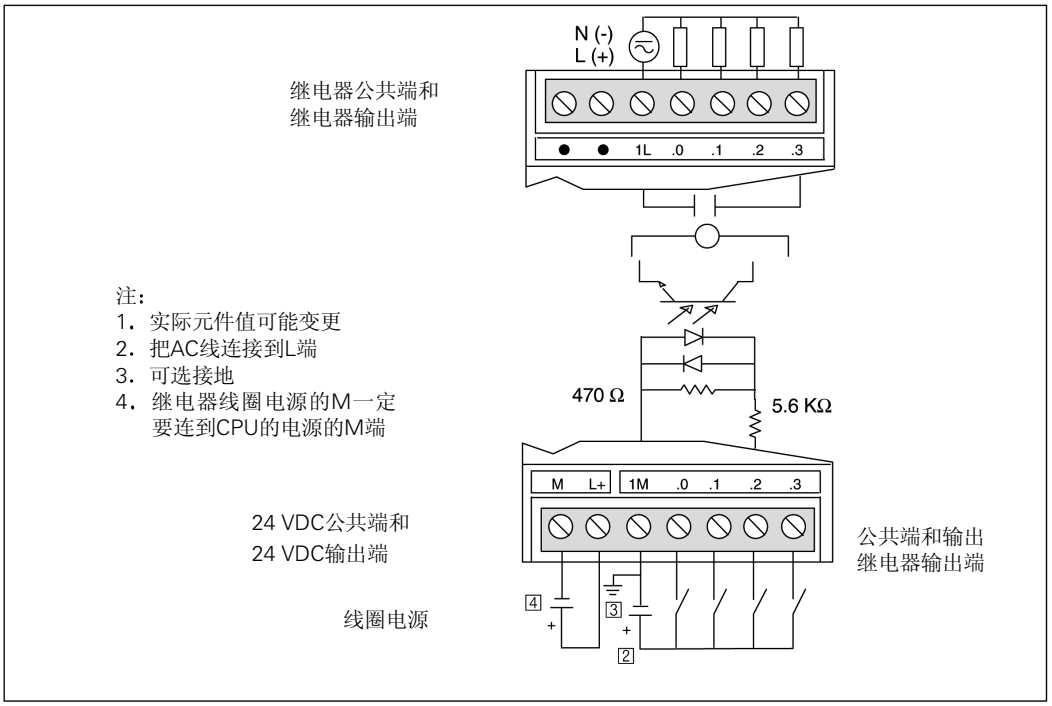
型 号 定货号	EM 223 DI4/DO4×24VDC 6ES7 223-1BF20-0XA0	EM 223 DI4/DO4×继电器 6ES7 223-1HF20-0XA0
总体特性		
外形尺寸 (长× 宽×高) 重量 功耗	46 mm x 80 mm x 62 mm 160 g 2 W	46 mm x 80 mm x 62 mm 170 g 2 W
输入特性		
输入点数 输入类型	4 输入 漏型/源型 (IEC Type 1 漏型)	4 输入 漏型/源型 (IEC Type 1 漏型)
输入电压 允许的最大连续值 浪涌 标称值 逻辑 1 信号 (最小) 逻辑 0 信号 (最大)	 30 VDC 35 VDC / 0.5 s 24 VDC / 4 mA, 标准值 15 VDC / 2.5 mA, 最小 5 VDC / 1 mA, 最大	 30 VDC 35 VDC / 0.5 s 24 VDC / 4 mA, 标准值 15 VDC / 2.5 mA, 最小 5 VDC / 1 mA, 最大
隔离 光电隔离 隔离组	 500 VAC ，1 分钟 4 点	 500 VAC ，1 分钟 4 点
输入延时 最大 连接 2 线接近开关传感器 (Bero) 允许漏电流	 4.5 ms 1 mA	 4.5 ms 1 mA
电缆长度 不屏蔽 屏蔽	 300 m 500 m	 300 m 500 m
同时接通的输入数 40 ℃ 55 ℃	 4 4	 4 4
输出特性		
输出点数 输出类型 **输出电压** 允许范围 标称值 最大电流时逻辑 1 信号 带 10 KΩ 负载时，逻辑 0 信号	4 固态-MOSFET 20.4 ~ 28.8 VDC 24 VDC 20 VDC, 最小 0.1 VDC, 最大	4 继电器，干触点 5 ~ 30 VDC 或 5 ~ 250 VAC - - -

表 A-8 EM 223 24 VDC 4输入/ 4输出，和 EM 223 24 VDC 4输入/ 4继电器输出技术规范

型 号 定货号	EM 223 DI4/DO4×24VDC 6ES7 223-1BF20-0XA0	EM 223 DI4/DO4×继电器 6ES7 223-1HF20-0XA0
输出电流		
逻辑 1 信号	0.75 A	2.00 A
输出组数	1	1
输出接通的最大个数	4	4
每组 - 水平安装 (最多)	4	4
每组 - 垂直安装 (最多)	4	4
每组最大电流	3 A	8 A
灯负载	5 W	30 W DC/200 W AC
接通状态电阻 (触点电阻)	0.3 Ω	开始使用时最大 0.2 Ω
每点的漏电流	10 µA，最大	-
浪涌电流	8 A ，100 ms，最大	7 A，触点闭合时
过流保护	无	无
隔离		
光电隔离	500 VAC，1 分钟	-
隔离电阻	-	当开始使用时，最小 100 MΩ
线圈到触点的隔离	-	1500 VAC，1 分钟
触点之间的隔离	-	750 VAC，1 分钟
每组	4点	4点
感性负载嵌位		
重复的能量吸收	1 W，所有通道	-
<0.5 LI^2 x 开关速率		
嵌位电压限制	L+ − 48 V	-
输出延时		
Off 到 On	50 µs，最大	-
On 到 Off	200 µs，最大	-
继电器		
开关延时	-	10 ms，最大
机械寿命 (无负载)	-	10,000,000 开/关 循环
带标称负载时触点寿命	-	100,000 开/关 循环
电缆长度		
不屏蔽	150 m	150 m
屏蔽	500 m	500 m
电能消耗		
从+5 VDC (从 I/O 总线)	40 mA	40 mA
从 L+	-	当接通时每点为 9 mA
L+ 线圈电源电压范围	-	20.4 ~ 28.8 VDC



图A-13　EM 223 数字量混合输入/输出模块（4×24 VDC 输入/4×24 VDC 输出）连接器端子图



图A-14　EM 223 数字量混合输入/输出模块（4×24 VDC 输入/4× 继电器输出）连接器端子图

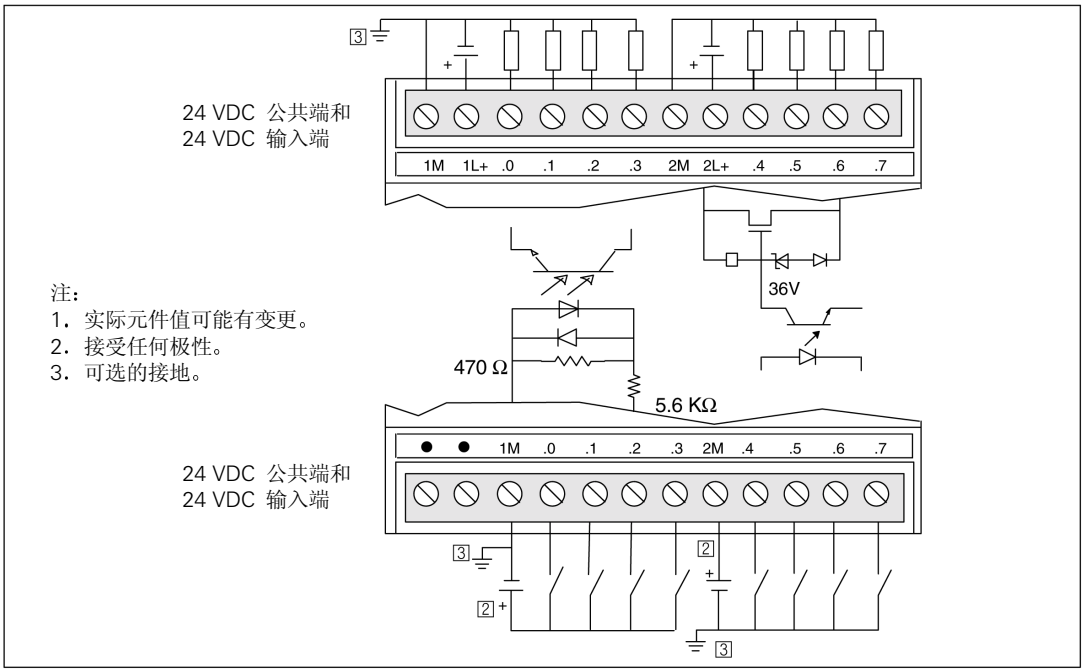
A.9 EM223 数字量混合模块，8 输入 / 8 输出性能参数

表 A-9 EM223 24 VDC 8 输入 /8 输出，和 EM223 24 VDC 8 输入 /8 继电器输出技术规范

型 号 定货号	EM223 24VDC 输入/输出 6ES7 223-1BH20-0XA0	EM223 24VDC 输入/继电器输出 6ES7 223-1PH20-0XA0
总体特性		
外形尺寸 (长× 宽×高) 重量 功耗	71.2 mm x 80 mm x 62 mm 200 g 3 W	71.2 mm x 80 mm x 62 mm 300 g 3 W
输入特性		
输入点数 输入类型	8 输入 漏型/源型 (IEC Type 1 漏型)	8 输入 漏型/源型 (IEC Type 1 漏型)
输入电压 允许的最大连续值 浪涌 标称值 逻辑 1 信号 (最小) 逻辑 0 信号 (最大)	 30 VDC 35 VDC / 0.5 s 24 VDC / 4 mA, 标准值l 15 VDC / 2.5 mA, 最小 5 VDC / 1 mA, 最大	 30 VDC 35 VDC / 0.5 s 24 VDC / 4 mA, 标准值 15 VDC / 2.5 mA, 最小 5 VDC / 1 mA, 最大
隔离 光电隔离 隔离组	 500 VAC ，1 分钟 4 点	 500 VAC ，1 分钟 4 点
输入延时 最大	 4.5 ms	 4.5 ms
连接 2 线接近开关传感器 (Bero) 允许漏电流	 1 mA	 1 mA
电缆长度 不屏蔽 屏蔽	 300 m 500 m	 300 m 500 m
同时接通的输入数 40 ° C 55 ° C	 8 8	 8 8
输出特性		
输出点数 输出类型	8 路数字输出 固态-MOSFET	8 路数字输出 继电器，干触点
输出电压 允许范围 标称值 最大电流时逻辑 1 信号 带 10 KΩ 负载时，逻辑 0 信号d	 20.4 ~ 28.8 VDC 24 VDC 20 VDC, 最小 0.1 VDC, 最大	 5 ~ 30 VDC 或 5 ~ 250 VAC - - -

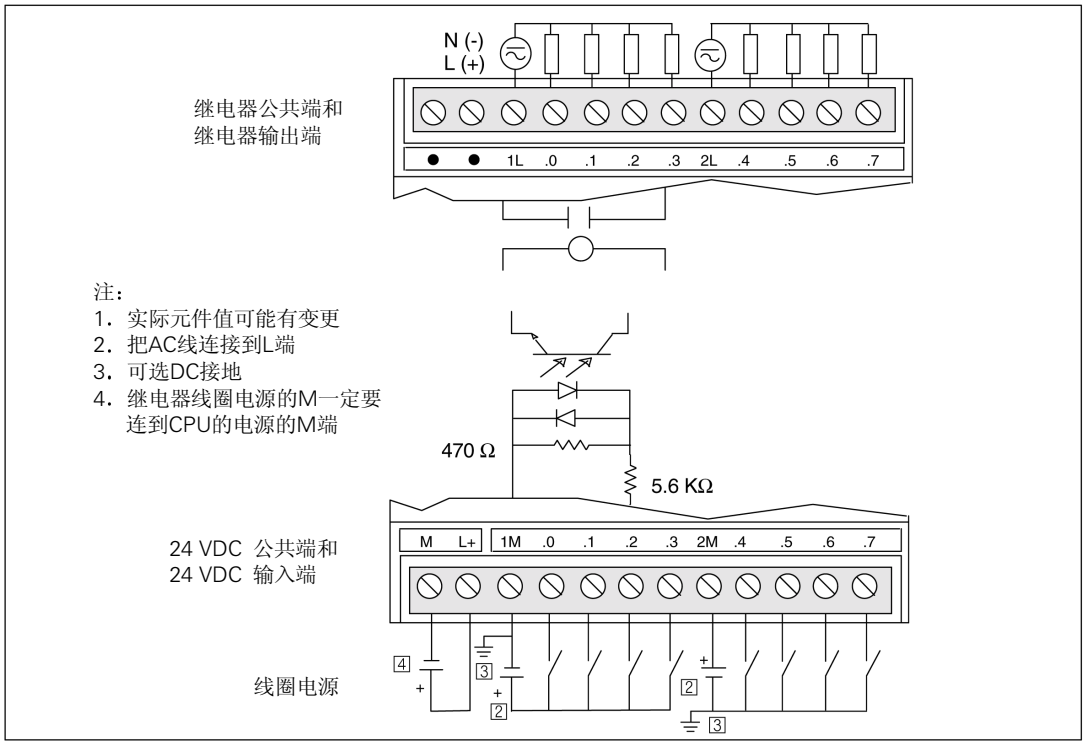
表 A-9 EM223 24 VDC 8 输入 /8 输出，和 EM223 24 VDC 8 输入 /8 继电器输出技术规范

型 号 定货号	EM223 24VDC 输入/输出 6ES7 223-1BH20-0XA0	EM223 24VDC输入/继电器输出 6ES7 223-1PH20-0XA0
输出电流		
逻辑 1 信号	0.75 A	2.00 A
输出组数	2	2
输出接通的最大个数	8	8
每组 - 水平安装 (最多)	4	4
每组 - 垂直安装 (最多)	4	4
每组最大电流	2 A	8 A
灯负载	5 W	30 W DC/200 W AC
接通状态电阻 (触点电阻)	0.3 Ω	开始使用时最大 0.002 Ω
每点的漏电流	10 µA, 最大	-
浪涌电流	8 A ，100 ms, 最大	7 A，触点闭合时
过流保护	无	无
隔离		
光电隔离	500 VAC，1 分钟	-
隔离电阻	-	当开始使用时，最小 100 M Ω
线圈到触点的隔离	-	1500 VAC，1 分钟
触点之间的隔离	-	750 VAC，1 分钟
每组	4点	4点
感性负载嵌位		
重复的能量吸收 < 0.5 LI^2 x 开关速率	1 W, 所有通道	-
嵌位电压限制	正负 48 V	-
输出延时		
Off 到 On	50 µs, 最大	-
On 到 Off	200 µs, 最大	-
继电器		
开关延时	-	10 ms，最大
机械寿命 (无负载)	-	10,000,000 开/关 循环
带标称负载时触点寿命	-	100,000 开/关 循环
电缆长度		
不屏蔽	150 m	150 m
屏蔽	500 m	500 m
电能消耗		
从+5 VDC (从 I/O 总线)	100 mA	80 mA
从 L+	-	当接通时每点为 9 mA
L+ 线圈电源电压范围	-	20.4 ~ 28.8 VDC



注：
1．实际元件值可能有变更。
2．接受任何极性。
3．可选的接地。

图 A-15　　EM223 数字混合模块 (8×24VDC输入/8×VDC 输出) 连接器



注：
1．实际元件值可能有变更
2．把AC线连接到L端
3．可选DC接地
4．继电器线圈电源的M一定要连到CPU的电源的M端

图 A-16　　EM223 数字8×24 VDC 输入 /8× 继电器 输出端子图

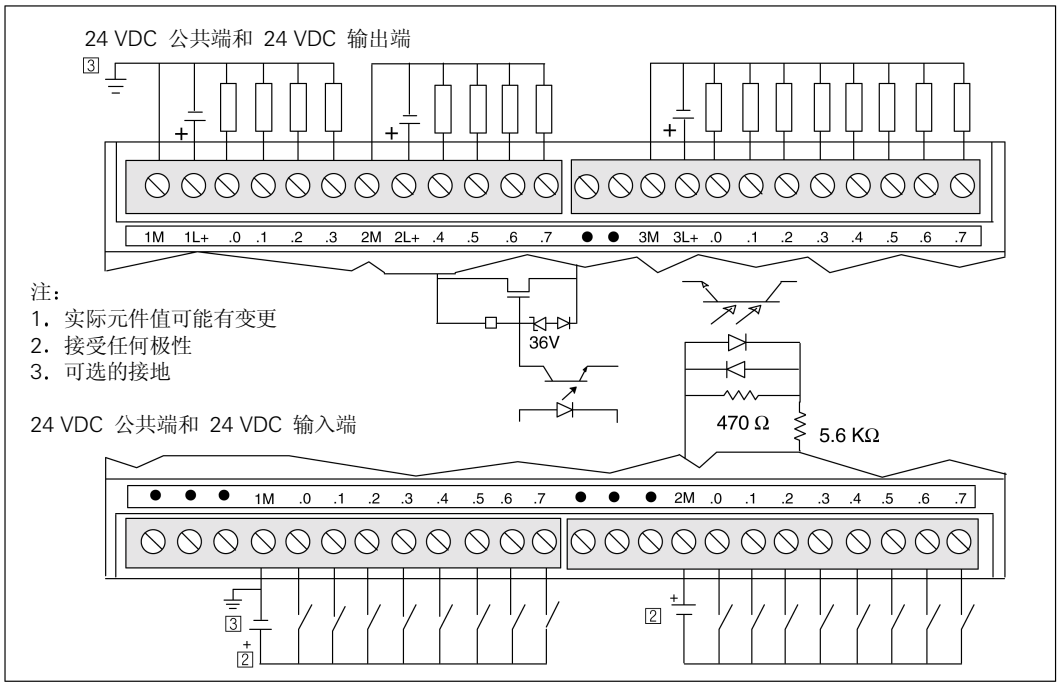
A.10 EM223 数字量混合模块，16输入 / 16输出性能参数

表 A-10 EM223 24 VDC 16 输入/16 输出，和 EM223 24 VDC 16 输入/ 16 继电器输出技术规范

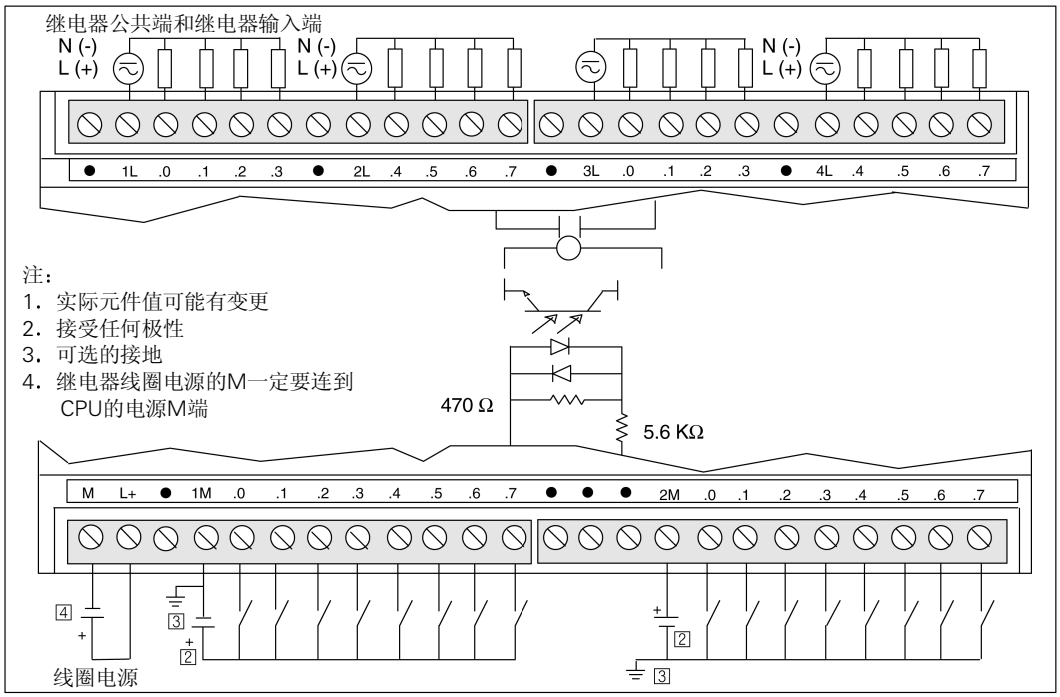
型 号 定货号	EM223 DI16/DO16XDC24V 6ES7 223-1BL20-0XA0	EM223 DI16/DO16XDC24V/Rly 6ES7 223-1PL20-0XA0
总体特性		
外形尺寸 (长× 宽×高) 重量 功耗	137.3 mm x 80 mm x 62 mm 360g 6 W	137.3 mm x 80 mm x 62 mm 400 g 6 W
输入特性		
输入点数 输入类型	16输入 漏型/源型 (IEC Type 1 漏型)	16输入 漏型/源型 (IEC Type 1 漏型)
输入电压 允许的最大连续值 浪涌 标称值 逻辑 1 信号 (最小) 逻辑 0 信号 (最大)	 30 VDC 35 VDC / 0.5 s 24 VDC / 4 mA, 标准值I 15 VDC / 2.5 mA, 最小 5 VDC / 1 mA, 最大	 30 VDC 35 VDC / 0.5 s 24 VDC / 4 mA, 标准值 15 VDC / 2.5 mA, 最小 5 VDC / 1 mA, 最大
隔离 光电隔离 隔离组	 500 VAC ，1 分钟 8 点	 500 VAC ，1 分钟 8点
输入延时 最大	 4.5 ms	 4.5 ms
连接 2 线接近开关传感器 (Bero) 允许漏电流	 1 mA	 1 mA
电缆长度 不屏蔽 屏蔽	 300 m 500 m	 300 m 500 m
同时接通的输入数 40 ° C 55 ° C	 16 16	 16 16
输出特性		
输出点数 输出类型	16 路数字输出 固态-MOSFET	16 路数字输出 继电器，干触点
输出电压 允许范围 标称值 最大电流时逻辑 1 信号 带 10 KΩ 负载时，逻辑 0 信号	 20.4 ~ 28.8 VDC 24 VDC 20 VDC, 最小 0.1 VDC, 最大	 5 ~ 30 VDC 或 5 ~ 250 VAC - - -

表 A-10 EM223 24 VDC 16 输入/16 输出，和 EM223 24 VDC 16 输入/ 16 继电器输出技术规范

型 号 定货号	EM223 24VDC 输入/输出 6ES7 223-1BL20-0XA0	EM223 24VDC输入/继电器输出 6ES7 223-1PL20-0XA0
输出电流		
逻辑 1 信号	0.75 A	2.00 A
输出组数	3	4
输出接通的最大个数	16	16
每组 - 水平安装 (最多)	4/4/8	4
每组 - 垂直安装 (最多)	4/4/8	4
每组最大电流	3/3/6 A	8 A
灯负载	5 W	30 W DC/200 W AC
接通状态电阻 (触点电阻)	0.3 Ω	开始使用时最大 0.2 Ω
每点的漏电流	10 µA, 最大	-
浪涌电流	8 A ，100 ms, 最大	7 A，触点闭合时
过流保护	无	无
隔离		
光电隔离	500 VAC，1 分钟	-
隔离电阻	-	当开始使用时，最小100 M Ω
线圈到触点的隔离	-	1500 VAC，1 分钟
触点之间的隔离	-	750 VAC，1 分钟
每组	4/4/8点	4点
感性负载嵌位		
重复的能量吸收 < 0.5 LI^2 x 开关速率	1 W, 所有通道	-
嵌位电压限制	正负 48 V	-
输出延时		
Off 到 On	50 µs, 最大	-
On 到 Off	200 µs, 最大	-
继电器		
开关延时	-	10 ms，最大
机械寿命 (无负载)	-	10,000,000 开/关 循环
带标称负载时触点寿命	-	100,000 开/关 循环
电缆长度		
不屏蔽	150 m	150 m
屏蔽	500 m	500 m
电能消耗		
从+5 VDC (从 I/O 总线)	160 mA	150 mA
从 L+	-	当接通时每点为 9 mA
L+ 线圈电源电压范围	-	20.4 ~ 28.8 VDC



图A-17　EM223数字量混合输入/输出模块（16×24 VDC 输入/16×24 VDC 输出）连接器端子图



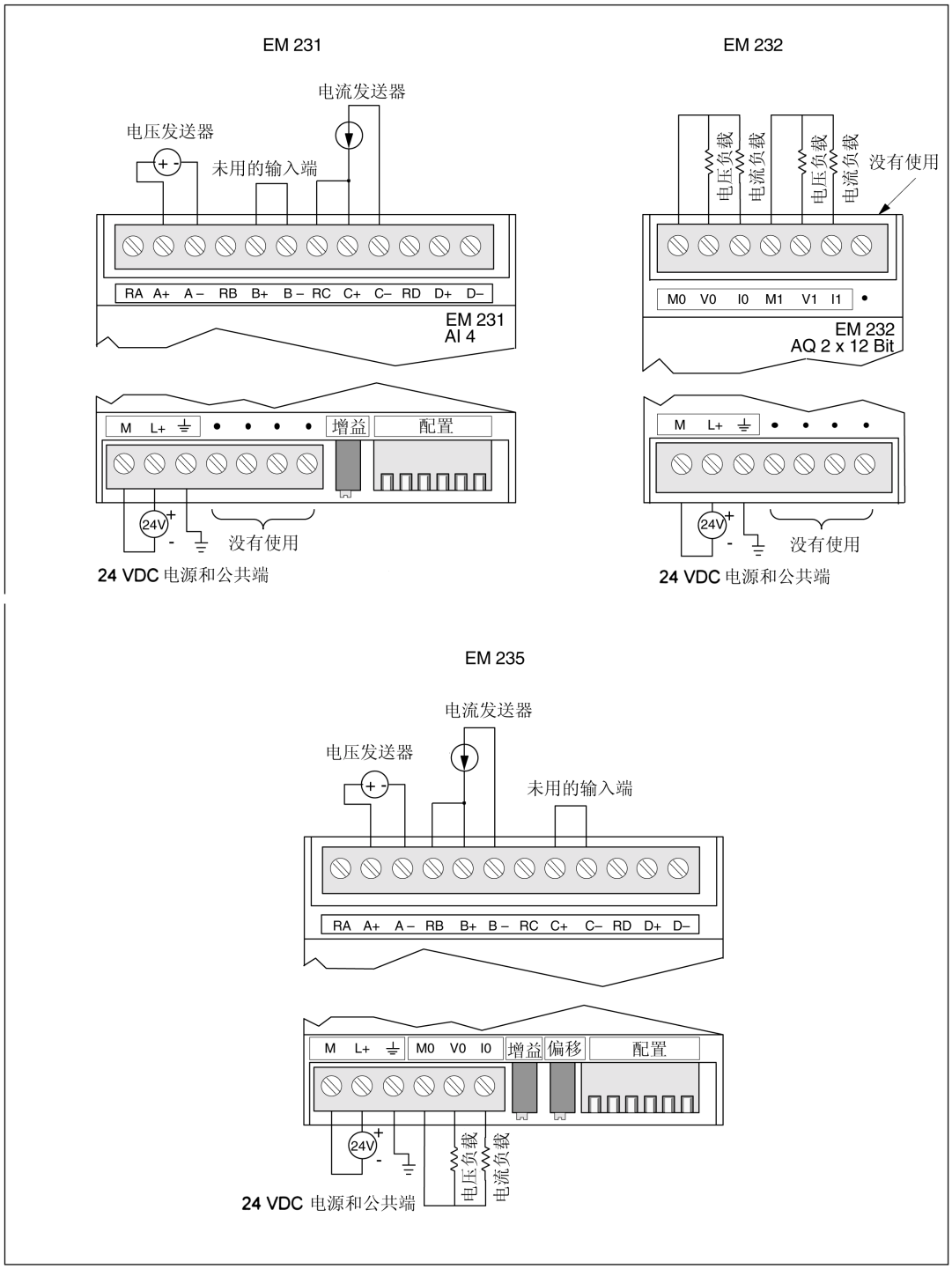
图A-18　EM223数字量混合输入/输出模块（16×24 VDC 输入/16×继电器输出）连接器端子图

A.11 EM 231，EM 232和EM 235模拟量输入，输出和组合模块的技术规范

表A-11 EM 231，EM 232和EM 235模拟量输入，输出和组合模块的技术规范

说明 订货号	EM 231 AI4 x 12位 6ES7 231-0HC20-0XA0	EM 232 AQ2 x 12位 6ES7 232-0HB20-0XA0	EM 235 AI4/AQ1 x 12位 6ES7 235-0KD20-0XA0	
	输入技术规范	输出技术规范	输入技术规范	输出技术规范
通用技术规范				
尺寸(W x H x D) 重量 功率损失(耗散)	71.2 mm x 80 x 62 mm 183 g 2 W	46mm x 80mm x 62mm 148 g 2 W	71.2mm x 80 x 62 mm 186 g 2 W	
物理I/O数量	4模拟量输入点	2模拟量输出点	4模拟量输入点，1模拟量输出点	
功耗 从+5V DC (从I/O总线) 从L+ L+电压范围， 级2或DC传感器供电	20 mA 60 mA 20.4至28.8	20 mA 70 mA(2个输出均为20 mA) 20.4至28.8	30 mA 60 mA(输出为20mA) 20.4至28.8	
LED指示器	24 VDC电源良好 ON=没有故障， OFF=无24VDC电源	24 VDC电源良好 ON=没有故障， OFF=无24VDC电源	24 VDC电源良好 ON=没有故障， OFF=无24VDC电源	
模拟量输入技术规范				
数据字格式 双极性，全量程范围 单极性，全量程范围	(见图A-21) -32000至+32000 0至32000		(见图A-21) -32000至+32000 0至32000	
输入阻抗	≥10 MΩ		≥10 MΩ	
输入滤波器衰减	-3 db @ 3.1 Khz		-3 db @ 3.1 Khz	
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	
模拟量输入点数	4		4	
隔离(现场侧到逻辑线路)	无		无	
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10 V，0至5V ±5V，±2.5V 0至20mA		0至10 V，0至5V 0到 1V,0到 500mV, 0到100mV,0到 50mV ±10V，±5V，±2.5V ±1V，±500mV ±250mV，±100mV ±50mV，±25mV 0到20mA	
输入分辨率 电压(单极性) 电压(双极性) 电流	见表A-5		见表A-13	
模拟量到数字量的转换时间	<250 uS		<250 uS	
模拟量输入阶跃响应	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)	

说明 订货号	EM 231 A14 x 12位 6ES7 231-0HC20-0XA0	EM 232 AQ2 x 12位 6ES7 232-0HB20-0XA0	EM 235 A14/AQ1 x 12位 6ES7 235-0KD20-0XA0	
	输入技术规范	输出技术规范	输入技术规范	输出技术规范
模拟量输入技术规范				
模拟量输出点数		2		1
隔离(现场侧到逻辑线路)		无		无
信号范围 电压输出 电注输出		 ±10V 0 到 20mA		 ±10V 0到 20mA
分辨率全量程 电压 电流		 12位 11位		 12位 11位
数据字格式 电压 电流		 -32000到+32000 0到+32000		 -32000到+32000 0到+32000
精度 最差情况,（0°~55℃） 电压输出 电流输出 典型情况（25℃） 电压输出 电流输出		 满量程的2% 满量程的2% 满量程的0.5% 满量程的0.5%		 满量程的2% 满量程的2% 满量程的0.5% 满量程的0.5%
设置时间 电压输出 电流输出		 100us 2mS		 100us 2mS
最大驱动 电压输出 电流输出		 最小 5000Ω 最大500Ω		 最小 5000Ω 最大500Ω



图A-19　用于EM 231,EM 232和EM235扩展模块的连接器的端子标识

输入校准

校准调节影响模拟多路转换器后的仪器放大器级（见图A-22）。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。

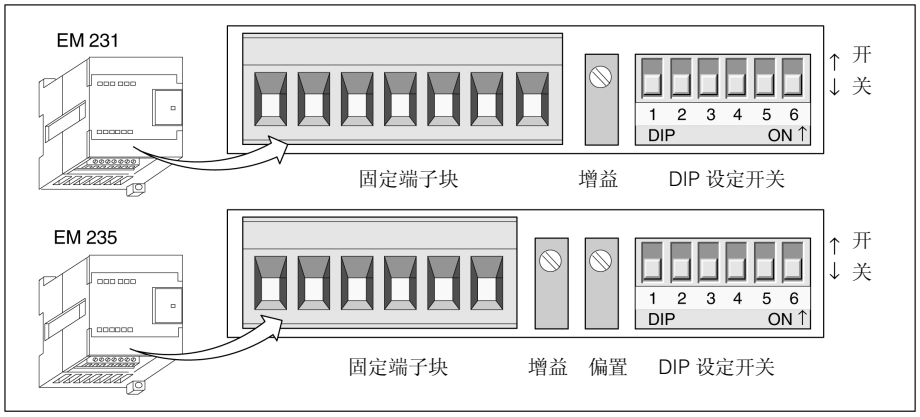
为了达到表中所列的技术参数，应起动用于模块所有输入的模拟输入滤波器。计算平均值时，选择64次或更多的采样次数。

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. 用一个传感器，一个电压源或一个电流源，将零值信号加到一个输入端。
4. 使入适当的输入通道，并读出CPU中测量值。
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-20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-20　用于EM 231，EM 235的校准电位计和配置DIP开关

EM 231配置

表A-12所示为如何用DIP开关设置EM 231模块。开关1、2和3可选择模拟量输入范围。所有的输入设置成相同的模拟量输入范围。下表中，ON为接通，OFF为断开。

表A-12 EM 231选择模拟量输入范围的开关表

单极性			满量程输入	分辨率
SW1	SW2	SW3		
ON	OFF	ON	0到10V	2.5mV
	ON	OFF	0到5V	1.25mV
			0到20mA	5uA
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

EM 235配置

表A-13所示为如何用DIP开关设置EM 235模块。开关1到6可选择模拟量输入范围和分辨率。所有的输入设置成相同的模拟量输入范围和格式。表A-14所示为如何选择单/双极性（开关6）、增益（开关4和5）和衰减（开关1、2和3）。下表中，ON为接通，OFF为断开。

表A-13 EM 235选择模拟量输入范围和分辨率的开关表

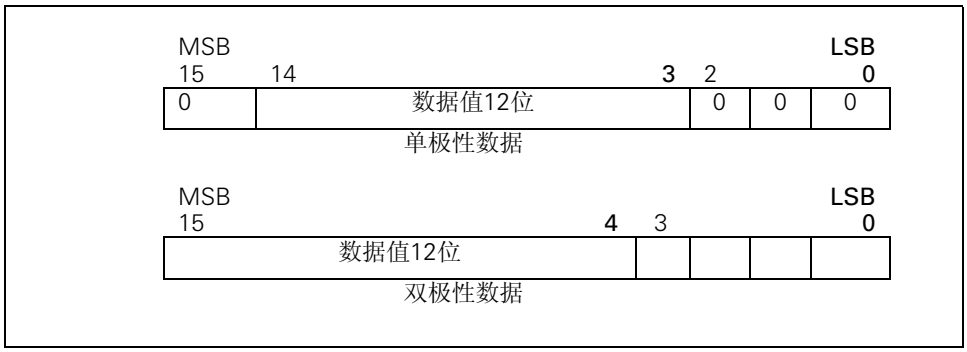
单极性						满量程输入	分辨率
SW1	SW2	SW3	SW4	SW5	SW6		
ON	OFF	OFF	ON	OFF	ON	0到50mV	12.5μV
OFF	ON	OFF	ON	OFF	ON	0到100mV	25μV
ON	OFF	OFF	OFF	ON	ON	0到500mV	125uA
OFF	ON	OFF	OFF	ON	ON	0到1V	250μV
ON	OFF	OFF	OFF	OFF	ON	0到5V	1.25mV
ON	OFF	OFF	OFF	OFF	ON	0到20mA	5μA
OFF	ON	OFF	OFF	OFF	ON	0到10V	2.5mV
双极性						**满量程输入**	**分辨率**
SW1	**SW2**	**SW3**	**SW4**	**SW5**	**SW6**		
ON	OFF	OFF	ON	OFF	OFF	±25mV	12.5μV
OFF	ON	OFF	ON	OFF	OFF	±50mV	25μV
OFF	OFF	ON	ON	OFF	OFF	±100mV	50μV
ON	OFF	OFF	OFF	ON	OFF	±250mV	125μV
OFF	ON	OFF	OFF	ON	OFF	±500	250μV
OFF	OFF	ON	OFF	ON	OFF	±1V	500μV
ON	OFF	OFF	OFF	OFF	OFF	±2.5V	1.25mV
OFF	ON	OFF	OFF	OFF	OFF	±5V	2.5mV
OFF	OFF	ON	OFF	OFF	OFF	±10V	5mV

表A-14　EM 235选择单/双极性、增益和衰减的开关表

EM 235开关						单/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-21 为CPU中模拟量输入字中12位数据值的存放位置。



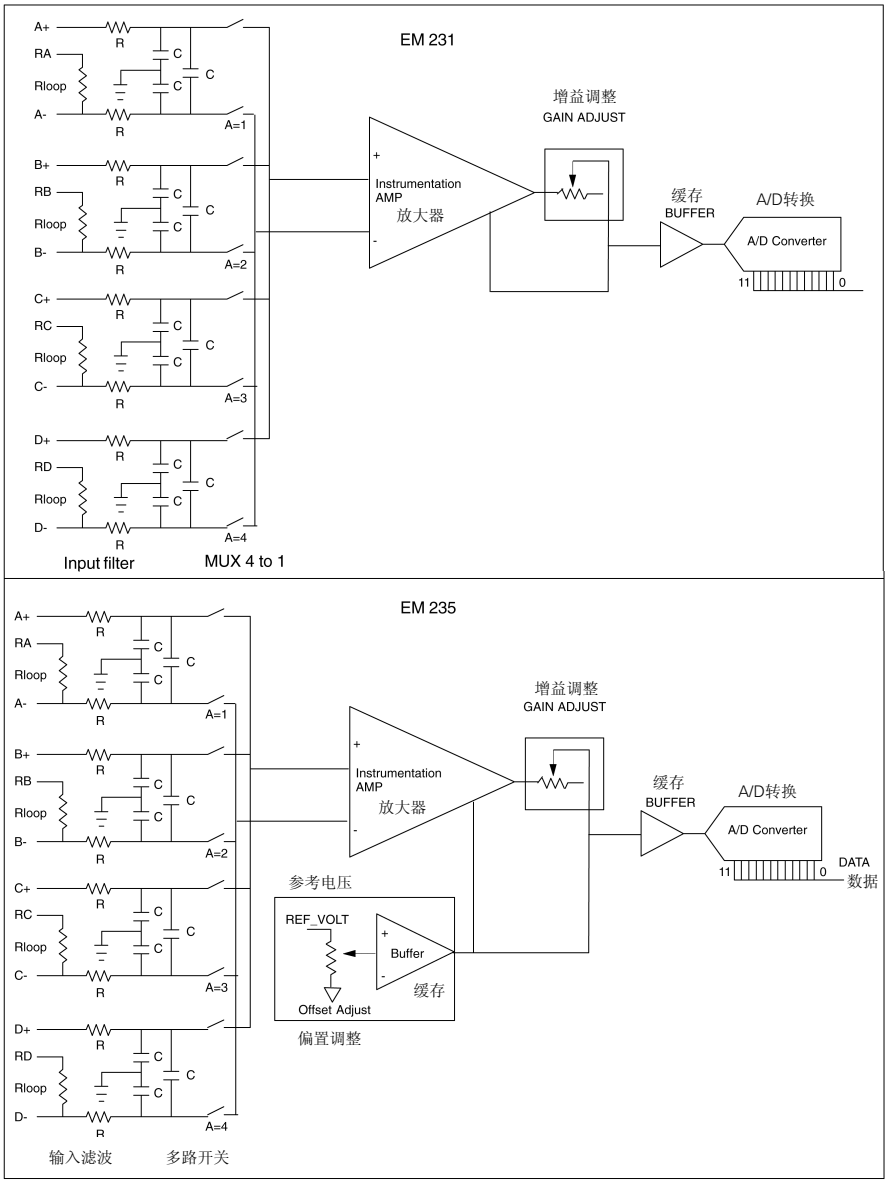
图A-21　EM 231和EM 235数据输入字格式

注意

模拟量到数字量转换器(ADC)的12位读数，其数据格式是左端对齐的。最高有效位是符号位：0表示是正值数据字，对单极性格式，3个连续的0使得ADC计数数值每变化1个单位则数据字的变化是以8为单位变化的。对双极性格式，4个连续的0使得ADC计数数值每变化1个单位，则数据字的变化是以16为单位变化的。

EM 231和EM 235的输入方框图

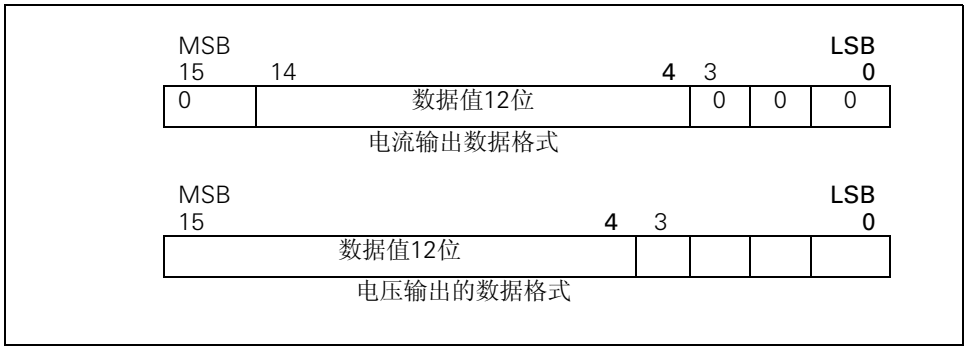
图A-22为EM 231和EM 235的输入方框图。



图A-22　EM 231 和 EM 235的输入方框图

EM 232和EM 235输出数据字格式

图A-23为CPU中模拟量输出字中12位数据值的存放位置。



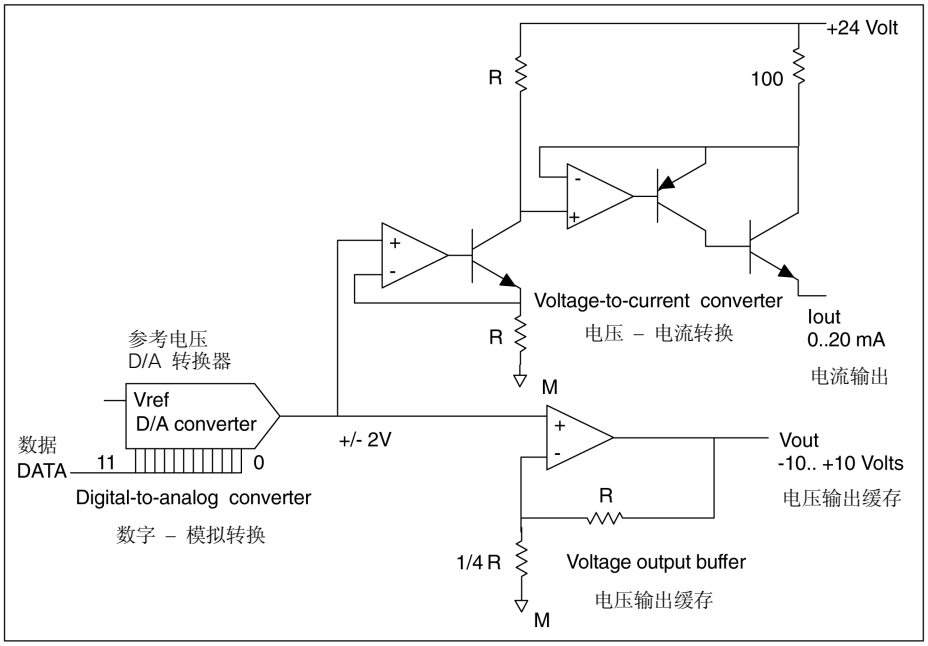
图A-23　EM 232和EM 235输出数据字格式

注意

数字量到模拟量转换器（DAC）的12位读数，其输出数据格式是左端对齐的，最高有效位：0表示是正值数据字，数据在装载到DAC寄存器之前，4个连续的0是被裁断的，这些位不影响输出信号值。

EM 232和EM 235输出方框图

图A-24所示为EM 232和EM 235输出方框图。



图A-24　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。
- 仅在传感器侧端接屏蔽。
- 未用通道应短接，见图A-19。
- 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，应以正确的角度相交。
- 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

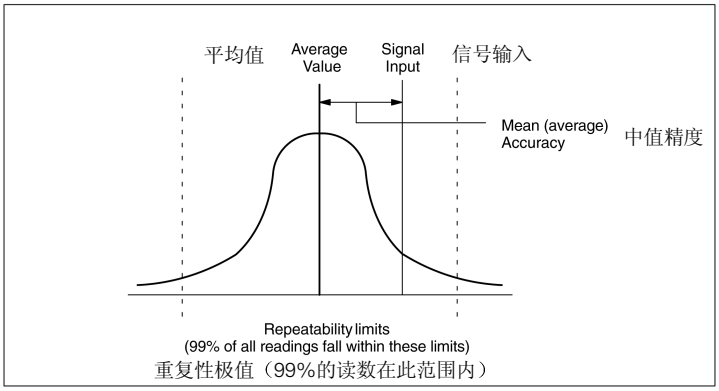
建议EM 231和EM 235扩展模块不用于热电耦。

理解和使用模拟量输入模块：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149us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。

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。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）。重复性在图A-25中用钟形曲线描述。该图表示99%重复性极限，各个读数的中值或平均值，以及以图形表示的平均精度。表A-15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-25　精度定义

表A-15 EM 231 和 EM 235 的技术规范

满量程输入范围	重复性		平均精度[1, 2, 3, 4]	
	全量程的%	计数	满量程的%	计数
EM 231的技术规范				
0到5V	±0.075%	±24	±0.01%	±32
0到20mA				
0到10V				
±2.5V		±48	±0.05%	
±5V				
EM 235的技术规范				
0到50mV	±0.075%	±24	±0.25%	±80
0到100mV			±0.2%	±64
0到500mV			±0.05%	±16
0到1V				
0到5V				
0到20mA				
±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．在所选择的输入范围标定后进行测量

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．存在有通道到通道的传递转换误差，这是由于模拟量多路转换器限制的设置时间而引起的，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

模拟量技术规范和定义

- 精度：设定点对期望值的偏差
- 分辨率：LSB（最低有效位）的改变反映到输出所产生的影响

国际或国家机构的标准

这些模块遵照以下国际或国家机构的标准：UL508 列表（工业控制装置）；CSA C22.2 认证证书号 142 （过程控制装置）；FM 级 1，分部 2，组 A，B，C 和 D，危险场所，T4A；VDE0160：用于电力设备的电子装置；欧洲共同体（CE）低电压指导规程 73/23/EEC，EN61131-2；可编程序控制器 - 装置要求；欧洲共同体（CE）EMC 指导规程89/336/EEC。

A.12 EM 277 PROFIBUS DP模块的技术规范

表A-16 EM 277 PROFIBUS-DP模块的技术规范

说明 订货号	EM 277 PROFIBUS – DP 6ES7 277 – 0AA20 – 0XA0
物理数据	
尺寸（W×H×D） 重量 功率损失（耗散）	71mm×80mm×62mm 175 g 2.5W
通信性能	
节点数 电气接口 隔离（外部信号到PLC逻辑） PROFIBUS – DP/MPI波特率（自动设置）	1 port RS – 485 500 VAC(电气) 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和MPI从站
电缆长度 直到93.75K波特 187.5K波特 500K波特 1到1.5M波特 3到12M波特	 1200m 1000m 400m 200m 100m
网络容量 站地址设置 每段最大站数 每个网络最大站数 MPI连接	 0 – 99（由旋转开关设定 32 126，最大到99个EM 277站 6个，2预留（1个为编程器PG，1个为OP）
功耗 +5 VDC（从I/O点数）	 150mA
24 VDC 输入电源的要求 电压范围 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 纹波噪声（< 10MHz） [1] 隔离（输入电源到模块逻辑）	 20.4到28.8 VDC（级2或从PLC来的传感器电源） 30mA 60mA 180mA < 1 V 峰到峰（最大） 500 VAC，1分钟
通信口的5V电源 每个端口的最大电流 **隔离（输入电源到模块逻辑）**	 90mA 500 VAC，1分钟
通信口的24 V 电源 电压范围 每个端口的最大电流 电流限制 隔离	 20.4到28.8 VDC 120mA 0.7到2.4A 没有隔离与输入24 VDC的线路相同

[1] 24 VDC电源不对模块逻辑供电，24V 电源用于通信端口。

兼容性

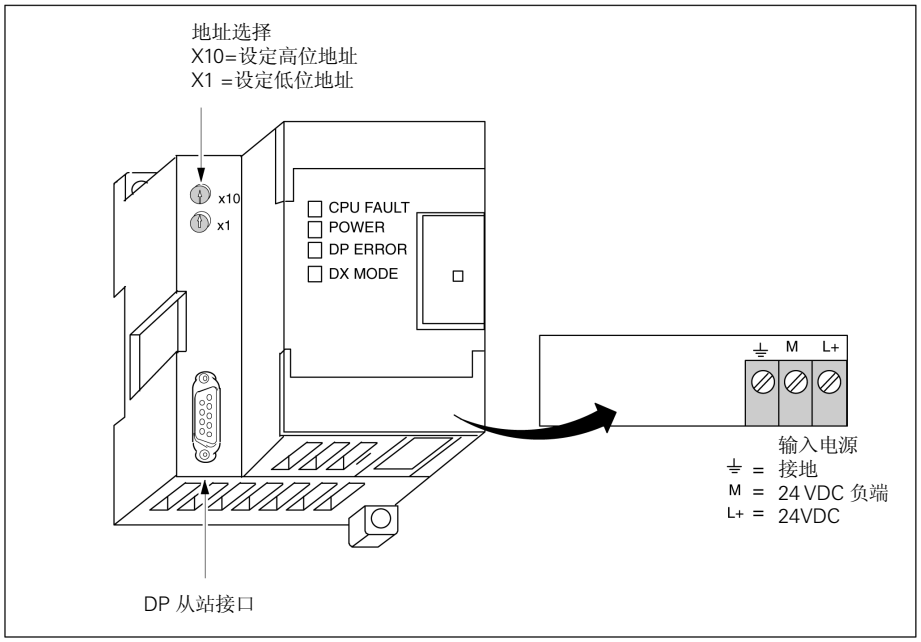
EM 277 PROFIBUS – DP从站模块，设计用于和S7 – 200 PLC一起工作的智能扩展模块，见表A-17。

表A-17　EM 277 PROFIBUS – DP模块和S7 – 200 PLC的兼容性

CPU	说明	订货号
CPU 222 版本1.10或更高	CPU 222 DC/DC/DC	6ES7 212 – 1AB21 – 0XB0
	CPU 222 AC/DC/Relay	6ES7 212 – 1BB21 – 0XB0
CPU 224 版本1.10或更高	CPU 224 DC/DC/DC	6ES7 214 – 1AD21 – 0XB0
	CPU 224 AC/DC/Relay	6ES7 214 – 1BD21 – 0XB0
CPU 226 版本1.00或更高	CPU 226 DC/DC/DC	6ES7 216 – 2AD21 – 0XB0
	CPU 226 AC/DC/Relay	6ES7 216 – 2BD21 – 0XB0

地址开关和LED

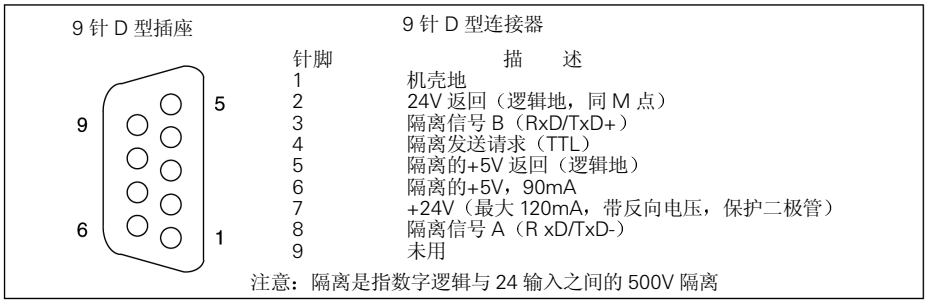
地址开关和状态LED位于模块的正面，见图A-26。 EM 277状态LED见表A-20。



图A-26　EM 277 PROFIBUS – DP模块的前视图

DP从站端口连接器

DP从站端口连接器的插针输出示于图A-27。



图A-27 DP从站端口连接器的插针

分布式外围设备（DP）的标准通信

PROFIBUS-DP（或DP标准）是由欧洲标准EN 50170和国际标准IEC 61158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，然后初始化起动其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。每次对从站的事务处理为写输出和读输入。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。

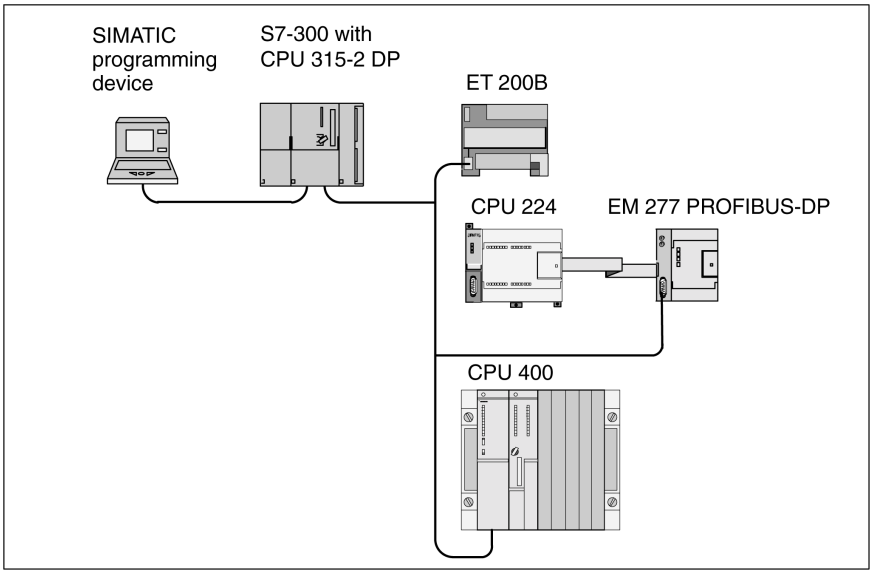
一旦DP主站已将参数和I/O配置写入到DP从站，而且从站已从主站那里接收到参数和配置，则主站就拥有那个从站。从站只能接收自其主站的写请求。网络上的其它主站可以读取该从站的输入和输出，但是它们不能向该从站写入任何信息。

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。（支持的波特率见表A-16）。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-28表示有一个CPU 224和一个EM 277 PROFIBUS-DP模块的PROFIBUS网络。在这种场合，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CPU 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-28　一个PROFIBUS 网络上的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。

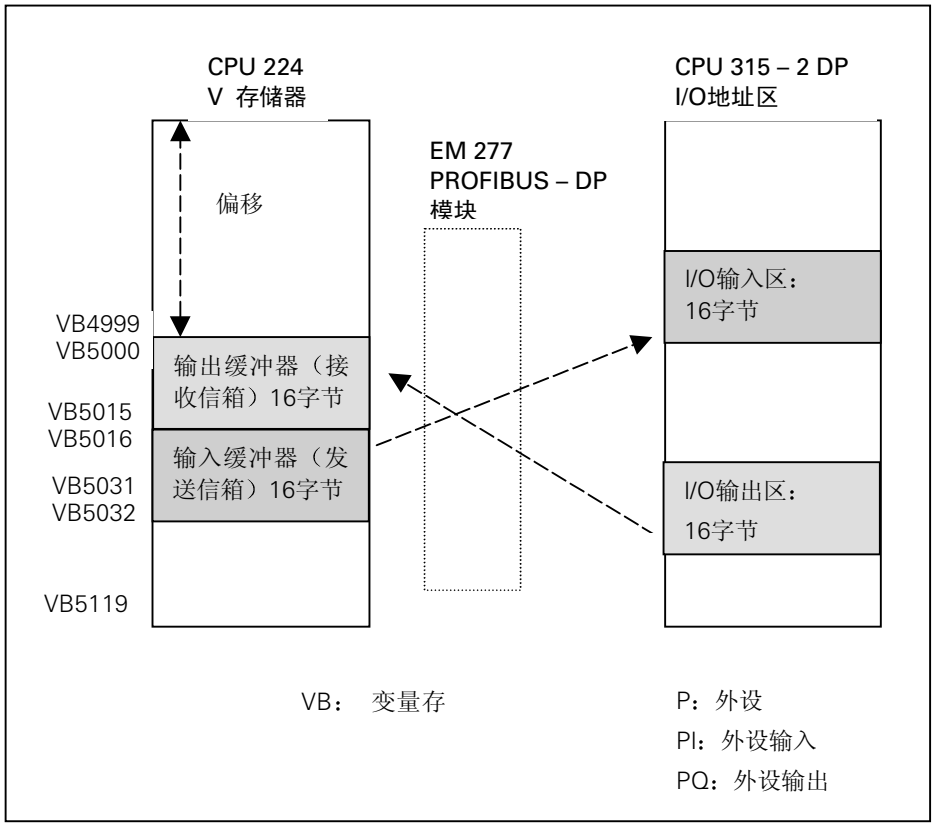
主站通过将其输出区来的信息发送给从站的输出缓冲区（称为“接收信箱”），与其每个从站交换数据。从站将其输入缓冲区（称为发送信箱）的数据返回给主站的输入区，以响应从主站来的信息。

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配置确定输入和输入缓冲区的大小。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-29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。

注

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数手册。



图A-29 CPU 224 V存储器和PROFIBUS – DP I/O地址区的举例

表A-18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的2个字和输出的2个字。

表A-18　EM 277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	缓冲器一致性
18	8字	8字	
19	12字	12字	
20	16字	16字	

输入和输出缓冲器的地址可以配置在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。

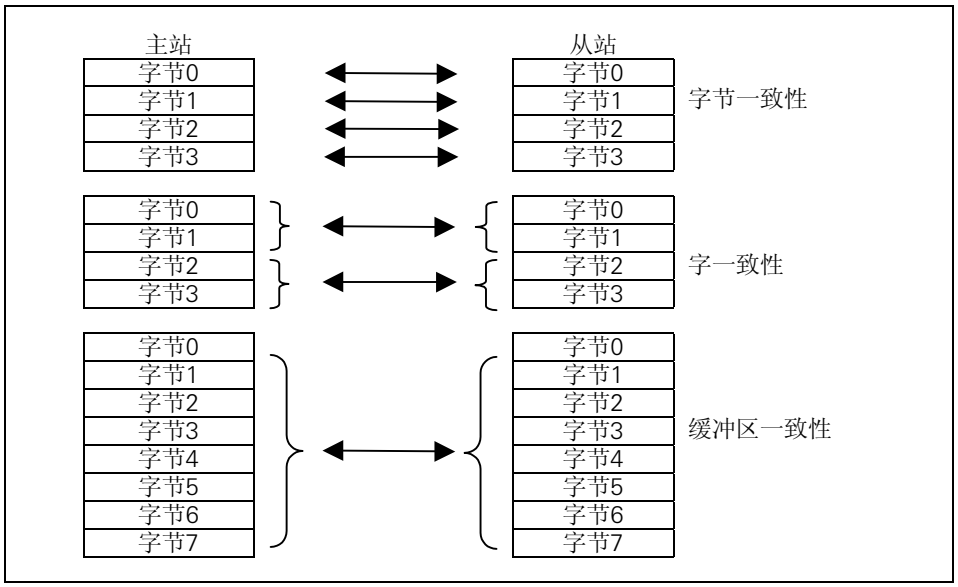
数据的一致性

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。

在CPU内处理或移动DP I/O数据时，字和缓冲区一致性迫使CPU停止任何其它处理（例如用户中断）。如果传送的数据值是整数，则必须采用字一致性。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。

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-30表示不同类型的一致性。



图A-30　字节，字和缓冲区数据的一致性。

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。

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。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存储器内。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。

SMB 200到SMB 249提供有关EM 277 PROFIBUS-DP从站模块的状态信息（如果它是I/O链中的第一个智能模块）。如果EM 277是I/O链中的第二个智能模块，那么，EM 277的状态是从SMB 250到SMB 299获得的。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主站的组态集。用户应检查SMB 224，并确保在使用SMB 225至SMB229或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-19）。

注

用户不能通过写入SMB 225到SMB229或SMB 275至SMB 279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-19 SMB 200至SMB 299的专用存储器字节

DP是第一个智能模块	DP是第二个智能模块	说明
SMB 200至SMB 215	SMB 250至 SMB 265	模块名（16 ASCII字符） “EM 277 Profibus DP”
SMB 216至SMB 219	SMB 266至 SMB 269	S/W 版本号（4 ASCII字符） x x x x
SMW 220	SMW 270	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 272	DP从模块的站地址，由地址开关（0-99+进制）设定
SMB 223	SMB 273	保留
SMB 224	SMB 274	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 275	DP标准协议-从站的主站地址（0至126）
SMW 226	SMW 276	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 278	DP标准协议-输出数据的字节数
SMB 229	SMB 279	DP标准协议-输入数据的字节数
SMB 230至SMB 249	SMB 280至 SMB 299	保留-电源接通时清除

注：每当DP从模块接收组态/参数化信息时更新SMB 225至SMB 229和SMB 275至SMB279。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-20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24 VDC用户电源	-	-	24 VDC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

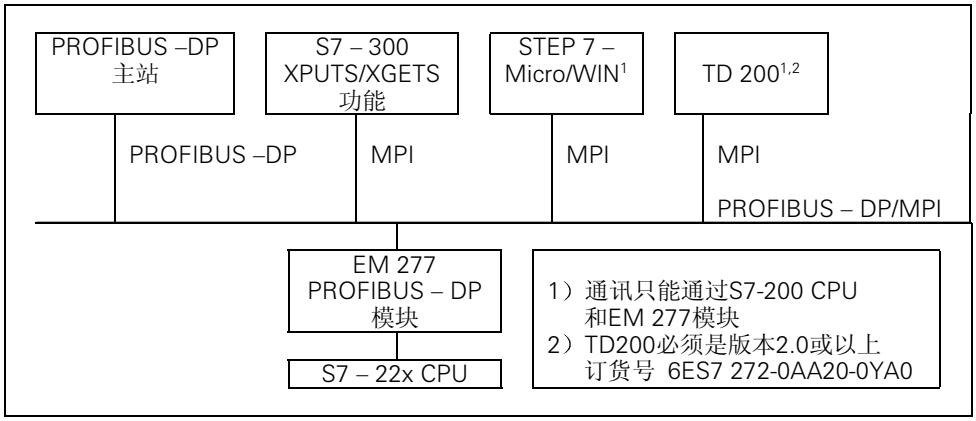
EM 277 PROFIBUS-DP模块可作为连接到其它MPI主站的通信接口，而不论该模块是否用作PROFIBUS-DP从站。该模块利用S7-300/400的XGET/XPUT功能，提供从S7-300/400到S7-200的连接。应用MPI或PROFIBUS参数集的STEP 7-Micro/WIN软件和一个网络卡（例如CP 5611），一个OP设备或TD 200（版本2.0或以上，订货号6ES7 272-0AA20-0YA0），可经过EM 277 PROFIBUS-DP模块，与S7-200进行通信。

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-31的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-31说明PROFIBUS-DP MPI网络。



图A-31 PROFIBUS-DP/MPI网络

设备数据库文件：GSD

不同的PROFIUS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。

EM 277 PROFIBUS-DP GSD文件的列表

```
;=============================================================
; 有一个DPC31的EM 277 PROFIBUS-DP的GSD文件
; MLFB：6ES7 277-0AA20-0XA0
; DATE：07-Oct-1999
;
;=============================================================
#Profibus_DP
; General parameters
GSD_Revision            =1
Vendor_Name             = “Siemens”
Model_Name              = “EM 277 PROFIBUS-DP”
Revision                = “V1.00”
Ident_Number            =0x089D
Protocol_Ident          =0
Station_Type            =0
FMS_supp                =0
Hardware_Release        = “1.00”
Software_Release        = “1.00”
9.6_supp                =1
19.2_supp               =1
45.45_supp              =1
93.75_supp              =1
187.5_supp              =1
500 _ supp              =1
```

```
1.5M _ supp                    = 1
3M _ supp                      = 1
6M _ supp                      = 1
12M _ supp                     = 1
Maxtsdr _ 9.6                  = 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
24 V _ Pins                    = 2
；slave – Specification：
OrderNumber =  “6ES7 277 – 0AA20 – 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”
Unsigned 1600 – 5119
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
；Module Definition List
Module =  “2 Bytes Out/ 2 Bytes In -  ” 0x31
EndModule
Module =  “8 Bytes Out/ 8 Bytes In -  ” 0x37
```

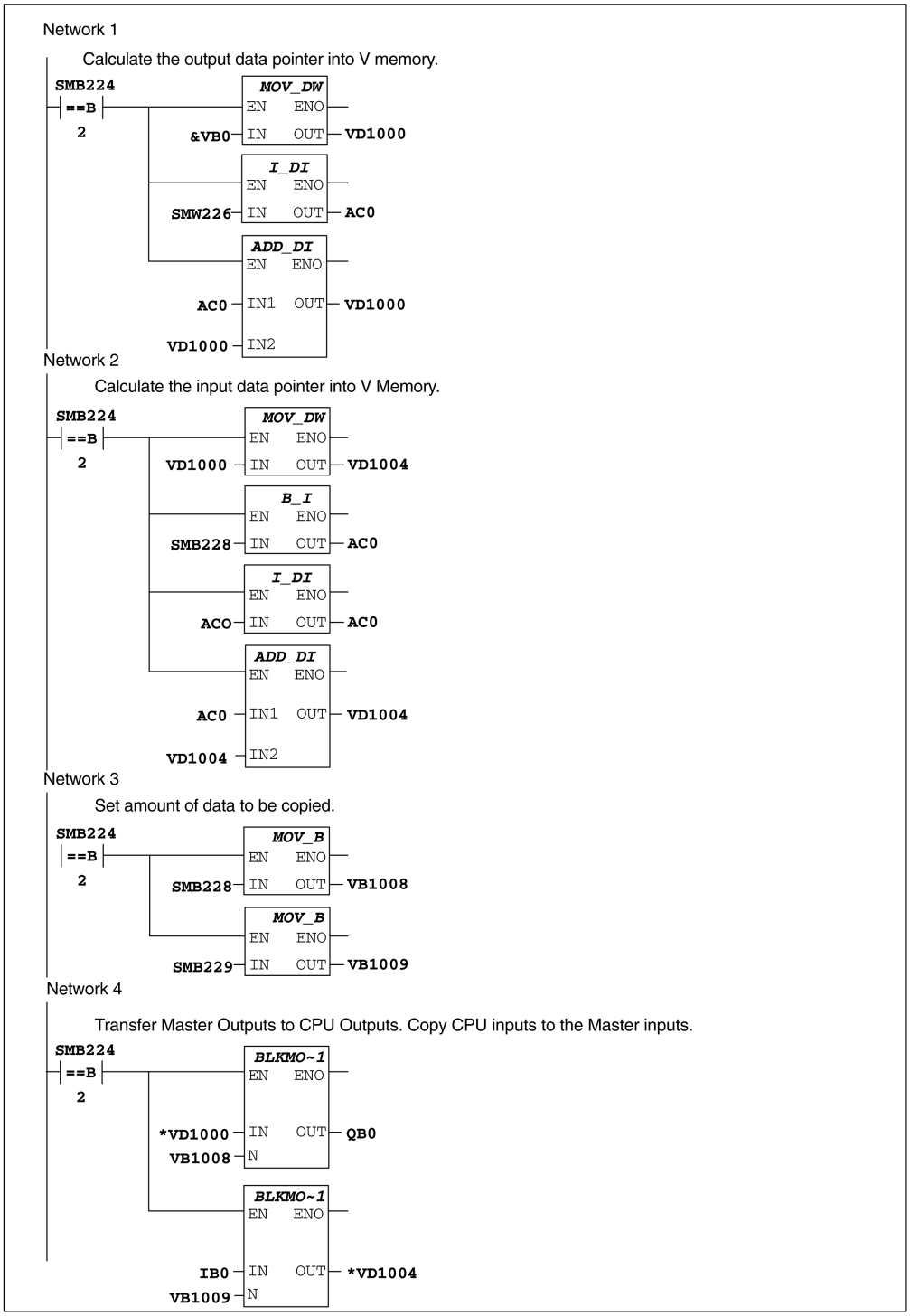
```
EndModule
Module = “32 Bytes Out/ 32 Bytes In - ” 0xC0，0x1F，0x1F
EndModule
Module = “64 Bytes Out/ 64 Bytes In - ” 0xC0，0x3F，0x3F
EndModule
Module = “1 Word   Out/ 1 Word In - ” 0x70
EndModule
Module = “2 Word   Out/ 2 Word In - ” 0x71
EndModule
Module = “4 Word   Out/ 4 Word In - ” 0x73
EndModule
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
```

用于到CPU 224的DP通信的示样程序

图A-32是用语句表生成的 CPU 224例子程序，它使用在SM存储器中的DP端口的信息。图A-33为相同的程序，但使用梯形图编程。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。

```
// 示样DP程序
// DP从站的组态信息
// 程序使用了以下数据
//
// SMW220        DP模块出错状态
// SMB 224       DP状态
// SMB 225       主站地址
// SMW 226       V 存储器中输出的偏移
// SMB 228       输出数据的字节数
// SMB 229       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
//
网络1
//
// 计算到V存储器的输出数据的指针
//
 LDB =    SMB 224，2                  // 是否(处在数据交换模式)
MOVD      & VB0，VD 1000              // 输出缓冲区从VB0开始的偏移
ITD       SMW 226，AC0                // 加上Vmem的偏移以得到输出缓冲区的偏移量
+D        AC0，VD 1000                //

网络2
//
// 计算到V存储器的输入数据的指针
//
LDB =     SMB 224，2                  // 是否（处在数据交换模式）
MOVD      VD 1000，VD 1004            // 得到输出指针地址
BTI       SMB 228，AC0                // 将输出的字节数加上输出
ITD       AC0，AC0                    // 指针，得到起始的输入指针
+D        AC0，VD 1004                //

网络3
//
// 设定要复制数据的数量
//
LDB =     SMB 224，2                  // 是否（处在数据交换模式）
MOVB      SMB 228，VB 1008            // 得到复制的输出字节的数量
MOVB      SMB 229，VB1009             // 得到拷复制的输入字节的数量

网络4
//
// 传送主站输出数据到CPU的输出，复制CPU的输入
// 到主站的输入
//
LDB =     SMB 224，2                  // 是否（处在数据交换模式）
BMB       *VD 100，QB0，VB 1008       // 复制主站输出到CPU输出
BMB       IB0，*VD 1004，VB 1009      // 复制CPU输入到主站输入
```

图A-32　用于到CPU 224的DP通信的STL示样程序



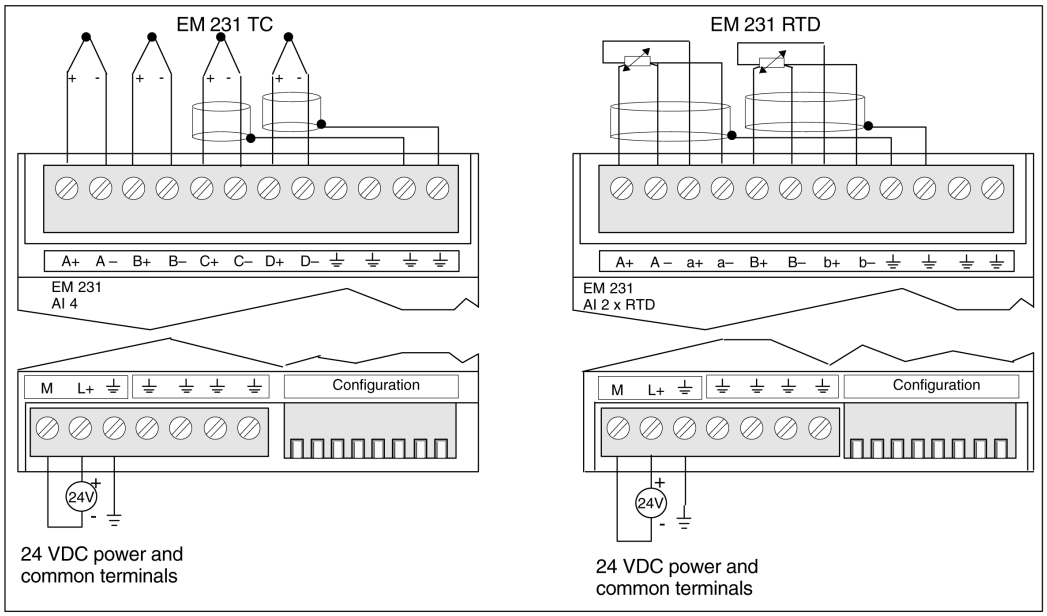
图A-33　到CPU 224的DP通信的梯形图示样程序

A．13　EM 231热电偶，EM 231 热电阻模块的技术规范

表A-21　EM 231 热电偶和EM 231 热电阻模块的技术规范

说明 订货号	EM 231 AI 4×热电偶 6ES7 231-7PD20-0XA0	EM 231 AI 2×　热电阻 6ES7 231-7PB20-0XA0
通用技术规范		
尺寸（W×H×D） 重量 功率损失（耗散）	71.2mm×80mm×62mm 210g 1.8W	71.2mm×80m×62mm 210g 1.8W
物理 I/O 点数	4模拟量输入点	2模拟量输入点
功耗 从+5VDC（从I/O总线） 从L+ L+电压范围，级2或DC传感器供电	 87mA 60mA 20.4到 28.8 VDC	 87mA 60mA 20.4到28.8 VDC
LED指示器	24V DC电源指示灯；ON=无故障，OFF=没有24 VDC电源，SF故障指示灯；ON=模块故障，SF闪烁=输入信号故障，OFF=无故障	24V DC电源指示灯；ON=无故障，OFF=没有24 VDC电源，SF故障指示灯；ON=模块故障，SF闪烁=输入信号故障，OFF=无故障
模拟量输入技术规范		
隔离 现场侧到逻辑 现场侧到24 V DC 24 V DC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	悬浮型热电隅	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，3920 ppm，3850.55 ppm，3916 ppm，3902 ppm） Pt-10000Ω(α=3850 ppm) Cu-9.035Ω(α=4720 ppm) Ni-10Ω,120Ω,1000Ω (α为 6720 ppm,6178 ppm) R-150Ω，300Ω，600ΩFS
输入分辨率 温度 电压 电阻	 0.1° C/0.1° F 15位加符号位	 0.1° C/0.1° F 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 @ 50Hz/60Hz/400Hz	85 dB @ 50Hz/60Hz/400Hz
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 @ 21 kHz	-3 dB @ 3.6 kHz

说明 订货号	EM 231 AI 4×热电偶 6ES7 231-7PD20-0XA0	EM 231 AI 2× 热电阻 6ES7 231-7PB20-0XA0
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℃	



图A-34　EM 231热电偶和热电阻模块的端子接线图

兼容性

热电阻和热电偶模块设计成与S7-200 CPU 222，CPU 224，和CPU 226一起工作。

这些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，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 热电阻和热电偶模块要安装在环境温度稳定的地方。

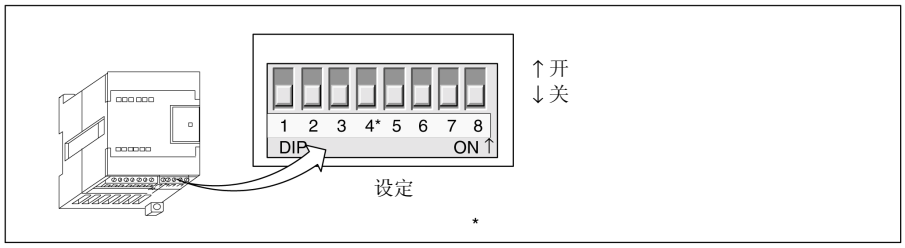
EM 231热电偶模块

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。用户必须利用DIP开关选择热电偶类型，断线检测，温度测量单位，冷端补偿，以及传感器熔断方向。所有连接到该模块的热电偶都必须是同一类型的。

组态EM 231热电偶模块

组态DIP开关位于模块的底部，如图A-35所示，为了使DIP的设置生效，您必须对PLC或用户24 V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-22到表A-26。



图A-35　组态DIP开关，用于EM 231热电偶模块

选择热电偶类型　通过DIP开关1，2，3选择热电偶类型，见表A-22。

表A-22　选择热电偶类型

热电偶类型	SW1	SW2	SW3
J（缺省值）	0	0	0
K	0	0	1
T	0	1	0
E	0	1	1
R	1	0	0
S	1	0	1
N	1	1	0
+/- 80mV	1	1	1

选择传感器熔断方向　通过DIP开关5与选择传感器熔断方向（正向标定或负向标定），见表A-23。

表A-23　选择传感器熔断方向

熔断方向	SW5
正向标定（+3276.7度）	0
负向标定（-3276.8度）	1

选择断线检测　通过注入 25uA 电流到输入端进行断线检测，断线检测使能开关可以启动或禁止检测电流。断线检测始终在进行，即使关闭了检测电流。如输入信号超过±200mV ，则EM 231热电偶模块开始断线检测，如检查到断线，则测量读数被设定成由熔断方向所设定的相应标定值（即+32767度或-32768度）。为了实现表A-24所要求的功能，可通过设置DIP开关6以启动和禁止检查断线的电流。

表A-24　选择导线开路检查

断线	SW6
启动断线检测电流	0
禁止断线检测电流	1

注

- 断线检测电流源可能干扰某些低电平信号源，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。

选择温度测量单位 EM 231热电偶模块可以测量摄氏温度值或华氏温度值，在模块内部进行摄氏温度值到华氏温度值的转换，使用DIP开关7选择温度测量单位，见表A-25。

表A-25 选择温度标定

测量单位	SW7
摄氏（℃）	0
华氏（℉）	1

选择冷端补偿 使用热电偶必须进行冷端补偿，如没有启用冷端补偿，则模块的转换会出现错误，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。使用DIP开关8启用或禁止冷端补偿，见表A-26。

表A-26 选择冷端运行

冷端启用	SW8
冷端补偿启用	0
冷端补偿禁止	1

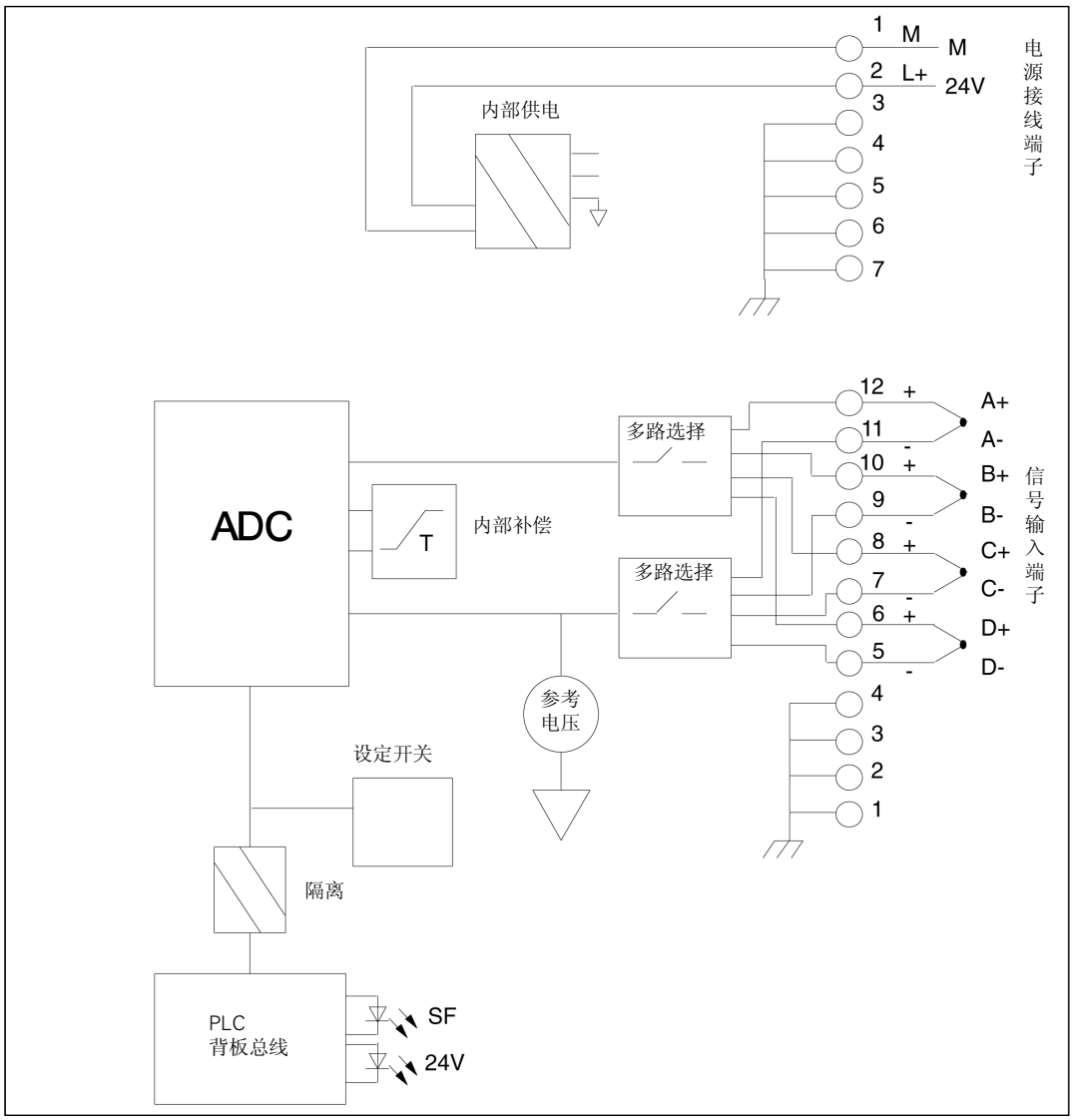
注

- 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
- 超过规范所规定的模块温度范围时，有可能导致模块冷端补偿出错

EM 231 热电偶模块的接线

直接将热电偶接线到EM 231热电偶模块。为了达到最佳的抗干扰性能应使用屏蔽线，如使用屏蔽线，应将屏蔽层连接到模块上1到4号接地端子。这些接地端子和电源侧的接地端子3到7是相同的。如果有一个热电偶的输入通道没有使用，必须将这个不用的输入短路或将其并行连接到另一个通道，这样就避免了由于悬浮输入所造成的有效输入通道阻塞错误。

您必须将用户电源连接到电源连接器的端子1和2，必须将电源连接器的端子3连接到邻近的机柜地，见图A-36。



图A-36　热电偶模块的线路图

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-25。

表A-27　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	*

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，SMB 11等，参阅附录C）

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

注

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。

注

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，模拟量滤波可防碍出错条件的检测。

热电偶的基本知识

任何二种金属，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，二根不同的金属导线连接到模块的输入信号接线端子。二根不同金属导线彼此连接处形成传感器的热电偶。在二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二个热电偶。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隅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。

热电偶模块测量范围

表A-28和表A-29表示每种热电偶的温度范围和准确度。

表A-28　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	
:	:								
-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-29　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	
10 进制	16进制								
32767		>2192.0℉	>2502.0℉	>752.0℉	>1832.0℉	>3214.0℉	>2372. 0℉	>94.071mV	
↑	↑					↑		↑	↑
32511	7EFF							94.071mV	OR
27649	6C01					3214.0℉			
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℉			↑				
:	:			↑					
18320	4790				1832.0℉				
:	:			7520℉					
7520	1D60					752.0℉			NR
:	:								
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		
:	:		低于范围						
-4270	EF52			-427.0℉	-427.0℉				
:	:			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℉	<-94.07℉	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 ）。

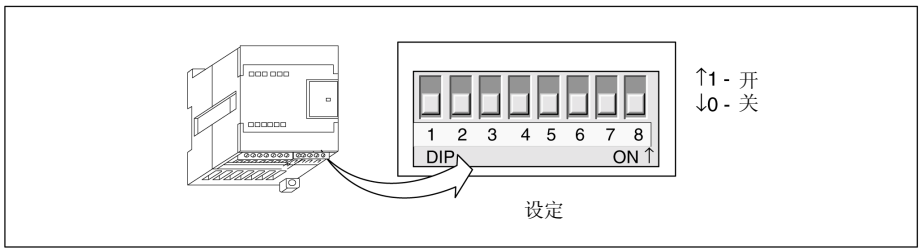
↓ 表示所有小于该值但大于断线阀值的模拟量都报告为下溢出值，-32768（0x8000）。

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，必须使用DIP开关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。连接到模块的热电阻必须是相同的类型。

配置EM 231 热电阻模块

DIP配置开关位于模块的底部，见图A-32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-37　配置DIP开关用于热电阻模块

选择热电阻类型　通过设定对应于热电阻的DIP开关1，2，3，4和5以选择热电阻类型，见表A-30。

表A-30　选择热电阻的类型

热电阻类型	SW1	SW2	SW3	SW4	SW5
100 Ω Pt0.003850（缺省值）	0	0	0	0	0
200 Ω Pt0.003850	0	0	0	0	1
500 Ω Pt0.003850	0	0	0	1	0
1000 Ω Pt0.003850	0	0	0	1	1
100 Ω Pt0.003920	0	0	1	0	0
200 Ω Pt0.003920	0	0	1	0	1
500 Ω Pt0.003920	0	0	1	1	0
1000 Ω Pt0.003920	0	0	1	1	1
100 Ω Pt0.00385055	0	1	0	0	0
200 Ω Pt0.00385055	0	1	0	0	1
500 Ω Pt0.00385055	0	1	0	1	0
1000 Ω Pt0.00385055	0	1	0	1	1
100 Ω Pt0.003916	0	1	1	0	0
200 Ω Pt0.003916	0	1	1	0	1
500 Ω Pt0.003916	0	1	1	1	0
1000 Ω Pt0.003916	0	1	1	1	1
100 Ω Pt0.00302	1	0	0	0	0
200 Ω Pt0.003902	1	0	0	0	1
500 Ω Pt0.003902	1	0	0	1	0
1000 Ω Pt0.003902	1	0	0	1	1
（备用）	1	0	1	0	0
100 Ω Ni0.00672	1	0	1	0	1
120 Ω Ni0.00672	1	0	1	1	0
1000 Ω Ni0.00672	1	0	1	1	1
100 Ω Ni0.006178	1	1	0	0	0
120 Ω Ni0.006178	1	1	0	0	1
1000 Ω Ni0.006178	1	1	0	1	0
10000 Ω Pt0.003850	1	1	0	1	1
10 Ω Cu 0.004270	1	1	1	0	0
150 Ω FS(电阻)	1	1	1	0	1
300 Ω FS(电阻)	1	1	1	1	0
600 Ω FS(电阻)	1	1	1	1	1

选择传感器熔断方向 使用开关6设置传感器熔断方向，见表A-31。

表A-31 选择传感器熔断方向

熔断传感器的极性	SW6
正向标定（+3276.7度）	0
负向标定（-3276.7度）	1

选择温度测量单位 热电阻模块能测量摄氏温度值或华氏温度值，在模块内部进行摄氏到华氏的转换。使用开关7选择温度测量单位，见表A-32。

表A-32 选择温度测量单位

测量单位	SW7
摄氏℃	0
华氏℉	1

选择接线方式 使用开关8选择接线方式，见表A-33。

表A-33 接线方式

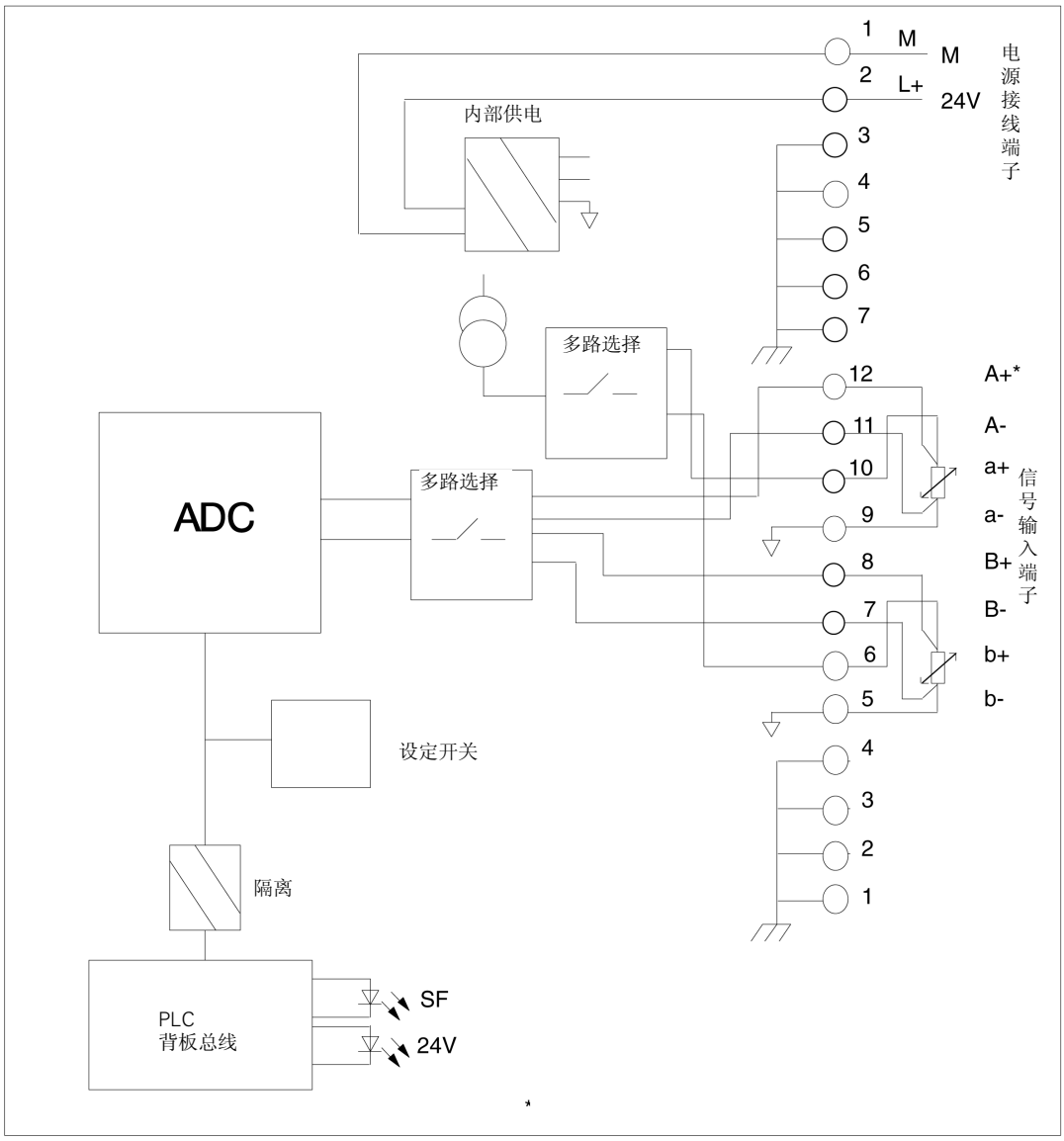
接线方式	SW8
3线	0
2线或4线	1

EM 231 热电阻模块的接线

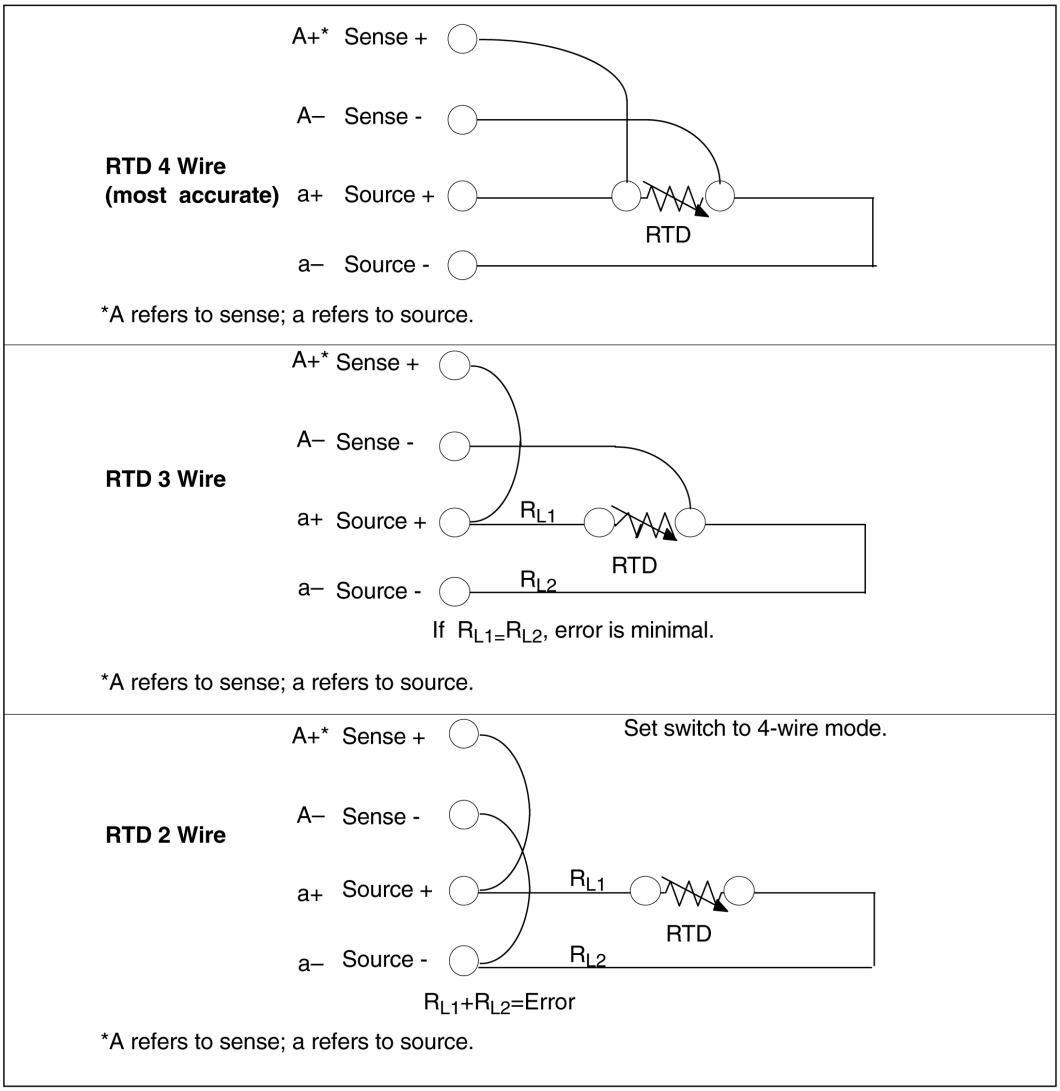
可以直接将EM 231 热电阻模块连接到S7-200模块，或使用扩展线，为了达到最佳的抗干扰性能应使用屏蔽线。如您使用屏蔽线，应将屏蔽层连接到模块上1到4号接地端子，这些接地端子和电源侧的接地端子3-7号是相同的。如果有一个热电阻输入通道没有使用，应在没有使用的输入通道接入一个电阻器，这样就避免由于悬浮输入所造成的有效输入通道阻塞的错误。

连接用户电源到电源连接器端子1和2，必须将电源连接器的端子3连接到邻近机柜的地（见图A-37）。

将热电阻模块接线到传感器有三种方法（见图A-38），其中精度最高的是4-线连接（见图A-39）。精度最低的是2-线连接，2-线连接只推荐用于可忽略接线误差的应用场合。



图A-38　热电阻模块内部电路结构



图A-39　热电阻到传感器的接线，4线，3线和2线方式。

EM231 热电阻状态指示器

热电阻模块提供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-34说明由EM 231热电阻模块所提供的状态指示器。

注

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（=225Ω/330Ω*27648）

表A-34 EM 231 热电阻状态指示器

出错类型	通道数据	SF 指示灯	24V 指示灯	范围状态位[1]	24伏模块故障状态位[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	*

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. 故障状态位是模块出错寄存器字节中的位2（SMB9，SMB11等。参阅附录C）
3. 诊断出错引起模块组态错误。在模块组态错误之前，模块故障状态位可能设置或没有设置。

如PLC已读到数据，则有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。

注

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，模拟量滤波可防碍出错条件的检测。

断线检测是由软件在模块同部完成的。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_35　用于热电阻类型的温度范围（℃）和精度

系统字(1个数字位)=0.1℃		Pt10000	Pt100,Pt200,Pt500,Pt1000	Ni100, Ni120, Ni1000	Cu9.035	0-150Ω	0-300Ω	0-160Ω	
10进制	16进制								
32767	7FFF								
32766	7FFE					↑	↑	↑	
32511	7EFF					176.383Ω	352.767Ω	705.534Ω	
29649	6C01					150.005Ω	300.011Ω	600.022Ω	
27648	6C00								
25000	61AB					150.000Ω	300.000Ω	600.000Ω	
18000	4650								↑
15000	3A98								OR
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℃	0.005Ω	0.011Ω	0.022Ω	
0	0000	0.0℃	0.0℃	0.0℃	0.0℃	0.000Ω	0.000Ω	0.000Ω	
-1	FFFF	-0.1℃	-0.1℃	-0.1℃	-0.1℃	负值是不可能			
						↓	↓	↓	
-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℃	±0.15Ω	±0.3Ω	±0.6Ω	

*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）

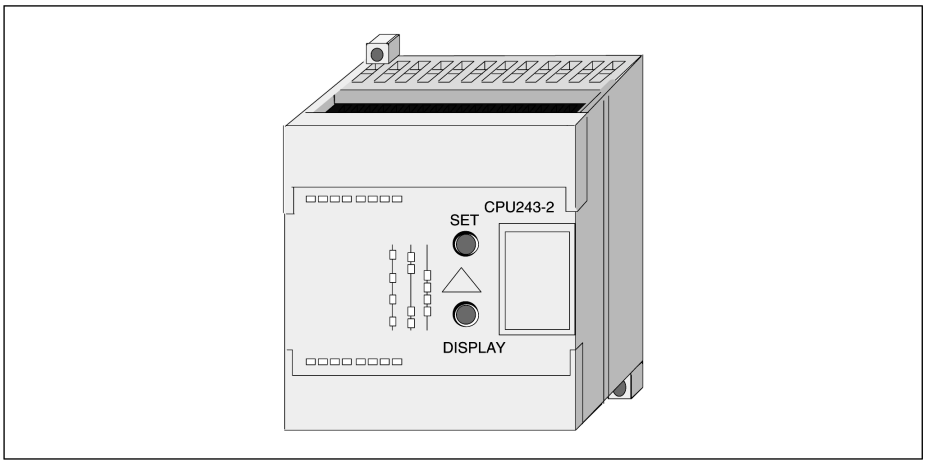
表A-36　用于热电阻类型的温度范围（℉）

系统字（1个数字位=0.1℉）		PT1000	PT100,Pt200,Pt500,Pt1000	Ni100,Ni120,Ni1000	Cu 9.035	
10进制	16进制					
32767	7FFF					
32766	7FFE					↑
						超出范围
		↑	↑			
18320	4790	1832.0℉	1832.0℉			
15620	3D04		1562.0℉			
11120	2B70	1112.0℉				
					↑	
5936	1730			↑	593.6℉	
5630	15FE			563.0℉		
5000	1388				500.0℉	
4820	12D4			482.0℉		
						额定范围
1	0001	0.1℉	0.1℉	0.1℉	0.1℉	
0	0000	0.0℉	0.0℉	0.0℉	0.0℉	
-1	FFFF	-0.1℉	-0.1℉	-0.1℉	-0.1℉	
-760	FD08			-76.0℉		
-1570	F9DE			-157.0℉		
				↓		
-3280	F330	-328.0℉	-328.0℉		-328.0℉	
-400	F060				-400.0℉	
-4054	F02A	-405.4℉	-405.4℉		↓	
		↓	↓			
-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）

A.14 CP 243-2通信处理器

说明 订货号	CP 243-2通信处理器 6GK7 243-2AX00-0XA0
AS-接口主站行规	M0/M1
接口 - 在PLC中的地址区位置 - 连接到AS-接口	 对应于2个I/O模块(DI/8 DO8和AI8/AQ8) 端子连接
功耗 - 通过AS-接口 - 通过背板总线	 100 mA，最大 220 mA，DC 5 V，典型
功率损失	约2W
允许的环境条件 - 运行温度 水平安装 垂直安装 - 运输/储存温度 - 相对温度	 0℃到+55℃ 0℃到+45℃ -40℃到+70℃ 95%；25℃
结构 - 模块格式 - 尺寸（W×H×D），mm - 重量	 扩展模块 71.2×80×62 约250g



图A-40 CP 243-2通信处理器

- 用前面板上的LED显示连接的从站的运行状态和就绪状态。
- 用前面板上的LED显示出错状态（包括AS-接口电压错误，组态错误）。
- 创新的SIMATIC S7-200新一代产品紧凑的外壳设计。

应用

CP 243-2是AS-接口主站连接部件，专门设计用于S7-200 CPU 22X。与AS-接口的连接显著地增加了S7-200可利用的输入和输出数（每个CP在AS-接口上最大为124数字输入/124数字输出）。在S7-200上，最多可同时运行二个CP 243-2。

设计

CP 243-2和扩展模块相同的方式连接到S7-200。它有：

- 二个接线端子，用于AS-接口电缆的直接连接。
- 前面板上的发光二极管，用来显示所有连接的和激活的从站状态及其就绪状态。
- 二个按钮，用来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SET组态接受。

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占用二个逻辑模块位置。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-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-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-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-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-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。

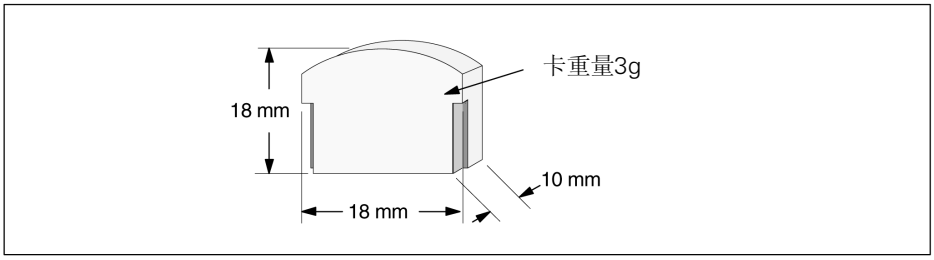
为工业应用而设计

- 由于显著地增加了可利用的数字和模拟量输入/输出，使SIMATIC S7-200更为灵活，更能适应于各种不同的应用。
- 由于能使用按钮进行组态，从而缩短了准备时间。
- 在偶然出现故障时，由于发光二极管能显示以下状态从而缩短了停机和维修时间：
  - CP状态
  - 显示所有连接的从站及其就绪状态
  - 监视AS-接口的通讯电源电压

A.15 可选扩展卡

订货号	
6ES7 291 8GE20 0XA0	用户程序卡
6ES7 291 1AA20 0XA0	时钟卡（带电池）
6ES7 291 8BA20 0XA0	电池卡

卡的技术参数	
存储器卡存储	程序、数据和组态结果
存储器卡（保存时间）	典型值：200天
时钟存储器精度	2 分钟/月，25℃ 7 分钟/月，0℃~55℃



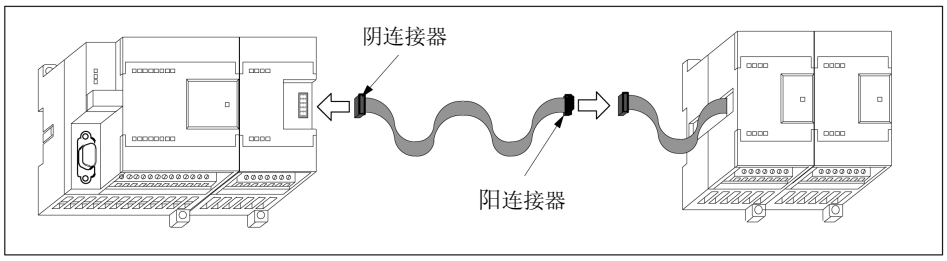
总体特性	
电池	3V,30 mA小时，Renata CR 1025
外形尺寸	9.9×2.5mm
型号	锂电池（<0.6g）
典型寿命	10年

A.16 I/O 扩展电缆

订货号：6ES7 290-6AA20-0XA0

总体特性	
电缆长度	0.8m (32英寸)
重量	25g
连接器型号	10 芯扁平线

I/O 扩展电缆的典型安装

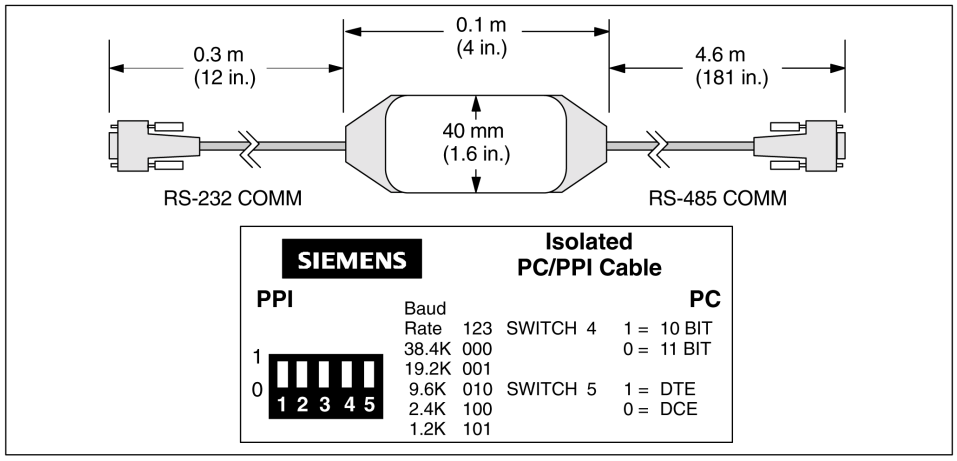


注：
在CPU和扩展模块的连接中只能使用一条扩展电缆

A．17　PC/PPI编程电缆

订货号：　6ES7 901-3BF20-0XA0

总体特性	
电源电压	14.4 to 28.8 VDC
24V电源的电流	50 mA 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 s max.
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
延时	4 μ s max., RS-485 to RS-232, 1.2 μ s max., RS-232 to RS-485
隔离	500 VDC
RS-485一侧的电气特性	
共模电压范围	-7V to +12V, 1 second 3V RMS
接收器输入阻抗	5.4K Ω　min
终端	10K Ω　to +5V on B,PROFIBUS pin 3 10K Ω　to GND on A,PROFIBUS pin 8
接收器阈值/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 minimum
接收器阈值/灵敏度	0.8 V min.low,2.4V max,high, 0.5 V typical hysteresis
发送器输出电压。	+/- 5V min @ R_L =3K Ω



图A-42　PC/PPI 电缆尺寸

表A-37　PC/PPI电缆的波特率选择开关

波特率	开（1=上）
38400	000
19200	001
9600	010
4800	011
2400	100
1200	101
600	110

表A-38　调制解调器的使用

调制解调器使用	开关（1=上）
10-位调制解调器	1
11-位调制解调器	0

表A-39　PC/PPI电缆引脚

Pinout	开关（1=上）
DCE	0
DTE	1

表A-40 RS485 RS232 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	24 V 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	+5 V（带 100Ω 串联电阻）	6	数据设置就绪（DSR）（不用）
7	24 V 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-41　RS485与RS232 DTE连接器的引脚

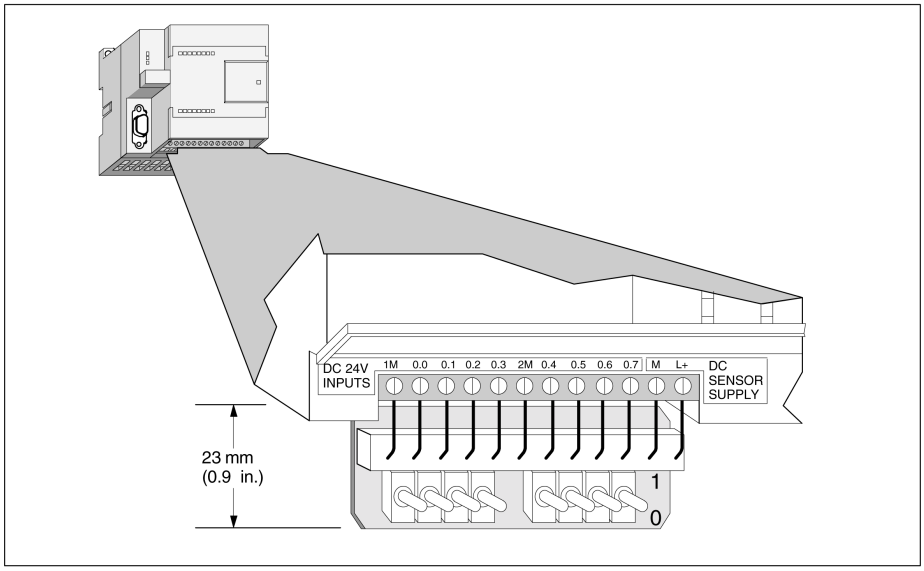
RS-485		RS-232	
针号	RS-485 连接器引针	针号	RS-232 DTE 连接器引针[1]
1	地（RS-485 逻辑地）	1	数据载波检测（DCD）（不用）
2	24 V 返回（RS-485 逻辑地）	2	接收数据（RD）
3		3	（从PC/PPI 电缆输出）
4	信号 B （RxD/TxD+）	4	发送数据（TD）
5		5	（输入到 PC/PPI）
	RTS(TTL 电平)		数据终端就绪（DTR）
			（不用）
	地（RS-485 逻辑地）		地（RS-232 逻辑地）
6	+5 V（带 100Ω 串联电阻）	6	数据设置就绪（DSR）（不用）
7	24 V 电源	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-阳型9到25针的转换

A．18　输入仿真器

表A-40

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 (3.6×1.4×0.85in.)
重量	0.02Kg(0.04lb.)	0.03 Kg(0,08lb.)	0.04Kg(0,08lb.)
点数	8	14	24



图A-43　输入仿真器的安装

注

这些输入仿真器不能用于Class 1 DIV 2或Class 1 Zone 2危险环境。

附录 B：错误代码

B

有关错误代码的信息有助于查找 S7-200 CPU 模块的问题。

本章概述

B.1 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。表 B-1 列出了从 CPU 上可读到的致命错误代码 及其描述。

表 B-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	用户程序检查和错误
0002	编译后的梯形图程序检查和错误
0003	扫描看门狗超时错误
0004	内部 EEPROM 错误
0005	内部 EEPROM 用户程序检查错误
0006	内部 EEPROM 配置参数检查错误
0007	内部 EEPROM 强制数据检查错误
0008	内部 EEPROM 缺省输出表值检查错误
0009	内部 EEPROM 用户数据、DB1 检查错误
000A	存储器卡失灵
000B	存储器卡上用户程序检查和错误
000C	存储器卡配置参数检查和错误
000D	存储器卡强制数据检查和错误
000E	存储器卡缺省输出表值检查和错误
000F	存储器卡用户数据、DB1 检查和错误
0010	内部软件错误
0011	比较接点间接寻址错误
0012	比较接点非法值错误
0013	存储器卡空，或者 CPU 不识别该卡

B.2 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 B-2 列出了这些非致命错误代码及其描述。

表 B-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

B.3 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 B-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 B-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：你必须把一个网络分成多个网络
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 指令或删条 NEXT 指令
0086	无 NEXT：加条 NEXT 指令，或删条 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	保留
008C	标号重复 (LBLNINT，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	保留
0098	在运行模式进行非法编辑
0099	隐含程序网络太多

附录 C：特殊存储器 (SM) 标志位

C

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。

SMB0: 状态位

如表 C-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 C-1 特殊存储器字节 SMB0 (SM0.0 - SM0.7)

SM 位	描 述
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 C-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 (置 1) 或复位 (置 0)。

表 C-2 特殊存储器字节 SMB1 (SM1.0 - SM1.7)

SM 位	描 述
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 C-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。

表 C-3 特殊存储器字节 SMB2

SM 位	描 述
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

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 C-4 所示，根据该位来废弃错误消息。

表 C-4 特殊存储器字节 SMB3 (SM3.0 - SM3.7)

SM 位	描 述
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C-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 C-5 特殊存储器字节 SMB4 (SM4.0 - SM4.7)

SM 位	描 述
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 C-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 C-6 特殊存储器字节 SMB5 (SM5.0 - SM5.7)

SM 位	描 述
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 - SM5.6	保留
SM5.7	当 DP 标准总线出现错误时，将该位置 1

SMB6: CPU 识别 (ID) 寄存器

如表 C-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 C-7 特殊存储器字节 SMB6

SM 位	描 述
格式	MSB 7 … LSB 0 x x x x r r r r CPU ID register
SM6.4 – SM6.7	xxxx = 0000 = CPU 212/CPU 222 0010 = CPU 214/CPU 224 0110 = CPU 221 1000 = CPU 215 1001 = CPU 216/CPU 226
SM6.0 – SM6.3	保留

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 C-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 C-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 7 … LSB 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 IO 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 32 DI QQ 00 无输出 01 2 AI 或 8 DI 10 4 AI 或 16 DI 11 8 AI 或 32 DI 奇数字节：模块错误寄存器 MSB 7 … LSB 0 C | 0 | 0 | b | r | p | f | t C: 配置错误 b: 总线错误或校验错误 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 C-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 C-9 特殊存储器字 SMW22 到 SMW26

SM 字	描 述
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 C-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 C-10 特殊存储器字节 SMB28 和 SMB29

SM 字节	描 述
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 C-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 C-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
SM30.6 和 SM30.7	SM130.6 和 SM130.7	Pp 校验选择 00 = 不校验 01 = 奇校验 10 = 不校验 11 = 偶校验
SM30.5	SM130.5	d 每个字符的数据位 0 = 8 位/ 字符 1 = 7 位/ 字符
SM30.2 到 SM30.4	SM130.2 到 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 = 600 波特 111 = 300 波特
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。

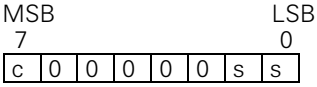
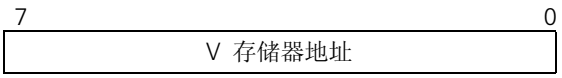
SMB31 和 SMW32: 永久存储器 (EEPROM) 写控制

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 C-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 C-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: 软件命令  SMB32: 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，由CPU复位
SMW32	SMW32 提供 V 存储器中被存数据相对于 V0 的偏移地址，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 C-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C-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 C-14 所示，SMB36 到 SM65 用于监视和控制高速计数器 HSC0、HSC1和 HSC2 的操作。

表C-14　　特殊存储器字节 SMB36 到 SMB65

SM 字节	描　　述
SM36.0 到 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 = 4 X 速率，1 = 1 X 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写新的当前值
SM37.7	HSC0 有效位：1＝ 有效
SMB38 SMB39 SMB40 SMB41	HSC0 新的当前值 SMB38 是最高有效字节，SMB41 是最低有效字节
SMB42 SMB43 SMB44 SMB45	HSC0 新的预置值 SMB42 是最高有效字节，SMB45 是最低有效字节
SM46.0 到 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=4× 速率，1=1× 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 写新的当前值
SM47.7	HSC1 有效位：1= 有效
SMB48 SMB49 SMB50 SMB51	HSC1 新的当前值 SMB48 是最高有效字节，SMB51 是最低有效字节
SMB52 到 SMB55	HSC1 新的预设值 SMB52 是最高有效字节，SMB55 是最低有效字节
SM56.0 到 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

表 C-14　特殊存储器字节 SMB36 到 SMB65

SM 字节	描　　述
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=4× 速率，1=1× 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 写新的当前值
SM57.7	HSC2 有效位：1=有效
SMB58 SMB59 SMB60 SMB61	HSC2 新的当前值 SMB58 是最高有效字节，SMB61 是最低有效字节
SMB62 SMB63 SMB64 SMB65	HSC2 新的预设值 SMB62 是最高有效字节，SMB65 是最低有效字节

SMB66 到 SMB85: PTO/PWM 寄存器

如表 C-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。见第九章的 9.5 节高速输出指令关于这些位的完整描述。

表 C-15　　特殊存储器字节 SMB66 到 SMB85

SM 字节	描　　述
SM66.0 到 SM66.3	保留
SM66.4	PTO 0 包络溢出：0= 无溢出，1= 有溢出 (由于增量计算错误)
SM66.5	PTO 0 包络溢出：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= 1 µs，1= 1 ms
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
SMB68 SMB69	PTO 0/PWM 0 周期 (2 ~ 65,535 时间基准) ； SMB68 是最高有效字节，SMB69 是最低有效字节
SMB70 SMB71	PWM 0 脉冲宽度值 (0 ~ 65,535 时间基准) ； SMB70 是最高有效字节，SMB71 是最低有效字节
SMB72 SMB73 SMB74 SMB75	PTO 0 脉冲计数值 (1 ~ 2^{32} -1) ； SMB72 是最高有效字节，SMB75 是最低有效字节
SM76.0 到 SM76.3	保留
SM76.4	PTO1 包络益处; 0= 无溢出，1= 有溢出 (由于增量计算错误)
SM76.5	PTO1 包络益处; 0 = 通过用户命令不溢出, 1 = 通过用户命令溢出
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 = 1 µs/tick，1 = 1 ms/tick
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
SMB78 SMB79	PTO1/PWM1 周期值 (2 到 65,535 时间基准) ； SMB78 是最高有效字节，SMB79 是最低有效字节
SMB80 SMB81	PWM1 脉冲宽度值 (0 到 65,535 时间基准) ； SMB80 是最高有效字节，SMB81 是最低有效字节
SMB82 SMB83 SMB84 SMB85	PTO1 脉冲计数值 (1 到 2^{32} -1) ； SMB82 是最高有效字节，SMB85 是最低有效字节

SMB86 到 SMB94, SMB186 到 SMB194: 接收信息控制

如表 C-16 所示，SMB86 到 SMB94 和 SMB186 到 SMB194 用于控制和读出接收信息指令得状态。

表C-16　　特殊存储器字节 SMB86 到 SMB94，SMB186 到 SMB194

口 0	口 1	描　　述
SMB86	SMB186	MSB 7 … LSB 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 7 … LSB 0 en sc ec il c/m tmr bk 0　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 SMB90 或 SMB190 1=使用 SMB90 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 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 is a don't care 3.中断检测:　il=0,sc=1,bk=1,SMW90 is a don't care 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 is a don't care 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

表 C-16　　特殊存储器字节 SMB86 到 SMB94，SMB186 到 SMB194

口 0	口 1	描　　述
SMB89	SMB189	信息字符的结束
SMB90 SMB91	SMB190 SMB191	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，SM90 (或 SM190) 是最高有效字节，SM91 (或 SM191) 是最低有效字节。
SMB92 SMB93	SMB192 SMB193	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。SM92 (或 SM192) 是最高有效字节，SM93 (或 SM193) 是最低有效字节。
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。

SMB98 和 SMB99

如表 C-17 所示，SMB98 和 SMB99 给出有关扩展模块总线的错误号。

表 C-17　　特殊存储器字节 SMB98 和 SMB99

SM 字节	描　　述
SMB98 SMB99	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。 SMB98 是最高有效字节。

SMB131 到 SMB165: HSC3, HSC4 和 HSC5 寄存器

如表 C-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 C-18　　特殊存储器字节 SMB130 到 SMB165

SM 字节	描　　述
SMB131 到 SMB135	保留
SM136.0 到 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 到 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 写新的当前值
SM137.7	HSC3 有效位：1 = 有效
SM138 到 SM141	HSC3 新当前值：SM138 是最高有效字节，SM141 是最低有效字节
SM142 到 SM145	HSC3 新预设值：SM142 是最高有效字节，SM145 是最低有效字节
SM146.0 到 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

表 C-18　　特殊存储器字节 SMB130 到 SMB165

SM 字节	描　　述
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 写新的当前值
SM147.7	HSC4 有效位：1 = 有效
SMB148 到 SMB151	HSC4 新当前值：SM148 是最高有效字节，SM151 是最低有效字节
SMB152 到 SMB155	HSC4 预设值：SM152 是最高有效字节，SM155 是最低有效字节
SM156.0 到 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 到 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写新的当前值
SM157.7	HSC5 有效位：1 = 有效
SMB158 到 SMB161	HSC5 新当前值：SM158 是最高有效字节，SM161 是最低有效字节
SMB162 到 SMB165	HSC5 预设值：SM162 是最高有效字节，SM165 是最低有效字节

SMB166 到 SMB194: PTO0, PT1 包络定义表

如表 C-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 C-19 特殊存储器字节 SMB166 到 SMB194

SM 字节	描 述
SMB166	PTO0的包络步当前计数值
SMB167	保留
SMB168 SMB169	PTO0的包络表V 存储器地址 (从V0开始的偏移量)，SM168 是地址偏移量 的最高有效字节
SMB170 到 SMB175	保留
SMB176	PTO1的包络步当前计数值
SMB177	保留
SMB178 到 SMB179	PTO1的包络表V 存储器地址 (从V0开始的偏移量)，SM178 是地址偏移量 的最高有效字节
SMB180 到 SMB194	保留

SMB200 到 SMB 299：智能模块状态

SMB200 到 SMB 299预留给智能扩展模块的状态信息，例如 EM 277 PROFIBUS-DP模块。SMB200 到 SMB 249 预留给系统的第一个扩展模块(离CPU最近的模块)；SMB250 到 SMB 299 预留给第二个智能模块。参见附录 A 可得到模块如何使用 SMB 200 到 SMB 299 的信息。

附录 D：S7-200 故障处理指南

D

表 D-1　S7-200 故障处理指南

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 • 输出被强置	• 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 2.4 节。 • 修改程序 • 检查接线，如果不正确，要改正 • 检查输出的负载率 • 检查 CPU 是否有被强置的 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 • 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 B.1节： • 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 2.3 节的接线指南。控制盘良好接地 和高电压与低电压不并行引线是很重要的。 — 把 24 VDC 传感器电源的 M 端子接到地。
电源损坏	电源线引入过电压	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 2.3 节。
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 2.3 节的接线指南。控制盘良好接地和高电压与低电压不并行引线是很重要的。把 24 VDC 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， 请参考 5.2 节。
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 2.3 节的接线指南和第 7 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–to–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 B

附录 E：S7-200 定货号

E

CPU	定 货 号
CPU 221 DC/DC/DC 6 输入 /4 输出	6ES7 211-0AA21-0XB0
CPU 221 AC/DC/继电器 6 输入 /4 输出	6ES7 211-0BA21-0XB0
CPU 222 DC/DC/DC 8 输入 /6 输出	6ES7 212-1AB21-0XB0
CPU 222 AC/DC/继电器 8 输入 /6 输出	6ES7 212-1BB21-0XB0
CPU 224 DC/DC/DC 14 输入 /10 输出	6ES7 214-1AD21-0XB0
CPU 224 AC/DC/继电器 14 输入 /10 输出	6ES7 214-1BD21-0XB0
CPU 226 DC/DC/DC 24 输入 /16 输出	6ES7 216-2AD21-0XB0
CPU 226 AC/DC/继电器 24 输入 /16 输出	6ES7 216-2BD21-0XB0

扩 展 模 块	定 货 号
EM 221 24 VDC 数字 8 输入	6ES7 221-1BF20-0XA0
EM 222 24 VDC 数字 8 输出	6ES7 222-1BF20-0XA0
EM 222 继电器 8 输出	6ES7 222-1HF20-0XA0
EM 223 24 VDC 数字组合 4 输入 /4 输出	6ES7 223-1BF20-0XA0
EM 223 24 VDC 数字组合 4 输入 /4 继电器输出	6ES7 223-1HF20-0XA0
EM 223 24 VDC 数字组合 8 输入 /8 输出	6ES7 223-1BH20-0XA0
EM 223 24 VDC 数字组合 8 输入 /8 继电器输出	6ES7 223-1PH20-0XA0
EM 223 24 VDC 数字组合 16 输入 /16 输出	6ES7 223-1BL20-0XA0
EM 223 24 VDC 数字组合 16 输入 /16 继电器输出	6ES7 223-1PL20-0XA0
EM 231 24 VDC 模拟量输入，4 输入	6ES7 231-0HC20-0XA0
EM 231 24 VDC 模拟量输出，2 输出	6ES7 232-0HB20-0XA0
EM 235 24 VDC 模拟量组合，4 输入 /1 输出	6ES7 235-0KD20-0XA0
EM 231 24 VDC 模拟量输入 热电阻，2 输入	6ES7 231-7PB20-0XA0
EM 231 24 VDC 模拟量输入热电偶，4 输入	6ES7 231-7PD20-0XA0
EM 277 PROFIBUS-DP	6ES7 277-0AA20-0XA0
CP 243-2 通讯处理器	6GK7 243-2AX00-0XA0

卡 和 电 缆	定 货 号
MC 291, 32K×8 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,, 8 米, CPU 22x/EM	6ES7 290-6AA20-0XA0
电缆, PC/PPI, 隔离, 5– 开关, 5 米	6ES7 901-3BF20-0XA0

编 程 软 件	定 货 号
STEP 7-Micro/WIN 32 (V3.1) 单用户授权 (软盘)	6ES7 810-2BA01-0YX0
STEP 7-Micro/WIN 32 (V3.1) 升级授权 (软盘)	6ES7 810-2BA01-0YX3
STEP 7-Micro/WIN 32 (V3.1) 单用户授权 (CD–ROM)	6ES7 810-2BC01-0YX0
STEP 7-Micro/WIN 32 (V3.1) 升级授权 (CD–ROM)	6ES7 810-2BC01-0YX3
STEP 7-Micro/WIN 32 工具软件包单用户授权 (CD–ROM)	6ES7 810-2TC00-0YX0

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
S7–200 点到点接口通讯手册 (英语/德语)	6ES7 298-8GA00-8XH0
CP 243-2 通讯处理器手册 (英语)	6GK7 243-2AX00-8BA0
S7–200 可编程控制器系统手册 (英语)	6ES7 298-8FA21-8BH0

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	6XVI 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 (带编程接口)	6ES7 972-0BA40-0XA0
RS 485 总线连接器，35° 电缆出口 (不带编程接口)	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 503-1DB10
OP7 操作员界面	6AV3 607-1JC20-0AX1
OP17 操作员界面	6AV3 617-1JC20-0AX1
TP 070A 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：CPU 221/222 EM22x 12 端子块盖，CPU 224 18 端子块盖，EM 22x 7 端子块盖， CPU 盖， EM 盖	6ES7 291-3AX20-0XA0
8 位模拟器	6ES7 274 1XF00-0XA0
14 位模拟器	6ES7 274 1XH00-0XA0
24 位模拟器	6ES7 274 1XK00-0XA0

附录 F：STL 指令的执行时间

F

使能位对执行时间的影响

当使能位满足 (栈顶值为 ON 或 1) 时，一条 STL 指令的基本执行时间 (表 F-4) 为执行指令逻辑、或功能所需要的时间。对于某些指令，功能的执行视使能位存在与否：只有使能位满足 (栈顶值为 ON 或 1) 时，CPU 才会执行该功能。如果使能位不满足 (栈顶值为 OFF 或 0)，使用“无使能位”指令来计算程序执行时间。表 F-1 为无使能位 (栈顶值为 OFF 或 0) 的STL指令的执行时间，适用于每个 S7 200 CPU 模块。

表 F-1 无使能位的指令执行时间

无使能位的指令	S7–200 CPU
所有 STL 指令	3 µs

间接寻址对执行时间的影响

STL 指令的基本执行时间 (表 F-4) 为使用直接寻址时，执行该指令所需时间。如果你的程序使用间接寻址，在每条间接寻址指令上再附加如表 F-2所示的执行时间。

表 F-2 间接寻址的附加时间

间接寻址的指令	S7–200 CPU
每个间接寻址指令	22 µs

执行时间

访问一些存储区 (例如 AI、AQ、L和累加器) 需要附加执行时间。表 F-3 给出了访问每个这样的操作数时需要附加到基本执行时间的时间因子。

表 F-3 访问所选存储区时的附加执行时间

存 储 区	S7–200 CPU
模拟输入 (AI)	149 µs
模拟输出 (AQ)	73 µs
局部存储区 (L)	5.4 µs
累加器 (AC)	4.4 µs

STL 指令的基本执行时间

表 F-4 列出了所有 S7–200 CPU 模块的 STL 指令基本执行时间。

表 F-4 STL 指令的执行时间 (单位 µs)

指 令	描 述		S7–200 CPU (µs)
=	基本执行时间：	I L SM, T, C, V, S, Q, M	0.37 19.2 1.8
+D	基本执行时间：		55
-D	基本执行时间：		55
*D	基本执行时间：		92
/D	基本执行时间：		376
+I	基本执行时间：		46
-I	基本执行时间：		47
*I	基本执行时间：		71
/I	基本执行时间：		115
=I	基本执行时间：	本机输出 扩展模块输出	29 39
+R	基本执行时间： 最大执行时间		110 163
-R	基本执行时间： 最大执行时间		113 166
*R	基本执行时间： 最大执行时间		100 130
/R	基本执行时间： 最大执行时间		300 360
A	基本执行时间：	I L SM, T, C, V, S, Q, M	0.37 10.8 1.1
AB < =, =, >=, >, <, <>	基本执行时间：		35
AD < =, =, >=, >, <, <>	基本执行时间：		53
AI	基本执行时间：	本机输入 扩展模块输入	27 35
ALD	基本执行时间：		0.37
AN	基本执行时间：	I L SM, T, C, V, S, Q, M	0.37 10.8 1.1
ANDB	基本执行时间：		37
ANDD	基本执行时间：		55
ANDW	基本执行时间：		48
ANI	基本执行时间：	本机输入 扩展模块输入	27 35
AR <=, =, >=, >, <, <>	基本执行时间		54
ATCH	基本执行时间		20
ATH	总计 = 基本时间+ (长度) *(长度系数) 基本执行时间 (固定长度) 基本执行时间 (变长度) 长度系数 (LM)		 41 55 20
ATT	基本执行时间		70
AW < =, =, >=, >, <, <>	基本执行时间		45
BCDI	基本执行时间		66

表 F-4 STL 指令的执行时间 (单位 µs)

指 令	描 述	S7–200 CPU (µs)
BIR	基本执行时间： 本机输入	43
	扩展模块输入	51
BIW	基本执行时间： 本机输入	42
	扩展模块输入	52
BMB	总计 = 基本时间 + (长度) * (LM)	
	基本执行时间 (固定长度)	21
	基本执行时间 (变长度)	51
	长度系数 (LM)	11
BMD	总计 = 基本时间 + (长度) * (LM)	
	基本执行时间 (固定长度)	21
	基本执行时间 (变长度)	51
	长度系数 (LM)	11
BMW	总计 = 基本时间 + (长度) * (LM)	
	基本执行时间 (固定长度)	21
	基本执行时间 (变长度)	51
	长度系数 (LM)	16
CALL	无参数：	
	执行时间	15
	带参数：	
	总执行时间 =	
	基本时间 + Σ(输入操作数处理时间)	
	基本执行时间	32
	输入操作数处理时间 (位操作数)	23
	输入操作数处理时间 (字节操作数)	21
	输入操作数处理时间 (字操作数)	24
	输入操作数处理时间 (双字操作数)	27
COS	基本执行时间	1525
	最长执行时间	1800
CRET	总执行时间 =	
	基本时间 + Σ(输出操作数处理时间)	
	基本执行时间	13
	输出操作数处理时间 (位操作数)	21
	输出操作数处理时间 (字节操作数)	14
	输出操作数处理时间 (字操作数)	18
	输出操作数处理时间 (双字操作数)	20
CRETI	基本执行时间	23
CTD	基本执行时间：计数输入端的转换	48
	基本执行时间：其它	36
CTU	基本执行时间：计数输入端的转换	53
	基本执行时间：其它	35
CTUD	基本执行时间：计数输入端的转换	64
	基本执行时间：其它	45
DECB	基本执行时间	30
DECD	基本执行时间	42
DECO	基本执行时间	36
DECW	基本执行时间	37
DISI	基本执行时间	18
DIV	基本执行时间	119
DTCH	基本执行时间	18

表 F-4 STL 指令的执行时间 (单位 µs)

指 令	描 述	S7–200 CPU (µs)
DTR	基本执行时间 最大执行时间	60 70
ED	基本执行时间	15
ENCO	最小执行时间 最大执行时间	39 43
END	基本执行时间	0.9
ENI	基本执行时间	53
EU	基本执行时间	15
EXP	基本执行时间 最长执行时间	1170 1375
FIFO	总计 = 基本时间 + (LM) * (长度) 基本执行时间 长度系数 (LM)	 70 14
FILL	总计 = 基本时间 + (LM) *× (长度) 基本执行时间 (固定长度) 基本执行时间 (变长度) 长度系数 (LM)	 29 50 7
FND <, =, >, <>	总计 = 基本时间 + (LM) *× (长度) 基本执行时间 长度系数 (LM)	 85 12
FOR	总计 = 基本时间 + (LM) *× (重复次数) 基本执行时间 循环系数 (LM)	 64 50
GPA	基本执行时间	31
HDEF	基本执行时间	35
HSC	基本执行时间	37
HTA	总计 = 基本时间 + (LM) *× (长度) 基本执行时间 (固定长度) 基本执行时间 (变长度) 长度系数 (LM)	 38 48 11
IBCD	基本执行时间	114
INCB	基本执行时间	29
INCD	基本执行时间	42
INCW	基本执行时间	37
INT	1个中断的典型执行时间	47
INVB	基本执行时间	31
INVD	基本执行时间	42
INVW	基本执行时间	38
JMP	基本执行时间	0.9
LBL	基本执行时间	0.37
LD	基本执行时间: I L SM, T, C, V, S, Q, M SM0.0	0.37 10.9 1.1 0.37
LDB <=, =, >=, >, <, <>	基本执行时间	35
LDD <=, =, >=, >, <, <>	基本执行时间	52
LDI	基本执行时间: 本机输入 扩展模块输入	26 34
LDN	基本执行时间: I L SM, T, C, V, S, Q, M	0.37 10.9 1.1

表 F-4 STL 指令的执行时间 (单位 µs)

指 令	描 述	S7–200 CPU (µs)
LDNI	基本执行时间: 本机输入 扩展模块输入	26 34
LDR<=, =, >=, >, <, <>	基本执行时间	55
LDS	基本执行时间	0.37
LDW <=, =, >=, >, <, <>	基本执行时间	42
LIFO	基本执行时间	70
LN	基本执行时间 最长执行时间	1130 1275
LPP	基本执行时间	0.37
LPS	基本执行时间	0.37
LRD	基本执行时间	0.37
LSCR	基本执行时间	12
MEND	基本执行时间	0.5
MOVB	基本执行时间	29
MOVD	基本执行时间	38
MOVR	基本执行时间	38
MOVW	基本执行时间	34
MUL	基本执行时间	70
NEXT	基本执行时间	0
NETR	基本执行时间	179
NETW	总计 = 基本时间 + (LM) * (长度) 基本执行时间 长度系数 (LM)	 175 8
NOP	基本执行时间	0.37
NOT	基本执行时间	0.37
O	基本执行时间: I L SM, T, C, V, S, Q, M	0.37 10.8 1.1
OB < =, =, >=, >, <, <>	基本执行时间	35
OD < =, =, >=, >, <, <>	基本执行时间	53
OI	基本执行时间: 本机输入 扩展模块输入	27 35
OLD	基本执行时间	0.37
ON	基本执行时间: I L SM, T, C, V, S, Q, M	0.37 10.8 1.1
ONI	基本执行时间: 本机输入 扩展模块输入	27 35
OR<=, =, >=, >, <, <>	基本执行时间	55
ORB	基本执行时间	37
ORD	基本执行时间	55
ORW	基本执行时间	48
OW < =, =, >=, >, <, <>	基本执行时间	45
PID	基本执行时间 对 PID 重新计算 (K_c<T_s/T_i) 和 (K_c<T_d/T_s) 时的附加数。当指令上次执行后改变K_c，T_s，T_i 或 T_s 时，或变成自动控制时， 就会重新计算。	750 1000
PLS	基本执行时间: PWM PTO 单段 PTO 多段	 57 67 92

表 F-4 STL 指令的执行时间 (单位 μs)

指 令	描 述	S7–200 CPU (μs)
R	对长度=1 和常数 (例如： R V0.2,1) 执行时间：操作数 = C 执行时间：操作数 = T 执行时间：其它操作数 否则, 总执行时间=基本执行时间 + (LM) * (长度) 基本执行时间：操作数 = C, T 基本执行时间：其它操作数 长度系数 (LM) ：对操作数 = C 长度系数 (LM) ：对操作数 = T 长度系数 (LM) ：对其它操作数 如果长度存在变量中，而不是常数，则在基本执行时间 上加上：	 17 24 5 19 28 8.6 16.5 0.9 29
RCV	基本执行时间	80
RET	总执行时间= 基本时间 + Σ(输出操作数处理时间) 基本执行时间 输出操作数处理时间 (位操作数) 输出操作数处理时间 (字节操作数) 输出操作数处理时间 (字操作数) 输出操作数处理时间 (双字操作数)	 13 21 14 18 20
RETI	基本执行时间	23
RI	总计 = 基本时间 + (LM) * (长度) 基本执行时间 长度系数 (本机输出) 长度系数 (扩展模块输出) 如果长度存在变量中，而不是常数，则在基本执行时间上 加上：	 18 22 32 30
RLB	总计 = 基本时间 + (LM) * (长度) 基本执行时间 长度系数 (LM)	 42 0.6
RLD	总计 = 基本时间 + (LM) * (长度) 基本执行时间 长度系数 (LM)	 52 2.5
RLW	总计 = 基本时间 + (LM) * (长度) 基本执行时间 长度系数 (LM)	 49 1.7
RRB	总计 = 基本时间 + (LM) * (长度) 基本执行时间 长度系数 (LM)	 42 0.6
RRD	总计 = 基本时间 + (LM) * (长度) 基本执行时间 长度系数 (LM)	 52 2.5
RRW	总计 = 基本时间 + (LM) * (长度) 基本执行时间 长度系数 (LM)	 49 1.7

表 F-4 STL 指令的执行时间 (单位 μs)

指 令	描 述	S7–200 CPU (μs)
S	对长度=1 和常数 (例如： S V0.2,1) 执行时间 否则， 总执行时间=基本执行时间 + (LM) * (长度) 基本执行时间：对所有其它操作数 长度系数 (LM) ：对所有其它操作数 如果长度存在变量中，而不是常数，则在基本执行时间上 加上：	 5 27 0.9 29
SBR	基本执行时间	0
SCRE	基本执行时间	0.37
SCRT	基本执行时间	17
SEG	基本执行时间	30
SHRB	总计 = 基本时间 + (LM1)* (长度) + ((长度/8)* LM2) 基本执行时间 (常数长度) 基本执行时间 (变量长度) 长度系数 (LM1) 长度系数 (LM2)	 76 184 1.6 4
SI	总计 = 基本时间 + (LM) * (长度) 基本执行时间 长度系数 (LM) (本机输出) 长度系数 (LM) (扩展模块输出) 如果长度存在变量中，而不是常数，则在基本执行时间上 加上：	 18 22 32 30
SIN	基本执行时间 最长执行时间	1525 1800
SLB	总计 = 基本时间 + (LM) * (长度) 基本执行时间 长度系数 (LM)	 43 0.7
SLD	总计 = 基本时间 + (LM) * (长度) 基本执行时间 长度系数 (LM)	 53 2.6
SLW	总计 = 基本时间 + (LM) * (长度) 基本执行时间 长度系数 (LM)	 51 1.3
SPA	基本执行时间	243
SQRT	基本执行时间 最大执行时间	725 830
SRB	总计 = 基本时间 + (LM) < (长度) 基本执行时间 长度系数 (LM)	 43 0.7
SRD	总计 = 基本时间 + (LM) * (长度) 基本执行时间 长度系数 (LM)	 53 2.6
SRW	总计 = 基本时间 + (LM) * (长度) 基本执行时间 长度系数 (LM)	 51 1.3
STOP	基本执行时间	16
SWAP	基本执行时间	32
TAN	基本执行时间 最长执行时间	1825 2100
TODR	基本执行时间	2400

表 F-4 STL 指令的执行时间 (单位 µs)

指 令	描 述	S7–200 CPU (µs)
TODW	基本执行时间	1600
TOF	基本执行时间	64
TON	基本执行时间	64
TONR	基本执行时间	56
TRUNC	基本执行时间 最大执行时间	103 178
WDR	基本执行时间	16
XMT	基本执行时间	78
XORB	基本执行时间	37
XORD	基本执行时间	55
XORW	基本执行时间	48

附录G：S7-200 快速参考信息

G

本附录包含下面的内容：

- 特殊存储器位
- 中断事件描述
- S7-200 CPU 存储器范围和特性概述
- 高速计数器 HSC0，HSC1，HSC2，HSC3，HSC4，HSC5
- S7-200 指令

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到二进值转换出错
SM0.7	开关放置在RUN位置时为1	SM1.7	ASCII到十六进值转换出错

表 G-2 中断时间描述

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：接收字符	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/Z 相		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/Z 相		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 S7-200 CPU 存储器范围和特性汇总

描述	范围				存取格式			
	CPU 221	CPU 222	CPU 224	CPU 226	位	字节	字	双字
用户程序区	2K字	2K字	4K字	4K字				
用户数据区	1K字	1K字	2.5K字	2.5K字				
输入映像寄存器	I0.0-I15.7	I0.0-I15.7	10.0-I15.7	10.0-I15.7	Ix.y	IBx	IWx	IDx
输出映像寄存器	Q0.0-Q15.7	Q0.0-Q15.7	Q0.0-Q15.7	Q0.0-Q15.7	Qx.y	QBx	QWx	QDx
模拟输出 （只读）	-	AIW0-AIW30	AIW0-AIW30	AIW0-AIW30			AIWx	
模拟输出 （只写）	-	AQW0-AQW30	AIW0-AQW30	AIW0-AQW30			AQWx	
变量存储器（V）[1]	VB0.0- VB2047.7	VB0.0- VB2047.7	VB0.0- VB5119.7	VB0.0- VB5119.7	Vx.y	VBx	VWx	VDx
局部存储器（L）[2]	LB0.0-LB63.7	LB0.0- LB63.7	LB0.0-LB63.7	LB0.0-LB63.7	Lx.Y	LBx	LWx	LDx
位存储器（SM）	M0.0-M31.7	M0.0-M31.7	M0.0-M31.7	M0.0-M31.7	Mx.Y	MBx	MWx	MDx
特殊存储器（SM） 只读	SM0.0-SM179.7 SM0.0-SM29.7	SM0.0-SM179.7 SM0.0-SM29.7	SM0.0-SM179.7 SM0.0-SM29.7	SM0.0-SM179.7 SM0.0-SM29.7	SMx.y	SMBx	SMW x	SMD x
定时器 保持接通延时 1ms 保持接通延时 10 ms 保持接通延时 100 ms 接通/断开延时 1 ms 接通/断开延时 10 ms 接通/断开延时 100 ms	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，T36， 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	Tx		Tx	
计数器	C0-C255	C0-C255	C0-C255	C0-C255	Cx		Cx	
高速计数器	HC0，HC3 HC4，HC5	HC0，HC3 HC4，HC5	HC0-HC5	HC0-HC5				HCx
顺控继电器（S）	S0.0-S31.7	S0.0-S31.7	S0.0-S31.7	S0.0-S31.7	Sx.y	SBx	SWx	SDx
累加器	AC0-AC3	AC0-AC3	AC0-AC3	AC0-AC3		ACx	ACx	Acx
跳转/标号	0-255	0-255	0-255	0-255				
调用/子程序	0-63	0-63	0-63	0-63				
中断程序	0-127	0-127	0-127	0-127				
PID回路 通讯口	0-7 0	0-7 0	0-7 0	0-7 0				

1所有V存储器可以保存在永久存储器中
2LB60到LB63为STEP 7-Micro/WIN 32 V3.0或更高版本保留

表 G-4 高速计数器HSC0，HSC3，HSC4，HSC5

模式	HSC0			HSC3	HSC4			HSC5
	10.0	10.1	10.2	10.1	10.3	10.4	10.5	10.4
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				HSC4			
	10.6	10.7	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	立即复位一个区域

数学、增减指令		
+I	IN1，OUT	整数、双整数或实数加法 IN1+OUT=OUT
+D	IN1，OUT	
+R	IN1，OUT	
-I	IN1，OUT	整数、双整数或实数减法 OUT-IN1=OUT
-D	IN1，OUT	
-R	IN1，OUT	
MUL	IN1，OUT	整数或实数乘法 IN1*OUT=OUT
*R	IN1，OUT	
***D, *I**	IN1，OUT	整数或双整数乘法
DIV	IN1，OUT	整数或实数除法 IN1/OUT=OUT
/R	IN1，OUT	
/D,/I	IN1，OUT	整数或双整数除法
SQRT	IN，OUT	平方根
LN	IN，OUT	自然对数
EXP	IN，OUT	自然指数
SIN	IN，OUT	正弦
COS	IN，OUT	余弦
TAN	IN，OUT	正切
INCB	OUT	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
LBL	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、转换和结束
SCRT	N	
SCRE		

传送、移位、循环和填充指令		
MOVB MOVW MOVD MOVR BIR BIW	OUT OUT OUT OUT ZN，OUT ZN，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）

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	Convert Byte to Integer Convert Integer to Byte 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进值格式 把16进值格式转换成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ASCII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

附录H：S7-200 应用示例

H

本章概述

H.1 模拟电位器

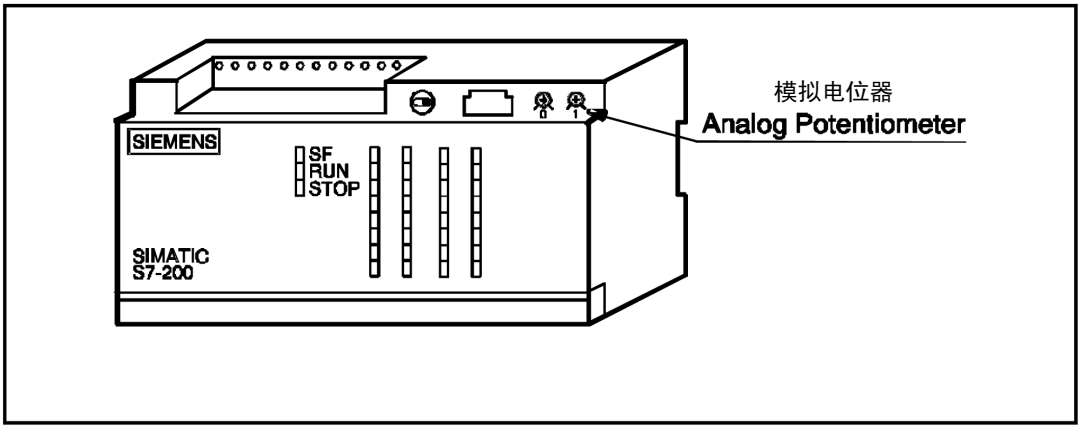
概述

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。

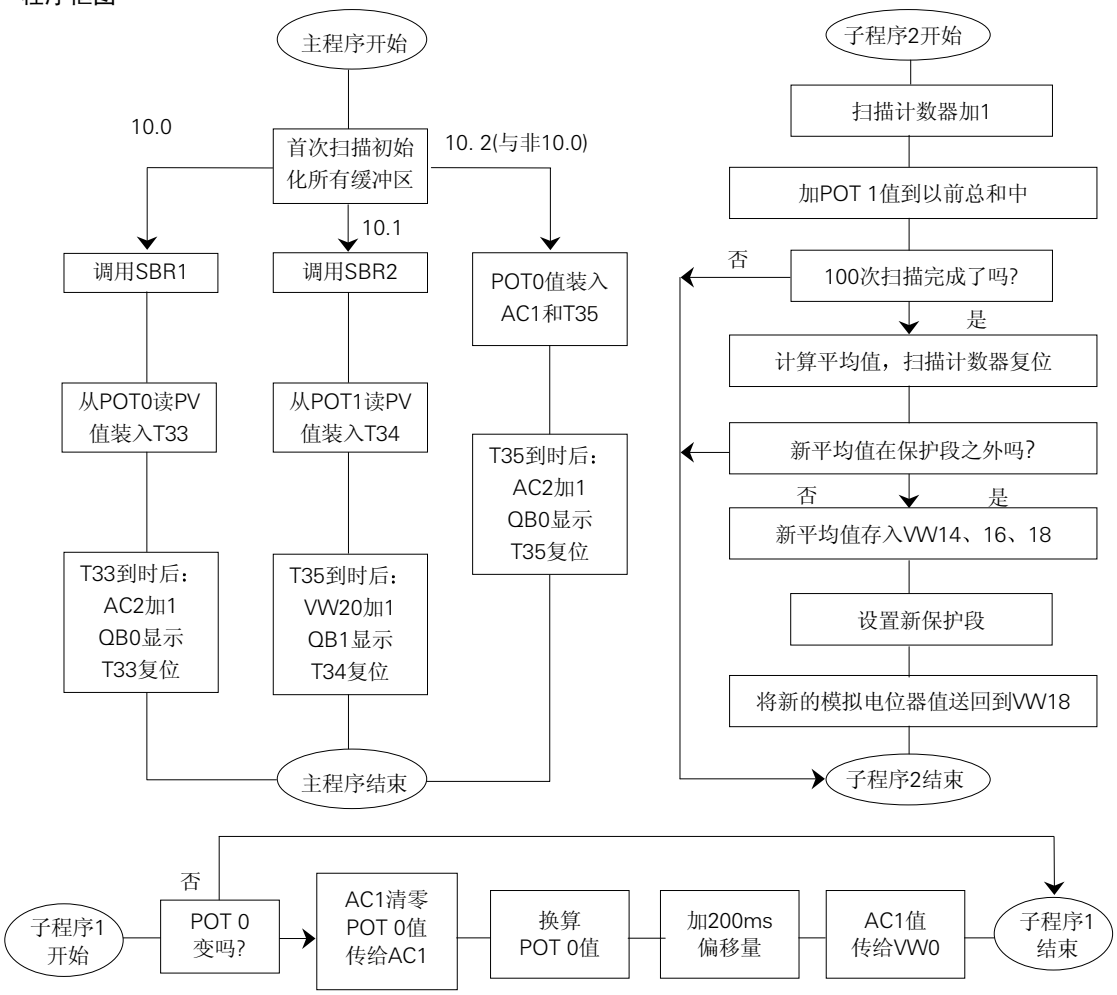
需要一把小螺丝刀用以调整电位器的位置。

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。
+1      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。
+1      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。
-1      +3，VW14          // 设置新的低保护限。
+1      +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

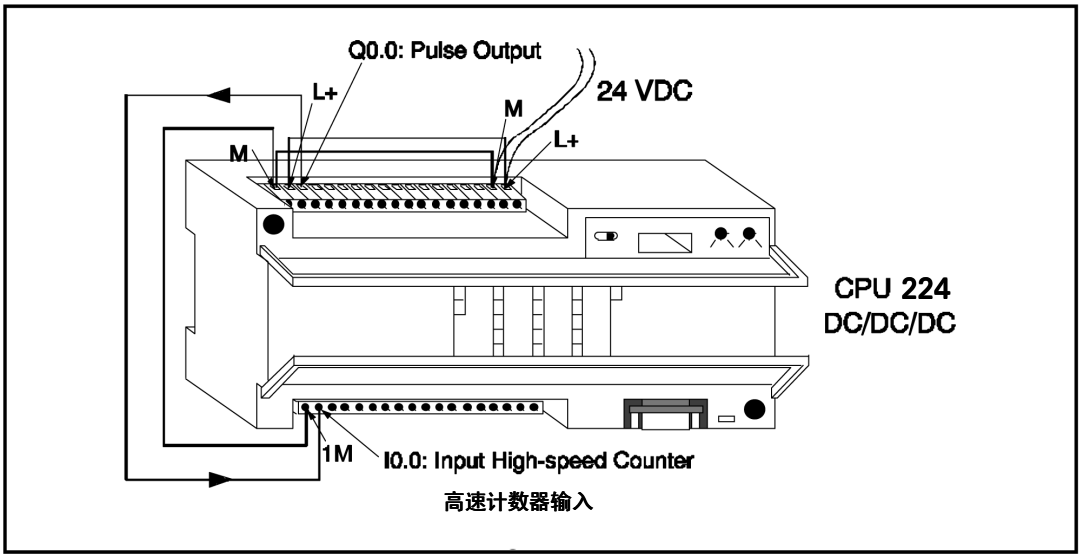
概述

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。

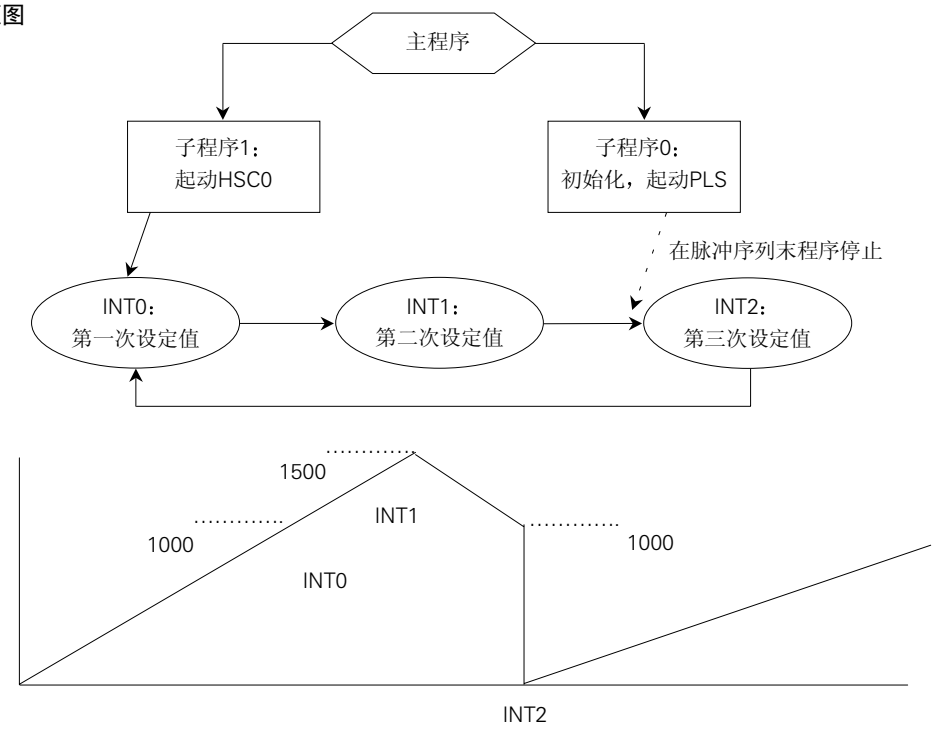
本例用脉冲输出（PLS）来为HSC产生高速计数信号，PLS可以产生脉冲串和脉宽调制信号，例如用来控制伺服电机。既然利用脉冲输出，必须选用CPU 224DC/DC/DC。

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

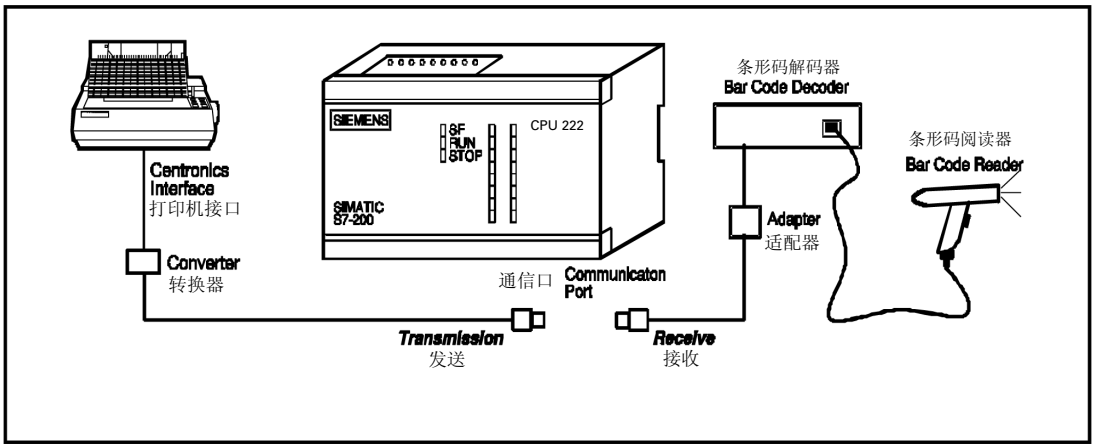
概述

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：

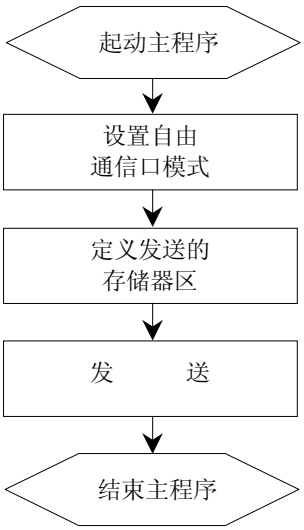
- 奇偶校验
- 每个字符的位数
- 波特率

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



机程序框图



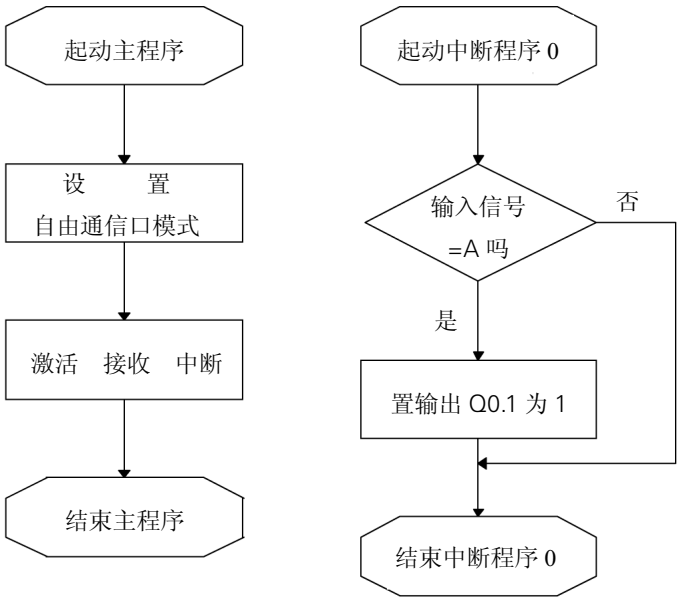
打印机程序和注解

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

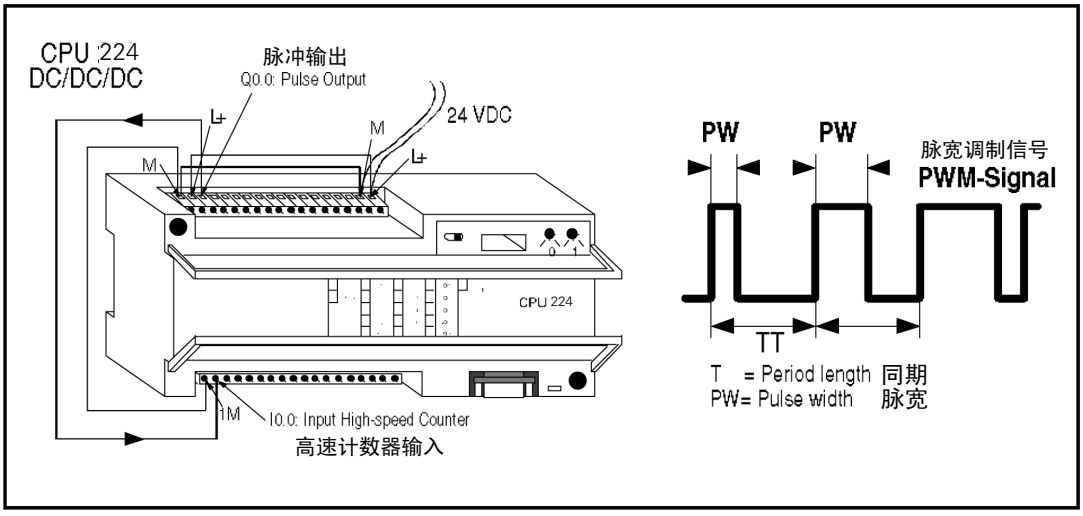
概述

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。

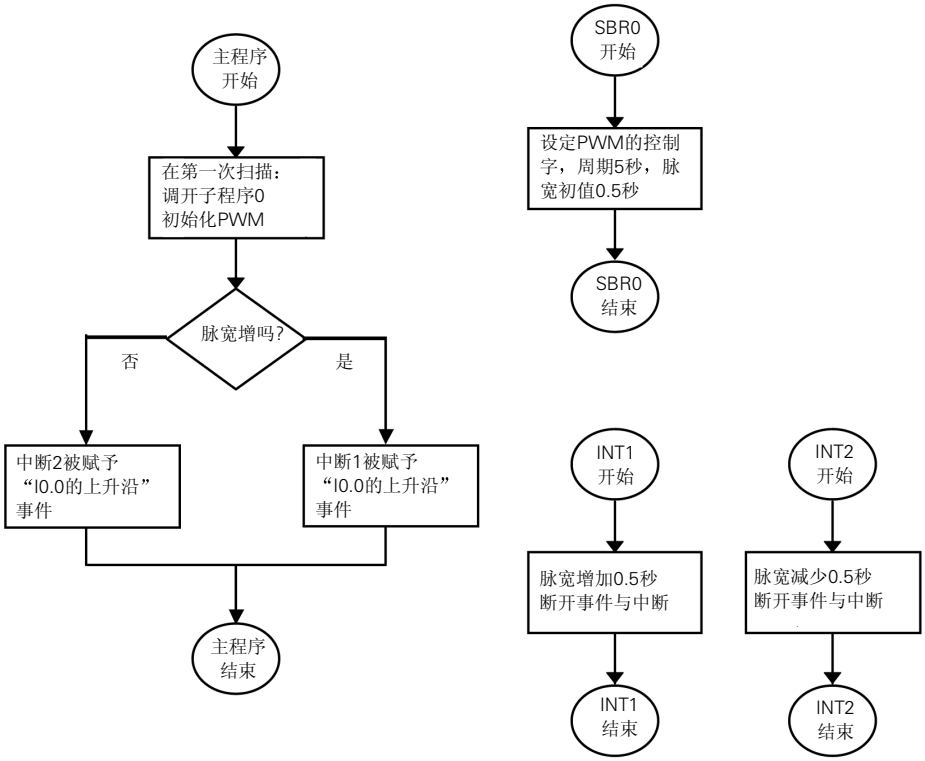
下面这个例子说明了脉宽调制（PWM）是如何工作的。输出端Q0.0输出方波信号，其脉宽每周期递增0.5秒，周期固定为5秒，并且脉宽的初始值为0.5秒。当脉宽达到设定的最大值4.5秒时，脉宽改为每周期递减0.5秒，直到脉宽为零为止。以上过程周而复始。

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。
```

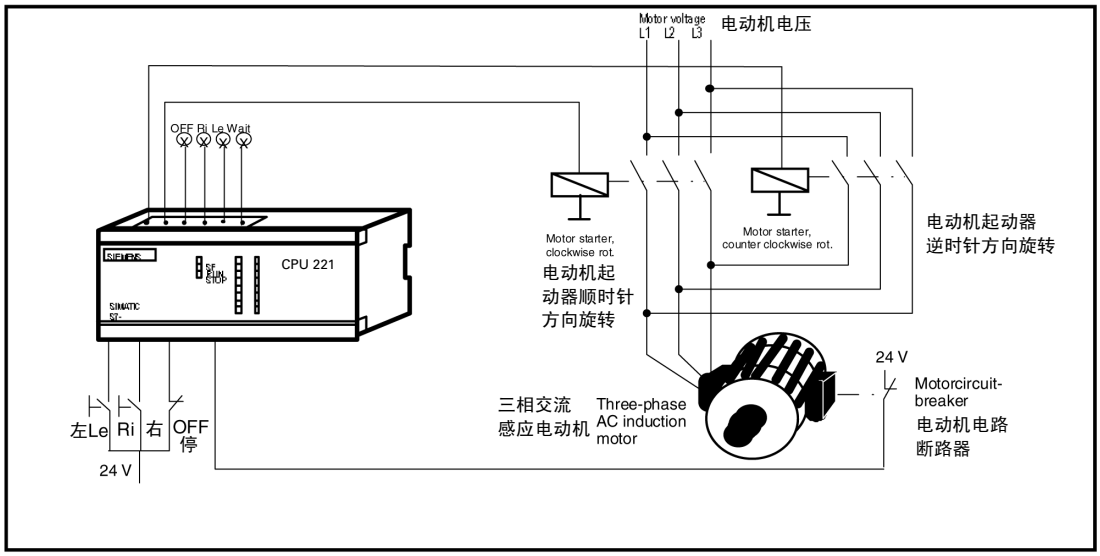
H.5 可逆电动机起动器电路——适用于改变三相交流感应电动机旋转方向

概述

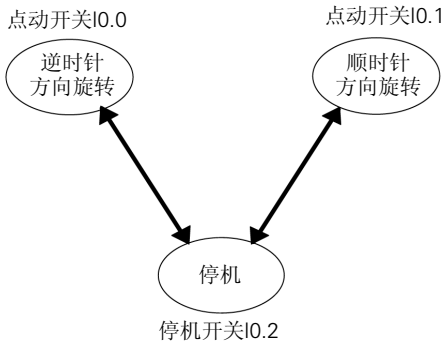
这个示例程序用于控制可双向运转的三相感应电动机。

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。
```

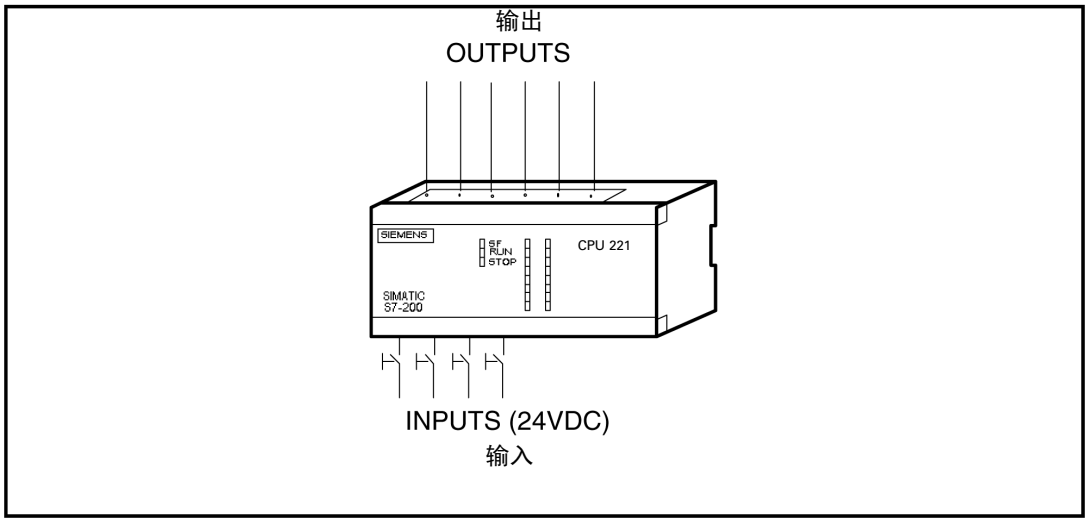
H.6 步执行顺序（事件鼓定时器）

概述

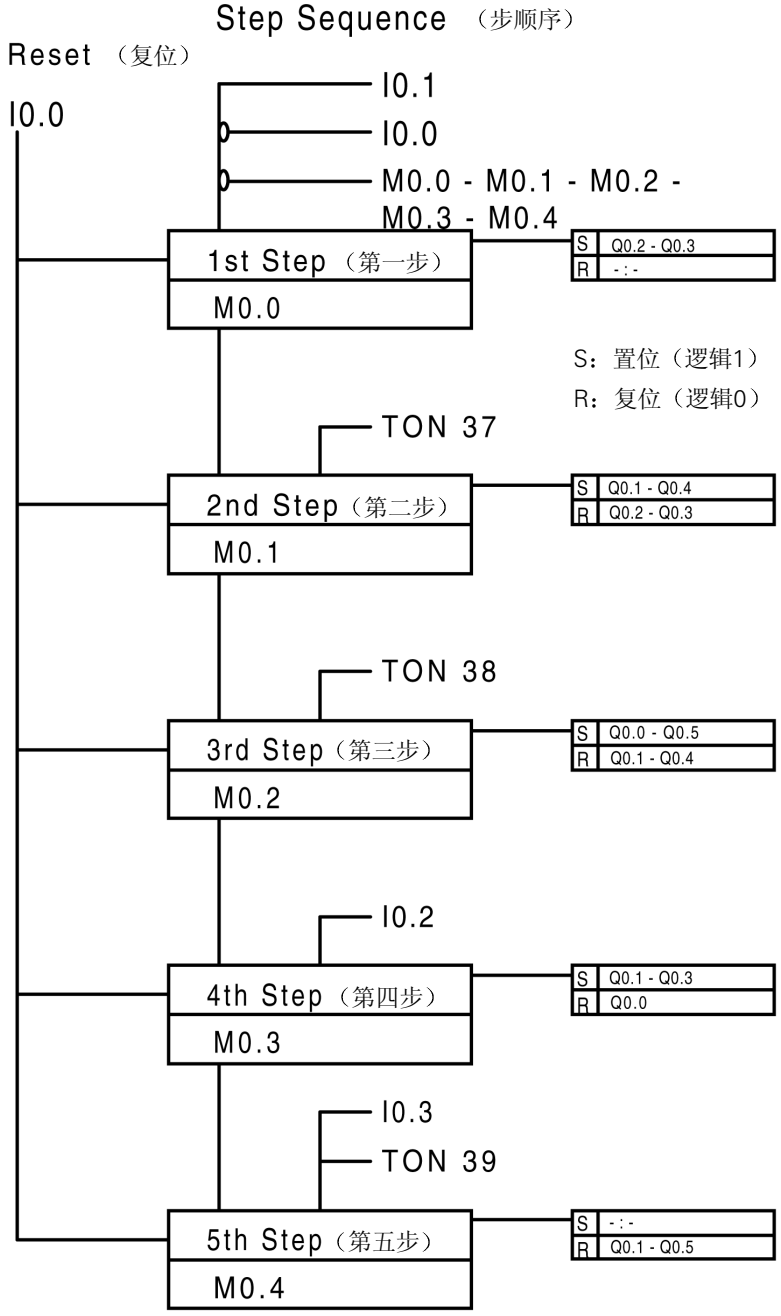
本程序实现了一个按事件步顺序执行的例子。每步均包含一系列的动作，一步紧跟一步，并且只有所有前提条件均满足时，才能执行。如下所示

	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。

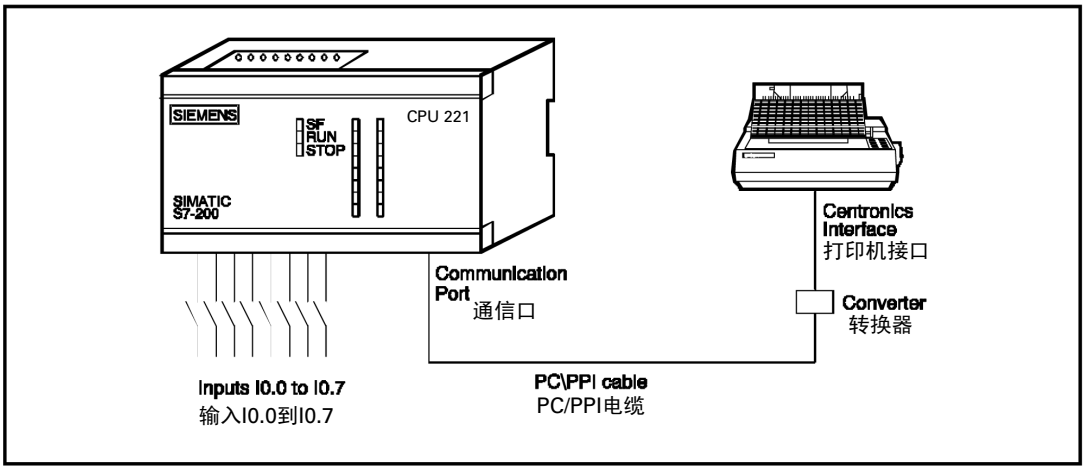
程序包含以下功能：

输入I0.0为1时，打印文字“SIMATIC S7-200”：

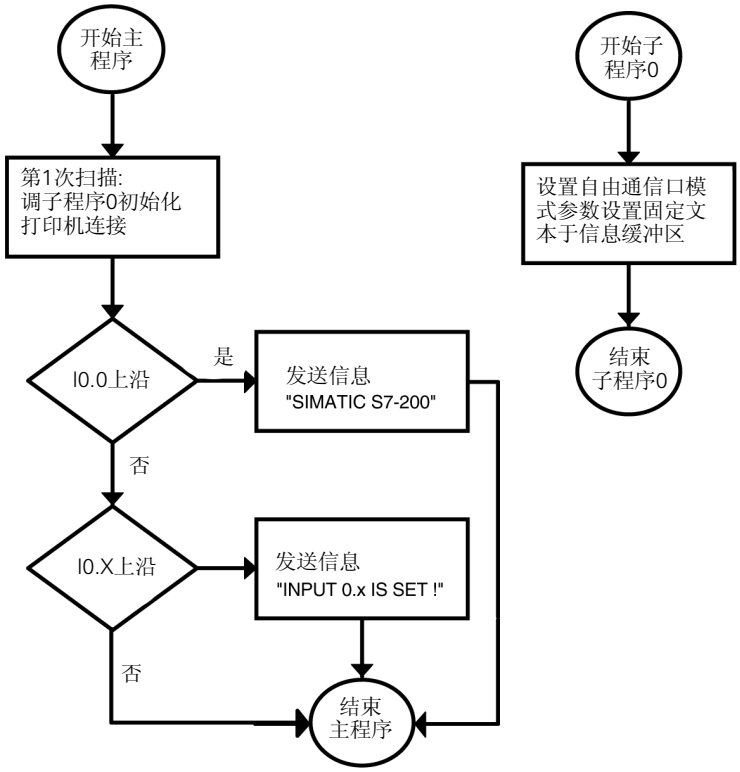
输入I0.1到I0.7为1时，打印句子“INPUT 0.×IS SET！”(其中×分别为1，2，……，7)。

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      I0.0                    // 起动打印输入I0.0
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      I0.0                    // 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␣！。

RET                               // 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

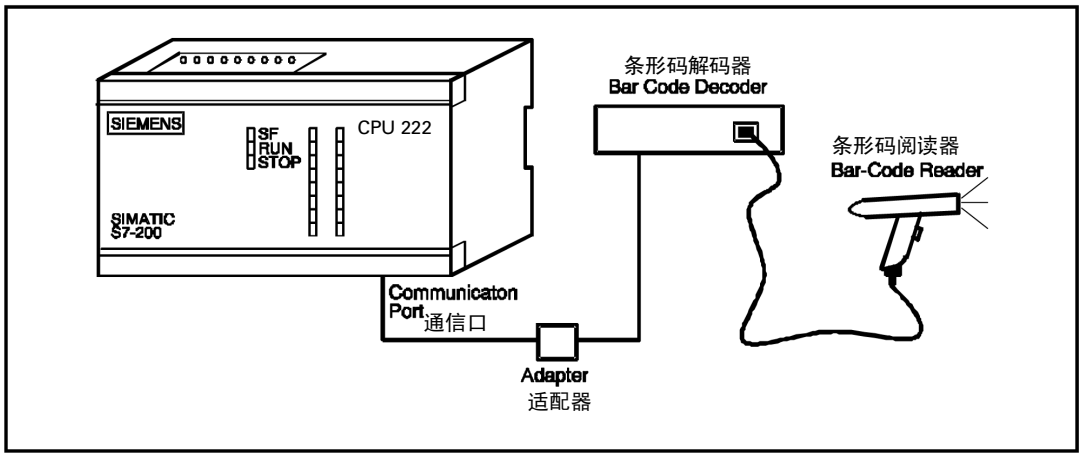
概述

本例说明如何将SIMATIC S7-222或S7-224与条形码阅读器配合作用。

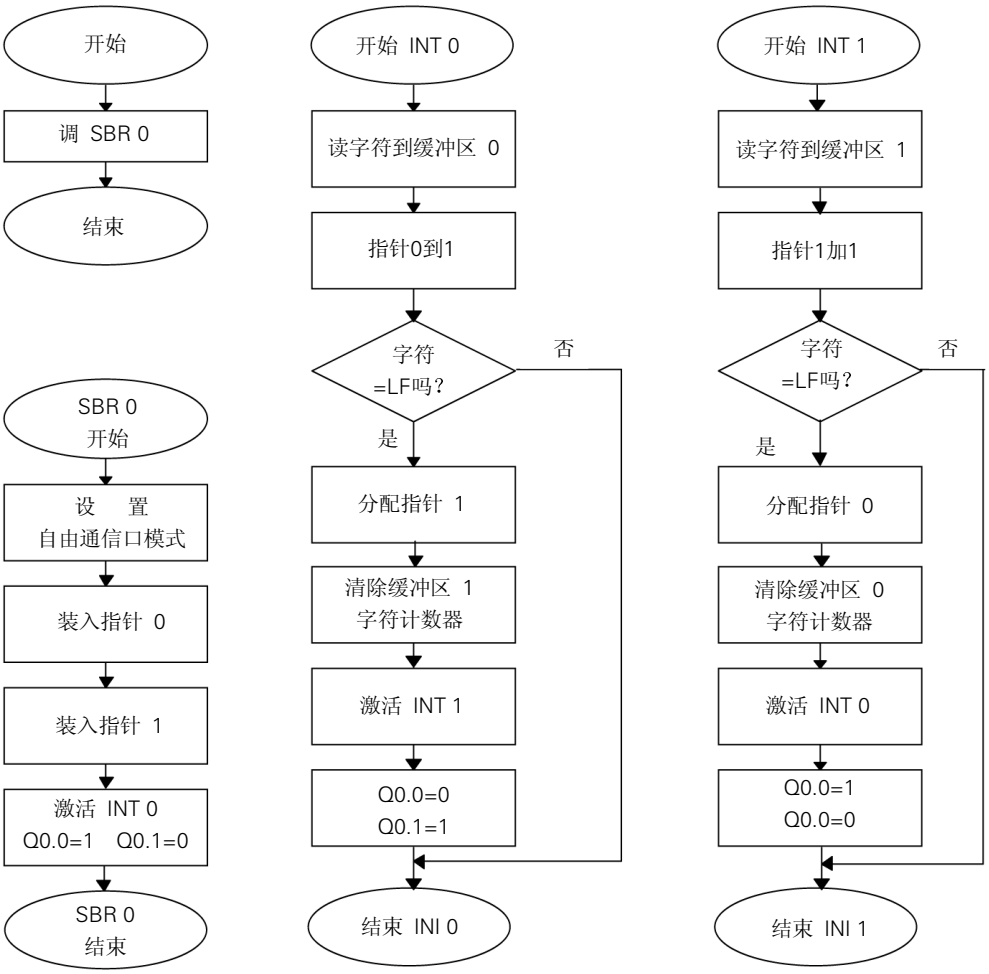
读入条形码的信息并经解码器翻译后，再通过自由通信口模式（Freeport Mode）把信息传入SIMATIC。在S7-222或S7-224的内存中有两个缓冲区，用来存储条形码信息，这两个缓冲区轮流地存储每次新读入的条形码。

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
```

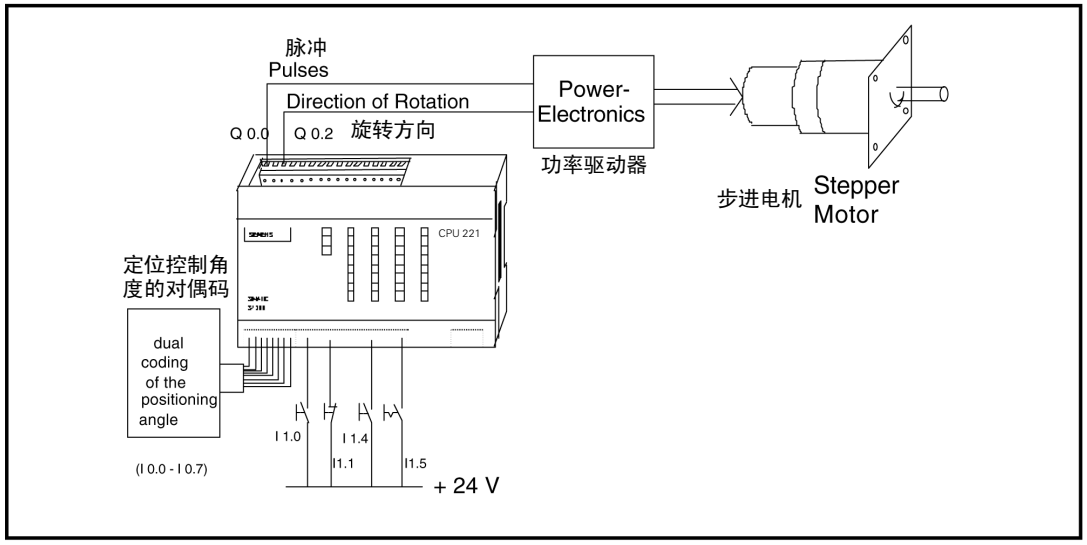
```
INT     0                       // 缓冲区0接收
MOVB SMB2，*VD56                // 字符装入缓冲区0
INCD   VD56                     // 指针加1，指向缓冲区的下一个位置。
INCW   VW54                     // 字符计数器加1
LDB=   SMB2，10                 // 若字符是LF，则
MOVD VD60，VD66                 // 使指针VD66指向缓冲区1，
MOVW +0，VW64                   // 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

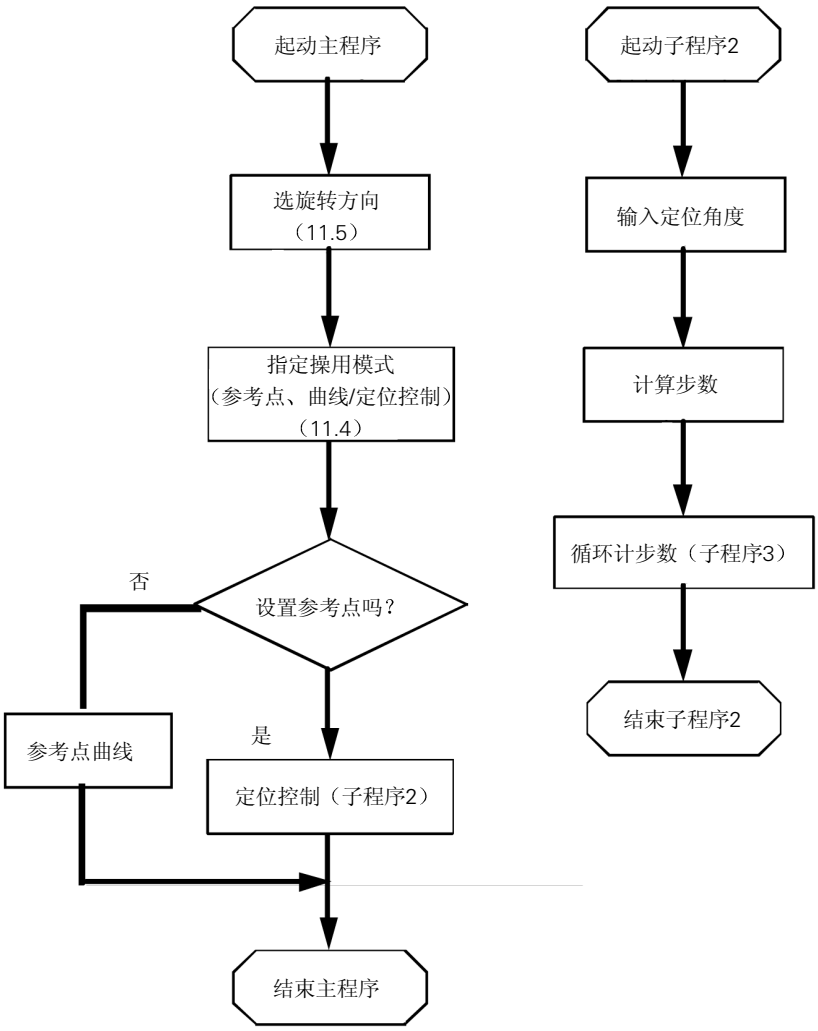
概述

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

一、初始化

在程序的第一个扫描周期（SM0.1=1），初始化重要参数。选择旋转方向和解除联锁，类似于应用示例22。

二、设置和取消参考点

如果还没有确定参考点，那么参考点曲线（Reference Point Curve）应从按“START”（起动）按钮（I1.0）开始。CPU有可能输出最大数量的控制脉冲。在所需的参考点，按“设置/取消参考点”开关（I1.4）后，首先调用停止电机的子程序。然后，将参考点标志位M0.3置成1，再把新的操作模式“定位控制激活”显示在输出端Q1.0。

如果I1.4的开关已被激活，而且“定位控制”也被激活(M0.3=1)，则切换到“参考点曲线”操作模式。在子程序1中，将M0.3置成0，并取消“定位控制激活”的显示(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IB0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。

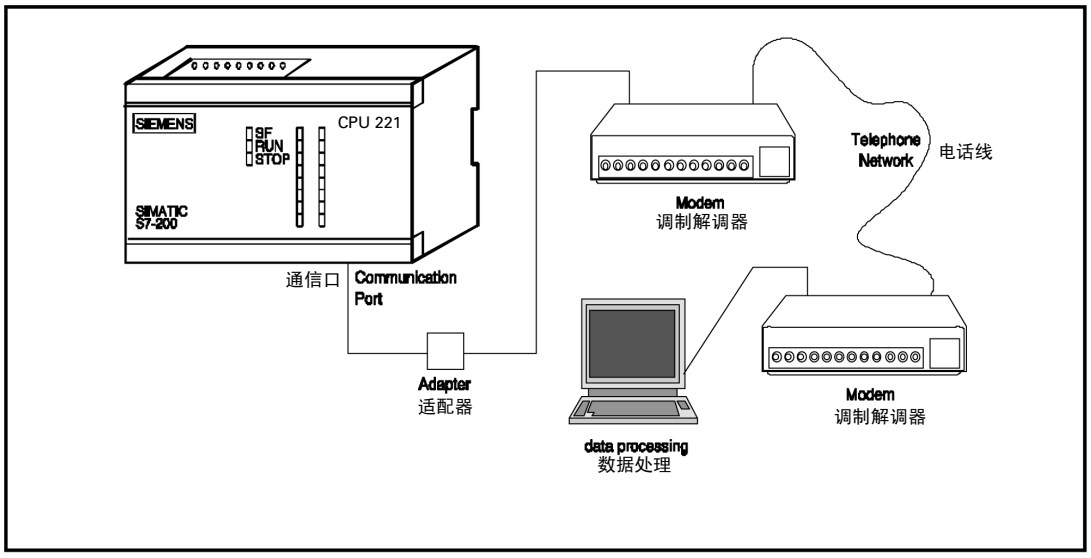
H.10 SIMATIC S7-221通过自由通信口模式控制贺氏（Hayes）调制解调器

概述

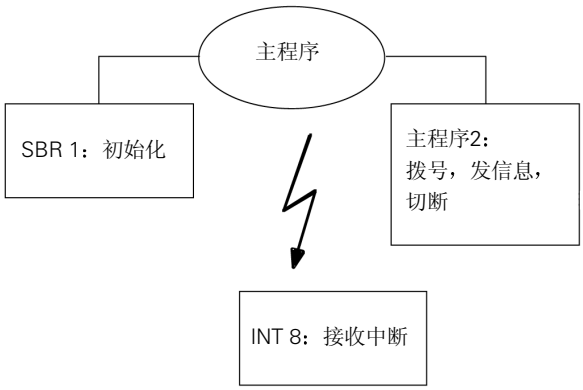
这一应用描述了如何通过SIMATIC调制解调器来使用与标准贺氏（Hayes）兼容的调制解调器，以及如何发送信息串。与S7-200相连的调制解调器，通过自由通信口模式（Freeport Mode）拨号叫另一台S7-200。由于贺氏调制解调器只能支持7个奇偶数据位，因而不能使用PPI模式，所以只能通过自由通信口模式来发送信息。

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
```

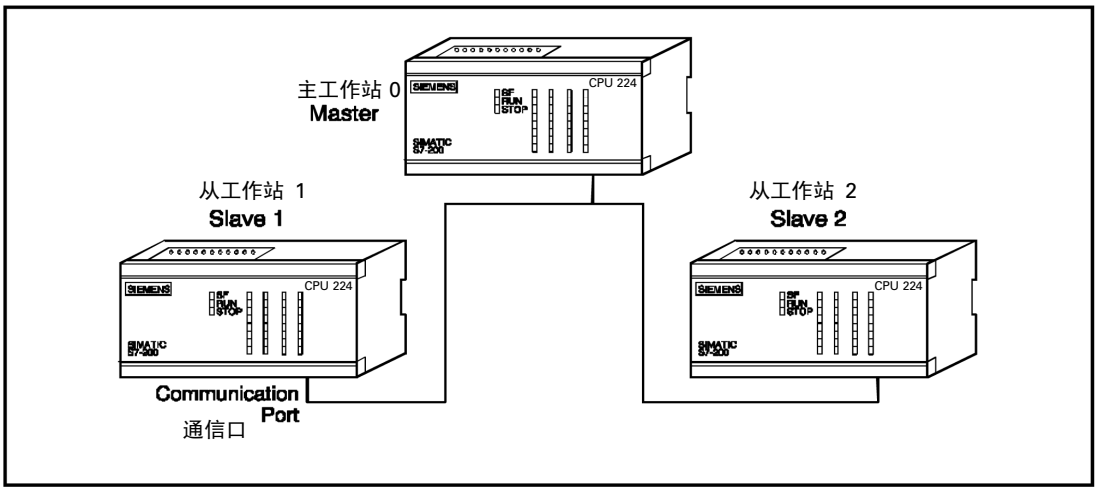
H.11 几台SIMATIC S7-200PLC使用自由通信口模式连接在一个远程I/O网络上

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



主工作站程序结构

Main	主程序
SBR0	选择PPI通信或Freeport（自由通信口）通信
INT0	接收定时器中断程序
INT1	发送定时器中断程序
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
```

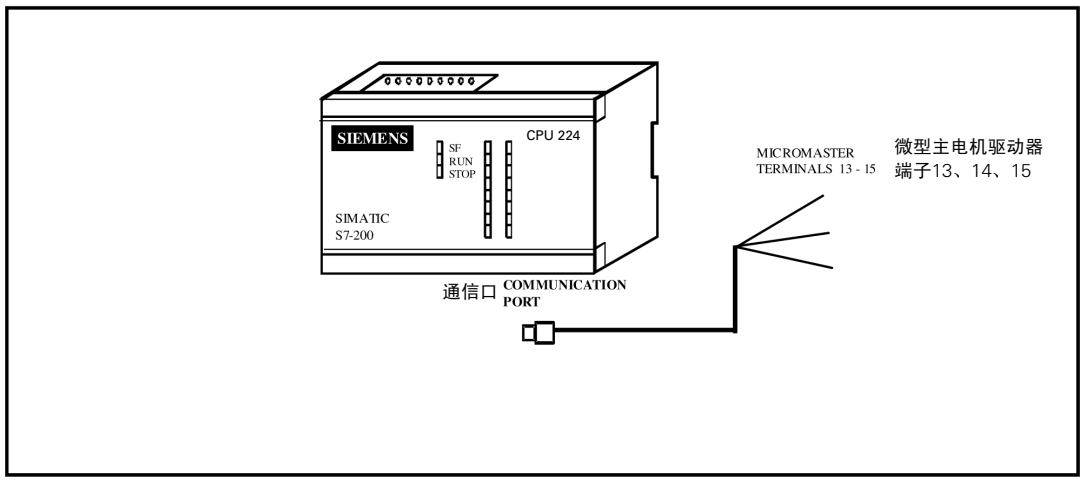
```
LD      SM1.0           // 如果接收到四个字符，则
ATCH    +13，8          // 使接收FCS有效
CRETI

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

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

+1      VW202，VW200            // + 因子（增加）
LD      I0.5                    // 倍增吗
+1      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

-1      VW202，VW200            // 一因子（降低）
LD      I0.5                    // 倍减?
-1      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
```

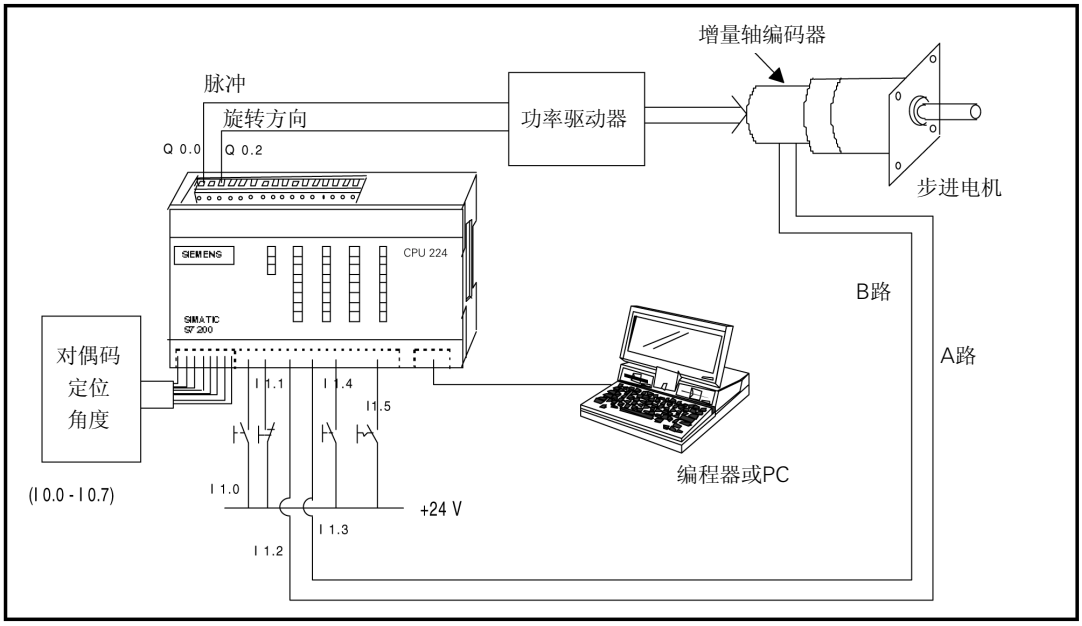
```
                                    // BCC正确，检验信息的其它部分
LDB=        VB114，16#02            // STX第一个字符吗？
AB=         VB115，16#0C            // 长度=12吗？
AB=         VB116，VB102            // 将信息发往同一从设备吗？
// * * * *

// 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

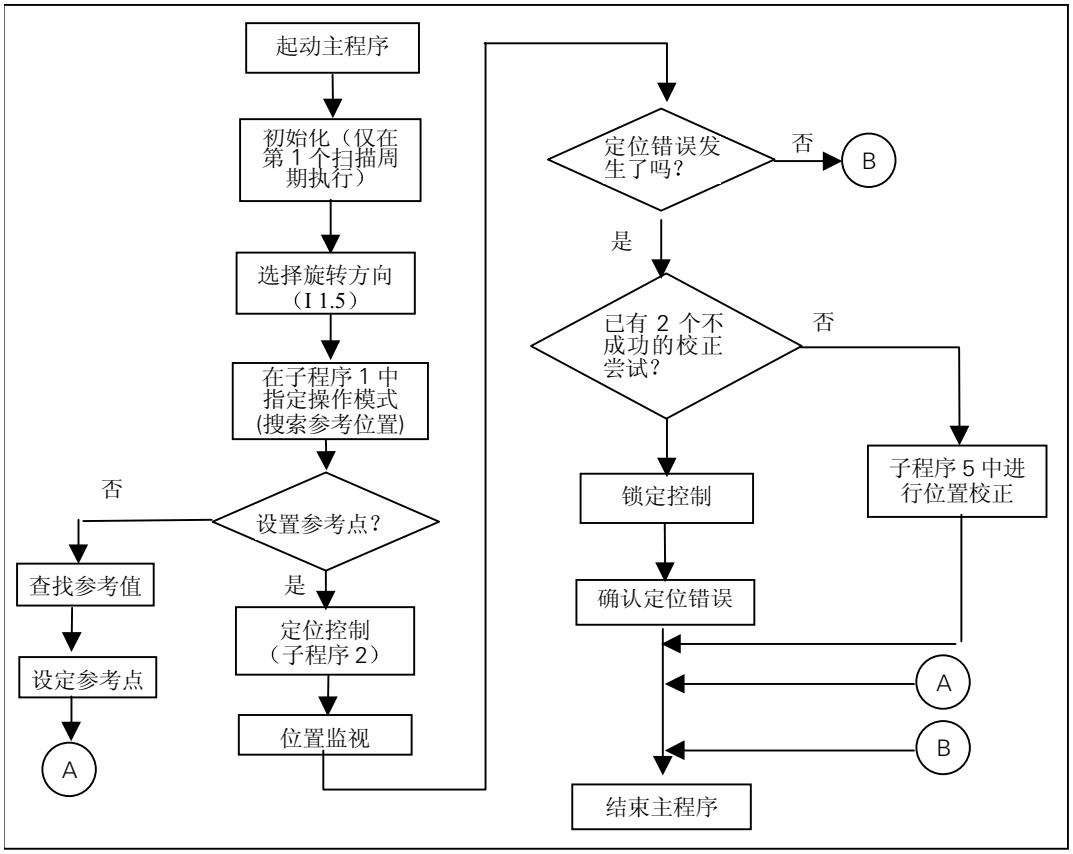
概述

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：

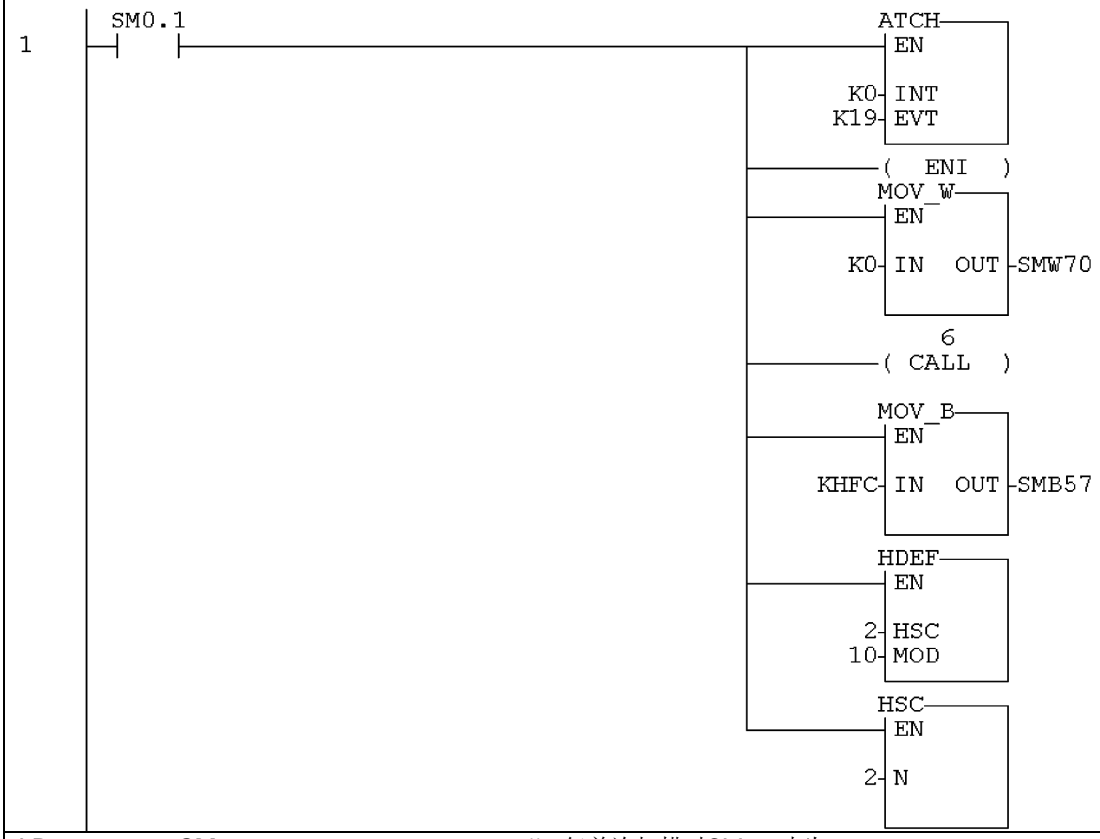
输入：	I 0.0——I 0.7	以度为单位的定位角（对偶码）
	I 1.0	“电机启动”按钮“START”
	I 1.1	“电机停止”按钮“STOP”
	I 1.2	传感器信号，A路
	I 1.3	传感器信号，B路
	I 1.4	“设置/取消参考点”按钮（确认开关）
	I 1.5	选择旋转方向的开关
输出：	Q 0.0	脉冲输出
	Q 0.2	旋转方向信号
	Q 1.0	操作模式的显示
	Q 1.1	定位错误的显示
标志位：	M 0.1	电机运转标志位
	M 0.2	锁定标志位
	M 0.3	参考点标志位
	M 0.4	完成第一次定位标志
	M 1.1	T1等待时间到标志位
	MD8,MD12	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)

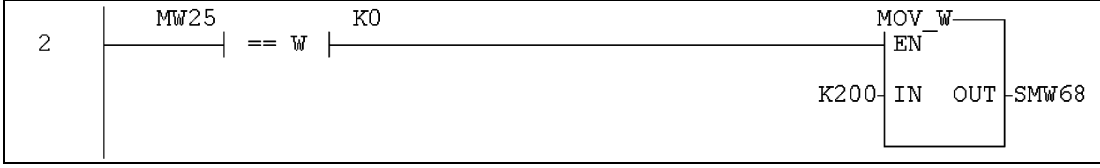
// 主程序

// 初始化



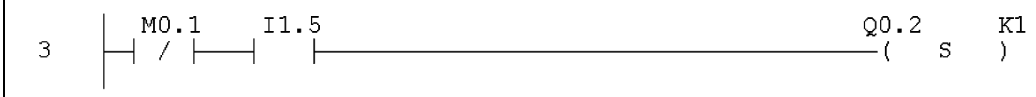

```
LD      SM0.1          //  仅首次扫描时SM0.1才为1
ATCH    0，19          //  把中断程序0分配给中断事件19（PLS0脉冲串终止）
ENI                    //  允许中断
MOVW    0，SMW70       //  脉冲宽度=0（脉宽调制）
CALL    6              //  在子程序6中进行初始化

// 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



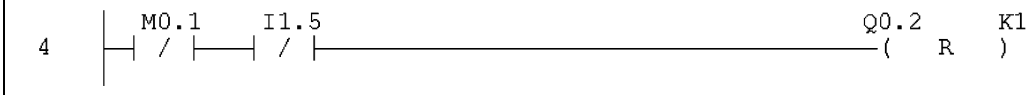
```
LDW=    MW25，0          // 若没有错误定位
MOVW    200，SMW68       // 则高速定位（T=200μs）
```

// 设置逆时针旋转



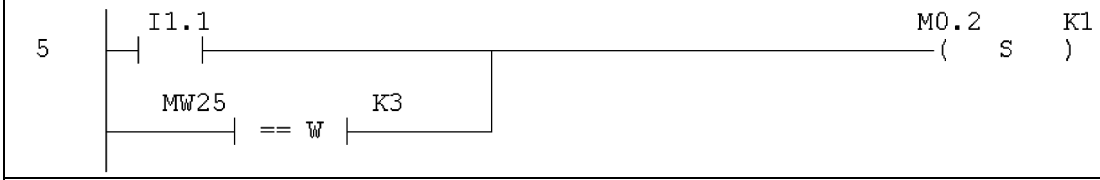

```
LDN     M0.1             // 若电机停止（M0.1=0）
A       I1.5             // 且按下旋转方向开关（I1.5=1）
S       Q0.2，1          // 则逆时针旋转（Q0.2=0）
```

// 设置顺时针旋转



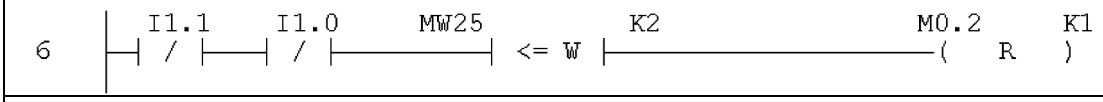

```
LDN     M0.1             // 若电机停止（M0.1=0）
AN      I1.5             // 且未按旋转方向开关（I1.5=0）
R       Q0.2，1          // 则顺时针旋转（Q0.2=0）
```

// 锁定



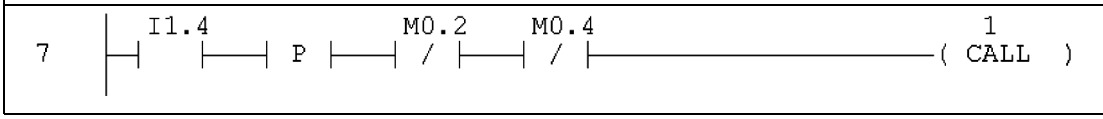

```
LD      I1.1             // 若按“电机停止（stop）”钮
OW=     MW25，3          // 或有3个错误定位
S       M0.2，1          // 则激活锁定（M0.2=1）
```

// 解除锁定



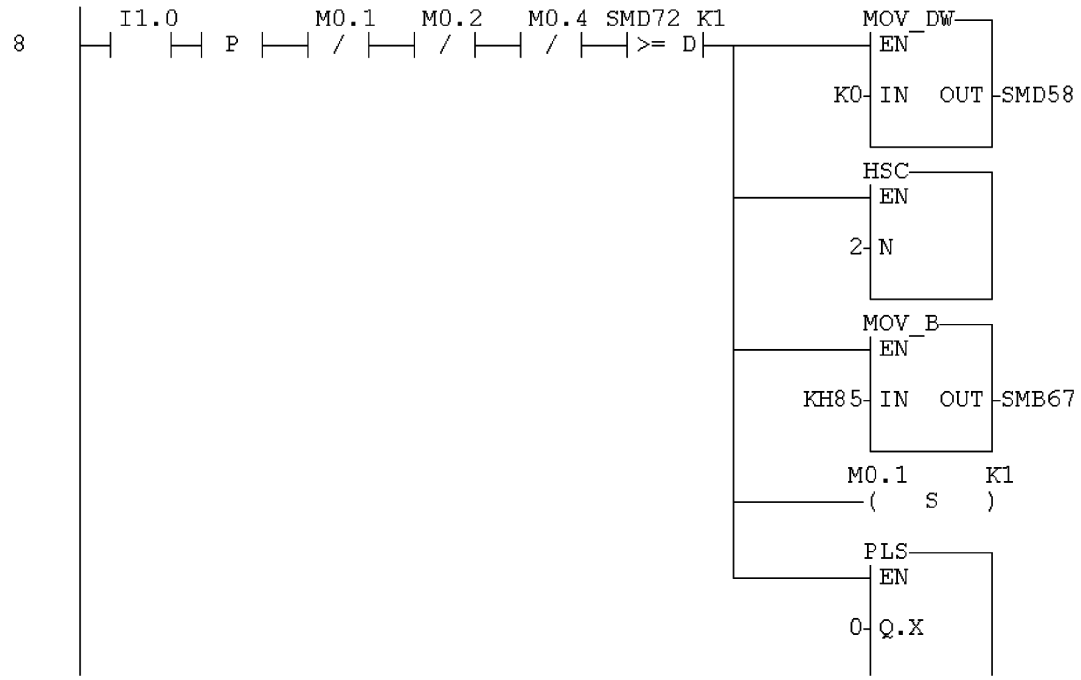

```
LDN     I1.1             // 若未按电机停止钮“STOP”（I1.1=0）
AN      I1.0             // 且未按电机起动钮“START”（I1.0=0）
AW<=    MW25，2          // 且<2个错误定位
R       M0.2，1          // 则解除锁定
```

// 指定操作模式（检索参考/定位）



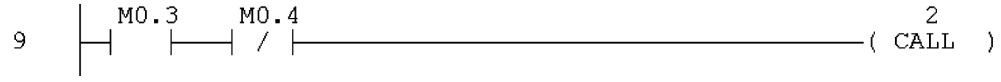
```
LD      I1 .4     // 若按下“设置/删除参考点”按针（I1.4=1）
EU                // 且上向沿
AN      M0.2      // 且无锁定（M0.2=0）
AN      M0.4      // 且无定位（M0.4=0）
CALL    1         // 则调用子程序1指定操作模式
```

// 起动电机



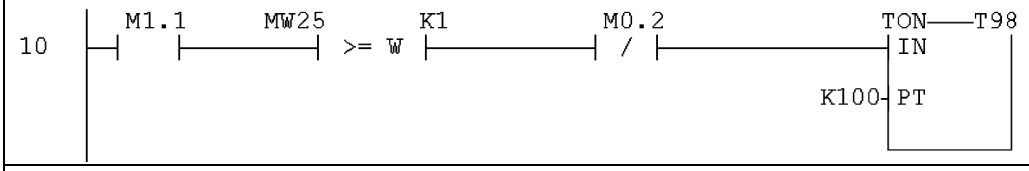

```
LD      I1.0      // 若按“电机起动”接钮“START”（I1.0=1）
EU                // 且上升沿
AN      M0.1      // 且电机在停止状态（M0.1=0）
AN      M0.2      // 且无联锁（M0.2=0）
AN      M0.4      // 且无定位控制（M0.4=0）
AD > =  SMD72，1  // 且步数 > 1，则
MOVD    0，SMD58  // 置HSC2的起始值为0
HSC     2         // 起动HSC2
MOVB    16#85，SMB67 // 激活脉冲输出功能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

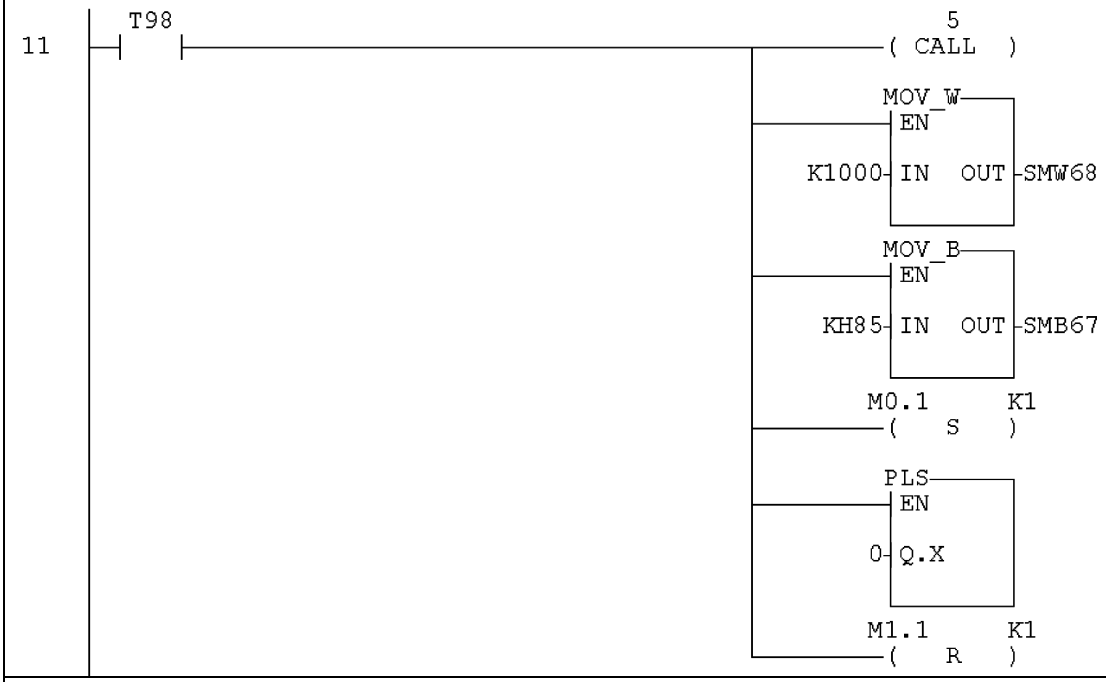



```
LD      M0.3      // 若“定位”操作模式（M0.3=1）
AN      M0.4      // 且尚未定位（M0.4=0）
CALL    2         // 则调用子程序2定位
```

// 位置校正



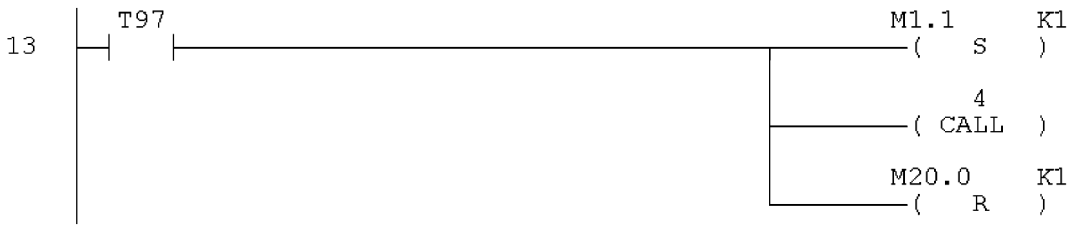

```
LD      M1.1          // 若T1到时（M1.1=1）
AW >=   MW25,1        // 且检测出错误定位（MW25≥1）
AN      M0.2          // 且未激活锁定（M0.2=0）
TON     T98,100       // 则起动等待定时器T2（1s）
```




```
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）
```

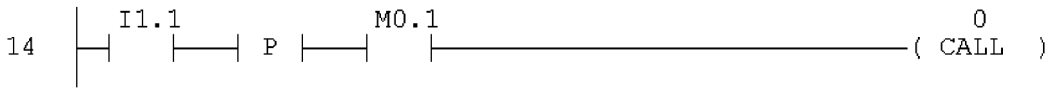
// 位置监视

```
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）
```



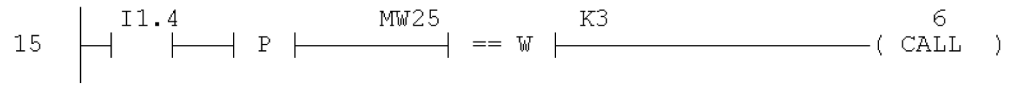

```
LD      T97        // 若T1到时（T97=1），则
S       M 1.1，1   // T1标志置位（M1.1=1）
CALL    4          // 在子程序4中调用位置监视
R       M20.0，1   // 脉冲输出结束标志位复位（M20.0=0）
```

// 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”
```

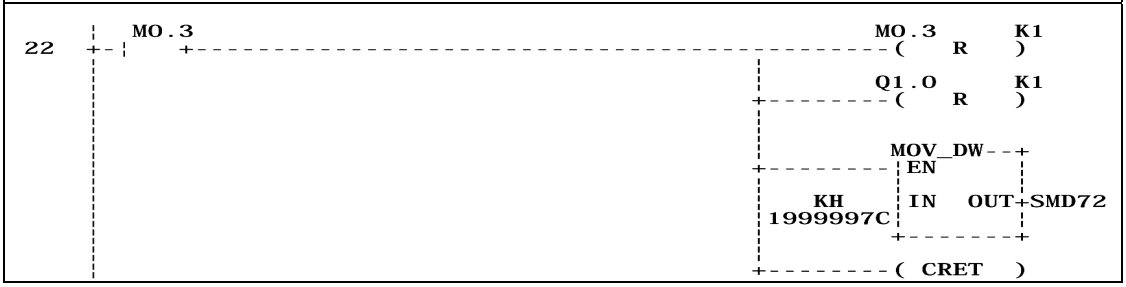

```
SBR         0                   // 子程序0
LD          SM0.0               // SM0.0总是1
MOVB        16#CB，SMB67        // 激活脉冲宽度调制（即置PTO0的控制位）
PLS         0                   // Q0.0停止输出脉冲
R           M0.1，1             // 对“电机运转”标志复位（M0.1=0）
```


```
RET                             // 子程序0结束
```

```
// 子程序1“指定操作模式”
```

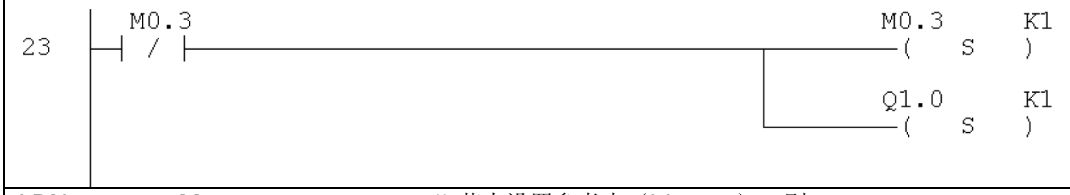

```
SBR         1                   // 子程序1
LD          M0.1                // 若电机运转（M0.1=1）
CALL        0                   // 则调用子程序0停止电机
```

```
// 开始检索参考点
```

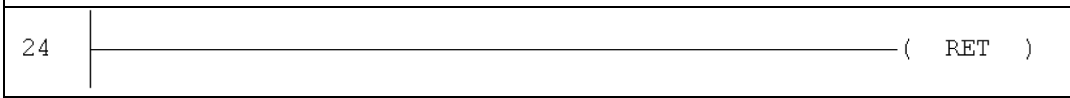


```
LD      M0.3                    // 若“定位”标志激活（0.3=1），则
R       M0.3，1                 // 参考点标志复位（M0.3=0）
R       Q1.0，1                 // 删除“定位激活”信号(Q1.0=0)
MOVD    16#1999997C，SMD72      // 为新的参考点设置最大脉冲数
CRET                            // 条件返回
```

// 请求定位控制

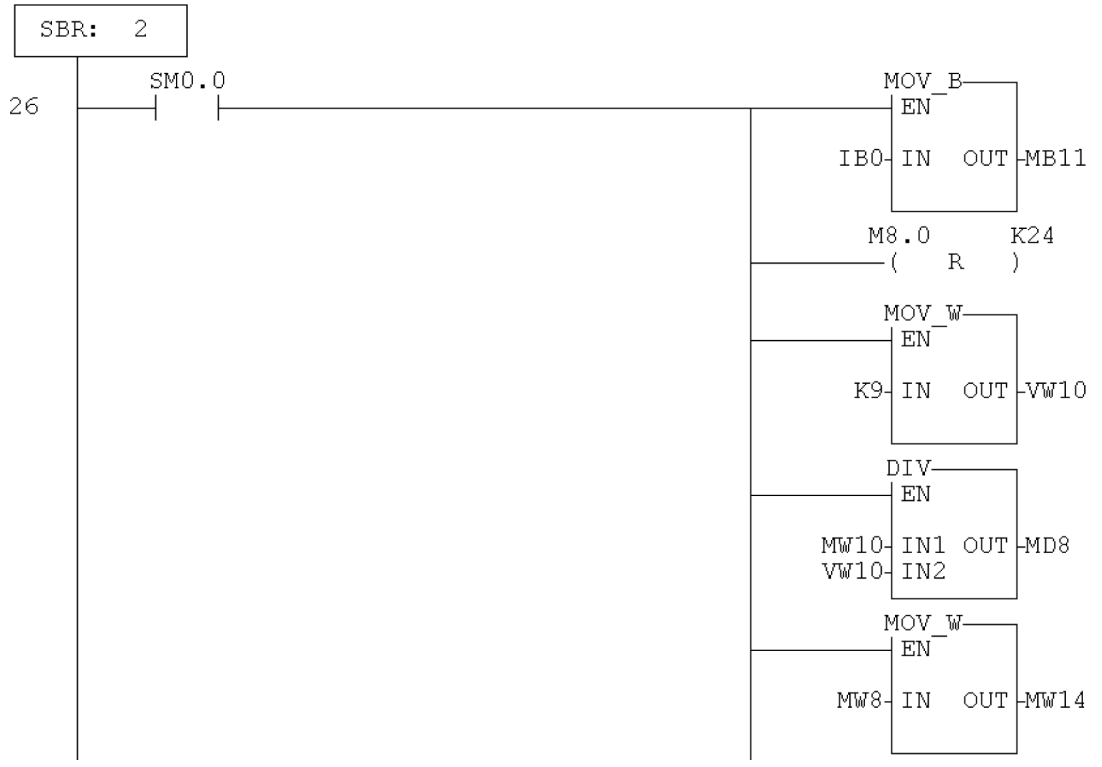


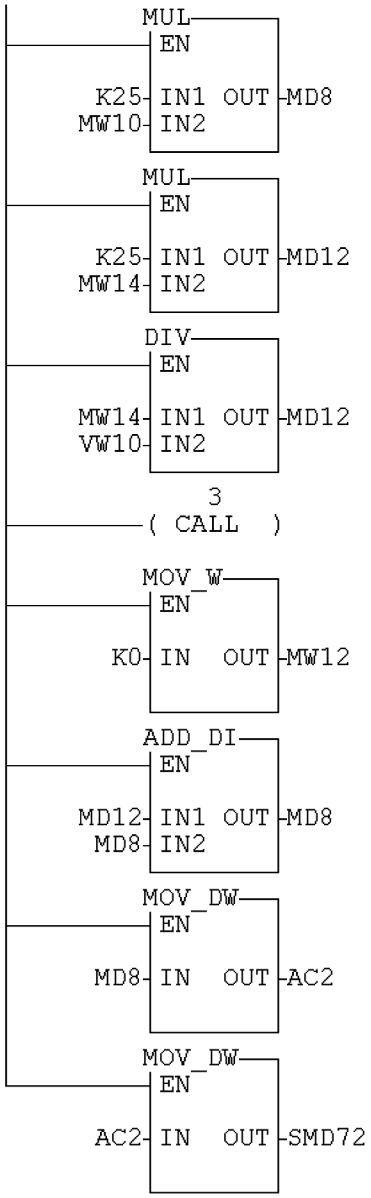

```
LDN     M0.3                    // 若未设置参考点（M0.3=0），则
S       M0.3，1                 // 参考点标志位置位（M0.3=1）
S       Q1.0，1                 // 输出“定位激活”信号（Q 1.0=1）
```




```
RET                             // 子程序1结束
```

// 子程序2“计算步数，允许偏差极限”

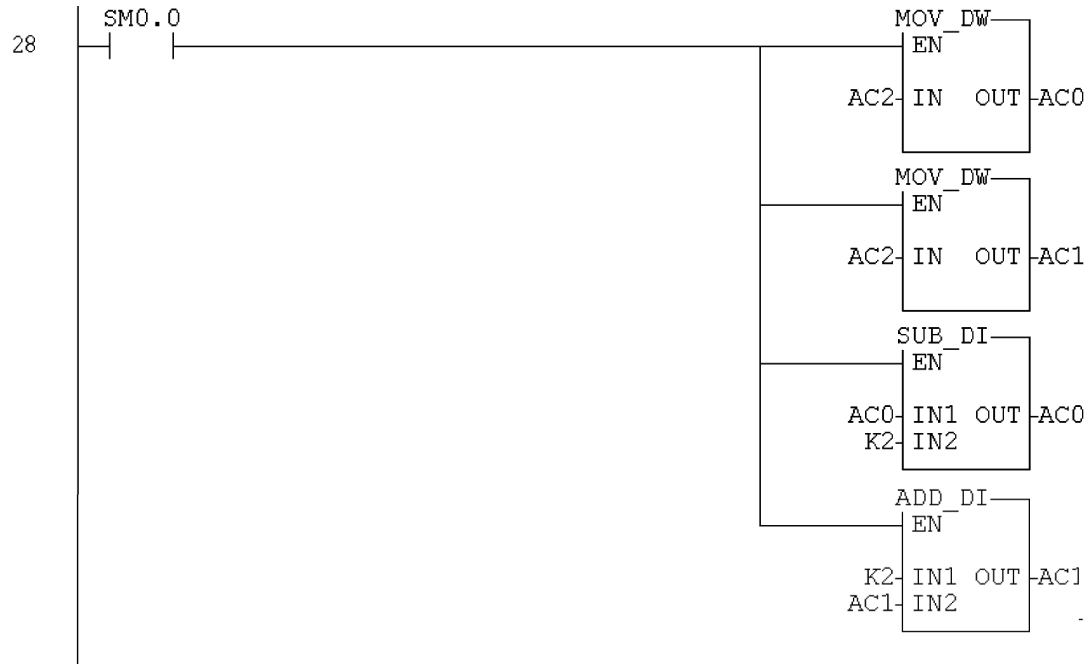





```
SBR2                    // 子程序2
LD      SM0.0           // SM0.0总为1
MOVB    IB0，MB11       // 把预设定位角从输入字节IB0复制到MD8的最低有效字节MB11
R       M8.0，24        // MB8至MB10请零
MOVW    K9，VW10        // 把9置入VW10
DIV     VW10，MD8       // 角度/9=q1+r1(q1=商，r1=余数)
MOVW    MW8，MW14       // 把r1（余数）存入MD12
MUL     25，MD8         // 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
```

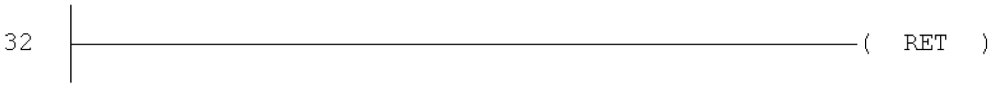



```
LD      SM0.0           // SM0.0总为1
MOVD    AC 2，AC0       // AC2存入AC0
MOVD    AC2，AC1        // AC2存入AC1
-D      2，AC0          // 最低限值（AC2-2=AC0）
+D      2，AC1          // 最高限值（AC2+2=AC1）
```


```
RET                                    // 子程序2结束
```

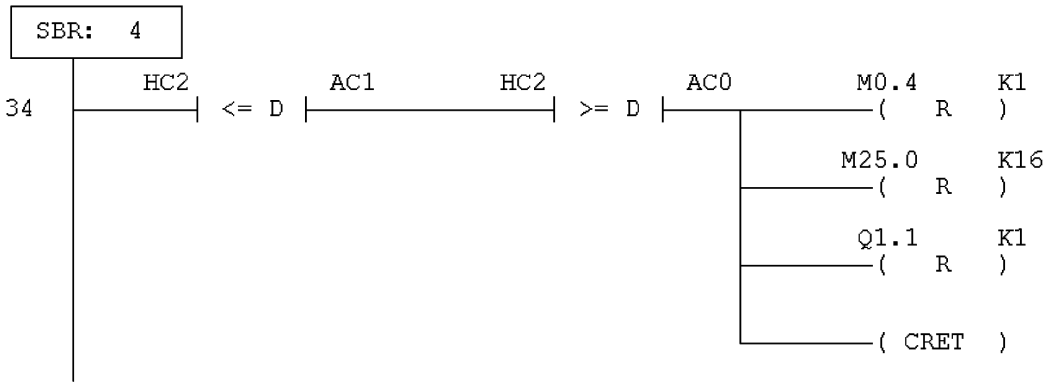
// 子程序3“4舍5入步数”


```
SBR       3          // 子程序3
LDW>=     MW12，5    // 如果r2>=5/9
INCW      MW14       // 则步数增加1
```

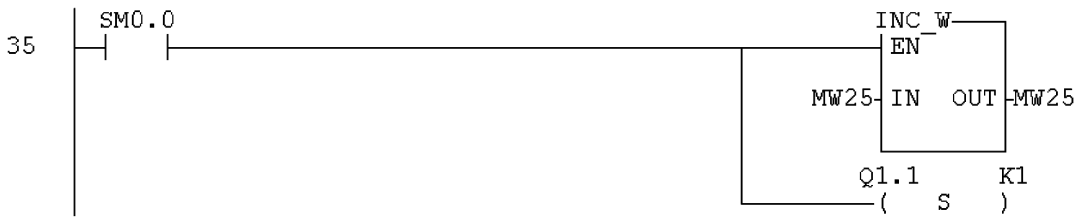



```
RET                                    // 子程序3结束
```

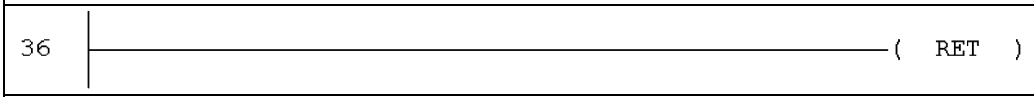
// 子程序4“位置监视”




```
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
```

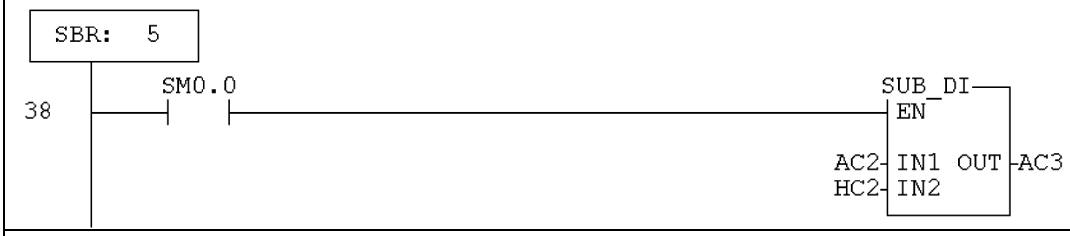



```
LD      SM0.0     // 错误定位
INCW    MW25      // 错误定位计数器加1
S       Q1.1，1   // 显示错误定位
```

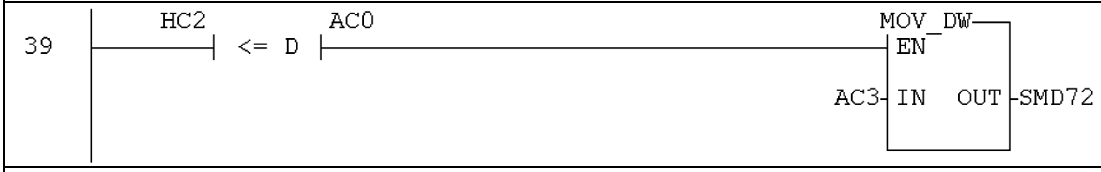



```
RET               // 子程序4结束
```

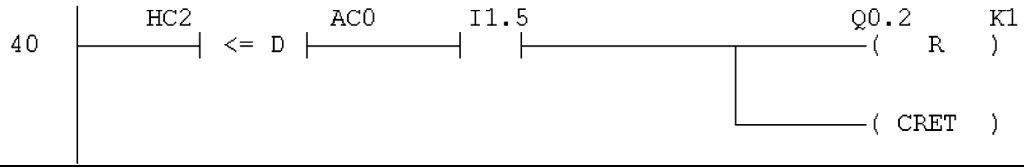
// 子程序5“计算校正步数”



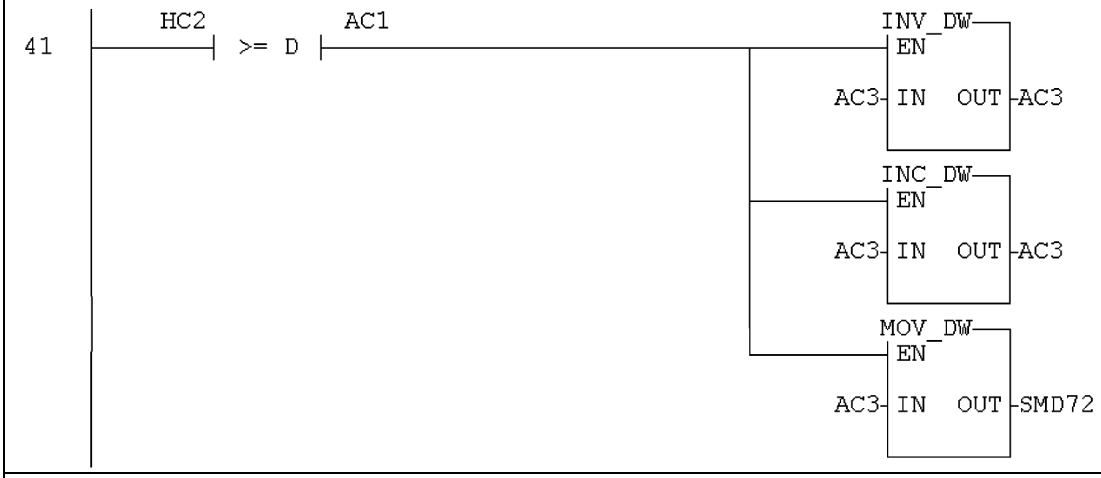

```
SBR     5         // 子程序5
LD      SM0.0     // SM0.0总为1
MOVD    AC2，AC3  // AC2存入AC3
-D      HC2，AC3  // 设定值－实际值=AC3
```



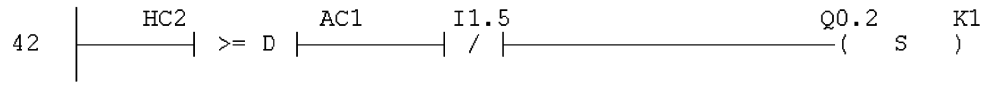

```
LDD<=   HC2，AC0  // 若实际值<设定值AC0
MOVD    AC3，SMD72// 则步数=设定值－实际值
```



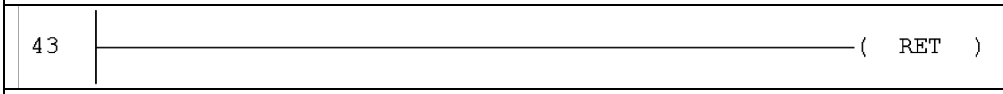

```
LDD<=   HC2，AC0  // 若实际值<设定值AC0
A       I1.5      // 且逆时针方向旋转（I1.5=1）
R       Q0.2，1   // 则顺时针方向校正
CRET              // 条件返回
```




```
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
```

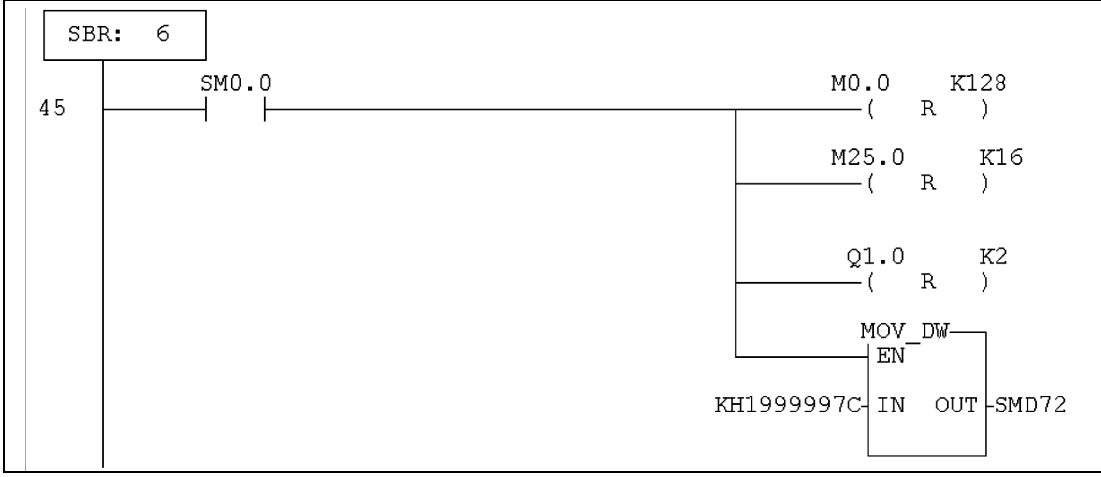



```
LDD>=    HC2，AC1        // 若实际值>设定值AC1
AN       I1.5            // 且顺时针方向旋转时（方向开关I1.5=0）
S        Q0.2，1         // 则逆时针方向校正
```

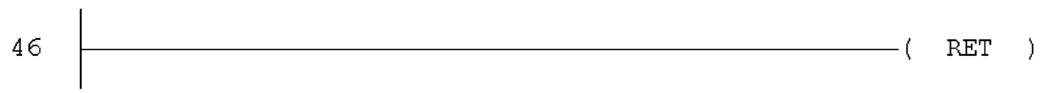



```
RET                      // 子程序5结束
```

```
// 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）

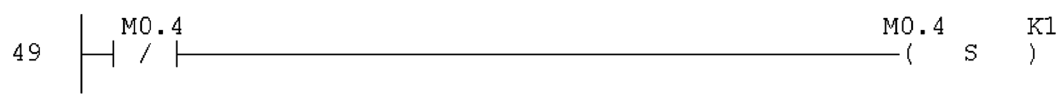


RET	// 子程序6结束

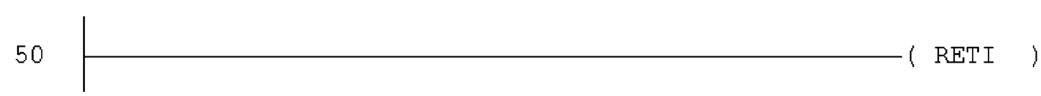
中断程序

// *
// 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）



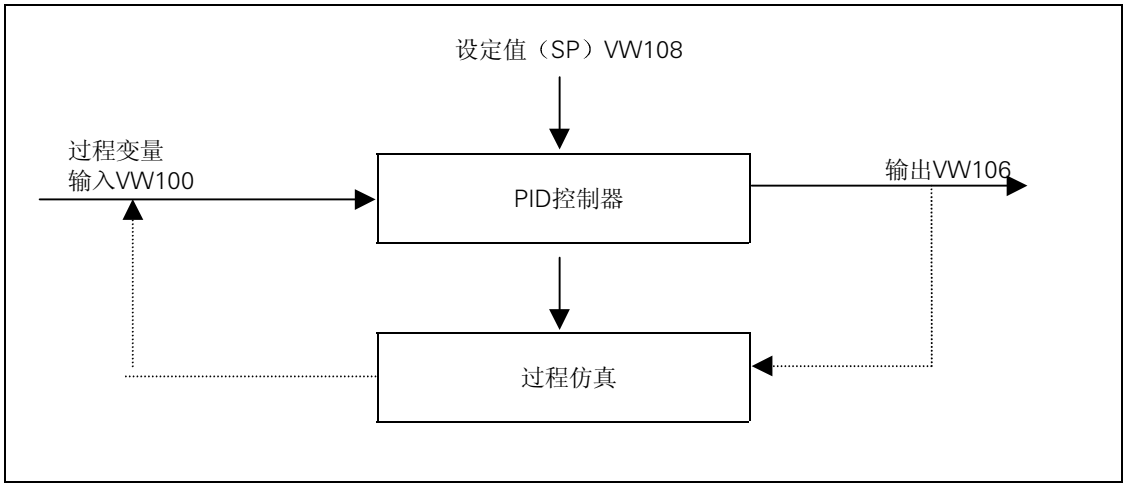
RET1	// 中断程序0结束

H.14 用S7-200实现PID控制

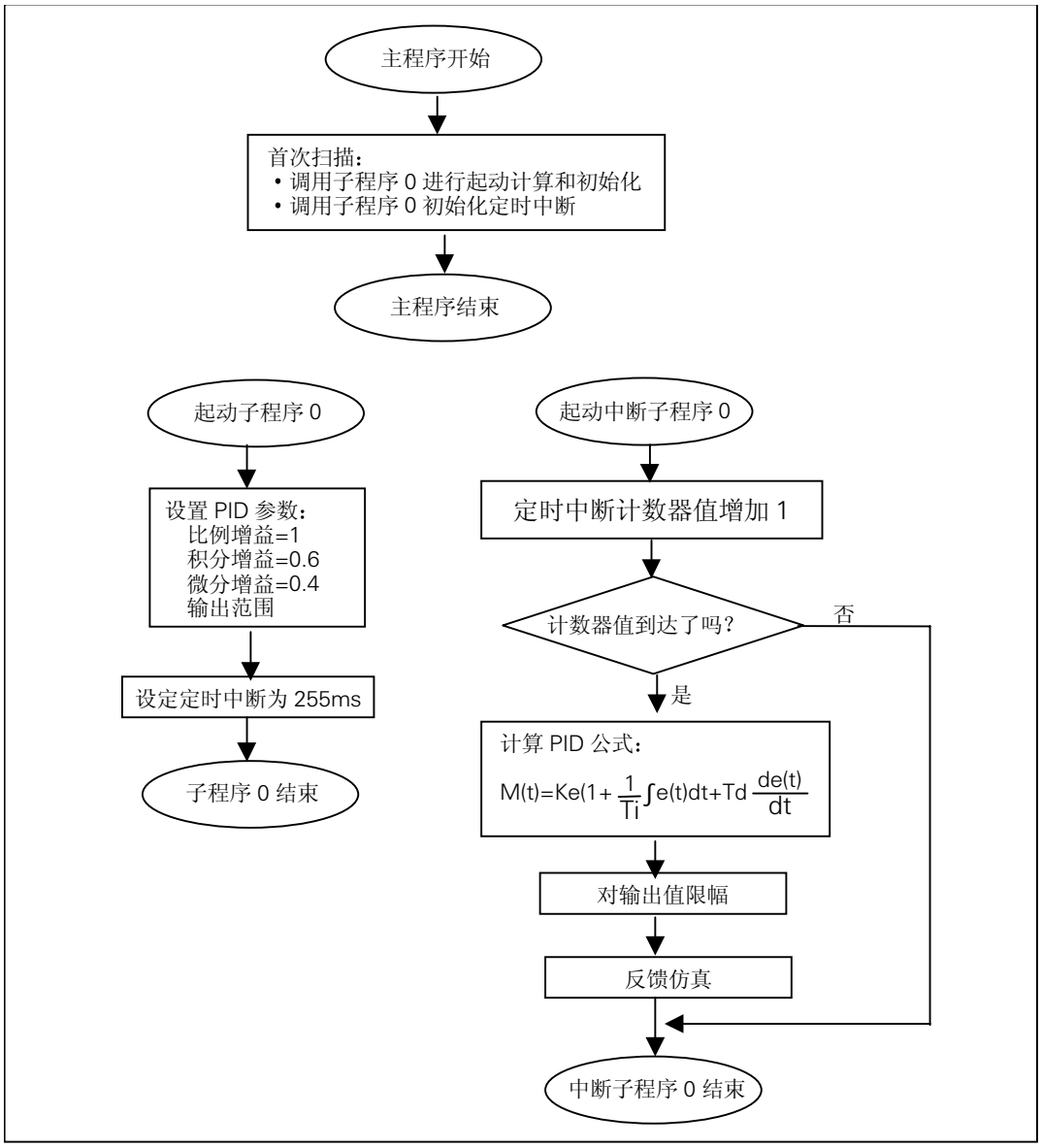
概述

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



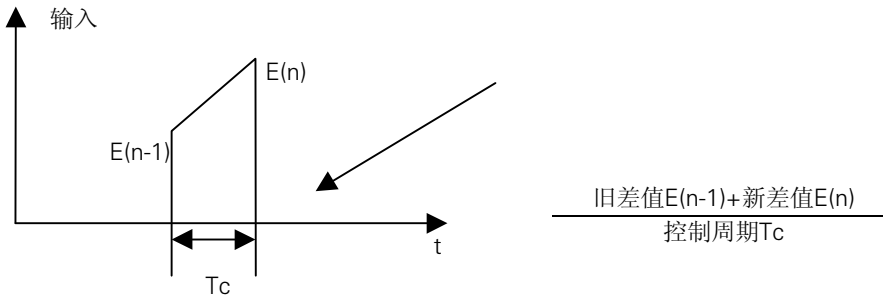
程序框图


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

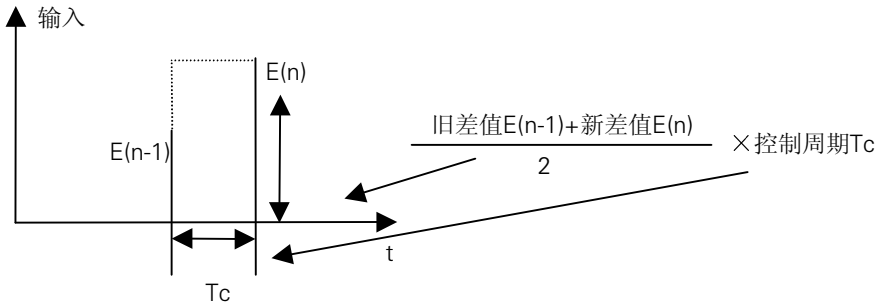
程序及注解

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：

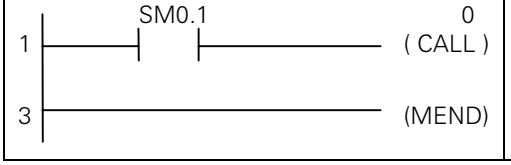


处理过程中的反馈输入变量来自仿真
程序全长370个字

LAD	STL（IEC）

//主程序

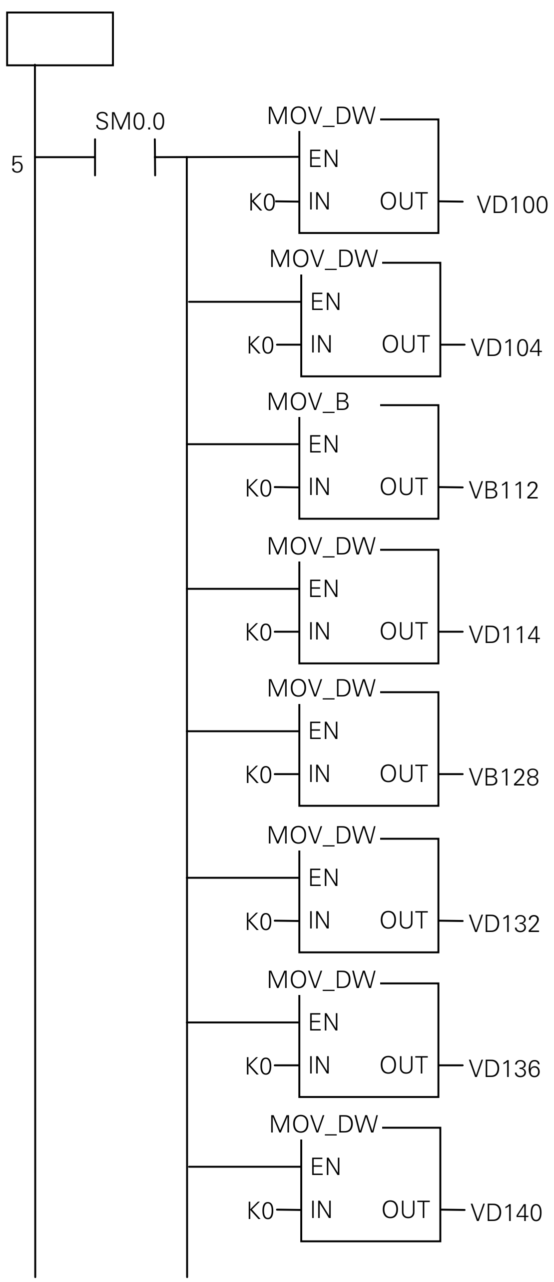
```
// VW100=过程变量（PV）输入量
// 如果模拟模块已准备好，它将是一个模拟输入字，并假设该输入的最大范围是
// -2047~+2047
// VW102=新控制差值（运行期间计算）E(n)
// VW104=为输出值而设的计算缓冲器
// VW106=输出值
// VW108=设定值（在CPU 214中，此值能够从模拟调节装置POTS中调入）
// VW110=预置定时中断个数，如果控制周期Tc大于255ms，则在此内存字中置数；
// 计算Tc的公式：
//                    VW110×VB113=Tc（单位：ms）
// VW112=未用
// VW113=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
```



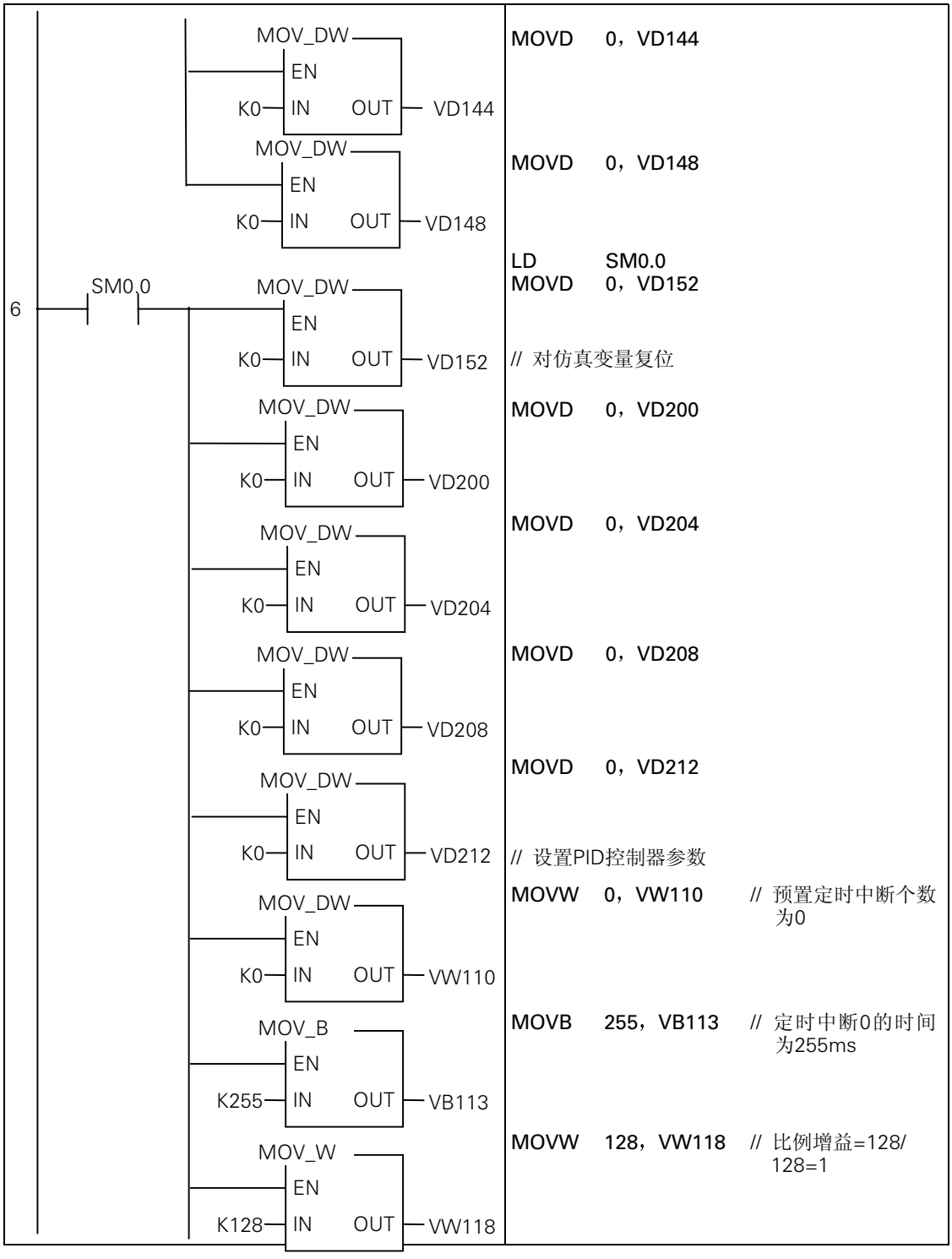

```
LD      SM0.1   // 仅首次扫描时SM0.1才为1
CALL    0       // 调用子程序0，进行起动运
                   算，初始化，设置定时中断
// 用户程序码段
MEND            // 主程序结束
```

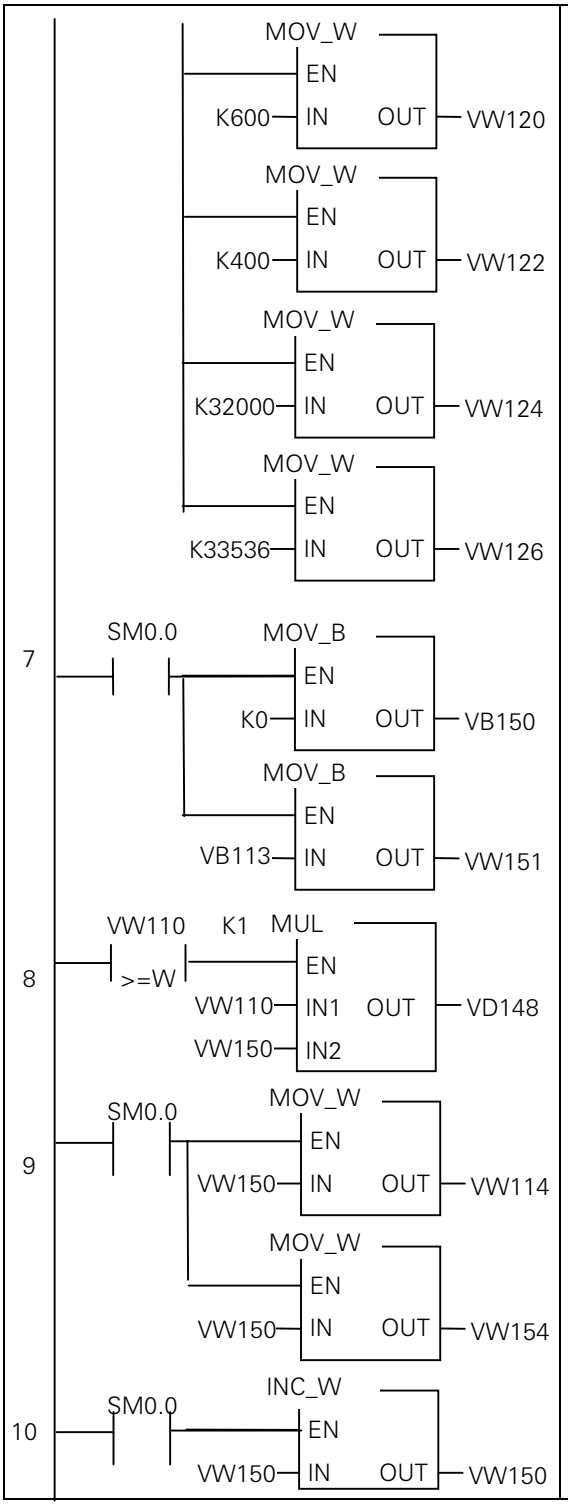
//子程序

```
// ***********************************************************
// 功能：起动子程序
// ***********************************************************
```

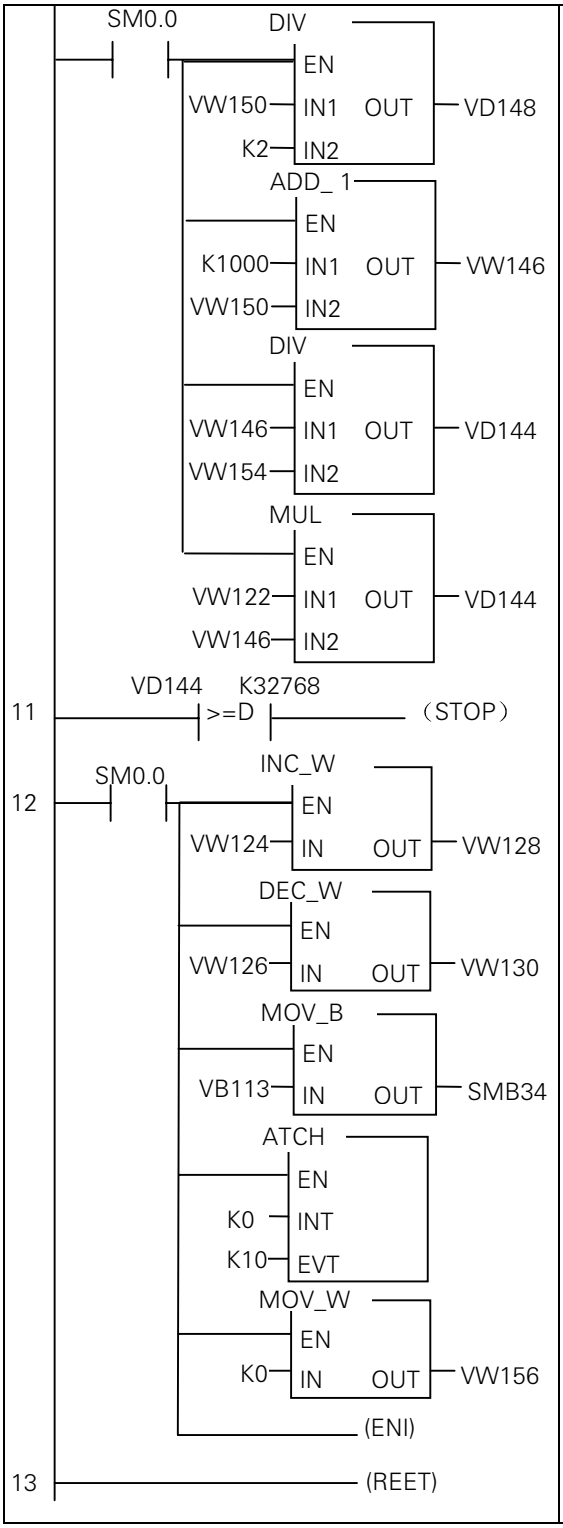



```
SBR     0               // 子程序0

// 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
```


MOV_DW
EN
K0
IN
OUT
VD144
MOV_DW
EN
K0
IN
OUT
VD148
6
SM0.0
MOV_DW
EN
K0
IN
OUT
VD152
MOV_DW
EN
K0
IN
OUT
VD200
MOV_DW
EN
K0
IN
OUT
VD204
MOV_DW
EN
K0
IN
OUT
VD208
MOV_DW
EN
K0
IN
OUT
VD212
MOV_DW
EN
K0
IN
OUT
VW110
MOV_B
EN
K255
IN
OUT
VB113
MOV_W
EN
K128
IN
OUT
VW118
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



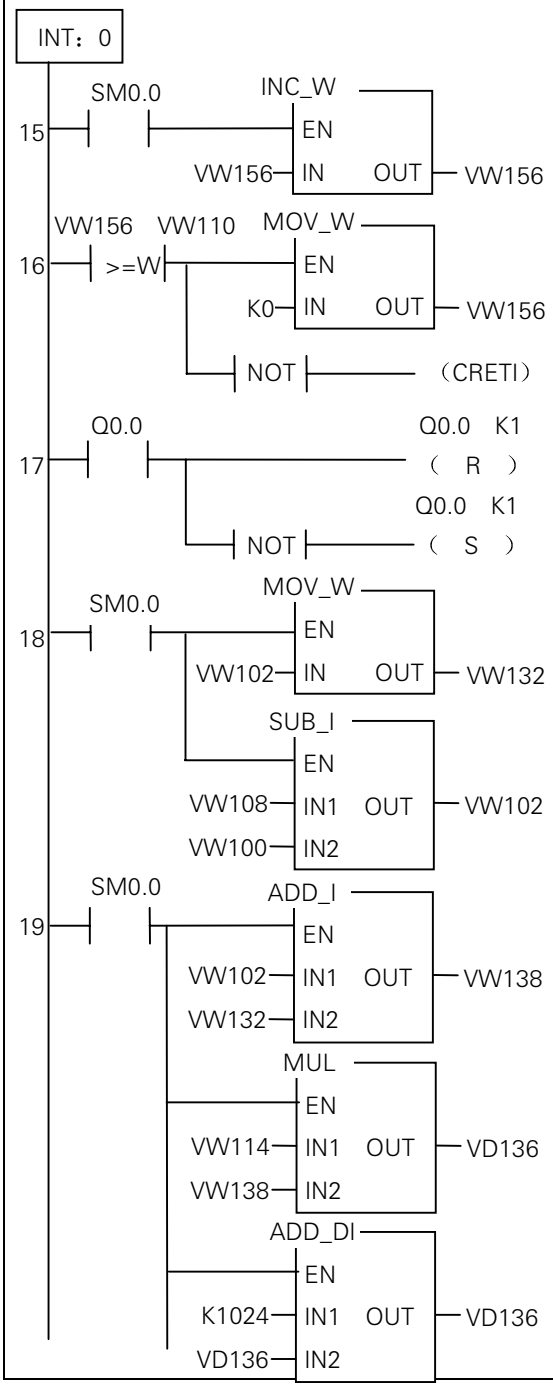

```
MOVW    600，VW120      // 积分增益= 600/1024=0.6
MOVW    400，VW122      // 微分增益=400/1024=0.4
MOVW    32000，VW124    // 最大输出值是+32000
MOVW    -32000，VW126   // 最小输出值是-32000
LD      SM0.0           //  SM0.0总为1
MOVB    0，VB150        // 运算缓冲器VB150清零
MOVB    VB113，VB151    // 将定时中断时间移入运算缓冲器VB151
LDW>=VW110，1           // 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
```



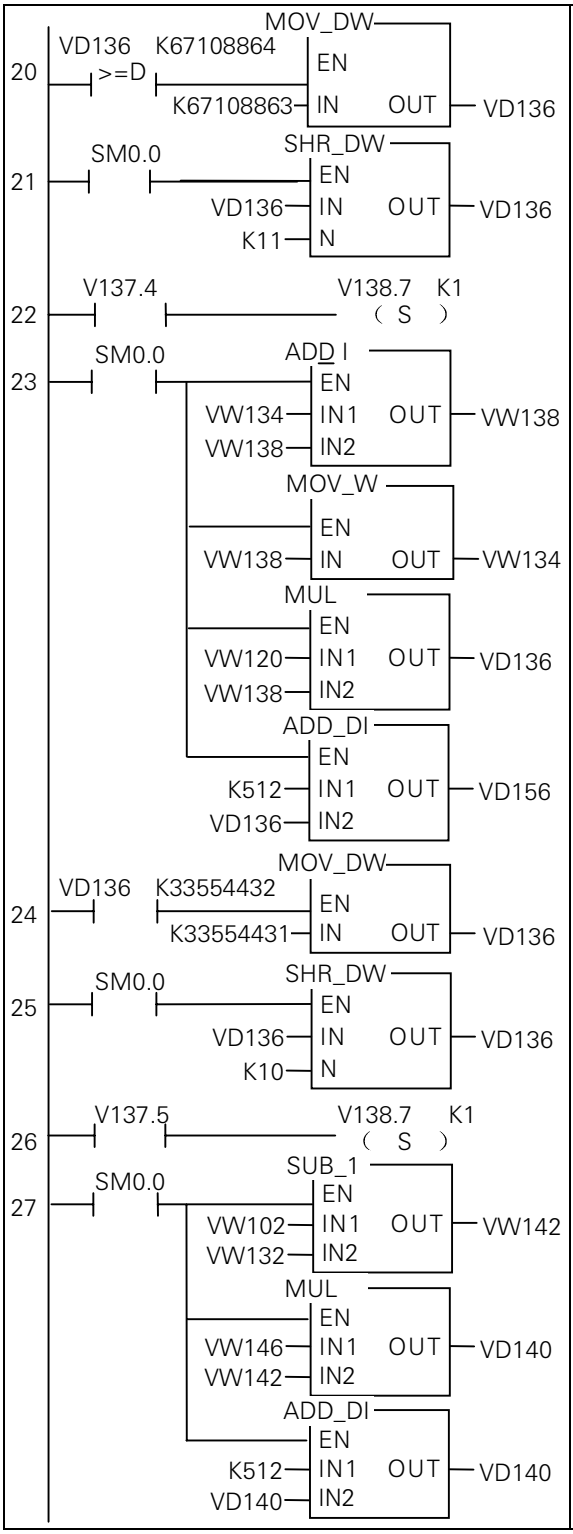

```
DIV     2，VD148          // (Tc+1)÷2
MOVW    1000,VW146        // 将1000置入微分因子缓冲器VW146
+1      VW150,VW146       // (TC+1)÷2的低端字加1000，这是为了移位
DIV     VW154,VD144       // 被Tc除
MUL     VW122,VD144       // 与微分增益相乘（1…1000）
LDD>=VD144，32768         // 检查是否上溢出，即计算值是否大于或等于2的15次方
                          // （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

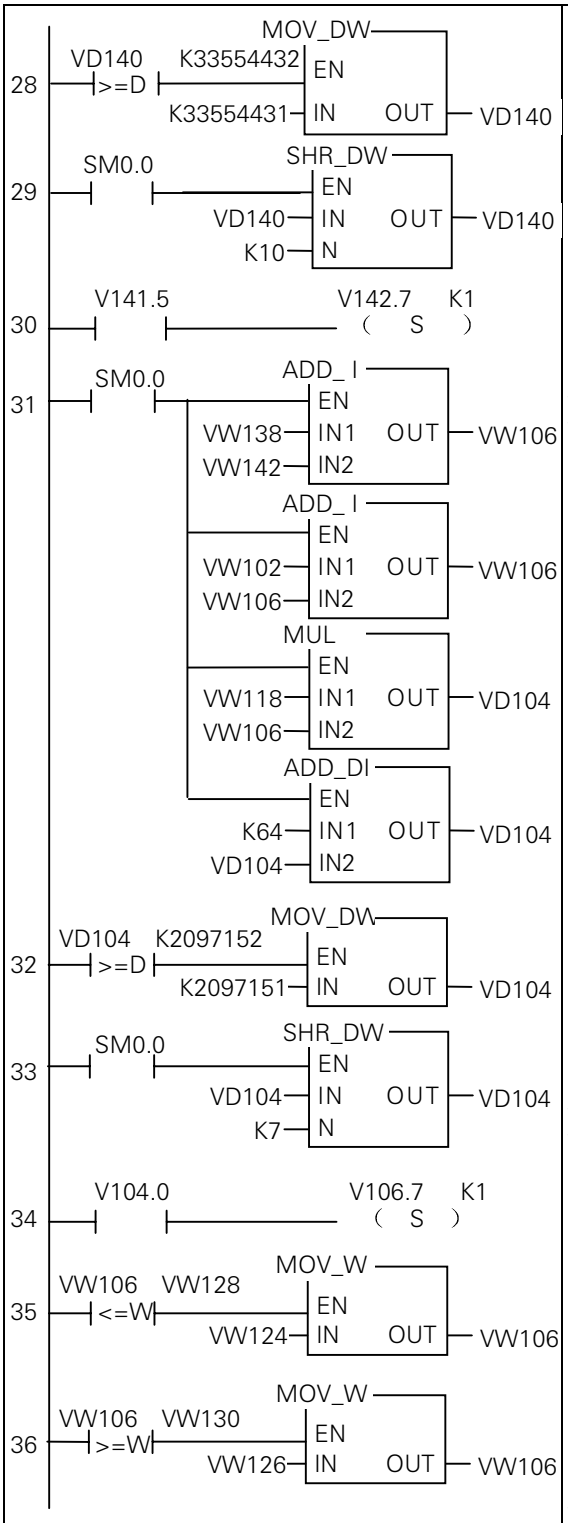
```
//＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
//功能：由定时中断0调用的PID程序
//＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
```



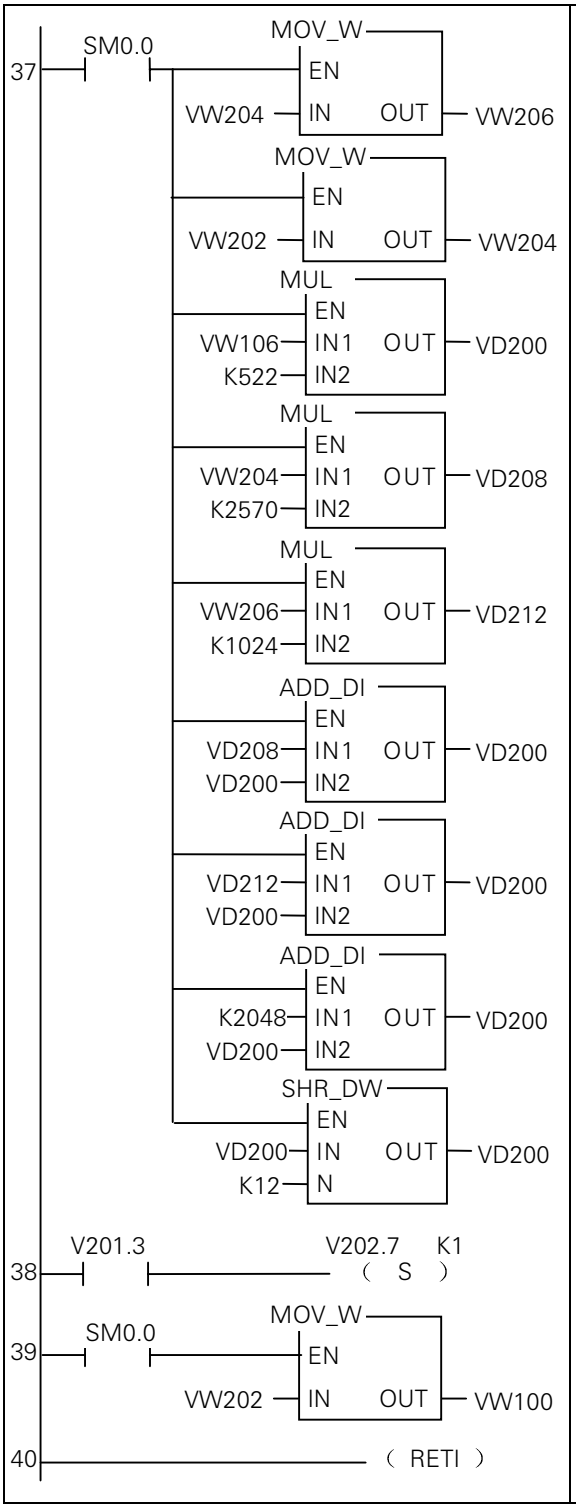

```
INT      0                  // 中断程序0

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
```



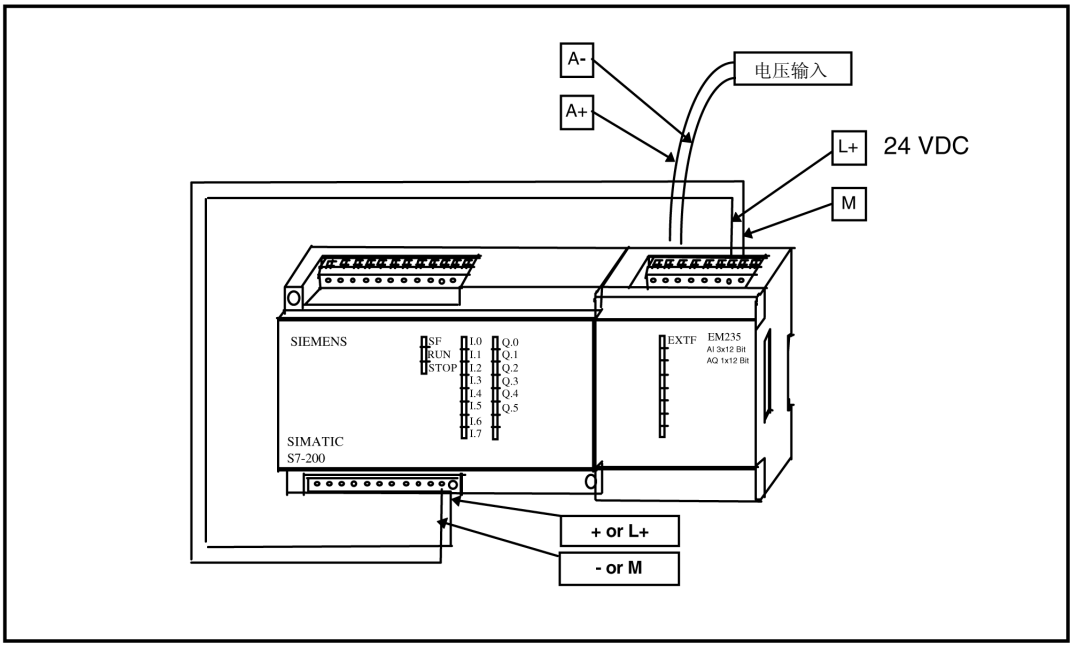

```
// 反馈输入通道
LD      SM0.0              // SM0.0总为1
MOVW    VW204，VW206       // 保存旧值，VW204存入VW206
MOVW    VW202，VW204       // 保存旧值，VW202存入VW204
MOVW    VW106，VW202       // 控制器输出值VW106存入VW202
MUL     522，VD200
MOVW    VW204，VW210
MUL     2570，VD208
MOVW    VW206，VW214
MUL     1024，VD212
+D      VD208，VD200
+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
```

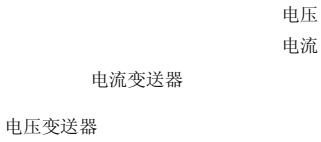
H.15 模拟量输入的处理

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





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。

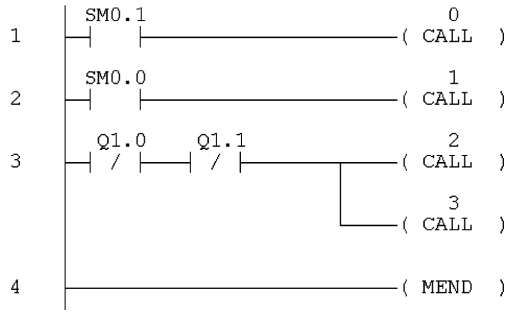
当输入数为负值时，最高有效字增添1；若为正值，最高有效字增添0来校正输入值。

本程序长度为118个字。

LAD（S7-MicroDOS）	STL(IEC)

// 主程序

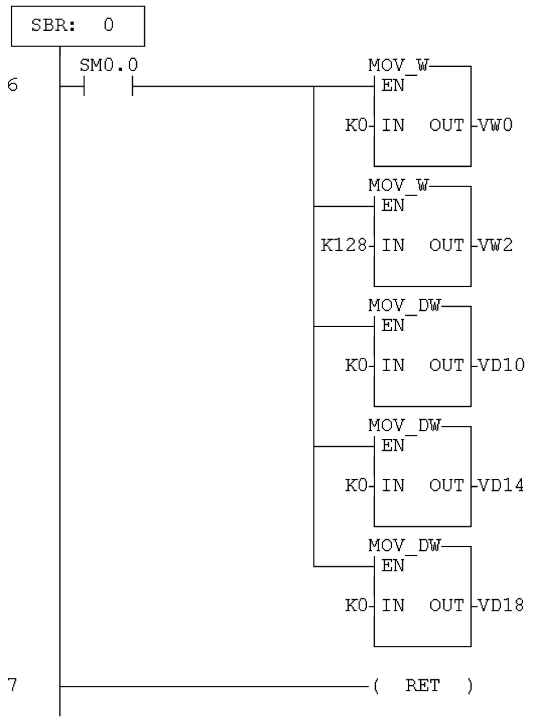
// 主程序




```
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
```

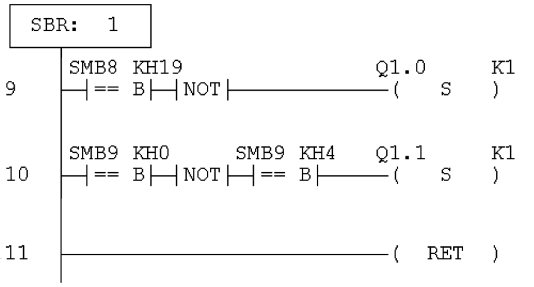
//子程序

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// 功能：初始化
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```



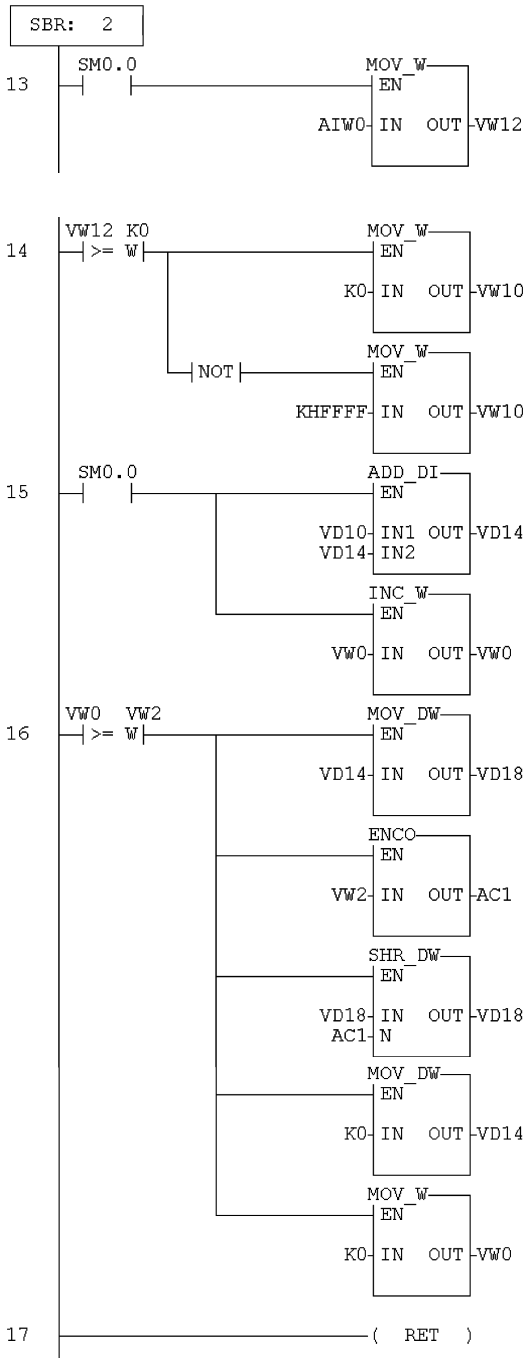

```
SBR     0           // 子程序0
LD      SM0.0       // SM0.0总为1

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
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```



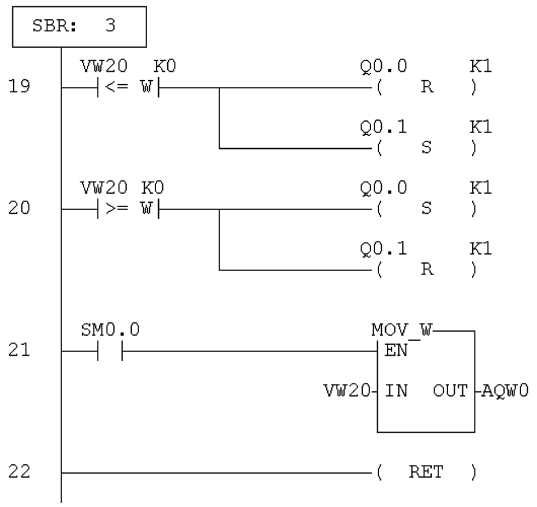

```
SBR     1               // 子程序1
LDB=    SMB8，16#19     // 检查第一个扩展模块
                           是否存在，
NOT                     //
S       Q1.0，1         // 若不存在，则置
                           Q1.0=1；
LDB=    SMB9，16#00     // 检查第一个扩展模块
                           是否有错误，
NOT                     //
AB=     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
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```



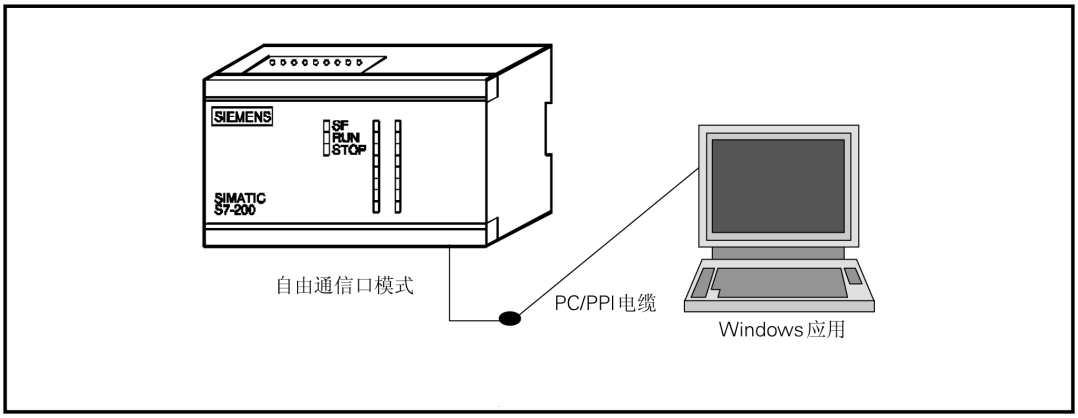

```
SBR     3           // 子程序3
LDW<=   VW20，-160  // 如果平均值为负
R       Q0.0，1     // 则关断Q0.0
S       Q0.1，1     // 且接通Q 0.1，用来显示当前平均值为负；

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

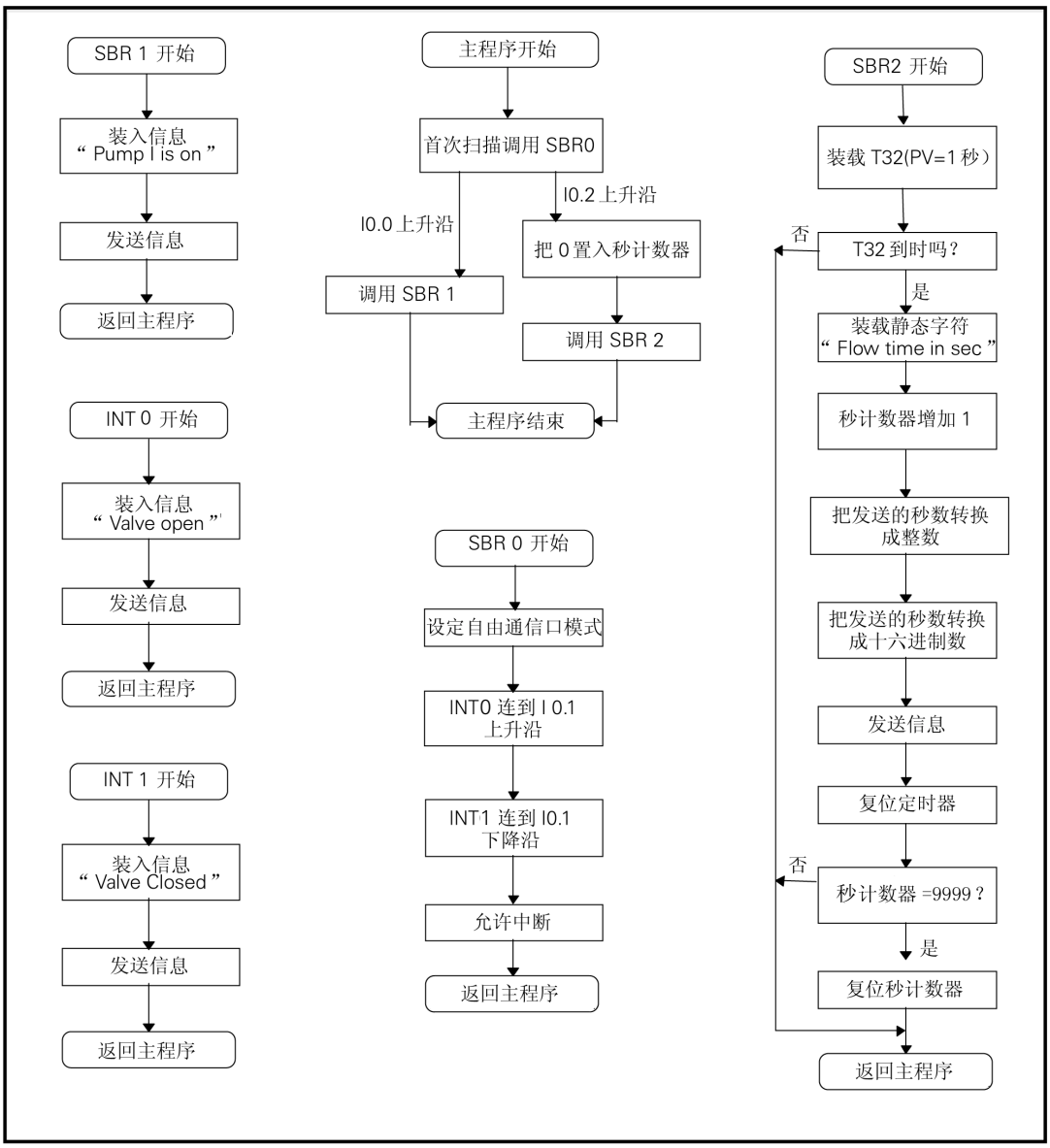
概述

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。

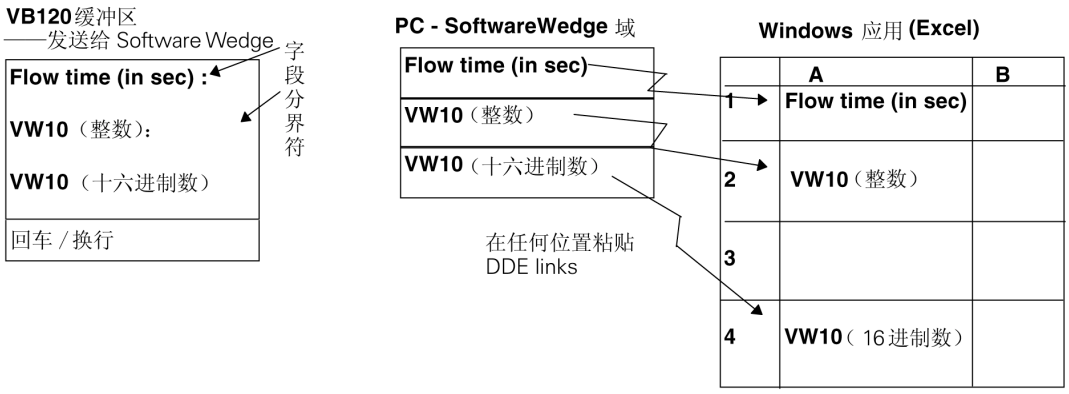
例图



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

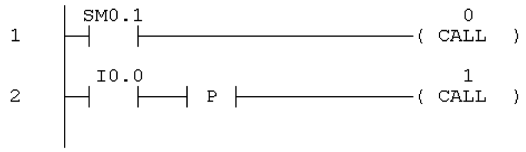
LAD (S7-MicroDOS)	STL(IEC)

主程序

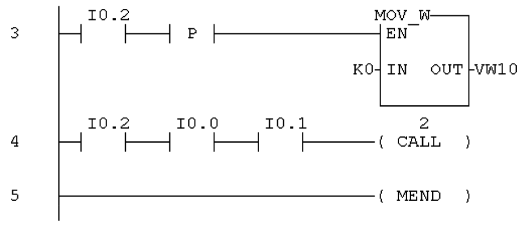
// 标题：Windows界面，并在Windows里读
// 该程序描述了利用适当的软件，S7-200系列CPU可以发送信息给任何Windows
// 应用产品（见上面）
// 本例给出了一个简单的“泵站”系统。假定I0.0为起动“主泵”的开关，I0.1为开或
// 关紧急阀的开关，I0.2为开或关主阀的开关（主阀控制液体流出）
// 操作员坐在计算机可以看到三个状态变化中的任一个信息。I0.0控制一个静态信息
// “Pump 1 is on”的出现。I0.1控制着信息“Valve open”或
// “Valve closed”交替出现。I0.2控制显示液体流出的时间，这个信息在 I0.2
// 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

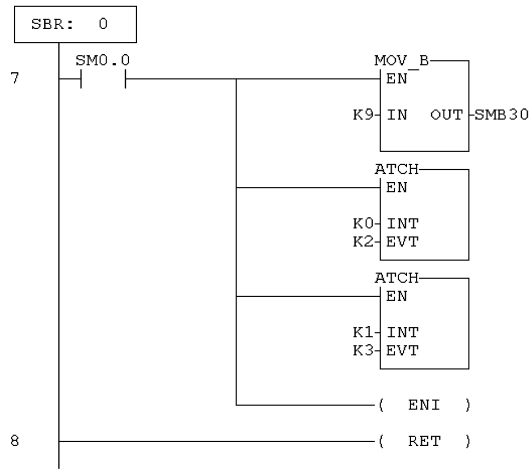



```
LD      SM0.1   // 首次扫描位 SM0.1=1
CALL    0       // 调用SBR 0
LD      I0.0    // 若主泵开关I0.0=1
EU              // 且上升沿
CALL    1       // 则调用SBR1
```



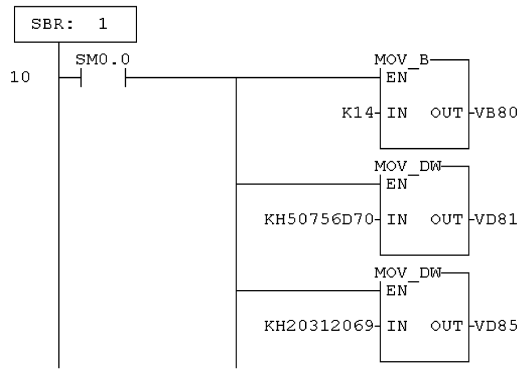

```
LD      I0.2        // 若主阀开关I0.2=1
EU                  // 且上升沿
MOVW    0，VW10     // 则定时器置0
LD      I0.2        // 若主阀开关I0.2=1
A       I0.0        // 且泵运转（I0.0=1）
A       I0.1        // 且紧急阀打开（I0.1=1）
CALL    2           // 则调用SBR2
MEND                // 主程序结束
```

子程序

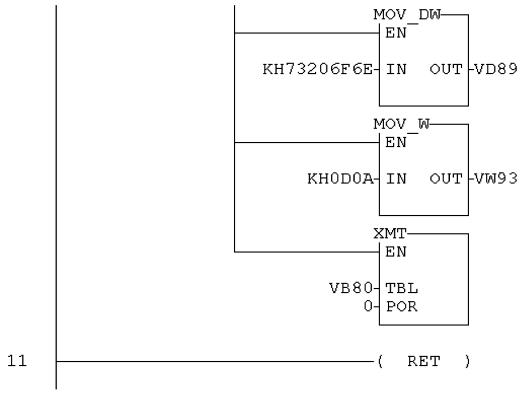



```
SBR     0           // 初始化子程序0

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
```

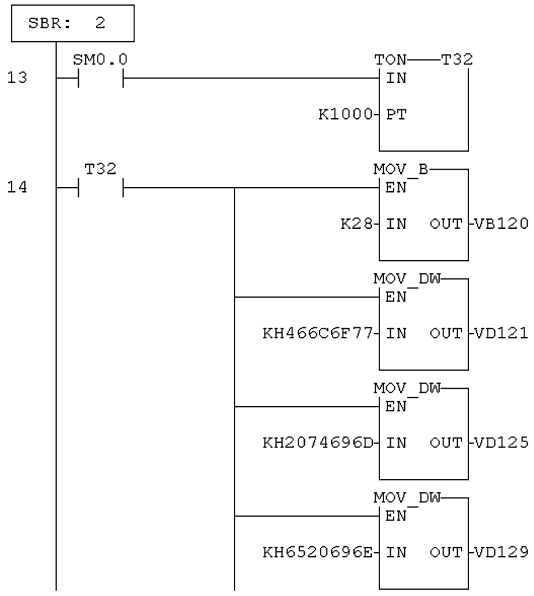



```
SBR1                // 写“Pump 1 is on”

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”
```

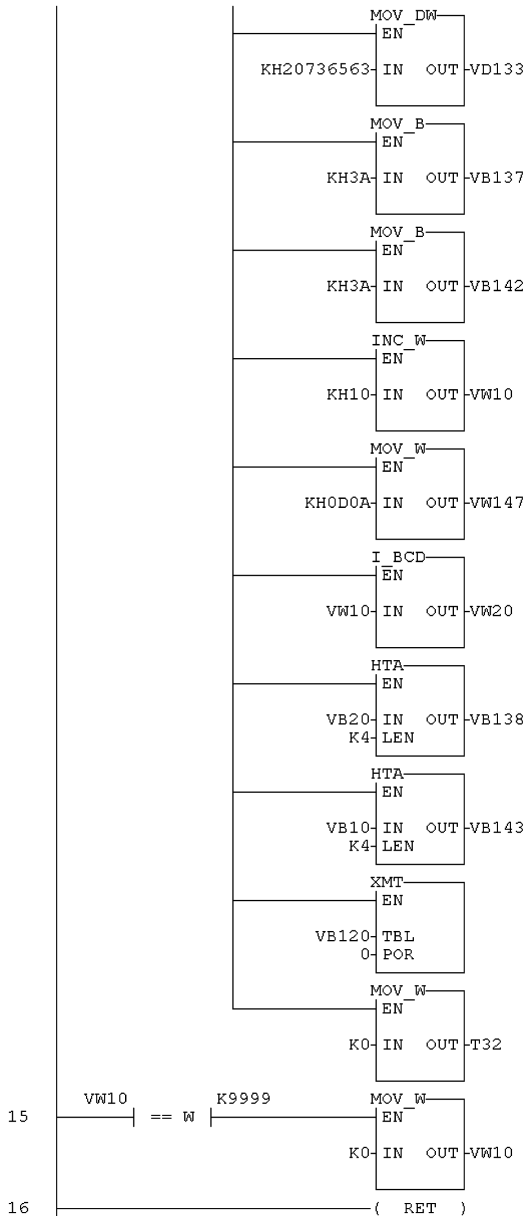



```
MOVD    16#73206F6E，VD89
                        // “s on”
MOVW    16#0D0A，VW93
                        // 回车换行表示信息结束
XMT     VB80，0         // 发送（VB80中已存放
                           所发送信息的字节数）
RET                     // 返回主程序
```

```
// 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）”
```



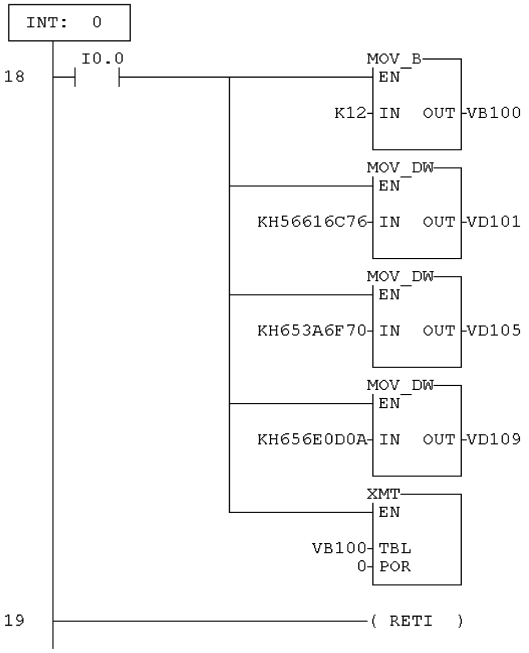

```
SBR     2               // 子程序2
LD      SM0.0           // SM0.0总为1

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
```

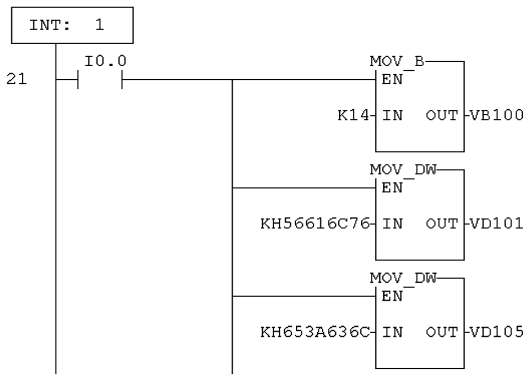
中断程序

```
// 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。
```

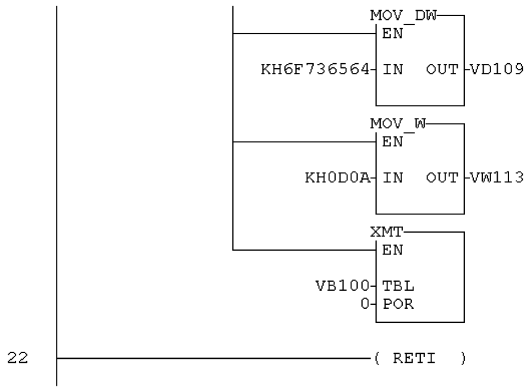



```
INT     0              // 中断程序0
LD      SM0.0          // SM0.0总为1
MOVB    12，VB100      //置发送信息长度（12个字节）
MOVD    16#56616C76，VD101
                       // “Valv”
MOVD    16#653A6F70,VD105
                       // “e：op”，“：”是字段分界符
MOVD    16#656E0D0A，VD109
                       // “en”，回车换行表示信息结束
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
```