百例成才系列丛书

西门子 S7-200 PLC 应用100例

杨后川　张瑞　高建设　曾劲松　编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

百例成才系列丛书

- 电工检修208例
- 单片机C语言应用100例
- 西门子S7-200 PLC应用100例
- 常用家电经典检修384例
- 三菱PLC应用100例
- Protel DXP 2004 应用100例

策划编辑：王敬栋
责任编辑：王凌燕

ISBN 978-7-121-08457-7
9 787121 084577 >
定价：39.80元

百例成才系列丛书

西门子 S7-200 PLC 应用 100 例

杨后川　张　瑞
高建设　曾劲松　编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书主要以西门子 S7-200 PLC 为主体，按基础知识、扩展提高和高级应用的结构体系，由浅入深、循序渐进地介绍了 PLC 基本逻辑控制、高级功能模块、PLC 网络、人机界面及工程应用等综合内容，并以实例描述的形式进行表达。内容既注重系统、全面、新颖，又力求叙述简练、层次分明、通俗易懂。在编写形式上，既注重从实际应用的角度出发，又涵盖理论知识的阐述，使读者能够针对各自不同的需求，按照对应的应用范例，快速找到解决实际问题的方法，同时也能加深对相关理论知识的了解，利于扩展思路，提高解决问题的效率。

本书可供从事 PLC 控制系统设计、开发的广大科技人员阅读，也可以作为各类高等学校工业自动化、电气工程及自动化、计算机应用、机电一体化等相关专业的参考资料。

图书在版编目（CIP）数据

西门子 S7-200 PLC 应用 100 例 / 杨后川等编著. —北京：电子工业出版社，2009.4
（百例成才系列丛书）
ISBN 978-7-121-08457-7

Ⅰ. 西… Ⅱ. 杨… Ⅲ. 可编程序控制器—基本知识 Ⅳ. TM571.6

中国版本图书馆 CIP 数据核字（2009）第 032009 号

策划编辑：王敬栋
责任编辑：王凌燕
印　　刷：北京市顺义兴华印刷厂
装　　订：三河市双峰印刷装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
开　　本：787×1 092　1/16　印张：22　字数：563.2 千字
印　　次：2009 年 4 月第 1 次印刷
印　　数：4 000 册　　定价：39.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

可编程控制器（PLC）是在计算机技术、通信技术和继电器控制技术的发展基础上开发而来的，是一种数字运算操作的电子系统。它以微处理器为核心，用编写的程序进行逻辑控制、定时、计数和算术运算等，并通过数字量和模拟量的输入/输出来控制机械设备或生产过程。

目前，PLC 已广泛应用于机械制造、冶金、化工、电力、交通、采矿、建材、轻工、环保、食品等行业，既可用于老设备的技术改造，又可用于新产品的开发。因此，对于从事工业控制研发技术人员来说，PLC 系统的设计与应用已经成为了必须掌握的一门专业技术。

西门子公司的 S7-200 系列小型 PLC 具有功能强、性价比高的特点，深受国内用户的欢迎。由于 PLC 是一门应用性很强的技术，在入门与应用上，仅仅凭借西门子公司提供的说明书是很不够的。因此本书在有关资料的基础上，以编程和工程应用实例为主旨，按基础知识、扩展提高和高级应用的顺序，循序渐进、深入浅出地介绍了多种编程方法和 PLC 在工业应用中的问题。

全书共分九章，其中 1～3 章是基础知识内容，通过实例介绍 S7-200 PLC 的基本系统、编程指令及基本控制程序；4～6 章为扩展提高内容，重点介绍 PLC 扩展应用、顺序功能图设计和实际应用的综合编程方法；7～9 章为高级应用内容，对 PLC 通信、人机界面以及综合应用进行了描述。

本书由杨后川、张瑞、高建设、曾劲松编著，参加编写的人员还有李杰和杨玉琳等。本书的第 1 章、第 4 章和第 6 章由高建设编写，第 2 章和第 7 章的第 7 节由曾劲松编写，第 3 章第 2、3 节、第 5 章第 2、3 节和附录由杨后川编写，第 7 章的 1～6 节、第 8 章和第 9 章由张瑞编写，第 3 章第 1 节由李杰编写，第 5 章第 1 节由杨玉琳。全书由杨后川副教授和张瑞博士统稿并定稿。

苏智剑教授担任本书主审。他仔细审阅了全部书稿，提出了许多宝贵的意见和建议，在此表示诚挚的谢意！

在编写过程中，作者参阅和引用了西门子公司最新技术资料和有关院校、工厂、科研院所的一些教材、文献，有些正式出版的文献已在书的参考文献中列出，有些难免遗漏，对未能列出的文献和资料，编著者向其作者表示诚挚的感谢。

由于时间仓促，加之水平有限，书中的缺点和不足之处在所难免，敬请读者批评指正。

编著者

2009 年 1 月

目　录

第 1 章　认识西门子 S7－200 PLC

- 认识西门子 PLC 的硬件
- 认识西门子 PLC 的程序开发过程
- 理解西门子 PLC 的工作原理

PLC 是 Programmable Logic Controller 的缩写，意为可编程逻辑控制器。通常也可称为 PC（Programmable Controller），即可编程控制器。它是以微处理器为基础，综合了计算机技术、半导集成体技术、自动控制技术、数字技术和通信技术而发展起来的一种通用的工业自动化控制装置，在工业生产中已获得极其广泛的应用。PLC 技术和机器人技术、CAD/CAM 技术已经成为现代工业的 3 大支柱。德国西门子公司生产的 PLC 品种齐全，功能强大，性能优越，有很高的市场认可度。其中，西门子 S7-200 PLC 是一种深受市场欢迎的小型模块化 PLC，该系列 PLC 主要由 CPU 模块和丰富的扩展模块组成。可以根据实际需要，灵活配置，再加上其强大的指令系统可以近乎完美地满足小规模系统的控制要求。

本章以 3 个典型的应用实例介绍西门子 S7-200 PLC 的硬件组成、用户程序开发过程及其工作原理。

1.1 认识西门子 PLC 的硬件

实例 1：单输入/单输出控制

实例说明

本实例主要实现指示灯的控制。通过本实例，认识西门子 PLC 主要实现的功能。

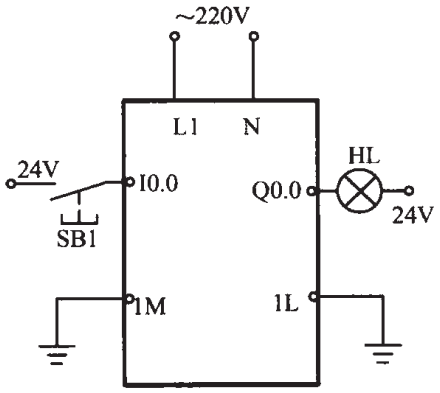


图 1-1 单输入/单输出控制系统 PLC 接线图

实例实现

如图 1-1 所示是最基本的 PLC 单输入/单输出控制系统的接线图。它由一个按钮、一个指示灯和一台 PLC 主机模块组成。最简单的情况是当按下按钮时，指示灯亮；松开按钮时，指示灯灭。

实例分析

从 PLC 的接线图中可以看出，PLC 模块连接了输入和输出，处于核心地位。

有了对 S7-200 PLC 的初步认识，下面主要介绍 S7-200 PLC 的主机模块和 I/O 接线。

1.1.1 S7-200 PLC 的主机模块

S7-200 PLC 的主机模块将一个微处理器、一个集成电源和一定数量的数字量 I/O 端子集成封装在一个独立、紧凑的设备中，从而形成了一个功能强大的微型 PLC。由于主机模块中封装了负责执行程序和存储数据的微处理器，因此也常被称为 CPU 模块。其外观面板布置如图 1-2 所示。

打开 CPU 模块的顶部端子盖可以看到电源及输出端子。CPU 模块通过电源端子获得工作电流。S7-200 PLC 可以接受交流 110 V/230 V 或直流 24 V 电源作为工作电源。需要注意的是，一个 CPU 模块的电源只能接交流电源或接直流电源。实例 1 中接的是交流 220V 的电源。揭开底部端子盖，可以看到输入端子及传感器电源。输入端子和输出端子是系统的控制点，

输入部分从现场设备（实例中的按钮）中采集信号；输出部分则控制泵、电动机及工业过程中的其他设备。

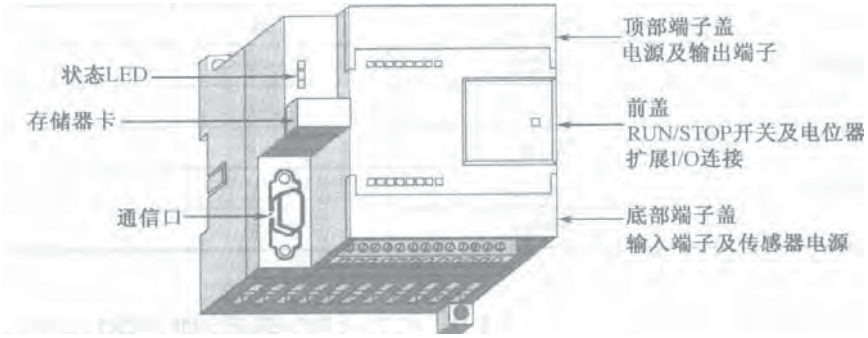


图 1-2　S7-200 CPU 模块面板布置

在前盖下面是 PLC 的工作模式（RUN/STOP）选择开关、电位器和扩展 I/O 连接端口。PLC 有 RUN 和 STOP 两种工作模式，只有在 RUN 模式时，用户编写的程序才会被执行。所以可通过模式开关来控制用户程序的执行。通过电位器可以使用户根据需要进行一些控制参数的输入。随着控制系统规模和功能的增加，一个 CPU 模块往往满足不了需要，这时可以通过扩展 I/O 连接端口进行扩展（S7-200 CPU221 除外），以提升 PLC 的控制能力和通信能力。

在 CPU 模块左上角的状态 LED 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。另外，通信端口允许将 S7-200 同编程器或其他一些设备连接起来。一些 CPU 具有内置的实时时钟，其他 CPU 则需要实时时钟卡。通过可选的插入式电池盒可延长 RAM 中的数据存储时间。通过选配 EEPROM 卡可扩展 PLC 的存储量。这些卡的使用都要通过 CPU 模块左中部的可选卡插槽来进行扩展。

S7-200 系列 PLC 主机的型号和规格较多，可以适应不同需求的控制场合。目前，该系列中主流的主机模块有 CPU221、CPU222、CPU224/ CPU224XP/ CPU224XPsi、CPU226 等模块。CPU22X 系列产品指令丰富、速度快、具有较强的通信能力。该系列主机模块的主要性能指标如表 1-1 所示。S7-200 系列 PLC 的扩展单元本身没有 CPU，只能与基本单元连接使用，用于扩展 I/O 端子数，增强控制功能。S7-200 系列 PLC 常用 I/O 扩展单元型号及输入/输出端子数的分配如表 1-2 所示。

表 1-1　S7-200 系列 PLC 主要性能指标

CPU 型号	CPU221	CPU222	CPU224	CPU224XP/XPsi	CPU226
本机数字量 I/O	6 DI/4 DO	8 DI/6DO	14 DI/10 DO	24 DI/16 DO	24 DI/16 DO
本机模拟量 I/O	—	—	—	2 AI/1 AO	—
最大数字量 I/O	6 DI/4 DO	40 DI/38 DO	94 DI/82 DO	94 DI/82 DO	128 DI/120 DO
最大模拟量 I/O	—	16	44	45	44
程序存储器容量（B）	4096	4096	12288	12288	16384
数据存储器容量（B）	2048	2048	8192	8192	10240
高速计数器通道	4（30kHz）	4（30kHz）	6（30kHz）	2（200kHz）+ 4（30kHz）	6（30kHz）
脉冲输出	2（20kHz）	2（20kHz）	2（20kHz）	2（100kHz）	2（20kHz）
最大 I/O 模块数	—	2	7	7	2
最大智能模块数	—	2	7	7	2

表 1-2 S7-200 系列 PLC 常用 I/O 扩展单元型号及输入/输出端子数的分配

类 型	型 号	输入端子	输出端子
数字量扩展模块	EM221	8	无
	EM222	无	8
	EM223	4/8/16	4/8/16
模拟量扩展模块	EM231	3	无
	EM232	无	2
	EM235	3	1

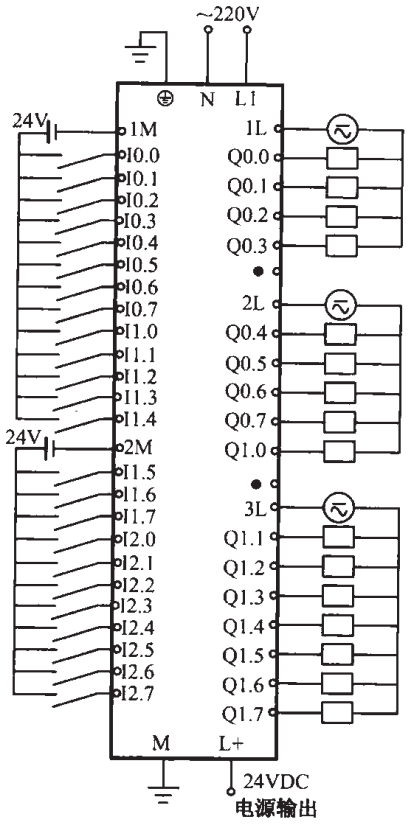


图 1-3 CPU226 AC/DC/继电器模块接线图

1.1.2 S7-200 系列 PLC 的 I/O 接线

下面以 CPU226 AC/DC/继电器模块的数字量输入、输出单元的接线为例来说明 S7-200 系列 PLC 的 I/O 接线。CPU226 指的是该 PLC 主机的型号，AC 指的是主机的电源类型是交流，DC 指的是该主机的输入模块的类型是直流的，与之相对应的还有交流输入模块。继电器指的是该主机输出模块的类型。除此之外，数字量输出模块还有直流和交流两种类型。因此，可以根据控制对象的需要灵活配置 I/O 模块的类型。如图 1-3 所示是 CPU226 AC/DC/继电器模块接线图。

该 CPU 模块共有 24 个数字量输入端子和 16 个数字量输出端子。其中 24 个输入端子被分成两组。第一组由输入端子 I0.0～I0.7、I1.0～11.4 共 13 个输入端子组成，每个外部输入的开关信号均由各输入端子接出，经一个直流电源终至公共端 1M；第二组由输入端子 I1.5～I1.7、I2.0～I2.7 共 11 个输入端子组成，每个外部输入信号由各输入端子接出，经一个直流电源终至公共端 2M。由于是直流输入模块，所以采用直流电源作为检测各输入接点状态的电源，且直流电源的极性可以任意设定。M、L+两个端子提供 DC24V/400mA 传感器电源，可以作为传感器的电源输出，也可以作为输入端的检测电源使用。16 个数字量输出端子分成三组。第一组由输出端子 Q0.0～Q0.3 共 4 个输出端子与公共端 1L 组成；第二组由输出端子 Q0.4～Q0.7、Q1.0 共 5 个输出端子与公共端 2L 组成；第三组由输出端子 Q1.1～Q1.7 共 7 个输出端子与公共端 3L 组成。每个负载的一端与输出端子相连，另一端经电源与公共端相连。由于是继电器输出方式，所以既可带直流负载，也可以带交流负载。负载的激励源由负载性质确定。输出端子排的右端 N、L1 端子是供电电源 AC110V/230V 输入端。该电源电压允许范围为 AC85～264V。

其他规格的主机模块和扩展模块的接线与之类似。

1.2　认识西门子 PLC 的程序开发过程

实例 2：电动机的启停控制

实例说明

本实例主要实现电动机的启停控制。通过本实例，认识西门子 PLC 控制系统的硬件连接及程序开发过程。

实例实现

图 1-4 是电动机启停控制的主电路，电动机启停由继电器 KM 来控制。继电器线圈的通电与否，由启动按钮（SB1）、停止按钮（SB2）通过 PLC 来控制。图 1-5 是电动机启停控制 PLC 接线图，按一下启动按钮（SB1），电动机启动；按一下停止按钮（SB2），电动机停止运转。在 PLC 控制中，还需要开发控制程序才能实现规定的控制功能。

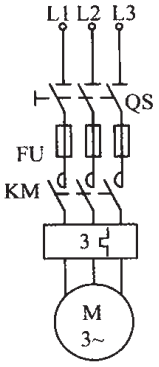


图 1-4　电动机启停控制主电路

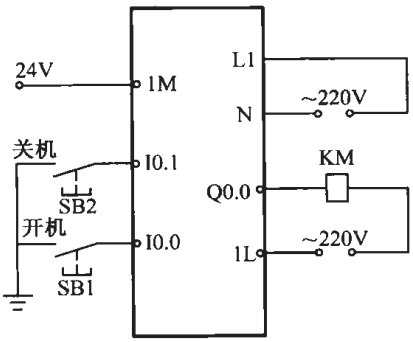


图 1-5　电动机启停控制 PLC 接线图

实例分析

电动机的启停控制的关键除了硬件接线外，还需要开发控制程序。该实例可用在对生产环境需要通风的风机控制上。如果将按钮抽象成一种条件，其应用范围更广。

1.2.1　PLC 的程序开发环境

开发 S7-200 系列 PLC 用户程序需要一台编程器，并将其和 CPU 模块连接起来。编程器可以是专用编程器，也可以是装有编程软件的 PC，后者更普遍一些。如图 1-6 所示就是一个常见的 PLC 用户程序开发系统。它由一台 PC、CPU 模块和将二者连接起来的 PC/PPI 通信电缆组成。

西门子 S7-200 系列 PLC 使用的是 STEP 7-Micro/WIN 系列编程软件。其操作界面如图 1-7 所示。各部分主要功能简介如下：

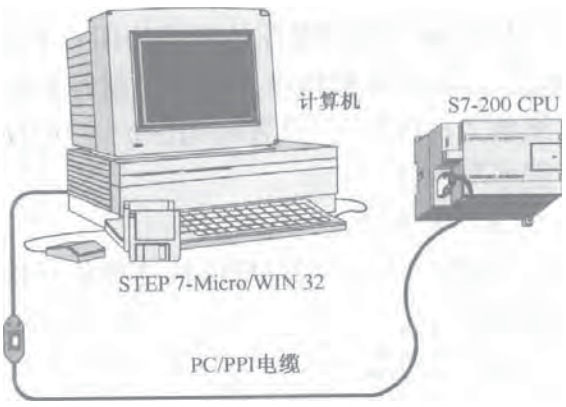


图 1-6　S7-200 PLC 用户程序开发系统

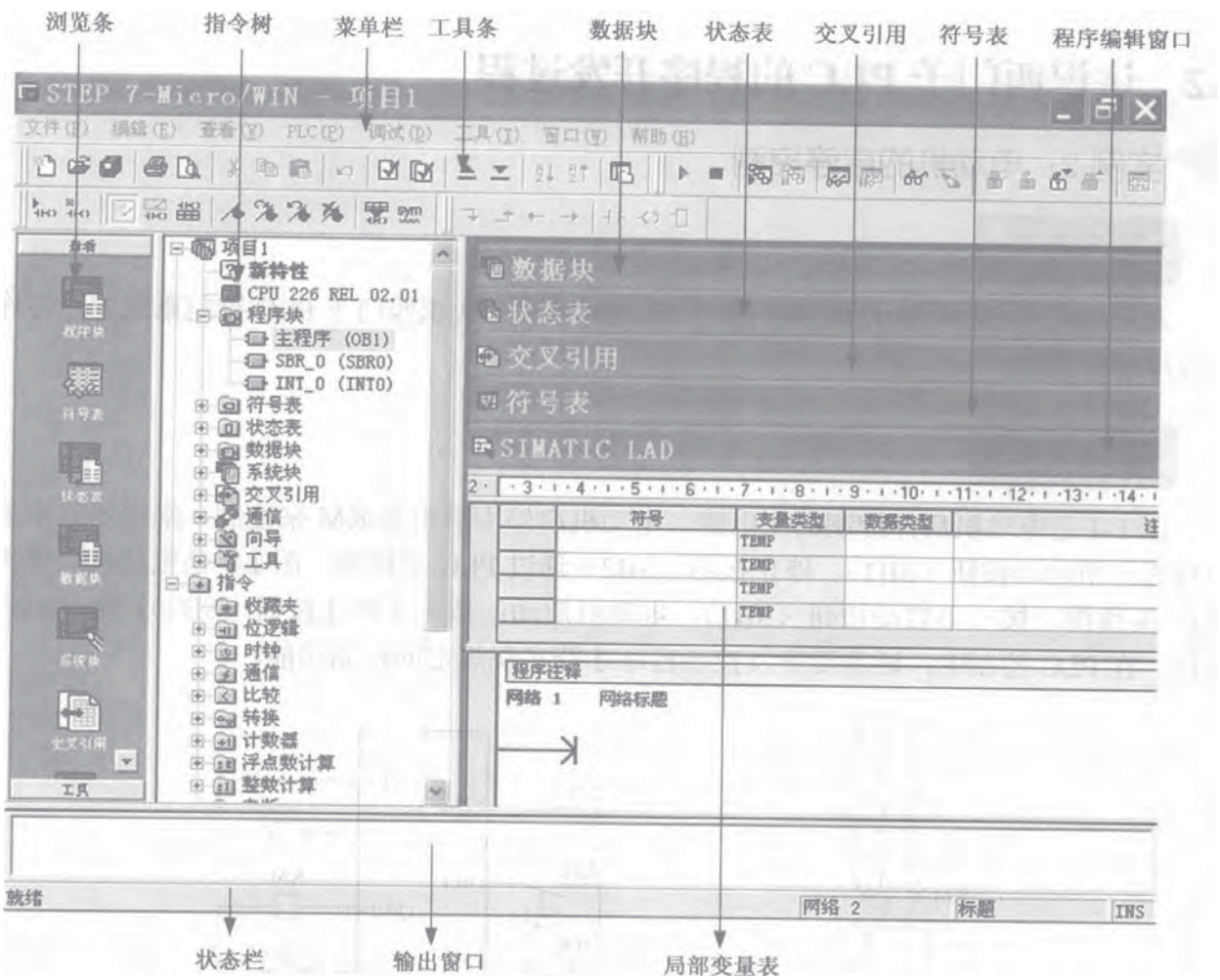


图 1-7 STEP 7-Micro/WIN 操作界面

（1）浏览条

浏览条显示编程特性的按钮控制群组。

“视图”按钮控制群中主要有程序块、符号表、状态图、数据块、系统块、交叉引用及通信显示等按钮控制。

“工具”按钮控制群中主要有显示指令向导、TD200 向导、位置控制向导、EM253 控制面板和调制解调器扩充向导等按钮控制。

（2）指令树

指令树以树形视图的形式为用户列出所有项目对象和当前程序编辑器（LAD、FBD 或 STU）所需的全部指令。通过用鼠标右键单击指令树中相应的文件夹可以进行插入附加程序组织单元（POU）、设置密码保护、打开/删除/编辑 POU 属性表，以及重新命名子程序及中断程序等操作。

（3）菜单栏

菜单栏允许使用鼠标或快捷键执行操作。可以定制“工具”菜单，在该菜单中增加自己的工具。

（4）工具条

工具条为最常用的 STEP 7-Micro/WIN 操作提供便利的鼠标存取。可以定制每个工具条的内容和外观。

（5）数据块

数据块允许用户显示和编辑数据块内容。

（6）状态表

状态图窗口允许用户将程序输入、输出或变量置入图表中，以便追踪其状态。可以建立多个状态表，以便从程序的不同部分检视组件。每个状态图在状态表窗口中有自己的标记。

（7）交叉引用

交叉引用允许用户检视程序的交叉引用和组件使用信息。

（8）符号表

符号表窗口允许用户分配和编辑全局符号（即可以在任何 POU 中使用的符号值，不只是建立符号的 POU）。可以建立多个符号表，可以在项目中增加一个 S7-200 系统符号预定义表。

（9）程序编辑窗口

程序编辑窗口包含用于该项目的编辑器（LAD、FBD 或 STL）局部变量表、程序和视图。如果需要，用户可以拖动分割条，扩充程序视图，并覆盖局部变量表。当用户在主程序一节（OBI）之外建立子例行程序或中断例行程序时，标记出现在程序编辑器窗口的底部。可单击该标记，在子程序、中断和 OBI 之间移动。

（10）状态栏

当在 STEP 7-Micro/WIN 中操作时，状态栏会提供操作状态信息。

（11）输出窗口

当编译程序或指令库时，输出窗口会提供信息。当输出窗口列出程序错误时，双击错误信息，会在程序编辑器窗口中显示适当的网络。

（12）局部变量表

局部变量表包含对局部变量所进行的赋值（即子例行程序和中断例行程序使用的变量）。在局部变量表中建立的变量使用暂时内存。地址赋值由系统处理，变量的使用仅限于建立此变量的 POU。

1.2.2　电动机启停控制程序的开发

前面对 PLC 开发的软、硬件环境进行了介绍，下面针对实例 2 进行 PLC 用户程序的实际开发。

1．建立新项目

双击“STEP 7-Micro/WIN”快捷方式图标，或者在“开始”菜单中选择“SIMATIC”→“STEP 7-Micro/WIN”命令，启动应用程序，系统自动打开一个新“STEP 7-Micro/WIN”项目，如图 1-8 所示。

2．程序输入

步骤 1：根据 PLC 接线图在符号表（Symbol Table）中输入 I/O 注释，如图 1-9 所示。

步骤 2：双击指令树中的程序块（Program Block），再双击主程序（MAIN）子项，然后在右侧的状态图窗口中逐个输入本例中的控制指令，如图 1-10 所示。

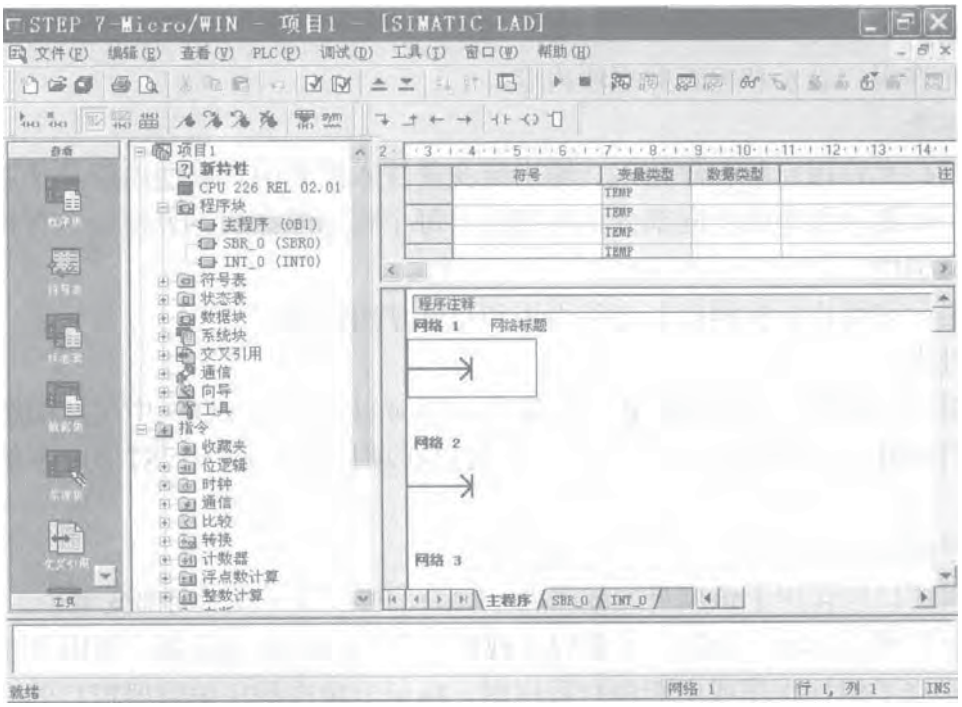


图 1-8 STEP 7-Micro/WIN 新建项目

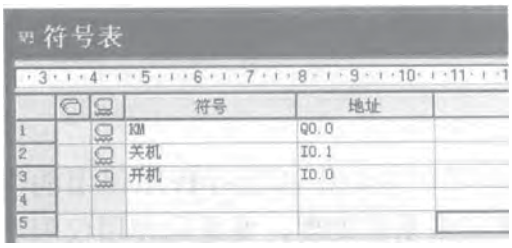


图 1-9 输入 I/O 注释

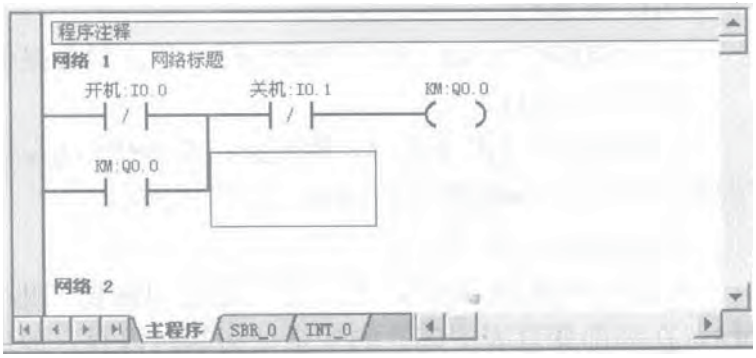


图 1-10 电动机启停控制程序

在该程序中，当按下开机按钮时，I0.0 导通，由于关机按钮是常开导通状态，此时输出端子 Q0.0 使能，接触器 KM1 线圈上电，电动机启动。同时，由于 Q0.0 使能，其常开触点闭合，所以开机按钮被屏蔽。也就是说松开开机按钮后，电动机仍保持运转。当按下关机按钮时，常闭触点 I0.1 断开，Q0.0 禁能，接触器 KM1 线圈失电，电动机停止运转。同时，常开触点 Q0.0 断开，对开机按钮的屏蔽被解除。至此一个工作周期结束。

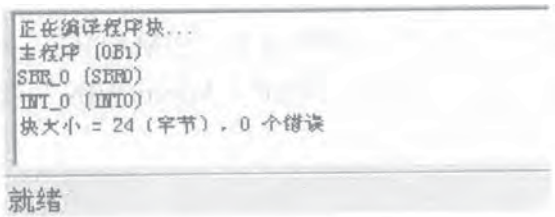


图 1-11 编译通过提示信息

程序指令输入完毕后，单击工具栏中的编译按钮进行程序编译，如果程序中有不合法的符号、错误的指令应用等情况，编译就不会通过，出错的详细信息会出现在状态栏里。可根据出错信息更正程序中的错误，然后重新编译。编译通过后状态栏里的信息提示如图 1-11 所示。

3. 程序的执行

要执行编译好的程序就要将程序传送到 PLC 中。首先将上位机软件与 PLC 主机之间的

通信建立起来，然后将编译好的程序下载到 PLC 中执行，下面是程序下载的具体步骤：

步骤 1：将 PLC 的运行模式设置为“停止”模式。可以通过工具条中的“停止”按钮，或者通过菜单选择“PLC”→“停止”命令。

步骤 2：单击工具条中的“下载”按钮，也可以通过菜单选择“文件”→“下载”命令启动下载对话窗口，如图 1-12 所示。

步骤 3：根据默认值，在初次执行下载命令时，“程序块”、“数据块”和“CPU 配置”（系统块）复选框被选择。如果不需要下载某一特定块，可以清除其对应的复选框。

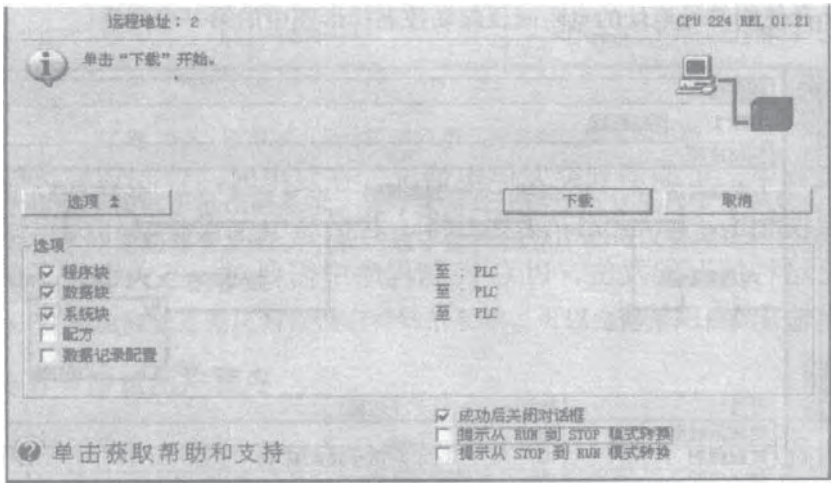


图 1-12　程序下载命令提示信息

步骤 4：单击“确定”按钮，开始下载程序。如果下载成功，弹出一个确认框显示以下信息：下载成功。

步骤 5：如果在程序开发软件 STEP 7-Micro/WIN 中设置的 PLC 类型与实际的类型不匹配，则会显示以下警告信息：“为项目所选的 PLC 类型与远程 PLC 类型不匹配。继续下载吗？”。

步骤 6：如果要更改 PLC 类型选项，选择“否”，终止程序下载。

步骤 7：从菜单中选择“PLC”→“类型”命令，调出“PLC 类型”对话框，如图 1-13 所示。在下拉菜单中选择与实际 PLC 相匹配的 PLC 类型，或者单击“读取 PLC”按钮，由软件自动读取正确的数值。单击“确定”按钮，关闭对话框。

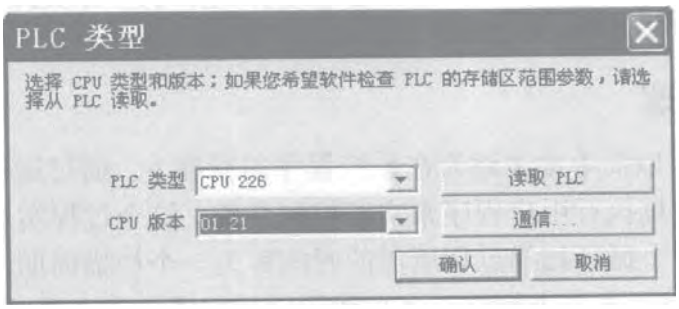


图 1-13　“PLC 类型”对话框

步骤 8：重复步骤 2 重新下载程序。

步骤 9：程序下载成功后，在运行 PLC 程序之前，将 PLC 从 STOP（停止）模式切换到

RUN（运行）模式。单击工具条中的“运行”按钮，也可以从菜单中选择“PLC”→“运行”命令，使 PLC 转换到 RUN（运行）模式。

至此，电动机启停控制程序的开发过程全部结束。本节中所涉及的 STEP 7-Micro/WIN 程序开发软件的具体应用可参考西门子 S7-200 PLC 编程手册。本实例所实现的功能在工业自动控制中很常见，读者可以举一反三，在实际工程中灵活应用。

1.3 理解西门子 PLC 的工作原理

实例 3：加电输出禁止程序

实例说明

在实际控制工程中，可能遇到突发停电情况，在复电时，控制环境可能仍处于原先得电工作状态，从而会使相应的设备立即恢复工作，这极易引发设备动作逻辑错乱，甚至发生严重事故。为了避免这种情况的发生，PLC 控制程序中需要对一些关键设备的控制端口（PLC 输出端口）做复电输出禁止控制。

实例实现

加电输出禁止程序运用了西门子 PLC 的特殊标志位存储器 SM0.3，SM0.3 为加电接通一个扫描周期，使 M1.0 置位为“1”，Q1.0 和 Q1.1 无论在 I2.0、I2.1 处于什么状态，均无输出，该程序如图 1-14 所示。

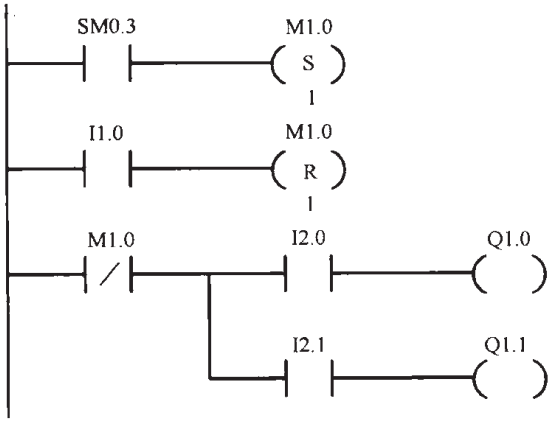


图 1-14 加电输出禁止程序

实例分析

普通继电器控制电路的工作原理告诉我们继电器控制电路梯形图中各行是并列执行的，加电输出禁止程序反映了 PLC 程序（用户程序）执行时不是并列执行的，而是按先后顺序执行的。

普通继电器控制电路的工作原理告诉我们继电器控制电路梯形图中各行是并列执行。那么 PLC 程序是如何执行的呢？要回答这个问题就需要先理解 PLC 的工作原理，这对于正确编制 PLC 控制程序是至关重要的。

1.3.1 PLC 的工作原理

PLC 的工作原理可以简单地表述为在系统程序的管理下，通过运行应用程序，对控制要求进行处理判断，并通过执行用户程序来实现控制任务。这个过程实质上是按顺序循环扫描的过程实现的。执行一个循环扫描过程所需的时间称为一个扫描周期。也就是说，在时间上 PLC 执行的任务是按串行方式进行的，其具体的运行方式与继电器-接触器控制系统及计算机控制系统都有着一定的差异。

1．循环扫描的工作原理

PLC 的一个工作过程一般有 5 个阶段：内部处理阶段、通信处理阶段、输入采样阶段、

程序执行阶段和输出刷新阶段。当 PLC 开始运行时，首先清除 I/O 映像区的内容，其次进行自诊断，然后与外部设备进行通信连接，确认正常后开始扫描。对每个用户程序，CPU 从第一条指令开始执行，按指令步序号做周期性的程序循环扫描，如果无跳转指令，则从第一条指令开始逐条执行用户程序，直至遇到结束符后又返回第一条指令，如此周而复始不断循环。因此，PLC 的工作方式是一种串行循环工作方式，如图 1-15 所示。

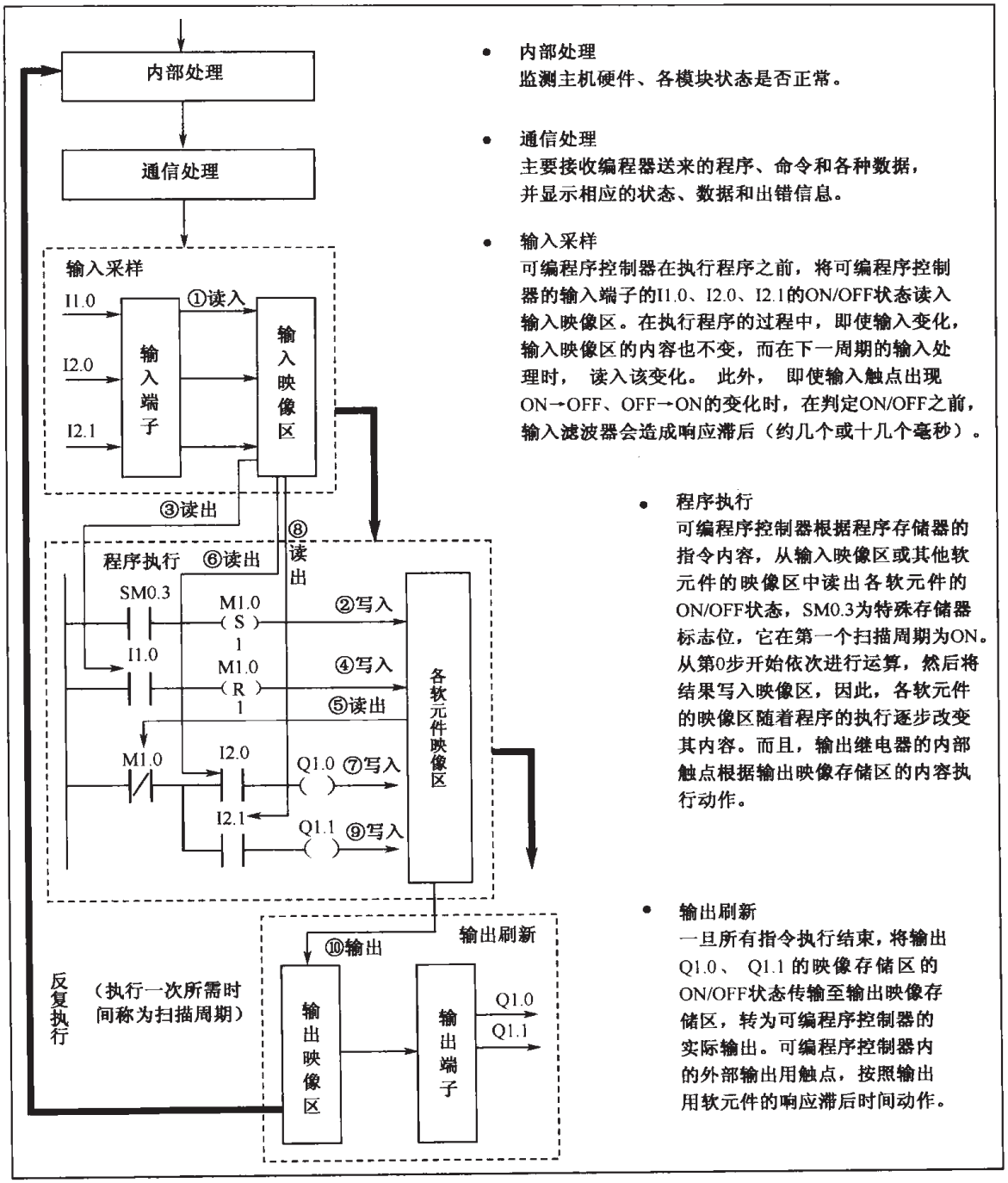


图 1-15　PLC 的循环扫描过程

（1）内部处理阶段

在这一阶段，CPU 执行监测主机硬件、用户程序存储器、I/O 模块的状态并清除 I/O 映像区的内容等工作，即 PLC 进行各种错误检测（自诊断功能），若自诊断正常，继续向下扫描。

（2）通信处理阶段

在通信处理阶段，CPU 自动监测并处理各种通信端口接收到的任何信息，即检查是否有编程器、计算机或上位 PLC 等通信请求，若有则进行相应处理，完成数据通信任务。譬如：PLC 接收编程器送来的程序、命令和各种数据，并把要显示的状态、数据、出错信息发送给编程器进行显示，这称为“监视服务”，一般在程序执行之后进行。

（3）输入采样阶段

在输入采样阶段，PLC 首先扫描所有的输入端子，按顺序将所有输入端的输入信号状态（0 或 1，表现为在接线端上是否在承受外加电压）读入输入映像寄存区。这个过程称为对输入信号的采样，或称输入刷新阶段。完成输入端刷新工作后，将关闭输入端子，转入下一步工作过程，即程序执行阶段。在程序执行期间即使输入端状态发生变化，输入状态寄存器的内容也不会发生改变，而这些变化必须等到下一个工作周期的输入刷新阶段才能被读入。

（4）程序执行阶段

程序执行阶段又称程序处理阶段，是 PLC 对程序按顺序执行的过程，对于常用的梯形图程序来说就是按从上到下、从左到右的顺序，依次执行各个程序指令。

在程序执行阶段，PLC 根据用户输入的控制程序，从第一条指令开始逐条执行，并将相应的逻辑运算结果存入对应的内部辅助寄存器（输入映像寄存器）和输出状态寄存器（输出映像寄存器）。在这个过程中，只有输入映像寄存区存放的输入采样值不会发生改变，其他各种数据，如在输出映像寄存器区或系统 RAM 存储区内的状态和数据，都有可能随着程序的执行随时发生改变。同时前面程序执行的结果可能被后面的程序所用到，从而影响后面程序的执行结果；而后面程序执行的结果不可能改变前面程序的扫描结果，只有到了下一个扫描周期再次扫描前面程序的时候才有可能起作用。但是，在扫描过程中如果遇到程序跳转指令，就会根据跳转条件是否满足来决定程序的跳转地址。当指令中涉及输入、输出状态时，PLC 从输入映像寄存器中“读入”上一阶段存入的对应输入端子状态，从输出映像寄存器“读入”对应输出映像寄存器的当前状态。然后，进行相应的运算，运算结果再存入元件映像寄存器中。对于元件映像寄存器来说，每一个元件（输出软继电器的状态）都会随着程序执行过程而变化。当最后一条控制程序执行完毕后，即转入输出刷新阶段。

（5）输出刷新阶段

当程序中所有指令执行完毕后，PLC 将输出状态寄存器中所有输出继电器的状态，依次送到输出锁存电路，并通过一定输出方式输出，驱动外部负载，这就形成了 PLC 的实际输出。

在上述 5 个阶段中，输入采样、程序执行和输出刷新是 PLC 执行用户程序的 3 个主要阶段。这 3 个阶段构成 PLC 一个工作周期，并循环执行，这就是 PLC 循环扫描工作方式的由来。由此可以总结出 PLC 在扫描过程中信号的处理规则。

2. PLC 的信号处理规则

（1）输入映像区中的数据，取决于本扫描周期输入采样阶段所处的状态。在程序执行和输出刷新阶段，输入映像区中的数据不会因为有新的输入信号而发生改变。

（2）输出映像区中的数据由程序中输出指令的执行结果决定。在输入采样和输出刷新阶段，输出映像区的数据不会发生改变。

（3）输出端子直接与外部负载连接，其状态由输出状态寄存器中的数据来确定。

3．PLC 的工作模式

西门子 S7-200 PLC 有 3 种工作模式，即运行（RUN）模式、暂停（STOP）模式和条件运行（TERM）模式。

（1）运行（RUN）模式

运行模式是执行应用程序的状态，此时不能向 PLC 写入程序。PLC 置于运行模式时，加电后，PLC 自动运行，反复执行反映控制要求的用户程序来实现控制功能直至 PLC 停机或切换到其他工作状态。PLC 处于运行（RUN）模式时，共完成 PLC 一个工作过程的 5 个阶段的操作。

（2）暂停（STOP）模式

暂停模式使 PLC 处于暂停状态，此时 PLC 仍将进行内部处理和通信处理两阶段内容，PLC 检查 CPU 模块内部的硬件是否正常，将监控定时器复位，以及完成一些其他内部工作，同时处理各种编程器的通信请求并显示相关内容。此模式一般用于程序的编制与修改。

（3）条件运行（TERM）模式

在此模式下，PLC 上的工作模式（STOP 或 RUN）可由编程装置通过通信方式来改变。此种模式多数用于联网的 PLC 网络或现场调试时使用。

4．扫描周期和响应时间

PLC 在运行状态时，执行一次扫描操作（即 5 个阶段的工作过程）所需的时间称为扫描周期，它是 PLC 的重要指标之一，其典型值为 0.5～100 ms。

扫描周期 T=（输入一点时间×输入端子数）+（指令执行速度×指令条数）+（输出一点时间×输出端子数）+故障诊断时间+通信时间

可见，扫描周期的长短主要取决于以下几个因素：CPU 执行指令的速度；执行每条指令占用的时间；程序中指令条数的多少。指令执行所需的时间与用户程序的长短、指令的种类和 CPU 执行速度有很大关系，一般来说，一个扫描过程中，故障诊断时间、通信时间、输入采样和输出刷新所占时间较少，执行指令的时间占了绝大部分。

PLC 的响应时间是指从 PLC 外部输入信号发生变化的时刻起至由它控制的有关外部输出信号发生变化的时刻之间的间隔，也称为滞后时间（通常滞后时间为几十毫秒）。它由输入电路的时间常数、输出电路的时间常数、用户语句的安排和指令的使用、PLC 的循环扫描方式及 PLC 对 I/O 的刷新方式等部分组成。这种现象称为 I/O 延迟响应或滞后现象。

由于 PLC 的这种周期循环扫描工作方式，决定了响应时间的长短与收到输入信号的时刻有关。响应时间可以分为最短响应时间和最长响应时间。

（1）最短响应时间

如果在一个扫描周期刚结束之前收到一个输入信号，在下一个扫描周期之前进入输入采样阶段，这个输入信号就被采样，使输入更新，这时响应时间最短。

（2）最长响应时间

如果收到一个输入信号经输入延迟后，刚好错过 I/O 刷新的时间，在该扫描周期内这个输入信号无效，要到下一个扫描周期输入采样阶段才被读入，使输入更新，这时响应时间最长。

由于 PLC 采用循环扫描的工作方式，即对信息串行处理方式，必定导致输入、输出延迟响应，产生滞后现象。对于一般工业控制要求，这种滞后现象是允许的。但是对那些要求响

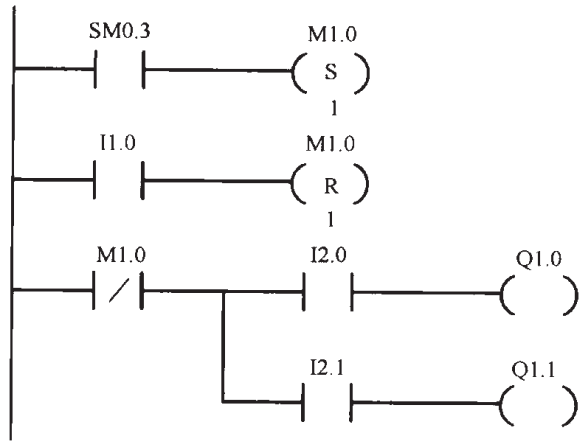
应时间小于扫描周期的控制系统则不能满足，这时可以使用智能 I/O 单元（如快速响应 I/O 模块）或专门的指令（如立即 I/O 指令），通过与扫描周期脱离的方式来解决。

1.3.2 用户程序的执行过程

在 PLC 复电进入 RUN 状态后，PLC 在自检及通信处理后，进行输入采样，而后按用户梯形图程序指令的要求，对于输出线圈按照从上到下的顺序执行，对于同一线圈按照从左到右的顺序依次执行，动作不可逆转（使用跳转指令的情况除外），最后输出刷新，之后循环往复执行，直至停止。对用户程序的执行过程的理解是设计 PLC 用户程序的关键，下面以实例 3 加电输出禁止程序为例，介绍用户程序的具体执行过程：

PLC 加电输出禁止程序循环扫描执行过程如图 1-16 所示。PLC 加电进入 RUN 状态后，SM0.3 接通一个扫描周期，使 M1.0 置位为“1”，M1.0 的常闭触点断开，从而切断了输出线圈 Q1.0、Q1.1 的控制逻辑，达到了输出被禁止的目的。当 Q1.0、Q1.1 所控制的设备准备好之后，譬如进入第 2 个循环时，可以转换 I1.0 的状态，使其为“1”，则 M1.0 被复位为“0”，对输出 Q1.0、Q1.1 的控制解除，并将控制权转移给 I2.0、I2.1，此时若 I2.0、I2.1 为“1”，Q1.0、Q1.1 置位为“1”。这样就避免了 PLC 复电后倘若 I2.0、I2.1 均处于 ON 状态导致 Q1.0、Q1.1 直接输出。

复电输出禁止程序在工程实际中经常能用到，本实例可以根据工程具体情况，稍加改造就可应用。



加电时 I2.0=1，I2.1=1

第 1 个 PLC 循环 I1.0=0	第 2 个 PLC 循环 I1.0=1
SM0.3=1，M1.0=1	SM0.3=0，M1.0=1
I1.0=0，M1.0=1	I1.0=1，M1.0=0
I2.0=1，Q1.0=0	I2.0=1，Q1.0=1
I2.1=1，Q1.1=0	I2.1=1，Q1.1=1

图 1-16 PLC 加电输出禁止程序循环扫描执行过程

思考题

1. 简述 PLC 的基本工作原理，并说明 PLC 在输入和输出的处理上有什么特点。
2. 什么是 PLC 的扫描周期？PLC 扫描周期的长短与哪些因素有关？
3. 用 STEP 7-Micro/WIN 软件编程时需要注意什么和准备什么？
4. PLC 执行用户程序的过程是如何的？

第 2 章　PLC 的指令系统

- S7-200 PLC 的基本指令
- S7-200 PLC 的功能指令

只有用户根据自己的控制要求编写用户程序并上载到 PLC 的用户程序存储器中后，PLC 才能按照用户的用途实现自动控制。用户要能准确地编写用户程序就必须熟悉编写程序时所使用的各种指令。通常把可编程控制器中所有指令的集合称为它的指令系统。

S7-200 PLC 主机中有两类指令集：IEC 1131-3 指令集和 SIMATIC 指令集，可以任选一种完成所需的控制任务。

IEC1131-3 指令集是国际电工委员会（IEC）制定的 PLC 国际标准 1131-3 Programming Language（编程语言）中推荐的标准语言，是不同 PLC 厂家的指令标准，只能用梯形图（LAD）和功能块图（FBD）编程语言编程，通常指令执行时间较长。

西门子公司遵照 IEC 标准为 S7-200 PLC 配备了 10 种 IEC 标准的基本指令，如逻辑堆栈、程序控制等。

SIMATIC 指令集是西门子公司为 S7-200 PLC 设计的编程语言，配备有 16 种非 IEC 标准的功能指令，这些指令通常执行时间短，而且可以用梯形图（LAD）、功能块图（FBD）和语句表（STL）3 种编程语言。

本章系统概括 SIMATIC 指令集中的常用指令及使用方法。

SIMATIC 指令通常由助记符和操作数组成。不同的指令，其操作数的数据类型往往是不同的。不同的 CPU 模块，由于其存储区域大小的不同使得各种数据类型的数值范围也往往是不同的。

操作数的数据类型有：位、字节（B）、字（W）、双字（D）。有关 S7-200 CPU 模块操作数的范围如表 2-1 所示。在表 2-1 中的“存取方式”栏，各字母表示的是不同的存储区域标识，其含义为：

表 2-1　S7-200 CPU 的操作数范围

存取方式		CPU221	CPU222	CPU224	CPU224XP	CPU226	示　例
位存取（字节.位）	I	0.0～15.7	0.0～15.7	0.0～15.7	0.0～15.7	0.0～15.7	I3.0
	Q	0.0～15.7	0.0～15.7	0.0～15.7	0.0～15.7	0.0～15.7	Q2.3
	V	0.0～2047.7	0.0～2047.7	0.0～8191.7	0.0～10239.7	0.0～10239.7	V2000.5
	M	0.0～31.7	0.0～31.7	0.0～31.7	0.0～31.7	0.0～31.7	M16.2
	SM	0.0～156.7	0.0～299.7	0.0～549.7	0.0～549.7	0.0～549.7	SM0.2
	S	0.0～31.7	0.0～31.7	0.0～31.7	0.0～31.7	0.0～31.7	S0.0
	T	0～255	0～255	0～255	0～255	0～255	T23
	C	0～255	0～255	0～255	0～255	0～255	C14
	L	0.0～63.7	0.0～63.7	0.0～63.7	0.0～63.7	0.0～63.7	L15.2
字节存取	IB	0～15	0～15	0～15	0～15	0～15	IB3
	QB	0～15	0～15	0～15	0～15	0～15	QB2
	VB	0～2047	0～2047	0～8191	0～10239	0～10239	VB11
	MB	0～31	0～31	0～31	0～31	0～31	MB12
	SMB	0～165	0～299	0～549	0～549	0～549	SMB132
	SB	0～31	0～31	0～31	0～31	0～31	SB10
	LB	0～63	0～63	0～63	0～63	0～63	LB15
	AC	0～3	0～3	0～3	0～255	0～255	AC2
	KB（常数）	常数	常数	常数	常数	常数	100

续表

存取方式		CPU221	CPU222	CPU224	CPU224XP	CPU226	示　例
字存取	IW	0～14	0～14	0～14	0～14	0～14	IW10
	QW	0～14	0～14	0～14	0～14	0～14	QW9
	VW	0～2046	0～2046	0～8190	0～10238	0～10238	VW100
	MW	0～30	0～30	0～30	0～30	0～30	MW21
	SMW	0～164	0～298	0～548	0～548	0～548	SMW200
	SW	0～30	0～30	0～30	0～30	0～30	SW12
	T	0～255	0～255	0～255	0～255	0～255	T24
	C	0～255	0～255	0～255	0～255	0～255	C31
	LW	0～62	0～62	0～62	0～62	0～62	LW45
	AC	0～3	0～3	0～3	0～3	0～3	AC2
	AIW	0～30	0～30	0～62	0～62	0～62	AIW12
	AQW	0～30	0～30	0～62	0～62	0～62	AQW25
	KB（常数）	常数	常数	常数	常数	常数	30000
双字存取	ID	0～12	0～12	0～12	0～12	0～12	ID
	QD	0～12	0～12	0～12	0～12	0～12	QD
	VD	0～2044	0～2044	0～8188	0～10236	0～10236	VD
	MD	0～28	0～28	0～28	0～28	0～28	MD
	SMD	0～162	0～296	0～546	0～546	0～546	SMD
	SD	0～28	0～28	0～28	0～28	0～28	SD
	LD	0～60	0～60	0～60	0～60	0～60	LD
	AC	0～3	0～3	0～3	0～3	0～3	AC
	HC	0～5	0～5	0～5	0～5	0～5	HC
	KB（常数）	常数	常数	常数	常数	常数	147483643

I：输入过程映像存储区；
Q：输出过程映像存储区；
V：变量存储区；
M：位存储区；
SM：特殊内存区（部分常用的特殊内存区含义见附录 A）；
S：顺序控制继电器存储区；
T：定时器内存区；
C：计数器内存区；
L：局部变量存储区；
AC：累加器；
AI：模拟量输入；
AQ：模拟量输出；
HC：高速计数器内存区。

2.1　S7-200 PLC 的基本指令

S7-200 PLC 的基本指令多用于开关量逻辑控制，主要包括位操作类指令、定时器和计数器指令、比较操作指令、移位操作指令、程序控制指令等，下面将一一介绍，并举例说明。

2.1.1 位操作类指令

顾名思义位操作类指令的操作数是位，主要是对 PLC 存储器中的某一位进行操作，包括位输入操作指令（也称触点指令），位输出操作指令，对位进行“与”、“或”、“非”等逻辑运算的逻辑操作指令，对位置 1 的置位指令、对位置 0 的复位指令，以及检测位发生边沿跳变的微分操作指令等。

1．触点指令

（1）标准触点指令

标准触点分为标准常开触点和标准常闭触点两种，其梯形图和语句表如图 2-1 所示。bit 的寻址范围为 I、Q、M、SM、T、C、V、S 和 L。

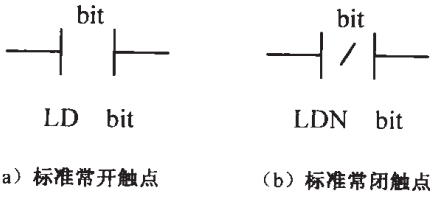


图 2-1 标准触点指令

常开触点在其寄存器位值为 0 时，其触点是断开的，触点的状态为 OFF 或 0；当寄存器位值为 1 时，其触点是闭合的，触点的状态为 ON 或 1。

常闭触点是在其寄存器位值为 0 时，其触点是闭合的，触点的状态为 ON 或 1；当其寄存器位值为 1 时，其触点是断开的，触点的状态为 OFF 或 0。

（2）立即触点指令

立即触点分为立即常开触点和立即常闭触点，其梯形图和语句表如图 2-2 所示。bit 的寻址范围为 I。

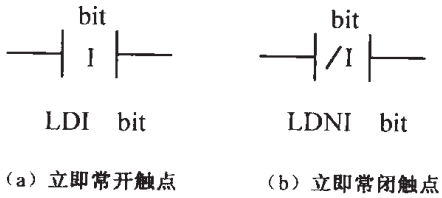


图 2-2 立即触点指令

当立即常开触点寄存器位值为 1 时，表示该触点闭合；当立即常闭触点寄存器位值为 0 时，表示该触点断开。指令中的“I”表示立即的意思。

标准触点指令与立即触点指令的差别是，执行立即指令时，CPU 直接读取其物理输入端子的值，但是不刷新相应映像寄存器的值。而执行标准触点指令时，CPU 读取的是其相应映像寄存器的值。每个从左母线开始的单一逻辑行，每个程序块（逻辑梯级）的开始，以及指令盒的输入端都必须使用标准触点指令或立即触点指令，如果实际编写程序中没有相应的标准触点或立即触点，则可以加上特殊内存器 SM0.0 这一位来实现。在程序执行过程中，标准触点和立即触点起开关作用。

2．输出指令

（1）输出操作指令

输出操作指令的梯形图和语句表如图 2-3 所示。bit 的寻址范围为 I、Q、M、SM、T、C、V、S 和 L。

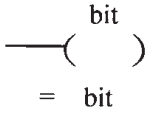


图 2-3 输出操作指令

输出操作指令将输出位的新数值（前面各逻辑运算的结果）写入输出映像寄存器，并根据写入结果控制其对应的触点。

（2）立即输出操作指令

立即输出操作指令的梯形图和语句表表示如图 2-4 所示。bit 的寻址范围只能为 Q。

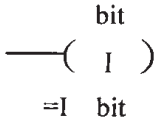


图 2-4 立即输出操作指令

立即输出操作指令将输出位的新数值（前面各逻辑运算的结果）立即写入到输出映像区，同时还直接驱动实际输出。也就是说使用立即输出操作指令时，输出的结果不受 PLC 扫描周期的限制，而在运行到该指令时直接驱动实际输出。

实例 4：位的设置

实例说明

通过本实例来说明如何对位进行设置。

实例实现

位设置的程序梯形图和指令如图 2-5 所示，特殊继电器 SM0.0 为上电后一直通电的继电器，这样在上电后一直给 Q0.1 置 1。

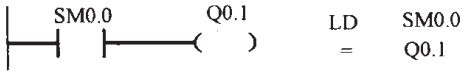


图 2-5　位设置程序

图 2-5 所对应的语句为：

```
LD      SM0.0
=       Q0.1
```

实例分析

在 PLC 的控制系统中，需要有一个指示灯显示 PLC 在正常运行。这样需要在 PLC 开机后，一直给某一位输出一个接通信号从而保证指示灯点亮。要实现这一功能就可以应用此实例。

3. 逻辑指令

（1）逻辑与操作指令

逻辑与操作指令梯形图由标准触点或立即触点串联构成。

逻辑与操作指令语句表为："A　bit"；"AN　bit"；"AI　bit"；"ANI　bit"。bit 的寻址范围为 I、Q、M、SM、T、C、V、S 和 L。

逻辑与只有当两个触点的状态都是 1（ON）时才有输出，两者中只要有一个为 0（OFF），就没有输出。

（2）逻辑或操作指令

逻辑或操作指令梯形图由标准触点或立即触点并联构成。

逻辑或操作指令语句表为："O　bit"；"ON　bit"；"OI　bit"；"ONI　bit"。bit 的寻址范围为 I、Q、M、SM、T、C、V、S 和 L。

逻辑或只要两个触点中有一个触点的状态是 1（ON）就有输出，只有当两者都为 0（OFF）时才没有输出。

（3）逻辑非操作指令

逻辑非操作指令梯形图和语句表如图 2-6 所示。

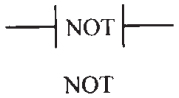


图 2-6　逻辑非操作指令

逻辑非操作就是把源操作数的状态取反作为目标操作数输出。当操作数的状态为 1（ON）时，取非后就为 0（OFF）；当操作数的状态为 0（OFF）时，取非后就为 1（ON）。逻辑非

操作只能与其他指令联合使用，本身没有操作数。

（4）串联电路的并联操作指令

串联电路的并联操作指令梯形图，是由多个触点串联构成一条支路，一系列这样的支路再相互并联构成复杂电路，即把多个“与”逻辑运算结果进行“或”的逻辑运算。指令在执行时，先算出各个“与”逻辑的结果，然后再把这些结果进行“或”逻辑运算后传送到输出。

串联电路的并联操作指令语句表表示：“OLD”（在并联第二个支路语句的后面用）。

（5）并联电路的串联操作指令

并联电路的串联操作指令梯形图，是由多个触点并联构成局部电路，一系列这样的局部电路再相互串联构成复杂电路，即把多个“或”逻辑运算结果进行“与”的逻辑运算。指令在执行时，先算出各个“或”逻辑的结果，然后再把这些结果进行“与”逻辑运算后传送到输出。

并联电路的串联操作指令语句表表示：“ALD”（在串联第二个支路语句的后面用）。

实例 5：电动机优先控制

实例说明

通过本实例来说明 PLC 是如何通过并联、串联实现逻辑控制的。

实例实现

在工业控制中，经常需要多个电动机按照一定的顺序分别启动和停止，要实现这样的功能就需要采用 PLC 的逻辑控制。

有 3 个电动机 M1～M3，电动机的启、停按钮及相应的控制寄存器与 PLC 的连接如表 2-2 所示，其控制程序梯形图和指令如图 2-7 所示。

表 2-2　电动机优先控制电路中 PLC 接口连接表

PLC 端口	I0.0	I0.1	Q0.0
电动机 M1 的控制口	启动按钮	停止按钮	电动机控制继电器
PLC 端口	I0.2	I0.3	Q0.1
电动机 M2 的控制口	启动按钮	停止按钮	电动机控制继电器
PLC 端口	I0.4	I0.5	Q0.2
电动机 M3 的控制口	启动按钮	停止按钮	电动机控制继电器

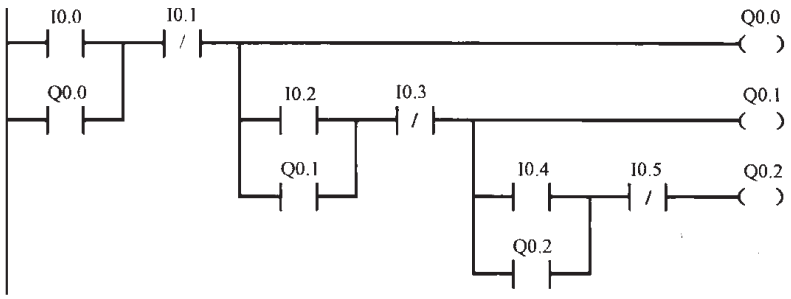


图 2-7　电动机优先控制程序

图 2-7 中程序的语句为：

```
LD    I0.0
O     Q0.0
```

```
AN      I0.1
=       Q0.0
LD      I0.2
O       Q0.1
AN      I0.3
ALD
=       Q0.1
LD      I0.4
O       Q0.2
ALD
AN      I0.5
=       Q0.2
```

实例分析

通过这样的控制就可以实现电动机的顺序启动，即前级电动机不启动时，后级电动机无法启动；前级电动机停止时，后面各级电动机也都同时停止。

4. 置位、复位操作指令

置位、复位操作指令也分为立即和非立即两种，其梯形图和语句表如图 2-8 所示。非立即置位、复位指令的 bit 的寻址范围为 I、Q、M、SM、T、C、V、S 和 L；立即置位、复位指令的 bit 的寻址范围为 Q；n 的寻址范围为 VB、IB、QB、MB、SMB、SB、LB、AC、常数、*VD、*AC 和*LD（“*”表示的是间接寻址）。

（a）非立即置位操作指令　（b）非立即复位操作指令　（c）立即置位操作指令　（d）立即复位操作指令

图 2-8　置位、复位操作指令

置位即置 1，复位即置 0。置位和复位指令可以将位存储区某一位开始的一个或多个（最多可达 255 个）同类存储器位置 1 或置 0。

当置位（复位）信号来临（1 或 ON）时，被置位置 1（被复位置 0），即使置位（复位）信号变为 0 以后，被置位（被复位）的状态仍然可以保持，直到使其复位（置位）信号的到来。在执行置位操作指令时，注意被置位的数目应是从指令中指定的位地址 bit 开始，共有 n 个。

执行立即指令，新值被同时写到物理输出端子和相应的映像寄存器，而非立即指令仅仅把新值写到映像寄存器。

实例 6：置位/复位指令实现电动机的启停控制

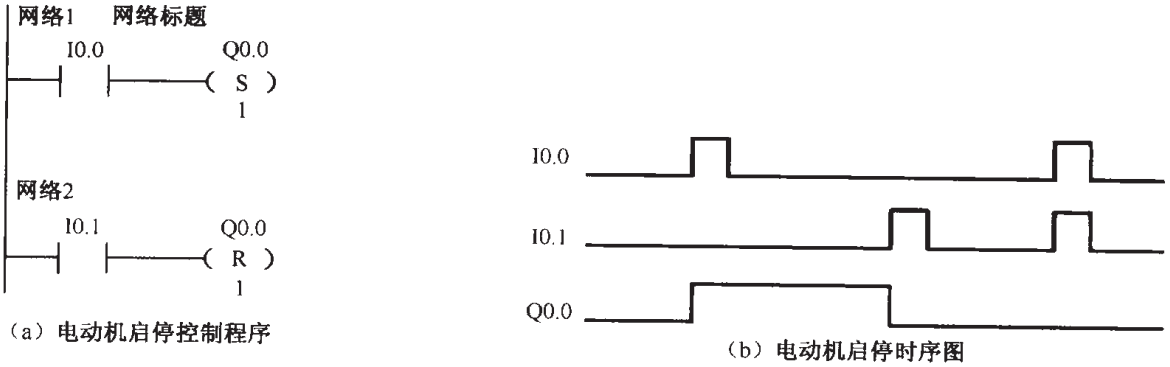
实例说明

通过本实例来说明 PLC 在执行置位和复位后相应的映像寄存器的变化。

实例实现

如图 2-9 所示的梯形图可以实现对 Q0.0 的置位与复位，根据 I0.0 与 I0.1 的状态，Q0.0 被相应的置位与复位，从而也就可以得到相应的时序。图 2-9 所示的梯形图所对应的语句为：

```
LD    I0.0
S     Q0.0 ,1
LD    I0.1
R     Q0.0 ,1
```



（a）电动机启停控制程序　　（b）电动机启停时序图

图 2-9　电动机启停控制程序与时序

实例分析

这一实例经常应用于 PLC 控制电动机系统中。在机电控制系统中需要使用按钮来控制电动机的启停。按下启动按钮，电路会瞬时接通，若没有自保持电路，松手后按钮电路会断开。但若要求电动机在按钮松开后电动机仍然运转，同时要求在按下停止按钮后电动机停止，就要使用置位、复位指令来实现这一功能了。

按图 2-9 所示的程序并将电动机的启停控制口按下面分配：I0.0－电动机启动按钮；I0.1－电动机停止按钮；Q0.0－电动机控制继电器，这样就可以实现电动机的起停控制了。

5. 微分指令

（1）上微分操作指令

上微分操作指令的梯形图和语句表如图 2-10(a)所示。

（a）上微分操作指令　　（b）下微分操作指令

图 2-10　微分操作指令

上微分是指某一位操作数的状态由 0 变为 1 的过程，即出现上升沿的过程。上微分操作指令的功能是在这个上升沿形成一个保持一个扫描周期的脉冲。接受这一脉冲控制的器件应写在这一脉冲出现的语句之后。

（2）下微分操作指令

下微分操作指令的梯形图和语句表如图 2-10（b）所示。

下微分是指某一位操作数的状态由 1 变为 0 的过程，即出现下降沿的过程。下微分操作指令的功能是在这个下降沿形成一个保持一个扫描周期的脉冲。接受这一脉冲控制的器件应写在这一脉冲出现的语句之后。

实例 7：输入信号的边沿检测

实例说明

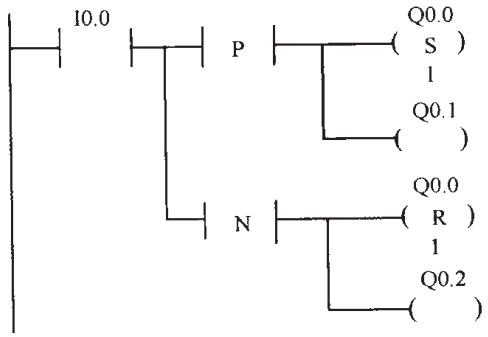
通过本实例来说明微分指令的应用。

实例实现

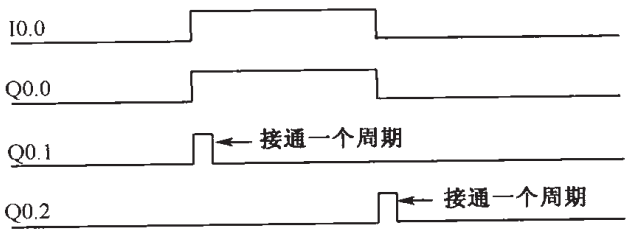
如图 2-11 所示的梯形图是将输入信号的跳变转换为脉冲，并利用输入信号的跳变实现输出信号的置位、复位控制。根据 I0.0 的状态变化，Q0.1 与 Q0.2 会产生一个扫描周期的脉冲信号，从而实现 Q0.0 的复位与置位。根据这些寄存器的关系可以得到相应的时序。

图 2-11 所示的梯形图对应的语句为：

```
LD      I0.0
LPS
EU
S       Q0.0 , 1
=       Q0.1
LPP
ED
R       Q0.0 , 1
=       Q0.2
```



（a）输入信号边沿检测程序



（b）输入信号边沿检测时序图

图 2-11　输入信号边沿检测程序与时序

实例分析

在 PLC 控制系统中，经常会用到检测输入信号的变化，也就是检测输入信号的跳变，输入信号的跳变包括由 1 变为 0 的下降沿和由 0 变为 1 的上升沿。利用本例可以将输入信号的跳变转换为脉冲，并利用输入信号的跳变实现输出信号的置位、复位控制的例子。

2.1.2 定时器和计数器指令

1. 定时器指令

定时器由集成电路构成，是 PLC 中重要的硬件编程元件。定时器编程时要先给出输入时间预设值。当定时器的输入条件满足时，定时器开始计时，当前值从 0 开始按一定的时间单位增加，当定时器的当前值达到预设值时，定时器发生动作，发出中断请求，以便 PLC 响应而做出相应的动作。利用定时器的输入与输出触点就可以得到控制所需的延时时间。

S7-200 PLC 为用户提供了 3 种类型的定时器：接通延时定时器（指令为 TON）、保留性接通延时定时器（指令为 TONR）和断开延时定时器（指令为 TOF），共 256 个（号码为 T0～T255）。

S7-200 PLC 定时器的分辨率有 3 个等级：1 ms、10 ms 和 100 ms，分辨率等级和定时器号码对应关系如表 2-3 所示。虽然接通延时定时器与断开延时定时器的编号范围相同，但是不能共享相同的计时器号。例如，在对同一个 PLC 进行编程时，不能既有 TON32，又有 TOF32。

从定时器的原理可以知道定时时间的计算公式为：

$$T = PT \times S$$ （T 为定时时间，PT 为预设值，S 为分辨率等级）

例如：TON 指令用定时器 T37，预设值为 120，则实际定时时间为：

$$T=120\times100\ \text{ms}=12000\ \text{ms}。$$

表 2-3 定时器的类型、分辨率和编号

定时器类型	分 辨 率	最 大 值	定时器号码
保留性接通延时定时器（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
接通延时定时器（TON） 断开延时定时器（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

定时器号码不仅仅是定时器的编号，它还包含两方面的变量信息：定时器位和定时器当前值。

定时器位：存储定时器的状态，当定时器的当前值达到预设值 PT 时，该位发生动作。

定时器当前值：存储定时器当前所累计的时间，它用 16 位符号整数来表示，故最大计数值为 32767。

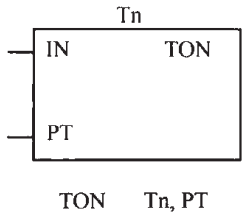


图 2-12 接通延时定时器指令

（1）接通延时定时器指令

接通延时定时器指令的梯形图和语句表如图 2-12 所示。TON 为定时器标识符，Tn 为定时器编号，IN 为启动输入端（数据类型为 BOOL 型），PT 为时间设定值输入端（数据类型为 INT 型）。接通延时定时器用于单一时间间隔的定时。

其具体工作过程是，当定时器的输入信号 IN 的状态为 0 时，定时器的当前值为 0，定时器位也为 0（常开触点断开，常闭触点闭合），定时器没有

工作。当输入信号由 0 变为 1 时，定时器开始工作，然后每过一个基本时间间隔，定时器的当前值加 1。当定时器的当前值等于或大于定时器的设定值 PT 时，定时器的延时时间到了，这时定时器位由 0 转换为 1（常开触点闭合，常闭触点断开）。在定时器输出状态改变后，定时器继续计时直到定时器值为 32767（最大值）时，才停止计时，当前值将保持不变。只要当前值大于 PT 值，定时器位就为 1，如果不满足这个条件，定时器位应为 0。

当 IN 信号由 1 变为 0，则定时器当前值复位（置为 0），定时器位也为 0。当 IN 从 0 变为 1 后维持的时间不足以使得当前值达到 PT 值时，定时器位也不会由 0 变为 1。

（2）保留性接通延时定时器指令

保留性接通延时定时器指令的梯形图和语句表如图 2-13 所示。TONR 为定时器标识符，Tn 为定时器编号，IN 为启动输入端，PT 为时间设定值输入端。

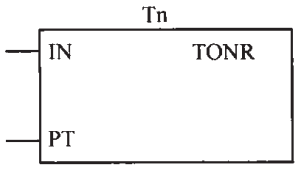


图 2-13　保留性接通延时定时器指令

保留性接通延时定时器用于对许多间隔的累计定时。

保留性接通延时定时器的原理与接通延时定时器基本相同。不同之处在于保留性接通延时定时器的当前值在 IN 从 1 变为 0 时，定时器位和当前值保持下来。当 IN 再次从 0 变为 1 时，当前值从上次的保持值继续计数，当累计当前值达到预设值时，定时器位为 1。当前值连续计数到 32767，才停止计时。因而保留性接通延时定时器的复位不能同普通接通延时定时器的复位那样使用 IN 从 1 变为 0，而只能使用复位指令 R 对其进行复位操作，使当前值清零。

（3）断开延时定时器指令

断开延时定时器指令的梯形图和语句表如图 2-14 所示，TOF 为定时器标识符，Tn 为定时器编号，IN 为启动输入端，PT 为时间设定值输入端。

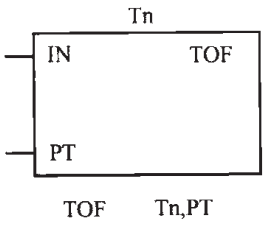


图 2-14　断开延时定时器指令

断开延时定时器用于断电后的单一间隔时间计时。

当定时器的输入信号 IN 的状态为 1 时，定时器的当前值为 0，定时器位为 1，定时器没有工作。只有当启动信号由 1 变为 0 时，定时器开始工作，每过一个基本时间间隔，定时器的当前值加 1。当定时器的当前值达到定时器的设定值 PT 时，到了定时器的延时时间，这时定时器位由 1 转换为 0，停止计时，当前值保持不变。

当 IN 信号由 0 变为 1，则当前值复位（置为 0），定时器位为 1。当 IN 从 1 变为 0 后维持的时间不足以使得当前值达到 PT 值时，定时器位也不会由 1 变为 0。

对于这 3 类定时器，其操作数的范围是相同的：输入信号 IN 的寻址范围为 I、Q、M、SM、T、C、V、S 和 L；PT 的寻址范围为 VW、IW、QW、MW、SW、SMW、LW、AIW、T、C、AC、常数、*VD、*LD 和*AC。

另外，由于 PLC 的工作过程是受扫描周期影响的，为了在保证定时精度的同时又能更快的执行 PLC 程序，西门子 S7-200 PLC 对于不同精度的定时器，当前值的刷新周期是不同的：

1 ms 分辨率定时器启动后，定时器对 1 ms 的时间间隔（时基信号）进行计时。定时器位和当前值每隔 1 ms 刷新一次，因而 1 ms 分辨率定时器的刷新和扫描周期是不同步的，在

一个扫描周期中要刷新多次。

10 ms 分辨率定时器启动后，定时器对 10 ms 的时间间隔进行计时。程序执行时，在每次扫描周期开始对 10 ms 定时器刷新，在一个扫描周期内定时器位和当前值保持不变，在扫描期间积聚的时间间隔在每次扫描开始时添加到当前值。

100 ms 分辨率定时器启动后，定时器对 100 ms 的时间间隔进行计时。只有在定时器指令执行时，100 ms 定时器位和当前值才被刷新。

在子程序和中断程序中不易使用 100 ms 定时器。子程序和中断程序不是每个扫描周期都执行的，那么在子程序和中断程序中的 100 ms 定时器的当前值就不能及时刷新，造成时基脉冲丢失，致使计时失准。在主程序中，不能重复使用同一个 100 ms 的定时器号，否则该定时器指令在一个扫描周期中多次被执行，定时器的当前值在一个扫描周期中多次被刷新。这样，定时器就会多计了时基脉冲，同样造成计时失准。因而，100 ms 定时器只能用于每个扫描周期内同一定时器指令执行一次，且仅执行一次的场合。

实例 8：定时器延迟控制

实例说明

利用定时器的延迟控制来实现电动机的启动和停止都在按钮操作一段时间后才能实现，从而保证设备的安全。

实例实现

电动机启、停按钮及控制继电器与 PLC 的连接如表 2-4 所示，其相应的程序梯形图和指令如图 2-15 所示，时序如图 2-16 所示。其对应的语句为：

```
LD      I0.0
TON T32, 5000
LD      I0.1
TOF     T34, 300
LD      T32
A       T34
=       Q0.0
TONR    T5, 300
LD      T5
=       Q0.1
```

表 2-4　电动机优先控制电路中 PLC 接口连接表

PLC 端口	I0.0	I0.1	Q0.0	Q0.1
外接控制口	总启动按钮	M1 启停按钮	M1 控制继电器	M2 控制继电器

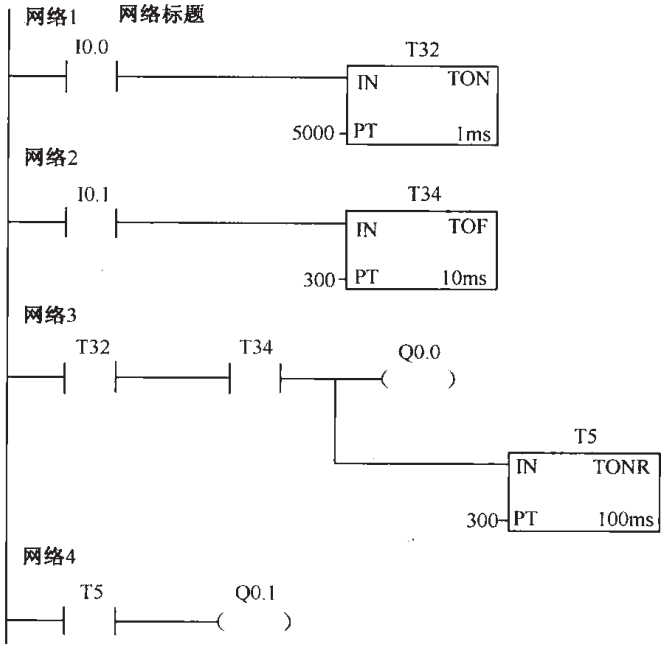


图 2-15　定时器延迟控制程序

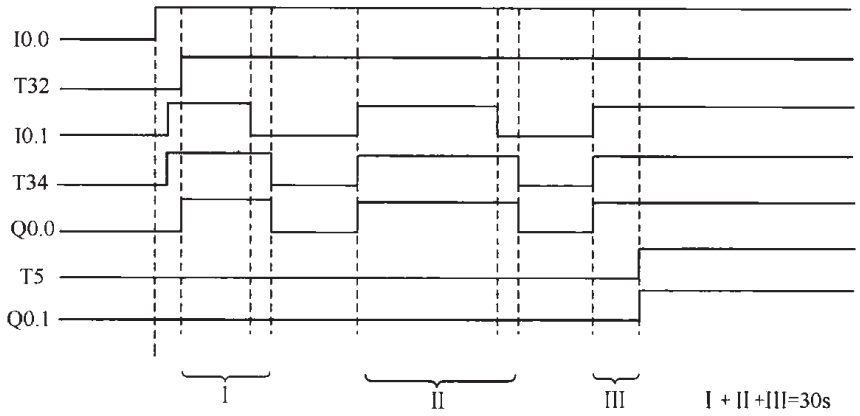


图 2-16　定时器延迟控制时序图

实例分析

按照表 2-4 的连接以及图 2-15 的梯形图，可以实现的功能如下：

（1）总启动按钮接通 5 s 后启动电动机 M1；

（2）电动机 M1 在运行过程中受电动机启停按钮的控制，在电动机 M1 的启停按钮断开后，需要过 3 s 后电动机才能停止，以保证电动机冷却。

（3）当电动机 M1 累计工作时间达到 30 s 时，控制系统自动启动电动机 M2。

在实际的生产控制中经常会碰到类似的情况，需要在按钮操作后一定时间，或者在上电后一定的时间实现某个动作，这类问题合理的使用定时器都可以实现这样的延迟控制。

2. 计数器指令

计数器与定时器的结构和使用基本相似，也由集成电路构成，是应用非常广泛的编程元件。计数器用来累计输入脉冲的次数，经常用来对产品进行计数。

编程时提前输入所计次数的预设值，当计数器的输入条件满足时，计数器开始运行并对

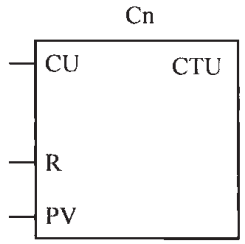
输入脉冲进行计数，当计数器的当前值达到预设值时，计数器发生动作，发出中断请求，以便 PLC 响应而做出相应的动作。

S7-200 PLC 计数器有 3 种类型：增计数器（指令为 CTU）、减计数器（指令为 CTD）和增减计数器（指令为 CTUD），共 256 个（号码为 C0～C255）。

与定时器一样，计数器号码不仅仅是计数器的编号，它包含两方面的变量信息：计数器位和计数器当前值。计数器位：存储计数器的状态，根据满足的条件使得计数器位置“1”或置“0”。计数器当前值：存储计数器当前所累计的脉冲个数，它用 16 位符号整数来表示，故最大计数值为 32767，最小值为–32767。

（1）增计数器指令 CTU

增计数器指令的梯形图和语句表如图 2-17 所示。CTU 为计数器标识符，Cn 为计数器编号，CU 为计数脉冲输入端（数据类型为 BOOL 型），R 为复位信号输入端（数据类型为 BOOL 型），PV 为脉冲设定值输入端（数据类型为 INT 型）。



CTU Cn, PV

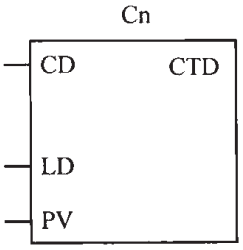
图 2-17 增计数器指令

增计数器在复位端 R 信号为 1 时，计数器复位，计数器的当前值为 0，计数器位也为 0。只有当复位端的信号为 0 时，计数器才可以工作。在计数端 CU 每个脉冲输入的上升沿，计数器的当前值进行加 1 操作。当计数器的当前值大于等于设定值 PV 时，计数器位变为 1，这时再来计数脉冲时，计数器的当前值仍不断地累加，直到 32767 时，停止计数。直到复位端 R 再次为 1，计数器被复位。

（2）减计数器指令 CTD

减计数器指令的梯形图和语句表如图 2-18 所示。CTD 为计数器标识符，Cn 为计数器编号，CD 为计数脉冲输入端（数据类型为 BOOL 型），LD 为装载输入端（数据类型为 BOOL 型），PV 为脉冲设定值输入端。

减计数器在装载输入端 LD 信号为 1 时，其计数器的设定值 PV 被装入计数器的当前值寄存器，此时当前值为 PV，计数器位为 0。只有当装载输入端的信号为 0 时，计数器才可以工作。在计数端 CD 每个脉冲输入的上升沿，计数器的当前值进行减 1 操作。当计数器的当前值等于 0 时，计数器位变为 1，并停止计数。这种状态一直保持到装载输入端 LD 变为 1，再次装入 PV 值之后，计数器位变为 0，才能再次重新计数。

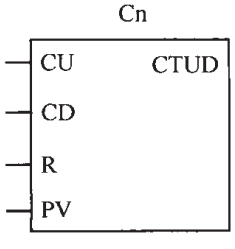


CTD Cn, PV

图 2-18 减计数器指令

（3）增减计数器指令 CTUD

增减计数器指令的梯形图和语句表如图 2-19 所示。CTUD 为计数器标识符，Cn 为计数器编号，CU 为增计数脉冲输入端，CD 为减计数脉冲输入端，R 为复位信号输入端，PV 为脉冲设定值输入端。



CTUD Cn, PV

图 2-19 增减计数器指令

增减计数器在复位端 R 信号为 1 时，其计数器的当前值为 0，计数器位也为 0。只有在复位端 R 的信号为 0 时，计数器才可以工作。每当一个增计数输入脉冲端 CU 上升沿到来时，计数器的

当前值进行加 1 操作。当计数器的当前值大于等于设定值 PV 时，计数器位变为 1。这时再来增计数脉冲时，计数器的当前值仍不断地累加，达到最大值 32767 后，下一个 CU 脉冲上升沿将使计数器当前值跳变为最小值（–32768）并停止计数。每当一个减计数脉冲上升沿到来时，计数器的当前值进行减 1 操作。当计数器的当前值小于设定值 PV 时，计数器位变为 0。再来减计数脉冲时，计数器的当前值仍不断地递减，达到最小值–32768 后，下一个 CD 脉冲上升沿使计数器的当前值跳变为最大值（32767）并停止计数。

在 3 种计数器中，CU、CD、LD 和 R 的寻址范围为 I、Q、M、SM、T、C、V、S 和 L；PV 的寻址范围为 VW、IW、QW、MW、SW、SMW、LW、AIW、T、C、AC、常数、*VD、*LD 和*AC。3 种计数器都可以用复位指令来复位，复位后，计数器位变为 0，计数器当前值变为 0（CTD 变为预设值 PV）。在一个程序中，同一个计数器号码只能使用一次，计数脉冲输入和复位信号输入同时有效时，优先执行复位操作。用语句表表示时，各计数器一定要按梯形图所示的各个输入端顺序输入，不能颠倒，这一点可以从实例 10 中看出。

实例 9：计数器控制

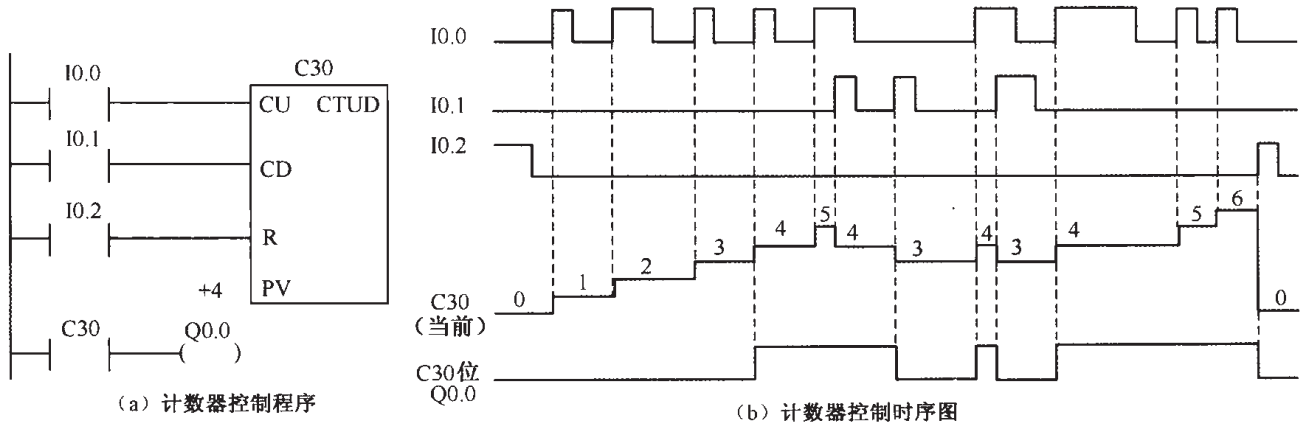
实例说明

利用本实例来说明计数器的使用，包括计数器的复位、计数器的计数，以及计数器值达到设定值后所能进行的操作。

实例实现

计数器控制的程序及其时序如图 2-20 所示，其对应的语句为：

```
LD      I0.0
LD      I0.1
LD      I0.2
CTUD    C30, +4
LD      C30
=       Q0.0
```



（a）计数器控制程序　　（b）计数器控制时序图

图 2-20　计数器控制程序及其时序图

实例分析

在 I0.2 输入端为 1 时，计数器复位；当 I0.2 输入端为 0 时，计数器进行计数，计数器的当前值是 I0.0 端输入脉冲的上升沿数目与 I0.1 端输入的上升沿数目的差，当计数器数值大于等于 4 时，计数器 C30 位通为 1，同时输出端 Q0.0 接通。

2.1.3 比较操作指令

比较操作指令将两个操作数按指定条件进行比较。在实际应用中，使用比较指令为上下限控制以及数值条件判断提供了方便。

比较操作指令的比较符有：=、>=、<=、>、<和< >（< >表示不等于）6 种。比较操作指令的类型有：字节比较、整数比较、双字整数比较和实数比较 4 类。字节比较的比较操作数是无符号的，其他类型的比较操作数是有符号的。

比较操作指令的梯形图、语句表如图 2-21 所示。在梯形图中比较符的表示有：==（等于）、>=（不小于）、<=（不大于）、>（大）、<（小于）和< >（不等于），而在语句表中的比较符的表示分别为：=、>=、<=、>、<和< >表示。数据类型在梯形图中的表示有：B（字节），I（整数），D（双整数）和 R（实数），而在语句表中的数据类型的表示分别为：B，W，D 和 R。逻辑关系表示的是比较操作后的结果与其前面位的逻辑关系，其表示有 LD（取位），A（与关系）和 O（或关系）。不同数据类型的比较指令，其操作数的寻址范围是不一样的：字节比较输入 IN1 和 IN2 的寻址范围为 IB、QB、MB、SMB、VB、SB、LB、AC、常数、*VD、*LD 和*AC；整数比较输入 IN1 和 IN2 的寻址范围为 IW、QW、MW、SW、SMW、T、C、VW、LW、AIW、AC、常数、*VD、*LD 和*AC；双整数比较输入 IN1 和 IN2 的寻址范围为 ID、QD、MD、SD、SMD、VD、LD、HC、AC、常数、*VD、*LD 和*AC；实数比较输入 IN1 和 IN2 的寻址范围为 ID、QD、MD、SD、SMD、VD、LD、AC、常数、*VD、*LD、*AC。

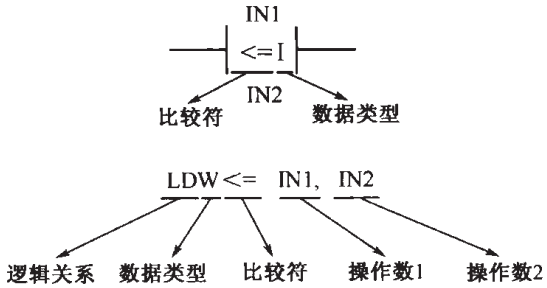


图 2-21 比较操作指令的梯形图、语句表

比较触点相当于一个带有条件的常开触点。当比较数 IN1 和比较数 IN2 的关系符合比较符的条件时，比较触点闭合，否则比较触点断开。

实例 10：数据的比较

实例说明

使用本实例来说明如何使用比较指令。

实例实现

数据的比较程序如图 2-22 所示。其相应的语句为：

```
LDB>     VB1,VB2
=        Q0.0
LD       I0.0
AW>=     C40,2000
=        Q0.1
LD       0.1
OD<      –150000000, VD1
=        Q0.2
LDR<=    VD2, 3.14
=        Q0.3
```

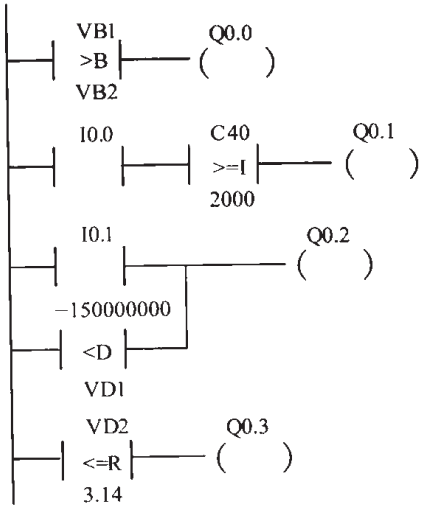


图 2-22　数据的比较程序

实例分析

VB1 中的值大于 VB2 中的值，Q0.0 为 ON；I0.0 为 ON 且计数器 C40 中的当前值大于等于 2000 时，Q0.1 为 ON；I0.1 为 ON 或–150000000 小于 VD1 中的值时，Q0.2 为 ON；VD2 中值的小于 3.14 时，Q0.3 为 ON。

实例 11：水位、水温控制

实例说明

本实例利用数据比较来实现水位、水温的控制。在热水箱中需要对水位和水温控制：水箱中水位低于下警戒水位时，打开进水阀给水箱中加水。当水位高于水箱中的上警戒水位时，关闭进水阀；水箱中的水温低于设定温度下限时，打开加热器给水箱中的水加热。当水温高于设定温度上限时停止加热；在加热器没有工作且进水阀关闭时打开出水阀，以便向外供水。

实例实现

进水阀继电器与 PLC 的输出端口 Q0.0 连接，出水阀继电器与 PLC 的输出端口 Q0.1 连接，水箱加热器的控制继电器与 PLC 的输出断开 Q0.2 相连接。其控制程序如图 2-23 所示。其相应的语句为：

```
LDD<    VD10, 300
=       Q0.0
LDD<    VD10, 1500
R       Q0.0, 1
LDD<    VD20, 50
=       Q0.1
LDD<    VD20, 60
R       Q0.1, 1
LDN     Q0.0
AN      Q0.1
=       Q0.2
```

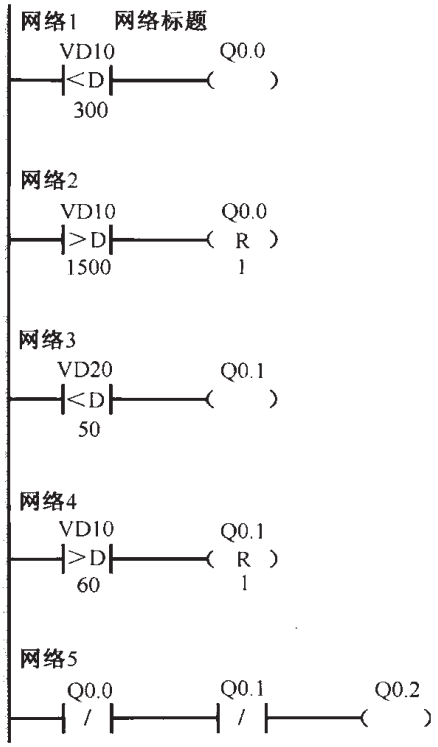


图 2-23 水位、水温控制程序

实例分析

使用液位计实时将热水器的水位由 PLC 的模拟量输入口输入（模拟量的输入见第 4 章）并传送到地址 VD10，同样使用温度计实时将热水器中的水温通过模拟量输入口输入后传送到 VD20；水箱中水位的上警戒值、下警戒值分别为 1500，300；水温的上、下限温度分别为 60℃、50℃。

2.1.4　移位操作指令

移位操作指令都是对无符号数进行处理，包括移位指令、循环移位指令和寄存器移位指令，执行时只需考虑被移位存储单元的每一位数字状态，而不用考虑数据值的大小。该类指令在一个数字量输出端子对应多个相对固定状态的情况下有广泛的应用。

1. 移位指令

移位指令有右移和左移两种，根据所移位数的长度分别又可分为字节型、字型和双字型。移位指令的梯形图和语句表如图 2-24 所示。移位数据存储单元的移出端与 SM1.1（溢出位）相连，最后被移出的位被放到 SM1.1 位存储单元。SHR_B、SHR_W 和 SHR_DW 为字节、字和双字右移标识符；相应地 SHL_B、SHL_W 和 SHL_DW 为字节、字和双字右移标识符；EN 为移位允许信号输入端（数据类型为 BOOL 型）；ENO 为功能框允许输出端（数据类型为 BOOL 型）；IN 为移位数据输入端（数据类型为 BYTE 型、WORD 型或 DWORD 型）；OUT 为移位数据输出端（数据类型为 BYTE 型、WORD 型或 DWORD 型），N 为移位次数输入端（数据类型为 BYTE 型）。移位指令中各有效操作数的寻址范围如表 2-5 所示。

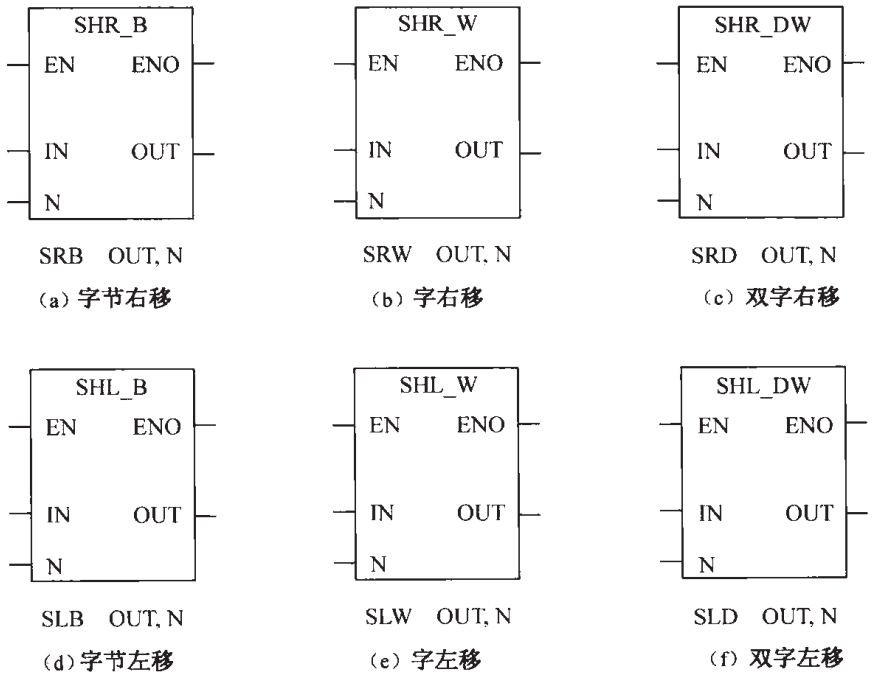


图 2-24　移位指令

表 2-5　移位指令操作数寻址范围

输入/输出	数据类型	操作数寻址范围
IN	字节	VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*LD、*AC
	字	VW、IW、QW、MW、SW、SMW、LW、T、C、AIW、AC、常数、*VD、*LD、*AC
	双字	VD、ID、QD、MD、SD、SMD、LD、AC、HC、常数、*VD、*LD、*AC
OUT	字节	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*LD、*AC
	字	VW、IW、QW、MW、SW、SMW、LW、T、C、AC、*VD、*LD、AC
	双字	VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*LD、*AC
N	字节	VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*LD、*AC

移位时，移出位进入 SM1.1，另一端自动补 0。SM1.1 始终存放最后一次被移出的位，移位次数为 N，如果所需移位次数大于移位数据的位数，则超出次数无效。如果移位操作使数据变为 0，则 SM1.0（零存储器位）自动置位。当移位允许信号 EN=1 时，被移位数 IN 根据移位类型相应的右移或左移 N 位，最左边或最右边移走的位依次用 0 填充，其结果传送到 OUT 中（在语句表中，IN 与 OUT 使用同一个单元）。字节、字和双字移位的最大实际可移位次数分别为 8、16、32。

2. 循环移位指令

循环移位指令与普通移位指令类似，有循环右移和循环左移两种，根据所移位数的长度分别又可分为字节型、字型和双字型。循环移位数据存储单元的移出端与另一端相连，同时又与溢出位 SM1.1 相连，所以最后被移出的位被移到另一端的同时，也被放到 SM1.1 位存储单元。

移位指令的梯形图和语句表如图 2-25 所示。ROR_B、ROR_W 和 ROR_DW 为字节、字和双字循环右移标识符；相应地 ROL_B、ROL_W 和 ROL_DW 为字节、字和双字循环左移标识符；其他操作数的含义和数据类型以及其寻址范围同普通移位指令一样，在此不再重复。在循环移位指令中如果移位次数设定值大于移位数据的位数，则在执行循环移位之前，系统先对设定值取以数据长度为底的模，用小于数据长度的结果作为实际循环移位的次数，因此，字节、字和双字移位的实际移位次数分别为取 8、16、32 为底的模所得的结果。如果移位操作使数据变为 0，则零存储器位 SM1.0 自动置位。

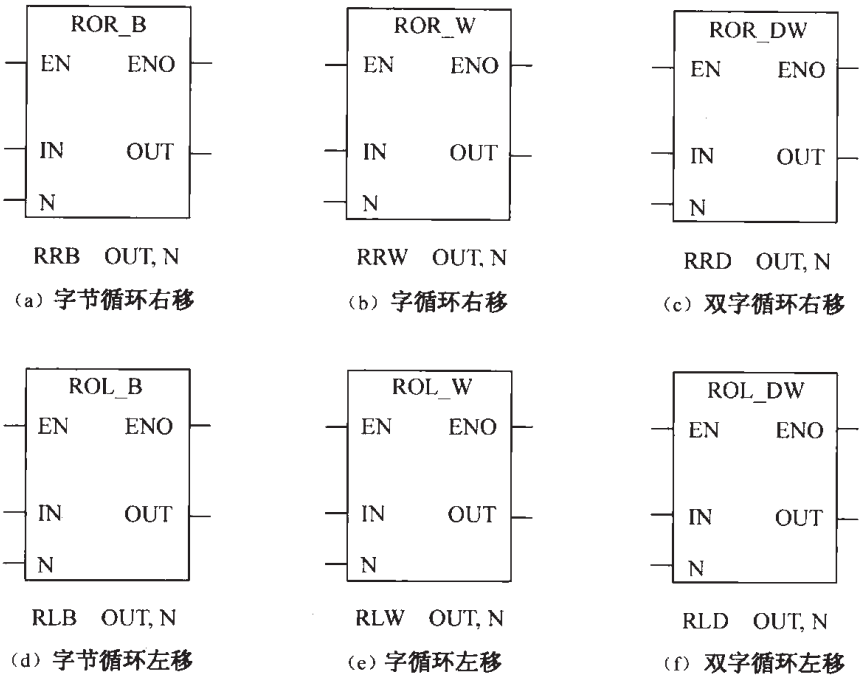


图 2-25 循环移位指令

实例 12：跑马灯的实现

实例说明

本实例是采用循环移位指令来实现霓虹灯上的跑马灯，也就是灯的亮、灭沿某一方向依次移动，给人的感觉就是灯在运动。

实例实现

将霓虹灯中每一灯的控制开关分别与 PLC 的输出端口 QB0 连接。根据所需显示的图案，确定 QB0 的哪些位上输出值为 1，哪些位上输出值为 0，从而确定 QB0 的值，假若根据需要所确定的 QB0 的值为 229，则其对应的二进制数为 11100101，这样与 QB0 端口相连接的 8 个灯成为亮、亮、亮、灭、灭、亮、灭、亮的图案，利用一定的方波信号就可以控制这些灯的亮、灭移动情况，其程序如图 2-26 所示，对应的语句为：

```
LD      SM.4
EU
RRB QB0, 1
```

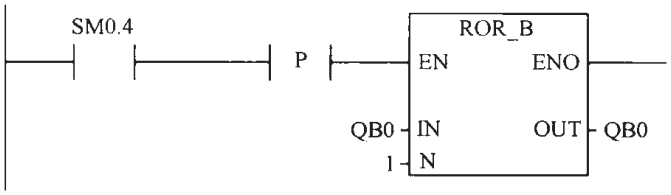


图 2-26　跑马灯程序

实例分析

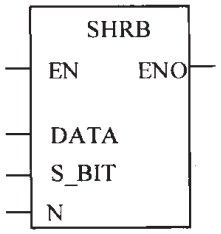
从附录 A 中可以查得 SM0.4 是周期为 1 分钟的方波信号，也就是每经过 1 分钟跑马灯就会向前一步，如果要跑马灯运行更快可以使用定时器或者周期较小的方波信号来控制循环移位指令；如果要改变跑马灯的运动方向只需把右移循环指令改换为左移循环指令即可，当然要跑马灯显示不同的形状，只需要将 QB0 中的值显示为不同的值就可以实现了。

3. 寄存器移位指令

寄存器移位指令将一个数值移入移位寄存器中，它提供了一种排列和控制产品流或者数据的简单方法。

寄存器移位指令的梯形图和语句表如图 2-27 所示。SHRB 为寄存器移位标识符，EN 为移位允许信号输入端（数据类型为 BOOL 型），DATA 为移位数值输入端（数据类型为 BOOL 型，该位的值将移入移位寄存器），S_BIT 为移位寄存器的最低位端（数据类型为 BOOL 型），N 为移位寄存器长度输入端（数据类型为 BYTE 型）。N 的寻址范围为 VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*LD 和*AC。DATA、S_BIT 的寻址范围为 I、Q、M、SM、T、C、V、S 和 L。

移位时，移出端与 SM1.1（溢出）相连，最后被移出的位被放到 SM1.1 位存储单元，移入端自动补以 DATA 的值补入。移位寄存器的长度没有字节型、字型、双字型之分，最大长度为 64 位，可正可负。移位长度 N 为正值时，正向移位，移位是从最低字节的最低位 S_BIT 移入，从最高字节的最高位 MSB.b 移出；N 为负值时，反向移位，移位是从最高字节的最高位 MSB.b 移入，从最低字节的最低位 S_BIT 移出。最高位的计算方法：



SHRB DATA, S_BIT, N

图 2-27　寄存器移位指令

MSB.b 的字节号 MSB 为：(|N|－1＋(S_BIT 的位号))/8＋S_BIT 的字节号。

MSB.b 的位号 b 为：(|N|－1＋(S_BIT 的位号))对 8 取模。

例如，如果 S_BIT 是 V33.4，N 是 14，那么 MSB.b 是 V35.1。具体计算如下：

MSB = V33＋(|14|－1＋4)/8 = V33＋17/8 =35；由于(|14|－1＋4)对 8 取模为 1，故 b=1。

实例 13：应用寄存器移位

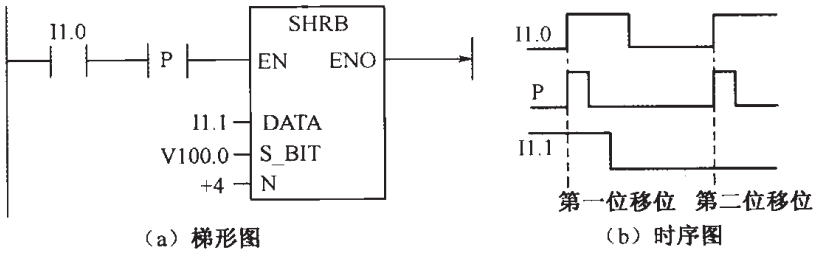
实例说明

本实例用来说明如何应用寄存器移位。

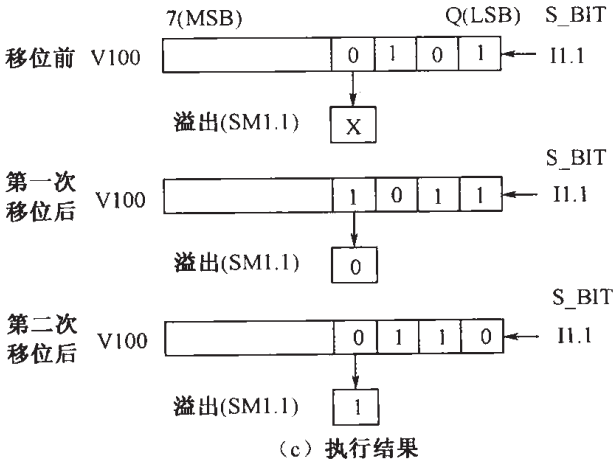
实例实现

应用寄存器移位程序如图 2-28（a）所示，其相应的语句为：

```
LD      I1.0
EU
SHRB    I1.1，V100.0，+4
```



（a）梯形图　　（b）时序图



（c）执行结果

图 2-28　应用寄存器移位

实例分析

使用该指令时，在每个扫描周期内，整个移位寄存器移动 1 位，所以要用上微分操作指令来控制 EN 端的状态，不然该指令就失去了应用的意义。图 2-28（a）的程序所执行后的时序图如图 2-28（b）所示，执行结果如图 2-28（c）所示。

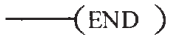
2.1.5　程序控制指令

程序控制指令使程序结构灵活，合理使用该指令可以优化程序结构，增强程序功能。这类指令主要包括结束、停止、看门狗复位、跳转与标号、循环、子程序和顺序控制继电器等指令。

1. 有条件结束指令

有条件结束指令梯形图和语句表如图 2-29 所示。执行该指令后，系统结束主程序，返回主程序起点。

有条件结束指令根据先前逻辑条件终止用户程序，可以在主程序内使用，但不能在子程序或中断程序内使用。

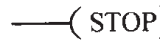


END

图 2-29　有条件结束指令

2. 停止指令

停止指令的梯形图和语句表如图 2-30 所示。



STOP

图 2-30　停止指令

停止指令不含操作数，执行该指令后，PLC 从 RUN（运行）模式进入 STOP（停止）模式，立即终止程序的执行。

使用该指令需注意：如果在中断程序内执行暂停指令，中断程序立即终止，并忽略全部等待执行的中断，继续扫描主程序的剩余部分，在当前扫描结束时从 RUN 模式转换到 STOP 模式。

3. 看门狗复位指令

看门狗复位指令梯形图和语句表如图 2-31 所示。

为了保证系统可靠运行，PLC 内部设置了系统监视定时器（WDT），用于监视扫描周期是否超时。每当扫描到 WDT 定时器时，WDT 定时器将复位。WDT 定时器有一设定值（100～300 ms），系统正常工作时，所需扫描时间小于 WDT 的设定值，WDT 定时器及时复位。系统在发生故障的情况下，扫描时间大于 WDT 设定值，该定时器不能及时复位，则报警并停止 CPU 运行，同时复位输出。这种故障称为 WDT 故障，以防止因系统故障或程序进入死循环而引起的扫描周期过长。

WDR

图 2-31　看门狗复位指令

系统正常工作时，如果希望扫描时间超过 WDT 定时器的设定值，或者预计发生大量中断事件，或者使用循环指令使扫描时间过长，可能在 WDT 定时器的设定值内不能返回主程序，为防止这些情况下 WDT 动作，可以考虑使用看门狗复位指令，重新触发 WDT，使其复位，在没有监视程序错误的条件下增加 CPU 系统扫描占用的时间。使用看门狗复位指令时应当小心，因为使用循环指令会造成阻止扫描完成或过度地延迟扫描完成时间。

下列程序只有在扫描循环完成后才能执行：通信（自由端口模式除外）；I/O 更新（立即 I/O 除外）；强制更新；SM 位更新（SM0、SM5～SM29 除外）；运行时间诊断程序；中断程序中的 STOP 指令；分辨率 10 ms 和 100 ms 定时器对于超过 25 s 的扫描不能正确地累计时间。

实例 14：PLC 故障控制

实例说明

在 PLC 运行过程中会出现许多料想不到的故障，为了避免故障发生所带来严重的后果，需要采取一定的手段保证 PLC 正常运行或者使其停止运行。在这些情况下往往会用到有条件结束指令、停止指令以及看门狗复位指令等。

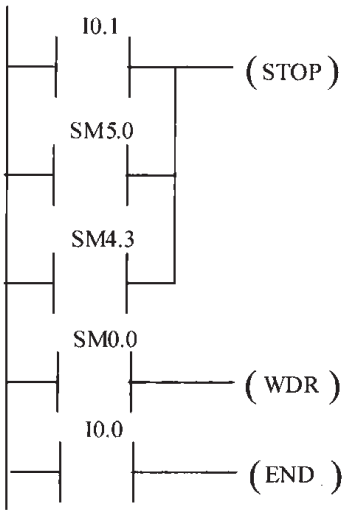


图 2-32　PLC 故障控制程序

实例实现

PLC 故障控制的简单程序如图 2-32 所示。其对应的语句为：

```
LD      I0.1
O       SM5.0
O       SM4.3
STOP
LD      SM0.0
WDR
LD      I0.0
END
```

实例分析

在这个程序中，PLC 在以下 3 种情况下会执行 STOP 停止指令，从而停止 PLC 的运行防止事故的发生。

（1）在 PLC 运行过程中如果现场出现了特殊情况，按下与 I0.1 相连接的按钮，使得 I0.1 位为 1；

（2）PLC 系统出现 I/O 错误；

（3）PLC 监测到系统程序出现了问题。

当循环程序很多或者中断很多时，虽然 PLC 是正常运行的，但会大大延长 PLC 的扫描周期而造成 WDT 故障。为了使 PLC 顺利运行，可以在适当的位置执行看门狗复位指令，重新触发 WDT，使其复位。

在 PLC 运行过程中，若不希望运行某一部分程序，则可在这段不希望运行的程序前面加上图 2-32 所示的最后一条指令，这样只要接通与 I0.0 相连的按钮，就会执行 END 指令，PLC 就会返回主程序起点，重新执行。

4. 跳转与标号指令

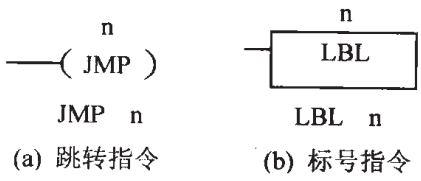
在程序执行时，由于条件的不同，可能会产生一些分支，这时就需要用到跳转和标号指令，根据不同条件的判断，选择不同的程序段执行程序。

跳转和标号指令的梯形图和语句表如图 2-33 所示。操作数 n 为数字 0～255。

当跳转条件满足时，跳转指令可以使程序流程转到具体的标号（n）处执行。标号指令

用来标记指令转移目的地的位置（n）。

跳转指令和标号指令必须配合使用，而且只能使用在同一程序块中，如主程序、同一个子程序或同一个中断程序。不能在不同的程序块间互相跳转。



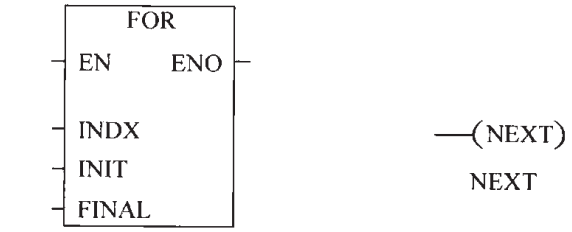
(a) 跳转指令　(b) 标号指令

图 2-33　跳转与标号指令

执行跳转后，被跳过程序段中各寄存器的状态会有所不同：Q、M、S、C 等寄存器的位将保持跳转前的状态；计数器 C 停止计数，当前值存储器保持跳转前的计数值；对定时器来说，因刷新方式不同而工作状态不同。在跳转期间，分辨率为 1 ms 和 10 ms 的定时器会一直保持跳转前的工作状态，原来工作的继续工作，到设定值后其位的状态也会改变，输出触点动作，其当前值存储器一直累计到最大值 32767 才停止。对分辨率为 100 ms 的定时器来说，跳转期间停止工作，但不会复位，存储器里的值为跳转时的值，跳转结束后，若输入条件允许，可继续计时，但已失去了准确计时的意义，所以在跳转段里的定时器要慎用。

5. 循环指令

循环指令，为解决重复执行相同功能的程序段提供了极大的方便，并且优化了程序结构。循环指令有两条：循环开始指令和循环结束指令，其梯形图和语句表如图 2-34 所示。FOR 和 NEXT 为标识符，EN 为循环允许信号输入端（数据类型为 BOOL 型），ENO 为功能框允许输出端（数据类型为 BOOL 型），INDX 为当前值输入端（数据类型为 INT 型），INIT 为循环初值输入端（数据类型为 INT 型），FINAL 为保留循环终值输入端（数据类型为 INT 型）。



（a）循环开始指令　（b）循环结束指令

图 2-34　循环指令

INDX 的寻址范围为 VW、IW、QW、MW、SW、SMW、LW、T、C、AC、*VD、*LD 和*AC。INIT、FINAL 的寻址范围为 VW、IW、QW、MW、SW、SMW、T、C、AC、LW、AIW、常数、*VD、*LD 和*AC。

在循环指令中，循环开始指令（FOR）用来标记循环体的开始，循环结束指令（NEXT）用来标记循环体的结束，FOR 和 NEXT 之间的程序段称为循环体。当程序运行到循环指令时，如果循环允许信号 EN 端为 1 时，PLC 就会自动的把循环初值输入端 INIT 的值复制给当前计数输入端 INDX，用 INDX 计数值与循环终值输入端 FINAL 的值进行比较，如果不大于终值，就执行循环体，每执行一次循环体，INDX 计数值增 1，并且将其结果同循环终值作比较，如果大于终值，则终止循环。

实例 15：循环指令的应用

实例说明

本实例用来说明如何使用循环指令。

实例实现

实现循环指令的梯形图如图 2-35 所示。

实例分析

在循环指令中 FOR 和 NEXT 指令必须成对使用，FOR 和 NEXT 可以如图 2-35 所示那样嵌套，但是嵌套最多为 8 层，而且各个嵌套之间不可有交叉现象，循环指令执行完毕后在下一次启用时指令将自动复位各参数，如果在循环块中对 INDX 计数值进行复值将会影响循环体执行的次数，如果循环指令的初值大于终值时，循环体不被执行。

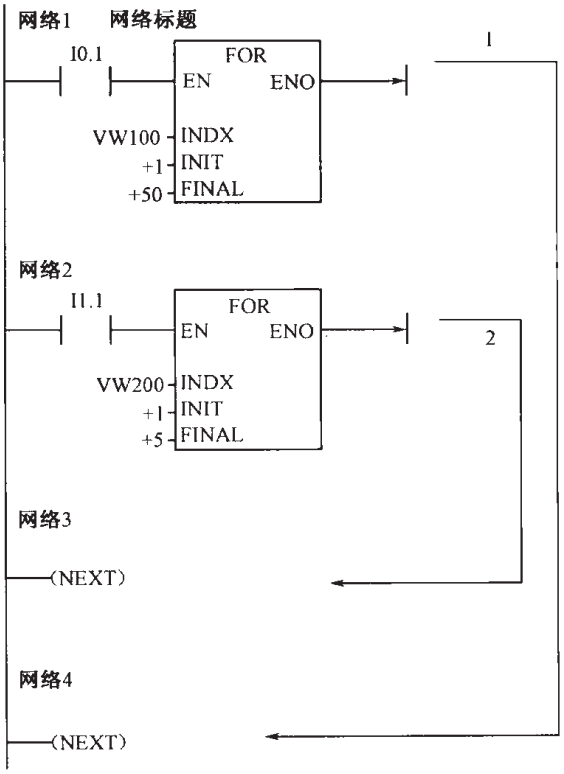


图 2-35 循环指令的应用

对于图 2-35 中循环指令的应用，当 I1.0 接通时，外层循环 1 执行 50 次；在执行外层循环 1 的过程中，每当 I1.1 接通时，内层循环 2 执行 5 次。

6. 子程序操作指令

在编写程序时，有的程序段需要多次重复使用。这样的程序段可以编成一个子程序，在满足执行条件时，主程序转去执行子程序，子程序执行完毕后，再返回来继续执行主程序。另外，有的程序段不仅需多次使用，而且要求程序段的结构不变，但每次输入和输出操作数

不同。对这样的程序段也可以编写成一个子程序，在满足执行条件时，主程序转去执行子程序，并且每次调用时赋予该子程序不同的输入和输出操作数，子程序执行完毕再返回去继续执行主程序。

子程序操作指令有两条：子程序调用指令和子程序返回指令，其梯形图和语句表如图 2-36 所示，n 为子程序标号（0～63）。

（a）子程序调用指令　　（b）子程序返回指令

图 2-36　子程序操作指令

子程序的调用由在主程序内使用的调用指令完成。当子程序调用允许时，调用指令将程序控制转移给子程序（SBR_n），程序扫描将转到子程序入口处执行。当执行子程序时，子程序将执行全部指令直至满足返回条件才返回，或者执行到子程序末尾而返回。当子程序返回时，返回到原主程序出口的下一条指令执行，继续往下扫描程序。

应用子程序操作指令应注意的问题：

（1）子程序由子程序标号开始，到子程序返回指令结束。S7-200 PLC 的 STEP7-Micro/WIN 编程软件为每个子程序自动加入子程序标号和无条件子程序返回指令，无需编程人员手工输入；如果需要在子程序执行过程中满足一定的条件就跳出子程序，也可以在子程序中添加子程序返回指令，从而由判断条件决定是否结束子程序调用。

（2）如果在子程序的内部又对另一个程序执行调用指令，则这种调用称为子程序的嵌套。子程序嵌套的深度最多为 8 级，但是不允许子程序直接递归调用。例如，不能从 SBR0 调用 SBR0。但是，允许进行间接递归调用。

（3）对于带参数的子程序调用指令应遵守下列原则：参数必须与子程序局部变量表内定义的变量完全匹配；参数顺序应为输入参数最先，其次是输入/输出参数，最后是输出参数；各子程序调用的输入/输出参数的最大限制是 16 个。

（4）累加器可在调用程序和被调用子程序之间自由传递，所以累加器的值在子程序调用时既不保存也不恢复。

实例 16：子程序的调用

实例说明与实现

子程序的调用程序如图 2-37 所示，其对应的语句为：

```
//主程序
LD      SM0.1
CALL    SBR_0
……
//子程序 SBR_0
……
LD      M14.3
RET
……
```

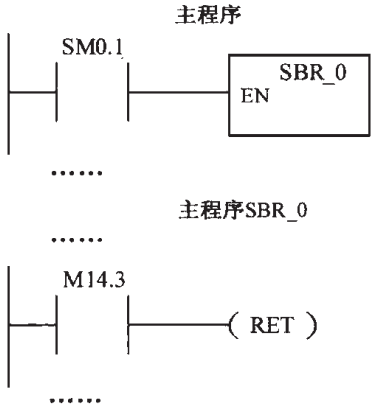


图 2-37　子程序调用程序

实例分析

主程序在首次扫描时，调用子程序 0，执行初始化操作；子程序中，如果 M14.3 闭合，则返回主程序。

7. 顺序控制继电器指令

顺序控制继电器指令有 3 条：顺序控制开始指令（SCR）、顺序控制转移指令（SCRT）和顺序控制结束指令（SCRE）。顺序控制程序从 SCR 开始到 SCRE 结束。

（1）顺序控制开始指令

顺序控制开始指令的梯形图和语句表如图 2-38 所示，Sn 为顺序控制继电器位（S0.0～S31.7）。

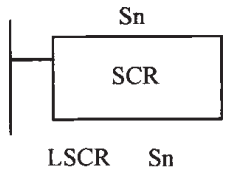


图 2-38　顺序控制开始指令

该指令定义一个顺序控制程序段的开始，Sn 是本段的标志位。当顺序控制继电器位 Sn=1 时，启动 SCR Sn 段的顺序控制程序。在执行到 SCR Sn 之前一定要使 Sn 置位才能进到 SCR Sn 顺序控制程序段。

（2）顺序控制转移指令

顺序控制转移指令的梯形图和语句表如图 2-39 所示，Sn 为顺序控制继电器位（S0.0～S31.7）。

该指令用来指定要启动的下一个程序段，实现本程序段与另一程序段之间的切换，Sn 是下一个程序段的标志位。当执行该指令时，一方面对下一段的 Sn 置位，以便让下一个程序段开始工作，另一方面同时对本段的 Sn 复位，以便本程序段停止工作。注意只有等执行到顺序控制结束指令时，才能过渡到下一个顺序控制程序段。

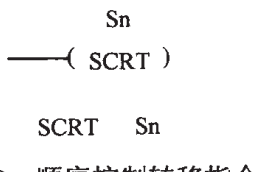


图 2-39　顺序控制转移指令

（3）顺序控制结束指令

顺序控制结束指令的梯形图和语句表如图 2-40 所示。

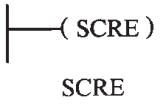


图 2-40　顺序控制结束指令

该指令用于结束本程序段。一个顺序控制程序段必须用该指令来结束。

使用顺序控制继电器指令时只能使用顺序控制继电器位 Sn 作为段标志位。一个顺序控制继电器位在各程序块中只能

使用一次。例如，如果在主程序中使用了 S2.0，就不能再在子程序、中断程序或是主程序的其他地方重复使用它。在一个顺序控制程序段内不允许出现循环程序结构和条件结束，即在段内不能使用 FOR、NEXT 和 END 指令。

实例 17：自动\手动切换控制

实例说明

在许多工业控制场合，不仅仅需要有自动控制的功能，还需要有手动控制的功能。若控制按钮处于自动挡的时候，PLC 自动执行自动控制程序而不执行手动程序；若控制按钮处于手动挡的时候，PLC 自动执行手动控制程序而不执行自动控制程序。这一功能可以用顺序控制来实现。

实例实现

用顺序控制实现自动\手动切换的程序梯形图如图 2-41 所示，其对应的语句为：

```
LD      I0.0
AN      I0.1
SCRT    S0.1
LD      I0.1
AN      I0.0
SCRT    S0.2
LSCR    S0.1
……      //自动运行程序
SCRE
LSCR    S0.2
……      //手动运行程序
SCRE
```

图 2-41　自动/手动切换控制程序

实例分析

若控制按钮处于自动挡时 I0.0 接通，相应的执行自动运行程序；如果控制按钮处于手动挡时 I0.1 接通，相应的执行手动运行程序。

实例 18：设备的初始化控制

实例说明

在工业控制中，常常需要给许多设备初始化后才能进入正常的控制阶段。这些初始化仅

仅只在 PLC 通电一开始的阶段运行，当 PLC 正常运行后，不再执行这些初始化程序，使用顺序控制继电器指令很容易实现这样的控制。

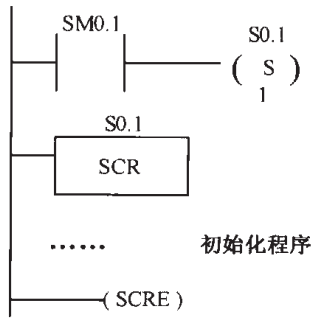


图 2-42　设备的初始化控制程序

实例实现

程序梯形图如图 2-42 所示，相应的语句为：

```
LD      SM0.1
S       S0.1 ,1
LSCR    S0.1
……      //初始化程序
SCRE
```

实例分析

特殊继电器 SM0.1 仅仅在 PLC 上电开始产生一个扫描周期的接通，因此 S0.1 所控制的顺序程序段仅仅在 PLC 上电的第一个扫描周期内运行，也就是实现了设备的初始化控制。

2.2　S7-200 PLC 的功能指令

随着 PLC 的广泛使用，PLC 不仅仅具备一些基本指令，而且发展出了许多的功能指令，这些功能指令使得 PLC 能方便地实现算术逻辑运算、程序流控制、通信等功能，从而满足客户的各种特殊需要，极大地拓宽了 PLC 的应用范围，增加了 PLC 编程的灵活性。

功能指令（Function Instruction）是指令系统中应用于复杂控制的指令，S7-200 PLC 的功能指令主要包括数据传送、数学运算、逻辑运算、表功能、数据转换、中断、高速处理、时钟、通信等指令。

2.2.1　数据传送指令

数据传送指令可用来在各存储单元之间进行一个或多个数据的传送，传送过程中数据值保持不变。根据每次数据传输的多少，可以分为单一传送指令和数据块传送指令。另外对于这两种传送指令，按其传送的数据类型又有字节、字和双字之分，对于单一数据传送其传送的数据类型还可以是实数。为了实现在同一个字内高、低位字节的交换，还有字节交换指令。

1. 单一传送指令

单一传送指令可用来进行一个数据的传送，数据类型可以是字节、字、双字和实数。其指令的梯形图和语句表如图 2-43 所示。

对于字节传送指令、字传送指令、双字传送指令以及实数传送指令，其功能是实现在传送运行信号 EN=1 时，把 IN 端口的数据传送到 OUT 所指示的存储单元。

传送字节立即读指令，其功能是在传送允许信号 EN=1 时，立即读取单字节物理输入区 IN 端口的数据，并传送到 OUT 所指的字节存储单元，一般用于对输入信号的立即响应。

传送字节立即写指令，其功能是在传送允许信号 EN=1 时，立即将 IN 单元的字节数据写到 OUT 所指的物理输出区。该指令用于把计算出的结果立即输出到负载。

各类指令的操作数寻址范围如表 2-6 所示。

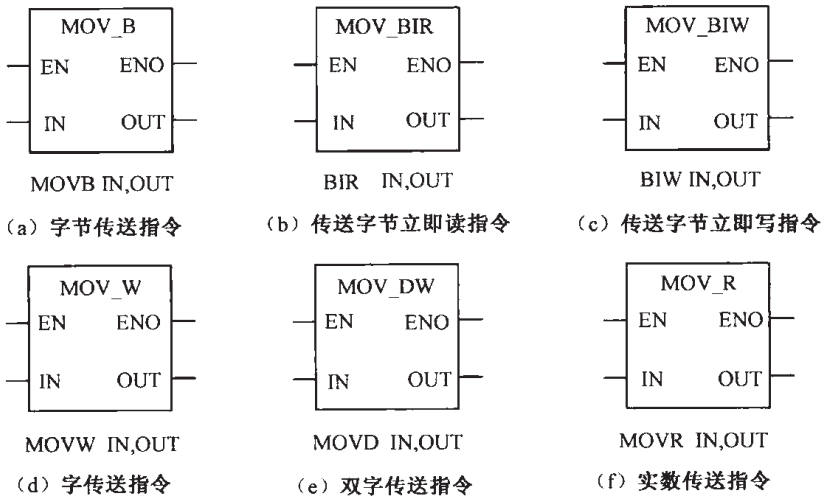


图 2-43　单一传送指令

表 2-6　单一传送指令操作数寻址范围

输入/输出	指 令 类 型	数据类型	操作数寻址范围
IN	字节传送	字节	VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*LD、*AC
	传送字节立即读	字节	IB、*VD、*LD、*AC
	传送字节立即写	字节	VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*LD、*AC
	字传送	字	VW、IW、QW、MW、SW、SMW、LW、T、C、AIW、AC、常数、*VD、*LD 和*AC
	双字传送	双字	VD、ID、QD、MD、SD、SMD、LD、HC、&VB、&IB、&QB、&MB、&SB、&T、&C、AC、常数、*VD、*LD 和*AC
	实数传送	实数	VD、ID、QD、MD、SD、SMD、LD、AC、常数、*VD、*LD、*AC
OUT	字节传送	字节	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*LD、*AC
	传送字节立即读	字节	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*LD、*AC
	传送字节立即写	字节	QB、*VD、*LD、*AC
	字传送	字	VW、T、C、IW、QW、SW、MW、SMW、LW、AC、AQW、*VD、*LD 和*AC
	双字传送	双字	VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*LD 和*AC
	实数传送	实数	VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*LD 和*AC

2. 数据块传送指令

数据块传送指令可用来进行一次多个（最多 255 个）数据的传送，数据块类型可以是字节块、字块和双字块，其梯形图和语句表如图 2-44 所示。其功能是在传送运行信号 EN=1 的条件下，把从输入端子 IN 为起点位置的 N 个相应数据类型的数据传送到 OUT 开始的 N 个对应数据类型的存储单元中。数据块指令的操作数寻址范围如表 2-7 所示。

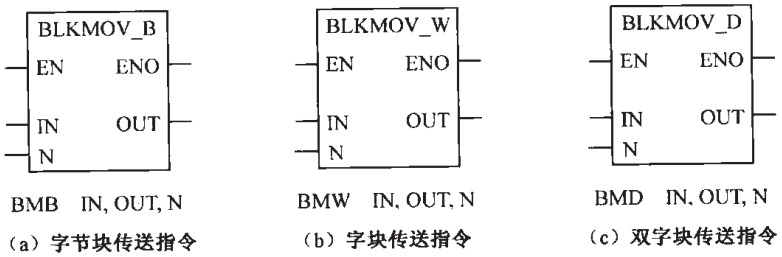


图 2-44　数据块传送指令

表 2-7 数据块传送指令操作数寻址范围

输入/输出	指 令 类 型	数据类型	操作数寻址范围
IN	字节块传送	字节	VB、IB、QB、MB、SB、SMB、LB、AC 、*VD、*LD *AC
	字传块送	字	VW、IW、QW、MW、SW、SMW、 LW、T、C、AIW、*VD、*LD、*AC
	双字块传送	双字	VD、ID、QD、MD、SD、SMD、LD、*VD、*LD 和*AC
OUT	字节块传送	字节	VB、IB、QB、MB、SB、SMB、LB、AC 、*VD、*LD *AC
	字传块送	字	VW、IW、QW、MW、SW、SMW、 LW、T、C、AIW、*VD、*LD、*AC
	双字块传送	双字	VD、ID、QD、MD、SD、SMD、LD、*VD、*LD 和*AC
N	全部数据块	字节	VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*LD、*AC

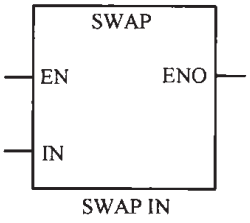


图 2-45 字节交换指令

3. 字节交换指令

字节交换指令的梯形图和语句表如图 2-45 所示。其功能是在交换允许信号 EN=1 时，将字型输入数据 IN 高位字节与低位字节进行交换，交换的结果仍存放在 IN 存储器单元中。字节交换指令不影响特殊标志位存储器位。IN 的寻址范围为 VW、IW、QW、MW、SW、SMW、LW、T、C、AC、*VD、*LD 和*AC。

2.2.2 数学运算指令

1. 加、减、乘和除指令

加法指令实现两个有符号数的相加操作，减法指令实现两个有符号数的相减操作，一般乘法指令实现两个有符号数的相乘操作，一般除法指令实现两个有符号数的相除操作（不保留余数）。由于操作数的不同，分别可以实现整数的加、减、乘、除；双整数的加、减、乘、除和实数加、减、乘、除。在利用加、减、乘、除指令进行运算时，如果是整数的四则指令运算，则进行运算的两操作数必须都是整数，其结果也将是整数；如果是双整数或实数的运算也一样，参与运算的两操作数也必须都是双整数或实数，运算结果也将是双整数或实数。

加、减、乘和除指令的梯形图和语句表如图 2-46 所示。指令操作数的寻址范围如表 2-8 所示。

表 2-8 数据块传送指令操作数寻址范围

输入/输出	数 据 类 型	操作数寻址范围
IN1、IN2	整数	VW、IW、QW、MW、SW、SMW、LW、AIW、 T、C、AC、常数、*VD、*LD、*AC
	双整数 实数	VD、ID、QD、MD、SD、SMD、 LD、HC、AC、常数、*VD、*LD、*AC
OUT	整数	VW、IW、QW、MW、SW、SMW、 LW、T、C、AC、*VD、*LD、*AC
	双整数 实数	VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*LD、*AC

使用梯形图时是将 IN1 与 IN2 运算的结果存放在 OUT 所指向的存储器；如果使用的是语句表，则常常会将某一个输入与输出公用一个存储地址单元，加法和乘法是将 IN2 与 OUT 共用，减法和除法是将 IN1 共用，因此语句表时的运算可以表示为：

IN1+ OUT→OUT，OUT−IN2→OUT，IN1*OUT→OUT，OUT/IN2→OUT

进行加、减、乘、除运算后会对特殊寄存器的一些位产生影响，因此在执行完这些指令后可以查看特殊寄存器里面的这些位的值，从而知道计算的结果是否正确。

受影响的特殊寄存器位有：SM1.0（零）、SM1.1（溢出位）、SM1.2（负）、SM1.3（被零除）。其具体含义是：

（1）SM1.1 指示溢出错误和非法数值。如果 SM1.1 被设置，那么 SM1.0 和 SM1.2 的状态不是有效的，原输入操作数不改变。

（2）如果 SM1.1 和 SM1.3 没有设置，那么运算操作带有有效的结果完成，SM1.0 和 SM1.2 包含有效的状态。

（3）如果在除法操作期间 SM1.3 被设置，那么其他运算状态位保持不变。

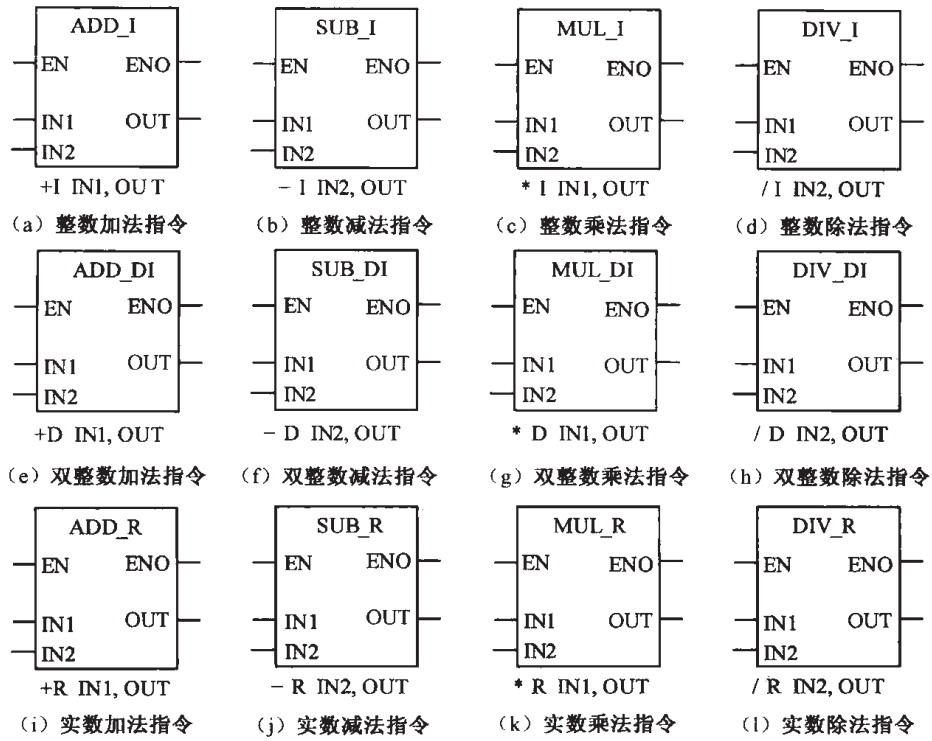


图 2-46　加、减、乘和除指令的梯形图和语句表

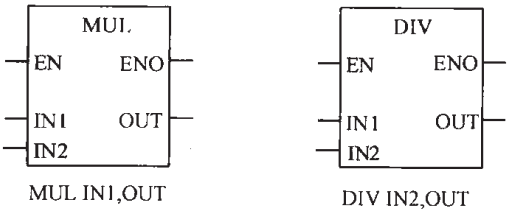
2. 完全整数乘法指令和完全整数除法指令

一般来说，乘法计算所得到的积要比乘数的位数高，而除法运算后还有余数问题，而一般乘法运算和一般除法运算不能解决这些问题，在 PLC 的数学运算指令中还有完全整数乘法指令和完全整数除法指令。完全乘法指令是将两个符号整数的 IN1 和 IN2 相乘，产生一个 32 位双整数结果 OUT。完全整数除法是将两个符号整数（16 位）的 IN1 和 IN2 相除，产生一个 32 位结果，其中，低 16 位为商，高 16 位为余数。其指令的梯形图和语句表如图 2-47 所示。

完全整数乘、除法指令其输入操作数 IN1（为整数）和 IN2（为整数）的寻址范围同表 2-8 中 IN1 和 IN2 的整数的范围一样，输出 OUT（为双整数）的寻址范围同表 2-8 中 OUT

的双整数寻址范围一样。

完全乘法指令，如果使用的是梯形图其运算的结果是 IN1*IN2→OUT；如果使用的是语句表，通常将 IN2 与 OUT 的低位字（16 位）共用一个地址单元，执行结果为 IN1*OUT→OUT。完全除法指令，如果使用的是梯形图其运算的结果是 IN1/IN2→OUT；如果使用的是语句表，通常将 IN1 与 OUT 的低位字（16 位）共用一个地址单元，执行结果为 OUT /IN1→OUT。完全乘、除法指令对特殊寄存器的影响同一般的乘、除法指令对特殊寄存器的影响一样，在此不再重复。



（a）完全整数乘法指令　　（b）完全整数除法指令

图 2-47　完全整数乘、除指令的梯形图和语句表

实例 19：用除法实现数据的分离

实例说明

在 PLC 的通信中，往往需要把接收到的数据进行分离以便使用，接收到某 16 位二进制数据，需要从这 16 位数据中把其高 4 位与其低 12 位分离。要实现这一目的，用完全整数除法就可以实现。

实例实现

假设需要分离的 16 位二进制数据存储在 MW0 中，将分离后的高 4 位数据存放于 MW4 中，低 12 位数据存放于 MW2 中，数据分离程序梯形图如图 2-48 所示，其对应的语句为：

```
LD      SM0.0
MOVW    MW0, MW4
DIV     16#1000, MD2
```

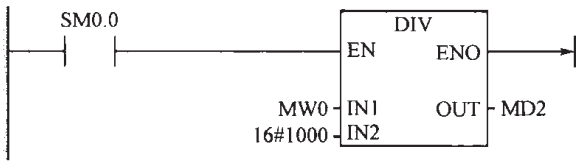


图 2-48　数据的分离程序

实例分析

假若数据分离前 MW0 中存储的数为 16#328E，根据完全除法指令，用该数据来除以 16#1000，则商为 16#3，余数为 16#28E，因此在 MD2 的低 16 位中存放商 16#3，在 MD2 的低 16 位中存放余数 16#28E，而 MD2 的低 16 位就是 MW4，高 16 位就是 MW2。

实例 20：按比例放大模拟值

实例说明

在工业控制中，经常使用传感器来检测一些模拟量，如使用温度传感器检测温度，但是由于传感器所采集到的是电压值，如何把传感器所采集到的值换算成物理量的实际值，这就需要按比例放大模拟值。例如，知道温度传感器在最低检测温度 T_{min} 时，其输出电压为 V_{min}，在最高检测温度 T_{max} 时，其输出电压为 V_{max}，需要找到输出电压为 V 时所对应的温度 T。这一问题可以通过 PLC 的四则运算实现。

对于比例传感器，温度可以用下式算出：

$$T=\frac{(T_{max}-T_{min})\cdot(V-V_{min})}{(V_{max}-V_{min})}+T_{min}$$

实例实现

利用 PLC 来实现，其程序梯形图如图 2-49 所示，其对应的语句为：

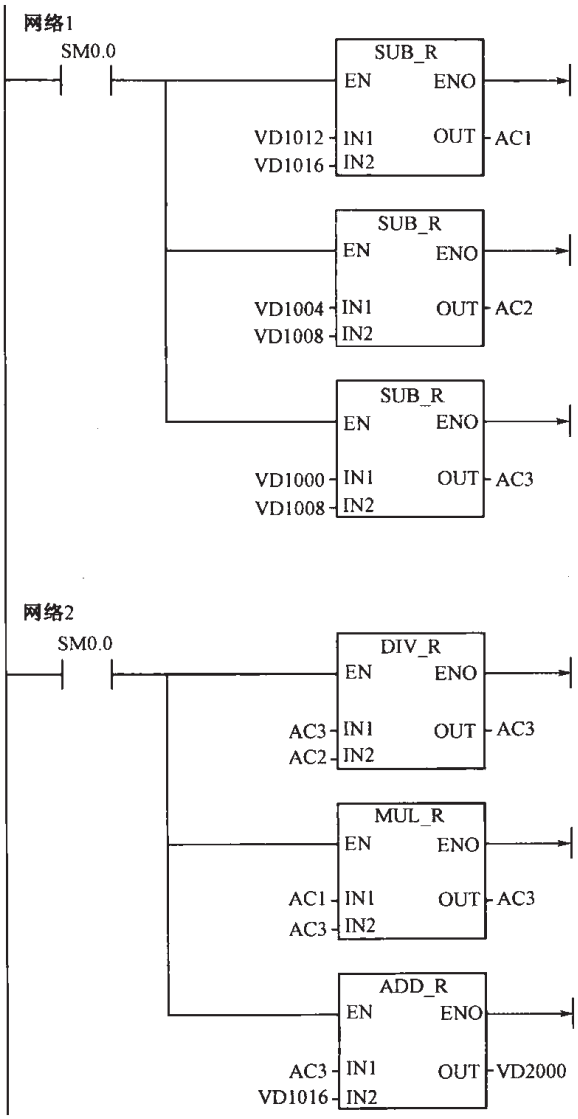


图 2-49　按比例放大模拟值程序

```
LD      SM0.0
MOVR    VD1012,AC1        // VD1012 值对应 Tmax
-R      VD1016,AC1        // VD1016 值对应 Tmin
MOVR    VD1008,AC2        // VD1008 值对应 Vmax
-R      VD1004,AC2        // VD1004 值对应 Vmin
MOVR    VD1000,AC3        // VD1000 值对应 V
-R      VD1008,AC3
LD      SM0.0
/R      AC2,AC3
*R      AC1,AC3
MOVR    AC3 ,VD2000
+R      VD1016 ,VD2000    // VD2000 值对应 T
```

实例分析

在转换前先将传感器标定的值存储在 PLC 内对应的存储器中，然后把传感器所采集到的模拟量也存入对应的位置，利用本实例中的程序就能得到对应的物理参数值。另外在一些需要放大模拟量的值的时候，或者在进行单位转换时也可以用这样的程序来实现。

3. 自增和自减指令

自增和自减指令是对无符号或有符号整数进行自动加 1 或减 1 的操作。操作数可以是字节、字或双字，其中字节增减是对无符号数操作。其梯形图和语句表如图 2-50 所示，功能是当允许信号 EN=1 时，把输入数 IN 加 1 或减 1，得到输出结果 OUT。在 LAD 中，执行结果为 IN+1→OUT 和 IN−1→OUT。在 STL 中，通常将 IN 与 OUT 共用一个地址单元，执行结果为 OUT+1→OUT 和 OUT−1→OUT。操作数的寻址范围如表 2-9 所示。其操作结果会对 SM1.0（零）、SM1.1（溢出）产生影响，另外对于字和双字的自增、自减指令还会对 SM1.2（负）产生影响。

表 2-9 自增、自减指令操作数寻址范围

输入/输出	数 据 类 型	操作数寻址范围
IN	字节	VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*LD、*AC
	字	VW、IW、QW、MW、SW、SMW、LW、AIW、T、C、AC、常数、*VD、*LD、*AC
	双字	VD、ID、QD、MD、SD、SMD、LD、HC、AC、常数、*VD、*LD、*AC
OUT	字节	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*LD 和*AC
	字	VW、IW、QW、MW、SW、SMW、LW、T、C、AC、*VD、*LD、*AC
	双字	VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*LD、*AC

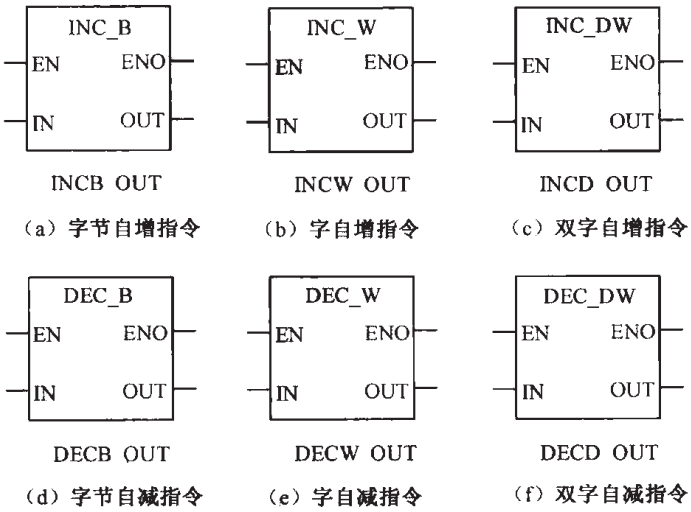


图 2-50　自增、自减指令

4. 数学函数指令

数学函数指令包括平方根、自然对数、指数、三角函数等常用的函数指令，这些指令的梯形图和语句表如图 2-51 所示。数学函数指令的输入和输出数据均为 32 位实数，结果如果大于 32 位二进制数表示的范围，则产生溢出，也就是使得 SM1.1 为 1，结果为零或者负值可以由符号位 SM1.0（零）和 SM1.2（负）得出。自然对数指令是对实数取自然对数。当求解以 10 为底的常用对数时，可以用/R（或 DIV_R）指令将该数的自然对数除以 LN10（约为 2.302585）。指数指令是对实数取以 e 为底的指数。可以用指数指令和自然对数指令相配合来完成以任意常数为底和以任意常数为指数的计算，例如，求 X 的 Y 次幂，输入指令：EXP（Y * LN （X））。正弦、余弦和正切指令是对实数弧度值进行相应的计算。如果输入值为角度，要先将角度值转化为弧度值，即使用*R 或（MUL_R）指令将该角度值乘以π/180° 。IN 的寻址范围均为 VD、ID、QD、MD、SD、SMD、LD、AC、常数、*VD、*LD 和*AC。OUT 的寻址范围均为 VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*LD 和*AC。

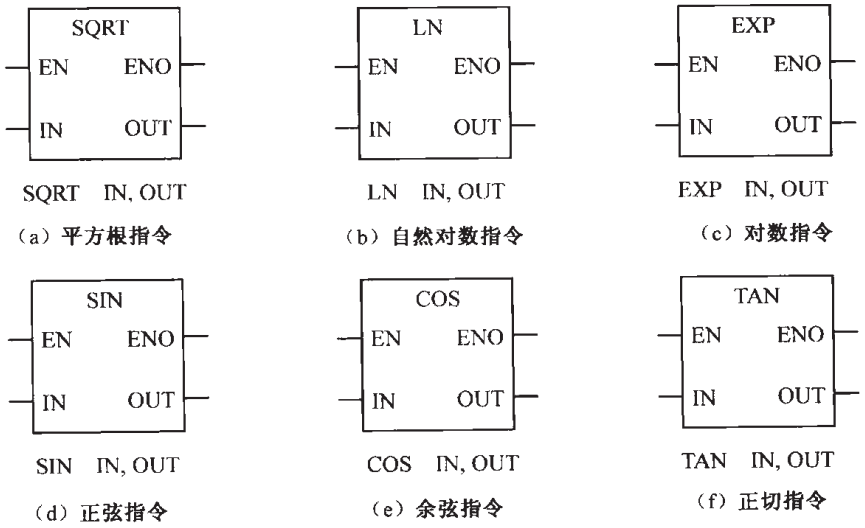


图 2-51　数学函数指令

实例 21：求解 75° 的正弦值

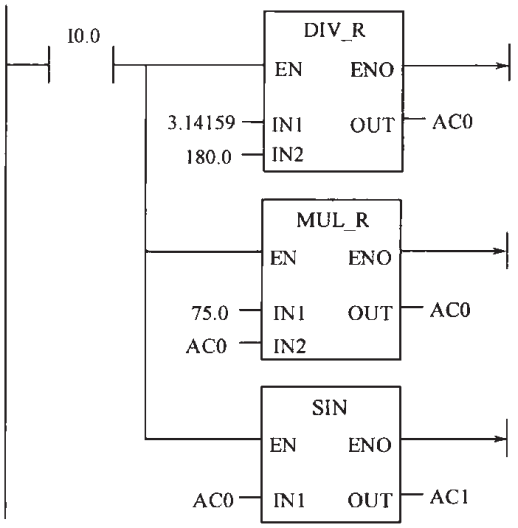


图 2-52 数学函数指令的应用

实例说明

在工业控制中有时为了计算某些三角形的高度或者某些距离需要用到数学函数指令。

实例实现

如果需要求解 sin75° 的值就可以利用图 2-52 所示的梯形图程序求解，得到的结果存储在 AC1 中。对应的语句为：

```
LD      I0.0
MOVR    3.14159, AC0
/R      180.0, AC0
*R      75.0, AC0
SIN     AC0, AC1
```

2.2.3 逻辑运算指令

逻辑运算是对逻辑数（无符号数）进行的逻辑处理。按运算性质的不同，有逻辑与、逻辑或、逻辑异或和取反等；按参与运算的操作数的长度，分为字节、字和双字逻辑运算操作。逻辑运算指令梯形图和语句表如图 2-53 所示。逻辑与、或、异或指令的功能是当允许信号 EN=1 时，把两输入操作数 IN1、IN2 按位进行逻辑与、或、异或等逻辑运算，然后得到逻辑输出结果 OUT，在语句表中 IN2 与 OUT 共用一个地址单元。逻辑取反指令的功能是，把操作数 IN 按位求反，得到逻辑输出结果 OUT，在语句表中 IN 与 OUT 共用一个地址单元。各逻辑指令操作数的寻址范围如表 2-10 所示。

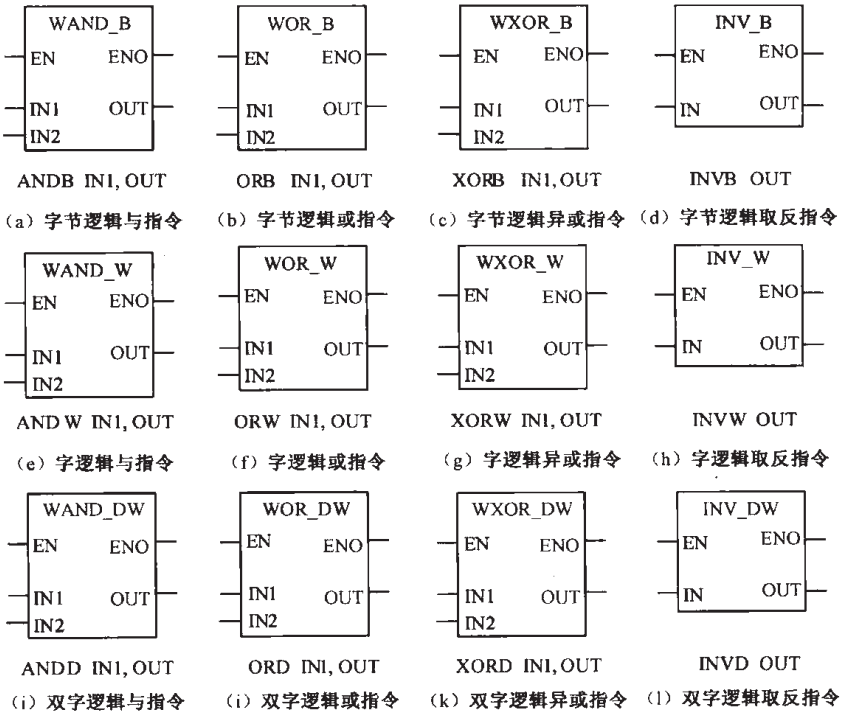


图 2-53 逻辑运算指令

表 2-10　逻辑运算指令操作数寻址范围

输入/输出	数 据 类 型	操作数寻址范围
IN1 IN2 IN	字节	VB、IB、QB、MB、SB、SMB、 LB、AC、常数、*VD、*LD、*AC
	字	VW、IW、QW、MW、SW、SMW、LW、 AIW、T、C、AC、常数、*VD、*LD、*AC
	双字	VD、ID、QD、MD、SD、SMD、LD、 HC、AC、常数、*VD、*LD、*AC
OUT	字节	VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*LD、*AC
	字	VW、IW、QW、SW、MW、SMW、 LW、T、C、AC、*VD、*LD、*AC
	双字	VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*LD、*AC

实例 22：利用逻辑运算指令实现数据分离

实例说明

同实例 19 中的数据分离一样，采用逻辑指令也可以实现。

实例实现

采用逻辑运算实现数据分离的梯形图程序如图 2-54 所示，其对应的语句为：

```
LD      SM0.0
MOVW    16#0FFF,MW2
ANDW    MW0,MW2
MOVW    16#F000,MW4
ANDW    MW0,MW4
SRW     MW4,12
```

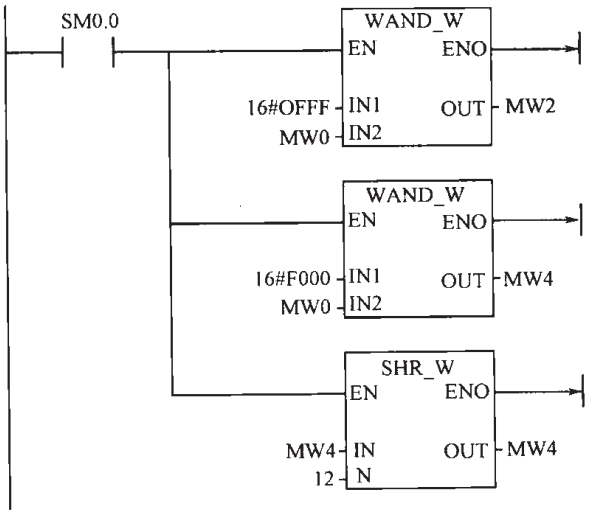


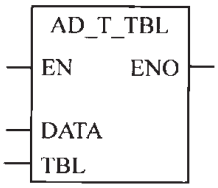
图 2-54　采用逻辑运算实现数据分离程序

实例分析

在程序运行中，将 MW0 中的数据与 16#0FFF 进行逻辑与运算后，将 MW0 的高 4 位全部变成了 0，因此也就实现了 MW0 的低 12 位的分离；将 MW0 中的数据与 16#F000 进行逻辑与运算后，将 MW0 的低 12 位全部变成了 0，然后进行移位操作，将数据向右移 12 位就实现了高 4 位的分离。灵活采用进行逻辑运算的值，同时结合移位指令，可以分离出任何所需位的值。

2.2.4 表功能指令

PLC 所用的数据多数是以数据表的形式存放在堆栈式的存储区中，为了对数据表的数据进行操作，需要使用表功能指令。表功能指令包括填表、查表、先进先出和后进先出指令。表功能指令实际就是对数据（只能是字型数据）的存取操作。



ATT DATA, TBL

图 2-55 填表指令

1. 填表指令

填表指令的梯形图和语句表如图 2-55 所示。DATA 为数值输入端，指出将被存储的字型数据或其地址；TBL 为表的首地址输入端，用于指明被访问的表格。

DATA 的寻址范围为 VW、IW、QW、MW、SW、SMW、LW、AIW、T、C、AC、常数、*VD、*LD 和*AC。

TBL 的寻址范围为 VW、IW、QW、SW、MW、SMW、LW、T、C、*VD、*LD 和*AC。

一个表由表地址（表的首地址）指明。表地址和第二个字地址所对应的单元分别存放两个参数值，第一个是最大填表数（TL），第二个是实际填表数（EC），指出已填入表的数据个数。

当允许信号 EN=1 时，将输入字型数据添加到指定的表中。新的数据添加在表中已有数据的后面。每向表中填加一个新的数据，实际填表数 EC 会自动加 1。一个表最多可填入 100 个数据（不包括最大填表数 TL 和实际填表数 EC）。

2. 查表指令

查表指令可以从字型数表中找出符合条件的数据在表中的数据编号，编号范围是 0～99。

查表指令的梯形图如图 2-56 所示。TBL 为表的首地址输入端，指明被访问的表格；PTN 为查表时进行比较的数据的输入端；INDX 用来指定存储地址，以存放表中符合查找条件数据的数据编号；CMD 是比较运算符编码输入端，它是一个 1～4 的数值，分别代表=、<>、<和>运算符。

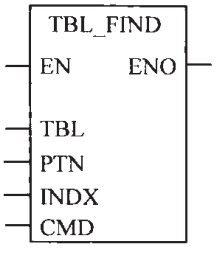


图 2-56 查表指令

TBL 的寻址范围为 VW、IW、QW、MW、SW、SMW、LW、T、C、*VD、*LD 和*AC。

PIN 的寻址范围为 VW、IW、QW、MW、SW、SMW、AIW、LW、T、C、AC、常量、*VD、*LD 和*AC。

INDX 的寻址范围为 VW、IW、QW、MW、SW、SMW、LW、T、C、AC、*VD、*LD 和*AC。

在查表指令的语句表中，运算符不采用编码形式，而是直接使用。

查表指令语句表表示："FND=　TBL, PTN, INDX"

"FND< >　TBL, PTN, INDX"

"FND<　TBL, PTN, INDX"

"FND>　TBL, PTN, INDX"

表查找指令执行之前，应先对 INDX 的内容清 0。当允许信号 EN=1 时，从 INDX 开始搜索表 TBL，寻找符合由 PTN 和 CMD 所决定的条件的数据。如果没有发现符合条件的数据，则 INDX 的值等于 EC；如果找到一个符合条件的数据，则将该数据在表中的编号装入 INDX 中。

表查找指令执行完成，找到一个符合条件的数据，如果想继续向下查找，必须先对 INDX 加 1，以重新激活表查找指令。

3. 表取数指令

从表中取出一个字型数据有两种方式：先进先出和后进先出。与取数方式相对应，表取数指令有两个：先进先出指令和后进先出指令，如图 2-57 所示。输入端 TBL 为表格的首地址，用以指明访问的表格；输出端 OUT 指明数值取出后要存放的目标地址单元。

TBL 的寻址范围为 VW、IW、QW、SW、MW、SMW、LW、T、C、*VD、*LD 和*AC。

DATA 的寻址范围为 VW、IW、QW、MW、SW、SMW、LW、T、C、AQW、AC、*VD、*LD 和*AC。

（a）先进先出指令　　（b）后进先出指令

图 2-57　表取数指令

（1）先进先出指令

当允许信号 EN=1 时，将表 TBL 的第一个数据项（不是第一个字）移出，并将它送到 DATA 指定的字单元中。

先进先出指令移出的数据总是最先进入表中的数据。每次从表中移出一个数据，剩余数据依次上移一个字单元位置，同时实际填表数 EC 会自动减 1。

（2）后进先出指令

当允许信号 EN=1 时，将表 TBL 的最后一个数据项移出，并将它送到 DATA 指定的字单元中。

后进先出指令移出的数据总是最后进入表中的数据。每次从表中移出一个数据，剩余数据位置保持不变，同时实际填表数 EC 会自动减 1。

实例 23：表中取数

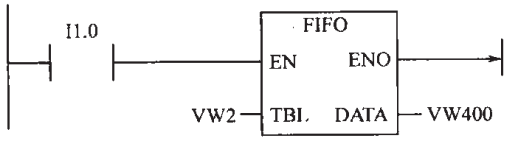
实例说明

在某些场合，需要用到较多的数据，在这种情况下，可以先把数据存取到表中，然后再从表中把数据取出来。

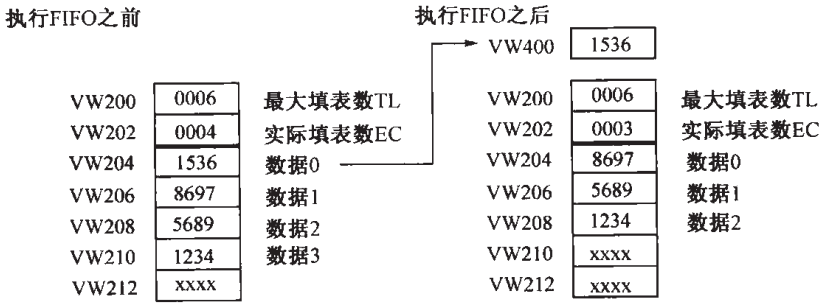
实例实现

图 2-58（a）是从表中取数的程序，其对应的语句为：

```
LD      I0.1
FIFO    VW200,VW400
```



（a）梯形图



（b）执行结果

图 2-58　表中取数程序及其执行结果

实例分析

当 I0.1 接通时，执行表中取数程序后，具体的取数过程如图 2-58（b）所示。

2.2.5　数据转换指令

数据转换指令是指对操作数的类型进行转换，包括数据的类型转换、码的类型转换以及数据和码之间的类型转换。

PLC 中的主要数据类型包括字节、整数、双整数和实数，主要的码制有 BCD 码（用二进制数来表示十进制数）、十进制数据和 ASCII 码（美国的国家标准所规定的用二进制数代表的字符）字符串等。不同性质的指令对操作数的类型要求不同，因此在指令使用之前需要将操作数转化成相应的类型，数据转换指令就可以完成这样的任务。

数据的转换指令主要包括：标准转换指令，ASCII 码转换指令，字符转换指令，编码、译码指令 4 类。

1. 标准转换指令

标准转换又有数字转换、四舍五入和取整、段码转换等。其中数字转换可以实现字节转为整数（BTI）、整数转为字节（ITB）、整数转为双整数（ITD）、双整数转为整数（DTI）、双整数转为实数（DTR）、BCD 码转为整数（BCDI）和整数转为 BCD 码（IBCD）。四舍五入指令（ROUND）将一个实数转为一个双整数值，并将四舍五入的结果存入 OUT 指定的变量中。取整指令（TRUNC）将一个实数转为一个双整数值，并将实数的整数部分作为结果存入 OUT 指定的变量中。段码转换是指用段码指令（SEG）产生一个点阵，用于点亮七段码显

示器的各个段。各指令的梯形图、语句表及其有关的信息如表 2-11 所示。其功能是，在输入允许信号 EN=1 的条件下，按照指令把 IN 输入的数据根据指令的形式转换为相应的数据输出到 OUT，根据转换的结果自动改变有关的特殊寄存器的值。标准转换指令操作数的寻址范围如表 2-12 所示。

标准转换中，要注意各转换的内容以及转换的数据类型。如果想将一个整数转换成实数以便于实现整数与实数间的运算，需要先将整数转换为双整数，然后再将双整数转换为实数，才能实现整数与实数间的运算；同样如果想将实数转换为整数，也需要将实数先转换为双整数，再将双整数转换为实数。需要注意的是，在将实数转换为整数，以及将双整数转换为整数时，不能超过转换数据的范围。

表 2-11　标准转换指令

指令名称	字节转为整数	整数转为字节	整数转为双整数	双整数转为整数
梯形图	B_I EN ENO IN OUT	I_B EN ENO IN OUT	I_DI EN ENO IN OUT	DI_I EN ENO IN OUT
语句表	BTI　IN, OUT	ITB　IN, OUT	ITD　IN, OUT	DTI　IN, OUT
备注	IN 的范围 0 到 9999 的整数，若 BCD 码无效则 SM1.6 为 1		符号位扩展到高字节中	转换数值太大而无法输出，SM1.1 置位且输出不变
指令名称	BCD 码转换为整数	整数转换为 BCD 码	双整数转为实数	
梯形图	BCD_I EN ENO IN OUT	I_BCD EN ENO IN OUT	DI_R EN ENO IN OUT	
语句表	BCDI　OUT	IBCD　OUT	DTR　IN, OUT	
备注	IN 范围 0 到 9999，若 BCD 码无效则 SM1.6 为 1		IN 为有符号数	
指令名称	四舍五入	取整指令	段码转换	
梯形图	ROUND EN ENO IN OUT	TRUNC EN ENO IN OUT	SEG EN ENO IN OUT	
语句表	ROUND　OUT	TRUNC　OUT	SEG　OUT	
备注	将实数 IN 转换成双整数，小数部分四舍五入	将实数 IN 转换成双整数，小数部分被舍去		

表 2-12　标准转换指令操作数寻址范围

输入/输出	数 据 类 型	操作数寻址范围
IN	字节	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, 常数
	字,整数	IW, QW, VW, MW, SMW, SW, T, C, LW, AIW, AC, *VD, *LD, *AC, 常数
	双整数	ID, QD, VD, MD, SMD, SD, LD, HC, AC, *VD, *LD, *AC, 常数
	实数	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, 常数
OUT	字节	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
	字,整数	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
	双整数,实数	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

段码指令是将输入字节低 4 位所表示的十六进制字符转换为七段码显示器的编码，如果要显示输入字节的高 4 位，则必须按照前面的实例把高 4 位分离出来，然后才能实现高 4 位的段码转换。表 2-13 给出了段码指令使用的七段码显示器的编码。每个七段显示码占用一字节，用它显示一个字符。

表 2-13　七段码显示器编码

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

输入 LSD	七段码显示器	输出 -gfe dcba
8	8	0111 1111
9	9	0110 0111
A	A	0111 0111
B	b	0111 1100
C	C	0011 1001
D	d	0101 1110
E	E	0111 1001
F	F	0111 0001

实例 24：BCD 码与整数之间的转换

实例说明

BCD 码与整数之间的转换，在实际应用中会经常遇到。例如，要将一个两位的十进制数利用数码管显示出来，就需要先将该数转换为 BCD 码，然后再使用段码指令，将转换后的 BCD 码转换为七段码显示器的编码，通过输出口与七段数码管相连接，才能显示。

实例实现

BCD 码与整数之间的转换梯形图程序如图 2-59 所示，其对应的语句为：

```
LD      I0.3
MOVW    VW0,VW2
IBCD    VW2    //将 VW0 中的整数转换为 BCD 码结果存储到 VW2 中
LD      I0.4
MOVW    VW2,VW0
BCDI    VW0    //将 VW2 中的 BCD 码转换为整数结果存储到 VW0 中
```

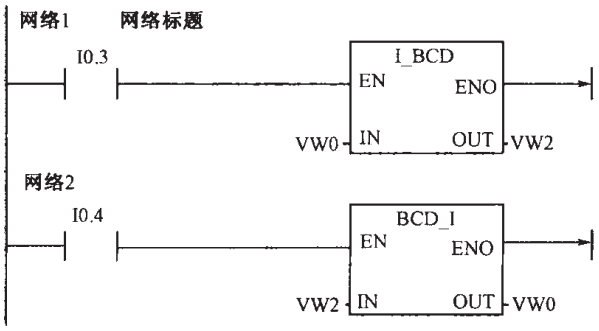


图 2-59　BCD 码与整数之间的转换程序

实例 25：双整数与实数之间的转换

实例说明

双整数与实数之间的转换经常用于数学计算中，在 PLC 数学运算中要求两个进行运算的操作数数据类型要一致，这就会出现双整数与实数之间的转换。

实例实现

双整数与实数之间的转换程序梯形图如图 2-60 所示，其对应的语句为：

```
LD      I0.3
DTR     VD0,VD4       // VD0 中的双整数转换为实数结果存储到 VD4 中
LD      I0.4
TRUNC   VD4,VD8       // VD4 中的实数转换为双整数结果存储到 VD8 中，小数舍去
LD      I0.4
```

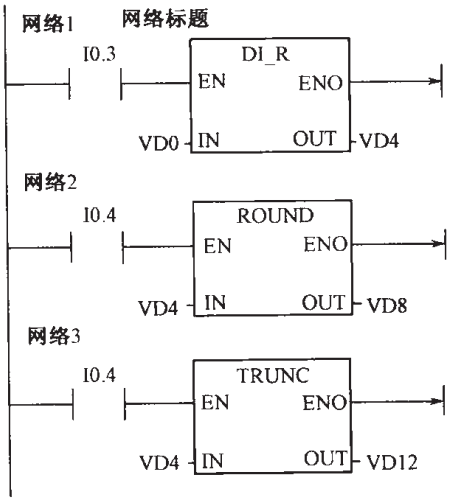


图 2-60　双整数与实数之间的转换程序

实例 26：英寸转换为厘米

实例说明

在 PLC 的运算和显示中经常会出现单位之间的转换，一般单位之间的转换用 PLC 数学运算指令就可以实现。但是对于某些单位之间的转换（如把英寸转换为厘米），由于不是整数的除法，就需要先对数据进行转换，然后才进行单位之间的换算。

实例实现

英寸转换为厘米的 PLC 程序梯形图如图 2-61 所示，其相应的语句为：

```
LD      I0.0
ITD     C10, AC1      //将计数器中要转换的数值（英寸）载入 AC1
```

```
DTR     AC1, VD0      //将数值转换为实数
MOVR    VD0,VD8
*R      VD4,VD8       //VD4 中的数值为 2.54，乘以 2.54 后转换为厘米
ROUND   VD8,VD12      //将数值转换为整数
```

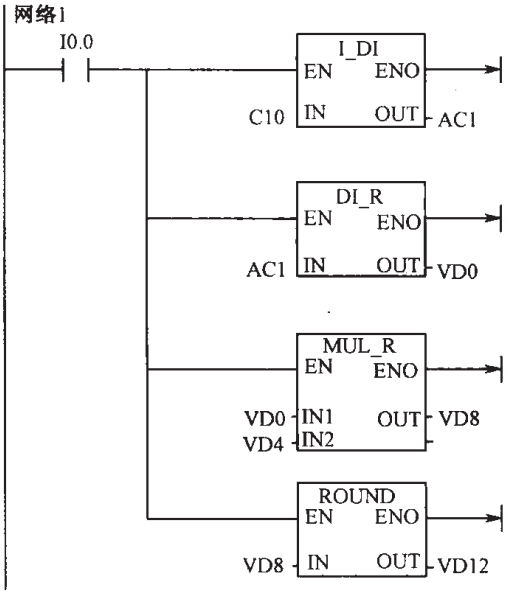


图 2-61　英寸与厘米的单位转换程序

2. ASCII 码转换指令

ASCII 码转换指令主要是指 ASCII 码与十六进制数之间的转换指令以及将整数、双整数、实数转换为 ASCII 码。

（1）ASCII 码与十六进制数之间的转换指令

ASCII 码与十六进制数之间的转换指令的梯形图和语句表如图 2-62 所示。

IN 和 OUT 的寻址范围为 VB、IB、QB、MB、SB、SMB、LB、*VD、*LD 和*AC。

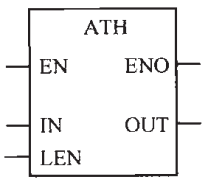
LEN 的寻址范围为 VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*LD 和*AC。

ASCII 码转换为十六进制数指令的功能是：当允许信号 EN=1 时，ASCII 码转换为十六进制数指令（ATH）将从 IN 开始的长度为 LEN（最大长度为 255）的 ASCII 码转换为十六进制数，并将结果送到 OUT 开始的字节进行输出。有效的 ASCII 码输入字符是 0～9（十六进制数值 30～39）和大写字符 A～F（十六进制数值 41～46）。而十六进制数转换为 ASCII 码指令的功能是：当允许信号 EN=1 时，十六进制数转换为 ASCII 码指令（HTA）将从输入字节 IN 开始的长度为 LEN（最大长度为 255）的十六进制数字转换为 ASCII 字符，并将结果送到 OUT 开始的字节进行输出。有效的十六进制输入数值是 30～39（ASCII 码字符 0～9）和 41～46（ASCII 码大写字符 A～F）。

（2）整数转换为 ASCII 码指令

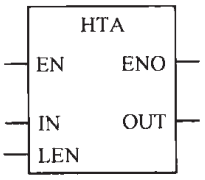
整数转换为 ASCII 码指令的梯形图和语句表如图 2-63 所示。

IN 的寻址范围为 VW、IW、QW、MW、SW、SMW、LW、AIW、T、C、AC、常数、*VD、*LD 和*AC。



ATH　IN, OUT, LEN

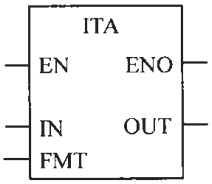
（a）ASCII码转换为十六进制数指令



HTA　IN, OUT, LEN

（b）十六进制数转换为ASCII码指令

图 2-62　ASCII 码与十六进制数之间的转换指令



ITA　IN, OUT, FMT

图 2-63　整数转换为 ASCII 码指令

FMT 的寻址范围为 VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*LD 和*AC。

OUT 的寻址范围为 VB、IB、QB、MB、SB、SMB、LB、*VD、*LD 和*AC。

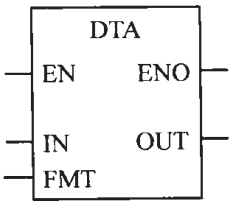
当允许信号 EN=1 时，将输入端（IN）的有符号整数根据格式 FMT 要求转换成 ASCII 码，转换结果置于 OUT 为起始字节地址的 8 个连续字节的输出缓冲区中。

格式 FMT 指定 ASCII 码字符串中分隔符的位置和表示方法，即小数点右侧的转换精度，以及是否将小数点显示为逗号或点号。FMT 占用一个字节，高 4 位必须为 0，低 4 位用 cnnn 表示。c 位指定整数和小数之间的分隔符：c=1，用逗号分隔；c=0，用小数点分隔。nnn 指定输出缓冲区中小数点右侧的位数，nnn 的有效范围是 0～5。指定小数点右侧的数字为 0 会使显示的数值无小数点。对于大于 5 的 nnn 数值为非法格式，此时无输出，用 ASCII 空格填充输出缓冲区。

输出缓冲区的格式符合以下规则：正值写入输出缓冲区时不带正号、负值写入输出缓冲区时带负号、小数点左侧的开头的 0（除去靠近小数点的那个之外）被省略、输出缓冲区内数值右对齐。

（3）双整数转换为 ASCII 码指令

双整数转换为 ASCII 码指令的梯形图和语句表如图 2-64 所示。



DTA　IN, OUT, FMT

图 2-64　双整数转换为 ASCII 码指令

IN 的寻址范围为 VD、ID、QD、MD、SD、SMD、LD、HC、AC、常数、*VD、*LD 和*AC。

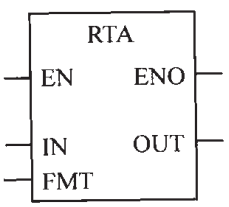
FMT 的寻址范围为 VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*LD 和*AC。

OUT 的寻址范围为 VB、IB、QB、MB、SB、SMB、LB、*VD、*LD 和*AC。

当允许信号 EN=1 时，将输入端（IN）的有符号双整数根据格式 FMT 要求转换成 ASCII 码，转换结果存入以 OUT 为起始字节地址的 12 个连续字节的输出缓冲区中。格式 FMT 与 ITA 指令中的 FMT 格式相同。

（4）实数转换为 ASCII 码指令

实数转换为 ASCII 码指令的梯形图和语句表如图 2-65 所示。



RTA　IN, OUT, FMT

图 2-65　实数转换为 ASCII 码指令

IN 的寻址范围为 VD、ID、QD、MD、SD、SMD、LD、AC、常数、*VD、*LD 和*AC。

FMT 的寻址范围为 VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*LD 和*AC。

OUT 的寻址范围为 VB、IB、QB、MB、SB、SMB、LB、*VD、*LD 和*AC。

当允许信号 EN=1 时，将输入端（IN）的实数根据格式 FMT 要求转换成 ASCII 码，转换结果置于 OUT 为起始字节地址的 3～15 个连续字节的输出缓冲区中。

格式 FMT 指定 ASCII 码字符串中分隔符的位置和表示方法，即小数点右侧的转换精度，以及是否将小数点显示为逗号或点号。FMT 占用一个字节，高 4 位用 ssss 表示，ssss 的值指定输出缓冲区的字节数（3～15 个），0、1 或 2 个字节无效，并且输出缓冲区的字节数应大于输入实数小数点右边的位数，低 4 位的定义与 ITA 指令相同。

输出缓冲区的格式符合以下规则：正值写入输出缓冲区时不带正号、负值写入输出缓冲区时带负号、小数点左侧的开头的 0（除去靠近小数点的那个之外）被省略、小数点右侧的数值按照指定的小数点右侧的数字位数被四舍五入、输出缓冲区的大小应至少比小数点右侧的数字位数多 3 个字节、输出缓冲器内数值右对齐。

实例 27：ASCII 码与十六进制数之间的转换

实例说明

有一些显示元件需要显示的是 ASCII 值，这样就要用到 ASCII 码与数之间的转换。

实例实现

图 2-66 是将 ASCII 码转换为十六进制数和将整数、实数转换为 ASCII 码的程序与转换结果，其对应的语句为：

```
LD      I3.0
ATH     VB30, VB40, 3
LD      I4.3
ITA     VW10, VB20, 16#0B
RTA     VD40, VB50, 16#A3
```

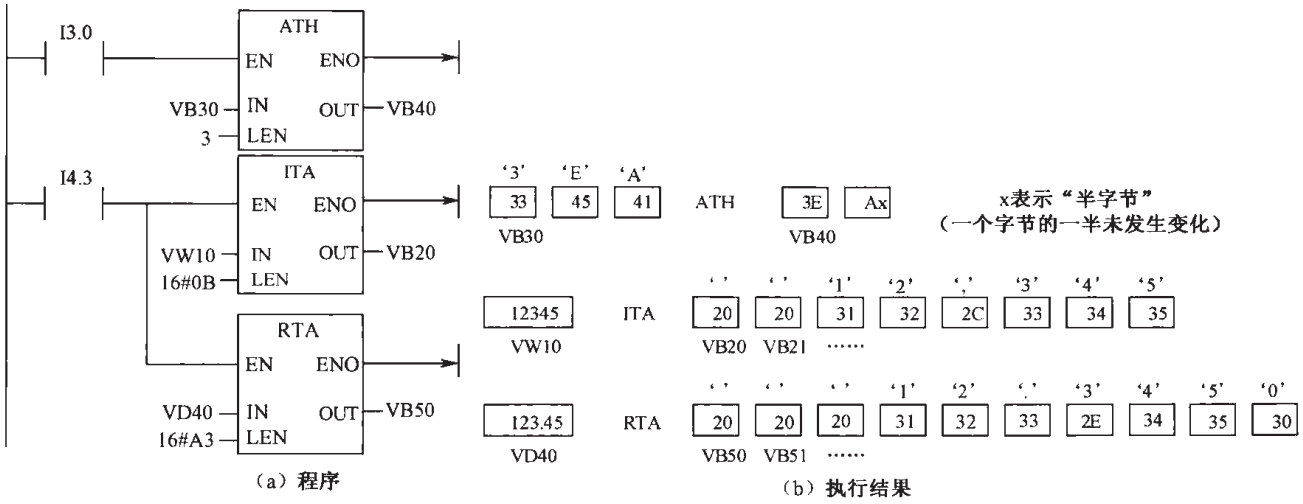


图 2-66　整数、实数转换为 ASCII 码指令的应用

实例分析

程序具体的执行过程为：从 VB30 开始的长度为 3 的 ASCII 码转换为十六进制数，并将结果送到 VB40 开始的字节进行输出；将 VW10 中的整数值转换为从 VB20 开始的 8 个 ASCII 码字符，使用 16#0B 的格式（用逗号作小数点，保留 3 位小数）；将 VD40 中的实数值转换成从 VB50 开始的 10 个 ASCII 码字符，使用 16#A3 的格式（用点号作小数点，后面跟 3 位小数）。

3. 编码、译码指令

编码、译码指令的梯形图和语句表如图 2-67 所示。在编码指令中 IN 的寻址范围为 VW、IW、QW、MW、SW、SMW、LW、AIW、T、C、AC、常数、*VD、*LD 和*AC；OUT 的寻址范围为 VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*LD 和*AC。在译码指令中 IN 的寻址范围为 VB、IB、QB、MB、SB、SMB、LB、AC、常数、*VD、*LD 和*AC；OUT 的寻址范围为 VW、IW、QW、MW、SW、SMW、LW、AQW、T、C、AC、*VD、*LD 和*AC。

编码指令（ENCO）的功能是：当允许信号 EN=1 时，将 16 位字型输入数据 IN 中值为 1 的最低有效位的位号（0～15）编码成 4 位二进制数，输出到 OUT 所指定的字节型单元的低 4 位。也就是 OUT 的低 4 位值为数据 IN 中值为 1 的最低位的位号。而译码指令（DECO）的功能是：当允许信号 EN=1 时，根据 8 位字节型输入数据 IN 的低 4 位所表示的位号（0～15）将 OUT 所指定的字单元的对应位置 1，其他位置 0。

（a）编码指令　　（b）译码指令

图 2-67　编码和译码指令

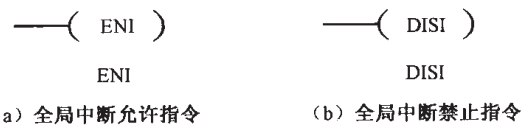
2.2.6　中断指令

中断是计算机在实时处理和实时控制中不可缺少的一项技术，应用十分广泛。所谓中断，是指当控制系统执行正常程序时，系统中出现了某些急需处理的异常情况或特殊请求，这时系统暂时停止执行当前程序，转去对随机发生的紧迫事件进行处理（执行中断服务程序），当该事件处理完毕后，系统自动回到原来被中断的程序继续执行。

中断事件的发生具有随机性，中断在 PLC 应用系统中的人机联系、实时处理、通信处理和网络中非常重要。S7-200 可以引发的中断事件分为通信口中断、I/O 中断和时基中断 3 类，共 34 项（编号 0～33），按优先级排列的中断事件如表 2-14 所示。

表 2-14 按优先级排列的中断事件

中断事件号	中断事件描述	优先级分组	按组排列的优先级
8	通信端口 0 接收字符	通信（最高）	0
9	通信端口 0 发送完成		0
23	通信端口 0 接收信息完成		0
24	通信端口 1 接收信息完成		1
25	通信端口 1 接收字符		1
26	通信端口 1 发送完成		1
19	PTO0 脉冲输出完成	I/O （中等）	0
20	PTO1 脉冲输出完成		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=设定值）		10
27	HSC0 输入方向改变		11
28	HSC0 外部复位		12
13	HSC1 CV=PV （当前值=设定值）		13
14	HSC1 输入方向改变		14
15	HSC1 外部复位		15
16	HSC2 CV=PV （当前值=设定值）		16
17	HSC2 输入方向改变		17
18	HSC2 外部复位		18
32	HSC3 CV=PV （当前值=设定值）		19
29	HSC4 CV=PV （当前值=设定值）		20
30	HSC4 输入方向改变		21
31	HSC4 外部复位		22
33	HSC5 CV=PV （当前值=设定值）		23
10	定时中断 0	定时（最低）	0
11	定时中断 1		1
21	T32 CT=PT （当前值=设定值）		2
22	T96 CT=PT （当前值=设定值）		3



（a）全局中断允许指令　　（b）全局中断禁止指令

图 2-68 全局中断允许/禁止指令的梯形图和语句表

1. 全局中断允许/禁止指令

全局中断允许/禁止指令的梯形图和语句表如图 2-68 所示。

（1）全局中断允许指令

全局性地允许所有被连接的中断事件。

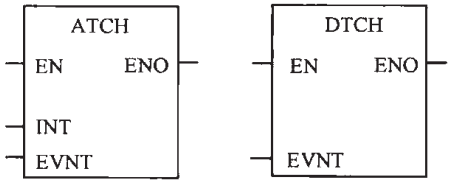
（2）全局中断禁止指令

全局性地禁止处理所有的中断事件。执行该指令后，出现的中断事件就进入中断队列排队等候，直到全局中断允许指令重新允许中断。

CPU 进入 RUN 运行模式时，自动禁止所有中断。在 RUN 运行模式中执行全局中断允许指令后，允许所有中断。只有在全局中断允许的条件下，系统才有可能执行中断。在全局中断禁止的条件下，任何中断程序都是不会被执行的。

2. 中断连接/分离指令

中断连接/分离指令的梯形图和语句表如图 2-69 所示。其操作数的寻址范围是：INT 为字节常量，表示中断程序号，取值范围为 0～127；EVNT 为字节常量，表示中断事件号，取值范围根据 CPU 的型号有所不同，CPU 221/222 为 0～12、19～23、27～33，CPU 224 为 0～23、27～33，CPU 226 为 0～33。



（a）中断连接指令　　（b）中断分离指令

图 2-69　中断连接/分离指令

中断连接指令用来建立某个中断事件（EVNT）和某个中断程序（INT）之间的联系，并允许这个中断事件。在调用一个中断程序前，必须用中断连接指令，建立某中断事件与中断程序的连接。当把某个中断事件和中断程序建立连接后，该中断事件发生时会自动开中断。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与多个中断程序建立连接。中断分离指令用来解除某个中断事件（EVNT）和某个中断程序（INT）之间的联系，并禁止该中断事件，使中断回到不激活或无效状态。

3. 中断返回指令

中断返回指令（条件返回）梯形图和语句表如图 2-70 所示。

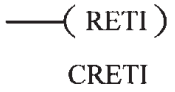


图 2-70　中断返回指令

中断返回指令用于中断程序中，根据前面逻辑条件决定是否从中断程序返回主程序。

中断程序必须以无条件中断返回指令作结束。S7-200 PLC 的 STEP7-Micro/WIN 编程软件自动在中断程序结尾添加了无条件中断返回指令，不需要用户自己再在程序末尾添加。

4. 使用中断应注意的问题

（1）中断程序中可以调用子程序

累加器和逻辑堆栈式的存储器在中断程序和被调用的子程序中是共用的。

（2）中断程序和主程序间可以共享数据

主程序和中断程序可以相互提供要用到的数据，但要考虑中断事件异步特性的影响，解决共享数据的一致性问题，因为中断事件会在主程序执行的任何地方出现。

有几种可以确保在主程序和中断程序之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。

语句表（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，那么通过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。

梯形图（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，那么只用 Move 指令（MOVB、MOVW、MOVD、MOVR）访问共享存储器单元，可以保证正确的共享访问。这些 Move 指令执行时不受中断事件影响。

语句表或梯形图程序共享多个变量。如果共享数据由一些相关的字节、字或双字组成，那么可以用全局中断允许/禁止指令（DISI 和 ENI）来控制中断程序的执行。在程序开始对共享存储器单元操作的地方禁止中断，所有影响共享存储器单元的操作完成后，再允许中断，但这种方法会导致对中断事件响应的延迟。

（3）通信口中断

PLC 的串行通信口可由梯形图或语句表程序来控制。通信口的这种操作模式称为自由端口模式。在自由端口模式下，可用程序定义波特率、每个字符位数、奇偶校验和通信协议。在执行主程序时，申请中断，才能定义自由端口模式，利用接收和发送中断可简化程序对通信的控制。

（4）I/O 中断

I/O 中断包括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 用输入 I0.0～I0.3 的上升沿或下降沿产生中断，则使上升沿或下降沿发生的事件被输入端子捕获。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条件。

高速计数器中断允许响应诸如当前值等于预置值、计数器计数方向改变和计数器外部复位等事件而产生的中断。每种高速计数器通过申请中断对高速事件实时响应。

脉冲串输出中断在完成指定脉冲输出时发生，指示脉冲数输出已完成。

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。

（5）时基中断

时基中断包括定时中断和定时器 T32/T96 中断。

定时中断又分为定时中断 0 和定时中断 1。定时中断 0 把周期时间写入 SMB34，定时中断 1 把周期时间写入 SMB35。CPU 支持定时中断，用定时中断指定一个周期性的活动，周期以 1 ms 为增量单位，周期时间可以为 5～255 ms。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，就开始计时。在连接期间，系统捕捉周期时间值，因而后来对 SMB34 和 SMB35 的更改不会影响周期。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等待，直到中断允许或队列满。

定时器 T32/T96 中断允许及时地响应一个给定时间间隔。这些中断只支持 1 ms 分辨率的延时接通定时器（TON）和延时断开定时器（TOF）T32 和 T96。定时器 T32 和 T96 在其他方面工作正常。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 1 ms

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。

（6）中断的优先级和排队

中断按固定的优先级顺序执行：通信中断（最高优先级）、I/O 中断（中等优先级）、时基中断（最低优先级）。

PLC 的 CPU 接到中断请求后，先查看优先级排队，以优先级最高到最低的顺序处理事件，没有中断嵌套。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，只执行一个中断程序。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。每个中断队列允许的最多中断数如表 2-15 所示。如果有多于队列所能保存数目的中断出现，则由中断队列溢出的特殊标志存储器位表明丢失的中断事件的类型，如表 2-16 所示。中断队列溢出的特殊标志存储器位只在中断程序中使用，因为在队列变空或返回到主程序时，这些位会被复位。

表 2-15　每个中断队列允许的最多中断数

队　　列	CPU221	CPU222	CPU224	CPU226
通信中断队列	4	4	4	8
I/O 中断队列	16	16	16	16
定时中断队列	8	8	8	8

表 2-16　中断队列溢出的特殊标志存储器位

描述	SM 位
通信中断队列溢出	SM4.0
I/O 中断队列溢出	SM4.1
定时中断队列溢出	SM4.2

（7）使用中断的限制

一个程序内最多可有 128 个中断。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。否则，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”。在中断程序内不能使用 DISI、ENI、HDEF、LSCR 和 END 指令。

（8）中断程序编程步骤

首先建立中断程序（同建立子程序方法相同），接着在中断程序中编写其应用程序，最后在主程序中使用中断指令。

实例 28：处理输入/输出中断程序

实例说明

在 PLC 控制中，经常会用到需要及时处理 PLC 某些端口的输入数据，这时利用 I/O 中断就很容易实现。

实例实现

如图 2-71 所示是处理定时中断的程序，其相应的语句为：

```
// 主程序
LD      SM0.1          //仅首次扫描时，SM0.1 才为 1，进行以下初始化
MOVB    0,AC0          //将计数累加器 AC0 清 0
ATCH    INT_0:INT0,0   //输入 I0.0 为上升沿时激活事件中断 0
```

```
ATCH    INT_1:INT1,1      //输入 I0.0 为上升沿时激活事件中断 1
ENI                       //允许中断
LDN     M0.0              //如果存储器的标志位 M0.0 为 0 状态
AB>=    16#FE,AC0         //且计数累加器 AC0 的当前计数值小于或等于 254
A       SM0.5             //且 0.5s 脉冲
EU                        //且上升沿
INCW    AC0               //那么计数累加器 AC0 加 1
LD      M0.0              //如果存储器的标志位 M0.0 为 1 状态
AB<-    16#1,AC0          //且计数累加器 AC0 的当前计数值大于或等于 1
A       SM0.5             //且 0.5s 脉冲
EU                        //且上升沿
DECW    AC0               //那么计数累加器 AC0 减 1
//中断程序 0 结束
LD      SM0.0             //SM0.总是 1
S       M0.0,1            //将存储器的标志位 M0.0 置成 1
//中断程序 1 结束
LD      SM0.0             //SM0.总是 1
R       M0.0,1            //将存储器的标志位 M0.0 置成 0
```

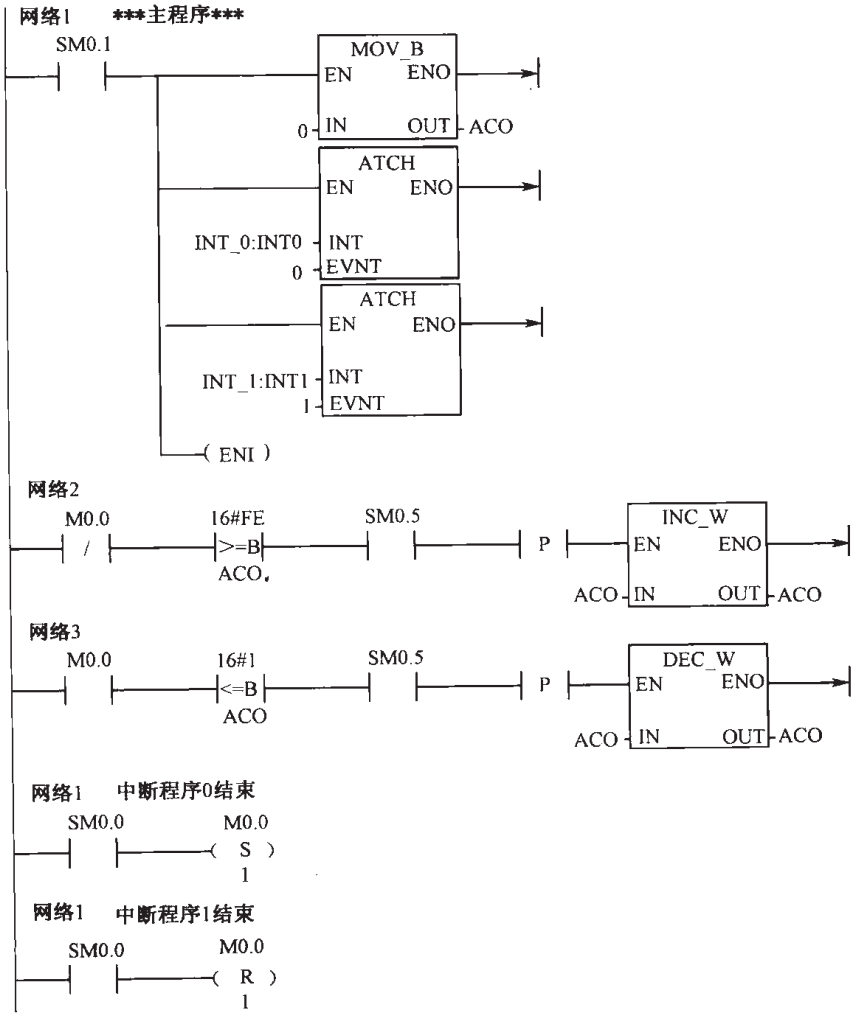


图 2-71 输入/输出中断程序

实例分析

本实例中，根据输入口 I0.0 来实现 0～255 的计数。当输入 I0.0 置为 1，则程序减计数；如果将输入 I0.0 置为 0，则程序加计数。在程序运行中如果输入 I0.0 的状态改变，则将立即激活输入/输出中断程序，中断程序 0 或 1 分别将存储器位 M0.0 置成 1 或 0。

实例 29：处理定时中断程序

实例说明

在很多情况下需要多次使用中断，而且在使用中需要断开某些中断，也就是说在不同的条件下需要关闭某些中断而连接另外的中断。

实例实现

如图 2-72 所示是处理定时中断的程序，其相应的语句为：

```
LD      SM0.1          //仅首次扫描处理
MOVB    50,SMB34       //设置定时中断 0 的事件基准为 50ms
MOVB    100,SMB35      //设置定时中断 1 的事件基准为 100ms
ATCH    INT_0:INT0.10  //指定定时中断事件 10 调用中断程序 0
ATCH    INT_1:INT1.11  //指定定时中断事件 11 调用中断程序 1
ENI                    //中断允许
LD      I0.1           //输入 I0.1
EU                     //上升沿
DTCH    10             //切断定时中断事件 10 与中断程序 0 的联系
DTCH    11             //切断定时中断事件 11 与中断程序 1 的联系
MOVB    100,SMB34      //设置定时中断 0 的事件基准为 100ms
MOVB    200,SMB35      //设置定时中断 1 的事件基准为 200ms
ATCH    INT_0:INT0.10  //恢复定时中断事件 10 调用中断程序 0
ATCH    INT_1:INT1.11  //恢复定时中断事件 11 调用中断程序 1
//中断程序 0
LD      SM0.0          //特殊存储器位 SM0.0 总是 1
S       Q0.0           //把输出 Q0.0 置位
//中断程序 1
LD      SM0.0          //特殊存储器位 SM0.0 总是 1
R       Q0.0,1         //把输出 Q0.0 复位
```

实例分析

在本例中 PLC 首次扫描时用定时中断来产生接通为 50 ms，关为 100 ms 的信号，并从 Q0.0 输出，然后一直保持，直到连接输入端 I0.1 的开关接通时，分离了首次扫描所利用的定时中断，然后连接新的定时中断，在新的定时中断中实现的是产生接通 100 ms，关闭 200 ms 的信号，并从 Q0.0 输出。

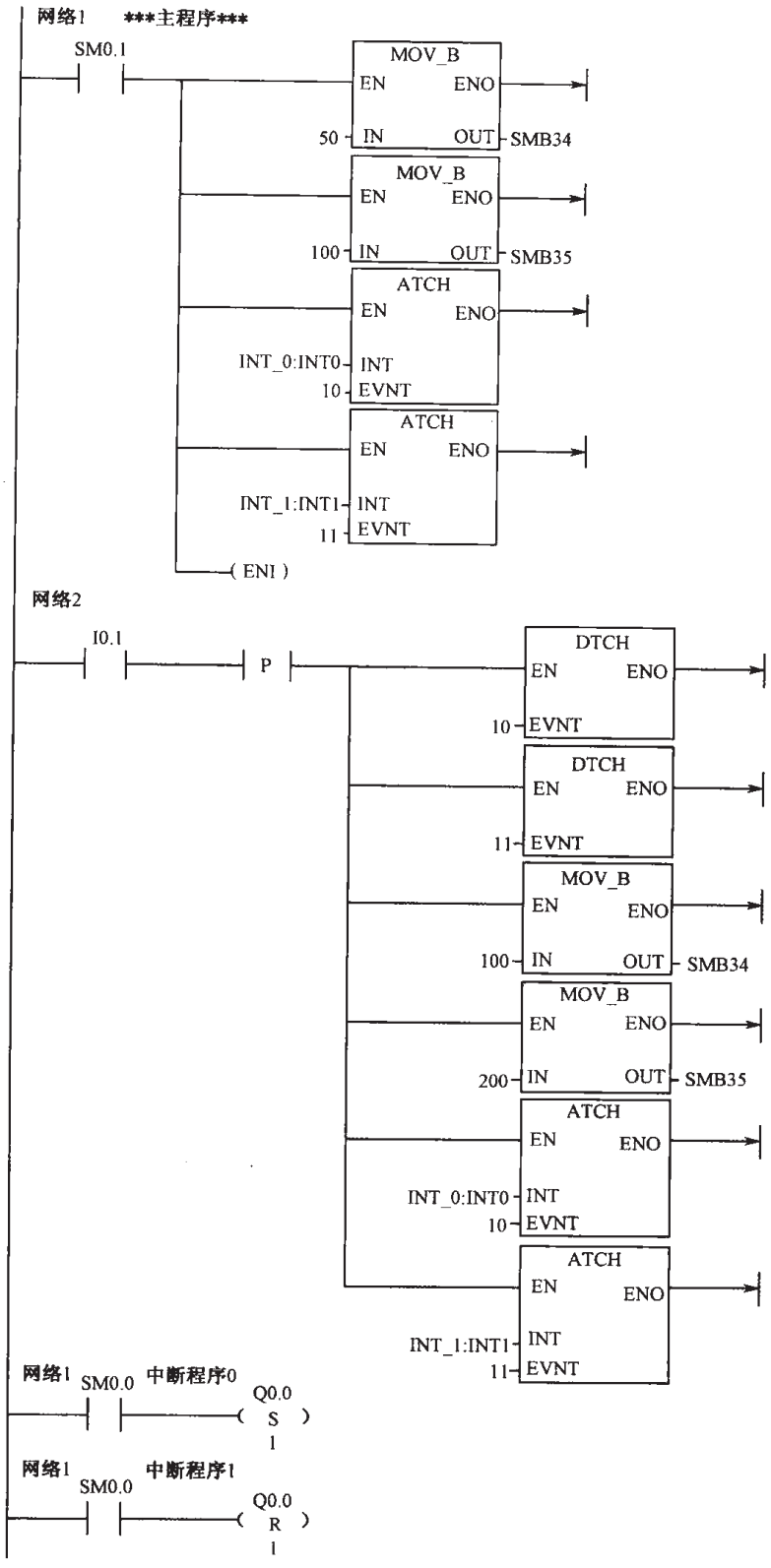


图 2-72 处理定时中断程序

实例 30：模拟量的定时采集

实例说明

在 PLC 的应用中，往往需要定时地采集模拟量以便于实现 PLC 的实时控制。

实例实现

模拟量的定时采集程序如图 2-73 所示，其对应的语句为：

```
LD          SM0.1       //首次扫描
CALL        SBR_0       //调用子程序 0
//子程序 SBR_0
LD          SM0.0
MOVB        100, SMB34  //设置定时中断的时间间隔为 100ms
ATCH        INT_0 ,10   //连接 INT_0 到定时中断 0（事件 10）
ENI                     //全局中断允许
//中断程序 0, 每 100ms 读 AIW4 的值
LD          SM0.0
MOVW        AIW4, VW100
```

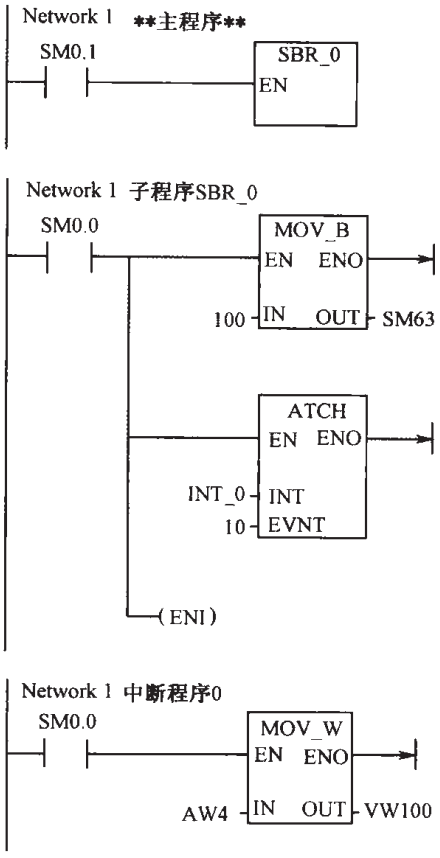


图 2-73　模拟量的定时采集程序

实例分析

首次扫描时调用子程序 0；在子程序中设置定时中断的时间间隔为 100 ms，连接中断程序 0 到定时中断 0（中断事件号 10），全局中断允许；在中断程序中，每 100 ms 读 AIW4 的值。

2.2.7 时钟指令

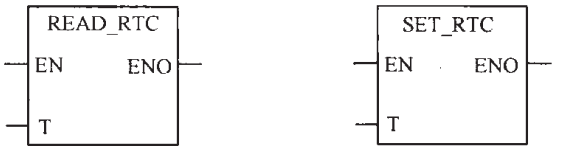
时钟指令用来读取或设定系统的日期和时间。利用时钟指令可以调用系统实时时钟，这对于实现控制系统的运行监视、运行记录等十分方便。

S7-200 PLC 中，CPU221 和 CPU222 安装有时钟卡，CPU244 和 CPU226 有内置时钟。内置时钟的时钟指令设有 8 个字节的时钟缓冲区，其格式如表 2-17 所示。

表 2-17 时钟缓冲区的格式

字　节	T	T+1	T+2	T+3	T+4	T+5	T+6	T+7
含　义	年	月	日	小时	分钟	秒	保留	星期
范　围	00～99	01～12	01～31	00～23	00～59	00～59	00	0～7
说明：1. 所有日期和时间值必须采用 BCD 格式编码。 2. 表示年份时，只用最低两位数（例如，2002 年表示为 16#02）。 3. 表示星期时，16#1=星期日，16#7=星期六，16#0 禁止星期表示法								

时钟指令主要有读取实时时钟指令和设定实时时钟指令，时钟指令的梯形图和语句表如图 2-74 所示。



（a）读取实时时钟指令　　（b）设定实时时钟指令

图 2-74 时钟指令

T 为时钟缓冲区的首地址，寻址范围为 VB、IB、QB、MB、SMB、SB、LB、*VD、*LD 和*AC。

读取实时时钟指令用来读取实时时钟。其功能是当 EN 输入有效时，读取系统当前时间和日期，并把它装入以 T 为起始字节地址的 8 个字节缓冲区。

设定实时时钟指令用来设定 PLC 实时时钟。其功能是当 EN 输入有效时，将含有时间和日期的 8 个字节缓冲区（起始地址是 T）的内容装入 PLC 时钟。S7-200 PLC 不检查和核实日期是否准确。无效日期（如 2 月 30 日）也可以被接受。因此，必须确保输入数据的准确性。

不能同时在主程序和中断程序中使用时钟指令，否则产生非致命错误，中断程序中的时钟指令将不被执行。

实例 31：设定 CPU 时钟

实例说明

在利用 PLC 进行控制时，有时需要将 CPU 的时钟设定正确，从而能准确地控制时间。这样就需要设定 CPU 的时钟。

实例实现

具体的设定 CPU 时钟的例子的程序如图 2-75 所示，其对应的语句为：

```
LD      I0.0
MOVB    16#06,VB100   //年：06 年
MOVB    16#05,VB101   //月：5 月
MOVB    16#30,VB102   //日：30 日
MOVB    16#21,VB103   //时：21 时
MOVB    16#00,VB104   //分：00 分
MOVB    16#00,VB105   //秒：00 秒
MOVB    16#0,VB106    //未分配
MOVB    16#03,VB107   //星期：星期二
TODW    VB100         //写时钟
```

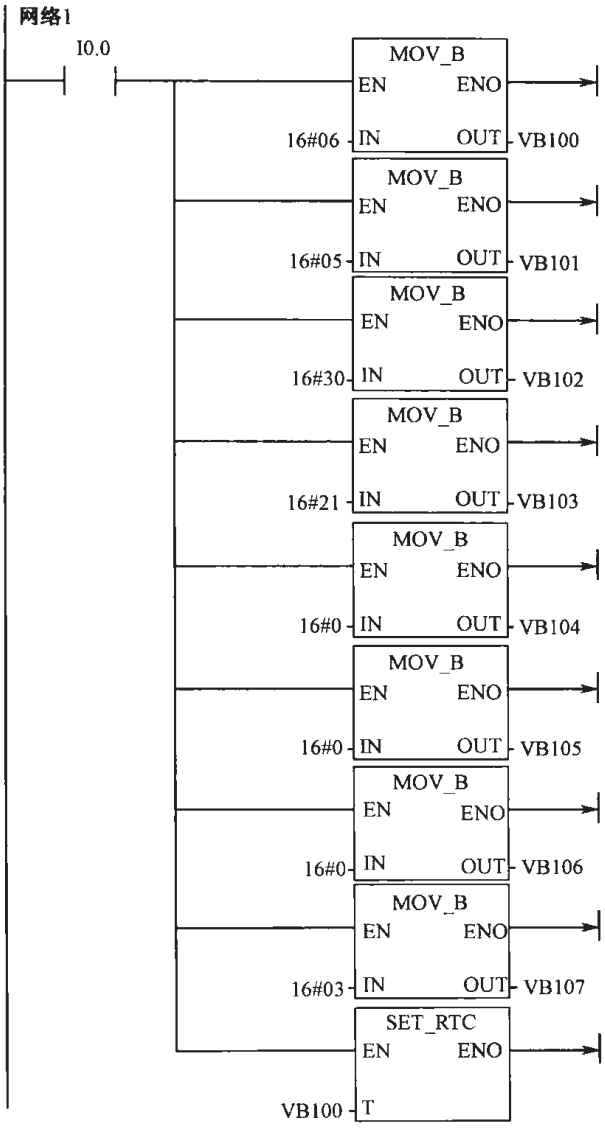


图 2-75　设定 CPU 时钟程序

实例分析

按照该程序 CPU 的时间将被设定到 06 年 5 月 30 日 21 时，星期二。

2.2.8 高速处理类指令

上面所讲述的 PLC 指令，除了部分定时指令和中断指令不受 PLC 扫描速度的影响外，其他指令都会受到 PLC 扫描速度的影响。另外，PLC 还有两类高速处理指令也不受 PLC 扫描速度的影响，这两类指令分别是高速计数器指令和高速脉冲输出指令。

1. 高速计数器及其指令

（1）高速计数器

高速计数器是脱离主机的扫描周期独立计数的，它可以对脉宽小于主机扫描周期的高速脉冲准确计数，即高速计数器计数的脉冲输入频率比 PLC 扫描频率高得多。因此高速计数器不像普通计数器要受 PLC 扫描速度的影响，能有效防止发生计数脉冲信号丢失的现象。高速计数器常用于电动机转速检测等场合。使用时，可由编码器将电动机的转速转化成脉冲信号，再用高速计数器对转速脉冲信号进行计数，当高速计数器的当前值等于预设值、计数方向改变或发生复位时，高速计数器提供中断，利用其产生的中断事件完成预定的操作。

不同型号的 PLC 主机，高速计数器的数量不同。CPU221 和 CPU222 有 4 个，它们是 HSC0 和 HSC3～HSC5；CPU224 和 CPU226 有 6 个，它们是 HSC0～HSC5。每一个高速计数器的最大计数频率取决于所使用的 CPU。

① 高速计数器的分类和工作模式。

高速计数器可分为 4 类：具有内部方向控制的单相计数、具有外部方向控制的单相计数、具有两个计数输入的两相计数和 A/B 两相正交计数。对于两相计数器，其各相计数均可运行于最大频率。在正交模式下，可选择 1×（1 倍）或 4×（4 倍）的最高计数频率。

高速计数器的工作模式分为 3 类（无复位和启动输入、有复位无启动输入、既有复位又有启动输入）12 种，各个高速计数器的工作模式及其输入端子如表 2-18～表 2-23 所示。各个高速计数器之间重复使用同一输入端子（如 I0.1、I0.4），但在同一程序中，一个输入端子只能确定为一种工作模式，不能同时分配给两种工作模式。

表 2-18 HSC0 的工作模式及输入端子

工作模式	说明	输入端子		
		I0.0	I0.1	I0.2
0	具有内部方向控制的单相增/减计数器 SM37.3=0，减计数 SM37.3=1，增计数	计数	—	—
1				复位
3	具有外部方向控制的单相增/减计数器 I0.1=0，减计数 I0.1=1，增计数	计数	方向	—
4				复位
6	具有增减计数输入的两相计数器	计数 （增）	计数 （减）	—
7				复位
9	A/B 相正交计数器 A 相超前 B 相 90°，顺时针方向旋转 B 相超前 A 相 90°，逆时针方向旋转	计数 （A 相）	计数 （B 相）	—
10				复位

表 2-19 HSC1 的工作模式及输入端子

工作模式	说明	输入端子			
		I0.6	I0.7	I1.0	I1.1
0	具有内部方向控制的单相增/减计数器	计数	—	—	—
1	SM47.3=0，减计数			复位	
2	SM47.3=1，增计数				启动
3	具有外部方向控制的单相增/减计数器	计数	方向	—	—
4	I0.7=0，减计数			复位	
5	I0.7=1，增计数				启动
6	具有增减计数输入的两相计数器	计数（增）	计数（减）	—	—
7				复位	
8					启动
9	A/B 相正交计数器	计数（A 相）	计数（B 相）	—	—
10	A 相超前 B 相 90°，顺时针方向旋转			复位	
11	B 相超前 A 相 90°，逆时针方向旋转				启动

当复位输入端上出现有效输入时，高速计数器将当前计数值清零并保持为零，直至使复位端上有效信号消失为止。当启动输入端上出现有效输入时，允许计数器计数。当启动输入无效时，计数值保持已有的值不变，并对计数脉冲信号不予理睬。当启动输入无效，同时复位输入有效时，当前值保持不变且忽略复位操作。当在复位输入有效时，若使启动输入有效，则清除当前值。

表 2-20 HSC2 的工作模式及输入端子

工作模式	说明	输入端子			
		I1.2	I1.3	I1.4	I1.5
0	具有内部方向控制的单相增/减计数器	计数	—	—	—
1	SM57.3=0，减计数			复位	
2	SM57.3=1，增计数				启动
3	具有外部方向控制的单相增/减计数器	计数	方向	—	—
4	I1.3=0，减计数			复位	
5	I1.3=1，增计数				启动
6	具有增减计数输入的两相计数器	计数（增）	计数（减）	—	—
7				复位	
8					启动
9	A/B 相正交计数器	计数（A 相）	计数（B 相）	—	—
10	A 相超前 B 相 90°，顺时针方向旋转			复位	
11	B 相超前 A 相 90°，逆时针方向旋转				启动

表 2-21 HSC3 的工作模式及输入端子

工作模式	说明	输入端子 I0.1
0	具有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	计数

表 2-22 HSC4 的工作模式及输入端子

工作模式	说明	输入端子		
		I0.3	I0.4	I0.5
0	具有内部方向控制的单相增/减计数器 SM147.3=0，减计数 SM14.3=1，增计数	计数	—	—
1				复位
3	具有外部方向控制的单相增/减计数器 I0.4=0，减计数 I0.4=1，增计数	计数	方向	—
4				复位
6	具有增减计数输入的两相计数器	计数（增）	计数（减）	—
7				复位
9	A/B 相正交计数器 A 相超前 B 相 90°，顺时针方向旋转 B 相超前 A 相 90°，逆时针方向旋转	计数（A 相）	计数（B 相）	—
10				复位

表 2-23 HSC5 的工作模式及输入端子

工作模式	说明	输入端子 I0.4
0	具有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	计数

高速计数器的工作模式必须先行定义方可使用，可通过指令 HDEF（高速计数设定指令）来完成。对于每一个高速计数器，只可使用一次 HDEF 指令。通常是利用首次扫描周期有效标志 SM 0.1 来调用一初始化子程序，在此子程序中，写入 HDEF 指令。

② 高速计数器使用的特殊标志位存储器。

每个高速计数器都有固定的特殊标志位存储器与之相配合，完成高速计数功能。具体对应关系如表 2-24 所示。

表 2-24 高速计数器使用的特殊标志位存储器

高速计数器	状态字节	控制字节	新当前值双字	新预设值双字
HSC0	SMB36	SMB37	SMD38	SMD42
HSC1	SMB46	SMB47	SMD48	SMD52
HSC2	SMB56	SMB57	SMD58	SMD62
HSC3	SMB136	SMB137	SMD138	SMD142
HSC4	SMB146	SMB147	SMD148	SMD152
HSC5	SMB156	SMB157	SMD158	SMD162

每个高速计数器都设定了一个状态字节，将当前计数方向状态、当前值等于预设值状态和当前值大于预设值状态存放在特殊标志位存储器的相应位中。状态字节中各状态位的功能如表 2-25 所示。程序运行时根据运行状况自动使某些位置位，可以通过程序来读相关位的状态，用以判断条件，实现相应的操作。

表 2-25 高速计数器的状态位

状　态　位	描　　述
SM××6.0～SM××6.4	不用
SM××6.5	当前计数方向状态位　0=减计数　1=增计数
SM××6.6	当前值等于预设值状态位　0=不等　1=相等
SM××6.7	当前值大于预设值状态位　0=小于等于　1=大于

每个高速计数器都设定了一个控制字节，通过对控制字节中指定位的编程，可以根据操作要求设置字节中各控制位，如复位与启动输入信号的有效状态、计数速率、计数方向、允许写入计数方向、允许写入预设值、允许写入当前值和允许执行高速计数指令等。控制字节中各控制位的功能如表 2-26 所示。

表 2-26　高速计数器的控制位

控　制　位	描　　述	适用的高速计数器
SM××7.0	复位有效电平控制位　0=高电平有效　1=低电平有效	HSC0，HSC1，HSC2，HSC4
SM××7.1	启动有效电平控制位　0=高电平有效　1=低电平有效	HSC1，HSC2
SM××7.2	正交计数速率选择控制位　0=4 倍计数率　1=1 倍计数率	HSC0，HSC1，HSC2，HSC4
SM××7.3	计数方向控制位　0=减计数　1=增计数	HSC0～HSC5
SM××7.4	写入计数方向允许控制位　0=不更新　1=更新	HSC0～HSC5
SM××7.5	写入预设值允许控制位　0=不更新　1=更新	HSC0～HSC5
SM××7.6	写入当前值允许控制位　0=不更新　1=更新	HSC0～HSC5
SM××7.7	高速计数指令执行允许控制位　0=禁止　1=允许	HSC0～HSC5

表 2-26 中的前 3 位（0、1 和 2 位）只有在 HDEF 指令执行时，才进行设置，在程序中其他位置不能更改（默认值为：启动和复位为高电平有效，正交计数速率为 4 倍，即 4 倍输入计数频率）。第 3 位和第 4 位可以在工作模式 0、1 和 2 下直接更改，以单独改变计数方向。后 3 位可以在任何模式下，并在程序中更改，以单独改变计数器的当前值、预设值或对高速计数器禁止计数。

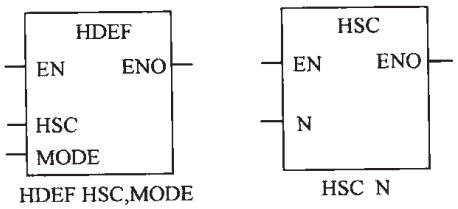
（2）高速计数器指令

高速计数器指令包括高速计数器定义指令和高速计数指令。

① 高速计数器定义指令。

高速计数器定义指令的梯形图和语句表如图 2-76(a) 所示。HSC 为高速计数器编号，字节型常量，范围是 0～5；MODE 为工作模式，字节型常量，范围是 0～11。

当 EN 输入有效时，为指定的高速计数器分配一种工作模式，即用来建立高速计数器与工作模式之间的联系。每个高速计数器使用之前必须使用该指令，而且只能使用一次。



（a）高速计数器定义指令　（b）高速计数指令

图 2-76　高速计数器指令

② 高速计数指令。

高速计数指令的梯形图和语句表如图 2-76（b）所示。N 为高速计数器编号，字节型常量，范围是 0～5。

当 EN 输入有效时，根据高速计数器特殊标志位存储器的状态，并按照高速计数器定义指令指定的工作模式，设置高速计数器并控制其工作。

（3）高速计数器的初始化

对高速计数器选择工作模式、设置控制字节、执行高速计数器定义指令、设定当前值和预设值、设置中断和执行高速计数指令等，称为高速计数器的初始化。

高速计数器的初始化，可以用主程序中的程序段来实现，但通常用子程序来实现。高速计数器在运行之前，必须要执行一次初始化程序段或初始化子程序。

高速计数器的初始化分为以下几个步骤：

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；
② 在初始化子程序中，按控制要求对高速计数器的控制字节赋值；
③ 执行高速计数器定义指令；
④ 将新当前值装入新当前值双字中；
⑤ 将新预设值装入新预设值双字中；
⑥ 设置中断事件；
⑦ 执行全局中断允许指令；
⑧ 执行高速计数指令；
⑨ 退出子程序。

实例 32：高速计数器指令的应用

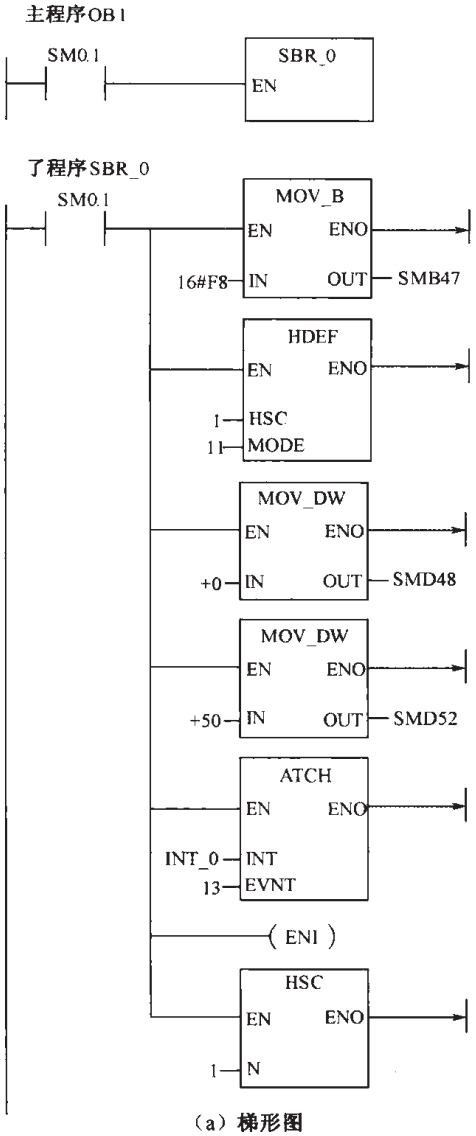
实例说明

在利用 PLC 进行一些运动控制或者需要统计一些高速脉冲信号的时候，常常用到高速计数器来进行计数，高速计数器能准确的计数，而不受 PLC 的扫描周期的影响。

实例实现

高速计数器指令的应用程序如图 2-77 所示，其对应的语句为：

```
//主程序 OB1
LD          SM0.1
CALL        SBR_0
//子程序 SBR_0
LD          SM0.1
MOVB        16#F8, SMB47
HDEF        1,11
MOVD        +0, SMD48
MOVD        +50,SMD52
ATCH        INT_0, 13
ENI
HIS         1
```



（a）梯形图

图 2-77　高速计数器指令的应用程序

实例分析

本实例中，在首次扫描时，调用初始化子程序 SBR_0。在子程序 SBR_0 中，将 16#F8 赋予 SMB47，配置 HSC1 工作模式 （启动和复位输入高电平有效、4 倍计数率的正交模式、计数方向为增计数、允许更新计数方向、允许更新预设值、允许更新当前值、允许执行高速计数指令），清除 HSC1 的初始值，置 HSC1 的预设值为 50，当 HSC1 的当前值=预设值时，连接中断服务程序 INT_0 到事件 13，全局中断允许，对 HSC1 执行高速计数。

2. 高速脉冲输出及其指令

高速脉冲输出功能是指在 PLC 的某些输出端产生高速脉冲，用来驱动负载，实现高速输出和精确控制，它在步进电动机控制中具有广泛应用。使用高速脉冲输出功能时，PLC 主机应选用晶体管输出型而不能采用继电器输出型，以满足高速输出的频率、快速响应的要求。

（1）高速脉冲输出

高速脉冲输出有高速脉冲串输出 PTO 和宽度可调脉冲输出 PWM 两种形式。高速脉冲输

出 PTO 主要用来输出指定数量的方波（占空比 50%），用户可以控制方波的周期和脉冲数；宽度可调脉冲输出 PWM 主要用来输出占空比可调的高速脉冲串，用户可以控制脉冲的周期和脉冲宽度。

每种 PLC 主机最多提供两个 PTO/PWM 发生器产生高速脉冲串或脉冲宽度可调的波形，一个发生器分配在输出端 Q0.0，另一个分配在输出端 Q0.1。高速脉冲输出端子与输出映像寄存器共用 Q0.0 和 Q0.1，但同一个输出端子只能用做一种功能。如果 Q0.0 和 Q0.1 在程序执行时用做高速脉冲输出，则只能做高速脉冲输出使用，其通用功能被自动禁止，任何输出刷新、输出置位、立即输出等指令都无效，只有高速脉冲输出不用的输出端子才可能做普通数字量输出端子使用。Q0.0 和 Q0.1 编程用做高速脉冲输出，但未执行高速脉冲输出指令时，可以用普通位操作指令设置这两个输出位，以控制高速脉冲的起始和终止电平。

① 高速脉冲输出使用的特殊标志位存储器。

每个高速脉冲输出对应一定数量的特殊标志位存储器。这些存储器包括控制字节存储器、状态字节存储器和参数数值存储器。它们用以控制高速脉冲的输出形式，反映输出状态和参数值。特殊标志位存储器的分配如表 2-27 所示。

每个高速脉冲输出都设定了一个状态字节，程序运行时，根据运行状态使某些位自动置位。可以通过程序来读相关位的状态，用此状态作为判断条件实现相应的操作。状态字节中各状态位的功能如表 2-28 所示。

表 2-27 高速脉冲输出使用的特殊标志位存储器

Q0.0 的存储器	Q0.1 的存储器	名称及描述
SMB66	SMB76	状态字节，在 PTO 方式下，跟踪脉冲串的输出状态
SMB67	SMB77	控制字节，控制 PTO/PWM 脉冲输出的基本功能
SMW68	SMW78	周期值，字型，PTO/PWM 的周期值，范围 10～65535μs 或 2～65535 ms
SMW70	SMW80	脉宽值，字型，PWM 的脉宽值，范围 0～65535μs 或 0～65535 ms
SMD72	SMD82	脉冲数，双字型，PTO 的脉冲数，范围 1～4294967295
SMB166	SMB176	段数，多段管线 PTO 进行中的段数
SMW168	SMW178	起始地址，多段管线 PTO 包络表起始字节相对变量存储器 V0 的偏移量

表 2-28 高速脉冲输出的状态位

Q0.0 的状态位	Q0.1 的状态位	描　述		
SM66.0～SM66.3	SM76.0～SM76.3	不用		
SM66.4	SM76.4	PTO 包络因增量计算错误终止	0=无错误	1=终止
SM66.5	SM76.5	PTO 包络因用户命令终止	0=无错误	1=终止
SM66.6	SM76.6	PTO 管线溢出	0=无溢出	1=溢出
SM66.7	SM76.7	PTO 空闲	0=执行中	1=空闲

每个高速脉冲输出都对应一个控制字节，通过对控制字节指定位的编程，根据操作要求设置字节中各控制位，如脉冲输出允许、PTO/PWM 模式选择、单段/多段选择、更新方式、时间基准和允许更新等。控制字节中各控制位的功能如表 2-29 所示。

表 2-29　高速脉冲输出的控制位

Q0.0 的控制位	Q0.1 的控制位	描　　述
SM67.0	SM77.0	PTO/PWM 更新周期值允许　0=不更新　1=允许更新
SM67.1	SM77.1	PWM 更新脉冲宽度值允许　0=不更新　1=允许更新
SM67.2	SM77.2	PTO 更新输出脉冲数允许　0=不更新　1=允许更新
SM67.3	SM77.3	PTO/PWM 时间基准选择　0=μs 单位时基　1= ms 单位时基
SM67.4	SM77.4	PWM 更新方式　0=异步更新　1=同步更新
SM67.5	SM77.5	PTO 单段/多段方式　0=单段管线　1=多段管线
SM67.6	SM77.6	PTO/PWM 模式选择　0=选用 PTO 模式　1=选用 PWM 模式
SM67.7	SM77.7	PTO/PWM 脉冲输出允许　0=禁止　1=允许

② 高速脉冲串输出 PTO。

高速脉冲串输出 PTO 主要是用来输出指定数量的方波（占空比 50%），用户可以控制方波的周期和脉冲数。状态字节中的最高位用来指示脉冲串输出是否完成。脉冲串输出完成同时可以产生中断，因而可以调用中断程序完成指定操作。

高速脉冲串输出 PTO 的周期单位可以是μs 或 ms，为 16 位无符号数据，周期变化范围是 10～65535 μs 或 2～65535 ms。通常设定周期值为偶数，若设置为奇数，则会引起输出波形占空比的轻微失真。如果编程时设定周期单位小于最小值，系统默认则按最小值进行设置。

高速脉冲串输出 PTO 的脉冲数用双字无符号数表示，取值范围是 1～4 294 967 295。如果编程时指定脉冲数为 0，则系统默认脉冲数为 1 个。

高速脉冲串输出 PTO 按指定的脉冲数和脉冲周期来控制脉冲串。如果要输出多个脉冲串，则允许脉冲串排队，以形成管线。当前输出的脉冲串完成之后，立即输出新脉冲串，这保证了脉冲串顺序输出的连续性。根据管线的实现方式，将 PTO 分为两种：单段管线和多段管线。

单段管线中只能存放一个脉冲串的控制参数（即入口），一旦启动了一个脉冲串进行输出，就需要用指令立即为下一个脉冲串更新特殊标志位寄存器，并再次执行脉冲串输出指令。当前脉冲串输出完成之后，自动输出下一个脉冲串。重复这一操作可以实现多个脉冲串的输出。单段管线中的各脉冲串可以采用不同的时间基准。单段管线输出多个高速脉冲时，编程复杂，而且有时参数设置不当会造成脉冲串之间的不平滑转换。

多段管线在变量存储器区 V 建立一个包络表。包络表中存储各个脉冲串的参数，相当于有多个脉冲串的入口。多段管线可以用高速脉冲输出指令 PLS 启动，运行时，自动从包络表中按顺序读出每个脉冲串的参数进行输出。编程时必须装入包络表起始变量（V 存储器区）的偏移地址，运行时只使用特殊存储器的控制字节和状态字节。

多段管线的包络表格式由包络段数和各段构成。每段长度为 8 个字节，包括脉冲周期值 16 位、周期增量值 16 位和脉冲计数值 32 位。包络表的格式如表 2-30 所示。

表 2-30　多段管线包络表的格式

从包络表开始的字节偏移量	名　　称	描　　述
0	段数	段数范围为 1～255。数 0 将产生非致命性错误，不产生 PTO 输出
1	段 1	初始周期，取值范围为 2～65 535
3		每个脉冲的周期增量，有符号整数，取值范围为–32 768～+32 767
5		输出脉冲数，取值范围为 1～4 294 967 295

续表

从包络表开始的字节偏移量	名　称	描　述
9	段 2	初始周期，取值范围为 2～65 535
11		每个脉冲的周期增量，有符号整数，取值范围为–32 768～+32 767
13		输出脉冲数，取值范围为 1～4 294 967 295
17	段 3	初始周期，取值范围为 2～65 535
19		每个脉冲的周期增量，有符号整数，取值范围为–32 768～+32 767
21		输出脉冲数，取值范围为 1～4 294 967 295
:	:	:
:	:	:

多段管线编程非常简单，而且具有按照周期增量存储器的数值自动增减周期的能力，这在步进电动机的加速和减速控制时非常方便。多段管线使用时的局限性是，在包络表中所有脉冲串的周期必须采用同一个基准，而且当多段管线执行时，包络表的各段参数不能改变。

③ 宽度可调脉冲输出 PWM。

宽度可调脉冲输出 PWM 主要是用来输出占空比可调的高速脉冲串，用户可以控制脉冲的周期和脉冲宽度。

宽度可调脉冲输出 PWM 的周期单位、脉冲数的规定与高速脉冲串输出 PTO 相同。当设定脉宽等于周期时（使占空比为 100%），输出连续接通；设定脉宽等于 0 时（使占空比为 0%），输出断开。

改变 PWM 波形特性的方法有两种：同步更新和异步更新。不改变脉冲的时基基准，只改变脉冲的宽度，称为同步更新；既改变脉冲的时基基准，又改变脉冲的宽度，称为异步更新。

同步更新使新、旧两个脉冲开始的上升沿和结束的下降沿保持同步，波形特性的变化发生在周期边沿，两个脉冲能平滑转换。异步更新使新旧两个脉冲波形不同步，会造成 PTO /PWM 功能被瞬时禁止，这会引起被控设备的振动。

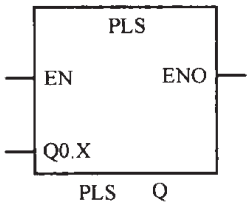


图 2-78　高速脉冲输出指令

（2）高速脉冲输出指令

高速脉冲输出指令的梯形图和语句表如图 2-78 所示。数据输入 Q 端取值范围必须是 0 或 1。

当 EN 输入有效时，首先检测用程序设置的特殊标志位存储器位，激活由控制位定义的脉冲操作，然后从 Q0.0 或 Q0.1 输出高速脉冲。高速脉冲串输出 PTO 和宽度可调脉冲输出 PWM 都由 PLS 指令激活输出。

（3）高速脉冲输出的初始化

对高速脉冲输出选择工作模式，设置控制字节，设定周期值、周期增量值和脉冲数，设置中断和执行高速脉冲输出指令等，称为高速脉冲输出的初始化。

高速脉冲输出的初始化，可以用主程序中的程序段来实现，但通常用子程序来实现。高速脉冲输出在运行之前，必须要执行一次初始化程序段或初始化子程序。

高速脉冲输出的初始化分为以下几个步骤：

① 用初次扫描存储器位 SM0.1=1，复位 Q0.0 或 Q0.1 为 0，并调用执行初始化操作的子程序。

② 在初始化子程序中，按控制要求对高速脉冲输出的控制字节赋值。

③ 将周期值、脉宽值、脉冲数等装入相应的特殊标志位存储器中或设定包络表。

④ 可选步骤：可以输出一个脉冲串，立即对一个相关功能进行编程；也可以使用脉冲串输出完成中断事件（事件号 19）来连接一个中断子程序，并执行全局中断允许指令。

⑤ 执行高速脉冲输出指令。

⑥ 退出子程序。

实例 33：高速脉冲输出指令的应用

实例说明

在运动控制中，需要给步进电动机输出一些脉冲信号，从而控制电动机的运行，为了准确的输出脉冲信号，往往采用高数脉冲输出指令。

实例实现

高速脉冲输出指令的应用程序如图 2-79 所示，其对应的语句为：

```
//主程序 OB1
LD          SM0.1
R           Q0.0 , 1
CALL        SBR_0
//子程序 SBR_0
LD          SM0.0
MOVB        16#8D, SMB67
MOVW        +500,SMW68
MOVD        +4,SMD72
ATCH        INT_0, 19
ENI
PLS         0
```

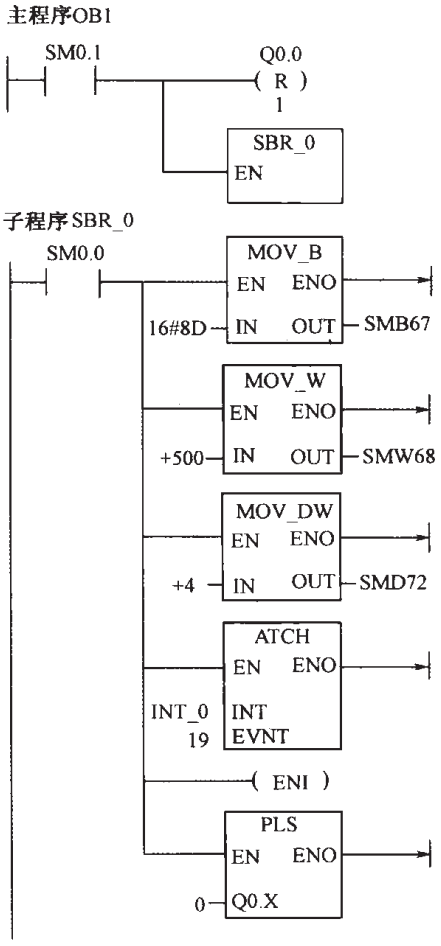


图 2-79　高速脉冲输出指令的应用程序

实例分析

该程序是单段管线高速脉冲串输出 PTO。首次扫描时，将 Q0.0 复位为 0，并调用子程序 SBR_0。在子程序中，设置控制字节 SMB67= 16#8D（不更新周期值、不更新脉冲宽度、允许更新输出脉冲数、周期单位是 ms、选择单段管线 PTO 模式、允许 PTO 脉冲输出），PTO 脉冲周期为 500 ms，脉冲数目为 4 个，使用脉冲串输出完成中断事件（事件号 19）来连接一个中断子程序 INT_0，允许全局中断，执行 PTO 脉冲输出。

思考题

1．已知某控制程序的语句的形式，请将其转换为梯形图的形式。

```
LD    I0.0
AN    T37
TON   T37,1000
LD    T37
LD    Q0.0
CTU   C10,360
LD    C10
O     Q0.0
=     Q0.0
```

2．写出如图 2-80 所示梯形图的语句程序。

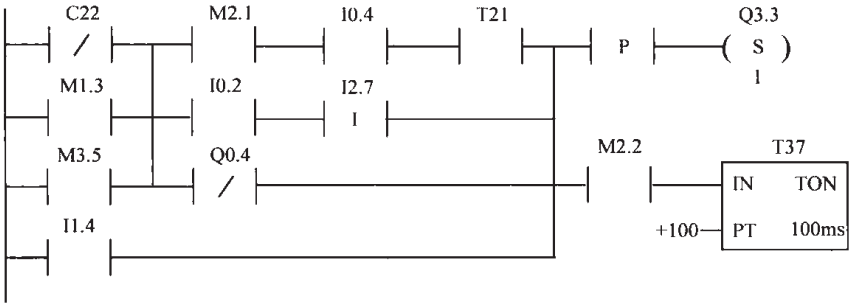
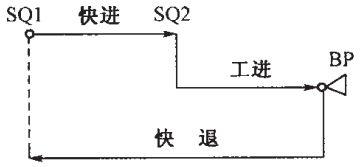


图 2-80　习题 2

3．有电动机 3 台，希望能够同时启动同时停车。设 Q0.0、Q0.1、Q0.2 分别驱动电动机的接触器。I0.0 为启动按钮，I0.1 为停车按钮，试编写程序。

4．组合机床的工作循环图及元件动作表如图 2-81 所示，试用置位复位指令编写程序。



（a）工作循环图

	YV1	YV2	YV3
原位	−	−	−
快进	+	−	−
工进	−	−	+
快退	−	+	−

（b）元件动作表

图 2-81　题 4

5．已知给出如图 2-82 所示的控制程序梯形图的形式，请将其转换为语句的形式。

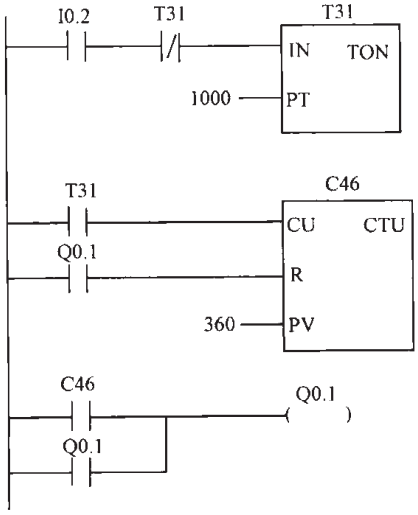


图 2-82　题 5

6．已知输入触点时序图如图 2-83 所示，结合程序画出 Q0.0 和 Q0.1 的时序图。

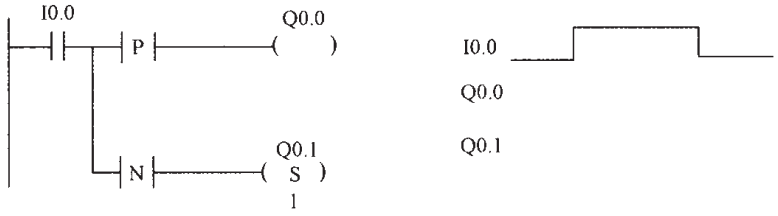


图 2-83　题 6

7．电动机星形－三角形降压启动，Q0.0 为电源接触器，Q0.1 为星接输出线圈，Q0.2 为三角接输出线圈，I0.1 为启动按钮，I0.0 为停止按钮，星形－三角形切换延时时间为 5s。试编写程序。

8．采用一只按钮每隔 3s 顺序启动 3 台电动机，试编写程序。

9．某组合机床的工作循环图及元件动作表如表 2-31 所示。现用顺序控制继电器指令编写控制程序。

表 2-31　题 9

输　入		
元件名称	元件符号	端子号
启动按钮	SB1	I0.0
原位行程开关	SQ1	I0.1
快进转工进行程开关	SQ2	I0.2
压力继电器	BP	I0.3
输　出		
元件名称	元件符号	端子号
进给电磁阀	YV1	Q0.0
退回电磁阀	YV2	Q0.1
工进电磁阀	YV3	Q0.2

10．将下面的梯形图程序转换成语句指令形式。

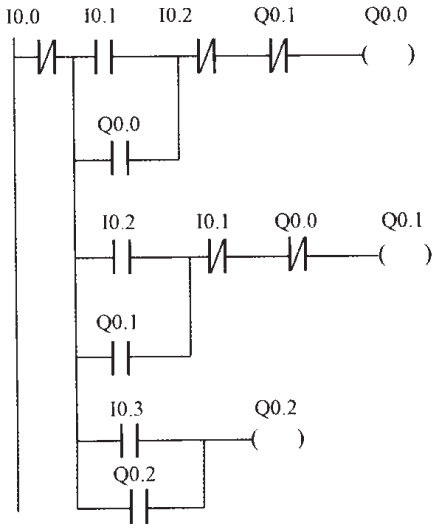


图 2-84　题 10

第3章 PLC系统的基本控制编程

- PLC程序的结构与编程规则
- 基本控制程序
- 常用典型环节或系统控制编程

程序是运用相应的指令和数据，遵循一定的规律，编制成具有一定控制功能的信息语言。PLC 程序分为系统程序和应用程序（用户程序）。系统程序是 PLC 生产厂家编制的程序，存放在系统内存中，用于系统控制。用户程序是 PLC 用户根据受控对象的生产过程和工艺要求，为解决实际应用问题而编制的程序。下面介绍 PLC 用户程序的结构、编程规则、基本控制程序及典型环节与系统。

3.1 PLC 程序的结构与编程规则

3.1.1 PLC 程序的结构

1. 用户程序的分类

S7-200 的控制程序分为主程序（OB1）、子程序（SBR0～SBR63）和中断程序（INT0～INT127）3 种。

主程序是用户程序的主体，在一个项目中只能有一个主程序，CPU 在每个扫描周期都要执行一次主程序指令。

子程序是程序的可选部分，最多可以有 64 个，合理使用子程序，可以优化程序结构，减少扫描时间。子程序一般在主程序中被调用，也可以在子程序或中断程序中被调用。只有被调用的子程序，才能够执行。

中断程序也是程序的可选部分，是用来及时处理与用户程序的执行时序无关的操作，或者不能事先预测何时发生的中断事件，最多可以有 128 个。它的调用由各种中断事件触发，而不是由用户程序调用，中断事件一般有输入中断、定时中断、高速计数器中断和通信中断等。可在其他程序中使用的寄存器也不允许被中断程序改写。

2. S7-200 的程序结构

S7-200 PLC 的用户程序结构可分为两种：线性程序结构和分块程序结构。

（1）线性程序结构

线性程序结构是指一个工程的全部控制任务被分成若干个小的程序段，按照控制的顺序依次排放在主程序中，如图 3-1 所示。编程时，用程序控制指令将各个小的程序段依次链接起来；程序执行过程中，CPU 不断扫描主程序，按照编写好的指令代码顺序地执行控制工作。

线性程序结构简单明了，但是仅适合控制量比较小的场合。控制任务越大，线性程序的结构就越复杂，CPU 执行效率就越低，系统越不稳定。

（2）分块程序结构

分块程序结构是指一个工程的全部控制任务被分成多个任务模块，每个模块的控制任务由子程序或中断程序完成。编程时，主程序和子程序（或中断程序）分开独立编写；在程序执行过程中，CPU 不断扫描主程序，碰到子程序调用指令就转移到相应的子程序中去执行，如图 3-2 所示，遇到中断请求就调用相应的中断程序。

分块程序结构虽然复杂一点，但是可以把一个复杂的控制任务分解成多个简单的控制任务。分块程序有利于代码编写，而且程序调试也比较简单。所以，对于一些相对复杂的工程控制，建议使用分块程序结构。

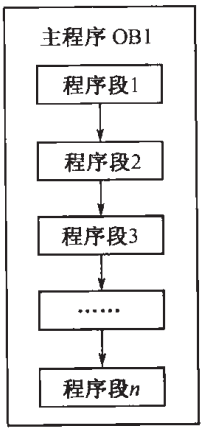


图 3-1　线性程序结构

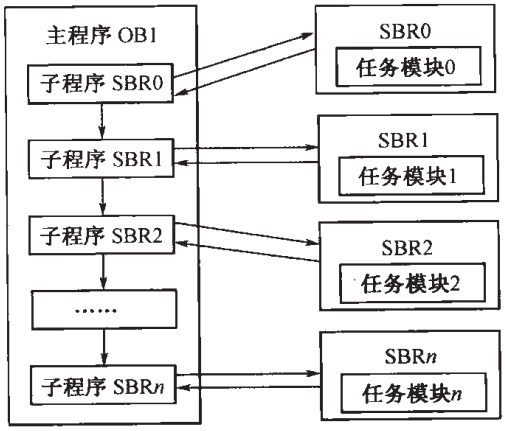


图 3-2　分块程序结构

3.1.2　编程技巧与规则

可编程控制器是按照逐行扫描执行的，因此在编制梯形图程序时，元器件或触点排列顺序对程序执行可能会带来很大影响，有时甚至使程序无法运行。为了使程序简短、清晰、执行速度快，节省用户程序区域，常需要对梯形图程序加以变换和化简。下面举例说明。

1．串并联梯形图程序编程规则

较复杂的串并联梯形图程序，为了简化程序，减少指令，有效地节约一些用户程序区域，一般遵循以下两个原则。

（1）在并联电路中，串联触点较多的电路编在梯形图上方，以减少指令数。

如图 3-3 所示的两个梯形图实现的逻辑功能一样，图 3-4（a）中串联触点较多的电路编在梯形图下方，增加了指令条数，多占用用户程序区域，不够合理，变换为图 3-4（b）所示的梯形图较合理。

图 3-3（a）的语句：

```
LD    I0.0
LD    I0.1
AN    I0.2
OLD
=     Q0.0
```

图 3-3（b）的语句：

```
LD    I0.1
AN    I0.2
O     I0.0
=     Q0.0
```

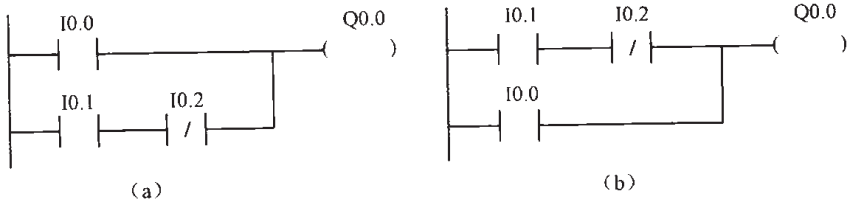


图 3-3　并联电路梯形图简化

（2）在串联电路中，并联触点较多的电路编在梯形图左边，可减少指令数。

如图 3-4 所示的两个梯形图实现的逻辑功能一样，但程序繁简程度却不同。图 3-4（a）和图 3-4（b）的不同在于：将串联的两部分电路左、右对换后，并联的两个分支上、下对换。变换后，原有的逻辑关系不变，但程序却简化了，指令数也减少了，因此，图 3-4（b）较合理。

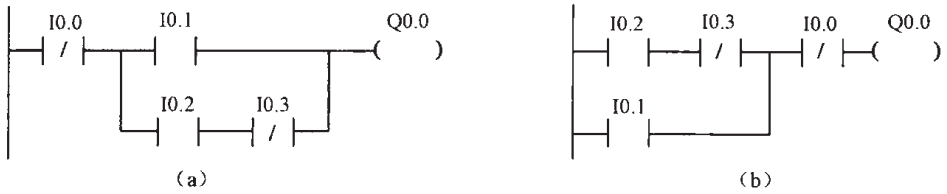


图 3-4　串联电路梯形图简化

图 3-4（a）的语句：

```
LDN   I0.0
LD    I0.1
LD    I0.2
AN    I0.3
OLD
ALD
=     Q0.0
```

图 3-4（b）的语句：

```
LD    I0.2
AN    I0.3
O     I0.1
AN    I0.0
=     Q0.0
```

经验证明，梯形图设计或变换可遵循“左沉右轻”，“上沉下轻”原则。

2．应使梯形图的逻辑关系尽量清楚，便于阅读检查和输入程序

如图 3-5（a）所示梯形图结构比较复杂，逻辑关系不够清楚，用 OLD、ALD 指令编程不便区分逻辑关系，可以重复使用一些触点画出它的等效电路梯形图，如图 3-5（b）所示，程序指令条数虽然增多，但逻辑关系清楚，便于阅读和编程。

图 3-5（a）的语句：

```
LD    I0.0        AN    I0.7
LDN   I0.1        OLD
A     I0.2        ALD
LD    I0.3        OLD
AN    I0.4        ALD
LD    I0.5        =     Q0.0
LD    I0.6
```

图 3-5（b）的语句：

```
LD    I0.0        LD    I0.0
AN    I0.1        A     I0.3
A     I0.2        AN    I0.4
LD    I0.0        A     I0.6
A     I0.3        AN    I0.7
AN    I0.4        OLD
A     I0.5        =     Q0.0
OLD
```

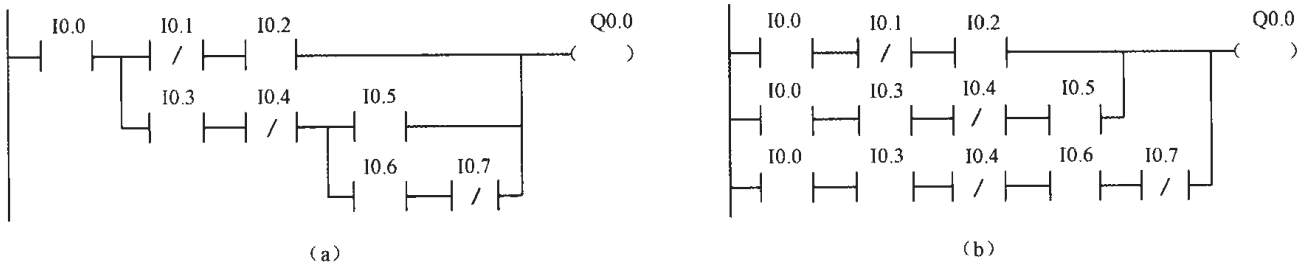


图 3-5　复杂电路梯形图处理

3．应避免出现无法编程的梯形图

如图 3-6（a）所示的桥式电路无法编程，它不符合梯形图执行的原则，需改画成如图 3-6（b）所示的 PLC 图形式。

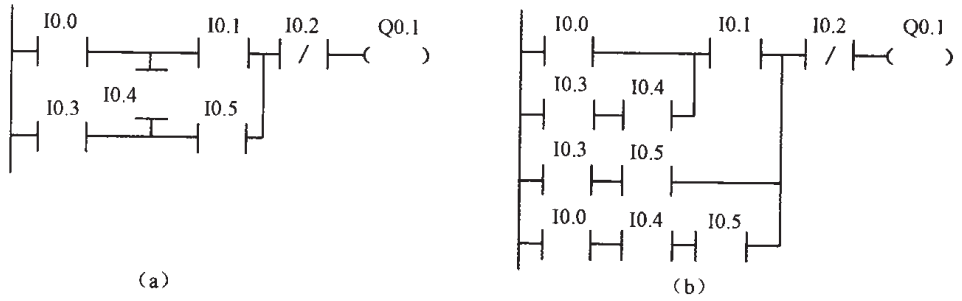


图 3-6　避免无法编程的梯形图

3.2　基本控制程序

3.2.1　自锁、互锁控制

自锁、互锁控制是梯形图控制程序中最基本的环节。常用于对输入开关和输出映像寄存器的应用编程控制。下面通过 3 个实例进行介绍。

实例 34：自锁控制

实例说明

自锁控制是 PLC 控制程序中常用的控制程序形式，也是常说的启停控制。

实例实现

具体实例可参见第 1 章实例 2 中图 1-10 所示的梯形图控制程序，在该程序中 I0.0 常开触点闭合、Q0.0 得电，此时 Q0.0 常开触点闭合，保证 Q0.0 持续得电，只有当常闭触点 I0.1 断开时，Q0.0 才断电。

实例分析

程序中，常闭触点 I0.0 为启动开关；常闭触点 I0.1 为停止开关；Q0.0 触点为自锁触点。这种自锁控制常用于以无锁定开关作启动开关，或者用只接通一个扫描周期的触点去启动一个持续动作的控制电路。

实例 35：互锁控制

实例说明

互锁控制就是在两个或两个以上输出映像寄存器网络中，只能保证其中一个输出映像寄存器接通输出，而不能让两个或两个以上输出映像寄存器同时输出，避免了两个或两个以上输出映像寄存器不能同时动作的控制对象同时动作。

实例实现

在如图 3-7 所示的 Q0.0 和 Q0.1 网络程序段中，当 I0.1 得电闭合，Q0.0 输出；由于 Q0.0

的常闭触点接在 Q0.1 网络中，即使当 I0.2 得电闭合，Q0.1 也不输出，只有当 I0.0 断开，Q0.0 断电后，当 I0.2 得电闭合，Q0.1 才输出，由于 Q0.1 的常闭触点接在 Q0.0 网络中，此时当 I0.1 得电闭合，Q0.0 也不输出。本例梯形图对应的语句如下：

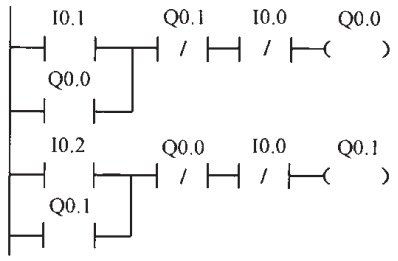


图 3-7　互锁控制程序

```
LD    I0.1
O     Q0.0
AN    Q0.1
AN    I0.0
=     Q0.0
LD    I0.2
O     Q0.1
AN    Q0.0
AN    I0.0
=     Q0.1
```

实例分析

在图 3-7 所示的程序中，Q0.0 和 Q0.1 的常闭触点分别接在对方网络中，只要一个网络输出映像寄存器先接通（如 Q0.0），而 Q0.0 的常闭触点断开，另一个网络中输出映像寄存器（如 Q0.1）才不能再接通，从而保证任何时候两者都不能同时启动，这种控制称为互锁控制，常闭触点 Q0.0 和 Q0.1 为互锁触点。这种互锁控制常用于被控的是一组不允许同时动作的对象，如电动机正、反转控制等。

实例 36：连锁控制

实例说明

在工程应用中，有些控制对象动作是在另一个控制对象动作的前提下才能动作，称之为连锁控制。下面以机床主轴电动机与润滑电动机启动为例，介绍连锁系统。

实例实现

在如图 3-8 所示梯形图程序中，输出映像寄存器 Q0.0 对应润滑电动机启动输出，Q0.1 为主轴电动机启动输出，由于有常开触点 Q0.0 接在 Q0.1 网络中，而在 Q0.0 网络中没有常开触点 Q0.1 接入，当 I0.1 得电闭合，Q0.0 得电输出，润滑电动机启动，其常开触点 Q0.0 闭合，如果这时当 I0.2 得电闭合，则 Q0.1 得电输出，主轴电动机启动。如果润滑电动机未启动，即使当 I0.2 得电闭合，由于其常开触点 Q0.0 未闭合，Q0.1 不得电，主轴电动机无法启动。本例梯形图对应的语句如下：

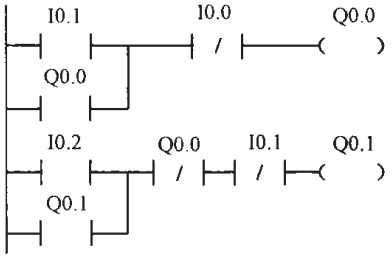


图 3-8　连锁控制程序

```
LD    I0.1
O     Q0.0
AN    I0.0
=     Q0.0
```

```
LD      I0.2
O       Q0.1
A       Q0.0
AN      I0.1
=       Q0.1
```

实例分析

在如图 3-8 所示程序中，在输出映像寄存器 Q0.0 和 Q0.1 网络中，由于有常开触点 Q0.0 接在 Q0.1 网络中，而在 Q0.0 网络中没有常开触点 Q0.1 接入，只有当 Q0.0 接通时，Q0.1 才有可能接通，只要 Q0.0 断开，Q0.1 就不可能接通，也就是说一方的动作是以另一方的动作为前提的，这种控制称为连锁控制。这种连锁控制常用于被控的是一组有连锁要求的动作对象，譬如在机床启动时，一般先启动润滑系统，后启动主轴系统，可以避免主轴未润滑时启动，减轻主轴磨损。

3.2.2　时间控制

在 PLC 控制系统中，时间控制用得非常多，其中大部分用于延时和定时控制。在 S7-200 型可编程控制器内部有 3 种类型的定时器和 3 个等级分辨率（1 ms、10 ms 和 100 ms）可以用于时间控制，用户在编程时会感到十分方便。下面通过 5 个实例介绍时间控制。

实例 37：瞬时接通/延时断开控制

实例说明

瞬时接通/延时断开控制要求在输入信号有效时，马上有输出，而输入信号无效后，输出信号延时一段时间才停止。

实例实现

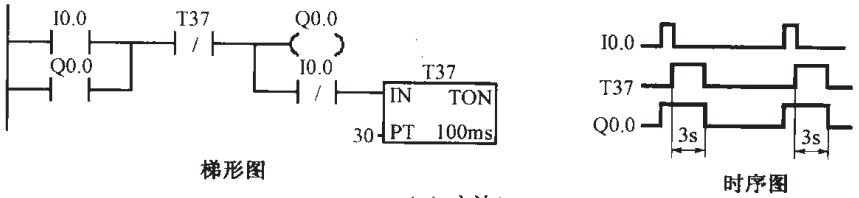
图 3-9 分别是瞬时接通/延时断开控制的梯形图和时序图。在图 3-9（a）所示的梯形图中，当 I0.0=ON 时，输出 Q0.0=ON 并自锁，当 I0.0=OFF 后，定时器 T37 工作，定时 3 s 后，定时器常闭触点断开，使输出 Q0.0 断开。在图 3-9（b）所示的梯形图中，当 I0.0 瞬间接通后断开，则 Q0.0=ON 且自锁，定时器 T37 工作 3 s 后，定时器触点闭合，使输出 Q0.0 断开。图 3-9 梯形图对应的语句分别如下。

图 3-9（a）的语句：

```
LD      I0.0
O       Q0.0
AN      T37
=       Q0.0
AN      I0.0
TON     T37, 30
```

图 3-9（b）的语句：

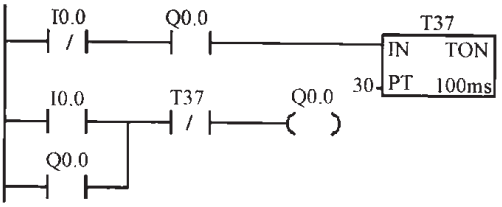
```
LDN     I0.0
A       Q0.0
TON     T37, 30
LD      I0.0
O       Q0.0
AN      T37
=       Q0.0
```



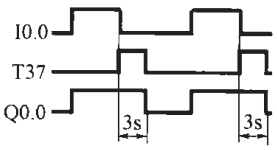
梯形图

时序图

（a）方法1



梯形图



时序图

（b）方法2

图 3-9　瞬时接通/延时断开

实例分析

在图 3-9 所示梯形图程序中，定时器工作因为 I0.0 变为 OFF 后，Q0.0 仍要保持得电状态 3 s，所以 Q0.0 的自锁触点是必需的。图 3-9（a）和（b）的工作原理相同，只是梯形图结构不同。瞬时接通/延时断开控制的梯形图用断开延时定时器可以使程序更简单。读者可以自行编制梯形图程序。

实例 38：延时接通/延时断开控制

实例说明

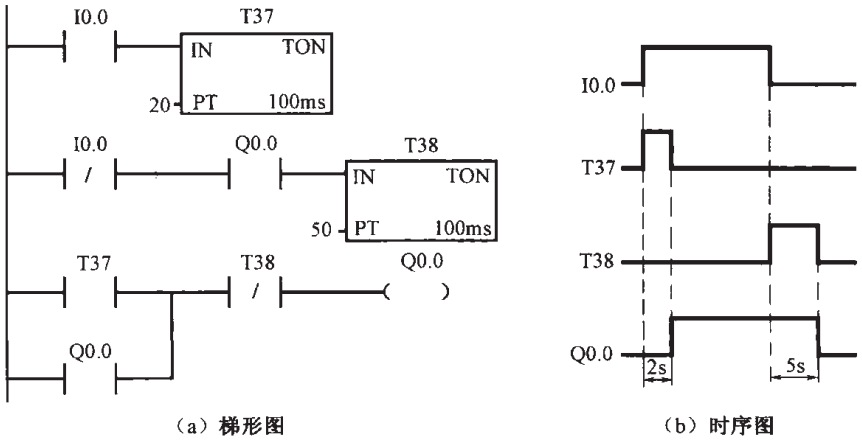
延时接通/延时断开控制要求输入信号 ON 后，停一段时间后输出信号才 ON；输入信号 OFF 后，输出信号延时一段时间才 OFF。与瞬时接通/延时断开控制相比，该控制电路多加了一个输入延时。

实例实现

图 3-10 是延时接通/延时断开控制的梯形图和时序图，图中 T37 延时 2 s 作为 Q0.0 的启动条件，T38 延时 5 s 作为 Q0.0 的断开条件，两个定时器配合使用实现 Q0.0 的输出。本例梯形图对应的语句如下：

```
LD      I0.0
TON     T37, 20
```

```
LDN     I0.0
A       Q0.0
TON     T38, 50
LD      T37
O       Q0.0
AN      T38
=       Q0.0
```



（a）梯形图　　（b）时序图

图 3-10　延时接通/延时断开

实例分析

图 3-8 梯形图程序中，使用 T37 和 T38 两个定时器，配合实现控制电路的功能，可以通过调整 T37 和 T38 的设定时间，得到需要的延时时间。延时接通/延时断开控制的梯形图用接通延时定时器和断开延时定时器可以使程序更简单。读者可以自行编制梯形图程序。

实例 39：多个定时器组合实现长延时控制

实例说明

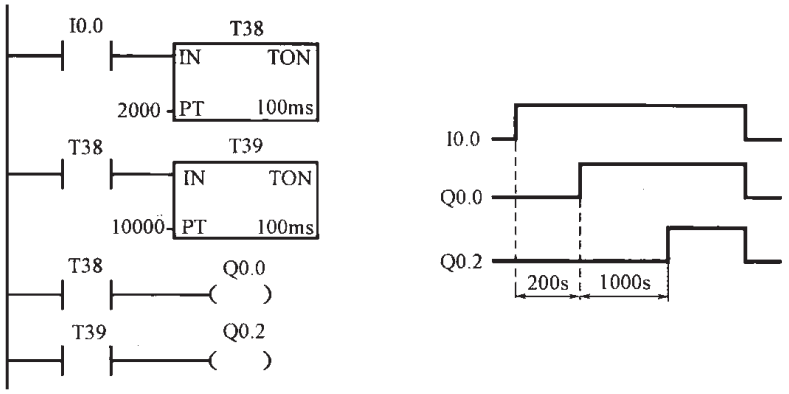
有些控制场合延时时间长，超出了定时器的定时范围，称为长延时，长延时电路可以以小时（h）、分钟（min）作为单位来设定。长延时控制可以使用多个定时器组合方式实现，也可以采用定时器和计数器组合方式实现，使用计数器组合也可以实现时钟控制。

实例实现

在图 3-11 所示的长延时控制程序中，Q0.0 的接通是由定时器 T38 实现的，Q0.2 的接通是由定时器 T38 与 T39 共同定时实现的，这就是多个定时器组合实现长延时的情况。本例梯形图对应的语句如下：

```
LD      I0.0
TON     T38, 2000
LD      T38
```

```
TON     T39, 10000
LD      T38
=       Q0.0
LD      T39
=       Q0.2
```



（a）梯形图　　（b）时序图

图 3-11　定时器串联实现长延时控制

实例分析

本例用多个定时器组合方式实现长延时控制，在图 3-11 所示的程序中，当输入 I0.0 端接通，T38 开始计时，经过 200 s 后，其常开触点 T38 闭合，Q0.0 接通，同时启动 T39 开始计时，经过 1000 s 后，Q0.2 接通。由此可见，T38 和 T39 共同延时 200 s+1000 s =1200 s 后 Q0.2 接通。

实例 40：定时器和计数器组合实现长延时控制

实例说明

本实例用定时器和计数器组合方式实现长延时情况。

实例实现

在图 3-12 所示的定时器和计数器组合实现长延时控制程序中，当输入 I0.0 端接通，T33 开始计时，经过 1s 后，其常开触点 T33 闭合，计数器 C0 开始递增计数，与此同时 T33 的常闭触点打开，T33 断电，常开触点 T33 打开，计数器 C0 仅计数一次，而后 T33 开始重新计时，如此循环……当 C0 计数器经过 1s×20=20 s 后，计数器 C0 有输出，其常开触点 C0 闭合，输出 Q0.0 接通。显然，输入 I0.0 端接通后，延时 1×20 s 后输出 Q0.0 接通。本例梯形图对应的语句如下：

```
LD      I0.0
AN      T33
TON     T33, 100
LD      T33
LDN     I0.0
CTU     C0, 20
```

```
LD      C0
=       Q0.0
```

实例分析

图 3-12 所示程序，是 T33 定时器和 C0 计数器组合实现长延时的典型情况，T33 定时器启动 C0 计数器计数，反复循环进行，到 C0 计数 20 次后，由于 C0 常开触点闭合，输出 Q0.0 接通实现长延时控制。

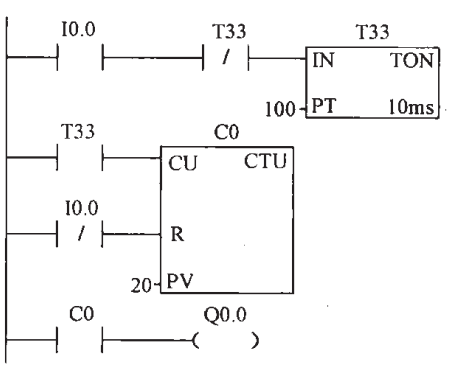


图 3-12　定时器和计数器组合实现长延时控制

实例 41：计数器串联组合实现时钟控制

实例说明

本实例是使用计数器组合实现时钟控制的例子。

实例实现

如图 3-13 所示是高精度时钟控制程序，秒脉冲特殊存储器 SM0.5 作为秒发生器，用于计数器 C51 的计数脉冲信号，当计数器 C51 的计数累计值达到设定值 60 次时（即为 1 min 时）计数器位置“1”，即 C51 的常开触点闭合，该信号将作为计数器 C52 的计数脉冲信号；计数器 C51 的另一常开触点使计数器 C51 复位（称为自复位式）后，使计数器 C51 从 0 开始重新计数。相似地，计数器 C52 计数到 60 次时（即为 1 小时时）其两个常开触点闭合，一个作为计数器 C53 的计数脉冲信号，另一个使计数器 C52 自复位，又重新开始计数；计数器 C53 计数到 24 次时（即为 1 天），其常开触点闭合，使计数器 C53 自复位，又重新开始计数，从而实现时钟功能。输入信号 I0.1、I0.2 用于建立期望的时钟设置，即调整分针、时针。本例梯形图对应的语句如下：

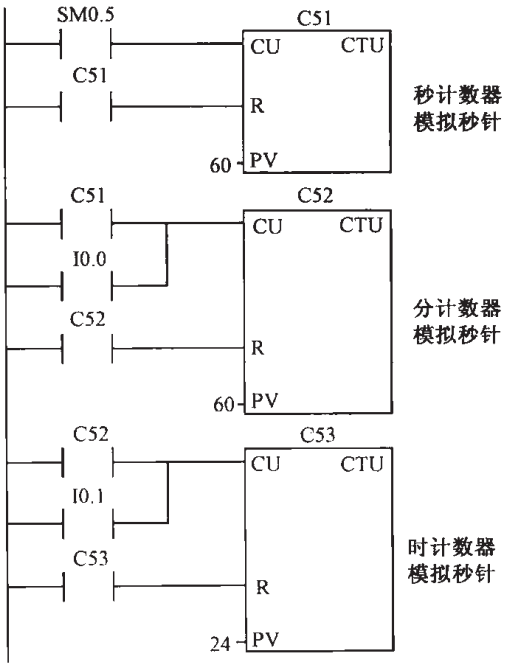


图 3-13　计数器组合实现时钟控制

```
LD      SM0.5
LD      C51
CTU     C51，60
LD      C51
O       I0.0
LD      C52
CTU     C52，60
LD      C52
O       I0.1
LD      C53
CTU     C53，24
```

实例分析

计数器串联组合实现时钟控制，实现 24 小时（即 1 天）时钟控制，常称为高精度时钟控制，如果加入显示屏输出部分，就可以作为 PLC 电子时钟。

3.2.3 脉冲触发控制

脉冲触发控制在 PLC 控制中属常见控制情况，可用微分操作指令或定时器实现。下面以具体实例介绍脉冲触发控制。

实例 42：用微分操作指令实现脉冲触发

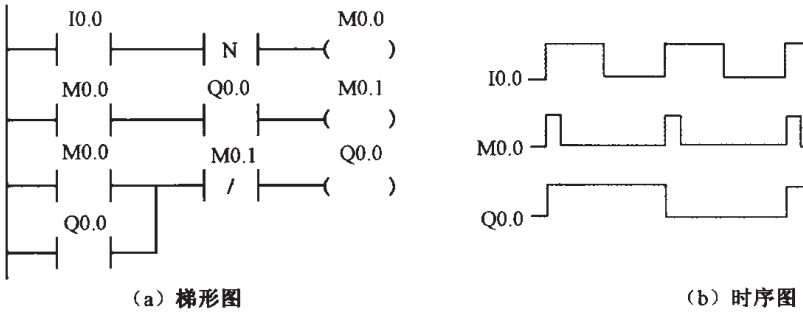
实例说明

用微分操作指令┤P├或┤N├实现脉冲触发控制。

实例实现

如图 3-14 所示，是用微分操作指令实现脉冲触发控制的，在输入 I0.0 的控制下，输出 Q0.0 不断实现翻转（ON/OFF…）。脉冲触发序列周期与输入信号 I0.0 的周期一致。本例梯形图对应的语句如下：

```
LD    I0.0
ED
=     M0.0
LD    M0.0
A     Q0.0
=     M0.1
LD    M0.0
O     Q0.0
AN    M0.1
=     Q0.0
```



（a）梯形图　　（b）时序图

图 3-14　微分操作指令实现脉冲触发控制

实例分析

本实例使用基本微分操作指令┤P├或┤N├实现触发脉冲，程序简单，运行效率高，占

用机时少，是非常适用的控制形式。

实例 43：用定时器实现周期脉冲触发控制

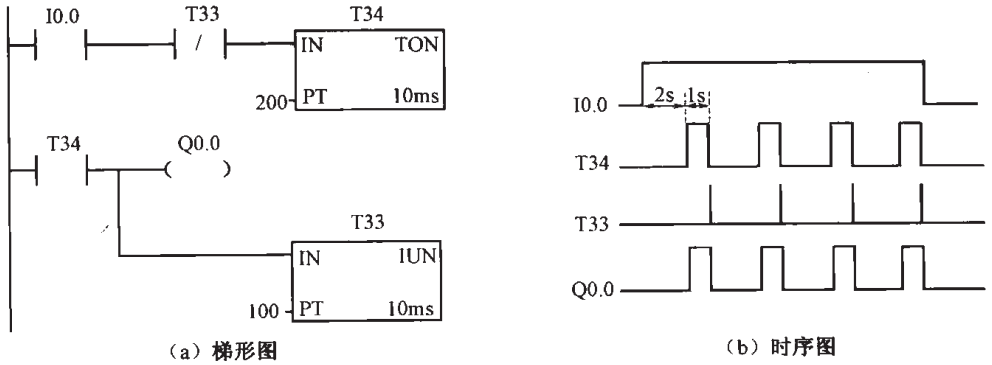
实例说明

利用定时器实现周期脉冲触发，且可根据需要灵活改变占空比。

实例实现

如图 3-15 所示是用两个定时器产生脉冲触发的例子，当输入 I0.0 接通时，输出 Q0.0 为脉冲序列，接通和断开交替进行。接通时间为 1 s，由定时器 T33 设定；断开时间为 2 s，由定时器 T34 设定。本例梯形图对应的语句如下：

```
LD      I0.0
AN      T33
TON     T34，200
LD      T34
=       Q0.0
TON     T33，100
```



（a）梯形图　　（b）时序图

图 3-15　定时器实现周期脉冲触发控制

实例分析

周期脉冲触发控制程序，也叫做闪烁控制程序（又称为振荡控制程序）。改变两个定时器 T33 和 T34 的时间常数，可以改变脉冲周期和占空比，是非常简洁适用的脉冲触发控制程序。

实例 44：用定时器实现脉宽可控的脉冲触发控制

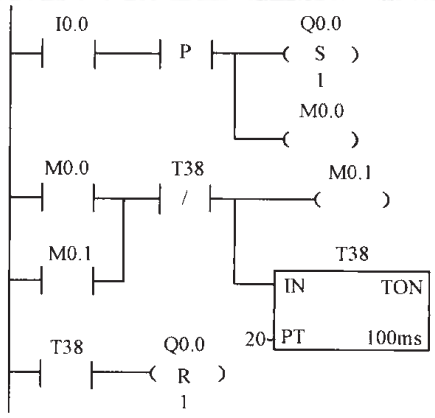
实例说明

在输入信号宽度不规范的情况下，如果需要脉冲宽度可控的触发脉冲，如何实现？在实例 43 周期脉冲触发控制程序的基础上，增加上升沿脉冲指令和 S/R 指令，结合定时器可以在输入信号宽度不规范的情况下，产生一个脉冲宽度固定的脉冲序列，该脉冲宽度通过改变定时器设定值 PT 进行调节。

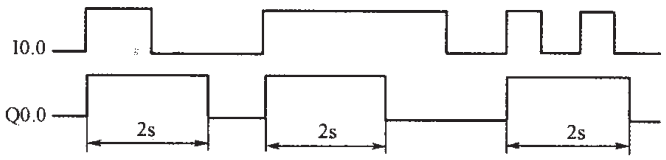
实例实现

如图 3-16 所示是使用定时器产生脉宽固定的触发脉冲的例子。该例子使用了上升沿脉冲指令和 S/R 指令，找出 Q0.0 的开启和关断条件，使其不论在 I0.0 的宽度大于或小于 2 s，都可以使 Q0.0 的宽度为 2 s。然后让定时器 T38 的计时输入逻辑在上升沿脉冲宽度小于设定脉冲宽度时，对输入脉冲宽度进行扩宽；在上升沿脉冲宽度大于设定脉冲宽度时，对输入脉冲宽度进行截取；在两个上升沿脉冲之间的距离小于设定脉冲宽度时，对后产生的上升沿脉冲无效。此三种情况见图 3-16（b）时序图；T38 在计时到后产生一个信号复位 Q0.0，然后自复位。本例梯形图对应的语句如下：

```
LD      I0.0
EU
S       Q0.0，1
=       M0.0
LD      M0.0
O       M0.1
AN      T38
=       M0.1
TON     T38，20
LD      T38
R       Q0.0，1
```



（a）梯形图



（b）时序图

图 3-16　脉宽可控脉冲触发控制

实例分析

该实例应用微分上升沿┤P├指令，将 I0.0 的不规则输入信号，转化为瞬时触发信号，通过 S/R 指令将 Q0.0 置位或复位，Q0.0 置位时间长短由定时器 T38 设定值 PT 的大小决定，因此 Q0.0 的宽度不受 I0.0 接通时间长短的影响。

3.2.4　分频控制

实例 45：二分频控制

在许多控制场合，需要对控制信号进行分频，常见的有二分频、四分频控制。下面以二分频为例，介绍分频控制的实现。

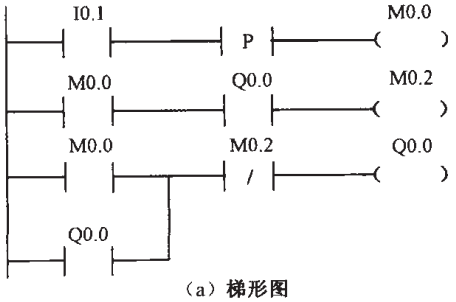
实例说明

二分频控制程序将输入信号脉冲 I0.1 分频输出，输出脉冲 Q0.0 为 I0.1 的二分频。

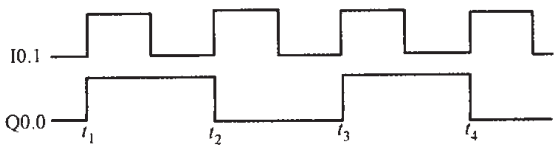
实例实现

在如图 3-17 所示二分频电路的梯形图和时序图中，当输入 I0.1 在 t_1 时刻接通（ON），此时内部标志位存储器 M0.0 上将产生单脉冲。然而输出映像寄存器 Q0.0 在此之前并未得电，其对应的常开触点处于断开状态。因此，扫描程序至第 2 行时，尽管 M0.0 得电，内部标志位存储器 M0.2 也不可能得电。扫描至第 3 行时，Q0.0 得电并自锁。此后这部分程序虽多次扫描，但由于 M0.0 仅接通一个扫描周期，M0.2 不可能得电。Q0.0 对应的常开触点闭合，为 M0.2 的得电做好了准备。等到 t_2 时刻，输入 I0.1 再次接通（ON），M0.0 上再次产生单脉冲。因此，在扫描第 2 行时，内部标志位存储器 M0.2 条件满足得电，M0.2 对应的常闭触点断开。执行第 3 行程序时，输出映像寄存器 Q0.0 断电，输出信号消失。以后，虽然 I0.1 继续存在，但由于 M0.0 是单脉冲信号，虽多次扫描第 3 行，输出映像寄存器 Q0.0 也不可能得电。在 t_3 时刻，输入 I0.1 第三次出现（ON），M0.0 上又产生单脉冲，输出 Q0.0 再次接通。t_4 时刻，输出 Q0.0 再次断电……得到输出正好是输入信号的二分频。这种逻辑每当有控制信号时，就将状态翻转（ON→OFF→ON→OFF→…），因此也可用作脉冲发生器。本例梯形图对应的语句如下：

```
LD      I0.1
EU
=       M0.0
LD      M0.0
A       Q0.0
=       M0.2
LD      M0.0
A       Q0.0
AN      M0.2
=       Q0.0
```



（a）梯形图



（b）时序图

图 3-17　二分频电路

实例分析

该实例梯形图程序中，用微分上升沿┤P├指令和两个内部标志位存储器 M0.0 与 M0.2 将规则频率的 I0.1 输入信号，转化为脉宽为 I0.1 两倍的 Q0.0 信号输出。

3.2.5 报警控制

故障报警控制是电气自动控制系统中不可缺少的重要环节，也是 PLC 控制系统中的常用部分。标准的报警功能应该是声光报警。下面分别以单故障报警控制与多故障报警控制为例，介绍故障报警控制的设计思路。

实例 46：单故障报警控制

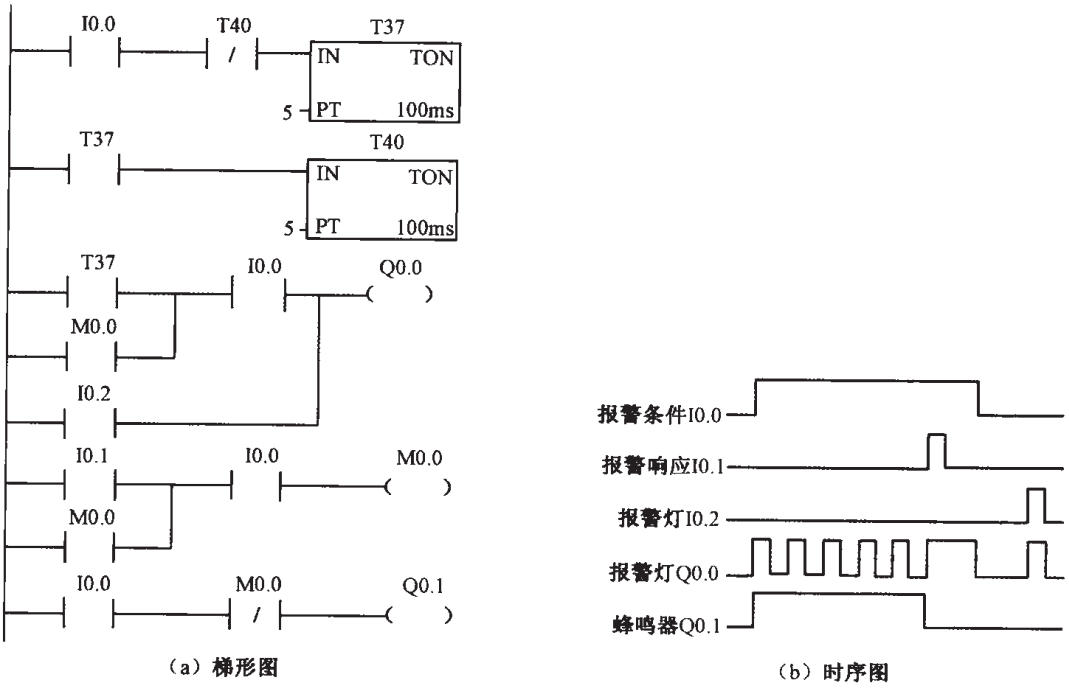
实例说明

本实例是用蜂鸣器和报警灯对一个故障实现声光报警控制的报警控制系统。

实例实现

如图 3-18 所示的梯形图程序为单故障报警控制程序。输入端子 I0.0 为故障报警输入条件，即 I0.0=ON 要求报警。输出 Q0.0 为报警灯，Q0.1 为报警蜂鸣器。输入条件 I0.1 为报警响应。I0.1 接通后，Q0.0 报警灯从闪烁变为常亮，同时 Q0.1 报警蜂鸣器关闭。输入条件 I0.2 为报警灯的测试信号。I0.2 接通，则 Q0.0 接通。本例梯形图对应的语句如下：

```
LD    I0.0
AN    T40
TON   T37，5
LD    T37
TON   T40，5
LD    T37
O     M0.0
A     I0.0
O     I0.2
=     Q0.0
LD    I0.1
O     M0.0
A     I0.0
=     M0.0
LD    I0.0
AN    M0.0
=     Q0.1
```



（a）梯形图　　（b）时序图

图 3-18　单故障报警控制

实例分析

该实例用定时器 T37 和定时器 T40 构成振荡控制程序，当故障报警条件 I0.0 接通后，每 0.5 s Q0.0 和 Q0.1 通断声光报警一次，反复循环，直到报警结束。

实例 47：多故障报警控制

实例说明

在实际的工程应用中，出现的故障可能不只一个，而是多个，这时的报警控制程序与一个故障的报警程序是不一样的。在声光多故障报警控制程序中，一种故障对应于一个指示灯，蜂鸣器只要用一个就可以，因此，程序设计时要将多个故障用一个蜂鸣器鸣响。

实例实现

图 3-19 为两种故障标准报警控制梯形图，图中故障 1 用输入信号 I0.0 表示；故障 2 用 I0.1 表示；I1.0 为消除蜂鸣器按钮；I1.1 为试灯、试蜂鸣器按钮。故障 1 指示灯用信号 Q0.0 输出；故障 2 指示灯用信号 Q0.1 输出；Q0.3 为报警蜂鸣器输出信号。本例梯形图对应的语句如下：

```
脉冲发生器
LDN     T40
TON     T37，10
LD      T37
TON     T40，20
故障指示灯 1
```

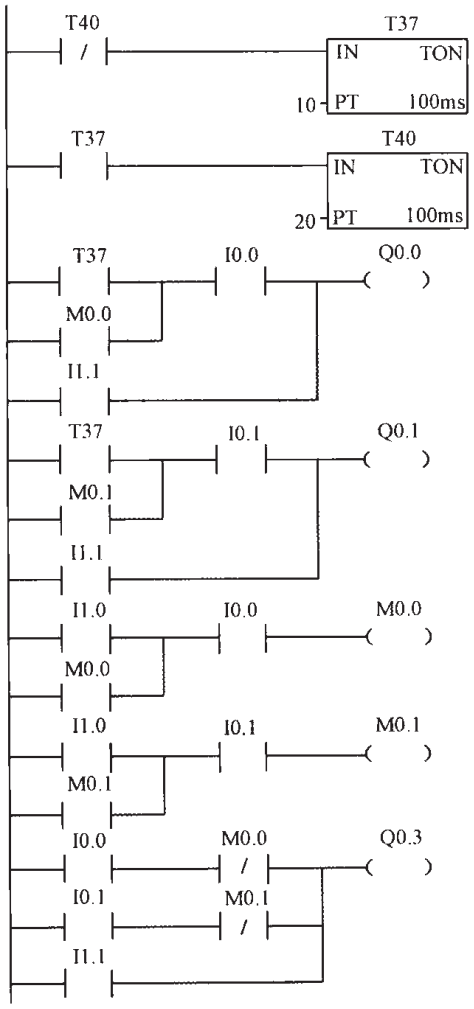


图 3-19 两种故障报警控制

```
LD      T37
O       M0.0
A       I0.0
O       I1.1
=       Q0.0
故障指示灯 2
LD      T37
O       M0.1
A       I0.1
O       I1.1
=       Q0.1
故障消除蜂鸣器逻辑 1
LD      I1.0
O       M0.0
A       I0.0
=       M0.0
故障消除蜂鸣器逻辑 2
LD      I1.0
O       M0.1
A       I0.1
=       M0.1
报警蜂鸣器
LD      I0.0
AN      M0.0
LD      I0.1
AN      M0.1
OLD
O       I1.1
=       Q0.3
```

实例分析

在图 3-19 所示的两种故障标准报警控制梯形图程序设计中，关键是当任何一种故障发生时，按消除蜂鸣器按钮后，不能影响其他故障发生时报警蜂鸣器的正常鸣响。该程序由脉冲触发控制、故障指示灯、蜂鸣器逻辑控制和报警控制电路四部分组成，采用模块化设计，值得读者在实际使用时参考。照此方法可以实现更多故障报警控制。

3.2.6 计数控制

实例 48：扫描计数控制

实例说明

在某些场合下需要计算扫描次数，一般可采用扫描计数控制程序来实现。

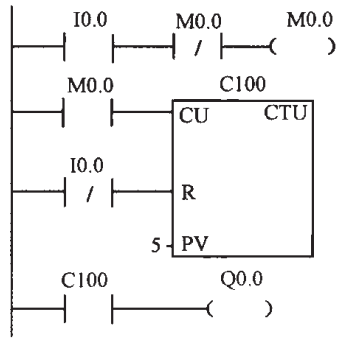
实例实现

如图 3-20 所示是扫描计数控制程序。输入 I0.0 接通后，内部标志位存储器 M0.0 每隔一个扫描周期接通一次，扫描周期用 T 表示。计数器 C100 对扫描次数进行计数，达到设定值时计数器 C100 有输出，其常开触点 C100 接通，输出映像寄存器 Q0.0 启动。本例梯形图对应的语句如下：

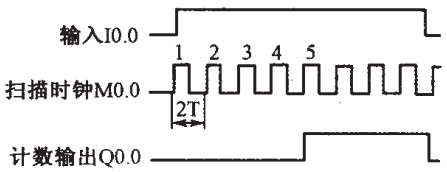
```
LD      I0.0
AN      M0.0
=       M0.0

LD      M0.0
LDN     I0.0
CTU     C100，5

LD      C100
=       Q0.0
```



（a）梯形图



（b）时序图

图 3-20　扫描计数控制程序

实例分析

本程序使用内部标志位存储器 M0.0 和计数器 C100 计数 PLC 内部扫描次数，程序简单适用，能很好地满足工程应用的需要。

实例 49：6 位数计数控制

实例说明

S7-200 PLC 的计数器的计数值为–32 767～32 767，计数位数不超过 5 位数，如果要进行 6 位数计数，需要将计数器串联构成 6 位加法计数器。

实例实现

如图 3-21 所示是 6 位数计数控制程序，其构成的 6 位数是 123456。计数器输入脉冲 I0.1，

复位输入脉冲 I0.0，当计数脉冲 I0.1 满 123 次后，C50 计数器的常开触点 C50 接通，C48 计数器在脉冲 I0.1 到来时计数，当 C48 计数器计满 1000 次后，C51 计数器计数一次，而后 C48 再计满 1000 次后，C51 计数一次，直到 C51 计数满 456 次，即共计数满 123+456*（999+1）= 123 456 次后，输出 Q0.0 接通。本例梯形图对应的语句如下：

```
LD      I0.1
EU
=       M0.0

LD      M0.0
A       C50
LD      C48
O       I0.0
A       M0.0
CTU     C48，999

LD      M0.0
LD      I0.0
CTU     C50，123

LD      C48
LD      I0.0
CTU     C51，456

LD      C51
=       Q0.0
```

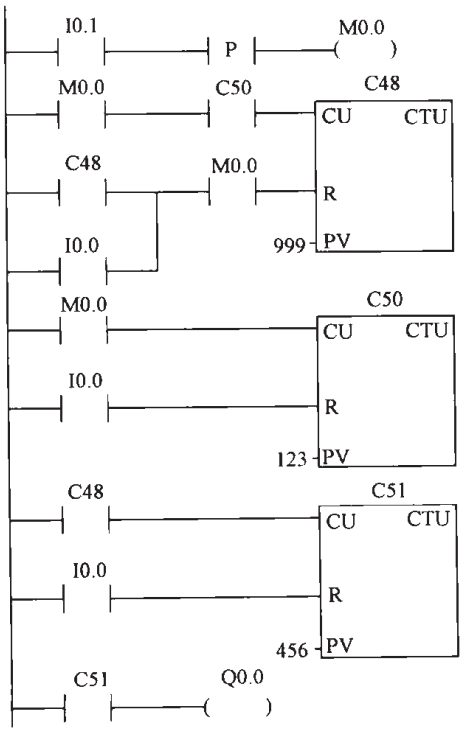


图 3-21　6 位数计数控制程序

实例分析

该控制程序是将 C48 和 C51 计数器串联得到的计数次数，再与 C50 计数次数相加的结果。C48 计数器计满 1000 次后，由其常开触点 C48 与 M0.0 复位，构成循环计数，叫做循环计数器。

3.2.7　顺序控制

顺序控制在工业控制系统中应用十分广泛。传统的控制器件继电器—接触器只能进行一些简单控制，且整个系统十分笨重庞杂，接线复杂，故障率高，有些更复杂的控制可能根本实现不了。而用 PLC 进行顺序控制则变得轻松简便，可以用各种不同指令，编写出形式多样、简洁清晰的控制程序。甚至一些非常复杂的控制也变得十分简单。下面介绍三种实用的顺序控制程序。

实例 50：用定时器实现顺序控制

实例说明

用定时器对被控对象实现顺序启停控制。

实例实现

如图 3-22 所示是用定时器编写的实现顺序控制的梯形图程序。该程序执行的结果是：当 I0.0 总启动开关闭合后，Q0.0 先接通。经过 5 s 后 Q0.1 接通，同时将 Q0.0 断开。再经过 5 s 后 Q0.2 接通，同时将 Q0.1 断开。又经过 5 s 后 Q0.3 接通，同时将 Q0.2 断开。再经过 5 s 又将 Q0.0 接通，同时将 Q0.3 断开。如此循环往复，实现了顺序启动/停止的控制。本例梯形图对应的语句如下：

```
LD      I0.0
O       T40
O       Q0.0
EU
AN      I0.1
AN      Q 0.1
=       Q0.0
TON     T37, 50

LD      T37
O       Q0.1
AN      I0.1
AN      Q 0.2
=       Q0.1
TON     T38, 50
```

```
LD      T38
O       Q0.2
AN      I0.1
AN      Q 0.3
=       Q0.2
TON     T39, 50

LD      T39
O       Q0.3
AN      I0.1
AN      Q 0.4
=       Q0.3
TON     T40, 50
```

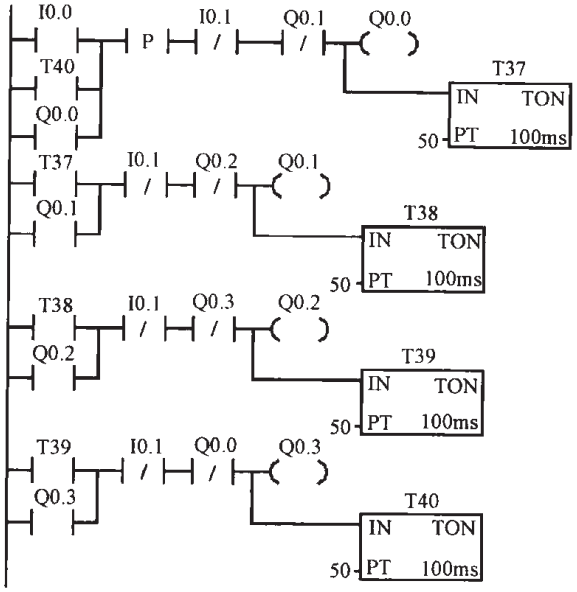


图 3-22　定时器实现顺序控制程序

实例分析

从该程序的实现可以看出，用定时器实现顺序控制的实质就是运用定时器的定时与延时功能，在不同时间点上实现被控对象的启停。

实例 51：用计数器实现顺序控制

实例说明

用计数器减 1 计数的原理，对被控对象实现顺序启停控制。

实例实现

如图 3-23 所示是用计数器编写的梯形图程序。当 I0.0 第一次闭合时 Q0.0 接通，第二次

闭合时 Q0.1 接通，第三次闭合时 Q0.2 接通，第四次闭合时 Q0.3 接通，同时将计数器复位，又开始下一轮计数。如此往复，实现了顺序控制。这里 I0.0 既可以是手动开关，也可以是内部定时时钟脉冲，后者可实现自动循环控制。程序中使用比较指令，只有当计数值等于比较常数时相应的输出才接通。所以每一个输出只接通一拍，且当下一输出接通时上一输出即断开。本例梯形图对应的语句如下：

```
LD      I0.0
LD      C40
A       I0.0
CTD     C40，4

LDW=    C40，3
=       Q0.0

LDW=    C40，2
=       Q0.1

LDW=    C40，1
=       Q0.2

LDW=    C40，1
OW=     C40，4
=       Q0.3
```

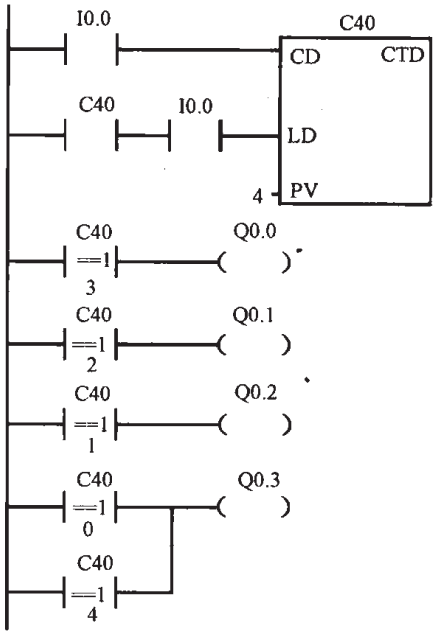


图 3-23　计数器实现顺序控制程序

实例分析

该程序利用减 1 计数器 C40 进行计数，由控制触点 I0.0 闭合的次数，驱动计数器计数，结合比较指令，将计数器的计数过程中间值与给定值比较，确定被控对象在不同时间点上的启停，从而实现控制各输出接通的顺序。

实例 52：用移位指令实现顺序控制

实例说明

用移位指令将移位数据存储单元中的数据位移动，当某数据位为“1”时，利用该位启动其后的输出，对被控对象实现顺序启停控制。

实例实现

如图 3-24 所示是用左移移位指令编写的顺序控制梯形图程序。该程序利用一个开关触点 I0.1 实现对输出映像寄存器 Q0.0、Q0.1、Q0.2 和 Q0.3 的顺序控制。I0.1 为移位脉冲控制触点，I0.1 每闭合一次 VB1 左移一位。当 VB1 前 4 位初始值为 0 时，VB1 的第零位置为 1，即 V1.0 为 1，此时输出 Q0.0 被接通，当 I0.1 第一次闭合时 VB1 左移一位 1，于是 VB1 中 V1.1 接通，使输出 Q0.1 被接通，同时 V1.0 断开。此后 I0.1 每闭合一次，VB1 置位的 1 左移

一位，使 VB1 的一位接通，从而接通一个输出端子。如此实现了将各输出顺序接通与断开。当 I0.1 第三次闭合时，Q0.3 被接通，当 I0.1 第四次闭合时，将 V1.0 各位复位，于是又开始新一轮循环。本例梯形图对应的语句如下：

```
LND     V1.0
AN      V1.1
AN      V1.2
AN      V1.3
S       V1.0, 1

LD      I0.1
EU
SLB     VB1, 1

LD      V1.0
=       Q0.0

LD      V1.1
=       Q0.1

LD      V1.2
=       Q0.2

LD      V1.3
=       Q0.3

LD      V1.4
A       I0.1
R       V1.0, 8
```

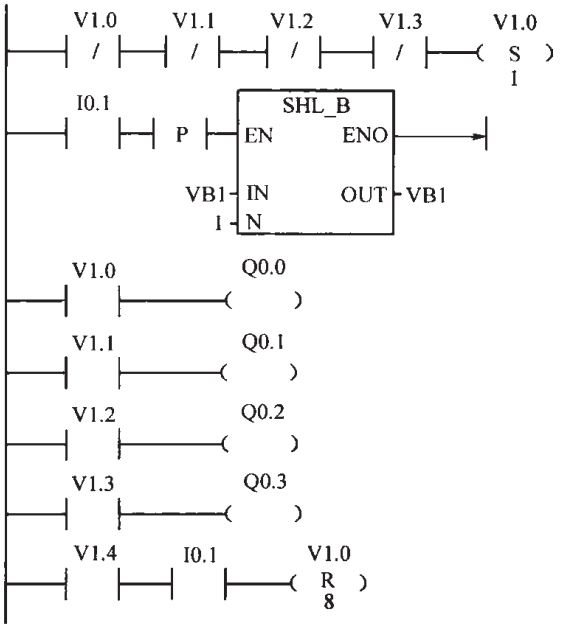


图 3-24　移位指令实现顺序控制程序

实例分析

该程序用左移位指令将移位数据存储单元中的数据位左移，利用左移的位启动其后的输出，确定被控对象在不同时间点上的启停。

除了上面介绍的顺序控制方法外，还有其他方法，譬如顺序控制功能指令，读者可根据上面的介绍，自行开发出更多更好的控制程序。

3.2.8　循环控制

循环控制也是 PLC 控制程序的常见情况，如循环计数控制、周期连续进行的顺序控制等均属循环控制范畴，属基本控制程序。下面以彩灯闪亮控制为例介绍循环控制。

实例 53：彩灯闪亮循环控制

实例说明

PLC 实现彩灯闪亮控制，具有结构简单、变换形式多样、价格低的特点，应用广泛。彩灯控制变换形式主要有三种，长通类、变换类和流水类。长通类是指彩灯照明或衬托底色作用，一旦彩灯接通，将长时间亮，没有闪烁；变换类是指彩灯的定时控制作用，彩灯时亮时灭，形成需要各种变换，如字形变换、色彩变换、位置变换等，其特点是定时通断，频率不高；流水类是指彩灯变换速度快，犹如行云流水、星光闪烁，其特点虽也是定时通断，但频率较高。对长通类亮灯，控制简单，只需一次接通或断开，属一般控制；对变换类和流水类闪亮，则要按预定节拍产生一个“环形分配器”，这个环形分配器控制彩灯按预设频率和花样变换闪亮。

实例实现

如图 3-25 所示为彩灯闪亮循环控制程序。该程序控制 A、B、C、D 4 盏彩灯，工作时，按下启动按钮（即 I0.0 接通）4 盏灯间隔 2 s 依次点亮，然后 4 盏灯以同样的频率同时闪烁 1 次，如此循环往复。按下停止按钮（即 I0.1 断开）后，4 盏灯全部熄灭。本例梯形图对应的语句如下：

```
LN      开始按钮：I0.0
O       M0.0
AN      停止按钮：I0.1
=       M0.0

LD      SM0.1
MOVB    1，VB1      //亮灯控制字节初始化

LD      M0.0
```

```
AN      T37
TON     T37，20
LD      M0.0
A       T37
SLB     VB1，1      //定时器和移位指令构成彩灯闪烁脉冲发生器程序

LDB=    VB1，2
O       M0.2
A       M0.0
=       A 彩灯：Q0.0    // A 彩灯亮

LDB=    VB1，4
O       M0.2
A       M0.0
=       B 彩灯：Q0.1    // B 彩灯亮

LDB=    VB1，8
O       M0.2
A       M0.0
=       C 彩灯：Q0.2    //C 彩灯亮

LDB=    VB1，16
O       M0.2
A       M0.0
=       D 彩灯：Q0.3    // D 彩灯亮
LD      M0.0
AB=     VB1，32
S       M0.2，1     // 四盏彩灯全亮

LD      M0.0
AB=     VB1，64
MOVB    1，VB1
R       M0.2，1     //彩灯亮控制位、字节复位
```

实例分析

该程序用定时器、比较指令和左移位指令构成彩灯循环闪亮的环形分配器，控制彩灯循环闪亮，属变换类和流水类彩灯闪亮控制。

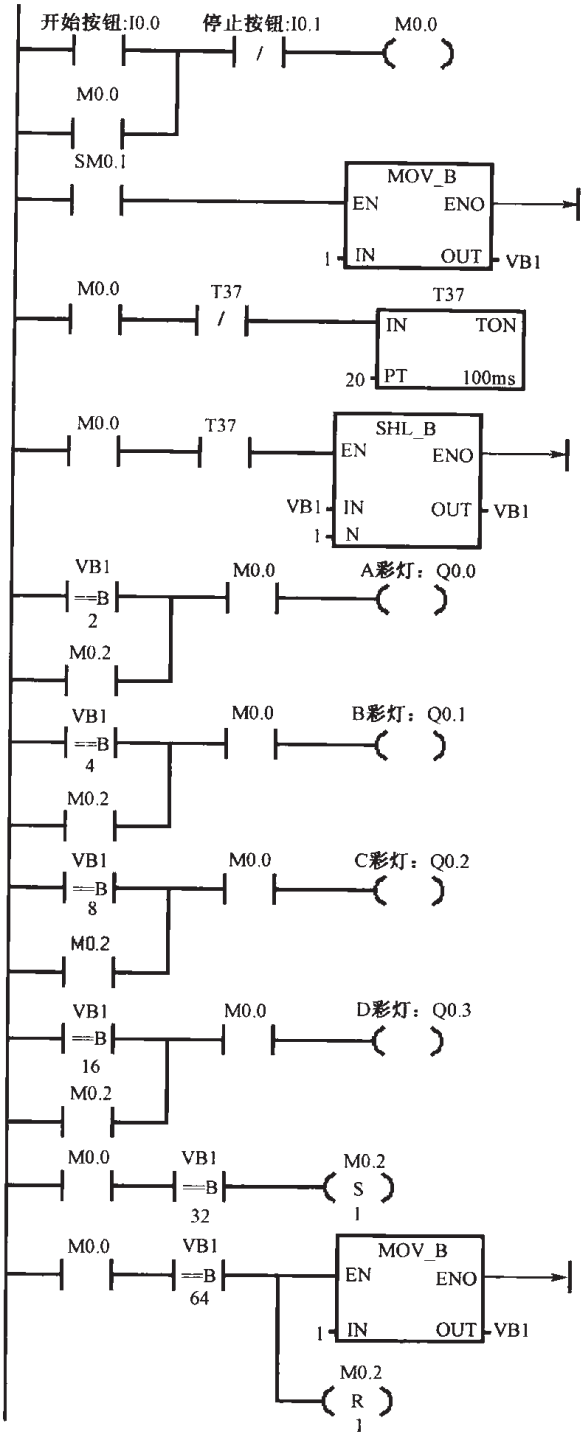


图 3-25　彩灯闪亮循环控制程序

3.2.9　多地点控制

多地点控制有时又称为异地控制，一般有两种情况：一种情况是多个开关、按钮或脉冲点共用一个 PLC 的接线端子，这类编程控制与一个开关、按钮或脉冲点接 PLC 的一接线端子一样，在此不过多介绍；另一种情况是多地点独立占用不同接线端子，控制同一输出端子的情况。这类多地点控制系统一般需要运用基本运算“与”、“或”、“非”等指令，同时还须列表分析建立控制的逻辑函数关系，根据逻辑函数关系设计梯形图程序。

实例 54：三地控制一盏灯

实例说明

要求在三个不同地方（A 地、B 地和 C 地）的开关独立控制一盏灯，任何一地的开关动作都可以使灯的状态发生改变。即不管开关是开还是关，只要有开关动作则灯的状态就发生改变。按此要求分配 PLC 的 I/O 地址为：A 地开关 S1 接 I0.0 端子，B 地开关 S1 接 I0.1 端子，C 地开关 S1 接 I0.2 端子；灯接在 Q0.0 端子上。

实例实现

三个不同的地方（A 地、B 地和 C 地）控制系统一般需要运用基本运算“与”、“或”、“非”等指令，同时还须列表分析建立控制的逻辑函数关系。

假如我们作如下规定：输入量为逻辑变量 I0.0、I0.1、I0.2，分别代表输入开关，输出量为逻辑函数 Q0.0，代表输出位寄存器；常开触点为原变量，常闭触点为反变量，常开触点闭合为“1”，断开为“0”，Q0.0 通电为“1”，不通电为“0”。这样可以控制要求列出逻辑函数真值表，如表 3-1 所示。

表 3-1 三地控制一盏灯逻辑函数真值表

I0.0	I0.1	I0.2	Q0.0
0	0	0	0
0	0	1	1
0	1	1	0
0	1	0	1
1	1	0	0
1	1	1	1
1	0	1	0
1	0	0	1

真值表按照每相邻两行只允许一个输入变量变化的规则排列，便可满足控制要求。根据此真值表可以写出输出与输入之间的逻辑函数关系式：

$$Q0.0=\overline{I0.0}\cdot\overline{I0.1}\cdot I0.2+\overline{I0.0}\cdot I0.1\cdot\overline{I0.2}+I0.0\cdot I0.1\cdot I0.2+I0.0\cdot\overline{I0.1}\cdot\overline{I0.2}$$

根据逻辑表达式，可设计出如图 3-26 所示梯形图程序，其梯形图对应的语句如下：

```
LDN   I0.0
AN    I0.1
A     I0.2
LDN   I0.0
A     I0.1
AN    I0.2
OLD
LD    I0.0
A     I0.1
```

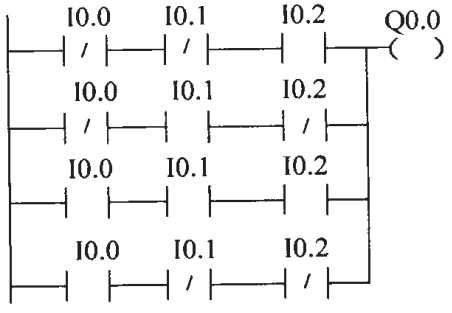


图 3-26 三地控制一盏灯的梯形图

```
A       I0.2
OLD
LD      I0.0
AN      I0.1
AN      I0.2
OLD
=       Q0.0
```

根据逻辑函数关系式设计程序，使编程者有章可循，更便于初学者掌握。当然根据控制要求也可设计如图 3-27 所示梯形图程序，其梯形图对应的语句如下（这个程序虽可实现控制要求，但其设计方法初学者不易掌握）：

```
LDN     I0.1
O       I0.2
LD      I0.1
ON      I0.2
ALD
A       I0.0
LDN     I0.0
O       I0.2
LD      I0.0
ON      I0.2
ALD
A       I0.1
OLD
LDN     I0.0
O       I0.1
LD      I0.0
ON      I0.1
ALD
A       I0.2
OLD
=       Q0.0
```

实例分析

这里举的例子是三地控制一盏灯，属多地点控制的实例，如图 3-26 和图 3-27 所示梯形图程序均能实现，只是逻辑层次关系清晰程度不一样，读者掌握难易也将会不同，从这个实例中可以发现其编程规律，并很容易的把它扩展到四地、五地甚至更多地点的控制。

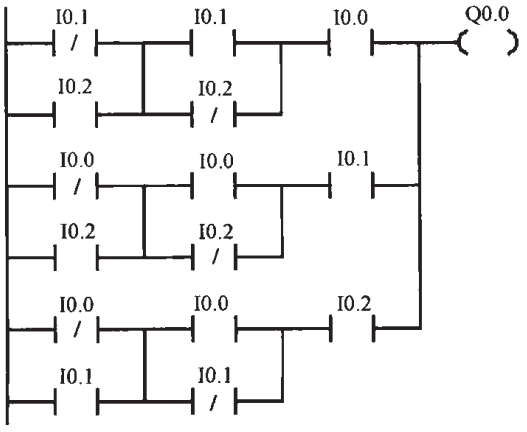


图 3-27　三地控制一盏灯的梯形图

3.2.10 高速计数器控制

在第 2 章高速计数输入/输出脉冲指令部分介绍高速计数器有 4 类，工作模式有 3 类 12 种，在使用时，首先要对高速计数器初始化，初始化可以用主程序中的程序段来实现，也可用子程序来实现，但通常用子程序来实现。而后是高速计数器执行及中断处理。下面以高速计数器模拟控制和测速控制应用为例，介绍高速计数器的常用控制方法。

实例 55：高速计数器模拟控制

实例说明

高速计数器模拟控制程序是用电压/频率 SFW01（Trankner Company）和 S7-200 CPU 214 的高速计数器 HSC 累计来自模拟量/频率转换器（A/F）的脉冲来模拟电压值的。

实例实现

在如图 3-28 所示梯形图程序中，主程序在第一个扫描周期调用初始化程序 SBR_0，仅在第一个扫描周期标志位 SM0.1=1 对子程序实现初始化。在子程序中，首先，把高速计数器 HSC1 的控制字节 SMB47 置为 16 进制数“FC”，其含义是：正方向计数，可更新预置值（PV），可更新当前值（CV），激活 HSC1，然后，用指令 HDEF 把高速计数器 HSC1 置成工作模式 0，既没有复位或起始输入，也没有外部的方向选择。当前值 SMD48 复位为 0，预置值 SMD52 置为“FFFF”（十六进制），定时中断 0 间隔时间 SMB34 置为 100 ms，中断程序 0 分配给定时中断 0（中断事件 10），并允许中断。用指令 HSC1 启动高速计数器。在中断程序中，每 100 ms 调用一次中断程序 0，读出高速计数器的数值后将其置零。通过 HSC1 计数值及变换关系（0～2000 Hz 对应于 0～10 V）来求被测的模拟电压值。本程序中，计数值除以 2，然后置入输出字节 QB0，以便通过 LED 来显示被测的电压值。显示值与 10 倍真实电压值相对应。例如，计数值为 200 除以 2 是 100，那么被测的模拟电压值就是 10.0V。因为计数器 100 ms 内共有 200 个计数脉冲，这正与 2000 Hz 与 10 V 相对应，假设计数值为 104，则实际电压值应为 5.2 V。在使用时需注意：定时中断间隔时间可在 5～255 ms 的范围内变化，然而通过设立一个标志，可根据需要来延长高速计数器的求值和复位时间，这样就有更长的扫描间隔，以便提高精确度，同时也会带来更长的更新时间。譬如，定时中断设为 100 ms，每调用一次，标志增加 1，仅当标志满 10 时，才对高速计数器求值和复位。也就是说，10 V 电压可接收的最大脉冲为 2000，这样，求值精确到 5/1000 V，即精确度是本程序的 10 倍，但同时速度也变成原来的 1/10。其梯形图对应的语句如下：

主程序：

```
LD      SM0.1          //仅首次扫描时，SM0.1=1
CALL    SBR_0：SBR0    //调用子程序 0
```

子程序：

```
LD      SM0.0          //SM0.0 总是 1
MOVB    16#FC          //设置 HSC1 控制字节：上升沿复位，上升沿启动，1*计数
                       //速率，正向计数，可改变方向，可更新 PV，可更新 CV（当
                       //前值），激活 HSC1
```

```
HDEF      1，0          //HSC1 工作于模式 0
MOVD      +0，SMD48     //HSC1 当前值复位
MOVD      16#FFFF，SMD52  //设置 HSC1 预置值，（本例未用）
MOVB      100，SMB34    //设置定时中断 0 间隔时间为 100ms
ATCH      INT_0：INT0，10  //指定定时中断事件 10，调用中断程序 0
ENI                     //允许所有中断
HSC       1             //启动高速计数器 HSC1
```

中断程序：

```
LD        SM0.0         //SM0.0 总是 1
MOVD      SMD48，VD100  // HSC1 的计数值存入 VD100
MOVD      +0，SMD48     //HSC1 当前值复位
MOVB      16#C0，SMB47  //重新设置 HSC1 控制字节：上升沿复位，上升沿启动，
                        //4*计数速率，反向计数，不改变方向，不更新 PV，
                        //可更新 CV，激活 HSC1
HSC       1             //启动高速计数器 HSC1
SRD       VD100，1      // HSC1 的计数值除以 2（即 VD00 右移 1 位）
MOVB      VB103，QB0    //在输出端 Q0.0 至 Q0.7，显示 10 倍被测电压值（0～100V）
```

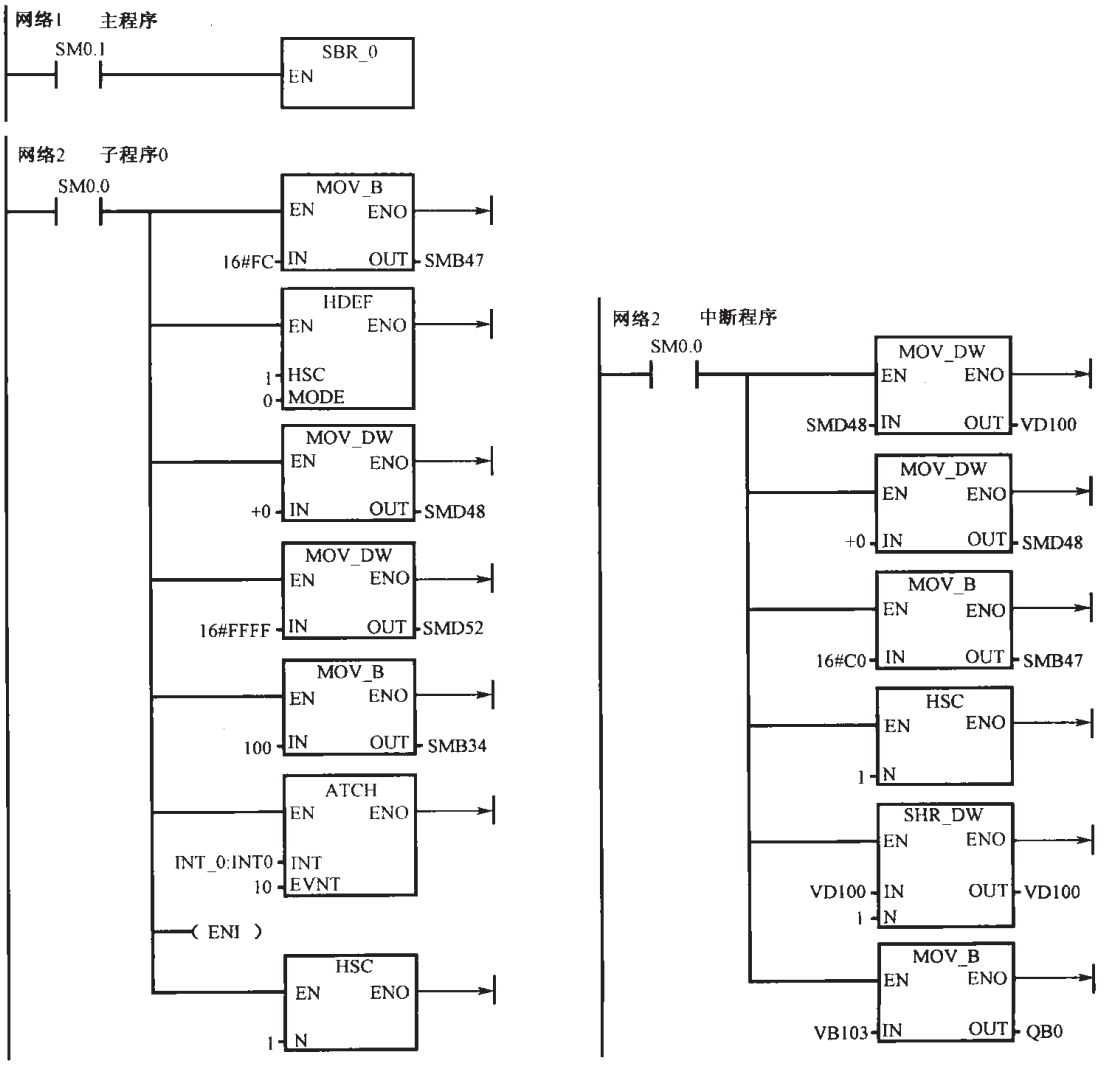


图 3-28　高速计数器的模拟量处理梯形图程序

实例分析

如图 3-28 所示梯形图程序，它由主程序、子程序 SBR_0 和中断程序 INT_0 三部分组成。主程序，在第一个扫描周期调用子程序 SBR_0；子程序 SBR_0 实现高速计数器和定时中断的初始化；中断程序 INT_0 是对高速计数器求值的定时中断程序。该实例利用 CPU 214 的高速计数器 HSC 及频率转换器来计算模拟电压。首先频率转换器将输入电压（0～10 V）转换为矩形脉冲信号（0～2000 Hz），再将此信号送入 CPU 214 高速记数器的输入端并累计脉冲数。当预置的间隔时间到后，通过累计脉冲数，计算出被测模拟电压值。

实例 56：高速计数器测速控制

实例说明

本实例是用一个高速计数器来测量转速，该程序是用主程序调用子程序的方式来实现。

实例实现

如图 3-29 所示是高速计数器测量转速的梯形图程序及实现原理，其梯形图对应的语句如下：

（1）主程序 OB1

网络 1　//程序初始化，在 PLC 第一次上电时执行子程序 SBR_0，进行初始化设置

```
LD      SM0.1
CALL    SBR_0: SBR0
```

（2）子程序 SBR_0

网络 1

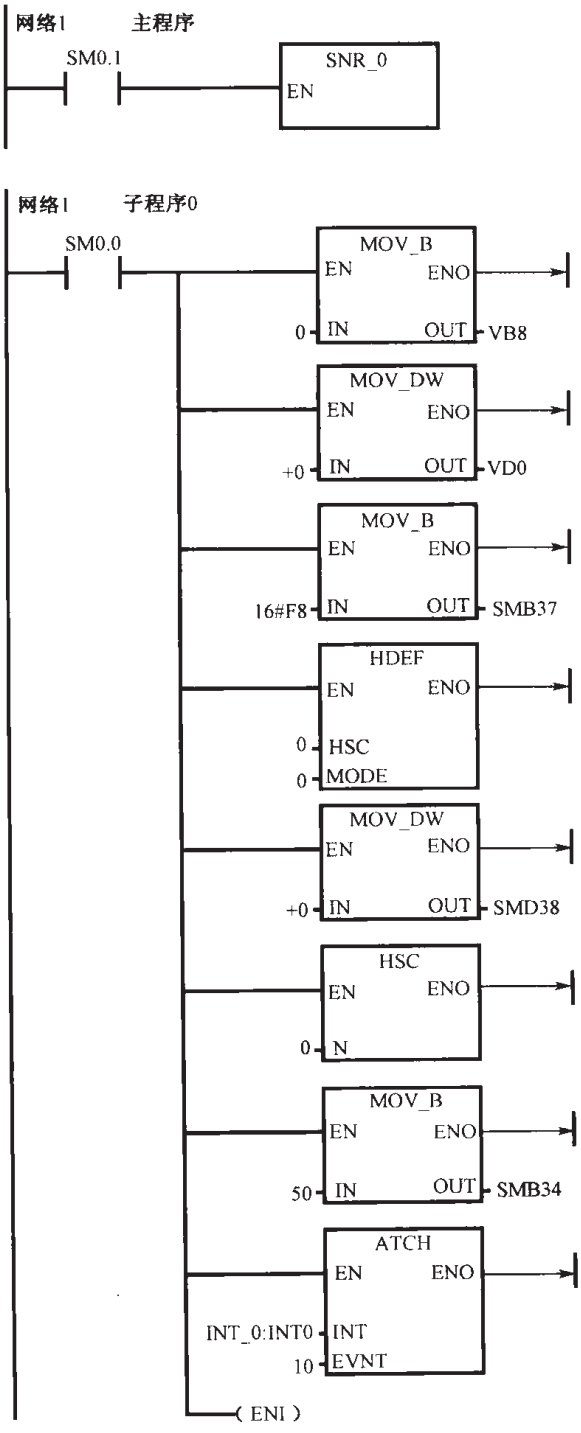
```
LD      SM0.0
MOVB    0,  VB8                 //将累加变量清零
MOVD    +0,  VD0
MOVB    16#F8,  SMB37           //设置高速计数器 HC0 的控制字节
HDEF    0,  0                   //设置工作模式
MOVD    +0,  SMD38              //设置高速计数器的初始值寄存器
HSC     0                       //执行 HSC 指令
MOVB    50,  SMB34              //将定时器中断 0 的时间设置为 50ms
ATCH    INT_0:INT0,  10         //将定时器中断 0 的处理程序设置为中断程序 INT_0
ENI
```

（3）中断程序 INT_0

网络 1

```
LD      SM0.0
+D      HC0,  VD0               //高速计数器的计数值与 VD0 进行累加
INCB    VB8                     //记录累加次数
HSC     0                       //执行 HSC 指令，重新为寄存器赋值
网络 2
```

```
LDB=    VB8,  32          //如果计数次数达到 32 次
MOVD    VD0,  VD4
/D      +32,  VD4         //求 32 次平均值
*D      100,  VD4
DTI     VD4,  VW10
MOVB    0,    VB8
MOVD    +0,   VD0
```



主程序：

程序初始化，在PLC上电运行的第一个扫描周期执行一次初始化子程序SBR_0，用于程序的运行的初始设置。

子程序SBR_0：

在PLC运行的第一个扫描周期，将用于记录累加数据次数和累加数据的中间变量VB8和VD0置0。

设置高速计数器HC0的控制字节SMB37，用十六进制表示（16#F8），也可以用二进制表示（2#11111000）。

设置高速计数器HC0工作模式为0，单相计数输入，没有外部控制功能。

设置高速计数器HC0初始值寄存器SMD38为0。

执行HSC指令，将控制字节SMB37、初始值/预置值寄存器（SMD38/SMD42）及工作模式写入高速计数器HC0。

设定定时中断事件的时间为50ms。

定时中断事件号10和中断处理程序INT_0建立关系。

允许中断，将定时中断事件和中断处理程序连接。

图 3-29　高速计数器测量转速梯形图程序

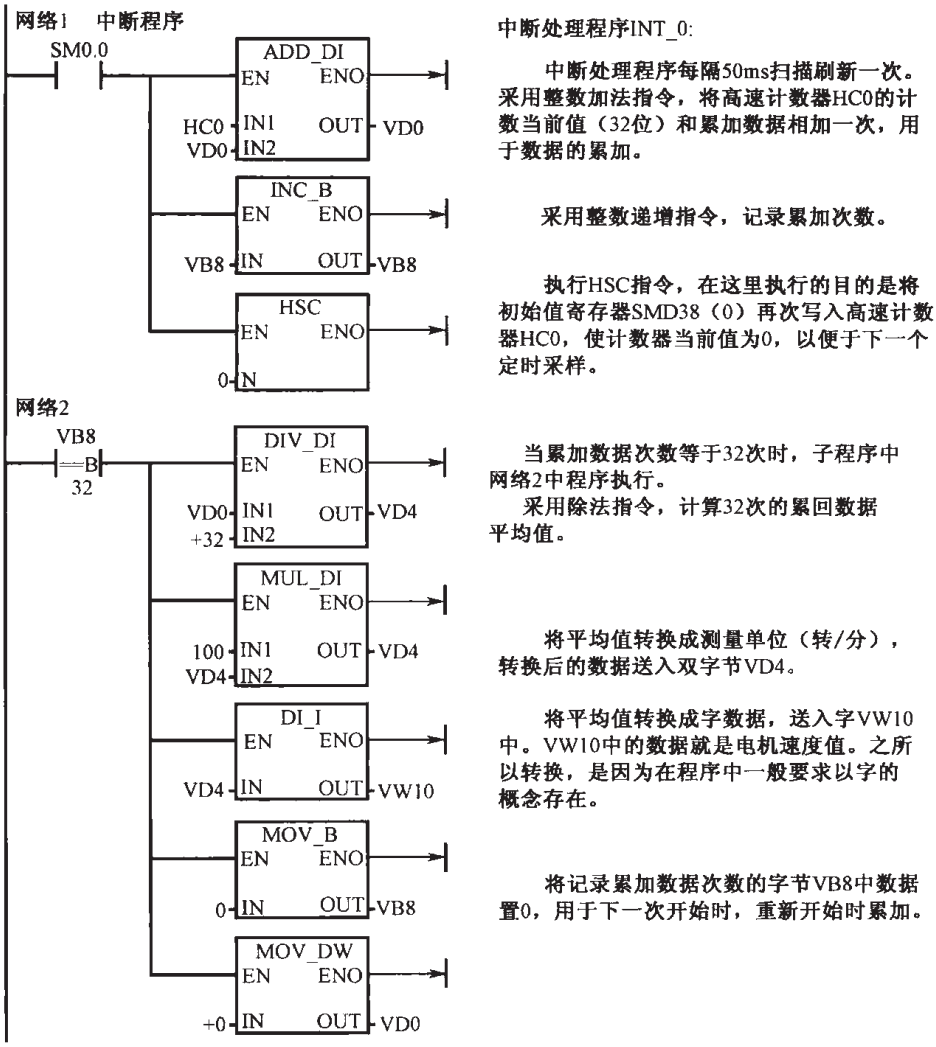


图 3-29 高速计数器测量转速梯形图程序（续）

实例分析

如图 3-29 所示梯形图程序，它由主程序、子程序 SBR_0 和中断程序 INT_0 三部分组成，实现高速计数器来测量转速控制。

基本控制程序是工程中经常用到的，是复杂程序的重要组成部分，对初学者来说是很重要的学习内容。此外，还需要学习掌握好 PLC 控制中的一些典型环节或简单实用的小系统控制程序的设计编程。

3.3 常用典型环节或系统控制编程

在基本控制程序基础上，下面介绍一些常用或典型的控制环节或系统控制编程。

实例 57：电动机正、反转控制

实例说明

电动机正、反转控制是电动机控制的重要内容，是工程控制中的典型环节，也是一个 PLC

控制系统开发人员必须熟练掌握和应用的重要部分。

实例实现

如图 3-30 所示是一个大家十分熟悉的电动机正、反转继电器控制电路图。图中用 KM0、KM1 的辅助触点实现自锁、互锁。

（1）首先确定 I/O 端子数

SB1、SB2、SB3 三个外部按钮是 PLC 的输入变量，需接在三个输入端子上，可分配为 I0.0、I0.1、I0.2；输出只有两个继电器 KM0、KM1，它们是 PLC 的输出端需控制的设备，要占用两个输出端子，可分配为 Q0.0、Q0.1。故整个系统需要用 5 个 I/O 端子：三个输入端子，两个输出端子。

下面列出 I/O 分配表：

输入端子：SB1：I0.0　　输出端子：KM0：Q0.0

　　SB2：I0.1　　KM1：Q0.1

　　SB3：I0.2

用于自锁、互锁的那些触点，因为无须占用外部接线端子而是由内部“软开关”代替，故不占用 I/O 端子。

（2）实际外部接线方法

如图 3-31 所示是 PLC 和外围设备的外部接线图，图中所表示的是 I0.0、I0.1、I0.2 共用一个“M”端，Q0.0、Q0.1 共用一个“L”端，输入开关均并联在直流电源 E 上，输出映像寄存器均并联在交流 220 V 电源上。直流电源由 PLC 供给，这时可直接将 PLC 电源端子接在开关上。而交流电源则是由外部供给。

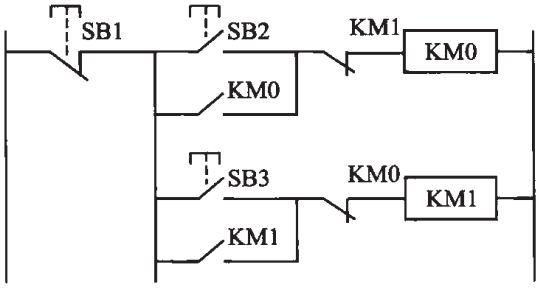


图 3-30　电动机正、反转继电器控制图

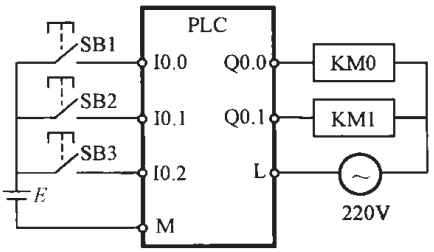


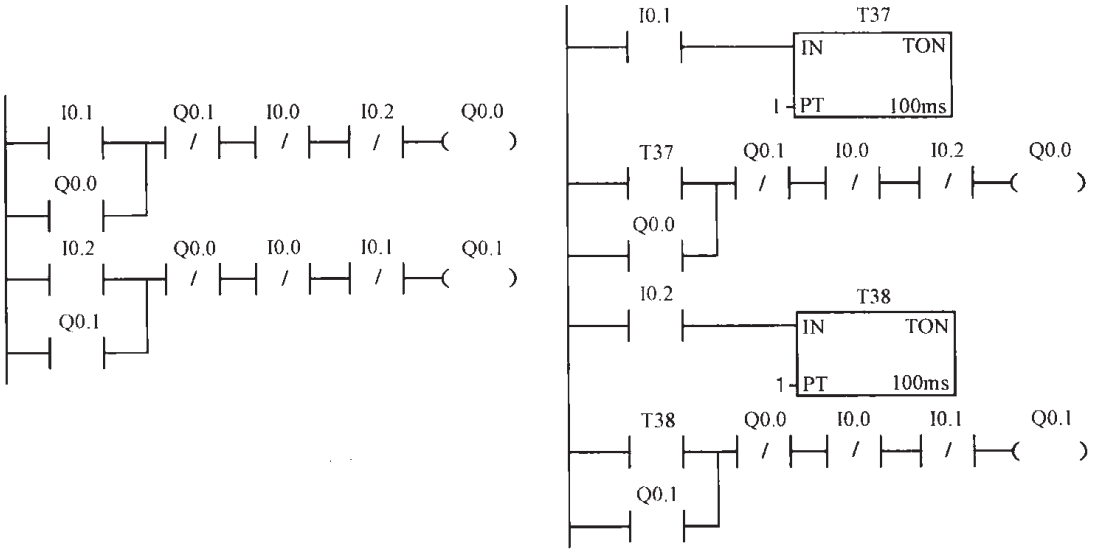
图 3-31　PLC 电动机控制外部接线图

（3）画梯形图

如图 3-32 所示梯形图程序中，I0.0、I0.1 和 I0.2 分别表示停止、正转和反转控制触点，Q0.0 和 Q0.1 分别表示电动机正转和反转输出映像寄存器，为了保证正、反转接触器 KM0 和 KM1 不会同时接通，如图 3-32（a）所示梯形图中采用按钮互锁（正转 I0.1 的常闭触点串入反转控制回路，反转 I0.2 的常闭触点串入正转控制回路）和输出映像寄存器触点互锁（正转输出映像寄存器 Q0.0 的常闭触点串入反转控制回路，反转输出映像寄存器 Q0.1 的常闭触点串入正转控制回路），保证 Q0.0 和 Q0.1 不会同时接通，属双互锁保险型。

但是，如图 3-32（a）所示梯形图程序仍然存在安全隐患。实际上，接触器通断变化的时间是极短的。如果电动机正转，Q0.0 及其相连的正转接触器接通；此时按反转按钮 SB3，

I0.2 触点动作使 Q0.0 断开，Q0.1 接通，PLC 的输出映像寄存器向外发出通断命令，正转接触器断开其主触点，电弧尚未熄灭时，反转接触器主触点已接通，将造成电源瞬时短路。为了避免这种情况发生，在图 3-32（a）所示梯形图程序基础上增加了两个定时器 T37 和 T38（如图 3-32（b）所示），进行正、反转切换时，被切断的接触器是瞬时动作的，而被接通的接触器要延时一段时间才动作，避免了电源瞬时短路。



（a）互锁控制梯形图　　（b）改进的互锁控制梯形图

图 3-32　电动机正反转控制梯形图

实例分析

该实例运用了自锁、互锁等基本控制程序，实现常用的电动机正、反转控制。因此，可以说基本控制程序是基本、大型和复杂程序的基础。实际设计程序时，还要考虑控制动作是否会导致电源瞬时短路等情况。

实例 58：电动机Y-△减压启动控制

实例说明

电动机Y-△减压启动控制是异步电动机启动控制中的典型控制环节，属常用控制小系统。

实例实现

下面就其控制系统程序设计原理和思路予以介绍。

（1）首先确定 I/O 端子数

电动机Y-△减压启动继电控制线路如图 3-33 所示。SB1 和 SB2 外部按钮是 PLC 的输入变量，KM1、KM2、KM3 是 PLC 的输出变量，列出 I/O 分配表如下。

输入端子：停止按钮 SB1：I0.0　　输出端子：KM1：Q0.1

启动按钮 SB2：I0.1　　KM2：Q0.2

KM3：I0.3

（2）PLC 与外部器件的接线

控制接线图如图 3-34 所示，图中电动机由接触器 KM1、KM2、KM3 控制，其中 KM3 将电动机定子绕组连接成星形，KM2 将电动机定子绕组连接成三角形。KM2 与 KM3 不能同时吸合，否则将产生电源短路。在程序设计过程中，应充分考虑由星形向三角形切换的时间，即由 KM3 完全断开（包括灭弧时间）到 KM2 接通这段时间应相互锁住，以防电源短路。

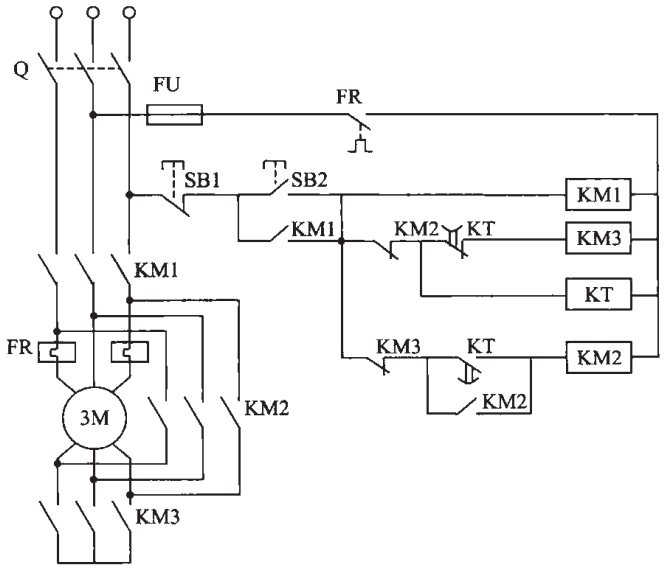


图 3-33　电动机Y-△减压启动控制接线图

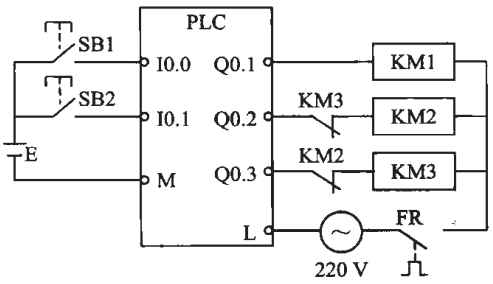


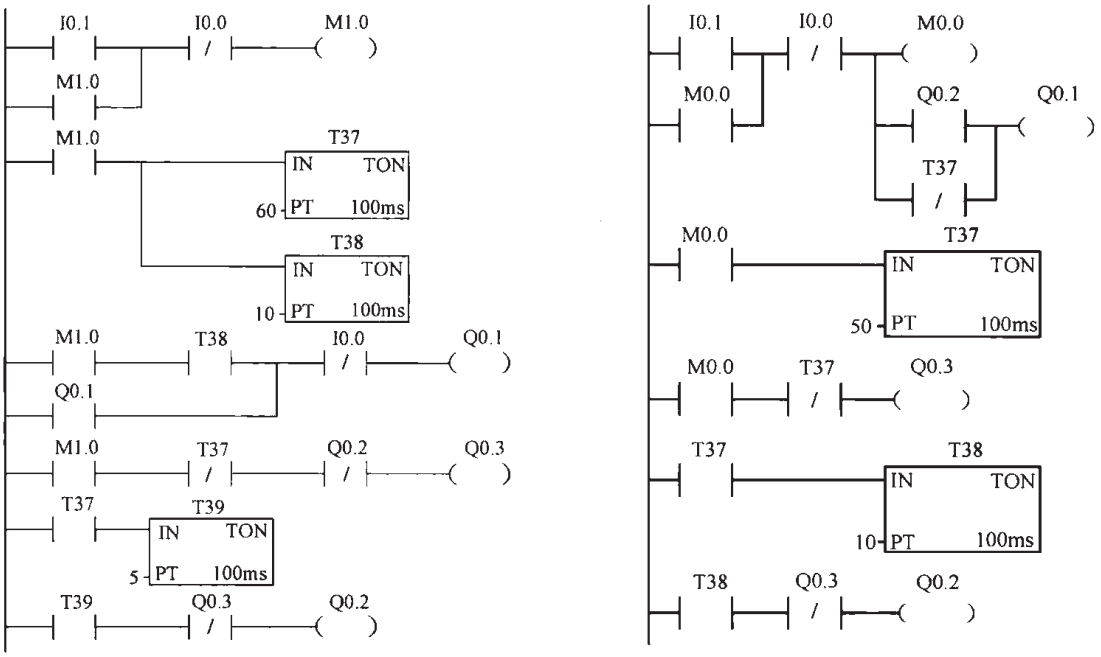
图 3-34　电动机Y-△减压启动控制接线图

（3）梯形图程序

电动机Y-△减压启动控制程序如图 3-35 所示，（a）、（b）两个控制程序功能相同。下面就其控制原理进行介绍。

在图 3-35（a）方案 1 所示梯形图程序中，启动时，按下 SB2，I0.1 常开闭合，此时 M1.0 接通，定时器 T37 和 T38 接通，Q0.3 也接通，KM3 接触器通电，T38 定时 1 s 后，Q0.1 接通，KM1 接触器通电，此时，电动机进入星形（Y）降压启动；星形（Y）降压启动 5 s 后，定时器 T37 已定时 6 s 了，KM3 接触器断电，定时器 T39 开始计时，计时 0.5 s 后，Q0.2 接通，KM2 通电，KM1 接触器已通电，此时电动机三角形（△）联结，进入正常工作状态。按下 SB1，M1.0 断电，电动机停止运行。

在图 3-35（b）方案 2 所示梯形图程序中，启动时，按下 SB2，I0.1 常开闭合，此时 M0.0 接通，定时器接通，Q0.1、Q0.3 也接通，KM1、KM3 接触器通电，电动机进入星形（Y）降压启动。延时 5 s 后，定时器 T37 动作，其常闭触点断开，使 Q0.1、Q0.3 断开，KM1、KM2 断电。T37 的常开触点闭合，接通定时器 T38，延时 1 s 后，T38 动作，Q0.1、Q0.2 接通，KM1、KM2 接通，电动机三角形（△）联结，进入正常工作。按下 SB1，M0.0 断电，电动机停止运行。



（a）方案1　　（b）方案2

图 3-35　电动机Y-△减压启动控制梯形图

实例分析

电动机Y-△减压启动属常用控制系统，在 3-34（a）程序中，使用 T37、T38、T39 定时器将电动机的星形（Y）减压启动到三角形（△）全压运行过程进行控制，在 Q0.2 和 Q0.3 两梯级中，分别加入互锁触点 Q0.3 与 Q0.2，保证 KM2 和 KM3 不能同时通电，此外，定时器 T39 定时 0.5 s，目的是 KM3 接触器断电灭弧，避免了电源瞬时短路。在 3-35（b）程序中，使用 T37 定时器，将 KM1 和 KM3 同时通电，电动机星形（Y）减压启动 5 s，而后将 KM1 断电，使用 T38 定时器，将 KM2 通电后，再让 KM1 通电，同样避免了电源瞬时短路。两控制程序均实现了电动机启动到平稳运行，说明实现相同的控制任务，可以设计出不同的控制程序，读者可根据控制的实际情况，开发出更好更优的控制程序。

实例 59：电动机的软启动控制

实例说明

电动机的软启动控制又称为电动机定子串电阻启动控制，属电动机控制中的常见情况。电动机的软启动控制程序说明了带短路软启动开关的鼠笼式三相感应异步电动机的自动启动过程。通过这种短路软启动控制，保证电动机首先减速启动，一定时间段后达到额定转速。

实例实现

电动机的软启动控制系统由 PLC I/O 分配、接线及控制程序几部分组成。设计实现过程如下：

（1）I/O 分配

PLC 的 I/O 端子分配如下。

输入端子：I0.0　启动按钮，常开触点，启动电动机；
　　I0.1　停止按钮，常闭触点，停止电动机；
　　I0.2　电动机电路断路器，常闭触点。
输出端子：Q0.0　电动机启动器；
　　Q0.1　旁路接触器。

（2）PLC 接线图

PLC 的接线图如图 3-36 所示。启动按钮接在输入端 I0.0，启动按钮关闭时实现电动机软启动。停止按钮接在输入端 I0.1，停止按钮断开时，电动机停止。电动机电路断路器接在输入端 I0.2，当电动机过载时电动机电路断路器断开，电动机停止。

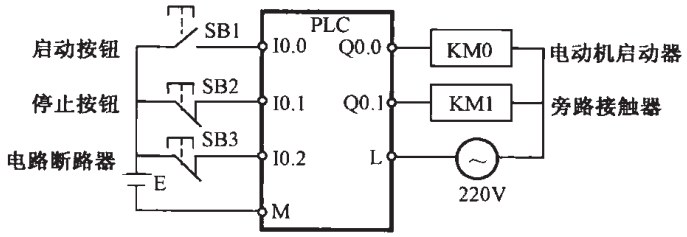


图 3-36　PLC 电动机控制外部接线图

（3）梯形图程序

梯形图程序如图 3-37 所示，电动机启动运行的条件是：内存标志位 M1.0 互锁取消。如果接在输入端 I0.0 的常开触点和接在输入端 I0.1 的常闭触点同时动作（即：I0.0 为 ON，I0.1 为 OFF），则设置内存标志位 M1.0 互锁，直至两个点动开关又回到初始状态，才取消互锁。其梯形图对应的语句如下：

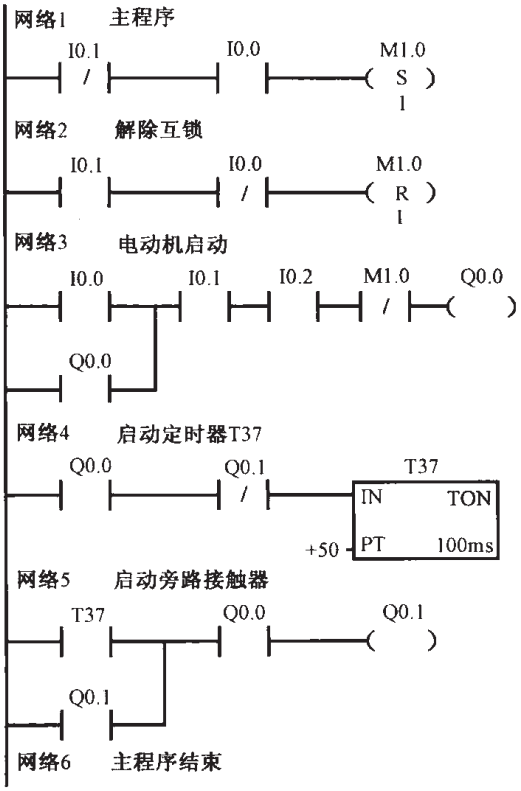


图 3-37　电动机软启动控制梯形图

```
LDN     I0.1        //OFF 点动开关动作
A       I0.0        //ON 点动开关动作
S       M1.0，1     //互锁内存标志位置位（M1.0=1）

LD      I0.1        //OFF 点动开关未动作
AN      I0.0        //ON 点动开关未动作
R       M1.0，1     //互锁内存标志位复位（M1.0=0）

LD      I0.0        //ON 点动开关未动作
O       Q0.0        //锁定电动机启动器
A       I0.1        //OFF 点动开关未动作
A       I0.2        //电动机电路断路器未动作
AN      M1.0        //互锁内存标志位未复位
=       Q0.0        //电动机启动

LD      Q0.0        //电动机运行
AN      Q0.1        //旁路接触器
TON     T37，+50    //启动 T37（50*100 ms=5 s）

LD      T37         // T37 溢出
O       Q0.1        //锁定旁路接触器
A       Q0.0        //电动机运行
=       Q0.1        //旁路接触器
```

实例分析

该实例梯形图程序控制原理分析如下：

内存标志位 M1.0 互锁取消后，按下 I0.0 的常开触点，即 ON 时，无互锁（M1.0），电动机电路断路器（I0.2）常闭触点未动作，I0.1 的常闭触点未动作。另外，再通过对 Q0.0 动作或逻辑运算完成启动锁定。此时，启动电阻还未被短接，电动机以减速启动。如果电动机已启动（Q0.0），并且用于旁路接触器的输出 Q0.1 还未被置位，计时器 T37 开始计时，计时 5 s 后，如果电动机仍处于启动状态（Q0.0），则启动接在输出端 Q0.1 的旁路接触器，通过对 Q0.1 动作或逻辑运算完成旁路锁定，电动机正常运行。

实例 60：物流检测控制

实例说明

物流检测是工业控制中的常见控制，使用范围广，如工业计数检测，零部件或产品合格检测等。下面以产品合格检测为例介绍物流检测控制的设计原理及思路。

图 3-38 是一个物流检测示意图。图中有三个光电传感器 BL1、BL2、BL3。BL1 检测有无次品到来，有次品到则“ON”。BL2 检测凸轮的凸起，凸轮每转一圈则发一个移位脉冲。

因物品间隔一定，故每转一圈有一个物品到，所以 BL2 实为检测物品到的传感器。BL3 检测有无次品落下。SB 是手动复位按钮，图中未画。当次品移至 4 号位时，控制电磁阀 YA 打开使次品落到次品箱内。若无次品则物品移至传送带右端则自动掉入正品箱内，于是可将次品和正品分开。

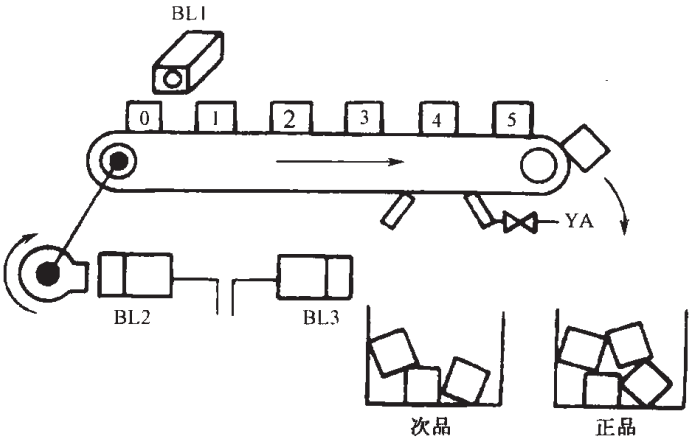


图 3-38　物流检测示意图

实例实现

根据此任务可设计该检测系统如下：

（1）I/O 分配

输入端子：I0.0：BL1　　输出端子：Q0.0：YA

I0.1：BL2

I0.2：BL3

I0.3：SB

（2）PLC 与现场器件的实际接线图如图 3-39 所示。

（3）梯形图（如图 3-40 所示），其梯形图对应的语句如下：

```
LD      I0.3
R       V100.0，4

LD      I0.1
EU
SHRB    I0.0，V100.0，4

LD      V100.3
S       Q0.0，1

LD      I0.2
R       Q0.0，1
```

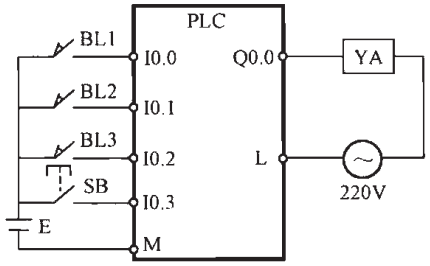


图 3-39 PLC 物流检测控制外部接线图

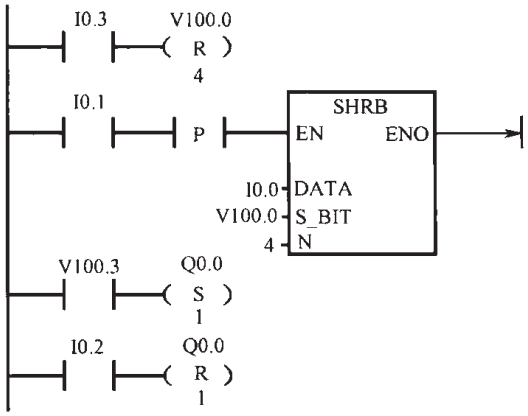
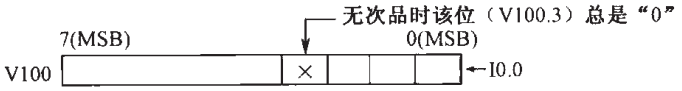


图 3-40 物流检测控制梯形图

实例分析

说明：I0.3 闭合，V100.0～V100.3 复位，当无次品来时，I0.0 总是“OFF”，于是 V100.0 中输入“0”。每来一个物品，I0.1 则“ON”一次，即发一次移位脉冲，V100 中左移一位。但因输入全是“0”，故移位后各位上也全是“0”，于是 V100.3 总是“OFF”。V100.0 与 V100.3 的关系图如下：



V100.3 作为 Q0.0 的置位端，当 V100.3 为“OFF”时，Q0.0 也为“OFF”，电磁阀 YA 不打开，物品全部到正品箱内。而当有次品来时 I0.0“ON”，此时 V100.0 中输入“1”。此后每来一个物品则 I0.1“ON”一次，发一个移位脉冲，使 V100.0 中的“1”左移一位。到第 4 个移位脉冲来时恰好这个“1”移至 V100.3 位上，于是 V100.3“ON”，将 Q0.0 接通，电磁阀打开，次品落下（此时次品也恰好移到传送带的 4 号位上）。BL3 检测到次品落下后，I0.2→“ON”，使 Q0.0→“OFF”，电磁阀重新关闭。这就是该检测系统的工作原理。

从该实例说明可以看出，这样的系统若用传统继电器控制实现是很麻烦的，而用 PLC 实现则十分简单。只用移位、置位指令即可编制这样简单的小程序。外围设备也十分简单，由几个输入光电开关、一个电磁阀即可构成这样的检测系统，大大简化了外部接线。而且可以随时根据需要更改程序。

实例 61：钻孔动力头控制

实例说明

钻孔动力头控制是工业加工制造中的常见情况，属比较典型的控制小系统。

实例实现

钻孔动力头小系统其控制程序设计一般按分析工艺流程图列出动作顺序表，分配 PLC I/O 接线端子，设计梯形图程序的步骤进行。

（1）工艺流程图与动作顺序表

如图 3-41 所示流程是某机械加工自动线中一个钻孔动力头的工艺流程。图中，1ST～3ST 为限位开关，SB 为启动按钮，1YA～3YA 为电磁阀，KT 为延时继电器。

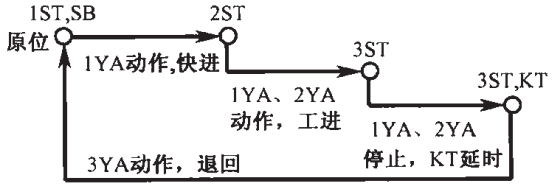


图 3-41　钻孔动力头工艺流程

开始时，动力头在原位，压着限位开关 1ST，即 1ST 闭合。当按下启动按钮 SB 后，电磁阀 1YA 动作，动力头快进；接近工件时，碰上位置开关 2ST，2ST 闭合，电磁阀 2YA 得电动作，动力头转为工进；工进到位时，碰上位置开关 3ST，则 1YA、2YA 断电，动力头停止前进；延时继电器 KT 接通，设延时 1s，延时结束，电磁阀 3YA 得电，动力头退回，当退回到原位时，碰着 1ST，此时，3YA 断电，动力头停止在原位上。每按一下启动按钮，重复一次上述循环过程。

钻孔动力头的动作顺序如表 3-2 所示。输入条件中，“1ST, SB”表示 1ST 与 SB 同时闭合；在输出栏中，“＋”表示电磁阀得电，“－”表示断电。

表 3-2　钻孔动力头的动作顺序表

步　序	输 入 条 件	输　出		
		1YA	2YA	3YA
原位	1ST	－	－	－
快进	1ST, SB	＋	－	－
工进	2ST	＋	＋	－
延时	3ST	－	－	－
退回	3ST, KT	－	－	＋
原位	1ST	－	－	－

（2）I/O 分配

根据动作顺序表与各开关、电磁阀等现场器件确定 PLC 的相应地址编号如下：

输入端子：I0.0：1ST　　输出端子：Q0.1：1YA

I0.1：2ST　　Q0.2：2YA

I0.2：3ST　　Q0.3：3YA

I0.3：SB

（3）PLC 与现场器件的实际接线图（如图 3-42 所示）

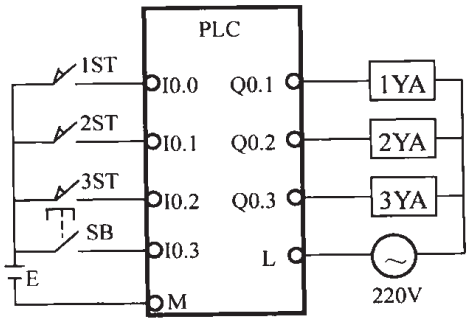


图 3-42　PLC 动力头控制外部接线图

（4）梯形图（如图 3-43 所示）

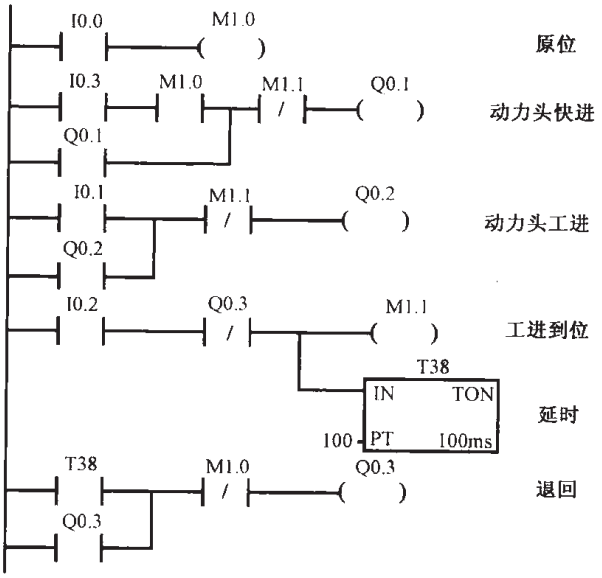


图 3-43 PLC 动力头控制梯形图

实例分析

满足图 3-41 所示工艺流程的梯形图如图 3-43 所示。动力头在原位时，碰着 1ST，输入映像寄存器常开触点 I0.0 闭合，内部标志位存储器 M1.0 得电，其常开触点闭合，常闭触点断开。按下启动按钮 SB，I0.3 闭合，输出映像寄存器 Q0.1 得电，驱动电磁阀 1YA，使动力头快进，同时，输出映像寄存器 Q0.1 的动合触点闭合、自锁。快进到位时，碰着 2ST，I0.1 闭合，输出映像寄存器 Q0.2 得电并自锁，驱动电磁阀 2YA，使动力头工进。工进到位时，碰着 3ST，I0.2 闭合，内部标志位存储器 M1.1 得电，其常闭触点断开，电磁阀 1YA、2YA 断电，动力头停止前进，同时定时器 T38 开始延时，延时时间结束后，定时器的常开触点闭合，输出映像寄存器 Q0.3 得电并自锁，驱动电磁阀 3YA，动力头退回。当动力头退回到原点时，I0.0 闭合，内部标志位存储器 M1.0 再次得电，其常闭触点断开，输出映像寄存器 Q0.3 断电，动力头停止在原位上。再按下启动按钮 SB 后，又重复上述过程。

实例 62：液位控制

实例说明

如图 3-44 所示为汽车涂装前处理生产线中的磷化槽示意图。汽车白车身挂在输送链上，经过磷化槽时会被涂上一层磷化膜。要求磷化液要保持一定的深度。液位由液位计来检测，图 3-44 中所示 A 为高液位，B 位低液位，C 为极低液位。初始时，按下启动按钮，加液泵起动，开始加液，当液位到 B 位置时给输送链发信号，表示输送链可以启动。液位到 A 位时停止加液。当液位为低液位时，液位开关 B 断开，此时启动加液泵（由接触器 KM 控制）加磷化液，等液位到了 A 液位，A 液位开关闭合，停掉加液泵，停止加液。如果由于某些原因（如泄露）导致液位处于 C 液位，则 C

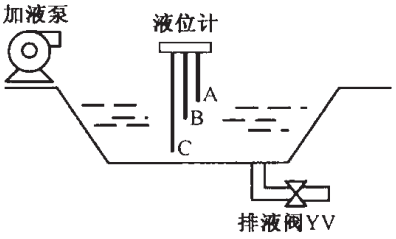


图 3-44 磷化槽液位控制示意图

液位开关断开，并发信号（中间继电器 KA 使能）给输送链控制系统，停止输送链运行。当磷化槽用一段时间后需要换液，此时，按下换液按钮，打开排液阀，并禁止输送链启动。当液位处于 C 液位以下时，延时 3 min 关闭排液阀。

实例实现

根据汽车涂装磷化槽液位控制工作过程，设计控制系统 PLC 接线图和相应的梯形图程序。

（1）磷化槽液位控制系统 PLC 接线图如图 3-45 所示。

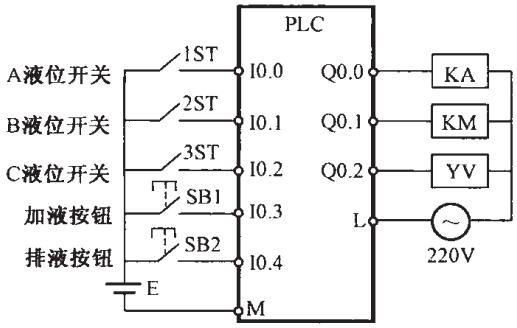


图 3-45　磷化槽液位控制系统 PLC 接线图

（2）磷化槽液位控制系统控制梯形图程序如图 3-46 所示。

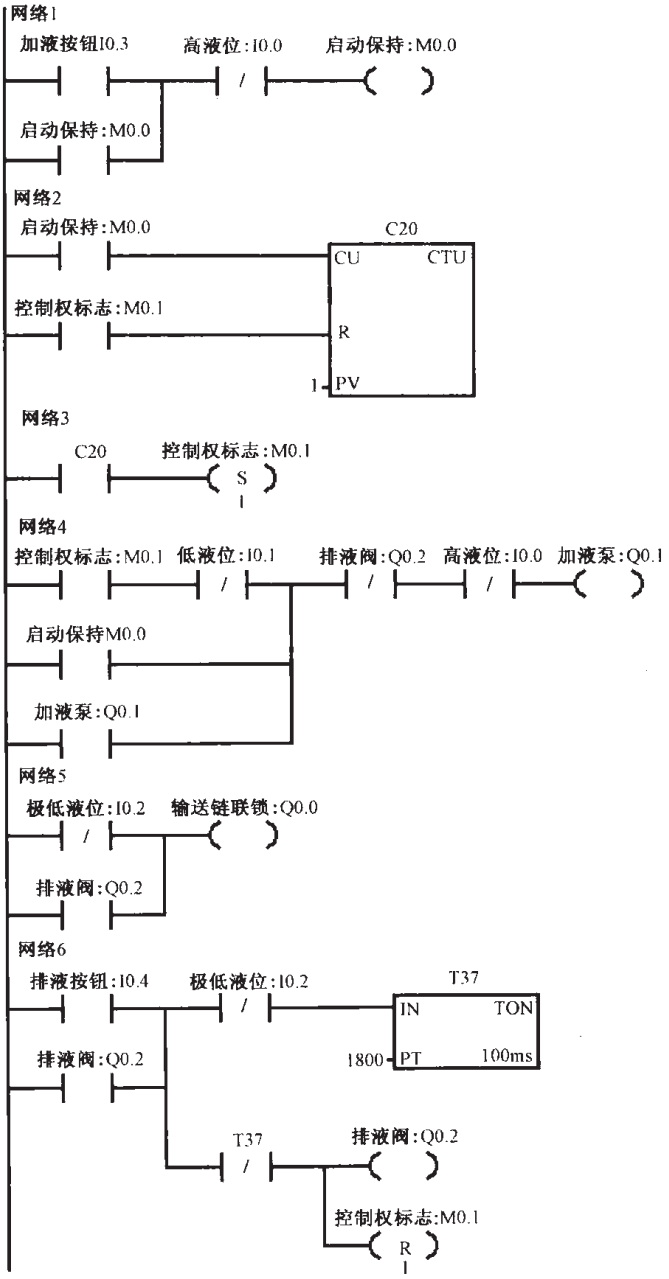


图 3-46　磷化槽液位控制系统梯形图

实例分析

该实例梯形图控制程序原理分析如下：

按下加液按钮 I0.3 后，M0.0 得电并保持，接着 M0.1 被置位为“1”，表示一个工作周期开始，磷化槽没有排液。由于此时液槽是空的，三个液位传感器均没信号输入，其常闭触点均导通，此时 Q0.1 得电，加液泵启动，开始往液槽里加磷化液。开始时，由于液位较浅，还没有到达 C 液位，所以输送链联锁信号 Q0.0 输出，禁止输送链启动。随着液位的增加，B 液位传感器信号 I0.1 到达，输送链联锁信号 Q0.0 断开，允许输送链启动。当液位到达最高位时，A 液位传感器信号 I0.0 到达，其常闭点断开，M0.0 失电，Q0.1 失电，加液泵停止工作。

随着时间推移，液位逐渐下降，当其下降到低于 B 液位时，I0.0、I0.1 信号依次断开，其常闭点接通，由于排液标志 M0.1 没有被复位，仍为“1”，其常开触点闭合，所以 Q0.1 得电，加液泵启动。当液位上升到 A 液位时，I0.0 信号到达，其常闭点断开，Q0.1 失电，加液泵停止工作。如此循环往复，使磷化槽的液位始终控制在 A 液位和 B 液位之间。在系统工作过程中，无论什么原因导致液位低于 C 液位，I0.2 信号断开，其常闭点接通，Q0.0 使能，发出输送链联锁控制信号，禁止输送链运行。

当需要更换磷化液时，按一下排液按钮 SB2，I0.4 常开点接通，Q0.2 使能并保持，使排液阀 YV 打开进行排液；同时将排液标志 M0.1 复位，所以，当液位低于 B 液位时加液泵也不会被启动。当液位降到 C 液位以下时，I0.2 信号断开，其常闭点接通，定时器 T37 开始 3 min 计时，计时结束后定时器的常闭触点断开，Q0.2 失电，排液阀 YV 关闭，排液结束。在排液期间，由于 Q0.2 一直使能，其常闭触点均断开，此时，即使按下加液按钮加液泵也不会被启动。只能等排液结束后，Q0.2 失电，其常闭触点均接通后，加液按钮才能控制加液泵的启动。

实例 63：音乐演奏程序

实例说明

音乐演奏程序运用 S7-200 CPU（DC/DC/DC）的脉冲输出功能演奏 WHISTLE（号笛）音乐。

实例实现

PLC 音乐演奏系统实现需要硬件和程序两大部分。

（1）硬件要求

① SIMATIC S7-200 CPU DC/DC/DC；

② 电源：115 V AC/24 V DC，0.9 A（通常 300 mA～400 mA 就可以）；

③ 扬声器；

④ 430 Ω电阻。

SIMATIC S7-200 的输出端 Q0.0 和 Q0.1 分别串联一只电阻（约 430 Ω，1/2 W）与扬声

器的一端相连，扬声器的另一端与公共输出端（1M）相连。SIMATIC S7-200 的电源输入端（1L+）接+24 V，各自的地线接在公共输出端（1M）。

（2）程序设计

① 程序设计原理：WHISTLE（号笛）音乐程序使用 25 个半音阶音符。程序设计时用音符表，通道 0 的乐曲信息表和通道 1 的乐曲信息表分别存放音符及乐曲信息，如将从“A”（440 Hz）开始的 25 个半音阶音符的音符周期时间与之对应的脉冲数存放在音符表（表 1）中，表 2 和表 3 分别存放通道 0 和 1 的乐曲信息，每个乐曲信息由两个字节音符组成，第一个字节是音符的参考号码（1～25），第二个字节是这个音符的时间单位数目（以 0.125 s 为一个时间单位）。为了使音调能持续 0.125 s 的频率发出，演奏时是两个音符同时进行，需将两个乐曲信息分别用脉冲通道 0 和通道 1 输出，并均被设置成脉冲序列输出（PTO），首先演奏每个通道的第一个音符，紧接着就请求第二个音符。这样就构成了深度为 1 的队列（一个在进程中，一个在队列中）。中断程序附着于 PTO 完成事件，第一个音符演奏完，中断程序调进下一个音符。这个过程继续下去，总是保持深度为 1 的队列，直到乐曲结束。

② 所用 PLC 内存单元：

音乐演奏程序使用了较多的 PLC 内存单元，用于存储音符表、指针及乐曲表通道等，具体分工如下：

V4～V103　　音符表；
V500～V503　指向音符表的指针；
V504～V507　指向通道 0 乐曲表的指针；
V508～V511　临时的工作寄存器；
V554～V557　指向通道 1 乐曲表的指针；
V600～V743　通道 0 乐曲表；
V800～V1059　通道 1 乐曲表。

③ 程序说明：音乐演奏程序由 1 个主程序、4 个子程序和 2 个中断程序组成。

主程序主要用于调用子程序，只在第一个扫描周期中执行，随着主菜单关闭，主程序运行随之结束，其梯形图程序如图 3-47 所示，其梯形图对应的语句如下：

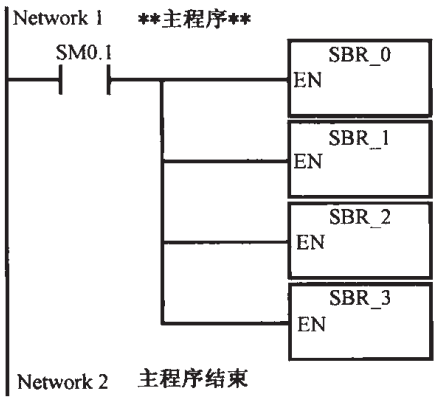


图 3-47　音乐演奏梯形图程序（主程序）

```
***主程序***
LD      First_Scan_On:SM0.1          // 第一次扫描 SM0.1=1
CALL    SBR_0:SBR0                   // 初始化音符表
CALL    SBR_1:SBR1                   // 初始化通道 0 乐曲表
CALL    SBR_2:SBR2                   // 初始化通道 1 乐曲表
CALL    SBR_3:SBR3                   // 初始化脉冲序列.
```

子程序 0 初始化程序所使用的音符，用 MOVD 指令把用十六进制数表示的音符存于 S7-200 的内存中，音符的前 4 个字符码表征音符的频率，后 4 个字符码表示持续音调 0.125 s 所需的脉冲数。此外还初始化脉冲序列输出，用特殊标志字节 SMB67 定义输出端 Q0.0 输出的方波特性，用 SMB77 定义输出端 Q0.1 输出的方波特性，其梯形图程序如图 3-48 所示，其

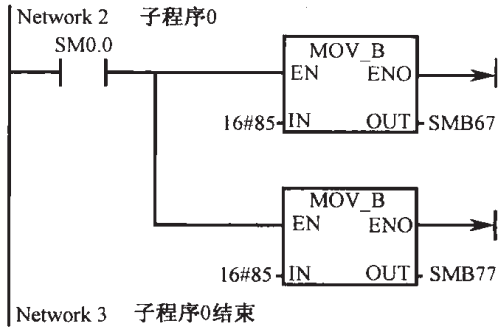


图 3-48 音乐演奏梯形图程序（子程序 0）

梯形图对应的语句如下：

```
***子程序 0***
LD      Always_On:SM0.0
MOVB    16#85, PLS0_Ctrl:SMB67   // 设置脉冲输出 0(PLS 0)的控制字
MOVB    16#85, PLS1_Ctrl:SMB77   // 设置脉冲输出 1(PLS 1)的控制字
RET
```

子程序 1 初始化通道 0 的乐曲表，演奏每个通道的第一个音符，通道 0 的音符参考号码和与之匹配的时间单位数被装入脉冲输出 0（PLS 0），同样的过程在通道 1 中进行（PLS 1），因此能同时演奏两个音符，其梯形图程序如图 3-49 所示，其梯形图对应的语句如下；

```
***子程序 1***
Network2
LD      Always_On:SM0.0              // Load SM0.0.
MOVD    &VB4, VD1500                 // 设置指向音符表的指针
MOVD    &VB104, VD1504               // 设置指向通道 0 的乐曲表的指针
MOVB    *VD1504, VB1503              // 取出通道 0 的第一个音符
SLW     VW1502, 2                    // 指向音符表(*4)
MOVW    *VD1500, PLS0_Cycle:SMW68    // 至通道 0 输出脉冲的周期时间
INCD    VD1500                       // 指针指向每单位时间的计数
INCD    VD1500                       // Another Pointer
INCD    VD1504                       // 指针指向时间单位数
MOVD    +0, VD1508                   // 清除内存 V508
MOVB    *VD1504, VB1511              // 取出音符的时间单位数
MUL     *VD1500, VD1508              // 计算音符的总计数值
MOVD    VD1508, PTO0_PC:SMD72        // 置音符的计数值
MOVD    &VB4, VD1500                 // 指针指向音符表
Network3
LD      Always_On:SM0.0              // Load SM0.0.
MOVD    &VB248, VD1554               // 指针指向通道 1 的乐曲表
MOVB    *VD1554, VB1503              // 取出通道 1 的第一个音符
SLW     VW1502, 2                    // 指向音符表(*4)
MOVW    *VD1500, PLS1_Cycle:SMW78    // 置通道 1 输出脉冲的周期时间
INCD    VD1500                       // 指针指向每单位时间的计数
INCD    VD1500                       //
INCD    VD1554                       // 指针指向时间单位数
MOVD    +0, VD1508                   // 清除内存 V508
MOVB    *VD1554, VB1511              // 取出音符的时间单位数
MUL     *VD1500, VD1508              // 计算音符的总计数值
MOVD    VD1508, PTO1_PC:SMD82        // 置音符的计数值
PLS     0                            // 演奏每个通道的第一个音符
```

```
PLS    1                    //
RET
```

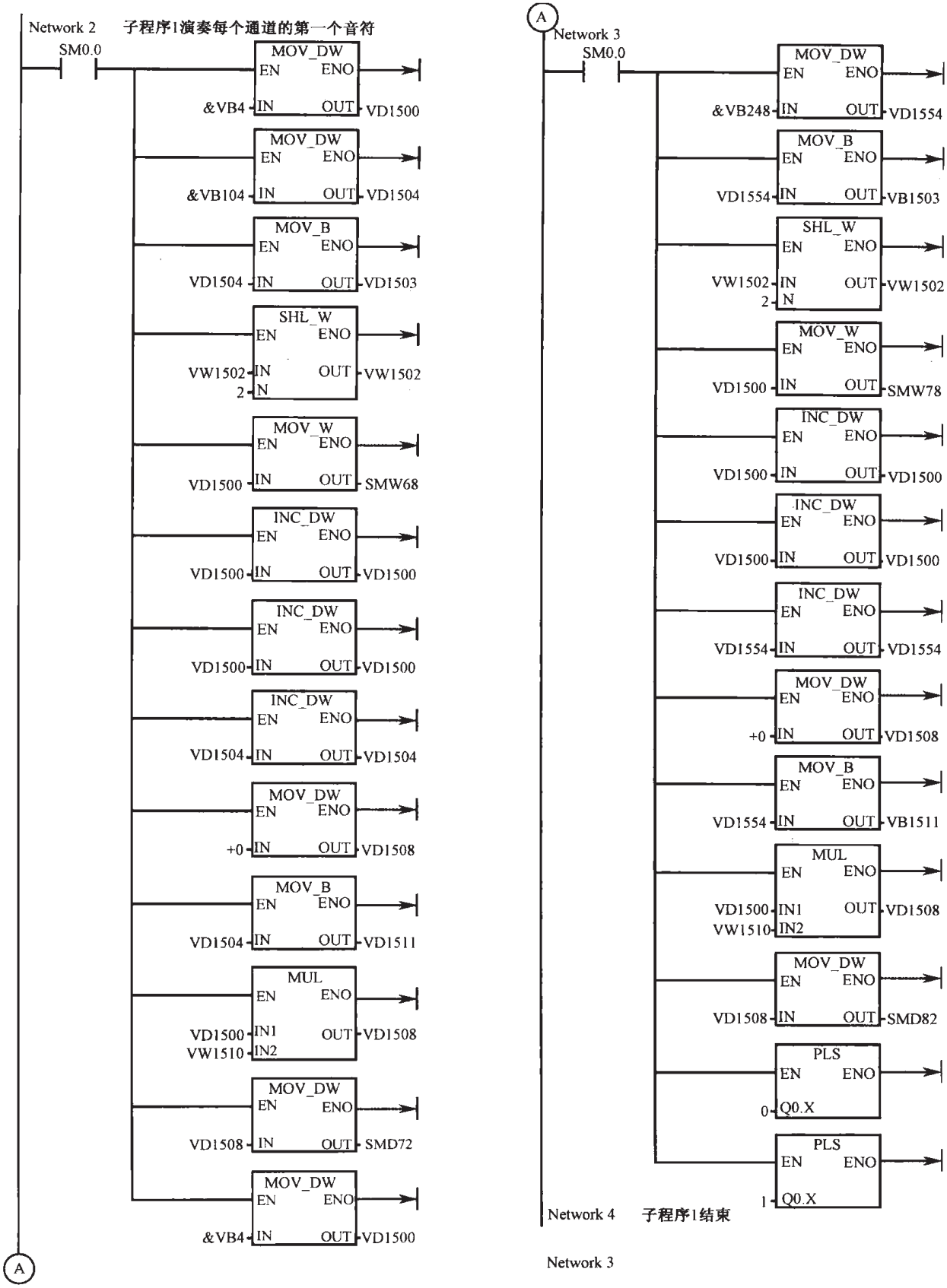


图 3-49　音乐演奏梯形图程序（子程序 1）

子程序 2 初始化通道 1 的乐曲表，把每个通道的第二个音符排入队列，通道 0 的音符参考号码和与之匹配的时间单位数被装入脉冲输出 0（PLS 0），同样的过程也在通道 1 中进行（PLS 1），因此能同时演奏两个音符，梯形图程序如图 3-50 所示，梯形图对应的语句如下；

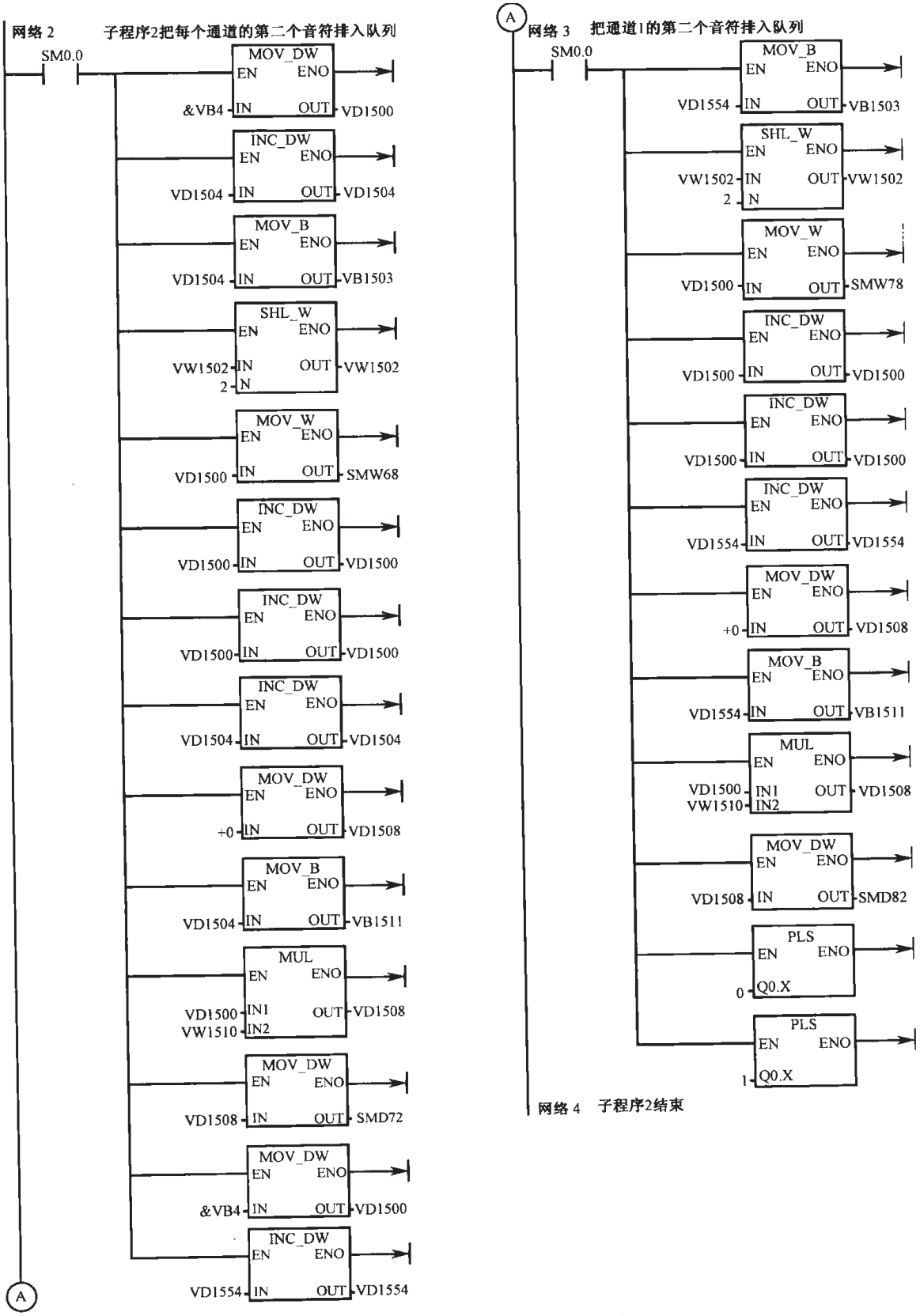


图 3-50 音乐演奏梯形图程序（子程序 2）

```
***子程序 2***
Network2
LD      Always_On:SM0.0                 // Load SM0.0.
MOVD    &VB4, VD1500                    // 指针指向音符表
INCD    VD1504                          // 指针指向通道 0 的第二个音符
MOVB    *VD1504, VB1503                 // 取出通道 0 的第二个音符
SLW     VW1502, 2                       // 指向音符表（乘以 4）
MOVW    *VD1500, PLS0_Cycle:SMW68       // 置通道 0 输出脉冲的周期时间
INCD    VD1500                          // 指针指向每单位时间的计数
INCD    VD1500                          //
INCD    VD1504                          // 指针指向时间单位数
MOVD    +0, VD1508                      // 清除内存 V508
MOVB    *VD1504, VB1511                 // 取出音符的时间单位数
MUL     *VD1500, VD1508                 // 计算音符的总计数值
MOVD    VD1508, PTO0_PC:SMD72           // 置音符的计数值
MOVD    &VB4, VD1500                    // 指针指向音符表
Network3
INCD    VD1554                          // 指针指向通道 1 的第二个音符
LD      Always_On:SM0.0                 // Load SM0.0.
MOVB    *VD1554, VB1503                 // 取出通道 1 的第二个音符
SLW     VW1502, 2                       // 指向音乐表（乘以 4）
MOVW    *VD1500, PLS1_Cycle:SMW78       // 置通道 1 输出脉冲的周期时间
INCD    VD1500                          // 指针指向每单位时间的计数
INCD    VD1500                          // Another pointer
INCD    VD1554                          // 指针指向时间单位数
MOVD    +0, VD1508                      // 清除内存
MOVB    *VD1554, VB1511                 // 取出音符的时间单位数
MUL     *VD1500, VD1508                 // 计算音符的总计数值
MOVD    VD1508, PTO1_PC:SMD82           // 置音符的计数值
PLS     0                               // 把每个通道的第二个音符排列入队
PLS     1                               //
RET
```

子程序 3 启动中断程序，实现音乐的连续演奏；其梯形图程序如图 3-51 所示，其梯形图对应的语句如下：

```
***子程序 3***
LD      Always_On:SM0.0
ATCH    INT_0:INT0, 19      // 把中断程序 0 附着于 PLS 0 完成的事件（事件 19）
ATCH    INT_1:INT1, 20      // 把中断程序 0 附着于 PLS 1 完成的事件（事件 20）
ENI                         // 允许中断
RET
```

中断程序 0 演奏通道 0 的下一个音符，是脉冲输出 0（PLS 0）的脉冲计数，其中断事件 19，梯形图程序如图 3-52 所示，其梯形图对应的语句如下：

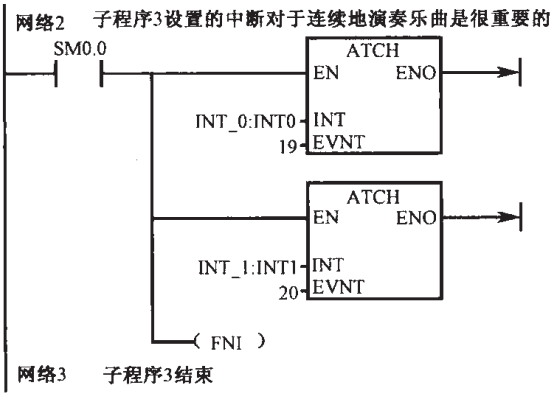


图 3-51 音乐演奏梯形图程序（子程序 3）

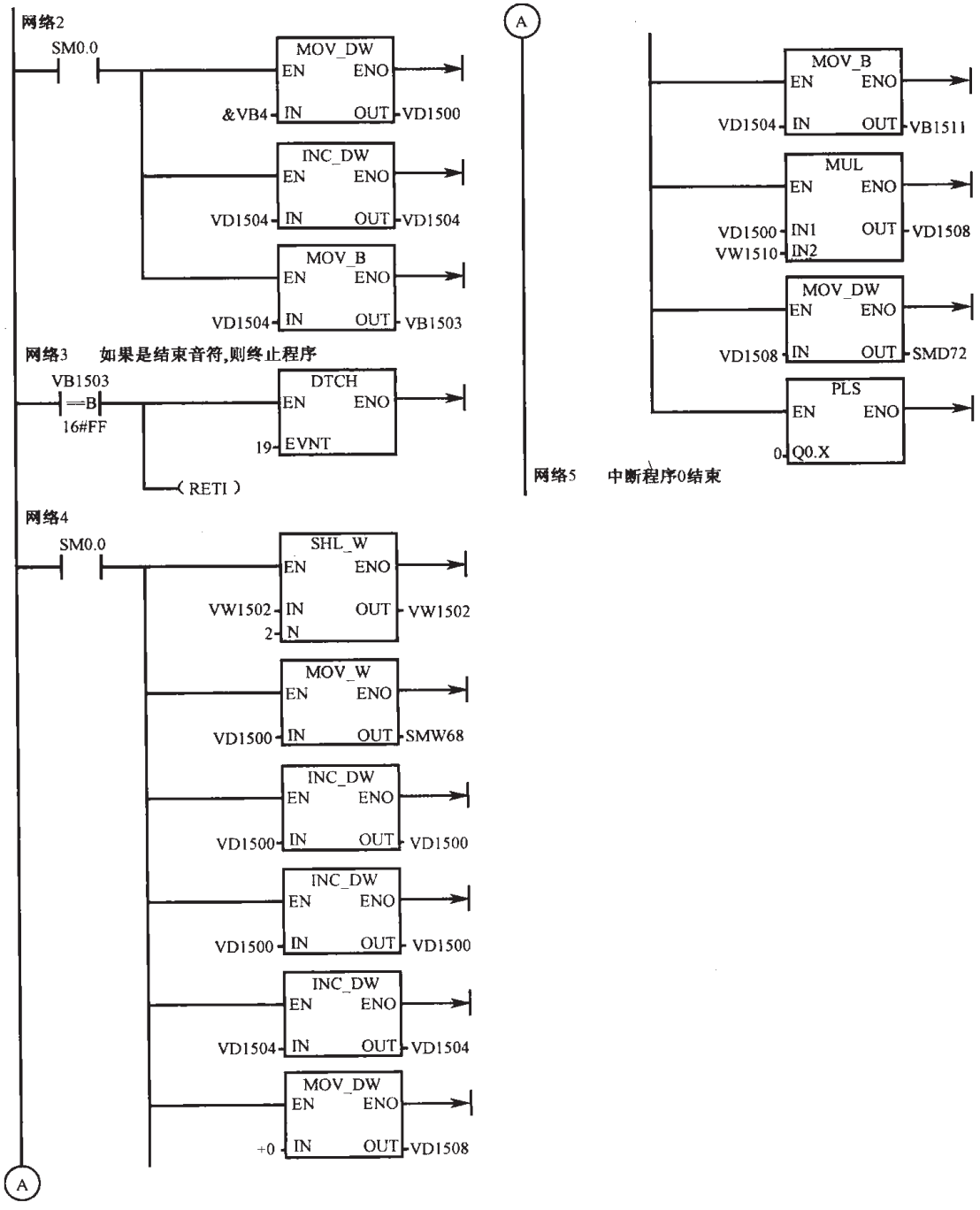


图 3-52 音乐演奏梯形图程序（中断程序 0）

```
***中断程序 0***
Network1
LD      Always_On:SM0.0                 // Load SM0.0.
MOVD    &VB4, VD1500                    // 指针指向音符表
INCD    VD1504                          // 指针指向通道 0 的下一个音符
MOVB    *VD1504, VB1503                 // 取出通道 0 的下一个音符
Network2 如果是结束音符，则终止程序
LDB=    VB1503, 16#FF
DTCH    19
CRETI
Network3
LD      Always_On:SM0.0                 // Load SM0.0.
SLW     VW1502, 2                       // 指向音符表(乘以 4)
MOVW    *VD1500, PLS0_Cycle:SMW68       // 置通道 0 输出脉冲的周期时间
INCD    VD1500                          // 指针指向每单位时间的计数
INCD    VD1500
INCD    VD1504                          // 指针指向时间单位数
MOVD    +0, VD1508                      // 清除内存 V508
MOVB    *VD1504, VB1511                 // 取出音符的时间单位数
MUL     *VD1500, VD1508                 // 计算音符的总计数值
MOVD    VD1508, PTO0_PC:SMD72           // 至音符的计数值
PLS     0                               // 请求通道 0 的下一个音符
RET
```

中断程序 1 是脉冲输出 1（PLS 1）的脉冲计数，其中断事件 20，演奏完一个音符后中断发生，其梯形图程序如图 3-53 所示，其梯形图对应的语句如下：

```
***中断程序 1***
Network1
LD      Always_On:SM0.0                 //
MOVD    &VB4, VD1500                    // 指针指向音符表
INCD    VD1554                          // 指针指向通道 1 的下一个音符
MOVB    *VD1554, VB1503                 // 取出通道 1 的下一个音符
Network2 如果是结束音符，则终止程序。
LDB=    VB1503, 16#FF
DTCH    20
CRETI
Network3
LD      Always_On:SM0.0
SLW     VW1502, 2                       // 指向音符表(乘以 4)
MOVW    *VD1500, PLS1_Cycle:SMW78       // 置通道 1 输出脉冲的周期时间
INCD    VD1500                          // 指针指向每单位时间的计数
INCD    VD1500                          // Another pointer
INCD    VD1554                          // 指针指向时间单位数
MOVD    +0, VD1508                      // 清除内存 V508
```

```
MOVB    *VD1554, VB1511          // 取出音符的时间单位数
MUL     *VD1500, VD1508          // 计算音符的总计数值
MOVD    VD1508, PTO1_PC:SMD82    // 置音符的计数值
PLS     1                        // 请求通道 1 的下一个音符
```

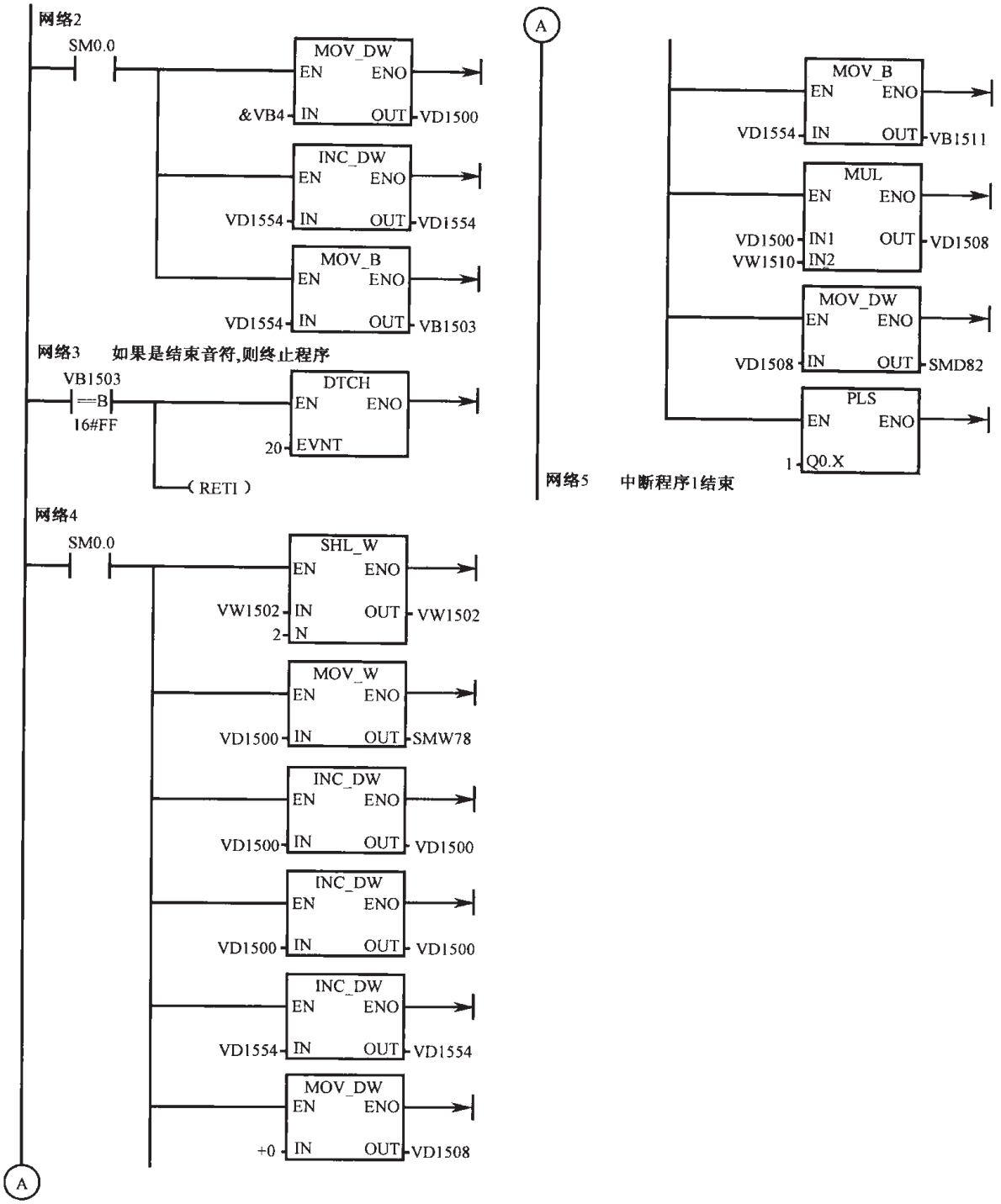


图 3-53 音乐演奏梯形图程序（中断程序 1）

实例分析

该应用实例展示了利用 S7-200 DC/DC/DC 脉冲输出功能演奏音乐。程序设计思路大致

是：首先，建立 25 个音符周期的音符表并初始化音符表；其次，初始化通道 0 和 1 乐曲表，然后初始化脉冲序列；最后，用队列中断程序附着于脉冲序列输出（PTO）完成事件，完成音符连续演奏。

思考题

1. 简述 S7-200 PLC 的主程序、子程序和中断程序间的相互关系。

2. 简述 S7-200 PLC 编程规则。

3. 试用定时器、计数器与比较指令、移位指令编制 M1、M2 和 M3 三台电动机顺序启动和停止控制的梯形图程序。

4. 编制抢答器程序。控制要求是：3 位参赛者的抢答按钮对应的 PLC 接线端子分别为 I0.0、I0.1 和 I0.2，相应信号灯分别为 Q0.0、Q0.1 和 Q0.2，主持人的启动按钮接线端子为 I0.3。当主持人读完试题，按下启动按钮后，3 位参赛者可抢答，最早按下抢答按钮的参赛者信号灯被点亮，其他参赛者灯不能被点亮。

5. 编制洗衣机清洗控制程序。控制要求是：当按下启动按钮对应的 PLC 接线端子 I0.0 后，电动机先正转 2 s，停 2 s，然后反转 2 s，停 2 s，如此重复 5 次，自动停止清洗。当按下停止按钮 I0.1 后，停止清洗。

6.简单的位置控制。控制要求：①用多齿凸轮与电动机联动，并用接近开关来检测多齿凸轮，产生的脉冲输入至 PLC 的计数器。②电动机转动至 4900 个脉冲时，使电动机减速，到 5000 个脉冲时，使电动机停止，同时剪切机动作将材料切断，并使脉冲计数复位。

输入/输出端子分配如下：

输　入		输　出	
元件名称	端子号	元件名称	端子号
启动按钮	I0.0	电极高速运转	Q0.0
停止按钮	I0.1	电极低速运转	Q0.1
接近开关	I0.2	剪切机	Q0.2
剪切结束	I0.3		

通过调用子程序 0 来对 HSC1 进行编程，设置 HSC1 以方式 11 工作，其控制字（SMB47）设为 16#F8；预设值（SMD52）为 50。当计数值完成（中断事件编号 13）时通过中断服务程序 0 写入新的当前值（SMD50）16#C8。

第 4 章 PLC 扩展系统

- S7-200 PLC 的系统配置
- 数字量扩展模块
- 模拟量扩展模块
- 位控模块
- PID 算法原理及指令介绍

随着控制系统的规模和复杂程度的增加，对 PLC 控制器的要求会越来越高，通常情况下单一的 CPU 模块难以胜任。这就需要针对具体的功能需求对 CPU 模块进行扩展。西门子 S7-200 PLC 属于紧凑型可编程控制器，但它具有强大的、灵活的扩展能力。可用的扩展模块包括 I/O 扩展模块和智能扩展模块。其中 I/O 扩展模块有数字量模块和模拟量模块；智能扩展模块有温度测量模块、位控模块、网络通信模块等扩展模块。本章结合实例，针对常用的数字量扩展模块、模拟量扩展模块、温度测量模块和位控模块及 PID 功能指令进行介绍。

4.1 S7-200 PLC 的系统配置

S7-200 任一型号的主机，都可单独构成一个独立的控制系统，其 I/O 配置是固定的，具有固定的 I/O 地址。为了实现更强的控制功能可以采用主机带扩展模块的方法扩展 S7-200 的系统配置。采用数字量模块或模拟量模块可扩展系统的控制规模，采用智能模块可扩展系统的控制功能。在利用扩展模块对 S7-200 主机进行扩展时要考虑以下问题：

（1）主机所允许的最大扩展模块的数量

S7-200 PLC 主机允许扩展模块的数量各不相同，其中 CPU221 主机不允许扩展，只能独立构成控制系统；CPU222 模块最多可带两块扩展模块；CPU224、CPU226 和 CPU226XM 最多可带 7 个扩展模块，且 7 个模块中最多只能带两个智能模块。

（2）CPU 输入/输出映像区的限制

S7-200 各类主机提供的数字量 I/O 映像区区域为 128 个输入映像寄存器（10.0～115.7）和 128 个输出映像寄存器（QO.0～Q15.7）。模拟量 I/O 映像区区域为 CPU222 模块为 16 入 116 出；CPU224 模块、CPU226 模块、CPU226XM 模块为 32 入 132 出，模拟量扩展模块总是以两个字节递增方式来分配空间。最大 I/O 配置不能超出此区域。

（3）内部电源的负载能力

CPU 模块和扩展模块正常工作时，需要 DC+5 V 工作电源。S7-200 内部电源单元提供的 DC+5 V 电源为 CPU 模块和扩展模块提供了工作电源。其中扩展模块所需的 DC+5 V 工作电源是由 CPU 模块通过总线连接器提供的。在配置扩展模块时，要保证各扩展模块消耗 DC+5 V 电源的电流总和不超过 CPU 模块所提供的电流值。S7-200 主机的内部电源单元还提供 DC+24 V 传感器电源。一般情况下，CPU 模块和扩展模块的输入、输出端子所用的 DC+24 V 电源是由用户外部提供。如果使用 CPU 模块内部的 DC+24 V 电源的话，应保证 CPU 模块及各扩展模块所消耗电流的总和不超过该内部 DC+24 V 电源所提供的最大电流。因此，系统配置后必须对主机内部 DC+5 V 和 DC+24 V 电源负载能力进行校验。

4.2 数字量扩展模块

数字量输入模块的每一个输入端子可接收一个来自用户设备的 ON/OFF 信号，如按钮、选择开关、限位开关、继电器或接触器的触点等。每个输入端子只能接一个外部输入信号。同样地，每个输出端子只能接一个 ON/OFF 型负载，如继电器线圈、接触器线圈、指示灯等。同一类型的输入/输出设备才能共用一个公共点，不可混用。

西门子 S7-200 PLC 的数字量扩展模块有数字量输入模块（EM221）、数字量输出模块

（EM222）和数字量混合模块（EM223）。每种模块又有不同的类型以更好地满足实际应用需要。各模块的类型如表 4-1 所示。由于各模块的接线方法比较简单，这里不再赘述，读者可参考西门子产品选型手册。

表 4-1 数字量扩展模块类型

名 称	类 型	规 格	电 源 要 求
EM221	输入	8DI—24 V DC	每通道输入电流为 4 mA，支持源型和漏型输入，但同一公共点输入类型必须一致。需要 5V DC 电源提供 30 mA 电流
	输入	8DI—120/230 VAC	120/230V AC 每通道电流为 6 mA/9 mA。需要 5 V DC 电源提供 30 mA 电流
	输入	16DI—24 V DC	每通道输入电流 4 mA，支持源型和漏型输入，但同一公共点输入类型必须一致。需要 5 V DC 电源提供 70 mA 电流
EM222	输出	8D—24 V DC	每路通道最大输出电流 0.75 A，每个公共点最大电流 10 A。需要 5 V DC 电源提供 50 mA 电流
	输出	8DO—继电器	每路通道最大输出电流 2 A，每个公共点最大电流 1 A。需要 5 V DC 电源提供 40 mA 电流
	输出	8DO—120/230 V AC	每路通道最大电流为 0.5 A AC，每个公共点最大电流为 0.5 A AC。需要 5V DC 电源提供 110 mA 电流
	输出	4DO—24 V DC	每路通道最大输出电流 0.75 A，每个公共点最大电流 10 A。需要 5 V DC 电源提供 40 mA 电流
	输出	4DO—继电器	每路通道最大输出电流 2 A，每个公共点最大电流 1 A。需要 5 V DC 电源提供 30 mA 电流
EM223	输入/输出	4DI/4D0—24 V DC	与 EM221/EM222 同类型的相同
	输入/输出	4DI/4D0—继电器	与 EM221/EM222 同类型的相同
	输入/输出	8DI/8D0—24 V DC	与 EM221/EM222 同类型的相同
	输入/输出	8DI/8D0—继电器	与 EM221/EM222 同类型的相同
	输入/输出	16DI/16DO—24 V DC	与 EM221/EM222 同类型的相同
	输入/输出	16DI/16D0—继电器	与 EM221/EM222 同类型的相同

PLC 系统配置时，要对各类输入、输出模块的输入、输出端子进行编址。主机提供的 I/O 具有固定的 I/O 地址。扩展模块的地址由 I/O 模块类型及模块在 I/O 链中的位置决定。编址时，按同类型的模块对各输入端子（或输出端子）顺序编址。数字量输入、输出映像区的逻辑空间是以 8 位（1 个字节）为递增的，编址时，对数字量模块物理点的分配也是按 8 点来分配地址的，即使有些模块的端子数不是 8 的整数倍，但仍以 8 点来分配地址。例如，4 入/4 出模块也占用 8 个输入端子和 8 个输出端子的地址，那些未用的物理点地址不能分配给 I/O 链中的后续模块，那些与未用物理点相对应的 I/O 映像区的空间就会丢失。对于输出模块，这些丢失的空间可用来作内部存储器标志位。对于输入模块却不可，因为每次输入更新时，CPU 都对这些空间清零。

实例 64：数字量扩展模块的 I/O 编址

实例说明

某一控制系统选用 CPU224，系统需要 24 个+24V DC 输入端子，20 个+24V DC 输出端子。

实例实现

在对该控制系统进行组态时要注意到 CPU 模块本身所带的 I/O 端子数。本例中 CPU224 本身带有 14 个输入端子和 10 个输出端子，因此需要扩展 10 个输入端子和 10 个输出端子。所以扩展模块可以选择 EM221 8DI－24V DC 模块 1 块，EM222 8DO－24V DC 模块 1 块和 EM223 4 输入/4 输出模块 1 块。组态上从左至右依次为 CPU224、EM221、EM222 和 EM223。各模块 I/O 端口对应的地址如表 4-2 所示。

表 4-2 PLC 各模块 I/O 编址

CPU224		EM221	EM222	EM223	
I0.0	Q0.0				
I0.1	Q0.1				
I0.2	Q0.2				
I0.3	Q0.3	I2.0	Q2.0		
I0.4	Q0.4	I2.1	Q2.1		
I0.5	Q0.5	I2.2	Q2.2	I3.0	Q3.0
I0.6	Q0.6	I2.3	Q2.3	I3.1	Q3.1
I0.7	Q0.7	I2.4	Q2.4	I3.2	Q3.2
I1.0	Q1.0	I2.5	Q2.5	I3.3	Q3.3
I1.1	Q1.1	I2.6	Q2.6		
I1.2		I2.7	Q2.7		
I1.3					
I1.4					
I1.5					

实例分析

多数情况下仅主机所带的 I/O 端口不能满足实际控制功能的需要，常常要进行 I/O 端口扩展和编址。在应用中可根据实际端口数量将本例进行适当拓展，可以很方便地实现对 I/O 端口的正确编址。

4.3 模拟量扩展模块

在工业控制系统中，除了要处理大量的数字量信号外，还经常要对一些模拟量进行处理，如温度、湿度、压力、流量等。这些物理量需要借助于相应传感器或变送器把相关的物理量转换成标准的电信号后传送给 PLC 进行处理。另外，有些现场设备需要用模拟量信号进行控制，如一些电动阀门、伺服电磁阀等。这就需要将 PLC 输出的数字量变换成模拟量信号，以满足控制设备的要求。模拟量扩展模块正是为完成这一任务而设计的。S7-200 PLC 相应的模拟量处理模块有模拟量输入模块 EM231、模拟量输出模块 EM232 和模拟量输入/输出混合模块 EM235。

4.3.1 模拟量输入模块 EM231

模拟量输入模块是把来自现场设备的标准电信号（如 4～20 mA 的电流信号），经过滤波去掉干扰信号后，再通过 A/D 转换，将模拟量信号变换成 PLC 能够处理的数字信号，然后

经过光电耦合器隔离后传送给 PLC 内部电路，供 PLC CPU 处理。这一过程如图 4-1 所示。

模拟量输入模块（EM231）具有 4 路或 8 路模拟量输入通道，每个通道占用存储器 AI 区域 2 个字节。该模块模拟量的输入值为只读数据。电压输入范围为单极性 0～10 V 或 0～5 V，双极性–5～+5 V 或–2.5～+2.5 V；电流输入范围为 0～20 mA。模拟量到数字量的最大转换时间为 250 μs。该模块需要直流 24 V 供电。可由 CPU 模块的传感器电源 DC 24 V/400 mA 供电，也可由用户提供外部电源供电。以 4 通道 EM231 模块为例，该模块的外部接线图如图 4-2 所示。

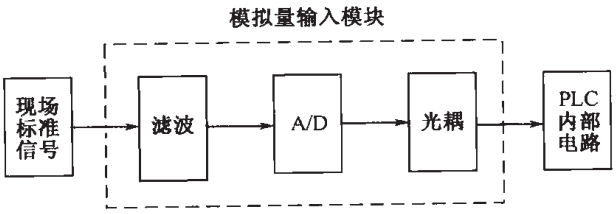


图 4-1　模拟量输入信号处理过程

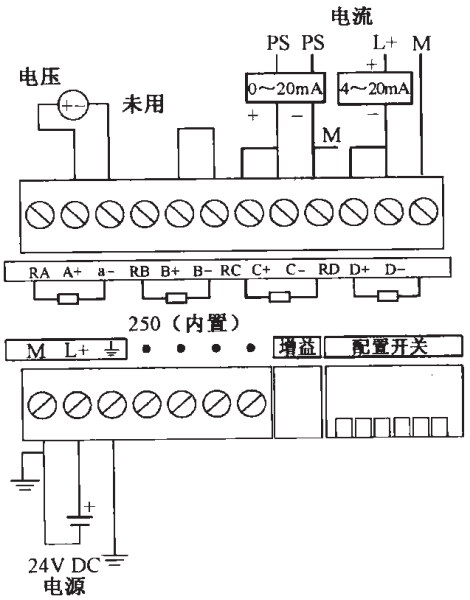


图 4-2　EM231 模拟量处理模块接线图

模块上部共有 12 个端子，每 3 个点为 1 组（例 RA、A+、A–）可作为一路模拟量输入通道，共 4 组，对于电压信号只用两个端子（如图 4-2 中的 A+、A–），电流信号要用 3 个端子（如图 4-2 中的 RC、C+、C–，其中 RC 与 C+端子短接）。未用的通道应将其短接。模块左下部的 M、L+端接入 DC 24V 电源。右端与之相邻的分别是校准电位器和组态开关（DIP）。需要注意的是为避免共模电压，需将 M 端与所有信号负端连接。

要使模拟量输入模块正常工作还需要通过组态开关（DIP）对其测量量程进行设置。表 4-3、表 4-4 分别是 4 通道和 8 通道 EM231 模块的组态开关表，其中“ON”表示闭合，“OFF”表示断开。组态设置只在电源接通时才能生效。因此，改变组态后要给模块重新上电才能使新的组态生效。

表 4-3　4 通道 EM231 模块组态开关表

单　极　性			满量程输入	分　辨　率
SW1	SW2	SW3		
ON	OFF	ON	0～10 V	2.5 mV
	ON	OFF	0～5 V	1.25 mV
			0～20 mA	5 μA
双　极　性			满量程输入	分　辨　率
SW1	SW2	SW3		
OFF	OFF	ON	±5 V	2.5 mV
	ON	OFF	±2.5 V	1.25 mA

表 4-4 8 通道 EM231 模块组态开关表

单极性			满量程输入	分辨率
SW3	SW4	SW5		
ON	OFF	ON	0～10 V	2.5 mV
	ON	OFF	0～5 V	1.25 mV
			0～20 mA	5 μA
双极性			满量程输入	分辨率
SW3	SW4	SW5		
OFF	OFF	ON	±5 V	2.5 mV
	ON	OFF	±2.5 V	1.25 mA

模拟量输入模块的输入信号经 A/D 转换后的二进制数在 CPU 中的存放格式如图 4-3 所示。模拟量转换到数字量的 12 位二进制数是左对齐的。最高有效位是符号位："0" 表示是正值，"1" 表示是负值。在单极性格式中，3 个连续的 "0" 使得 ADC 计数值每变化 1 个单位，数据字中则以 8 为单位变化。在双极性格式中，4 个连续的 "0" 使得 ADC 计数值每变化 1 个单位，数据字则以 16 为单位变化。单极性数据字格式的全量程范围设置为 0～32000，而双极性数据字格式的全量程范围设置为–32000～+32000。

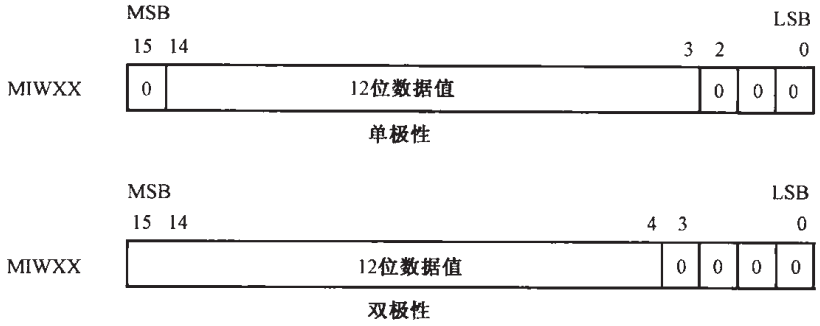


图 4-3 EM231 模块输入数据字格式

PLC 系统配置系统中对 EM231 进行编址时，从左至右依次从 AIW0 开始以 2 个字节递增编址，如 AI0，AI2，AI4，AI6……。要注意的是最大地址不能超过 CPU 模块所支持的最大范围。另外，当一个模块编址结束，对该模块右边最近的模块编址时其地址应与它前边模块最后一个通道地址相连续。

4.3.2 热电偶、热电阻扩展模块 EM231

在工业实际应用，尤其是过程控制系统中经常要涉及对温度的测量。常用的测温传感器有热电偶型和热电阻（RTD）型两种。为了应用方便西门子公司专门开发了测温度模块——EM231 热电偶、热电阻扩展模块。

（1）热电偶扩展模块

EM231 热电偶模块共有两种型号：EM231 4 路热电偶模块和 EM231 8 路热电偶模块。可以连接 J、K、E、N、S、T 和 R 共 7 种类型的热电偶。测量范围为±80 mV，模块响应时间为 405 ms。该模块需要直流 24 V 供电。可由 CPU 模块的传感器电源 DC24 V 供电，也可由用户提供外部电源供电。如图 4-4 所示为 4 通道热电偶模块的接线图。模块上部共有 12 个端子，从左边开始 8 个端子每 2 个为一组接热电偶传感器。上部右边为 4 个接地端子，分别

接 4 个传感器的屏蔽线。为了图示清晰，左边两个传感器的屏蔽线没有画出。下边最左边 3 个端子接电源信号。中间 4 个端子也是接地端子，在这里不用。右边为组态 DIP 开关，该组开关共有 8 个开关，其中前 3 个用以选择连接热电偶的类型，第 4 个必须至“0”（开关置于下边），开关 5 和开关 6 用于设定传感器断线检测方向和检测使能设置。开关 7 用于设定测量温度单位是摄氏度还是华氏度。开关 8 用于设置冷端补偿功能的开启/关闭。要使 DIP 开关起作用，需要给模块重新上电。需要注意的是，一个测量模块所接热电偶必须为同一类型。如果热电偶的输入通道未使用，应将其短接或将其并行连接到其他通道上。为了更好地抑制噪声最好使用屏蔽线进行接线。

（2）热电阻扩展模块

EM231 RTD 模块也有两种型号：EM231 2 路 RTD 模块和 EM231 4 路 RTD 模块。该模块可以连接多种不同型号的电阻，但同一个模块连接的电阻必须为同一类型。模块响应时间为 405 ms，若输入为 P_t10 000 响应时间为 700 ms。该模块需要直流 24 V 供电。可由 CPU 模块的传感器电源 DC24V 供电，也可由用户提供外部电源供电。如图 4-5 所示为 2 通道热电阻模块的接线图。模块上部共有 12 个端子，从左边开始 8 个端子每 4 个为一组接热电阻传感器。上部右边为 4 个接地端子，可以接 2 个传感器的屏蔽线。下部最左边 3 个端子接电源信号。中间 4 个端子也是接地端子，在这里不用。右边为组态 DIP 开关，该组开关共有 8 个开关，其中前 5 个用以选择连接热电阻的类型，开关 6 用于设定传感器断线检测或超出范围的正负极。开关 7 用于设定测量温度单位是摄氏度还是华氏度。开关 8 用于设置热电阻的接线方式。RTD 模块与传感器的接线方式有 2 线、3 线和 4 线 3 种接线方式。其中 4 线接方式精度最高。2 线接线精度最低，只可用于可忽略接线误差的应用场合。要使 DIP 开关起作用，需要给模块重新上电。如果热电阻的输入通道未使用，应连接一个阻值与 RTD 标称值相等的电阻。

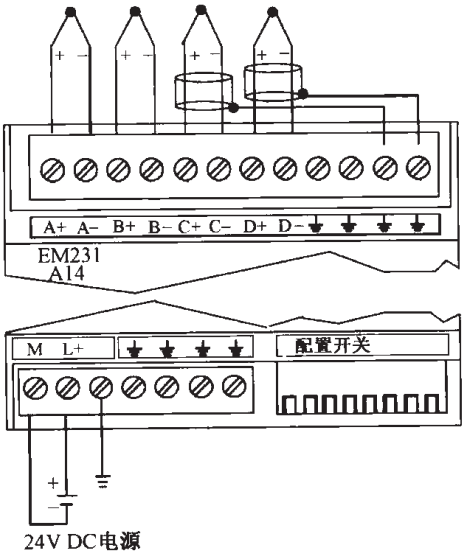


图 4-4　4 通道热电偶模块的接线图

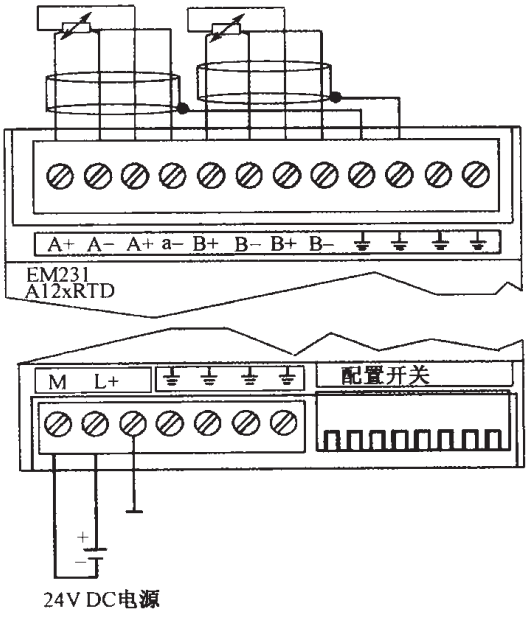


图 4-5　2 通道热电阻模块的接线图

热电偶扩展模块和热电阻模块通道数据格式均为 2 的补码，用字（16 位二进制数）来进行数据存储。数值单位为 0.1 度，如测量数据是 1002，则测量的温度为 100.2° 。热电偶扩展模块电压数据标定到 27648。例如，–60.0 mV 则报告为 20736（=–60 mV/80 mV*27648）。如

PLC 已读取到数据，则每 405 ms 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。所以为了保持通道数据总是为当前值，PLC 读数据的频度至少和模块更新频率相同。

4.3.3 模拟量输出模块 EM232

模拟量输出模块是把 PLC 输出的数字量经光电耦合器后，再经过 A/D 转换器后，将数字信号转换成模拟信号，经过运算放大器后驱动输出。该过程如图 4-6 所示。光电耦合器主要是将内外电路隔开，防止外部电磁干扰信号对 PLC 内部电路造成干扰。通常模拟量输出模块提供电压和电流输出。

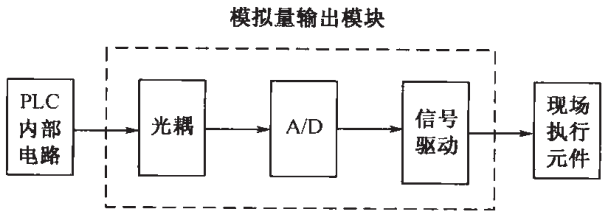


图 4-6 模拟量输出信号处理过程

模拟量输出模块（EM232）具有 2 路或 4 路模拟量输出通道，每个输出通道占用存储器 AQ 区域 2 字节。该模块输出的模拟量可以是电压信号也可以是电流信号。输出信号的范围是电压输出为–10 V～+10 V，电流输出为 0～20 mA。电压输出的设定时间为 100 μs，电流输出的设置时间为 2 ms。用户程序不能读取模拟量的输出值。该模块需要直流 24 V 供电。可由 CPU 模块的传感器电源 DC24 V/400 mA 供电，也可由用户提供外部电源供电。以 2 通道 EM232 模块为例，该模块的外部接线图如图 4-7 所示。模块上部有 7 个端子，左起每 3 个是 1 组，为 1 路模拟量输出，共两组。第 7 个端子空闲。在每一组中 V0 和 V1 端子接电压负载，I0 和 I1 端子接电流负载，M0 和 M1 端子则接为各组的公共端。输出模块下部 M、L+两端接+24 V DC 供电电源。

PLC CPU 模块处理后的 12 位数字输出格式如图 4-8 所示。数字量到模拟量转换器（DAC）的 12 位读数在其输出数据格式中是左对齐的。最高有效位是符号位：“0”表示正值，“1”表示负值。数据在装载到 DAC 寄存器之前 4 个连续的“0”是被截断的，这些位不影响输出信号值。电流输出数据字格式的全量程范围设置为 0～32000，而电压输出数据字格式的全量程范围设置为–32000～+32000。

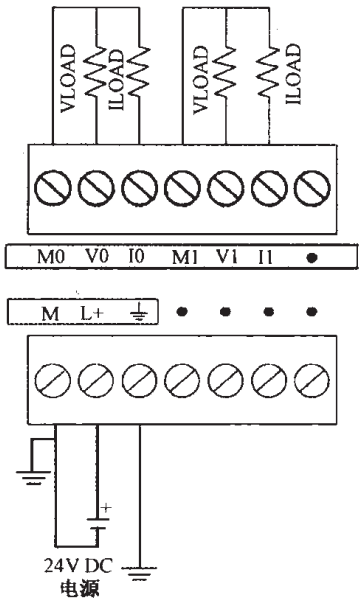


图 4-7 EM232 2 通道模拟量输出模块接线图

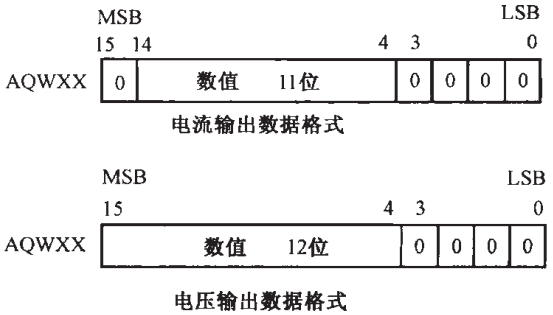


图 4-8 EM232 模块输出数据字格式

模拟量输入/输出模块（EM235）具有 4 路模拟量输入通道和 1 路模拟量输出通道。这为 S7-200 CPU 组态提供了很大的灵活性。在应用上与前述的 EM231、EM232 类似，这里不再赘述。下面通过实例来说明模拟量扩展模块的编程方法。

实例 65：CPU 扩展 EM231 进行模拟量输入信号测量

实例说明

模拟量输入模块 EM231 与 CPU 模块构成 PLC 控制系统，模拟输入信号的量程为±10 V。要求对模拟量输入信号进行测量。

实例实现

由于输入信号的量程为±10 V，因此应将 EM231 的组态开关设置为相应的量程。在工业现场存在较多的电磁干扰，这会使模拟信号不稳定。在实际应用中一般通过多次采样取平均值的方法来减少干扰信号的影响。本例中也采用这种方法设计控制程序，程序如图 4-9 所示。

图 4-9 梯形图对应的语句如下：

```
//主程序
LD      SM0.0
CALL    SBR0            // 初始上电，调用子程序 SBR0，进行初始化
LD      SM0.0
CALL    SBR1            // 调用模块检查子程序 SBR1
// 若模块正常（即标志位 Q1.0=0，Q1.1=0）则开始模拟量处理
LDN     Q1.0
AN      Q1.1
CALL    SBR2

//初始化子程序 SBR_0
// VW0:采样计数器，VW2:采样次数存储器，  VD10:当前采样值存储器，VD14:当前采样和存储器，  VD18:平均值存储器
LD      SM0.0
MOVW    0, VW0
MOVW    128, VW2
MOVD    0, VD10
MOVD    0, VD14
MOVD    0, VD18

//模块检查子程序 SBR1
// 检查第一个扩展模块是否存在，如果不存在则将 Q1.0 置 1
LDB=    SMB8, 16#19
NOT
S       Q1.0, 1
```

```
// 检查第一个扩展模块和电源是否正常，若不正常则将 Q1.1 置 1
LDB=    SMB9, 16#0
NOT
AB=     SMB9, 16#04
S       Q1.1, 1

//模拟量采样子程序 SBR_2
// 将模拟量输入值存在 VW12 中
LD      SM0.0
MOVW    AIW0, VW12
// 将输入值转换成双字
LDW>=   VW12, 0
MOVW    0, VW10
NOT
MOVW    16#FFFF, VW10
// 把采样值加到采样和中，并将采样次数加 1
LD      SM0.0
+D      VD10, VD14
INCW    VW0
// 若达到采样次数则把采样和复制到 VD18 中，并求平均值， 最后将采样和存储器和采样计数器清零
LDW>=   VW0, VW2
MOVD    VD14, VD18
ENCO    VW2, AC1
SRD     VD18, AC1
MOVD    0, VD14
MOVW    0, VW0
```

实例分析

本程序描述了模拟量模块 EM231 的功能。程序由一个主程序和三个子程序组成。程序开始执行时首先要调用第 1 个子程序进行有关参数的初始化，然后调用第 2 个子程序对模拟量模块的状况进行检测，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，若模拟量模块检查到电源出错，则将 CPU 上 Q1.1 接通。若模拟量模块工作正常则调用模拟输入采样子程序和模拟量输出子程序。在采样子程序中，从 AIW0 中取输入值，为了增加稳定性而以 128 次采样的平均值作为模拟量的测量值。为减少程序的扫描时间，在求平均值时采用了移位指令代替除法指令。

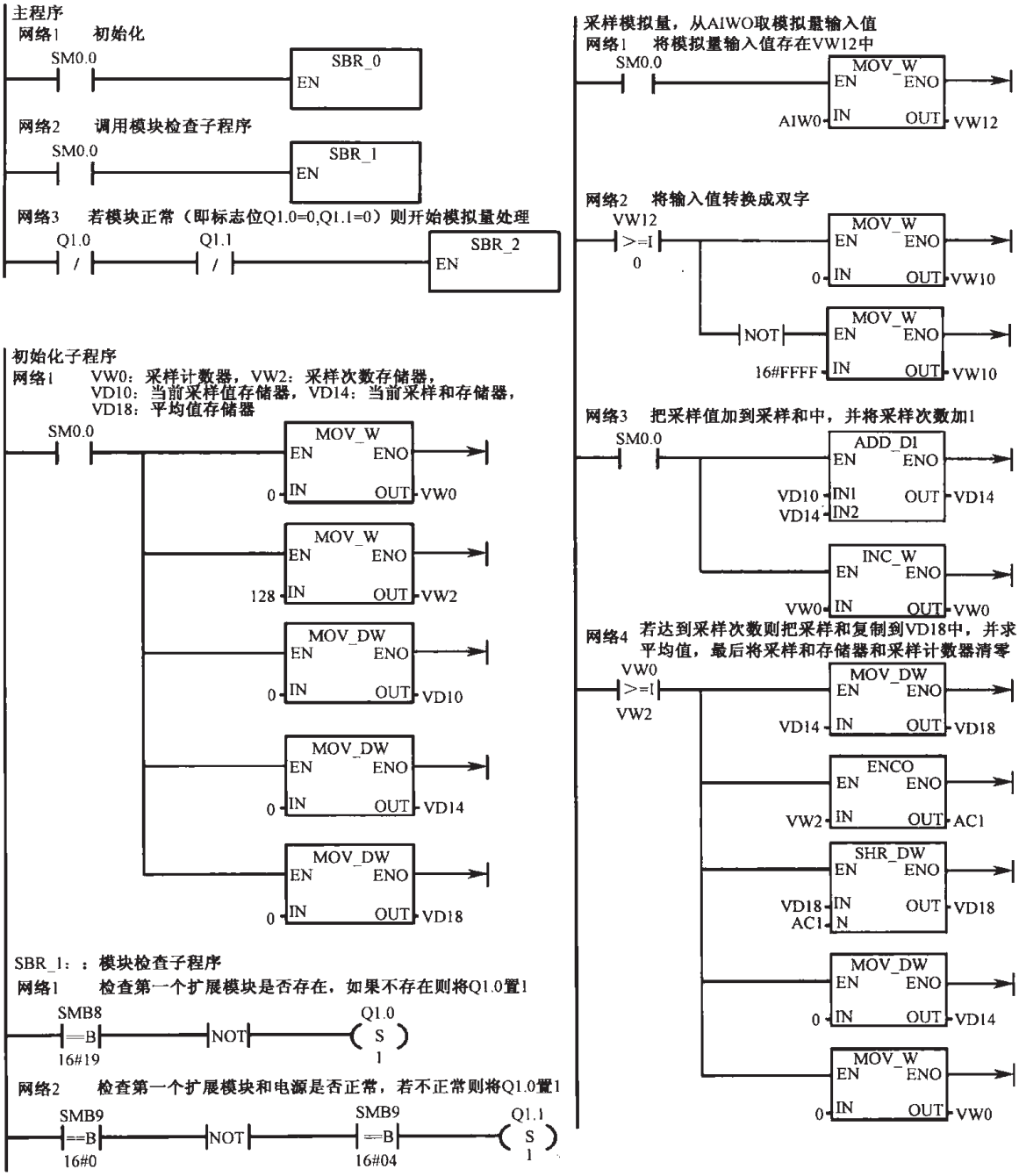


图 4-9　模拟量输入处理 PLC 程序

实例 66：CPU 扩展 EM235 实现温度控制

实例说明

一个温度控制系统要求将被控系统的温度控制在 10℃～100℃之间。目标温度设定为 50℃，当温度低于 40℃或高于 60℃时，应能通过加热器或冷却风扇进行调节，并以不同的指标灯指标系统所处的温度区间。

实例实现

选用模拟量输入/输出模块 EM235、温度传感器 PT100 与 PLC CPU222 模块构成控制系

统的基本单元。系统设置一个启动按钮来启动控制程序，设置绿（Q0.0）、红(Q0.1)、蓝（Q0.2）3 个指示灯来指示温度状态。当被控温度在要求范围内，绿灯亮，表示系统运行正常；当被控温度超过上限，红灯亮，同时启动冷却风扇（Q0.3）；当被控温度低于下限，蓝灯亮，同时启动加热器（Q0.4）。梯形图程序如图 4-10 所示。图 4-10 梯形图对应的语句如下：

```
// 初次上电时执行一次参数初始化
LD      SM0.1
MOVD    0, VD196
MOVW    1585, VW250
MOVW    6400, VW252
MOVW    100, VW260
MOVW    10, VW262
MOVW    40, VW266
MOVW    60, VW264
MOVW    50, VW268
MOVW    20000, AQW0
// 将来自温度变送器的模拟信号转换为实际的温度值
LD      SM0.0
MOVW    AIW2, VW200
-I      VW252, VW200
DIV     VW250, VD198
MUL     10, VD196
DIV     VW250, VD196
MOVW    VW198, VW160
MOVW    0, VW198
MUL     10, VD198
MOVW    VW160, VW198
+I      VW200, VW198
MOVW    VW200, VW116
// 温度超过 100℃，亮起红灯报警
LDW>=   VW200, VW260
=       Q0.1
// 温度低于 10℃并亮起报警
LDW<=   VW200, VW262
=       Q0.2
LDW<    VW200, VW260
AW>     VW200, VW262
=       Q0.0
// 温度超过 60℃，启动冷却风扇
LDW>=   VW200, VW264
S       Q0.3, 1
// 温度低于 40℃，启动加热器
LDW<=   VW200, VW266
S       Q0.4, 1
```

```
LDW=    VW200, VW268
R       Q0.4, 1
R       Q0.3, 1
```

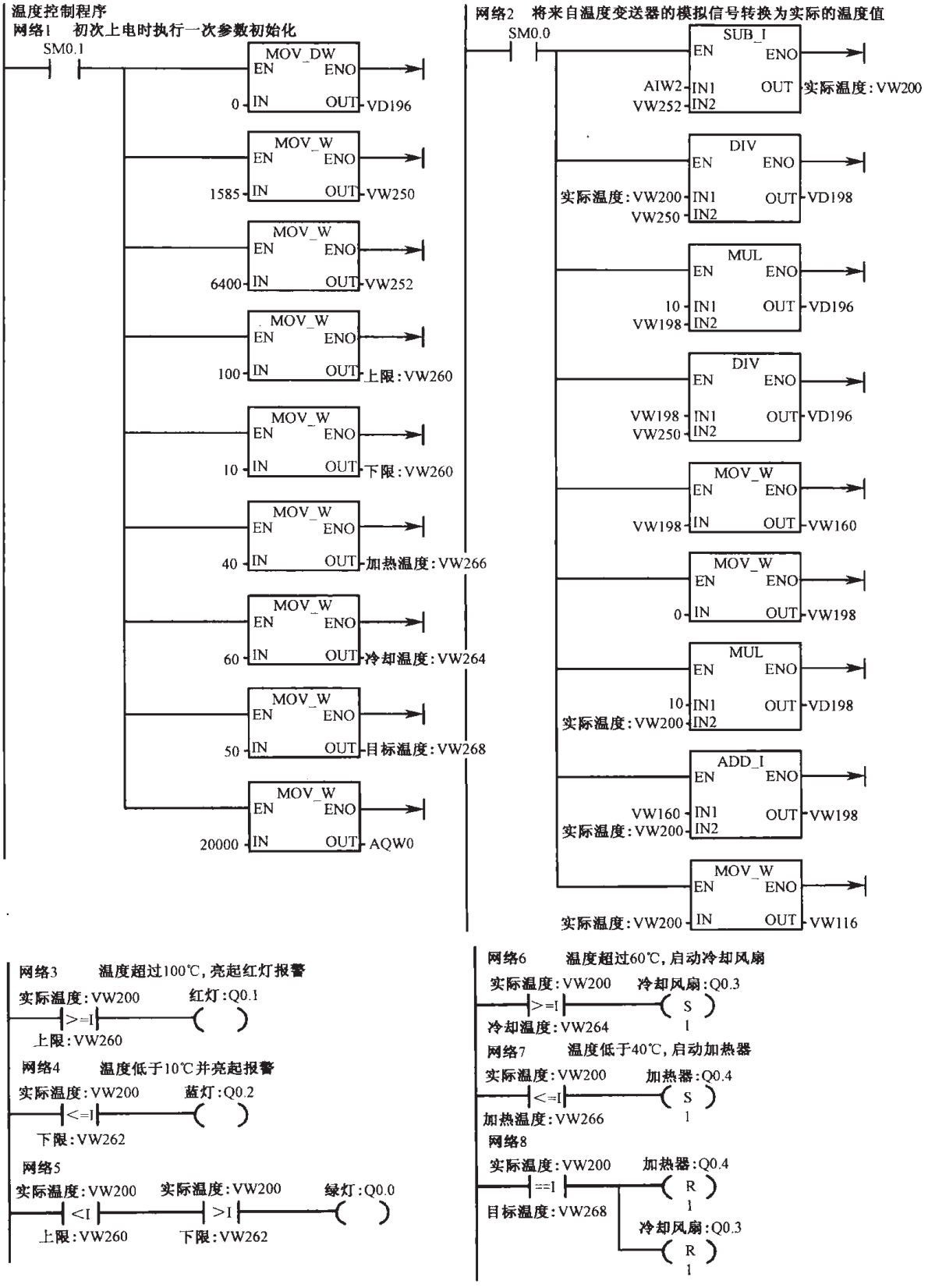


图 4-10　温度控制程序

实例分析

在被控系统中的温度测量点，温度信号经变送器变成 4～20 mA 的电流信号送入 EM235 的第 2 个模拟量输入通道 AIW2 中，PLC 读入温度值后，再取其平均值作为被控系统的实际温度值（为了避免与上例重复，本例程序中仅以 AIW2 的读数值来表示实际温度）。为了把温度传感器 PT100 随温度变化的电阻转换成相应的温度变化值，利用下面的温度公式求得：

T℃=（温度数字量–0℃偏置量）/1℃数字量

其中，温度数字量为存储在 AIWx（x=0,2,4）中的值；

0℃偏置量为在 0℃测量出的数字量，这里取为 6400；

1℃数字量为温度每升高 1℃的数字量，这里取为 1585。

温度传感器 PT100 在正常工作时需要 12.5 mA 的电流，这里采用 EM235 模块的输出通道来给温度传感器供电。由于 EM235 模块的工作电流被 DIP 开关设置为 0～20 mA，因此，为了获得 12.5 mA 的输出电流，应将 AQW0 的输出数设置为 20000（32000/20×12.5=20000）。

在实际应用中对一般模拟量进行处理时可直接应用本例中的方法。

4.4 位控模块

位控模块 EM253 是 S7-200 的特殊功能模块，它能够产生脉冲串，用于步进电动机和伺服电动机的速度和位置的开环控制。位控模块 EM253 提供了带有方向控制、禁止和清除输出的单脉冲输出。另外，专用输入允许将模块组态为包括自动参考点搜索在内的几种操作模式。模块为步进电动机或伺服电动机的速度和位置开环控制提供了统一的解决方案。定位模板 EM253 应用于位置控制的过程，实现起来非常简单。STEP 7-MicroWIN 提供了一个定位模板 EM253 配置的向导操作（Position Control Wizard），可以帮助用户迅速完成配置操作，并将相关配置存储在 S7-200 PLC 的 V 区内，更换位控模块时无需重新配置；同时，STEP7-MicroWIN 还提供了一个界面非常友好，专门用于调试、监控运动控制过程的调试界面（定位模板 EM253 Control Panel）。

4.4.1 位控模块 EM253 的硬件特性

位控模块可提供单轴、开环移动控制所需要的功能和性能。提供 12 Hz～200 kHz 的脉冲频，支持急停（S 曲线）或线性的加、减速功能。支持螺距回程误差补偿，有绝对、相对和手动 3 种控制方式。提供连续的位置控制，最多可以支持 25 个位置点的控制，每段运动轨迹包络可以以最多 4 种不同的速度来实现。另外，该模块还支持 4 种不同的参考点寻找模式，控制系统的测量单位可以采用脉冲数也可以采用工程单位（如英尺、毫米）。

S7-200 可以附加的定位模块 EM253 的个数取决于 CPU 的电源带载能力，其中 CPU222 可以带 1 块，CPU224/224XP 可以带 3 块，CPU226 可以带 5 块，而 CPU221 则不能扩展。定位模块 EM253 的硬件结构如图 4-11 所示，各接线端子的功能如表 4-5 所示。定位模板 EM253 集成有“5 个数字量输入端子”（STP，停止；RPS，参考点开关；ZP，零脉冲信号；LMT+，正方向硬极限位置开关；LMT–，负方向硬极限位置开关），“6 个数字量输出端子”（4 个

信号：DIS，CLR，P0，P1，或者 P0+、P0–，P1+、P1–），用于 S7-200 PLC 定位控制系统中。通过产生高速脉冲来实现对单轴步进电动机的开环速度、位置控制。通过 S7-200 PLC 的扩展接口，实现与 CPU 间的通信控制。

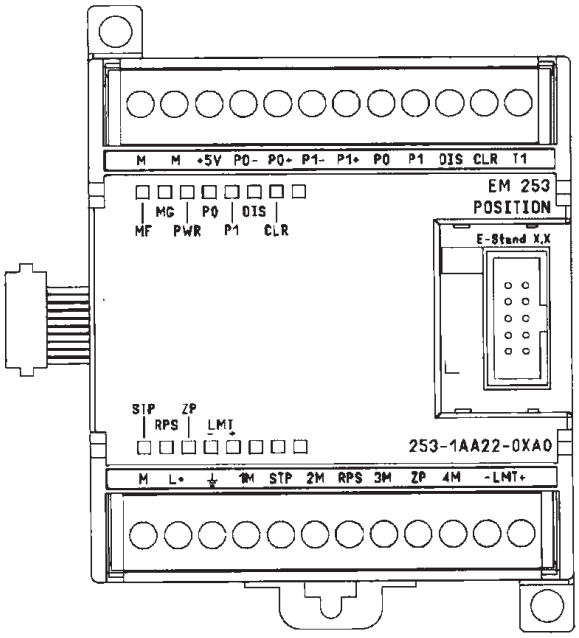


图 4-11　EM253 硬件结构图

表 4-5　定位模块 EM253 各端子功能表

端　子	功 能 描 述
STP	STP 为输入端子，可让模块停止脉冲的生成在位控向导中可选择所需要的 STP 操作
RPS	RPS（参考点开关）输入为绝对移动操作建立参考点或原位
ZP	ZP（零时钟脉冲）输入可帮助建立参考点或原位。典型地，电动机驱动器/放大器在电动机每旋转一周就产生一个 ZP 脉冲
LMT+ LMT–	LMT+和 LMT–输入将建立移动行程的最大限制。位置控制向导将允许配置 LMT+和 LMT–输入的操作
P0 P1 P0+、P0– P1+、P1–	P0 和 P1 为控制电动机移动及移动方向的开路漏极晶体管脉冲输出。P0+、P0–和 P1+、P1–为分别提供与 P0 和 P1 一样功能的差分脉冲输出，但它们所提供的信号质量更好。开路漏极输出和差分输出可同时全部激活。可以根据电动机驱动器/放大器的接口要求来选择使用哪一种脉冲输出
DIS	DIS 是一种用来禁用或启用电动机驱动器/放大器的开路漏极晶体管输出
CLR	CLR 是一种用来清除伺服脉冲计数寄存器的开路漏极晶体管输出
M、L+	L+为+24V 电源。M（共 3 个）为电源地
1M、2M3M、4M	1M、2M、3M、4M 分别为 STP、PRS、ZP、LMT+/1 端子的信号地
+5V	输出 5V 电压
T1	与+5V 、P0、P1、DIS 结合一起使用

位控模块 EM253 可以通过驱动器（放大器）与步进电动机或其他伺服电动机相连接构成开环控制系统。但其接线比其他扩展模块的接线要复杂，而正确的接线是保证电动机正常工作的基础。如图 4-12 所示为位控模块 EM253 与一台 SIMATIC FM-STEP 驱动器的接线图，

用户可根据实际需要在 EM253 的输入端接入停止开关、限位开关或参考位开关等外部开关。与其他型号电动机驱动器的连接方式，读者可参考西门子 PLC 选型手册。

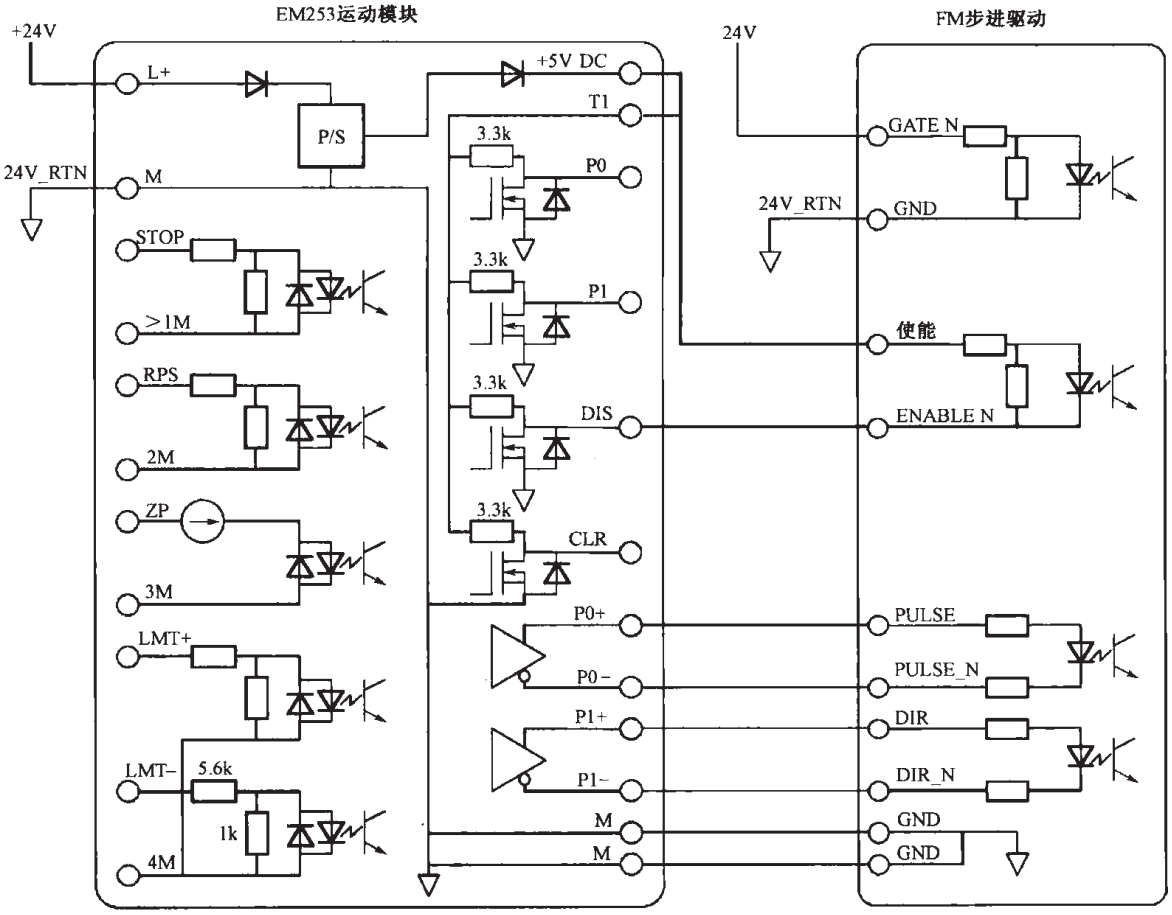


图 4-12　位控模块 EM253 与 SIMATIC FM-STEP 驱动器的接线图

4.4.2　位控模块 EM253 的配置

为了使模块能够控制移动，必须为位控模块创建配置/概要表。位置控制向导（Position Control Wizard）将一步一步地引导用户完成整个配置过程，从而使配置过程快速而方便。位置控制向导还允许脱机创建配置/概要表。可以在不连接 S7-200 CPU 的情况下，使用所安装的位控模块创建配置。

（1）启动位置控制向导

可以单击浏览条中的“工具”图标，然后双击“位置控制向导”图标，也可以选择菜单命令工具位置控制向导。

（2）选择用于 S7-200 PLC 的位置控制模式

选择“配置 EM253 位控模块操作”，如图 4-13 所示。

（3）输入定位模块 EM253 的逻辑位置

如图 4-14 所示，选择位控模块在 PLC 机架中的逻辑位置。

用户必须首先设置定位模块 EM253 的逻辑位置，才可以继续完成后面运动参数、运动轨迹包络的设置。“Position Control Wizard”配置工具，允许用户非常方便地通过 S7-200 PLC 编程窗口，读到已经正确接好线的定位模块 EM253 的逻辑位置。

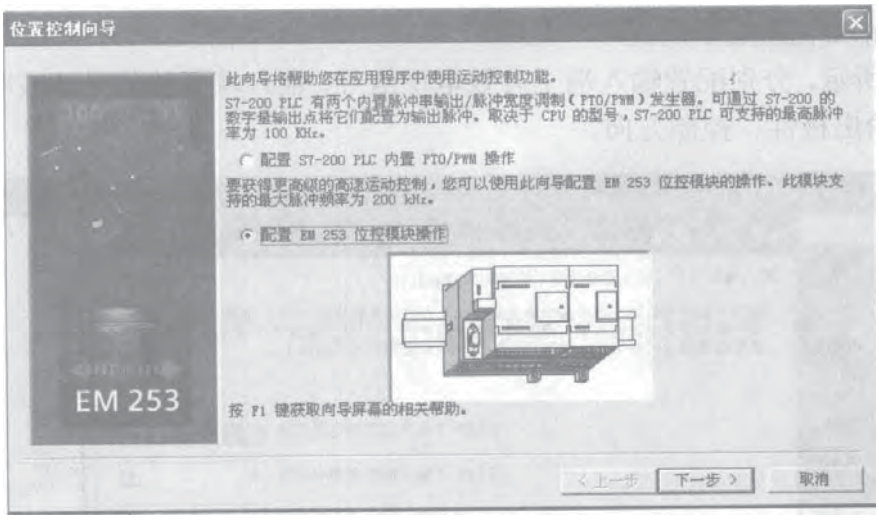


图 4-13　位控向导配置选择窗口

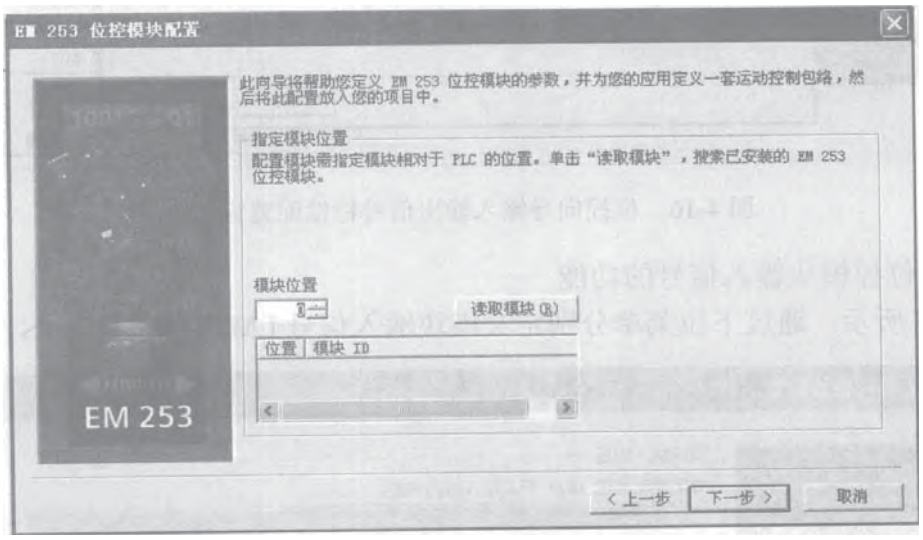


图 4-14　位控向导指定模块位置窗口

（4）输入系统的测量单位

如图 4-15 所示，设置位控系统中所用的度量单位（“工程量”或者“脉冲数/转”）。

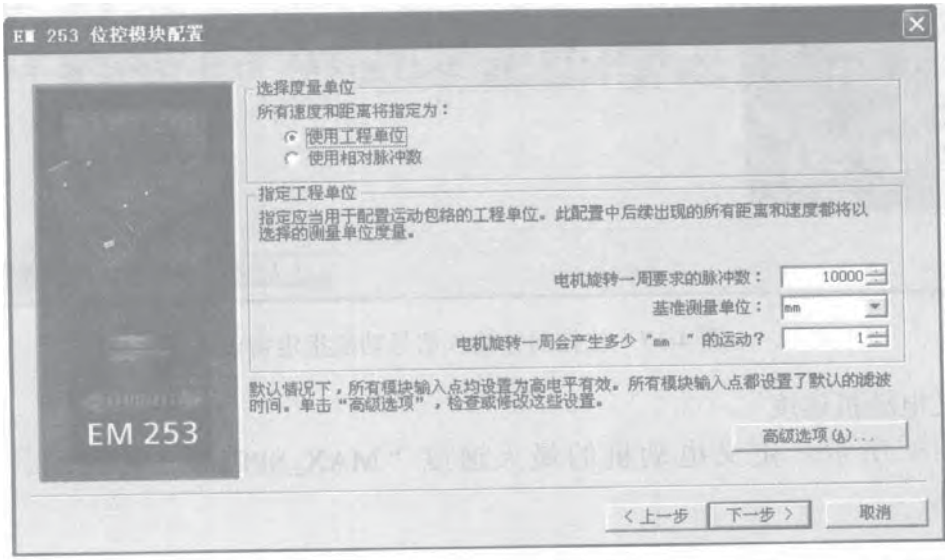


图 4-15　位控向导测量单位设定窗口

（5）编辑输入/输出端子配置

如图 4-16 所示，分别配置输入端子有效电平信号、输入端子的信号滤波时间和控制步进电动机的脉冲输出极性、控制方向。

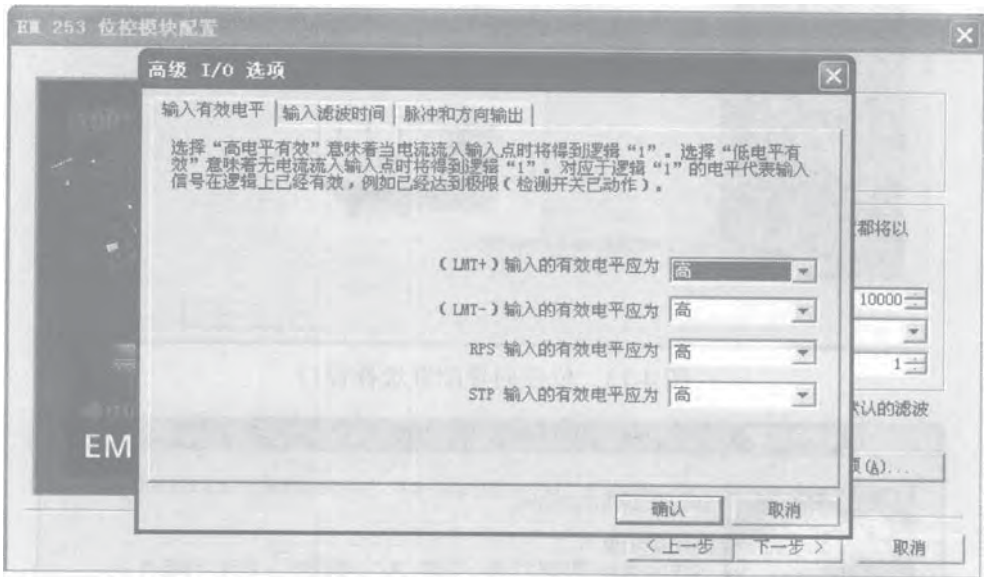


图 4-16　位控向导输入输出信号特性配置窗口图

（6）指定位控模块输入信号的功能

如图 4-17 所示，通过下拉菜单分别定义模块输入信号 LMT+、LMT–、STP 的功能。

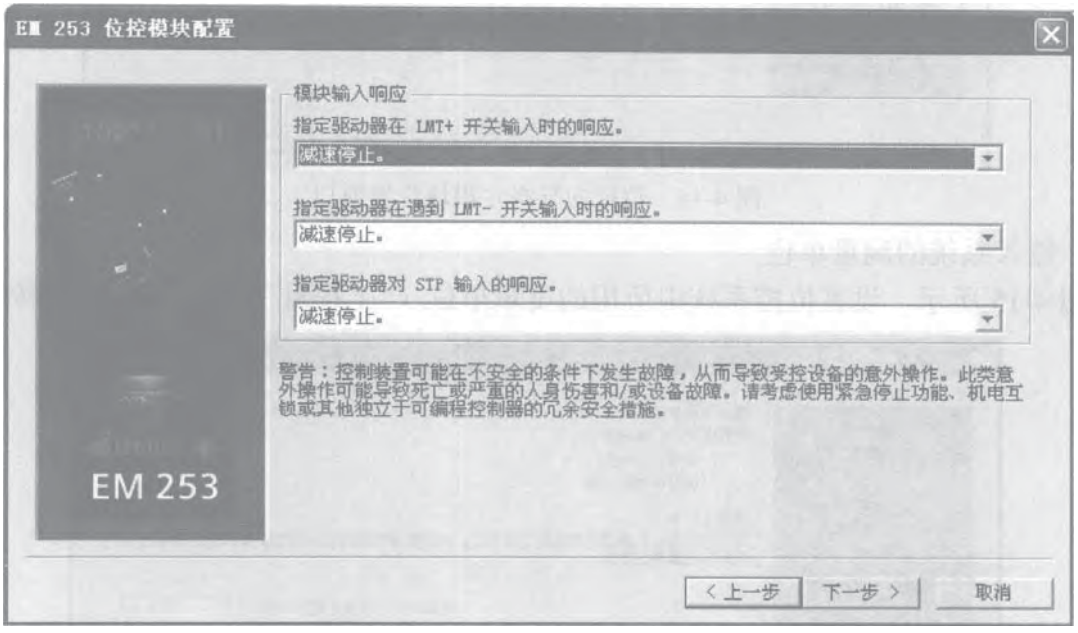


图 4-17　位控向导输入信号功能指定窗口

（7）定义电动机速度

如图 4-18 所示，定义电动机的最大速度“MAX_SPEED”和启动、停止的速度“SS_SPEED”。

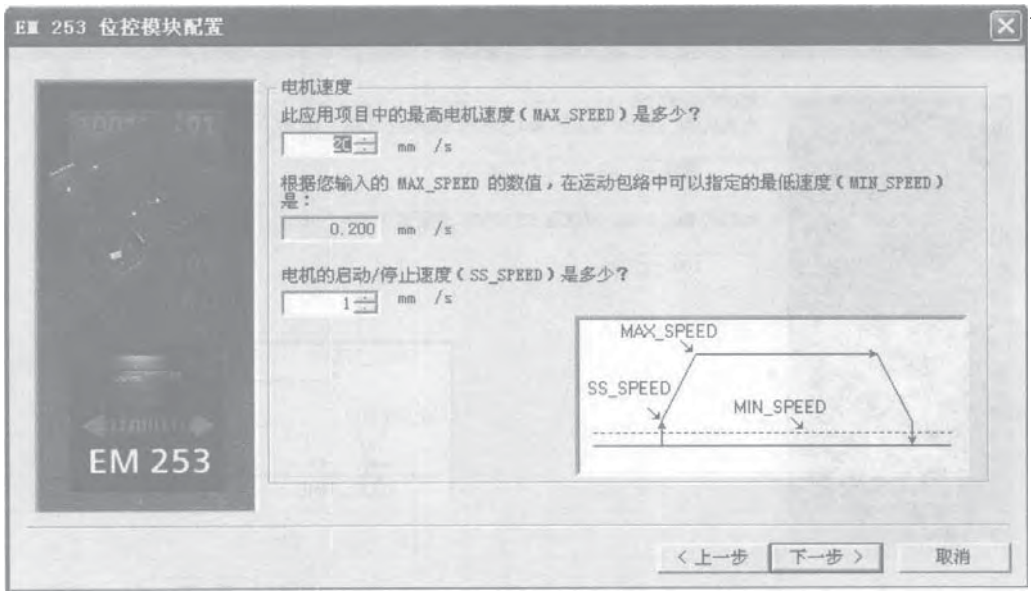


图 4-18　位控向导电动机速度设置窗口

（8）手动操作参数的设置

如图 4-19 所示，设置手动操作的速度和手动操作时间少于 0.5s 时，增量运动距离。

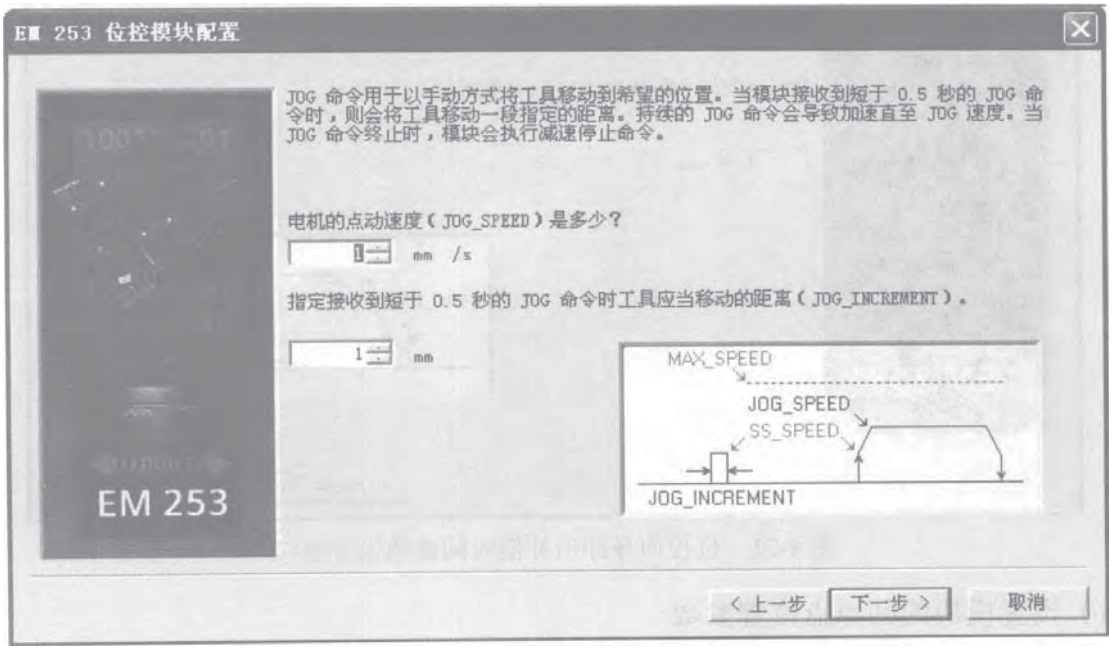


图 4-19　位控向导手动操作参数设置窗口

（9）加、减速度的时间参数设置

设置从“启动运动位置”到“设定速度”的加速度时间“ACCEL_TIME”。

设置运动到达终点位置的减速度时间“DECEL_TIME”。

（10）设置冲击补偿时间参数

如图 4-21 所示，设定“JERK_TIME”参数。

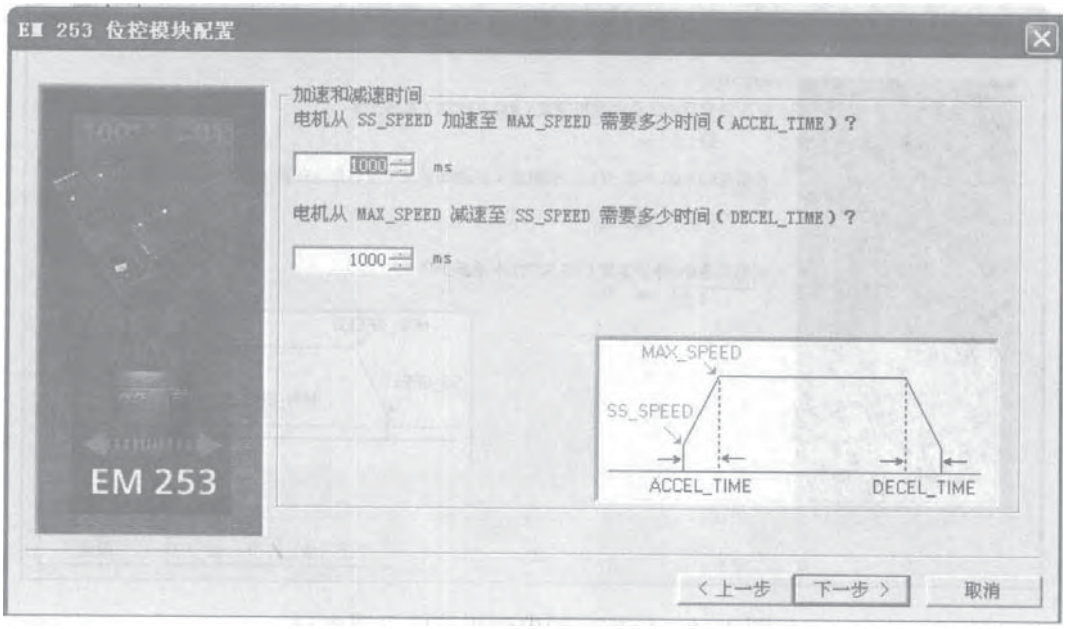


图 4-20　位控向导加、减速度时间参数设置窗口

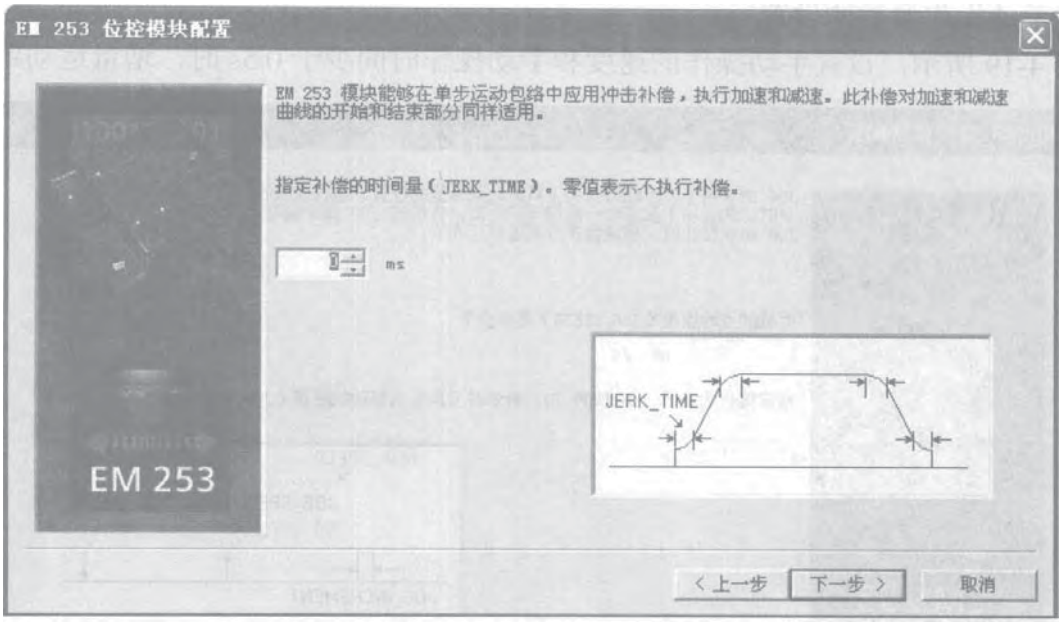


图 4-21　位控向导冲击补偿时间参数设置窗口

（11）设置模块的回原点位置参数

如图 4-22 所示为参考点设置选择窗口，若需要找寻原点操作则继续，否则进入运动轨迹包络设定窗口。

如图 4-23 所示，在参考点设置窗口中分别设置找寻原点操作的快速移动速度“RP_FAST”、精确定位移动速度“RP_SLOW”、运动方向“RP_SEEK_DIR”和确定原点的机械位置在原点开关的左侧或者右侧“RP_APPR_DIR”。设置完成之后单击“Advanced RP Options”按钮进入“高级 RP 选项”窗口（如图 4-24 所示），设置输入原点位置的偏置值“RP_OFFSET”和机械螺距补偿“BKLSH_COMP”。设置完成后进入参考点搜寻模式选择窗口（如图 4-25 所示）。找寻原点操作的工作模式有 0～4 共 5 种模式，每一种模式的时序图，可参考 S7-200 的系统手册。

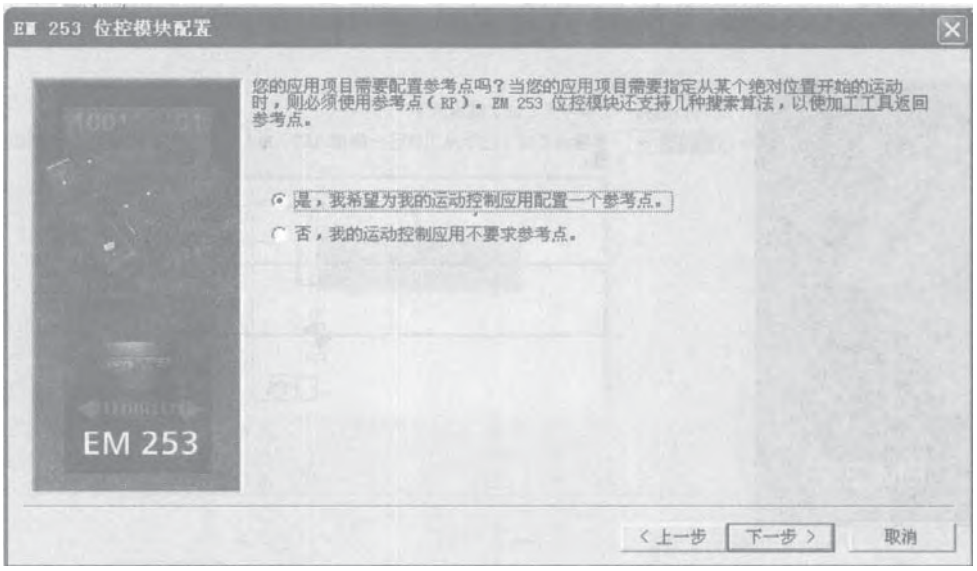


图 4-22　位控向导参考点设置选择窗口

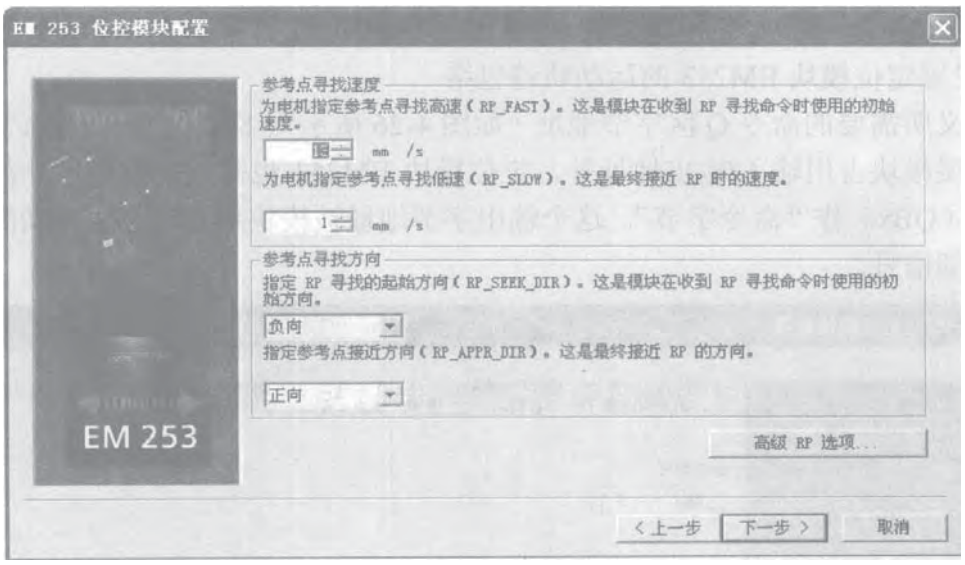


图 4-23　位控向导参考点找寻运动参数设置窗口

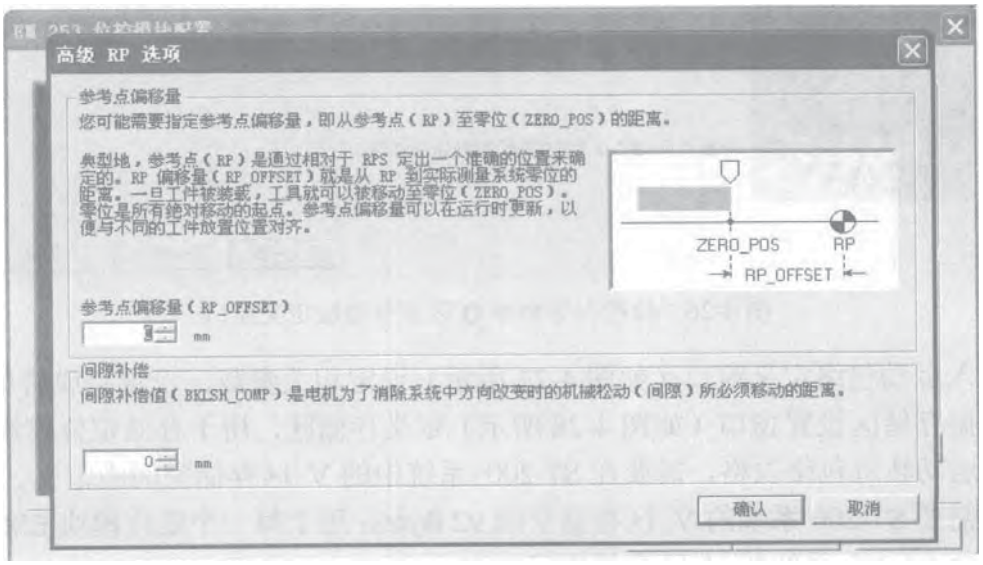


图 4-24　位控向导高级 RP 选项窗口

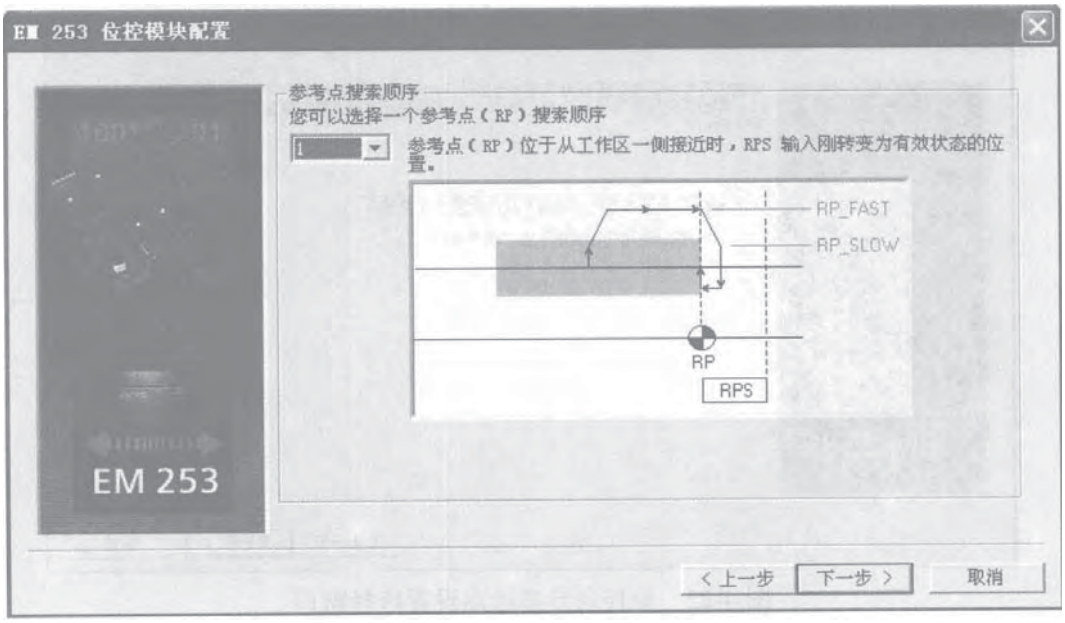


图 4-25　位控向导参考点搜寻模式选择窗口

（12）设置定位模块 EM253 的运动轨迹包络

首先定义所需要的命令 Q 区字节地址（如图 4-26 所示）。S7-200 系统中除了数字量和模拟量 I/O 扩展模块占用输入/输出地址外，定位模块 EM253 也需要在输出地址范围中占用一个字节地址（QBx）作“命令字节”。这个输出字节地址被模块用来进行运动功能控制，不直接连接到外部信号。

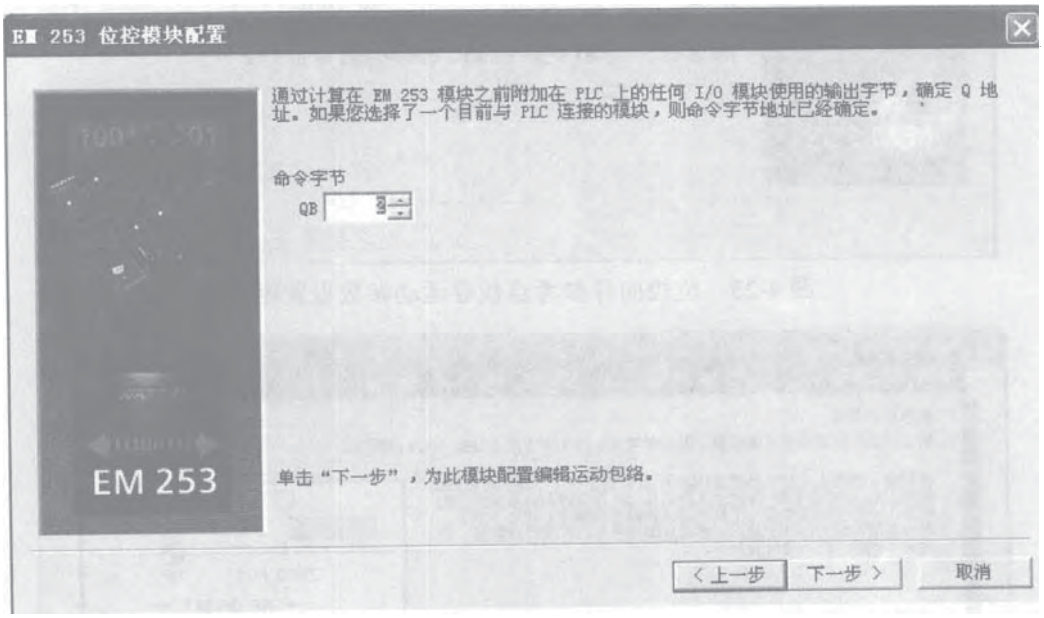


图 4-26　位控向导命令 Q 区字节地址定义窗口

接着进入运动包络定义窗口（如图 4-27 所示）设定相关参数。设置完成后单击“确认”按钮进入数据存储区设置窗口（如图 4-28 所示）定义存储区，用于存储定位模块 EM253 的参数配置、运动轨迹包络表格，需要在 S7-200 系统中的 V 区存储空间中定义。其中，运动参数的配置需要 S7-200 系统的 V 区存储空间 92 Bytes；用于每一个定位模块 EM 253 的运动轨迹包络需要 S7-200 系统的 V 区存储空间 34 Bytes。

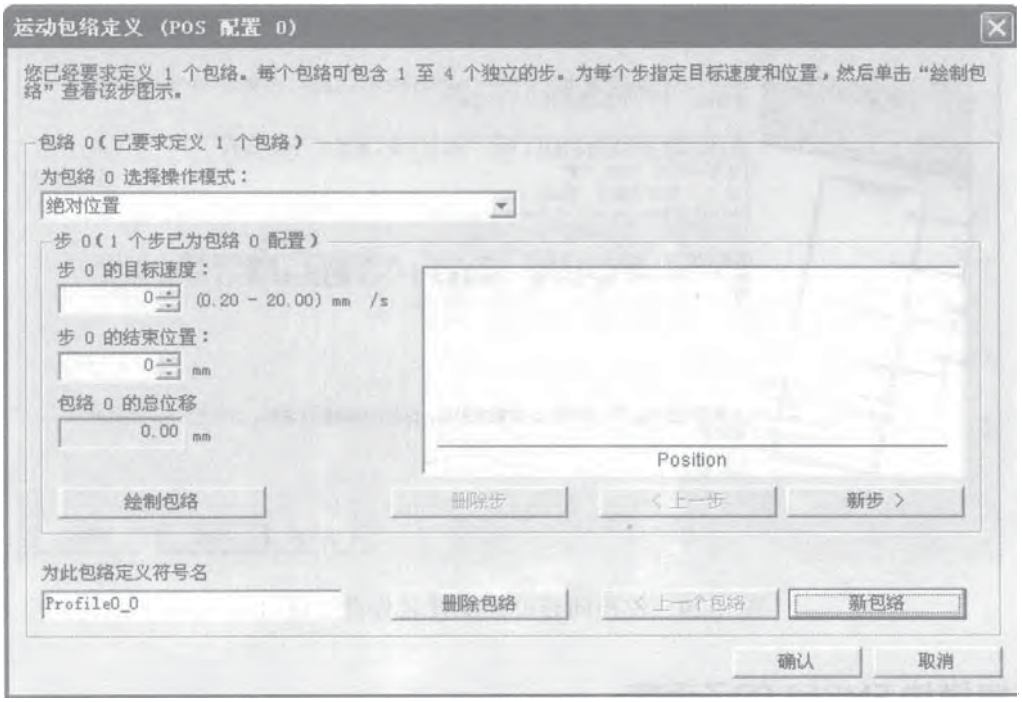


图 4-27　位控向导运动包络定义窗口

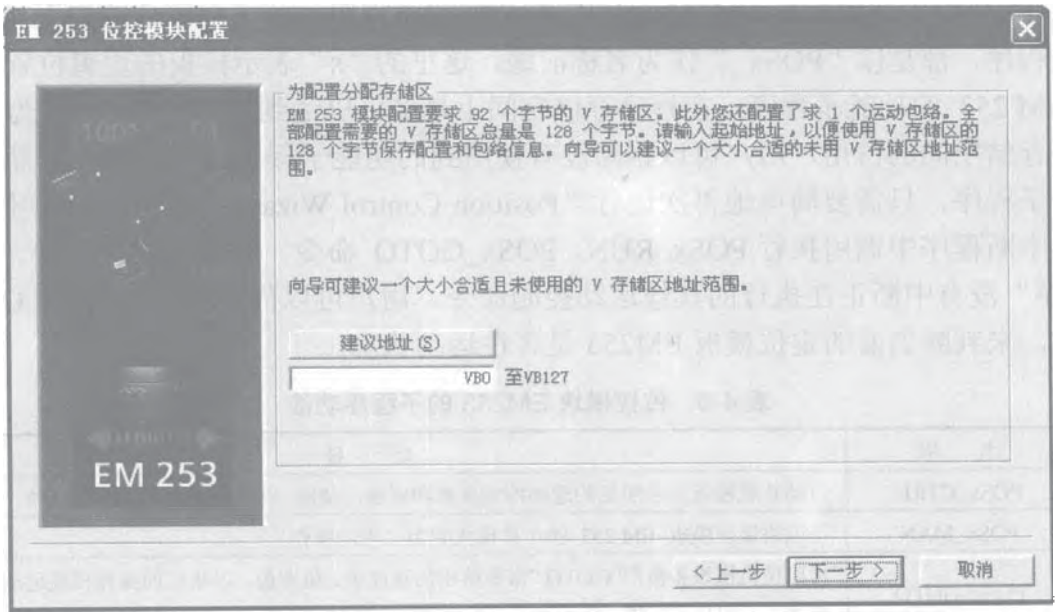


图 4-28　位控向导数据存储区设置窗口

（13）完成组态

完成上述任务后，进入位控配置名称设置窗口（如图 4-29 所示），设置位控配置的名称以方便以后使用时识别。最后单击“完成”按钮，位控向导将模块的组态参数和运动轨迹包络表插入用户的 S7-200 程序的数据块中，为位控参数生成了一个全局符号表，并在用户项目的程序块中增加了位控指令子程序。如果用户需要修改以上的参数和设置，只需要重新运行“位控向导”就可以。

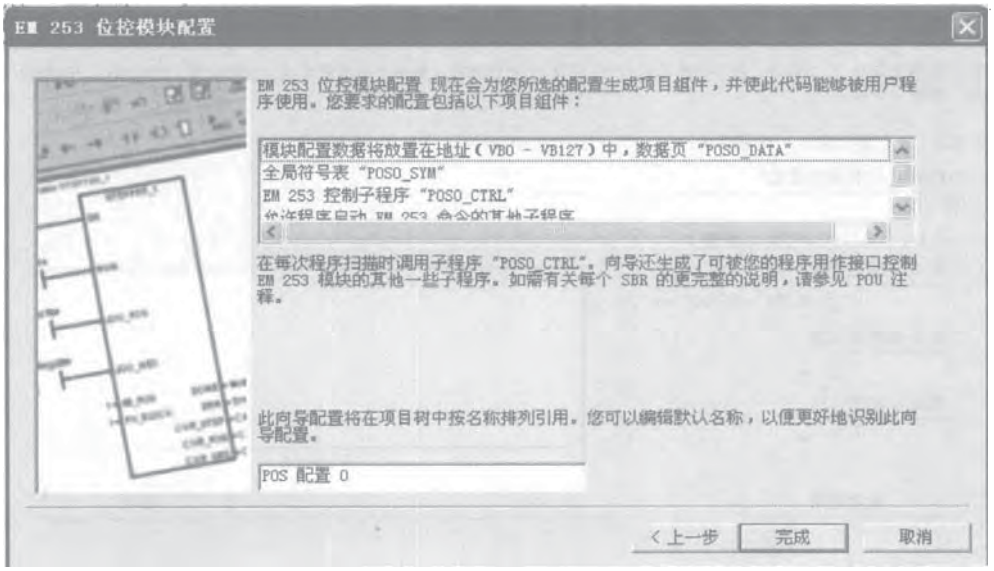


图 4-29 位控向导位控配置名称设置窗口

4.4.3 位控模块 EM253 的子程序

用户可以非常容易地通过“Position Control Wizard”配置工具，在 STEP 7-Micro/WIN 软件中生成 11 个用于定位模板 EM253 运动控制功能的子程序，各子程序功能如表 4-6 所示。每一个子程序，都是以“POSx_”作为名称前缀，这里的“x”表示模板的逻辑位置。用于定位模板 EM 253 的功能子程序，在程序存储空间上最多可以达到 1700 个字节。为了减少不必要程序存储空间的占用，用户可以删除没有使用到的功能子程序。如果需要恢复用户所删除的功能子程序，只需要简单地再次运行“Position Control Wizard”配置工具就可以了。用户可以在中断程序中调用执行 POSx_RUN、POSx_GOTO 命令。但是重要的是，一定要确保“中断程序”没有中断正在执行的其他运动控制命令。用户可以借助监测“POSx_CTRL”命令的输出，来判断当前的定位模板 EM253 是否在运动状态。

表 4-6 位控模块 EM253 的子程序功能

序 号	名 称	功 能
1	POSx_CTRL	自动装载模板已经配置的运动控制参数和轨迹，使能、初始化定位模板 EM253
2	POSx_MAN	可以将定位模板 EM 253 的工作模式置为“手动操作”
3	POSx_GOTO	可以使机械设备按照“GOTO”命令给出的速度值、位置值，以指定的操作模式运动到相应的机械设备坐标系位置
4	POSx_RUN	可以使电动机按照预先定义好的运动轨迹包络，移动到指定的机械位置
5	POSx_RSEEK	设定机械设备坐标系的参考原点位置
6	POSx_LDOFF	设定机械设备坐标系参考原点位置的偏置数值
7	POSx_LDPOS	使定位模板 EM253 改变当前的机械坐标位置值为输入参数值“New_Pos”
8	POSx_SRATE	使定位模板 EM253 改变配置参数加速度时间、减速度时间、轨迹拐点时间
9	POSx_DIS	使定位模板 EM253 在 DIS 输出端子上高电平输出
10	POSx_CLR	使定位模板 EM253 在 CLR 输出端子上产生一个 500ms 的脉冲
11	POSx_CFG	重新装载最新的定位模板 EM253 配置参数和预定义运动轨迹包络

以下的运动控制命令可以用于简单的位置控制任务：

“POSx_CTRL”、“POSx_DIS”，使用 SM0.0 置位操作，保证在每个 CPU 扫描周期内均执行。

“POSx_RSEEK”或者“POSx_LDPOS”，建立机械坐标系的参考点坐标。

“POSx_MAN”，手动操作机械设备。

“POSx_GOTO”，移动设备到指定的机械坐标位置。

“POSx_RUN”调用执行在“Position Control Wizard”配置工具中预定义的运动轨迹包络。

其他的运动控制指令，可以根据工艺的需要灵活选择。

实例 67：EM253 实现简单相对运动

实例说明

用位控模块 EM253 实现简单相对运动。

实例实现

相对运动可以用于切割机的等长切割或者其他设备的等长或定点操作。这种运动不需要寻找 RP 或移动包络，运动长度可以是脉冲数或工程单位，因此只用 POSx_CTRL 和 POSx_GOTO 指令便可实现预期的功能。控制程序如图 4-30 所示。图 4-30 梯形图对应的语句如下：

```
// 简单相对运动
// 启动控制指令，当 I0.2 接通时，位控模块放弃所有正在进行当中的命令
LD      SM0.0
=       L60.0
LD      I0.2
=       L63.7
LD      L60.0
CALL    SBR1, L63.7, M1.0, VB900, VD902, VD906, V910.0
// 运动开始，使系统进入自动模式
LD      I0.0
AN      I0.2
EU
S       Q0.2, 1
S       M0.1, 1
// 立即停止并关断自动模式
LD      I0.2
R       Q0.2, 1
// 运动到指定的切割位置
LD      Q0.2
=       L60.0
LD      M0.1
EU
```

```
=       L63.7
LD      L60.0
CALL    SBR3, L63.7, VD500, VD504, 1, I0.2, Q0.4, VB920, VD922, VD926
// 到达切割位置时，接通切割机，进行 2s 的切割
LD      Q0.2
A       Q0.4
TON     T33, 200
AN      T33
=       Q0.3
// 切割完成后开始新一轮运动
LD      Q0.2
A       T33
LPS
AN      I0.1
=       M0.1
LPP
A       I0.1
R       Q0.2, 1
```

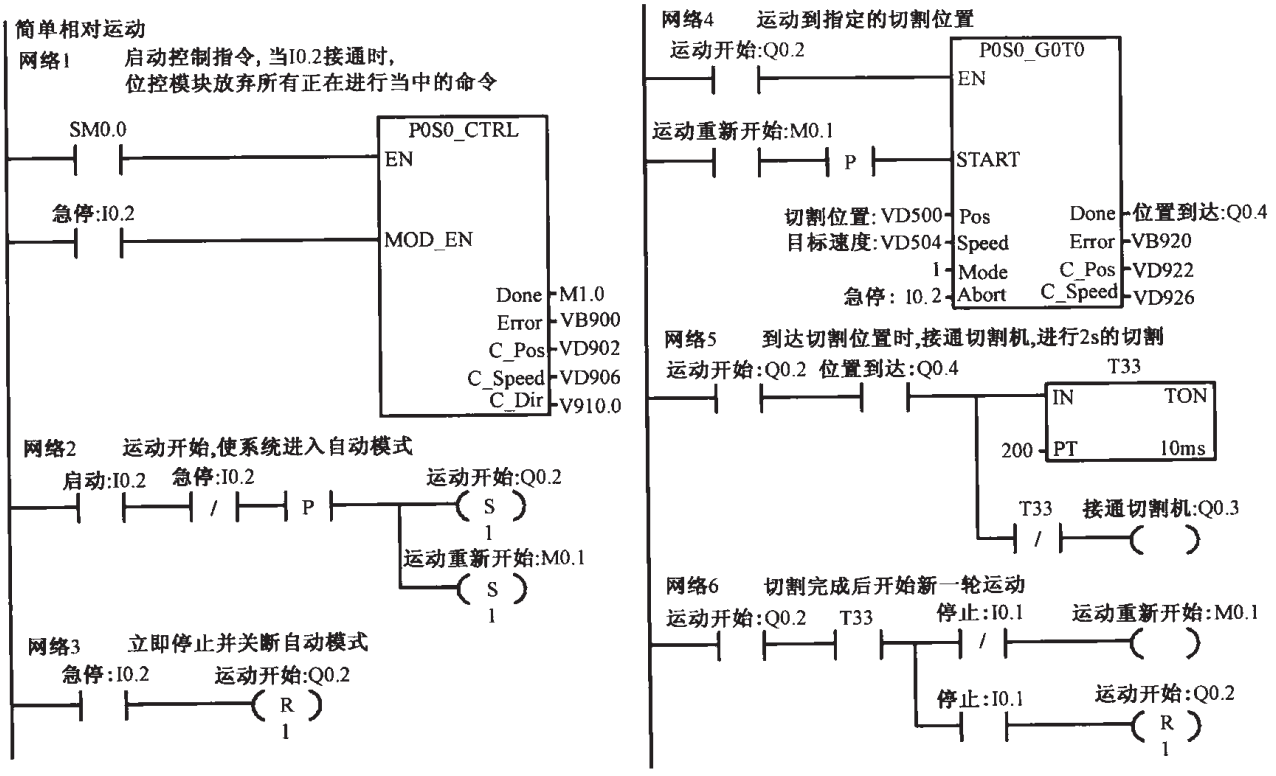


图 4-30 位控模块实现的相对运动

实例分析

程序开始运行时，向位控模块发出命令，装载组态、包络表，从而实现对位控模块的使能和初始化。当启动按钮（I0.0）接通时，设备启动，当设备运行到目标切割点时接通切割机，进行 2s 的切割动作。当停止按钮（I0.1）接通时，设备完成当前操作则停止。当急停按

钮（I0.2）接通时，设备终止任何运动并立即停止。切割点的相对位置和最大运动速度分别由 VD500 和 VD504 给定。

实例 68：EM253 实现典型的运动控制

实例说明

用位控模块 EM253 实现典型的运动控制。

实例实现

典型的运动控制一般包括回零功能（参考点寻找功能）、手动功能、自动功能和按照给定的轨迹运动的循迹功能。采用位控模块的 POSx_CTRL、POSx_RSEEK、POSx_MAN 和 POSx_RUN 指令来实现典型运动控制。控制程序如图 4-31 所示。

图 4-31 梯形图对应的语句如下：

```
LD      M1.1
LPS
AB=     VB930, 0
S       Q0.3, 1
SCRT    S0.2
LPP
AB<>    VB930, 0
SCRT    S1.0

SCRE

LSCR    S0.2
// 使用包络 1 运动到相应位置
LD      S0.2
=       L60.0
LD      S0.2
=       L63.7
LD      L60.0
CALL    POS0_RUN:SBR4, L63.7, VB288, I0.1, M1.2, VB940, VB941, VB942, VD944, VD948
// 自动模式下到达指定位置进行相应的操作
LD      M1.2
LPS
AB=     VB940, 0
S       Q0.4, 1
R       T33, 1
SCRT    S0.3
LPP
AB<>    VB940, 0
```

```
SCRT    S1.0

SCRE
Network 14
LSCR    S0.3
Network 15
LD      S0.3
TON     T33, 200
// 包络运动结束后重新启动，除非停止按钮接通
LD      T33
LPS
R       Q0.3, 1
R       Q0.4, 1
A       I0.2
SCRT    S0.1
LPP
AN      I0.2
R       M0.0, 4

SCRE

LSCR    S1.0
// 复位输出
LD      S1.0
R       Q0.3, 2
// 故障灯闪烁
LD      SM0.5
=       Q0.5
// 如果停止按钮接通，退出出错程序
LD      I0.2
R       M0.0, 9
R       S0.1, 8

SCRE
```

实例分析

程序分为手动功能和自动功能。手动功能能实现手动运行和正、反两方向的点动运动。自动运行功能是设备首先回到参考点，然后沿预先设定好的运动包络运动，然后设备重新回到参考点，开始新一轮的运动，直到有停止信号或急停信号才结束循环。运动包络由 STEP 7-Micro/Win 的位控向导生成。程序中 Q0.3 和 Q0.4 为运动设备所进行的两个动作控制位，在具体应用中读者可以酌情定义其功能。

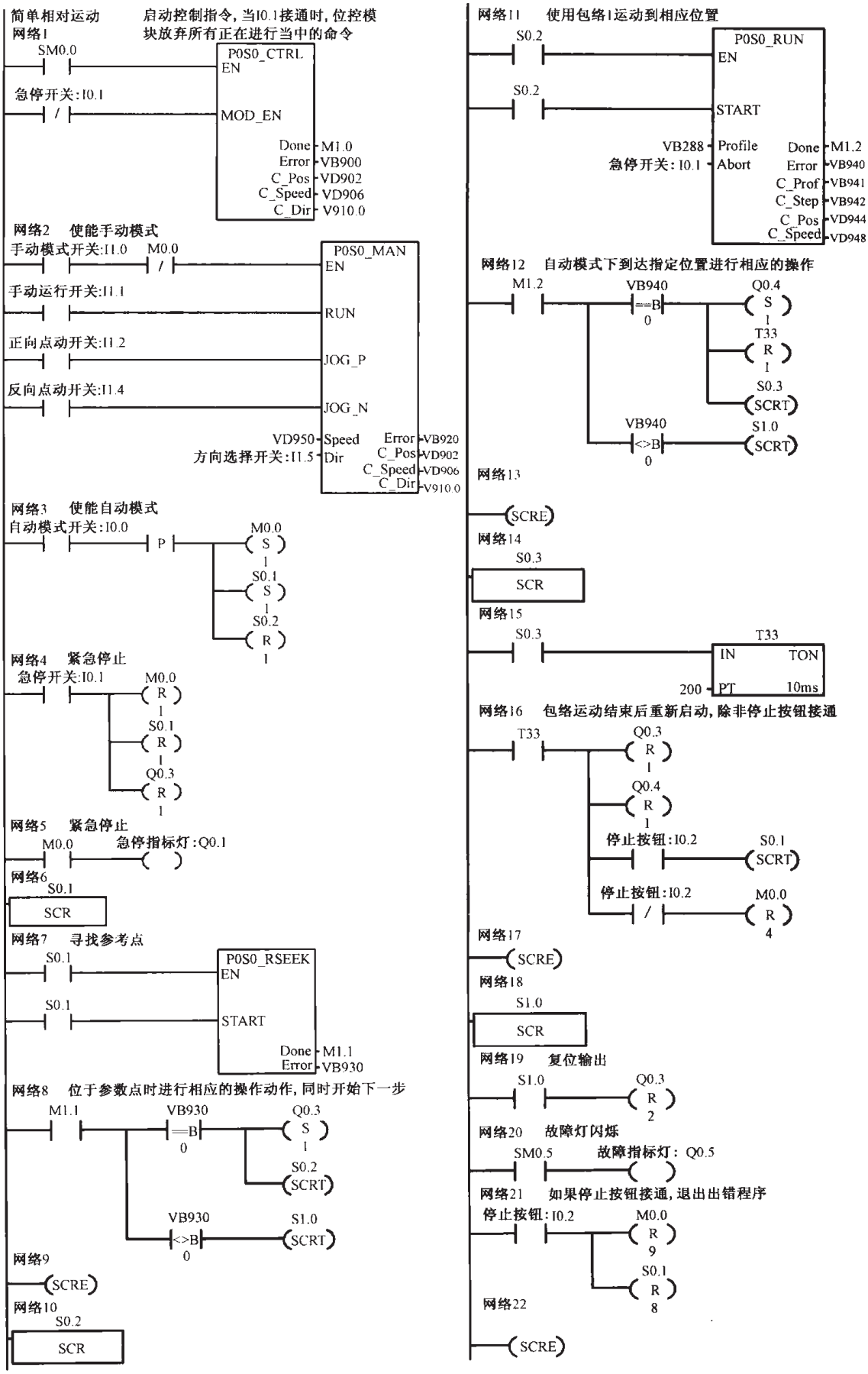


图 4-31　位控模块实现的典型运动图

4.5 PID 算法原理及指令介绍

在工业控制系统中常常用到闭环控制系统，而 PID 控制是常用的控制算法。PID 控制算法简单易懂，使用时可以不必弄清系统的数学模型，只要能检测出偏差就可以对系统实现准确、无静差的稳定控制。它作为最早实用化的控制算法已有 60 多年的历史，现在仍然是应用最广泛的工业控制算法。因此被称为控制领域的常青树。但是，PID 算法也有其局限性。PID 在控制非线性、时变、耦合及参数和结构的复杂系统时，效果不是太好。如果系统过于复杂，有时可能无论怎么调参数，都不易达到控制目的，需要采用其他更为先进的控制算法。

4.5.1 PID 算法介绍

PID 是比例（P）、积分（I）、微分（D）之意。标准的 PID 控制值与偏差（控制系统的设定值与实际值之差）、偏差对时间的积分、偏差对时间的微分，三者之和成正比，可以用式子表示如下：

$$M(t) = K_C(e + \int_0^t e\mathrm{d}t + \mathrm{d}e/\mathrm{d}t) + M_0 \qquad (4\text{-}1)$$

式中 $M(t)$——为时间函数，是 PID 控制回路的输出；

K_C——PID 控制回路增益；

e——PID 控制回路偏差；

M_0——PID 控制回路初始值。

PID 控制就是用 M（t）去进行控制对象的控制。计算偏差要使用反馈信号，所以 PID 控制是闭环控制。由式（4-1）可以看出，PID 控制算法由比例环节、积分环节和微分环节构成。

比例环节中比例系数越大，对同样的偏差，其控制作用也越强，意味着控制系统的反应速度也越快。但同时过大的比例系统也易引起系统超调量增大，导致系统振荡。积分环节主要是用于消除控制系统的累积误差，使系统成为无静差系统。微分环节则是与偏差的变化率有关系，偏差越大其控制作用越强，以抑制偏差的增大，如偏差的变化减小，其控制作用就减弱，以抑制偏差的减小。

由于计算机是数字化工作模式，在处理连续函数时，需将之离散化。式（4-1）离散形式为

$$M_n = K_C e_n + K_I \sum_{i=1}^{n} e_i + K_D(e_n - e_{n-1}) + M_0 \qquad (4\text{-}2)$$

式中 M_n——在第 n 采样时刻 PID 回路输出的计算值；

K_C——PID 控制回路增益；

e_n——在第 n 采样时刻 PID 控制回路的偏差；

e_{n-1}——在第 n–1 采样时刻 PID 控制回路的偏差（在第 n 采样时刻的偏差前值）；

K_I——PID 控制回路积分项系数；

K_D——PID 控制回路微分项系数；

M_0——PID 控制回路的初始值。

式（4-2）中，积分项是包括从第 1 个采样周期到当前采样周期的所有误差。计算中，没有必要保存所有采样周期的误差项。只需保存积分项前值 MX 即可。即

$$M_n = K_C e_n + K_I e_n + K_D(e_n - e_{n-1}) + MX = MP_n + MI_n + MD_n \tag{4-3}$$

式中　MX——积分前项值；

MP_n——在第 n 采样时刻的比例项；

MI_n——在第 n 采样时刻的积分项；

MD_n——在第 n 采样时刻的微分项。

比例项 MP_n 是增益 K_C 和偏差 e_n 的乘积，增益 K_C 决定输出对偏差的灵敏度。该项可以写为

$$MP_n = K_C e_n (SP_n - PV_n) \tag{4-4}$$

式中　SP_n——在第 n 采样时刻的给定值；

PV_n——在第 n 采样时刻的过程变量值。

积分项 MI_n 与偏差的和成正比。该项可以写为

$$MI_n = K_I e_n + MX = K_C T_S / T_I (SP_n - PV_n) + MX \tag{4-5}$$

式中　T_S ——采样周期；

T_I——积分时间常数。

积分项前值 MX 是第 n 采样周期前所有积分项之和。在每次计算出 MI_n 之后都要用 MI_n 去更新 MX。第一次计算时 MX 的初值被设置为 M_0。采样周期 T_S 是重新计算输出的时间间隔，而积分时间常数 T_I 控制积分项在整个输出结果中影响的程序。

微分项 MD_n 与偏差的变化成正比，该项可以写为

$$MD_n = K_D(e_n - e_{n-1}) = \frac{K_C T_D}{T_S}[(SP_n - PV_n) - (SP_{n-1} - PV_{n-1})] \tag{4-6}$$

为了避免给定值变化的微分作用而引起的跳变，可设定给定值不变（$SP_n = SP_{n-1}$），由此，式（4-6）可写为

$$MD_n = \frac{K_C T_D}{T_S}(PV_{n-1} - PV_n) \tag{4-7}$$

式中　T_D ——微分时间常数；

SP_{n-1}——在第 $n-1$ 采样时刻的给定值；

PV_{n-1}——在第 $n-1$ 采样时刻的过程变量值。

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。（对于增益值为 0.0 的 I 或 ID 控制，如果指定积分时间、微分时间为正，就是正作用回路；如果指定为负值，就是反作用回路。）

4.5.2　PID 回路指令

PID 回路指令（包含比例、积分、微分回路）在逻辑堆栈栈顶（TOS）值为 1 的前提下用来进行 PID 运算。如图 4-32 所示为 PID 回路控制指令。该指令有两个操作数 TBL 和 LOOP，二者均是 BYTE 型，限用 VB 区域。其中 TBL 是回路表的起始地址；LOOP 是回路号，可以是 0～7 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。PID 回

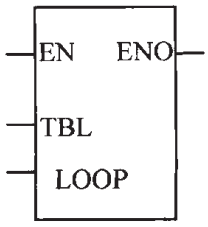


图 4-32　PID 指令

路指令根据输入和表（TBL）中的配置信息，对相应的 LOOP 执行 PID 回路计算。回路表包含 9 个参数，用来控制和监视 PID 运算。回路表格式如表 4-7 所示。

若要以一定的采样频率进行 PID 运算，采样时间必须输入到回路表中，且 PID 指令必须编入定时发生的中断程序中，或者在主程序中由定时器控制 PID 指令的执行频率。

表 4-7 回路表格式

偏移地址	变 量 名	数据类型	变量类型	描 述
0	过程变量（PV_n）	实数	输入	必须在 0.0～1.0 之间
4	设定值（SP_n）	实数	输入	必须在 0.0～1.0 之间
8	输出值（M_n）	实数	输入/输出	必须在 0.0～1.0 之间
12	增益（K_C）	实数	输入	比例常数，可正可负
16	采样时间（T_S）	实数	输入	单位为秒，必须是正数
20	积分时间（T_I）	实数	输入	单位为分钟，必须是正数
24	微分时间（T_D）	实数	输入	单位为分钟，必须是正数
28	积分项前项（MX）	实数	输入/输出	必须在 0.0～1.0 之间
32	过程变量前值（PV_{n-1}）	实数	输入/输出	最后 PID 运算的过程变量值，必须在 0.0～1.0 之间

S7-200 的 PID 回路没有设置控制方式，只有当 PID 控制器接通时，才执行 PID 运算。从这种意义上说，PID 运算存在一种“自动”运行方式。当 PID 运算不被执行时，称为“手动”模式。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，必须把手动方式下的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。

（1）置给定值（SP_n）=过程变量（PV_n）

（2）置过程变量前值（PV_{n-1}）=过程变量现值（PV_n）

（3）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，由于没有使能位的正跳变，那么就不能进行无扰动切换。

在许多控制系统中，有时只应用一种或两种环路控制的方法。例如，可能只需要比例控制或比例积分控制。选择期望的环路控制类型通过设置常量参数的数值来进行。如果不需要积分操作，那么应将积分时间常数设置为无穷大“INF”；如果不希望微分操作，那么应将微分时间数值设置为 0.0；如果不希望比例操作，那么应将增益设置为 0.0。

如果指令指定的回路表起始地址或 PID 回路号操作数超出范围，那么在编译期间，CPU 将产生编译错误，导致编译失败。PID 指令不检查回路表中的值是否在规定范围之内，因此，必须保证过程变量和设定值在 0.0～1.0 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。

4.5.3 PID 回路指令输入/输出变量数值转换

（1）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。过程变量与 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

制系统的实例中，过程变量应该是测量轮胎转速的测速计输入。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。在 PID 指令对这些现实世界的值进行运算之前，必须把它们转换成标准的浮点型表达形式。转换的第一步是把 16 位整数值转成浮点型实数值。下面的指令序列提供了实现这种转换的方法：

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

下一步是将现实世界的值的实数值表达形式转换成 0.0～1.0 之间的标准化值。下面的算式可以用于标准化给定值或过程变量值：

$$R_{Norm}=（R_{Raw}/S_{pan}）+Off_{set} \tag{4-8}$$

式中　R_{Norm}——标准化的实数值；

R_{Raw}——没有标准化的实数值或原值；

Off_{set}——单极性为 0.0，双极性为 0.5；

S_{pan}——值域大小，可能的最大值减去可能的最小值，单极性为 32000（典型值），双极性为 64000（典型值）。

下面的指令把双极性实数标准化为 0.0～1.0 之间的实数。通常用在第一步转换之后：

```
/R      64000.0, AC0        //累加器中的标准化值
+R      0.5, AC0            //加上偏置，使其在 0.0～1.0 之间
MOVR    AC0, VD100          //标准化的值存入回路表
```

（2）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。回路输出是 0.0～1.0 之间的一个标准化了的实数值。在回路输出可以用于驱动模拟输出之前，回路输出必须转换成一个 16 位的标定整数值。这一过程，是给定值或过程变量的标准化转换的逆过程。第一步是使用下面给出的公式，将回路输出转换成一个标定的实数值：

$$R_{Scal}=（M_n-Off_{set}）S_{pan} \tag{4-9}$$

式中　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。

Off_{set}，S_{pan} 的定义同式（4-8）。

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.,  AC0    //在累加器中得到刻度值
```

下一步是把回路输出的刻度转换成 16 位整数，可通过下面的指令序列来完成：

```
ROUND   AC0,   AC0      //把实数转换为 32 位整数
DTI     AC0,   LW0      //把 32 位整数转换为 16 位整数
MOVW    LW0,   AQW0     //把 16 位整数写入模拟输出寄存器
```

实例 69：水储罐恒压控制

实例说明

水储罐用于保持恒定水压。水以变化的速率不断地从水储罐取出。变速泵用于以保持充

足水压的速率添加水到储罐，并且也防止储罐空。此系统的设定值等于储罐达到充满 75%水位的设置。过程变量由浮点型测量器提供，它提供储罐充满程度的相同读数，可以从 0%（或空）到 100%（或全部满）之间变化。输出是泵速的数值，允许泵从最大速度的 0%到 100%运行。

实例实现

水箱水位控制梯形图程序如图 4-33 所示。

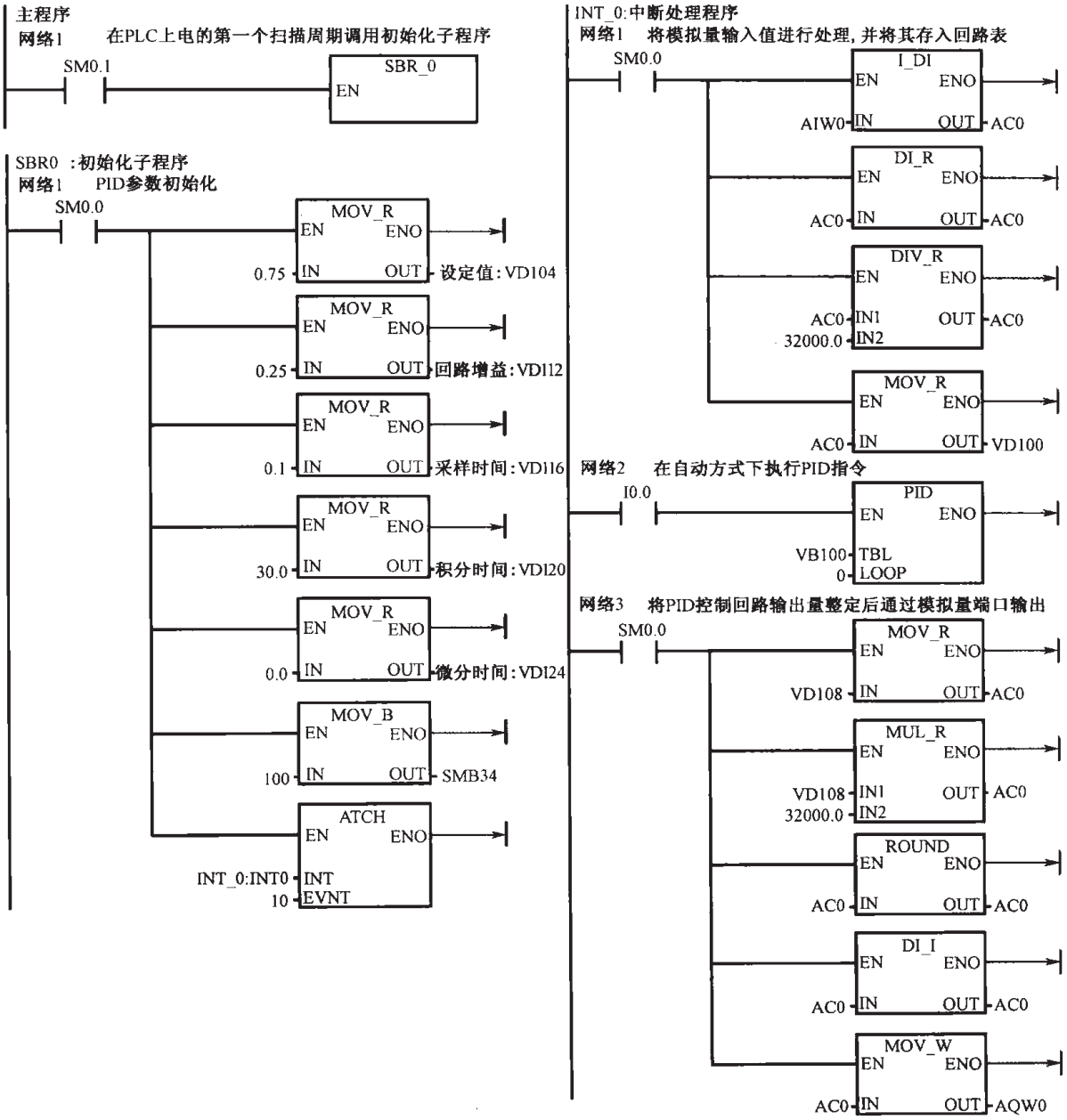


图 4-33 水箱水位的 PID 控制程序

图 4-33 梯形图对应的语句如下：

```
//主程序
// 在 PLC 上电的第一个扫描周期调用初始化子程序
LD      SM0.1
```

```
CALL    SBR0

//初始化子程序 SBR0
// PID 参数初始化
LD      SM0.0
MOVR    0.75, VD104
MOVR    0.25, VD112
MOVR    0.1, VD116
MOVR    30.0, VD120
MOVR    0.0, VD124
MOVB    100, SMB34
ATCH    INT0, 10

//中断处理程序 INT_0
// 将模拟量输入值进行处理，并将其存入回路表
LD      SM0.0
ITD     AIW0, AC0
DTR     AC0, AC0
/R      32000.0, AC0
MOVR    AC0, VD100
// 在自动方式下执行 PID 指令
LD      I0.0
PID     VB100, 0
// 将 PID 控制回路输出量整定后通过模拟量端口输出
LD      SM0.0
MOVR    VD108, AC0
MOVR    VD108, AC0
*R      32000.0, AC0
ROUND   AC0, AC0
DTI     AC0, AC0
MOVW    AC0, AQW0
```

实例分析

本系统的设定值为 75%时的水位，是预先确定的，直接输入循环表。进程变量作为来自浮点型测量器的单极、模拟数值提供。环路输出写入用于控制泵速的单极、模拟输出。模拟输入和模拟输出的 Span（扩展）都是 32 000。只有比例和积分控制可应用于此例。循环增益和时间常量从工程计算中确定，可以根据需要调整以获得最佳控制。时间常量的计算数值是：$K_C = 0.25$，$T_S = 0.1$s，$T_I = 30$min。泵速是手动控制的，直到水储罐为 75%满，阀打开允许水从储罐排出。同时，泵从手动切换到自动控制模式。数字输入用于将控制从手动切换到自动。此输入（I0.0）手动 / 自动控制：0 = 手动，1 = 自动。当处于手动控制模式，泵速度由操作员按 0.0～1.0 的实数值写到 VD108。

思考题

1. S7-200 PLC 扩展模块的 I/O 编址有何特点？
2. S7-200 PLC 的数字量扩展模块类型有哪些？各有何特点？
3. S7-200 PLC 的模拟量扩展模块类型有哪些？各有何特点？如何使用正确模拟量扩展模块？
4. S7-200 PLC 的位控模块有哪些特点与功能？配置时应该怎样设置参数？
5. S7-200 PLC 的通信量模块有哪些？
6. 简述 PID 控制的原理。S7-200 PLC PID 控制指令该如何使用？

第5章 顺序功能图程

- 基本概念
- 结构形式
- 顺序功能图的编程方法及梯形图表示

顺序功能图（Sequential Function Chart，SFC）是描述控制系统的控制过程、功能和特性的一种通用的技术语言，又叫做状态转移图、状态图或流程图，是设计 PLC 的顺序控制程序的有力工具。在 IEC 的 PLC 编程语言标准（IEC 61131-3）中，顺序功能图排在第一位。我国也于 1986 年颁布了顺序功能图的国家标准。

用顺序功能图设计顺序控制程序比直接用指令编程更简单，结构更清晰，可读性好，程序的调试和运行也很方便。设计过程比较规范，也相当直观，可以极大地提高工作效率。S7-200 PLC 采用顺序功能图法设计时，可用顺序控制继电器（SCR）指令、置位/复位（S/R）指令、移位寄存器指令（SHRB）等实现编程。

5.1 基本概念

顺序功能图用约定的几何图形、有向线段和简单的文字来说明和描述 PLC 的处理过程及程序的执行步骤。SFC 的基本元素有 3 个："步"、"路径"和"转换"，主要组成部分有步、转换、转换条件、路径（即有向连线）和动作（或命令）等。图 5-1 给出了一个较为典型的顺序功能图的例子。

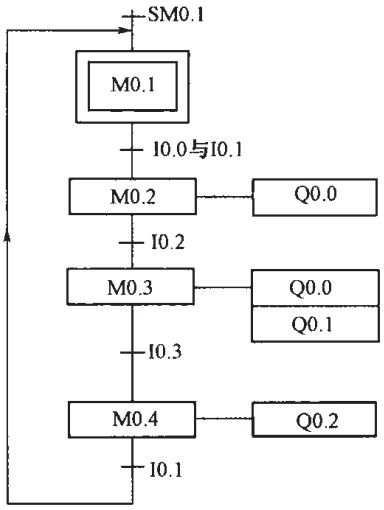


图 5-1 顺序功能图

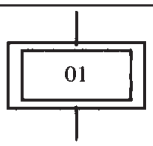
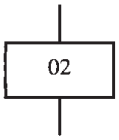
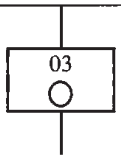
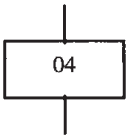
1. 步

步（Step）是顺序功能图中最基本的组成部分，是将一个工作周期分解为若干个顺序相连而清晰的阶段，对应一个相对稳定的状态。步用编程元件（如标志位存储器 M 和顺序控制继电器 S）代表，其划分的依据是 PLC 输出量的变化。在任何一步内，输出量的状态应保持不变，但两步之间的转换条件满足时，系统就由原来的步进入新的步。一个步可以是动作的开始、持续或结束，且步数划分得越多，过程描述得越精确。

（1）步的表示方法及其两种状态

步用一个带步编号的矩形方框表示，有时步编号也可用相应的编程元件的元件号表示，如表 5-1 所示。

表 5-1　步的图形符号

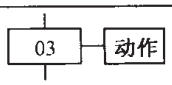
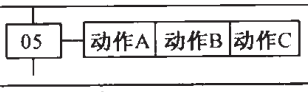
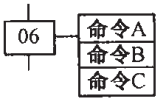
图 形 符 号	说　明
	初始步用带步编号的双线框表示
	步的一般编号，矩形的长宽比任意，步必须有编号，如图中的 2 表示步 2
	在步的图形符号中添加一个小圆表示该步是活动步，如图中的步 3 是活动步
	在步的图形符号中没有小圆，表示该步是非活动步（仅用于分析时），如图中的步 4 是非活动步

步有两种状态：活动态和非活动态。在某一时刻，某一步可能处于活动态，也可能处于非活动态。当步处于活动态时称其为“活动步”，与之相对应的命令或动作将被执行；与初始状态相对应的活动步称为“初始步”，每个顺序功能图中至少应该有一个“初始步”；某步处于非活动态时为“静止步”，相应的非存储型动作被停止执行。某步的状态用二进制逻辑值 X_*表示，例如 X_2=0 表示步 2 是静止步，X_3=1 表示步 3 是活动步。

（2）与步相应的动作或命令

一个控制系统可以分为施控系统和被控系统，施控系统发出一个或数个“命令（Command）”，而被控系统则执行相应的一个或数个“动作（Action）”，在活动步阶段这些动作或命令被执行。在功能图中，动作或命令用矩形框内的文字或符号表示，该矩形框与相应的步的图形符号相连。一个步可以同时与多个动作或命令相连，这些动作可以水平布置或垂直布置，如表 5-2 所示。这些动作或命令是同时执行的，没有先后之分。

表 5-2　动作与步相连的画法

图 形 符 号	说　明
	与步相对应的动作或命令用矩形框内的文字或符号表示。矩形框与步的图形符号用短线连接
	多个动作或命令与同一步相连，采用水平布置
	多个动作或命令与同一步相连，采用垂直布置

动作或命令的类型有很多种，如定时、延时、脉冲、存储型和非存储型等。当某步活动时，若与某步相连的动作或命令被执行，而该步静止时，此动作或命令返回到该步活动前的状态，此动作或命令类型是非存储型。若某动作在与之相连的步成为静止步时依然保持它在

该步为活动步时的状态，则此动作或命令类型为存储型。存储型动作或命令被后续的步激励并复位，仅能返回它的原始状态。动作或命令说明语句应正确选用以明确表明该动作或命令是存储型还是非存储型，且正确的说明语句还可区分动作与命令之间的差别。

由以上语句说明的动作或命令称为公用动作或命令，此外，使用动作或命令的修饰词可以在一步中完成不同的动作或命令，如表 5-3 所示。修饰词允许在不增加逻辑的情况下控制动作或命令。

表 5-3 动作或命令的修饰词

修 饰 词	类 型	说 明
N	非存储型	当步变为静止步时动作终止
R	复位	被修饰词 S、SD、SL 或 DS 启动的动作被终止
S	置位（存储）	当步变为静止步时动作继续，直到动作被复位
L	时间限制	步变为活动步时动作被启动，直到步变为静止步或设定时间到
P	脉冲	当步变为活动步，动作被启动并且只执行一次
D	时间延迟	步变为活动步时延迟定时器被启动，如果延迟之后步仍然是活动的，动作被启动和继续，直到步变为静止步
SL	存储与时间限制	步变为活动步时动作被启动，--直到设定的时间到或动作被复位
DS	延迟与存储	在延迟之后步仍然是活动的，动作被启动直到被复位
SD	存储与时间延迟	在时间延迟之后动作被启动，一直到动作被复位

2. 有向连线

有向连线表示步与步之间进展的路线和方向，也表示各步之间的连接顺序关系，有向连线也称为路径。有向连线的方向可以是水平的，也可以是垂直的，有时也可以用斜线表示。由于 PLC 的扫描顺序遵循从上到下、从左至右的原则，按照此原则发展的路线可不必标出箭头，如果不遵循上述原则，必须加箭头。在可以省略箭头的有向连线上，为了更易于理解也可以加箭头。如果垂直线和水平线没有内在的联系则允许它们交叉，否则不允许交叉。在复杂图或几张图中表示而使用相连线必须中断时，应在中断点处指明下一步的标号或来自上一步的编号和所在的页号，如步 100、10 页。

有向连线可分为选择和并行两种。选择连线间的关系是逻辑“或”的关系，哪条连线转换条件最先得到满足，这条连线就被选中，程序就沿着这条线往下执行。选择连线的分支与合并一般用单横线表示。并行路线间的逻辑关系是“与”的关系，只要转换条件满足，下面所有连线必须同时执行。并行连线的分支与合并一般用双横线表示。

3. 转换

步与步之间用有向连线连接，表示上一步活动结束与下一步活动开始的顺序连接关系。而两个步之间绝对不能直接相连，必须用一个转换隔开。转换是结束某一步的操作而起动下一步操作的条件，这种条件是各种控制信号综合的结果。步的活动状态的进展由转换的实现来完成，并与控制过程的发展相对应。转换在功能图中用与有向连线垂直的短横线表示，两个转换不能直接相连，必须用一个步隔开。转换也称变迁或过渡。图 5-2 中共有 4 种转换条件。

（1）转换条件

使系统由当前步进入下一步的信号称为转换条件，转换条件可以是外部的输入信号，也可以是 PLC 内部产生的信号，还可能是若干个信号与、或、非逻辑的组合。转换条件是与转换相关的逻辑命题。转换条件的表达形式有文字符号、布尔代数表达式、梯形图符号和二进制逻辑图符号 4 种，它们标注在转换的短线旁边（如图 5-2（a）所示）。使用最多的是布尔代数表达式。

转换条件 I1.0 和 $\overline{I1.0}$ 分别表示当二进制逻辑输入信号 I1.0 为“1”和“0”状态时，转换实现。符号 ↑I1.0 和 ↓I1.0 分别表示当 I1.0 从 0→1 和从 1→0 状态时转换实现。一般“↑”不用画出来。

图 5-2（b）中用高电平表示步 10 是活动步。转换条件 $I0.1 \cdot \overline{I0.2}$ 表示 I0.1 的常开触点和 I0.2 的常闭触点串联且 I0.1 的常开触点闭合和 I0.2 的常闭触点断开后转换发生。

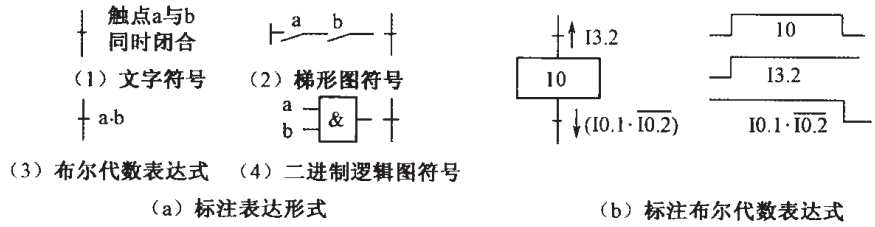


图 5-2　转换与转换条件

列写转换条件时应避免所有的冲突发生，例如两个转换以同一步为条件使能时，与这两个转换相连的转换条件应该保证是互斥的，即不能同时为真。

（2）转换的使能和实现

根据步的状态不同，转换可分为使能转换和非使能转换两种。所谓使能转换是指与某一转换相连的所有前级步都为活动步，如图 5-3（a）所示；否则此转换为“非使能转换”，如图 5-3（b）所示。如果某转换是使能转换，且该转换相应的转换条件满足，则此转换为“实现转换”，也称为“触发（Firing）”，如图 5-3（c）所示，即所有前级步变为静态步，而与此转换相连的后续步变为活动步，如图 5-3（d）所示；如果转换的前级步或后续步不止一个，实现转换称为同步转换，为了强调同步实现，有向连线的水平部分用双线表示，如图 5-3（e）所示。因此要实现一个转换必须要满足两个条件：①该转换为使能转换；②相应的转换条件全部满足，这两个条件缺一不可。

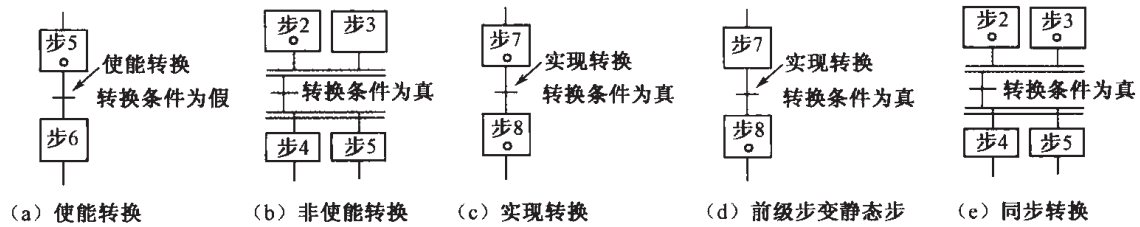


图 5-3　转换的使能

只有当某步的前级步是活动步时，该步才有可能变为活动步。代表各步的编程元件没有断电保持功能，则系统进入工作模式后因没有活动步而无法进行，因此为了使初始步是活动步，应采用 PLC 上电时的内部常闭触点启动初始步，并采用初始化脉冲 SM0.1 的常开触点

作为转换条件。如果系统有自动和手动两种工作方式，还应在系统由手动转入自动时用一个适当的信号将初始步置为活动步。

步、有向连线、转换的关系为：步经有向连线连接到转换，转换经有向连线连接到步。为了能在全部操作完成后返回初始状态，步和有向连线应构成一个封闭的环状结构。当工作方式为连续循环时，最后一步应该能够回到下一个流程的初始步，也就是循环不能够在某步被终止。

5.2 结构形式

根据功能图中序列有无分支及实现转换的不同，功能图的基本结构形式有 3 种：单序列、选择序列和并行序列，其他结构都是这 3 种结构的复合。

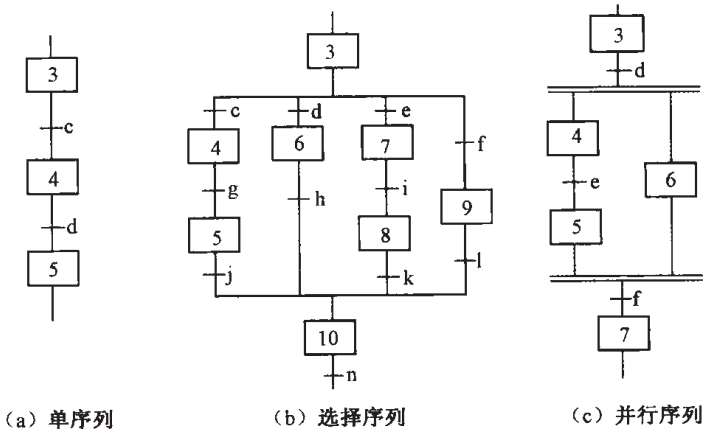
1. 单序列

如果一个序列中各步依次变为活动步，此序列称为单序列。在此结构中，每一步后面仅有一个转换，而每个转换后面也仅有一个步，如图 5-4（a）所示。

2. 选择序列

选择序列是指在某一步后有若干个单序列等待选择，一次只能选择一个序列进入。选择序列的开始部分称为分支，转换符号只能标在选择序列开始的水平线之下，如图 5-4（b）上半部所示。如果步 3 是活动步，当转换条件 d=1 时，则步 3 进展为步 6。与之类似，步 3 也可以进展为步 4，但是一次只能选择一个序列。

选择序列的结束称为合并，如图 5-4（b）下半部所示。几个选择序列合并到一个公共序列上时，用一条水平线和与需要重新组合序列数量相同的转换符号表示，转换符号只能标在结束水平线的上方。



（a）单序列　　（b）选择序列　　（c）并行序列

图 5-4　单序列、选择序列和并行序列

3. 并行序列

并行序列指在某一转换实现时，同时有几个序列被激活，也就是同步实现，这些同时被激活的序列称为并行序列。并行序列表示的是系统中同时工作的几个独立部分的工作状态。

并行序列的开始称为分支，如图 5-4（c）上半部所示，当步 3 是活动的，且 d=1 时，4、6 这两步同时变为活动步，而步 3 变为静止步。转换符号只允许标在表示开始同步实现的水平线上方。

并行序列的结束称为合并，如图 5-4（c）下半部所示。转换符号只允许标在表示合并同步实现的水平线下方。并行序列的活动和静止可以分成一段或几段实现。

在每一个分支点，最多允许 8 条支路，每条支路的步数不受限制。

4．跳步、重复和循环

（1）跳步：在生产过程中，有时要求在一定条件下停止执行某些原定动作，可用图 5-5（a）所示的跳步序列。这是一种特殊的选择序列，当步 1 为活动步时，若转换条件 f=1，b=0 时，则步 2、3 不被激活而直接转入步 4。

（2）重复：在一定条件下，生产过程需重复执行某几个工步的动作，可按图 5-5（b）绘功能图。它也是特殊的选择序列，当步 4 为活动步时，若转换条件 e=0 而 h=1 时，序列返回到步 3，重复执行步 3、4，直到转换条件 e=1 才转入步 7。

（3）循环：在序列结束后，用重复的办法直接返回到初始步，就形成了系统的循环，如图 5-5（c）所示。

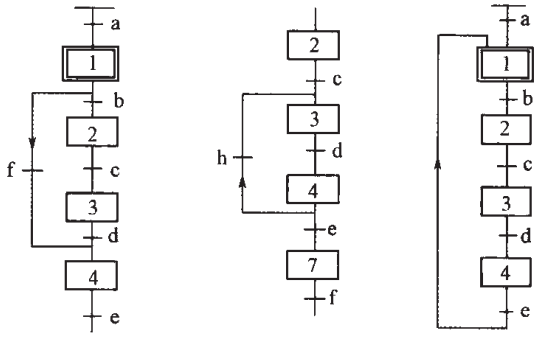


图 5-5　跳步、重复和循环

5．复合序列

复合序列是集单序列、选择序列、并行序列、循环序列于一体的一种序列。复合序列比较复杂，在描述实际问题时要仔细。

5.3　顺序功能图的编程方法及梯形图表示

根据顺序控制系统的功能要求，可以采用不同的方法设计出顺序功能图，然后可以很方便地将功能图转化为 PLC 的梯形图。为了便于将顺序功能图转化为梯形图，一般将步的代号及转换条件和各步的动作与命令用代表各步的编程元件的地址（如 M0.1）表示。

系统处于初始状态时，与初始步对应的编程元件应置为 1，而其他的编程元件应置为 0，因为在没有并行序列或并行序列未处于活动状态时，只能有一个活动步。在下面所讲述的各种方法中，假设程序开始时，系统已处于要求的初始状态下，且其余各步的编程元件均为 0

状态。初始步的转换利用初始化脉冲 SM0.1 触发。

5.3.1 使用通用逻辑指令的方法

所谓通用逻辑指令，是指与 PLC 的触点和输出线圈相关的指令，如 AN、O、=等，它是 PLC 最基本的指令。这种编程方法适用于各种型号的 PLC，是顺序功能图最基本的编程方法。

在顺序控制中，各步按照顺序先后接通和断开，犹如电动机按顺序地接通和断开，因此可以像处理电动机的启动、保持、停止那样，用典型的启保停电路解决顺序控制的问题。

（1）单序列的编程

根据功能图理论，设步 M_i 的前级步是活动的（即 M_{i-1}=1），且转换条件成立（即 I_i=1），步 M_i 应变为活动步。如果将 M_i 视为电动机，而 M_{i-1} 和 I_i 视为其启动开关，则 M_i 的启动电路由 M_{i-1} 和 I_i 的常开触点串接而成（如图 5-6 所示）。I_i 一般为非存储型触点，所以还要用 M_i 的常开触点实现自锁。同样，当 M_i 的后续步 M_{i+1} 变为活动步时，M_i 应变为静态步，因此应将 M_{i+1} 的常闭触点与 M_i 的线圈串联。下面以冲床动力头进给运动控制为例介绍单序列的编程。

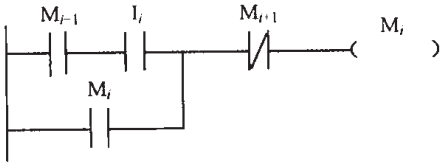


图 5-6 顺序控制“基本电路”

实例 70：冲床动力头进给运动控制

实例说明

图 5-7 是某一专用冲床动力头的进给运动示意图，系统的一个周期分为快进、工进和快退 3 步，另外还设置有一个等待启动的初始步。动力头初始状态停留在最左边，限位开关 I0.1 状态为 1。启动按钮为 I0.0，Q0.0～Q0.2 控制 3 个电磁阀，这 3 个电磁阀依次控制快进、工进和快退 3 步。按下启动按钮，动力头的运动如图 5-7 所示，工作一个循环后，动力头返回并停留在初始位置。该实例可用通用逻辑指令，置位、复位（S、R）指令和顺序控制 SCR 指令的方法编程。下面首先用通用逻辑指令方法编程实现。

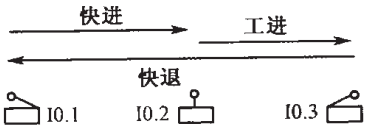


图 5-7 冲床动力头进给运动示意图

实例实现

图 5-8（a）是系统的功能图，根据功能图及上述的编程方法，可以很容易得到系统的梯形图，如图 5-8（b）所示。对于步 M0.0，设 M_i= M0.0，由功能图可知，M_{i-1}= M0.3，I_i=I0.1，M_{i+1}=M0.1，所以将 M0.3 和 I0.1 的常开触点串联作为 M0.0 的启动电路。在启动电路中还并联了 M0.0 的自保持触点。后续步 M0.1 的常闭触点串入 M0.0 的线圈，M0.1 接通时 M0.0 断开。在 PLC 开始运行时应将 M0.0 置为 1，否则系统无法工作，因此把仅在第一个扫描周期接通的 SM0.1 的常开触点与上述电路并联。

在功能图中，步划分的依据是输出量的变化，因此步与输出量的关系也较为简单。如果某一输出量仅在某一步中有输出，例如 Q0.2 仅在步 M0.3 中输出，此时可以将其线圈与对应步的存储器位 M0.3 的线圈并联。而在几步中有同一输出时，例如 Q0.1 在步 M0.1 和 M0.2 中均输出，为避免双线圈输出，采用 M0.1 和 M0.2 的常开触点并联后驱动 Q0.1。

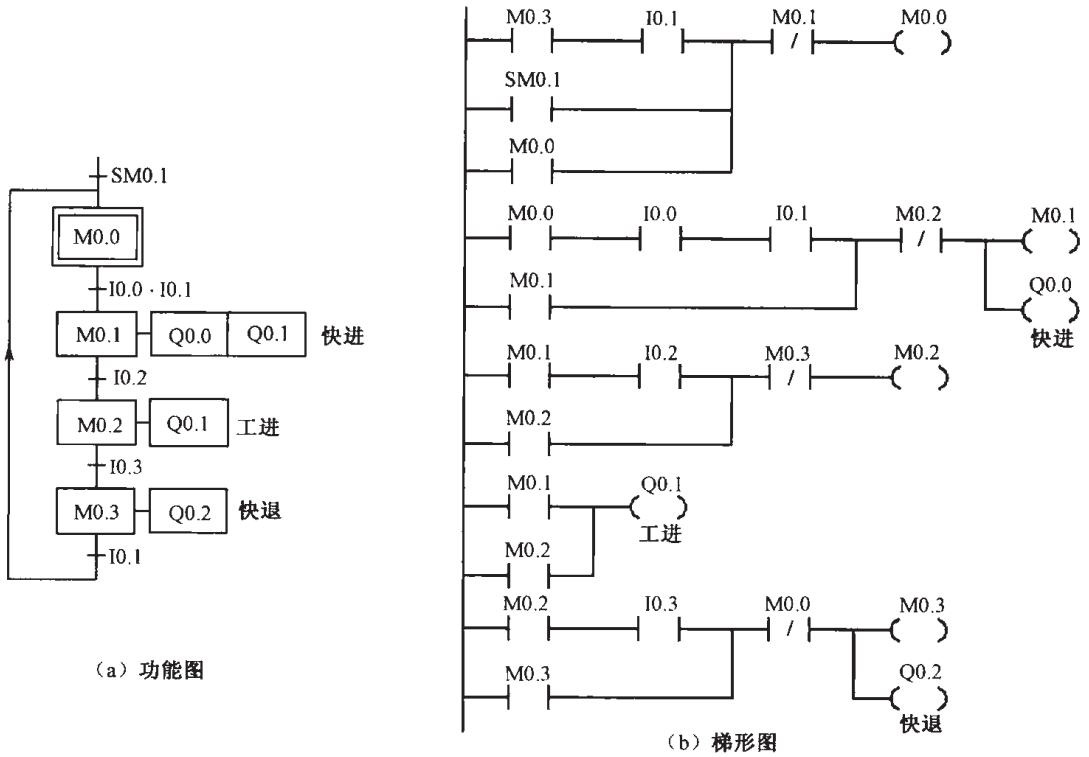


图 5-8　动力头功能图及梯形图

实例分析

该实例介绍了用通用逻辑指令的方法将单序列功能图转换为梯形图程序的编程方法及原理，也可用置位、复位（S、R）指令（见 5.3.2 单序列部分）和顺序控制 SCR 指令（见 5.3.3 单序列部分）等方法将功能图转换为梯形图程序的编程方法，注意比较它们的异同。

（2）选择序列的编程

选择序列编程的关键在于对其分支和合并的处理，转换实现的基本规则是设计复杂系统梯形图的基本规则。下面以自动门控制系统为例介绍选择序列中的分支与合并编程。

实例 71：自动门控制系统

实例说明

自动门控制系统实例以选择序列顺序功能图转换为梯形图程序，选择序列编程时重点是对分支与合并编程的处理，该实例可用通用逻辑指令，置位、复位（S、R）指令和顺序控制 SCR 指令的方法编程。下面先用通用逻辑指令编程。

实例实现

① 分支的编程。

如果某一步的后面有一个由 N 条分支组成的选择序列，该步可能转到不同的 N 步中去，应将这 N 个后续步对应的内部标志位存储器的常闭触点与该步的线圈串联，作为结束该步的条件。

图 5-9 是自动门控制系统的顺序功能图。人靠近自动门时，感应器 I0.0 为 ON，Q0.0 变为 ON，驱动电动机正转高速开门，碰到开门减速开关 I0.1 时，Q0.1 变为 ON，减速开门。碰到开门极限开关 I0.2 时电动机停转，开始延时。若在 1 s 内感应器检测到无人，Q0.2 变为 ON，启动电动机反转高速关门。碰到关门减速开关 I0.3 时，Q0.3 变为 ON，改为减速关门，碰到关门极限开关 I0.4 时电动机停转。在关门期间若感应器检测到有人停止关门，T38 延时 1 s 后自动转换为高速开门。

步 M0.4 之后有一个选择序列的分支，当它的后续步 M0.5、M0.6 变为活动步时，它应变为不活动步。所以需将 M0.5 和 M0.6 的常闭触点与 M0.4 的线圈串联。同样 M0.5 之后也有一个选择序列的分支，处理方法同 M0.4（如图 5-10 所示）。

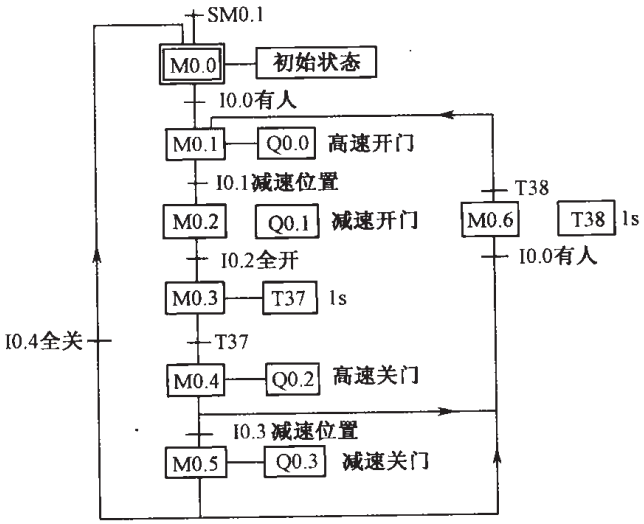


图 5-9 自动门控制系统顺序功能图

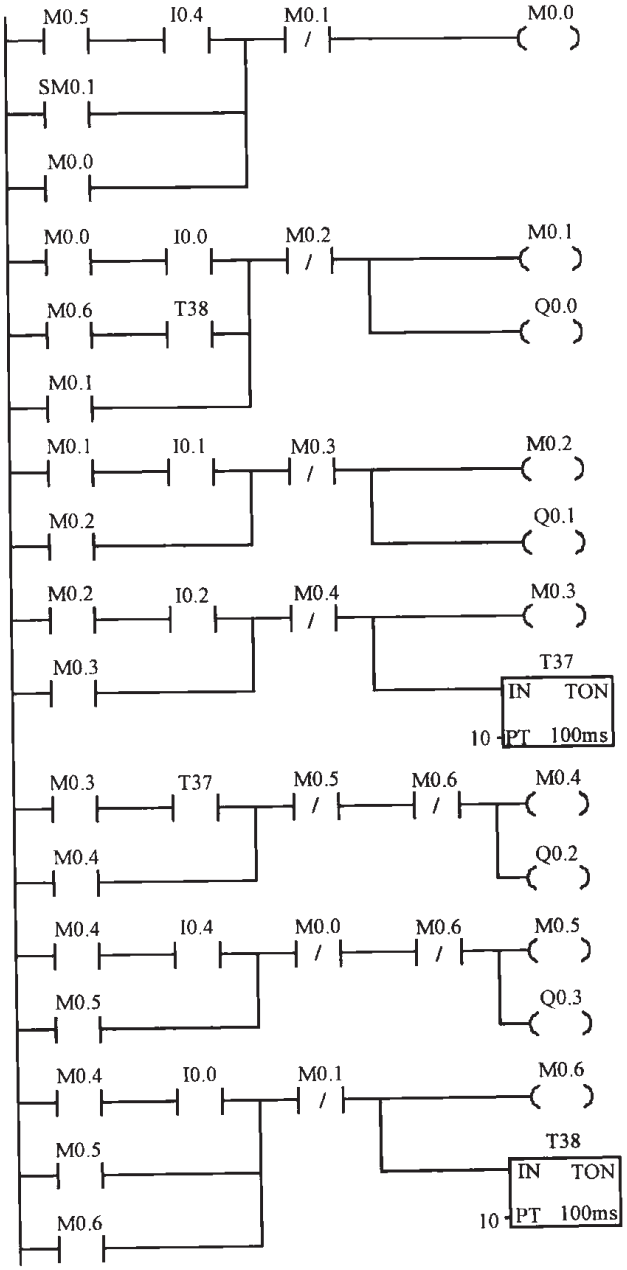


图 5-10 自动门控制系统梯形图

② 合并的编程

对于选择序列的合并，如果每一步之前有 N 个转换（即有 N 条分支在该步之前合并后进入该步），则代表该步的内部标志位存储器 M 的启动电路由 N 条支路并列而成，各支路由某一前级步对应的内部标志位存储器的常开触点与相应转换条件对应的触点或电路串联而成。

在图 5-9 中，步 M0.1 之前有一个选择序列的合并，当步 M0.0 为活动步并且转换条件 I0.0 满足，或 M0.6 为活动步，并且转换条件 T38 满足时，步 M0.1 都应变为活动步，即控制 M0.1 的启动、保持、停止电路的启动条件应为 M0.0 和 I0.0 的常开触点串联电路与 M0.6 和 T38 的常开触点串联电路进行并联（如图 5-10 所示）。

实例分析

该实例介绍了用通用逻辑指令的方法，对选择序列编程的分支与合并情形的功能图转换为梯形图程序的编程方法及原理，也可用置位、复位（S、R）指令（见 5.3.2 选择序列部分）和顺序控制 SCR 指令（见 5.3.3 选择序列部分）等方法将功能图转换为梯形图程序的编程方法，注意比较它们在分支与合并处理上的特点。

（3）并行序列的编程

并行序列编程与选择序列编程相类似，其关键也是对其分支和合并的处理。下面以专用钻床部分控制程序为例介绍并行序列中的分支与合并编程。

实例 72：专用钻床部分控制程序

实例说明

专用钻床部分控制程序实例以并行序列顺序功能图转换为梯形图程序，并行序列编程时重点与选择序列编程相同，也是对分支与合并编程的处理，该实例可用通用逻辑指令，置位、复位（S、R）指令和顺序控制 SCR 指令的方法编程。下面先用通用逻辑指令编程。

实例实现

① 分支的编程。

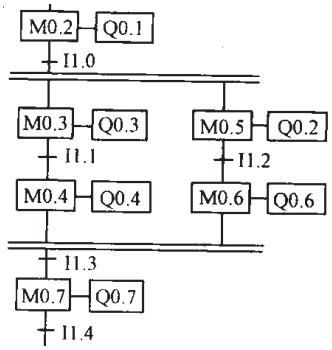
某并行序列某一步 M_i 的后面有 N 条分支，如果转换条件成立，并行序列中各单序列中的第一步应同时变为活动步，对控制这些步的启动、保持、停止电路使用相同的启动电路，实现这一要求，只需将 N 个后续步对应的软继电器的常闭触点中的任意一个与 M_i 的线圈串联，作为结束步 M_i 的条件。

如图 5-11（a）所示是专用钻床部分控制系统的部分控制程序顺序功能图，图中 M0.2 之后有一个并行序列的分支，当步 M0.2 为活动步，并且转换条件 I1.0=1 时，步 M0.3 和步 M0.5 同时变为活动步，即 M0.2 和 I1.0 的常开触点串联电路同时作为控制步 M0.3 和步 M0.5 的启动电路，如图 5-11（b）所示。

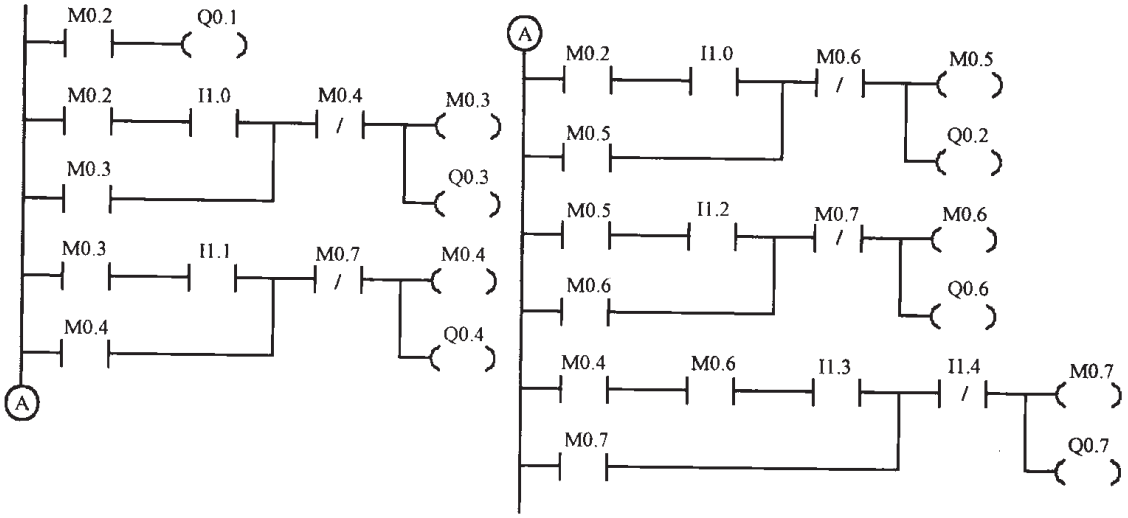
② 合并的编程。

当并行序列合并时，只有当各并行序列的最后一步都是活动步，且转换条件成立时，才能完成并行序列的合并。因此合并后的步的启动电路应有 N 条并联支路中最后一级步的软继

电器的常开触点与相应转换条件对应的电路串联而成。而合并后的步的常闭触点分别作为各并行序列的最后一步断开的条件。



（a）功能图



（b）梯形图

图 5-11　专用钻床部分控制程序功能图及梯形图

图 5-11（a）中步 M0.6 之前有一个并行序列的合并，该转换实现的条件是所有的前级步（即步 M0.4 和步 M0.6）都是活动步且转换条件 I1.3=1 满足。由此可知，应将 M0.4、M0.6 和 I1.3 的常开触点串联，作为控制步 M0.6 的启动电路，如图 5-11（b）所示。

实例分析

该实例介绍了用通用逻辑指令方法对并行序列编程的分支与合并情形的功能图转换为梯形图程序的编程方法及原理，也可用置位、复位（S、R）指令（见 5.3.2 并行序列部分）和顺序控制 SCR 指令（见 5.3.3 并行序列部分）等方法将功能图转换为梯形图程序的编程方法，注意比较并行序列编程与选择序列编程在分支与合并处理上的特点与异同。

5.3.2　使用置位、复位（S、R）指令的方法

几乎各种型号的 PLC 都有置位、复位（S、R）指令或相同功能的编程元件。使用通用逻辑指令实现的顺序功能控制同样也可以利用 S、R 指令实现。下面介绍使用 S、R 指令的以转换条件为中心的编程方法。

所谓以转换条件为中心，是指同一种转换在梯形图中只能出现一次，而对辅助存储器位可重复进行置位、复位。其编程思路为：设步 M_i 是活动的（即 M_i=1），且其后的转换条件成立（即 I_{i+1}=1），则步 M_i 应被复位，而后续步 M_{i+1} 应被置位（接通并保持）。因此可将 M_i 的常开触点和 I_{i+1} 对应的常开触点串联用作 M_i 复位和 M_{i+1} 置位的条件，该串联电路即为通用逻辑电路中的启动电路。而置位、复位则采用置位、复位指令。在任何情况下，代表步的存储器位的控制电路都可以用这一方法设计，每一个转换对应一个这样的控制置位和复位的电路块，有多少个转换就有多少个这样的电路块。这种方法特别有规律，梯形图与实现转换的基本规则之间有着严格的对应关系。用于复杂功能图的梯形图设计时不容易遗漏和出错。

（1）单序列编程

采用置位、复位（S、R）指令方法重新设计实例 69 冲床动力头进给运动控制，如图 5-7 所示冲床动力头的梯形图如图 5-12 所示。以步 M0.2 为例，如果步 M0.2 要实现转换，必须满足两个条件，首先 M0.2 是活动步，即 M0.2=1，其次为转换条件满足，即 I0.3=1。在梯形图中，可用 M0.2 和 I0.3 的常开触点组成串联电路表示上述条件。当电路接通时，两个条件同时满足。此时应将该转换的后续步变为活动步，即用置位指令“S M0.3,1”将 M0.3 置位；同时还应使用复位指令“R M0.2,1”将 M0.2 复位，使之变为不活动步。

控制置位、复位指令的串联电路只有一个扫描周期的接通时间，转换条件满足后前级步马上被复位，从而断开了此串联电路，而输出线圈至少应在某一步对应的全部时间内接通，因此不能将输出线圈与置位、复位指令并联，只能用代表步的存储器位的常开触点或它们的并联电路来驱动线圈。

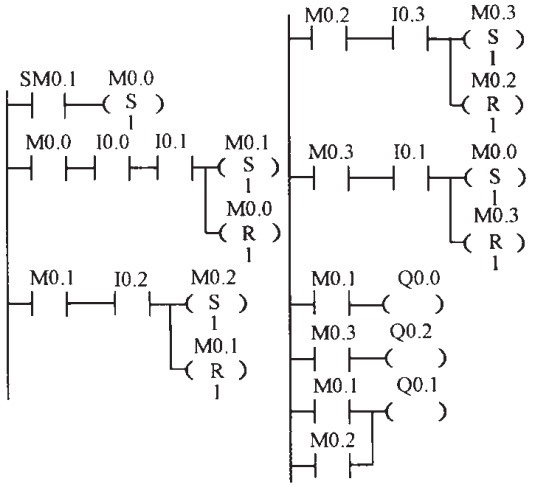


图 5-12　以转换为中心的冲床动力头梯形图

（2）选择序列编程

选择序列的分支与合并的编程与单序列的完全相同，除了与合并序列有关的转换以外，每一个控制置位、复位的电路块都由前级步对应的存储器位的常开触点和转换条件对应的触点组成的串联电路、一条置位指令和一条复位指令组成。

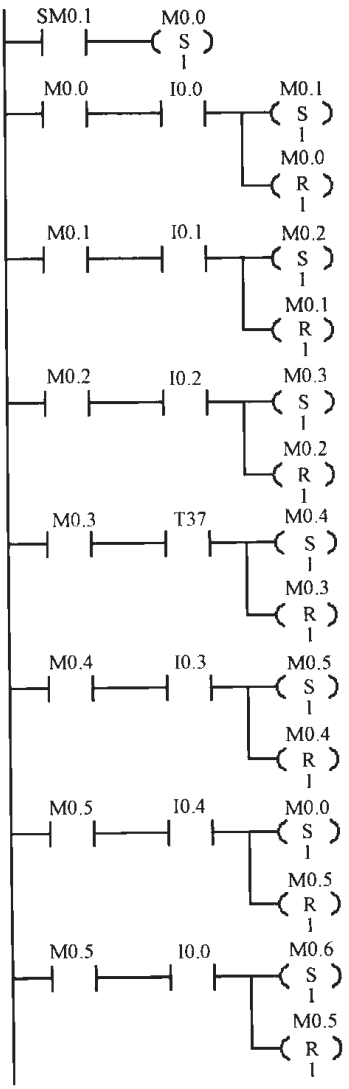
在实例 71 自动门控制顺序功能图中，I0.0，I0.3，I0.4 对应的转换与选择序列的分支、合并有关，它们的前级步和后续步都只有一个，其梯形图如图 5-13 所示。

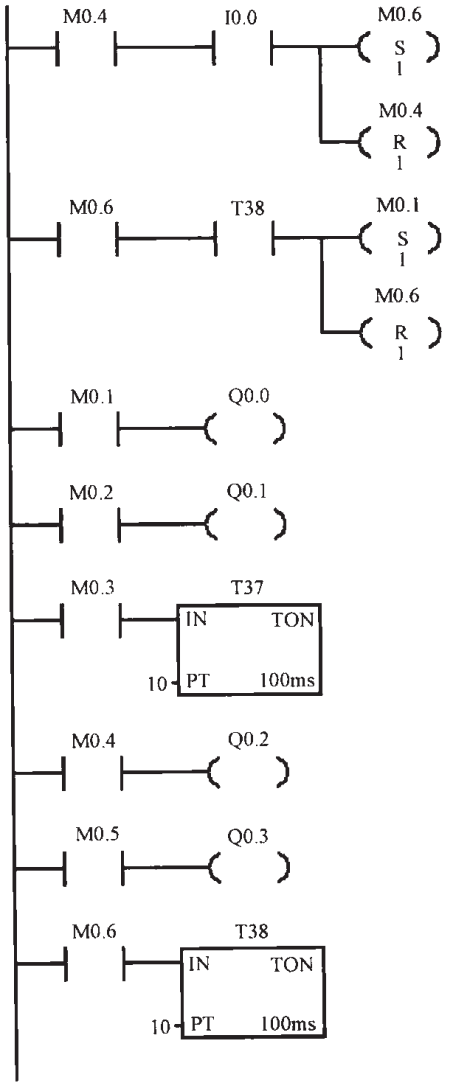
（3）并行序列编程

对于并行序列的分支，仍然是用前级步和转换条件对应的触点组成串联电路，只不过需

要置位的后续步的存储器位不止一个。将图 5-11（a）专用钻床部分控制程序的功能图利用置位、复位指令得到的梯形图如图 5-14 所示，当 M0.2 为活动步，转换条件满足，则步 M0.3 和 M0.5 应同时变为活动步，其实现是将 M0.2 和 I0.1 对应的常开触点组成串联电路，然后使 M0.3 和 M0.5 同时置位；置位的同时，还应使用复位指令使步 M0.2 变为静态步。图 5-14 是利用置位复位指令得到的梯形图，当 M0.2 为活动步，转换条件满足，则步 M0.3 和 M0.5 应同时变为活动步，其实现是将 M0.2 和 I0.1 对应的常开触点组成串联电路，然后使 M0.3 和 M0.5 同时置位；置位的同时，还应使用复位指令使步 M0.2 变为静态步。

对于并行序列的合并，用转换的所有前级步对应的存储器位的常开触点和转换对应的触点组成串联电路，驱动相应步的置位和复位，这时被复位的步的个数与并行序列的个数相同。I1.3 对应的转换之前有一个并行序列的合并，根据所讲述的方法，应将 M0.4、M0.6 和 I1.3 的常开触点串联，作为使后续步 M0.7 置位和 M0.4 和 M0.6 复位的条件。





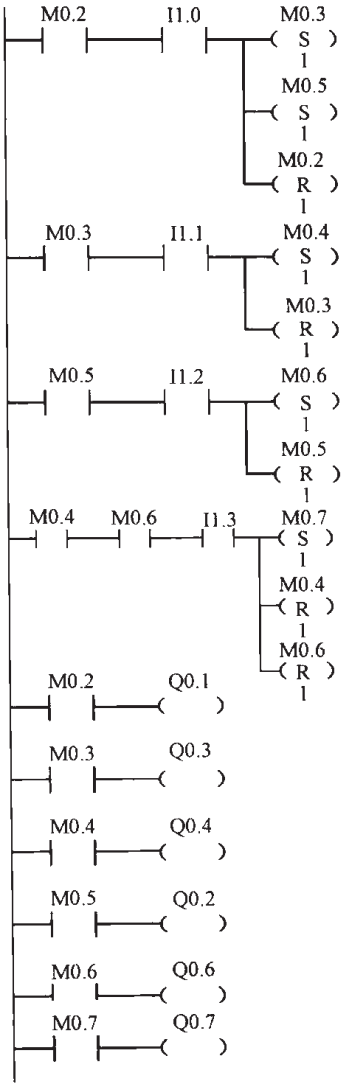


图 5-13　以转换为中心的自动门梯形图　　图 5-14　以转换为中心的专用钻床部分程序梯形图

5.3.3　使用 SCR 指令的方法

顺序控制指令是 PLC 生产厂家为用户提供的可使功能图编程简单化和规范化的指令。S7-200 PLC 提供了 3 条顺序控制指令 LSCR,SCRT,SCRE，LSCR 与 SCRE 之间的全部逻辑组

成顺序控制程序段。每一个 SCR 程序段一般有 3 种功能：

① 驱动处理：即在该段状态器有效时，处理相应的工作；有时也可能不做任何工作。

② 指定转移条件和目标：即满足什么条件后状态转移到何处。

③ 转换源自动复位功能：状态发生转换后，置位下一个状态的同时，自动复位原状态。

（1）单序列的编程

这里仍以实例 70 冲床动力头进给运动控制为例介绍单序列的 SCR 编程方法。在 SCR 段中，用转换条件对应的触点或电路驱动转换到后续步的 SCRT 指令，用 SM0.0 的常开触点驱动在该步中有输出的线圈。利用 SCR 指令编写的梯形图如图 5-15 所示。

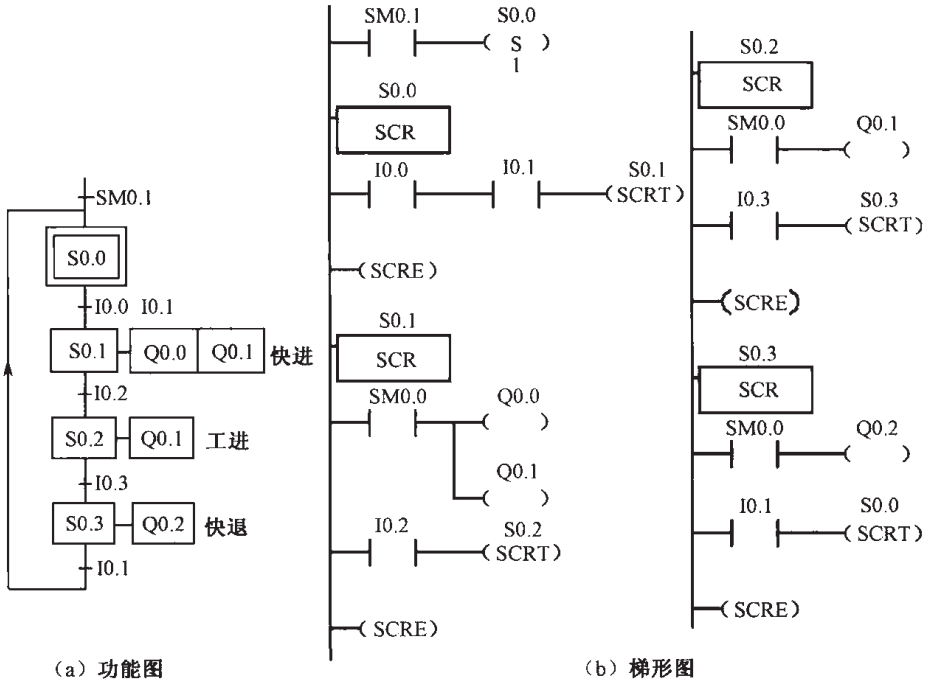
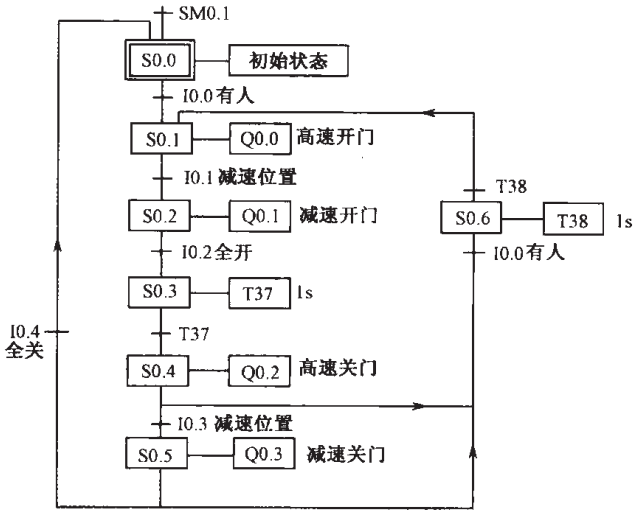


图 5-15　冲床动力头进给运动控制功能图及 SCR 指令编写的梯形图

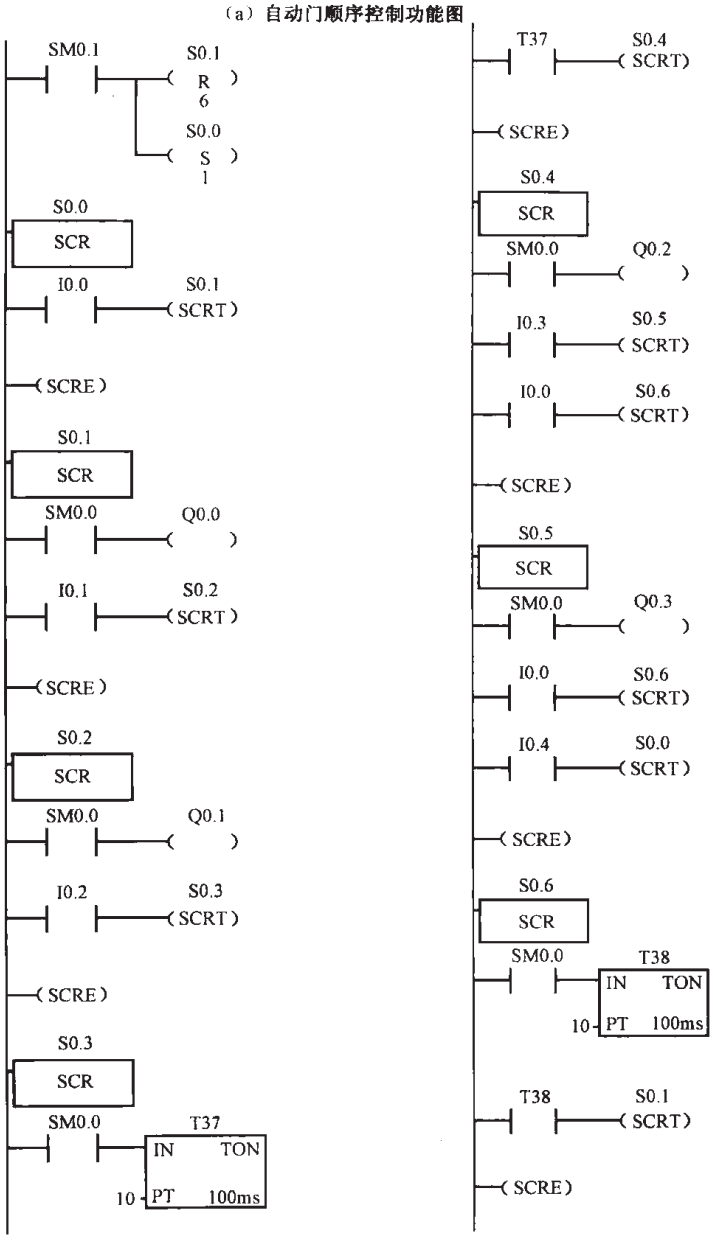
首先初始步 S0.0 由 SM0.1 置位变为活动步，在 SCR 段中，只有与 S0.0 相对应的那一段被执行。在初始状态下，动力头在最左面，I0.1=1，按下启动按钮 I0.0，则指令“SCRT S0.1”执行，使 S0.1 置位，以便让 S0.1 的 SCR 程序段执行，同时使 S0.0 变为 0 状态，即从初始步转换为快进步。在 S0.1 的 SCR 程序段中，SM0.0 的常开触点闭合，线圈 Q0.0 得电，动力头向右快进。当碰到减速开关 I0.2 时，I0.2=1，则指令“SCRT S0.2”执行，将实现快进步到工进步的转换。直到碰到右限位开关，I0.3=1，则指令“SCRT S0.3”执行，动力头快退，直到返回初始步。

（2）选择序列的编程

自动门控制系统的顺序功能图如图 5-16（a）所示，其 SCR 梯形图如图 5-16（b）所示。在梯形图中，分支在其前级步的 SCR 程序段中表示有几条分支，就有几条支路，每条支路由转换条件对应的触点和 SCRT 指令串联而成，当前级步为活动步，哪条分支的转换条件满足，转换就向哪条分支步发展。在自动门控制系统的顺序功能图中，S0.4 后有一个分支，当它为活动步，且 I0.3=1 时，则后续步 S0.5 变为活动步，S0.4 变为静态步。如果步 S0.4 为活动步，且 I0.0=1 时，则后续步 S0.6 变为活动步，S0.4 变为静态步。



（a）自动门顺序控制功能图



（b）SCR梯形图

图 5-16　自动门顺序控制功能图及 SCR 梯形图

对选择序列的合并来说，在每一个选择序列的最后一步对应的 SCR 程序段中，分别用各自的转换条件所对应的触点驱动指令“SCRT Sn”，其中 Sn 为实现转换合并后的第一步，这样就可以实现选择序列的合并。在图 5-16（a）中，步 S0.1 前有一个选择序列的合并。当 S0.0 为活动步，且转换条件 I0.0=1 时；或 S0.6 为活动步，且延时时间 T38 为 1s 时，步 S0.1 都应变为活动步。因此，在 S0.0 和 S0.6 对应的 SCR 程序段中，分别用 I0.0 和 T38 的常开触点驱动指令“SCRT S0.1”就能实现选择序列的合并。

（3）并列序列的编程

当并行序列有分支时，在梯形图中的实现是在并行序列前级步所对应的 SCR 段中，利用转换条件所对应的触点同时驱动并行序列各分支首步的 SCRT 指令。图 5-17 中步 S0.2 后是一个并行序列的分支，当 S0.2 为活动步，且转换条件 I0.1=1，步 S0.3 和步 S0.5 同时变为活动步，而 S0.2 变为静态步。因此，在 S0.2 的 SCR 程序段中，用 I0.1 的常开触点同时驱动指令“SCRT S0.3，SCRT S0.5”使步 S0.3 和步 S0.5 同时置位，成为活动步，而步 S0.2 被自动复位，变为静态步。

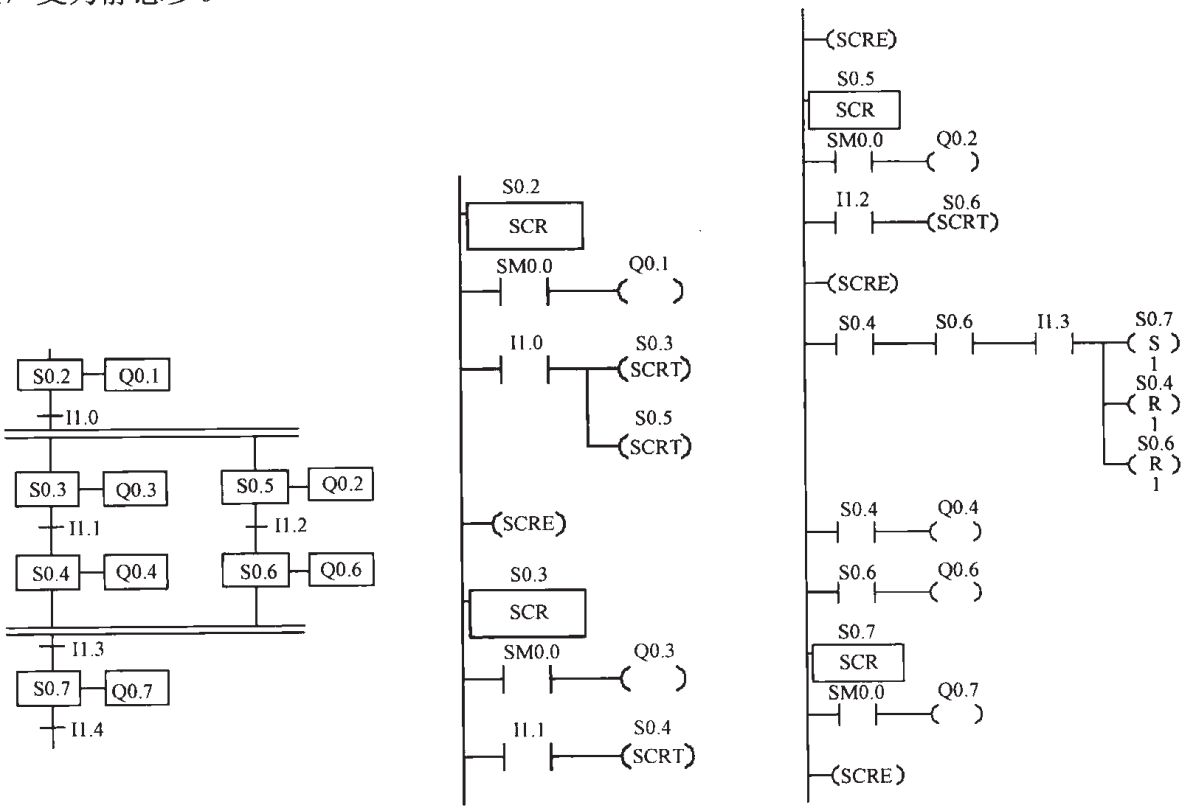


图 5-17　专用钻床部分程序功能图及梯形图

当并行序列合并时，只有当所有前级步为活动步，且转换条件满足时，才同时实现状态转换，完成新状态的启动。对此类情况，一般使用置位、复位的编程方法。图 5-17 中步 S0.7 前是一个并行序列的合并，当步 S0.4 和步 S0.6 同时为活动步，且转换条件 I1.3=1，S0.7 为活动步，而步 S0.4 和步 S0.6 同时变为静态步。因此，将步 S0.4 和步 S0.6 的常开触点与转换条件 I1.3 对应的常开触点串联，来控制对 S0.7 的置位及步 S0.4 和步 S0.6 同时复位，从而 S0.7 成为活动步，而步 S0.4 和步 S0.6 变为静态步。

顺序功能图是根据生产工艺和工序所对应的顺序和时序将控制输出划分为若干个时段，每一个时段对应设备运作的一组动作（步、路径和转换），该动作完成后根据相应的条件转

换到下一个时段完成后续动作，并按系统的功能流程依次完成状态转换。顺序功能图能清晰的反映系统的控制时序和逻辑关系，是可编程序控制器设计顺序控制程序的理想方法。

思考题

1．叙述顺序功能图中“步”、“路径”和“转换”之间的关系。

2．设计出如图 5-18 所示的顺序功能图的梯形图程序。

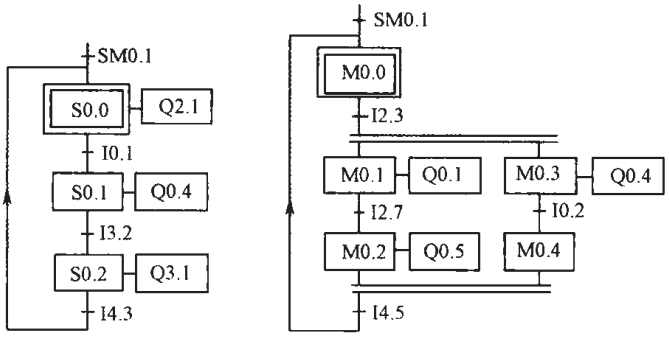


图 5-18　题 2 图

3．小车开始停在左边，限位开关 I0.0 为 1 状态。按下启动按钮后，小车开始右行，以后按图 5-19 所示顺序运行，最后返回并停在限位开关 I0.0 处。画出顺序功能图，并用通用逻辑指令，置位、复位指令和顺序控制指令设计梯形图。

4．设计一个三工位转台，其工作图如图 5-20 所示。3 个工位分别完成上料、钻孔和卸件的任务。工位 1 上料器的动作是推进，料到位后退回等待。工位 2 的动作较多，首先将工料夹紧，然后钻头向下进给钻孔，达到钻孔深度后，钻头退回原位；最后将工件松开，等待。工位 3 上的卸料器将加工完成的工件推出，推出后退回等待。

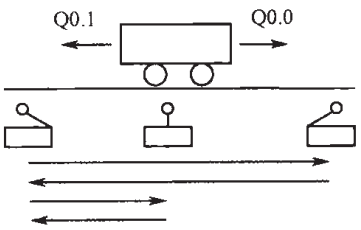


图 5-19　题 3 图

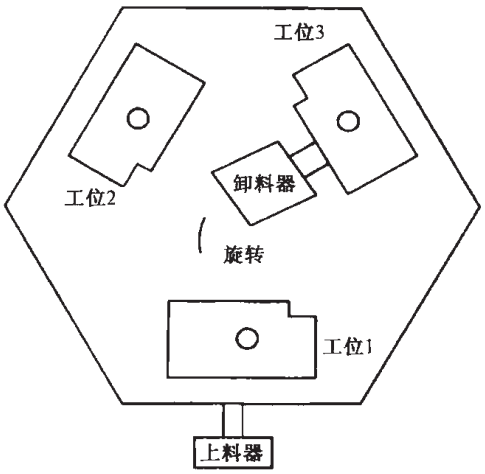


图 5-20　题 4 图

控制系统要求通过选择开关实现自动、半自动和手动操作。

按下启动按钮后，系统开始运行。选择开关处于手动操作时，每个可动部件可以点动调整。如果开关处于自动或半自动位置，且可动部位多在原位，则进入自动或半自动运行。3 个工位同时进行动作，待每个工位全部进行完毕后，工作台旋转 120°，完成一个工作循环，半自动循环结束。用一个启动按钮可以进入下一个半自动循环。如果选择开关处于自动，则自动重复进行，即工作台旋转 120° 后，3 个工位上料、钻孔和卸料再次同时进行。

第 6 章　PLC 控制系统应用

- PLC 控制系统设计的基本原则与步骤
- PLC 系统控制程序设计方法
- PLC 控制系统应用设计

前面各章介绍了西门子 S7-200 PLC 的基本指令和常用扩展模块的应用实例。学习 PLC 的最终目的是把它应用到实际的工业控制系统中去。虽然各种工业控制系统的功能、要求不同，但在设计 PLC 控制系统时，所遵循的设计原则、设计步骤和设计方法基本相同。PLC 控制系统的设计包括 3 个重要的环节：一是通过对控制任务的分析，确定控制系统的总体设计方案；二是根据控制系统要求确定硬件构成方案；三是设计出满足控制要求的控制程序。本章以具体的应用实例对 PLC 控制系统的一般设计原则与步骤和用户程序设计方法进行介绍。要设计出合理、可靠的 PLC 控制系统需要不断地实践，将各种程序设计方法灵活应用，融会贯通。

6.1 PLC 控制系统设计的基本原则与步骤

6.1.1 PLC 控制系统设计的基本原则

任何一种控制系统都是为了实现被控对象的工艺要求，以提高生产效率和产品质量。因此，在设计 PLC 控制系统时，应遵循以下基本原则。

1. 最大限度地满足被控对象的控制要求

充分发挥 PLC 的功能，最大限度地满足被控对象的控制要求，是设计 PLC 控制系统的首要前提，也是设计中最重要的一条原则。这就要求设计人员在设计前就要深入现场进行调查研究。既要收集控制现场的实用资料，也要收集相关的国内、国外先进技术资料。同时要注意与现场的工程管理人员、工程技术人员、现场操作人员紧密配合，拟定控制方案，共同解决设计中的重点问题和疑难问题。

2. 保证 PLC 控制系统安全可靠

保证 PLC 控制系统能够长期安全、可靠、稳定运行，是设计控制系统的另一条重要原则。这就要求设计者在系统设计、元器件选择、软件编程上要全面考虑，以确保控制系统安全可靠。例如，应该保证 PLC 程序不仅在正常条件下运行，而且在非正常情况下（如突然掉电再上电、按钮按错等），也能正常工作。

3. 力求简单、经济、使用及维修方便

一个新的控制工程固然能提高产品的质量和数量，带来巨大的经济效益和社会效益，但新工程的投入、技术的培训、设备的维护也将导致运行资金的增加。因此，在满足控制要求的前提下，一方面要注意不断地扩大工程的效益，另一方面要注意不断地降低工程的成本。这就要求设计者不仅应该使控制系统简单、经济，而且要使控制系统的使用和维护方便、成本低，不宜盲目追求自动化和高指标。

4. 适应发展的需要

由于技术的不断发展，控制系统的要求也将会不断地提高，设计时要适当考虑到今后控制系统发展和完善的需要。这就要求在选择 PLC、输入/输出模块、I/O 端子数和内存容量时，要适当留有裕量，以满足今后生产的发展和工艺的改进。

6.1.2 PLC控制系统设计的一般步骤和内容

PLC控制系统的设计一般包括下面6个步骤，需要注意的是在设计的过程中要始终遵循控制系统设计的基本原则。

1. 分析控制对象

在进行PLC控制系统设计之前，首先要对控制对象进行深入的研究分析，明确控制任务、工艺流程，弄清实现这些任务需要哪些输入信号，选用什么类型的输入元件，哪些信号需要输出及如何执行驱动负载等问题，另外，还需搞清楚哪些量需要监控、显示，是否需要故障诊断，需要哪些保护措施等事宜。

2. 确定控制系统总体方案

这一步对于整个设计任务的成败至关重要。要在分析控制对象的基础上确定电气控制方案。控制系统设计一般是一个先整体后局部，逐步细化并不断完善的过程。在这一过程中，先根据主要的控制功能大致确定一个初步的控制方案，然后再完善细节。如果有不合适的地方再做调整，直至能满足各项要求。

3. 确定I/O元件，选择PLC机型

在确定电气控制方案之后，可进一步研究系统的硬件构成。要选择合适的输入和输出元件；确定主回路各电器和保护器件；选择报警和显示元件等。根据所选用的电器或元件的类型和数量，计算所需PLC的输入、输出端子数，然后选择合适的PLC机型。在选择PLC机型时要确定是PLC单机还是PLC网络，一般80点以内的系统选用不须扩展模块的PLC单机。PLC输入/输出端子数要留有10%余量，存储容量和指令执行速度是机型选择的一个重要指标，一般考虑选用大公司的PLC产品；当输入回路中电源为AC85～240V、DC24V时，应加装电源净化元件，PLC内、外接DC 24V“一”端和“COM”端不共接；输出回路中输出方式：继电器输出适用于不同公共点间带不同交、直流负载，晶体管输出适宜高频动作。

4. 确定I/O端子地址分配，绘制PLC输入/输出接线图和主电路图

明确各输入电器与PLC输入端子的对应关系及各输出端子与各输出执行元件的对应关系，做出PLC的I/O端子地址分配表。进行I/O端子地址分配时尽可能考虑接线布线的方便，同一类型的输入/输出元件应尽量排在一起。然后绘制出PLC输入/输出接线图和主电路图。

5. 编制应用程序，安装硬件电器

完成上述任务之后就要编制应用程序。程序设计的质量关系到系统运行的稳定性和可靠性。在设计程序时应尽量将系统运行中各种可能出现的情况考虑周全，必要时要向用户征询意见，这样设计出来的程序才能符合实际需要。另外，编制程序时应尽量按功能分类，模块化编程，程序注释要清楚明晰，以便日后查阅方便。程序编制完成之后，如果有可能尽量用模拟软件仿真运行一下，这样能尽早发现程序中的不妥和错误之处，为下一步的联机调试做好准备。与此同时，要进行硬件电器安装、布线和调试，这样可以缩短工程周期。

6. 系统联机调试，编写技术文件

将编制好的程序下载到 PLC 中去，进行联机调试。调试时可以将控制系统分成功能块，逐块进行调试，最后进行整体联合调试，以提高调试效率。在调试中要注意对人、机的保护。可以先易后难，先空载再加载。在调试中发现不合适的地方及时进行程序或硬件调整。调试完毕后编写相应的技术文件。

6.2 PLC 系统控制程序设计方法

PLC 控制程序在整个 PLC 控制系统中处于核心地位，程序质量的好坏对整个控制系统的性能有直接的影响。然而 PLC 初学者对程序设计经常感到困惑，无从下手。PLC 程序设计也有一定的规律可循，对于一些特定的功能通常都有相对固定的设计方法。常用的程序设计方法有逻辑设计法、时序图设计法、继电器—接触器控制线路转换设计法、顺序功能图设计法和经验设计法等。在程序设计过程中究竟用哪种方法并无定论。事实上，对于一个一般规模的控制系统来说往往是多种设计方法的融会贯通。要想编好 PLC 程序需要在熟悉硬件，掌握基本指令和常用编程方法的基础上多借鉴、多实践、多总结，这样才能真正掌握 PLC 程序设计技术。

6.2.1 逻辑设计法

逻辑设计方法是以逻辑组合或逻辑时序的方法和形式来设计 PLC 程序，可分为组合逻辑设计法和时序逻辑设计法两种。这种设计方法既具有严密可循的规律性、明确可行的设计步骤，又具有简便、直观和十分规范的特点。实例 73、实例 74 分别为组合逻辑设计法和时序逻辑设计法的实例。

实例 73：通风系统运行状态监控

实例说明

在一个通风系统中，有 4 台电动机驱动 4 台风机运转。为了保证工作人员的安全，一般要求至少 3 台电动机同时运行。因此用绿、黄、红三色柱状指标灯来对电动机的运动状态进行指示。当 3 台以上电动机同时运行时，绿灯亮，表示系统通风良好；当两台电动机同时运行时黄灯亮，表示通风状况不佳，需要改善；少于两台电动机运行时红灯亮起并闪烁，发出警告表示通风太差，需马上排除故障或进行人员疏散。

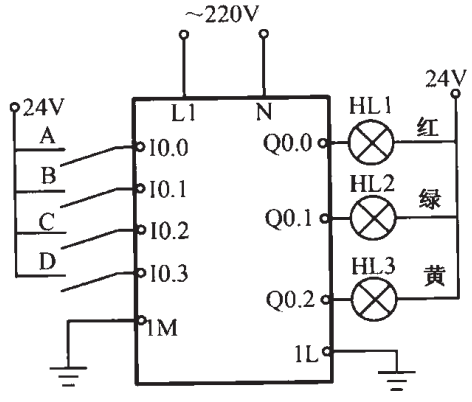


图 6-1 风机状态监视 PLC 接线图

实例实现

该系统 PLC 接线图如图 6-1 所示，图中 I0.0、I0.1、I0.2、I0.3 分别表示 4 台电动机运行状态检测传感器，当电动机运行时有信号输入，停止时无信号输入。梯形图控制程序如图 6-2 所示。

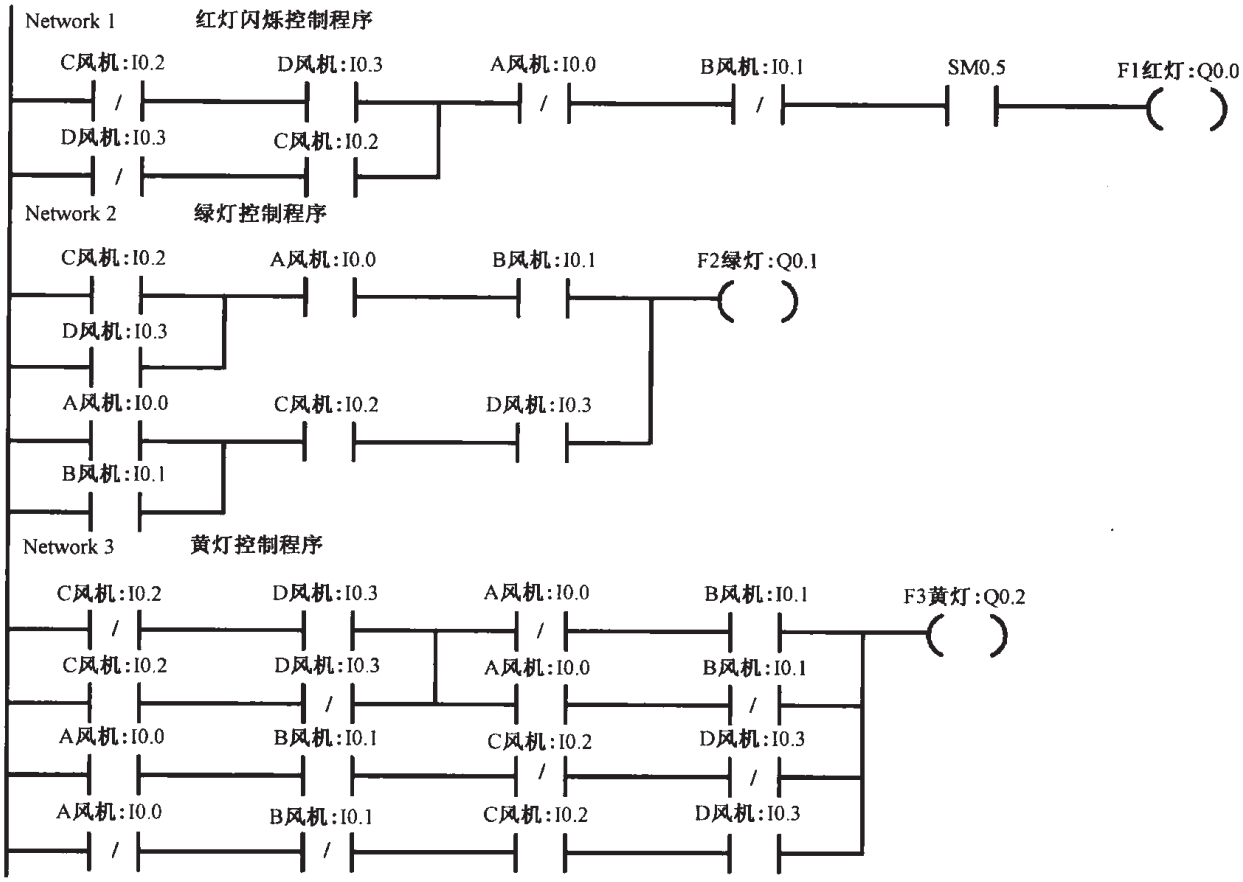


图 6-2　风机状态监视梯形图

图 6-2 梯形图对应的语句如下：

```
// 红灯闪烁控制程序
LDN     I0.2
A       I0.3
LDN     I0.3
A       I0.2
OLD
AN      I0.0
AN      I0.1
A       SM0.5
=       Q0.0
// 绿灯控制程序
LD      I0.2
O       I0.3
A       I0.0
A       I0.1
LD      I0.0
O       I0.1
A       I0.2
A       I0.3
OLD
```

```
=       Q0.1
// 黄灯控制程序
LDN     I0.2
A       I0.3
LD      I0.2
AN      I0.3
OLD
LDN     I0.0
A       I0.1
LD      I0.0
AN      I0.1
OLD
ALD
LD      I0.0
A       I0.1
AN      I0.2
AN      I0.3
LDN     I0.0
AN      I0.1
A       I0.2
A       I0.3
OLD
OLD
=       Q0.2
```

实例分析

用 A、B、C、D 来分别表示 4 台风机的运行状态。分别用 F1、F2、F3 表示红灯、绿灯和黄灯。3 盏灯的状态与系统的 3 种工作状态一一对应，下面分别针对这 3 种工作状态建立逻辑表达式。

（1）红灯闪烁

用“0”表示风机停止和指示灯“灭”，用“1”表示风机运行和指示灯“亮”（红灯的闪烁也用“亮”这种状态表示）。该种情况下的工作状态表如下：

A	B	C	D	F1
1	0	0	0	1
0	1	0	0	1
0	0	1	0	1
0	0	0	1	1
0	0	0	0	1

由状态表可得 F1 的逻辑函数：

$$F1 = A\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}\bar{C}\bar{D}$$

化简后得：

$$F1 = \bar{A}\bar{B}(\bar{C}D + C\bar{D}) + \bar{C}\bar{D}(\bar{A} + \bar{B})$$

根据该逻辑函数画出梯形图如图 6-3 所示。

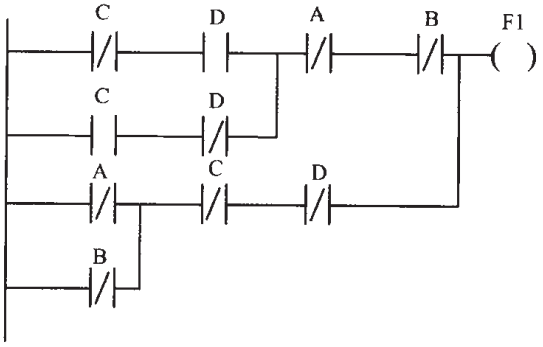


图 6-3　风机状态监视-红灯控制梯形图

（2）绿灯亮

列其工作状态表如下：

A	B	C	D	F2
1	1	1	0	1
1	1	0	1	1
1	0	1	1	1
0	1	1	1	1
1	1	1	1	1

由状态表可得 F2 的逻辑函数：

$$F2 = AB\bar{C}D + AB\bar{C}D + A\bar{B}CD + \bar{A}BCD + ABCD$$

化简后得：

$$F2 = AB(C + D) + CD(A + B)$$

根据该逻辑函数画出梯形图如图 6-4 所示。

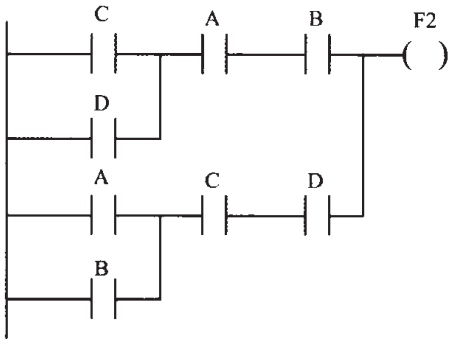


图 6-4　风机状态监视-绿灯控制梯形图

（3）黄灯亮

列其工作状态表如下：

A	B	C	D	F2
1	1	0	0	1
1	0	1	0	1
1	0	0	1	1
0	1	1	0	1
0	1	0	1	1
0	0	1	1	1

由状态表可得 F3 的逻辑函数：

$$F3 = A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + A\bar{B}\bar{C}D + \bar{A}BC\bar{D} + \bar{A}B\bar{C}D + \bar{A}\bar{B}CD$$

化简后得：

$$F3 = (\bar{A}B + A\bar{B})(\bar{C}D + C\bar{D}) + ABC\bar{D} + \bar{A}\bar{B}CD$$

根据该逻辑函数画出梯形图如图 6-5 所示。

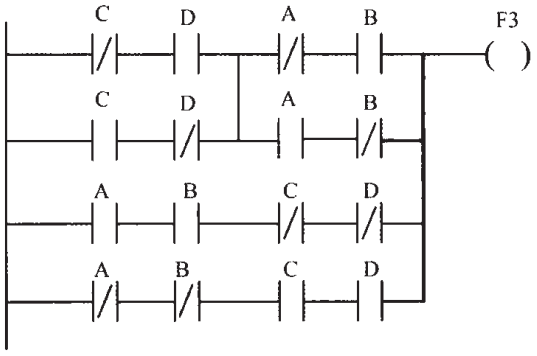


图 6-5　风机状态监视-黄灯控制梯形图

最后将红、绿、黄灯的控制梯形图合并，转换成 S7-200 PLC 控制程序，如图 6-2 所示。在红灯控制程序中常开点 SM0.5 是特殊存储器标志位，用来发生秒脉冲，以实现红灯闪烁。

实例 74：电动机交替运行控制

实例说明

有 M1 和 M2 两台电动机，按下启动按钮后，M1 运转 10 min，停止 5 min，M2 与 M1 相反，即 M1 停止时 M2 运行，M1 运行时 M2 停止，如此循环往复，直至按下停止按钮。

实例实现

该电动机控制系统的 I/O 接线图如图 6-6 所示。梯形图控制程序如图 6-7 所示。

图 6-7 梯形图对应的语句如下：

```
//两台电动机顺序控制程序
// 启停控制
LD      I0.0
O       M0.0
AN      I0.1
=       M0.0
// 定时器控制
LD      M0.0
LPS
AN      T38
TON     T37, 6000
LPP
AN      T38
```

```
TON       T38, 3000
// 电动机 M1 控制
LD        M0.0
AN        T37
=         Q0.0
// 电动机 M2 控制
LD        M0.0
A         T37
=         Q0.1
```

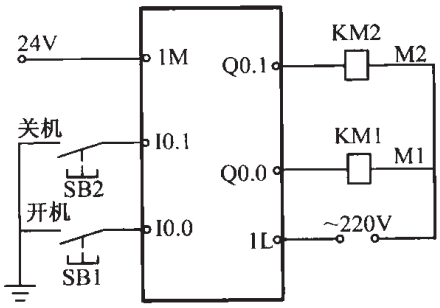


图 6-6　两台电动机顺序控制

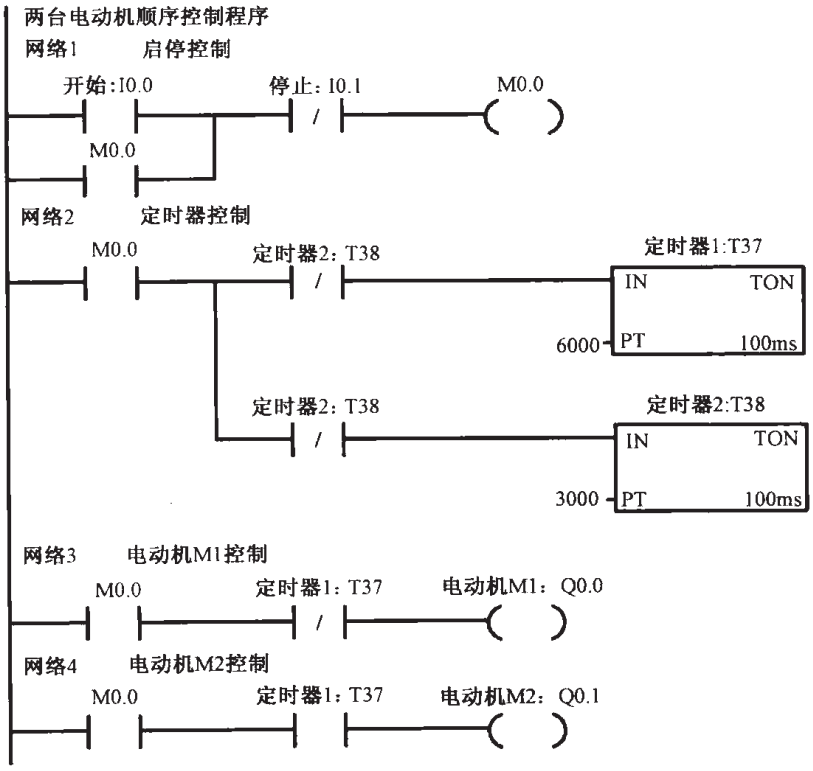


图 6-7　两台电动机顺序控制梯形图

实例分析

由于电动机 M1、M2 周期性交替运行，运行周期 T 为 15 min，则考虑采用延时接通定时器 T37（定时设置为 10 min）和 T38（定时设置为 15 min）控制这两台电动机的运行。当按下开机按钮 I0.0 后，T37 与 T38 开始计时，同时电动机 M1 开始运行。10 min 后 T37 定时时间到，并产生相应动作，使电动机 M1 停止，M2 开始运行。当定时器 T38 到达定时时间 15 min 时，T38 产生相应动作，使电动机 M2 停止，M1 开始运行，同时将自身和 T37 复位，程序进入下一个循环。如此往复，直到关机按钮按下，两个电动机停止运行，两个定时器也停止定时。

为了使逻辑关系清晰，用中间继电器 M0.0 作为运行控制继电器。根据控制要求画出两台电动机的工作时序图如图 6-8 所示。

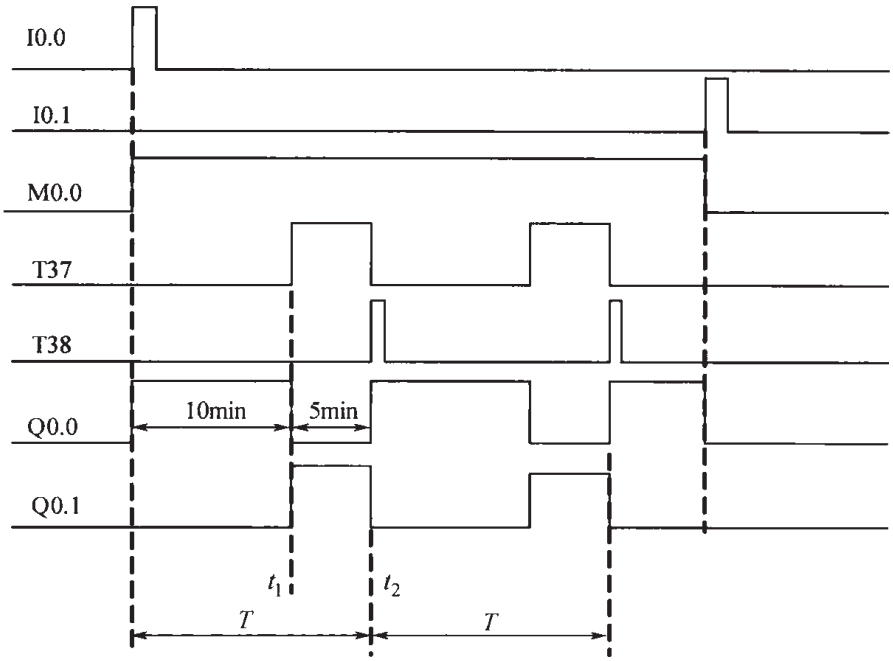


图 6-8 两台电动机顺序控制时序图

由该图 6-8 可以看出，t_1、t_2 时刻电动机 M1、M2 的运行状态发生改变，由前面的分析列出电动机运行的逻辑表达式

$$Q0.0 = M0.0 \cdot \overline{T37} \qquad Q0.1 = M0.0 \cdot T37$$

由此，可以根据上述分析结合编程经验，得到图 6-7 所示的梯形图程序。

6.2.2 移植设计法

移植设计法主要是用来对原有机电控制系统进行改造。PLC 控制取代继电器控制已是大势所趋，用 PLC 改造继电器控制系统，根据原有的继电器电路图来设计梯形图显然是一条捷径。这是由于原有的继电器控制系统经过长期的使用和考验，已经被证明能完成系统要求的控制功能，而继电器电路图又与梯形图极为相似，因此可以将继电器电路图经过适当的“翻译”，直接转化为具有相同功能的 PLC 梯形图程序，所以将这种设计方法称为“移植设计法”或“翻译法”。这种设计方法没有改变系统的外部特性，对于操作工人来说，除了控制系统的可靠性提高之外，改造前后的系统没有什么区别，他们不用改变长期形成的操作习惯。这种设计方法一般不需要改动控制面板及器件，因此可以减少硬件改造的费用和改造的工作量。

继电器电路图是一个纯粹的硬件电路图。将它改为 PLC 控制时，需要用 PLC 的外部接线图和梯形图来等效继电器电路图。可以将 PLC 想象成是一个控制箱，其外部接线图描述了这个控制箱的外部接线，梯形图是这个控制箱的内部“线路图”，梯形图中的输入位和输出位是这个控制箱与外部世界联系的“接口继电器”，这样就可以用分析继电器电路图的方法来分析 PLC 控制系统。在分析梯形图时可以将输入位的触点想象成对应的外部输入器件的触点，将输出位的线圈想象成对应的外部负载的线圈。外部负载的线圈除了受梯形图的控制外，还能受外部触点的控制。

将继电器电路图转换成为功能相同的 PLC 的外部接线图和梯形图的步骤如下：

（1）了解和熟悉被控设备的工作原理、工艺过程和机械的动作情况，根据继电器电路图分析和掌握控制系统的工作原理。

（2）确定 PLC 的输入信号和输出负载。继电器电路图中的交流接触器和电磁阀等执行机构如果用 PLC 的输出位来控制，它们的线圈在 PLC 的输出端。按钮、操作开关和行程开关、接近开关等提供 PLC 的数字量输入信号继电器电路图中的中间继电器和时间继电器的功能用 PLC 内部的存储器位和定时器来完成，它们与 PLC 的输入位、输出位无关。

（3）选择 PLC 的型号，根据系统所需要的功能和规模选择 CPU 模块、电源模块、数字量输入和输出模块，对硬件进行组态，确定输入/输出模块在机架中的安装位置和它们的起始地址。

（4）确定 PLC 各数字量输入信号与输出负载对应的输入位和输出位的地址，画出 PLC 的外部接线图。各输入和输出在梯形图中的地址取决于它们的模块的起始地址和模块中的接线端子号。

（5）确定与继电器电路图中的中间、时间继电器对应的梯形图中的存储器和定时器、计数器的地址。

（6）根据上述的对应关系画出梯形图。

实例 75：某卧式镗床继电器控制系统移植设计为 PLC 控制系统

实例说明

某卧式镗床继电器控制系统的电路图（如图 6-9 所示），包括主电路、控制电路、照明电路和指示电路。镗床的主轴电动机 M1 是双速异步电动机，中间继电器 KA1 和 KA2 控制主轴电动机的启动和停止，接触器 KM1 和 KM2 控制主轴电动机的正反转，接触器 KM4、KM5 和时间继电器 KT 控制主轴电动机的变速，接触器 KM3 用来短接串在定子回路的制动电阻。SQ1、SQ2 和 SQ3、SQ4 是变速操纵盘上的限位开关，SQ5 和 SQ6 是主轴进刀与工作台移动互锁限位开关，SQ7 和 SQ8 是镗头架和工作台的正、反向快速移动开关。

实例实现

改造后的 PLC 控制系统的外部接线图中，主电路、照明电路和指示电路同原电路不变，控制电路的功能由 PLC 实现，PLC 的 I/O 接线图如图 6-10 所示。

根据 PLC 的 I/O 对应关系，再加上原控制电路（图 6-9）中 KA1、KA2 和 KT 分别与 PLC 内部的 M300、M301 和 T37 相对应，可设计出 PLC 的梯形图如图 6-11 所示。

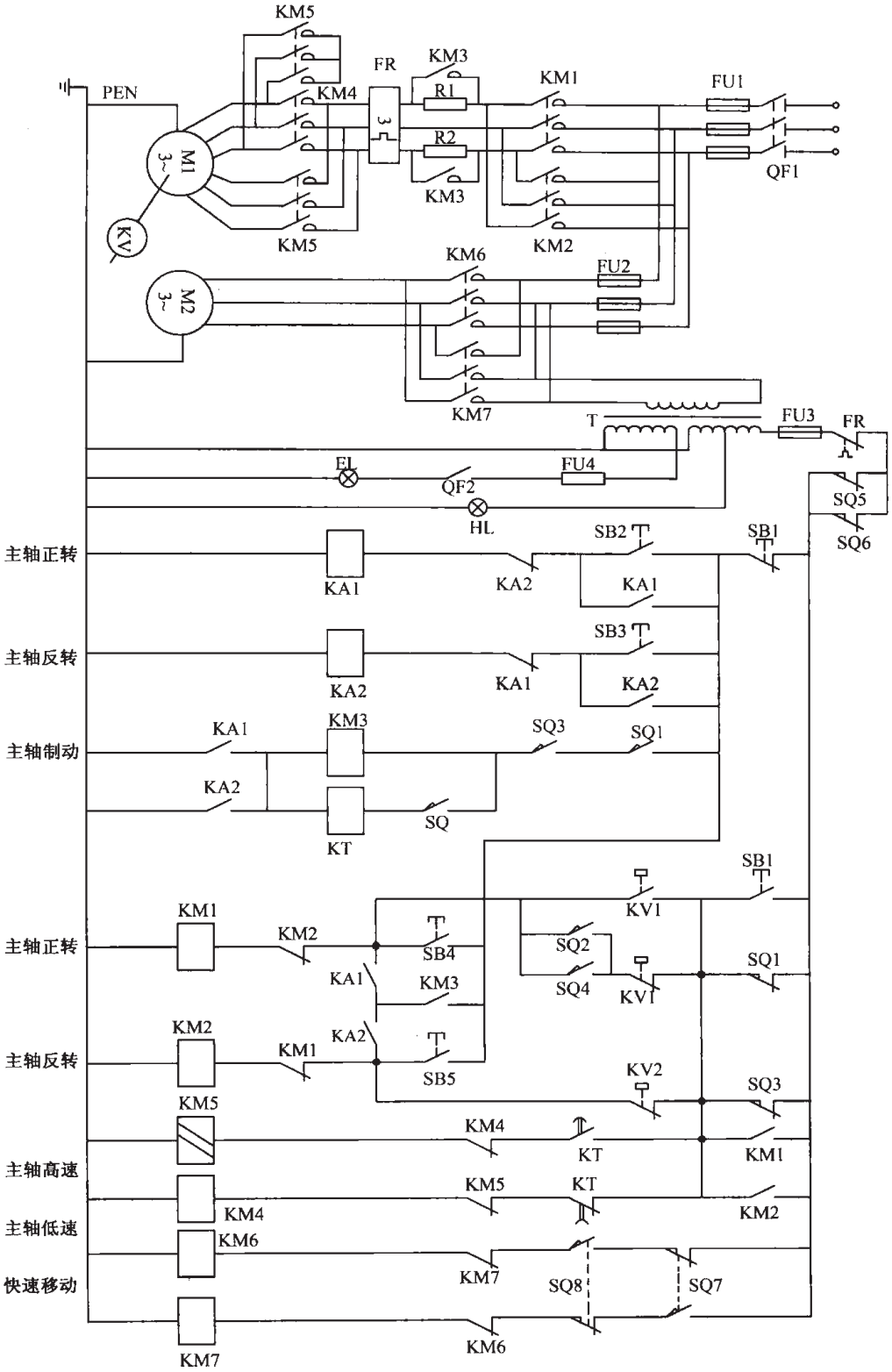


图 6-9 卧式镗床的继电器控制电路

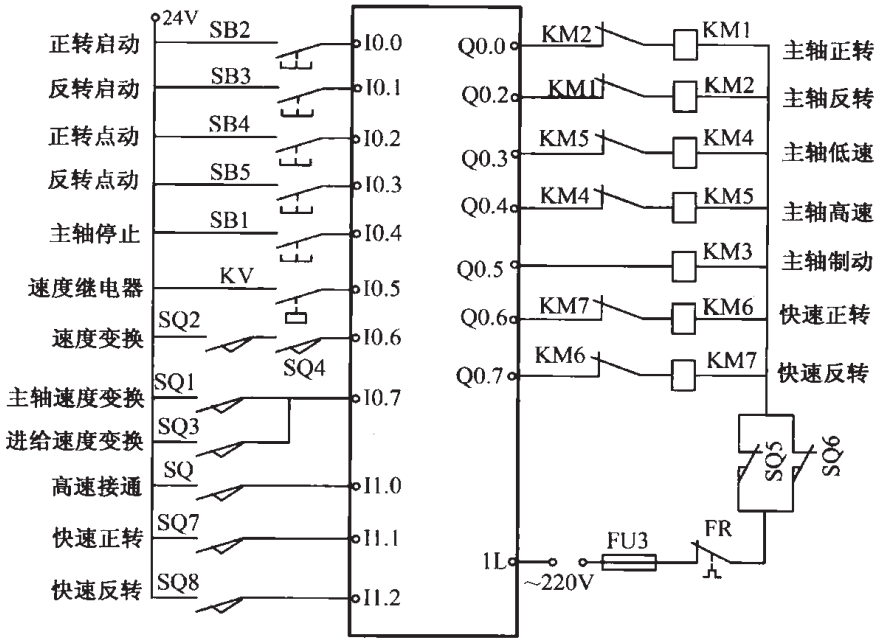


图 6-10　卧式镗床 PLC 控制系统 I/O 接线图

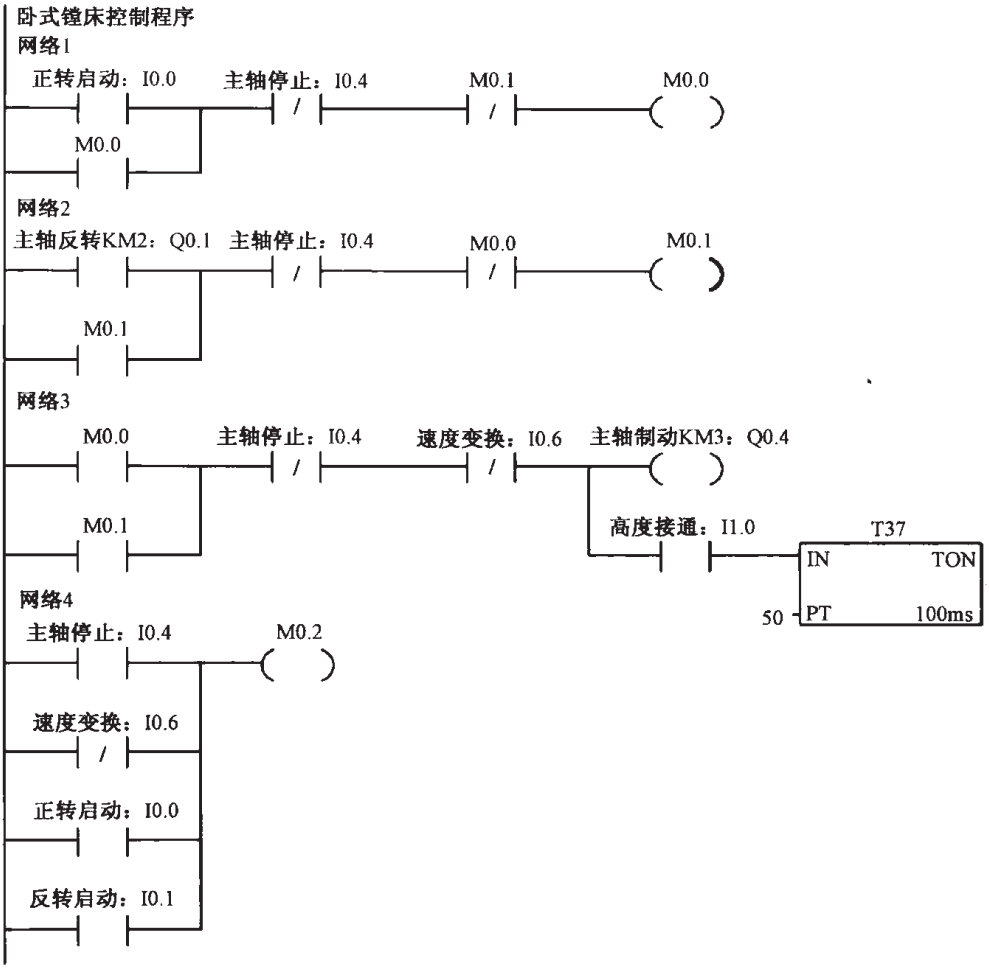


图 6-11　卧式镗床 PLC 控制系统的梯形图

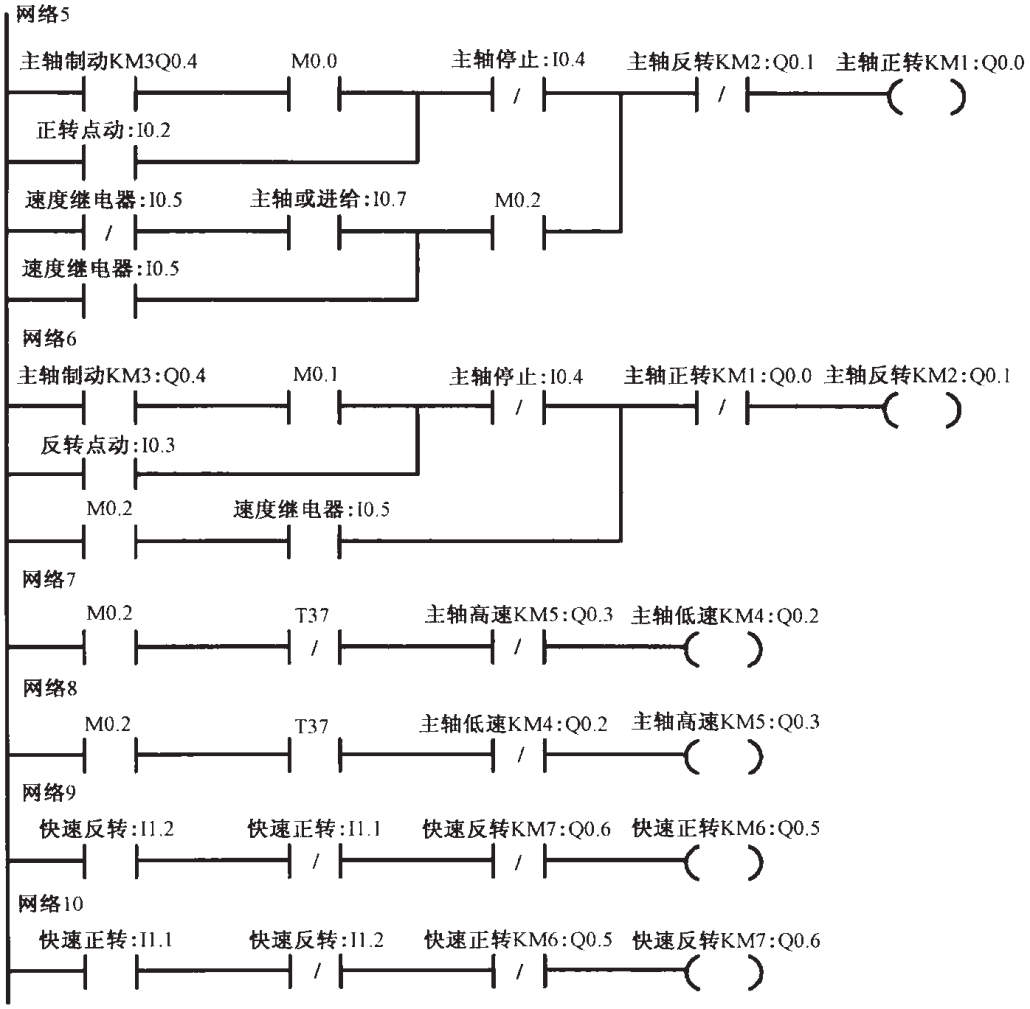


图 6-11　卧式镗床 PLC 控制系统的梯形图（续）

图 6-11 梯形图对应的语句如下：

```
//卧式镗床控制程序
// 主轴正转启动
LD      I0.0
O       M0.0
AN      I0.4
AN      M0.1
=       M0.0
// 主轴反转启动
LD      Q0.1
O       M0.1
AN      I0.4
AN      M0.0
=       M0.1
// 主轴制动
LD      M0.0
O       M0.1
AN      I0.4
```

```
AN      I0.6
=       Q0.4
A       I1.0
TON     T37, 50

LD      I0.4
ON      I0.6
O       I0.0
O       I0.1
=       M0.2
// 主轴正转
LD      Q0.4
A       M0.0
O       I0.2
AN      I0.4
LDN     I0.5
A       I0.7
O       I0.5
A       M0.2
OLD
AN      Q0.1
=       Q0.0
// 主轴反转
LD      Q0.4
A       M0.1
O       I0.3
AN      I0.4
LD      M0.2
A       I0.5
OLD
AN      Q0.0
=       Q0.1
// 主轴低速
LD      M0.2
AN      T37
AN      Q0.3
=       Q0.2
// 主轴高速
LD      M0.2
A       T37
AN      Q0.2
=       Q0.3
// 快速正转
```

```
LD      I1.2
AN      I1.1
AN      Q0.6
=       Q0.5
// 快速反转
LD      I1.1
AN      I1.2
AN      Q0.5
=       Q0.6
```

实例分析

设计过程中应注意梯形图与继电器电路图的区别。梯形图是一种软件，是 PLC 图形化的程序，PLC 梯形图是串行工作的，而在继电器电路图中，各电器可以同时动作（并行工作）。

根据继电器电路图设计 PLC 的外部接线图和梯形图时应注意以下问题：

（1）应遵守梯形图语言中的语法规定

由于工作原理不同，梯形图不能照搬继电器电路中的某些处理方法。例如在继电器电路中，触点可以放在线圈的两侧，但是在梯形图中，线圈必须放在电路的最右边。

（2）适当的分离继电器电路图中的某些电路

设计继电器电路图时的一个基本原则是尽量减少图中使用的触点的个数，因为这意味着成本的节约，但是这往往会使某些线圈的控制电路交织在一起。在设计梯形图时首要的问题是设计的思路要清楚，设计出的梯形图容易阅读和理解，并不是特别在意是否多用几个触点，因为这不会增加硬件的成本，只是在输入程序时需要多花一些时间。

（3）尽量减少 PLC 的输入和输出端子

PLC 的价格与 I/O 端子数有关，因此减少输入、输出信号的点数是降低硬件费用的主要措施。在 PLC 的外部输入电路中，各输入端可以接常开点或常闭点，也可以接触点组成的串、并联电路。PLC 不能识别外部电路的结构和触点类型，只能识别外部电路的通断。

（4）代换时间继电器

物理时间继电器有通电延时型和断电延时型两种。通电延时型时间继电器其延时动作的触点有通电延时闭合和通电延时断开两种。断电延时型时间继电器，其延时动作的触点有断电延时闭合和断电延时断开两种。在用 PLC 控制时，时间继电器可以用 PLC 的定时器或计数器或者是二者的组合来代替。

（5）设置中间单元

在梯形图中，若多个线圈都受某一触点串、并联电路的控制。为了简化电路，在梯形图中可以设置中间单元，即用该电路来控制某存储位，在各线圈的控制电路中使用其常开触点。这种中间元件类似于继电器电路中的中间继电器。

（6）设立外部互锁电路

控制异步电动机正反转的交流接触器如果同时动作，将会造成三相电源短路。为了防止出现这样的事故，应在 PLC 外部设置硬件互锁电路。

（7）重新确定外部负载的额定电压

PLC 双向晶闸管输出模块一般只能驱动额定电压 AC220V 的负载，如果系统原来的交流

接触器的线圈电压为 380 V，应换成 220 V 的线圈，或是设置外部中间继电器。

6.2.3　经验设计法

经验设计法，即在一些典型的控制电路程序的基础上，根据被控制对象的具体要求，进行选择组合，并多次反复调试和修改梯形图，有时需增加一些辅助触点和中间编程环节，才能达到控制要求。这种方法没有规律可遵循，设计所用的时间和设计质量与设计者的经验有很大的关系，所以称为经验设计法。经验设计法用于较简单的梯形图设计。

用经验设计法设计 PLC 程序时大致可以按下面几步来进行：分析控制要求、选择控制原则；设计主令元件和检测元件，确定输入输出设备；设计执行元件的控制程序；检查修改和完善程序。下面通过实例来介绍经验设计法。

实例 76：PLC 控制送料小车的经验设计

实例说明

如图 6-12 所示，送料小车在左限位开关 I0.4 处装料，20 s 后装料结束，开始右行，碰到右限位开关 I0.3 后停下来卸料，25 s 后左行，碰到左限位开关 I0.4 后又停下来装料，这样不停地循环工作，直到按下停止按钮 I0.2。按钮 I0.0 和 I0.1 分别用来启动小车右行和左行。

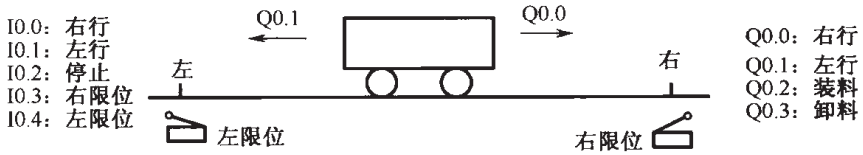


图 6-12　单处送料小车运行示意及 I/O 分配图

实例实现

以电动机正反转控制梯形图为基础，设计出的小车控制梯形图如图 6-13 所示。

图 6-13 梯形图对应的语句如下：

```
//单处送料小车运行控制梯形图
// 小车右行控制
LD      I0.0
O       T37
O       Q0.0
AN      I0.1
AN      I0.2
AN      I0.3
=       Q0.0
// 小车左行控制
LD      I0.1
O       T38
O       Q0.1
```

```
AN      I0.0
AN      I0.2
AN      I0.4
=       Q0.1
// 装料控制
LD      I0.4
=       Q0.2
TON     T37, 200
// 卸料控制
LD      I0.3
=       Q0.3
TON     T38, 250
```

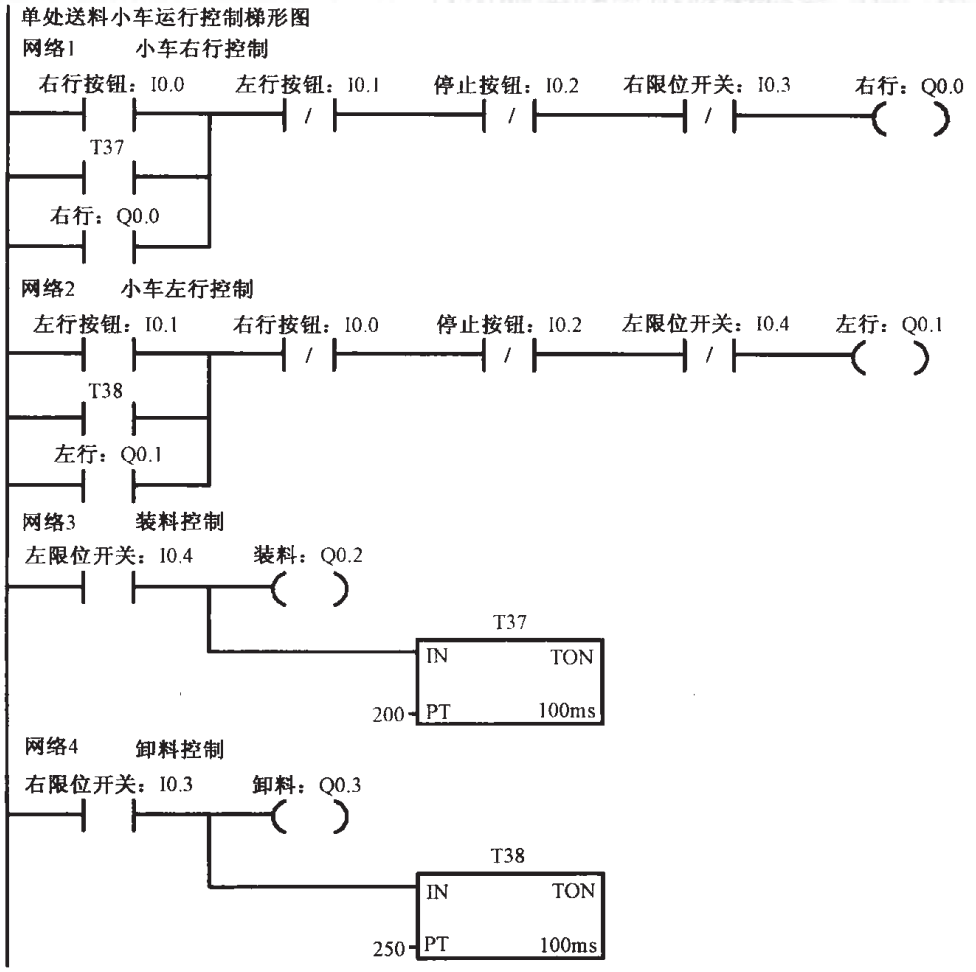


图 6-13 单处送料小车运行控制梯形图

实例分析

为使小车自动停止，将 I0.3 和 I0.4 的常闭触点分别与 Q0.0 和 Q0.1 的线圈串联。为使小车自动启动，将控制装、卸料延时的定时器 T37 和 T38 的常开触点，分别与手动启动右行和左行的 I0.0、I0.1 的常开触点并联，并用两个限位开关对应的 I0.4 和 I0.3 的常开触点分别接通装料、卸料电磁阀和相应的定时器。设小车在启动时是空车，按下左行启动按钮 I0.1，Q0.1

得电，小车开始左行，碰到左限位开关时，I0.4 的常闭触点断开，使 Q0.1 失电，小车停止左行。I0.4 的常开触点接通，使 Q0.2 和 T37 的线圈得电，开始装料和延时。20 s 后 T37 的常开触点闭合，使 Q0.0 得电，小车右行。小车离开左限位开关后，I0.4 变为“0”状态，Q0.2 和 T37 的线圈失电，停止装料，T37 被复位。对右行和卸料过程的分析与上面的基本相同。如果小车正在运行时按停止按钮 I0.2，小车将停止运动，系统停止工作。

经验设计法对于一些比较简单的程序设计是比较有效的，可以收到快速、简单的效果。但是，由于这种方法主要是依靠设计人员的经验进行设计，所以对设计人员的要求也就比较高，特别是要求设计者有一定的实践经验，对工业控制系统和工业上常用的各种典型环节比较熟悉。经验设计法没有规律可遵循，具有很大的试探性和随意性，往往需经多次反复修改和完善才能符合设计要求，所以设计的结果往往不是很规范，因人而异。

经验设计法一般适合于设计一些简单的梯形图程序或复杂系统的某一局部程序（如手动程序等）。如果用来设计复杂系统梯形图，存在以下两个问题。

（1）考虑不周、设计麻烦、设计周期长。

用经验设计法设计复杂系统的梯形图程序时，要用大量的中间元件来完成记忆、联锁、互锁等功能，由于需要考虑的因素很多，它们往往又交织在一起，分析起来非常困难，并且很容易遗漏一些问题。修改某一局部程序时，很可能会对系统其他部分程序产生意想不到的影响，往往花了很长时间，还得不到一个满意的结果。

（2）梯形图的可读性差、系统维护困难。

用经验设计法设计的梯形图是按设计者的经验和习惯的思路进行设计。因此，即使是设计者的同行，要分析这种程序也非常困难，更不用说维修人员了，这给 PLC 系统的维护和改进带来许多困难。

6.2.4　顺序功能图设计法

如果一个控制系统可以分解成几个独立的控制动作，且这些动作必须严格按照一定的先后次序执行才能保证生产过程的正常运行，这样的控制系统称为顺序控制系统，也称为步进控制系统。在工业控制领域中，顺序控制系统的应用很广，尤其在机械行业，几乎无例外地利用顺序控制来实现加工的自动循环。

所谓顺序控制设计法就是针对顺序控制系统的一种专门的设计方法。这种设计方法很容易被初学者接受，对于有经验的工程师，也会提高设计的效率，程序的调试、修改和阅读也很方便。PLC 的设计者们为顺序控制系统的程序编制提供了大量通用和专用的编程元件，开发了专门供编制顺序控制程序用的功能表图，使这种先进的设计方法成为当前 PLC 程序设计的主要方法。该方法的具体内容已在第 5 章有详述，这里不再赘述。

6.3　PLC 控制系统应用设计

在实际开发 PLC 控制程序时，对于控制规模较大的系统来说，可以根据控制系统各部分所要实现的功能将控制程序分成若干个控制模块，针对每个模块开发出与其对应的子程序，然后再开发出主程序对各子程序进行调度运行。这样就可以化繁为简，各个击破。对于每一个模块的程序开发可以在类似实例的基础上进行改造和扩展，这样能大大提高程序开发的效率。下面介绍一些 PLC 应用的典型实例。

实例 77：交通灯控制

实例说明

十字路口简单的交通信号灯布置如图 6-14 所示。由于东西方向的车流量较小、南北方向的车流量大，所以南北方向的放行时间为 30 s，东西方向的放行时间为 20 s。当在东西（或南北）方向的绿灯灭时，该方向的黄灯与南北（或东西）方向的红灯一起以 5Hz 的频率闪烁 5 s，之后，立即开始另一个方向的放行。当启动开关转向开始位置时，信号灯便以上述模式开始工作，直到启动开关转回到停止位置时，信号灯全部熄灭。

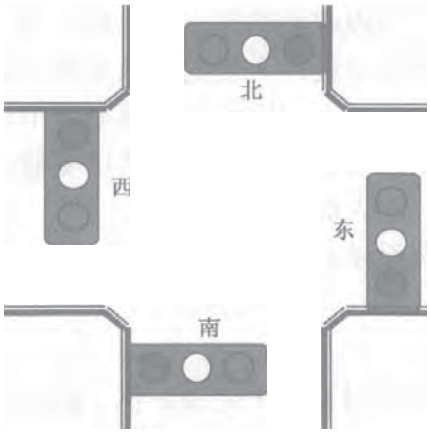


图 6-14　十字路口交通信号灯示意图

实例实现

在本实例中输入信号只有开关号一个。各信号灯的亮灭可由 PLC 的输出信号来控制，考虑到东西和南北方向的同色信号灯的亮灭状态分别相同，可将同一个方向上的两盏同色信号灯并联，由一个输出信号来控制。该交通信号灯控制系统的 I/O 分配如表 6-1 所示。

表 6-1　交通灯控制系统 I/O 分配表

输　入	输　出					
启动开关	南北绿灯	南北黄灯	南北红灯	东西绿灯	东西黄灯	东西红灯
I0.0	Q0.0	Q0.1	Q0.2	Q0.3	Q0.4	Q0.5

本实例是典型的时序逻辑控制系统，可以应用时序逻辑法来进行程序编制。首先画出该控制系统各信号的工作时序图，如图 6-15 所示。由时序图可以看出信号灯是按周期循环工作的，其工作周期 T 为 60 s。在一个工作周期中分为 4 个时间区段，各区段的分界点分别是 t_1、t_2、t_3 和 t_4 时刻，在这些时刻均有信号灯的状态发生改变。为了找出这些时间点可以分别用 PLC 中的延时接通定时器 T37、T38、T39、T40，这 4 个定时器均从 t_0 时刻开始定时，它们的定时时间分别为 30 s、35 s、55 s 和 60 s。由时序图可以写出各信号灯对应的逻辑表达式：

$$Q0.0 = I0.0 \cdot \overline{T37}$$

$$Q0.1 = T37 \cdot \overline{T38}$$

$$Q0.2 = T38 \cdot \overline{T39} + T39 \cdot \overline{T40}$$

$$Q0.3 = T38 \cdot \overline{T39}$$

$$Q0.4 = T39 \cdot \overline{T40}$$

$$Q0.5 = I0.0 \cdot \overline{T37} + T37 \cdot \overline{T38}$$

在这里信号灯的闪烁也看作常亮处理，只不过在把这些逻辑表达式转换为梯形图时要加上一个脉冲条件 SM0.5。4 个定时器的复位条件为当启动开关处于停止状态或一个工作周期结束时，即 T40 定时结束时。交通灯控制程序梯形图如图 6-16 所示。

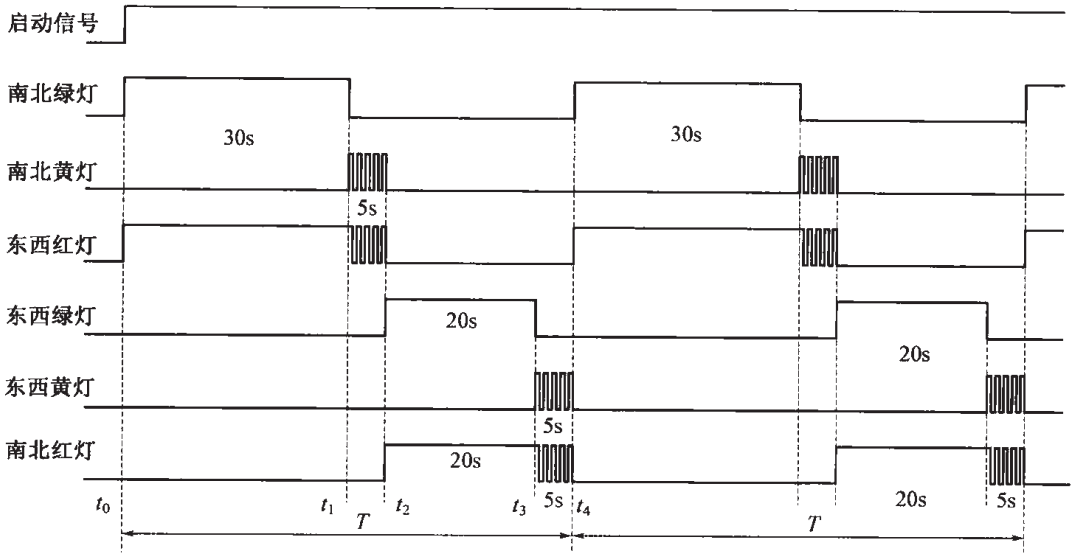


图 6-15　交通信号灯工作时序图

图 6-16 梯形图对应的语句如下：

```
//交通灯控制程序
// 定时器控制
LD      I0.0
AN      T40
TON     T37, 300
TON     T38, 350
TON     T39, 550
TON     T40, 600
// 南北绿灯控制
LD      I0.0
AN      T37
=       Q0.0
// 南北黄灯控制
LD      T37
AN      T38
A       SM0.5
=       Q0.1
// 南北红灯控制
LD      T38
AN      T39
LD      T39
AN      T40
A       SM0.5
OLD
=       Q0.2
// 东西绿灯控制
LD      T38
AN      T39
=       Q0.3
// 东西黄灯控制
```

```
LD      T39
AN      T40
A       SM0.5
=       Q0.4
// 东西红灯控制
LD      I0.0
AN      T37
LD      T37
AN      T38
A       SM0.5
OLD
=       Q0.5
```

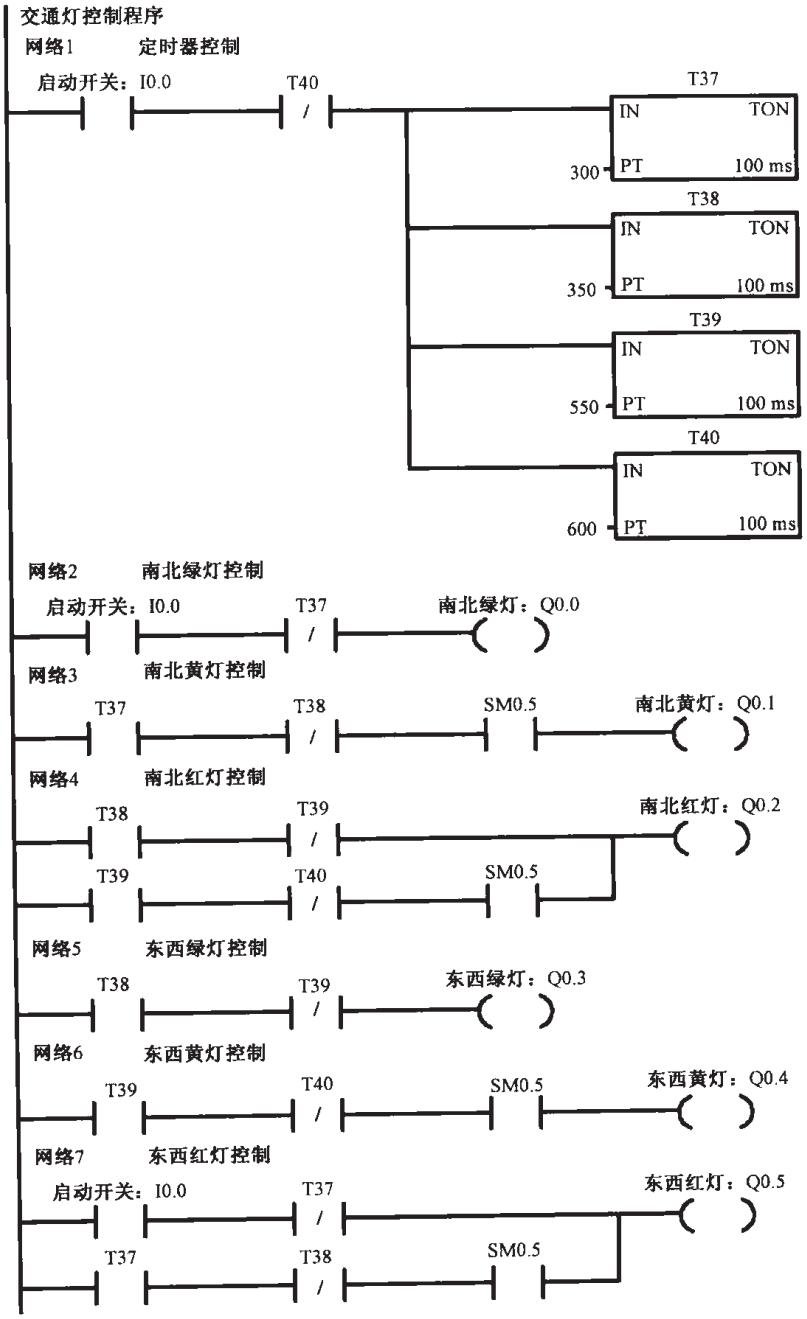


图 6-16 交通灯控制程序梯形图

实例分析

当控制开关 I0.0 转到启动位置时，4 个定时器均开始定时，此时南北绿灯和东西红灯亮；当定时器 T37 定时时间到时南北绿灯灭，南北黄灯和东西红灯开始闪烁；当定时器 T38 定时时间到时，南北黄灯和东西红灯均灭，东西绿灯和南北红灯亮；当定时器 T39 定时时间到时，东西绿灯灭，东西黄灯和南北红灯开始闪烁；当定时器 T40 定时时间到时，所有的定时器均被复位并重新开始定时，一个新的循环工作周期开始。在此期间，只要控制开关 I0.0 被转回停止位置，所有的定时器被复位，所有的信号灯均熄灭。

从本例中可以看出，在获取信号灯状态变化的时间点时，该程序所采用的定时器均是同时定时，同时复位，也就是说在一个工作周期内，各定时器定时时间均是相对于 t_0 时刻的绝对时间。当然也可以用“相对时间”来获得信号灯状态变化时间点，即一个定时器定时结束时启动另一个定时器。二者相比，采用前者能使思路清晰，编程较简单。而后者则使逻辑复杂，编程较困难。有兴趣的读者可以自己编程体会一下。

实例 78：工业机械手的 PLC 控制

实例说明

如图 6-17 所示为传送工件的某机械手的工作示意图，其任务是将工件从传送带 A 搬运到传送带 B。按启动按钮后，传送带 A 运行直到光电开关 PS 检测到物体，才停止，同时机械手下降。下降到位后机械手夹紧物体，2 s 后开始上升，而机械手保持夹紧。上升到位左转，左转到位下降，下降到位机械手松开，2 s 后机械手上升。上升到位后，传送带 B 开始运行，同时机械手右转，右转到位，传送带 B 停止，此时传送带 A 运行直到光电开关 PS 再次检测到物体，才停止循环。

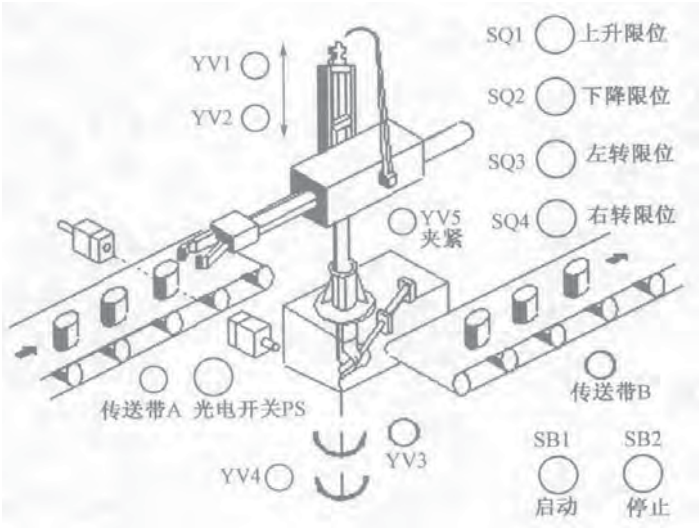


图 6-17　机械手控制示意图

实例实现

机械手的上升、下降和左转、右转的执行，分别由双线圈二位电磁阀控制汽缸的运动控制。当下降电磁阀通电，机械手下降，若下降电磁阀断电，机械手停止下降，保持现有的动

作状态。当上升电磁阀通电时，机械手上升。同样左转/右转也是由对应的电磁阀控制。夹紧/放松则是由单线圈的二位电磁阀控制汽缸的运动来实现，线圈通电时执行夹紧动作，断电时执行放松动作。并且要求只有当机械手处于上限位时才能进行左/右移动，因此在左/右转动时用上限条件作为联锁保护。由于上/下运动，左/右转动采用双线圈两位电磁阀控制，两个线圈不能同时通电，因此在上/下、左/右运动的电路中须设置互锁环节。

为了保证机械手动作准确，机械手上安装了限位开关 SQ1、SQ2、SQ3、SQ4，分别对机械手进行下降、上升、左转、右转等动作的限位，并给出动作到位的信号。光电开关 PS 负责检测传送带 A 上的工件是否到位，到位后机械手开始动作。该机械手控制系统的 I/O 分配表如表 6-2 所示。

表 6-2 机械手控制系统 I/O 分配表

输 入		输 出	
启动按钮	I0.0	上升 YV1	Q0.1
上升限位 SQ1	I0.1	下降 YV2	Q0.2
下降限位 SQ2	I0.2	左转 YV3	Q0.3
左转限位 SQ3	I0.3	右转 YV4	Q0.4
右转限位 SQ4	I0.4	夹紧 YV5	Q0.5
光电开关 PS	I0.6	传送带 A	Q0.6
停止按钮	I0.5	传送带 B	Q0.7

该机械手的控制程序梯形图如图 6-18 所示。

图 6-18 所示梯形图语句如下：

```
//工业机械手控制程序
// 启动回路
LD      I0.0
O       M0.0
AN      I0.5
=       M0.0
// 设置上限位标志位
LD      I0.1
O       M1.1
AN      Q0.2
=       M1.1
// 设置右限位标志位
LD      I0.4
O       M1.4
AN      Q0.3
=       M1.4
// 传送带工件检测标志
LD      I0.6
O       M1.6
A       M0.0
```

```
=       M1.6
// 传送带 A 控制
LD      M0.0
AN      M1.6
LD      M11.1
AN      M11.2
OLD
=       Q0.6
// 准备移位指令数据源
LD      M1.1
A       M1.4
AN      M10.1
AN      M10.2
AN      M10.3
AN      M10.4
AN      M10.5
AN      M10.6
AN      M10.7
AN      M11.0
AN      M11.1
A       M1.6
=       M10.0

LD      I0.5
R       M10.0, 10
R       M20.0, 1

LD      M10.0
LD      M10.1
A       I0.2
OLD
LD      M10.2
A       T37
OLD
LD      M10.3
A       I0.1
OLD
LD      M10.4
A       I0.3
OLD
LD      M10.5
A       I0.2
OLD
```

```
LD      M10.6
A       T38
OLD
LD      M10.7
A       I0.1
OLD
LD      M11.0
A       I0.4
OLD
LD      M11.1
A       I0.6
OLD
SHRB    M10.0, M10.1, 9
// 下降控制
LD      M10.1
O       M10.5
=       Q0.2
// 夹紧控制
LD      M10.2
S       M20.0, 1
TON     T37, 20

LD      M20.0
=       Q0.5
// 上升控制
LD      M10.3
O       M10.7
=       Q0.1
// 左转控制
LD      M10.4
=       Q0.3
// 夹紧复位
LD      M10.6
R       M20.0, 1
TON     T38, 20
// 右转及传送带 B 控制
LD      M11.0
AN      M11.1
=       Q0.7
=       Q0.4
```

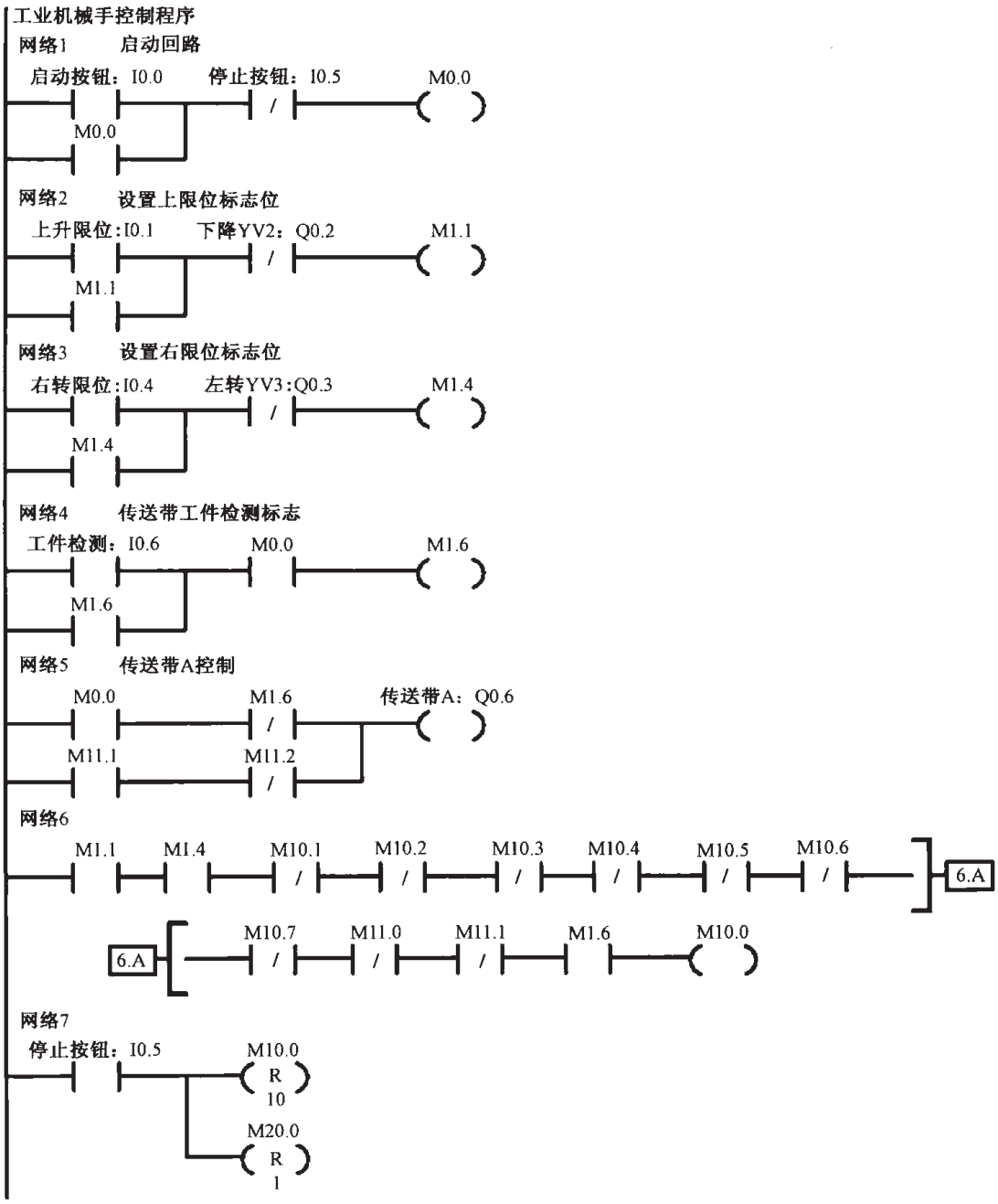


图 6-18　机械手控制梯形图程序

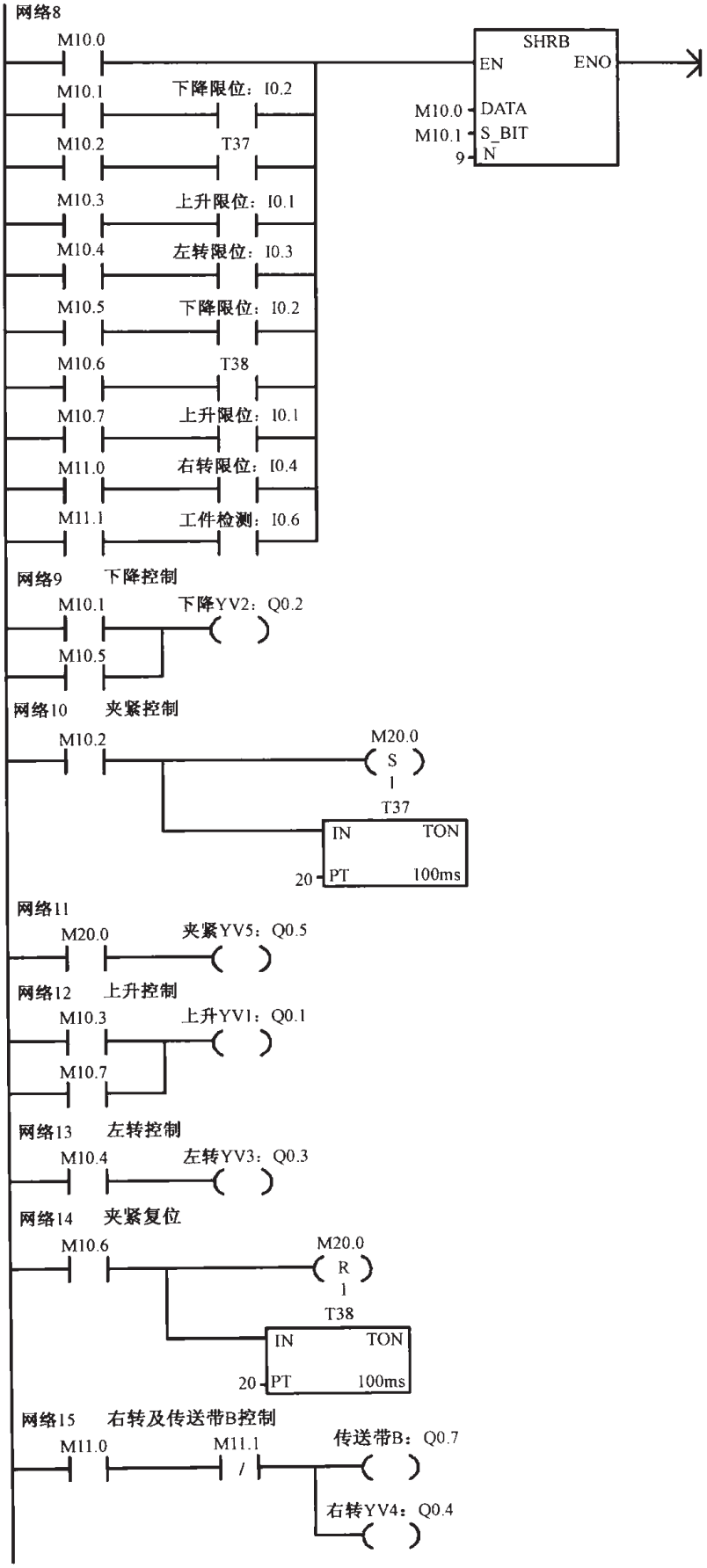


图 6-18 机械手控制梯形图程序（续）

实例分析

根据前面所述画出该机械手的功能流程图，如图 6-19 所示。流程图是一个按顺序动作的步进控制系统，在本例中采用移位寄存器编程方法。用移位寄存器 M10.1～1M11.2 位，代表流程图的各步，两步之间的转换条件满足时，进入下一步。移位寄存器的数据输入端 DATA（M10.0）由 M10.1～M11.1 各位的常闭接点、上升限位的标志位 M1.1、右转限位的标志位 M1.4 及传送带 A 检测到工件的标志位 M1.6 串联组成，即当机械手处于原位，各工步未启动时，若光电开关 PS 检测到工件，则 M10.0 置 1，这作为输入的数据，同时这也作为第一个移位脉冲信号。以后的移位脉冲信号由代表步位状态中间继电器的常开接点和代表处于该步位的转换条件接点串联支路依次并联组成。在 M10.0 线圈回路中，串联 M10.1～M11.1 各位的常闭接点，是为了防止机械手在还没有回到原位的运行过程中移位寄存器的数据输入端再次置 1，因为移位寄存器中的“1”信号在 M10.1～M11.1 之间依次移动时，各步状态位对应的常闭接点总有一个处于断开状态。当“1”信号移到 M11.2 时，机械手回到原位，此时移位寄存器的数据输入端重新置 1，若启动电路保持接通（M0.0=1），机械手将重复工作。当按下停止按钮时，使移位寄存器复位，机械手立即停止工作。若按下停止按钮后机械手的动作仍然继续进行，直到完成一周期的动作后，回到原位时才停止工作。由此可编制出该机械手的梯形图控制程序。

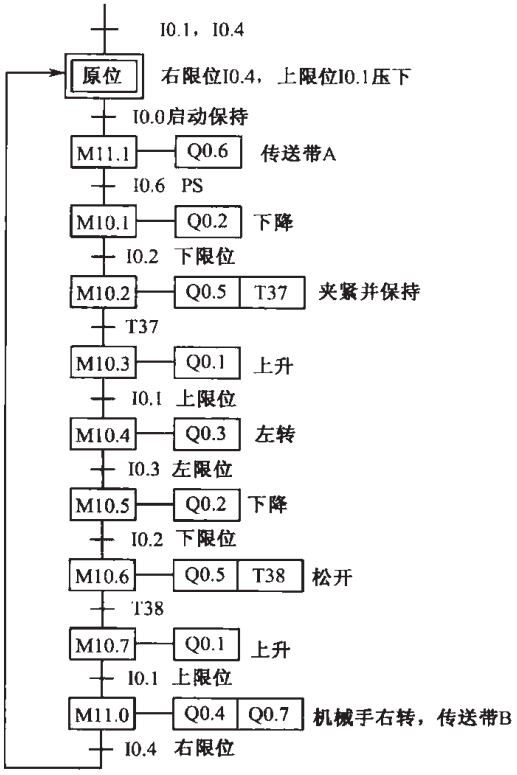


图 6-19　机械手功能流程图

实例 79：U 形板折板机的 PLC 控制

实例说明

U 形板折板机可以把金属板料折成如图 6-20 (a)所示的形状。U 形板折板机的基本组成结构如图 6-20 (b)所示。模板装在模板座上，模板两端的形状不同，加工出来的 U 形板料的开角形状就不同，折角的大小由左右的限位开关的位置决定。根据不同要求，模板可以随时更换。在气压的推动下，模板与模板座一起下移或上移。被加工的金属板料放在平台上，当模板下移压紧板料后，工作平台上的左、右折板在气压机构的推动下，可向上折，也可以折回，把板料加工成 U 形。

U 形板折板机的加工工艺过程如下：

（1）当模板上移到位，左、右折板返回原位时，将裁好的金属板料放在工作平台上；

（2）启动设备，模板开始下移，下移到位时压紧板料；

（3）接着左、右折板上折，上折到位压动左、右限位开关时，停止上折并保压 2 s；

（4）接着左、右折板返回，折板返回原位时自停；

（5）接着模板上移，上移到原位自停；

（6）取下 U 形板，一块板料的加工过程结束。

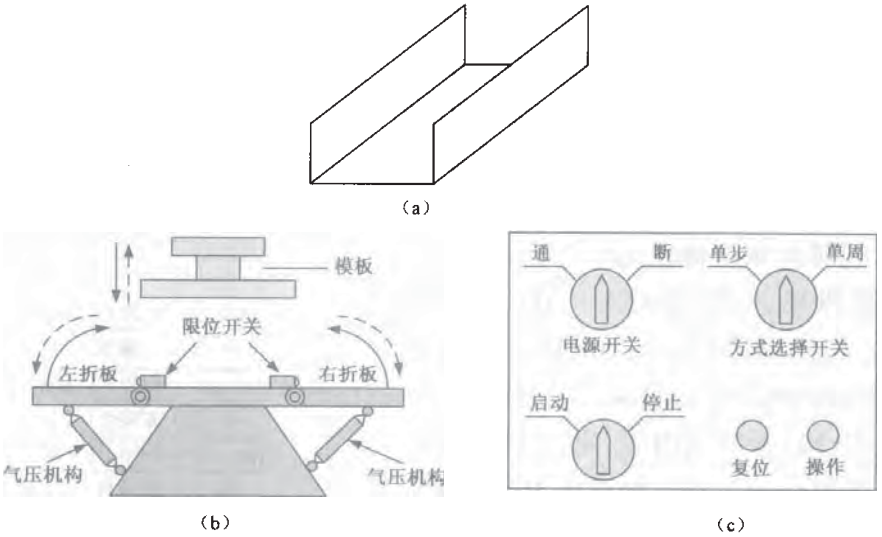


图 6-20　U 形板折板机的结构及操作盘示意图

实例实现

折板机的控制分为单步和单周期两种方式。单步方式时，按两次操作按钮执行一个工作步。单周期方式时，按一次操作按钮连续完成上述加工过程后自停，加工过程不循环。模板下移和上移是用双线圈的电磁阀控制的。当一个线圈通电时模板上移，当另一个线圈通电时模板下移。左、右折板上折和折回各用一个两线圈的电磁阀控制，当电磁阀的一个线圈通电时，折板上折，当另一个线圈通电时折板返回。两个折板必须都上折到位才能开始保压。折板机运行过程中可以停机。完成一个加工过程自动停机时，模板应在上方原位，左、右折板应返回到水平原位。停机再开机时，如果模板和左、右折板都在原位，则可以开始进行板料加工。如果不在原位，要将它们复位到原位才可以开始加工。手动复位时，用复位按钮来控制模板和两个折板的复位，但模板上移和折板的折回动作不能同时启动，以免两者发生摩擦而损坏模板和折板。自动运行和手动操作过程中，若误按操作按钮时不应出现误动作。

根据上述要求设计操作盘，如图 6-20（c）所示。操作盘上要设置：用于接通或断开 PLC 电源的开关（不占输入端子）；工作方式选择开关（占 1 个输入端子），分单周期位和单步位；启动/停止开关（占 1 个输入端子）；1 个复位按钮，若停机时模板和左/右折板不在原位，再开机时要进行复位；1 个操作按钮，手动和自动操作时，都要使用操作按钮。除了操作盘上介绍的输入元件外，左、右折板的上折限位各用一个限位开关，模板下移的控制需要一个限位开关。模板的上升和下移及折板的上折和折回所需的电磁阀（均为双线圈）都是输出执行元件。根据上述设置，折板机 PLC 控制系统 I/O 分配表如表 6-3 所示。

表 6-3　折板机 PLC 控制系统 I/O 分配表

输　入		输　出	
操作按钮	I0.0	模板下移	Q0.0
方式选择开关	I0.1	左折板上折	Q0.1
复位按钮	I0.2	右折板上折	Q0.2
启动/停止按钮	I0.3	左折板返回	Q0.3
左折板上限位开关	I0.4	右折板返回	Q0.4
右折板上限位开关	I0.5	模板上移	Q0.5
模板下移限位开关	I0.6		

折板机的 PLC 控制梯形图如图 6-21 所示。

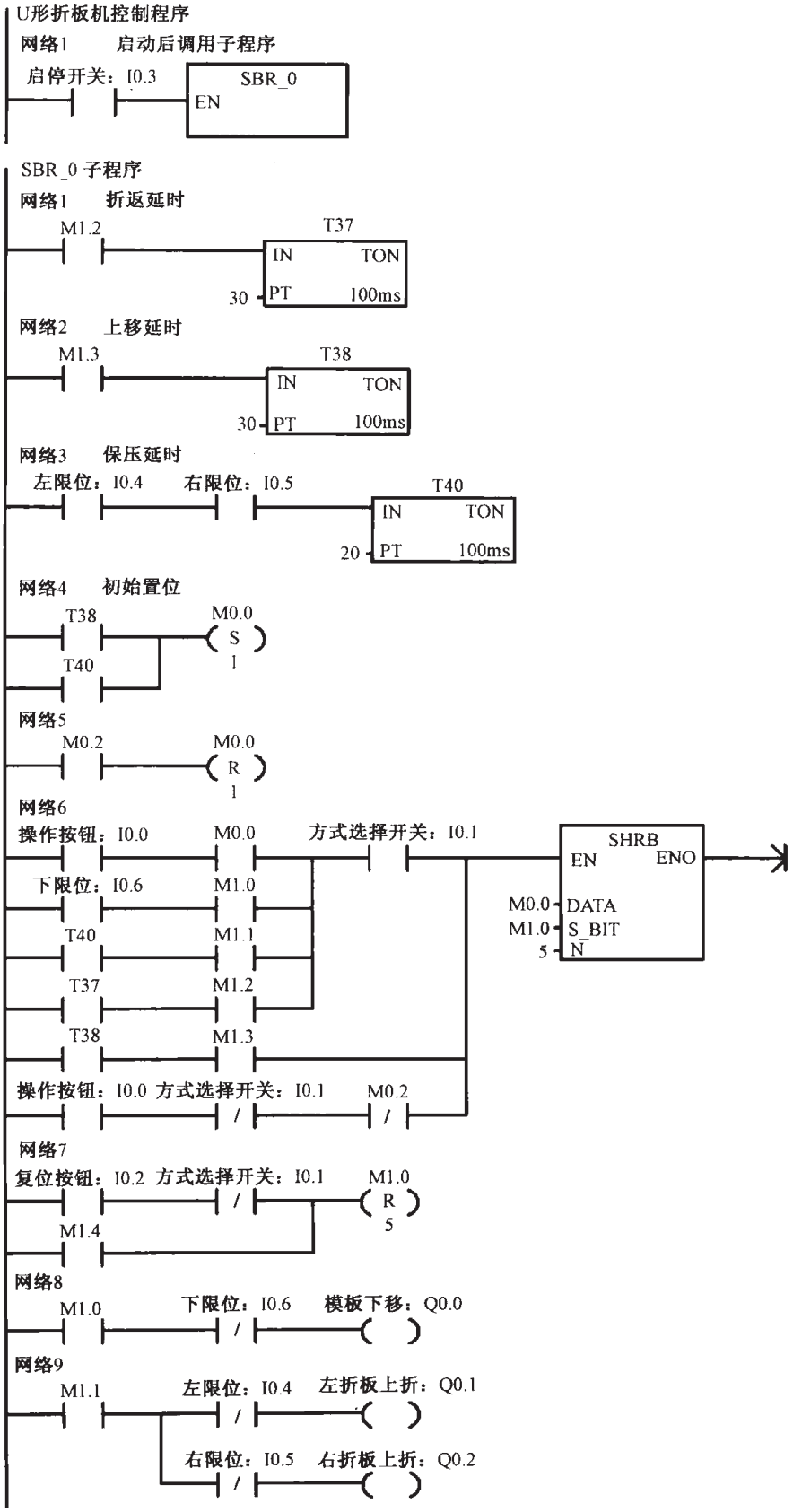


图 6-21　U 形折板机的 PLC 控制梯形图

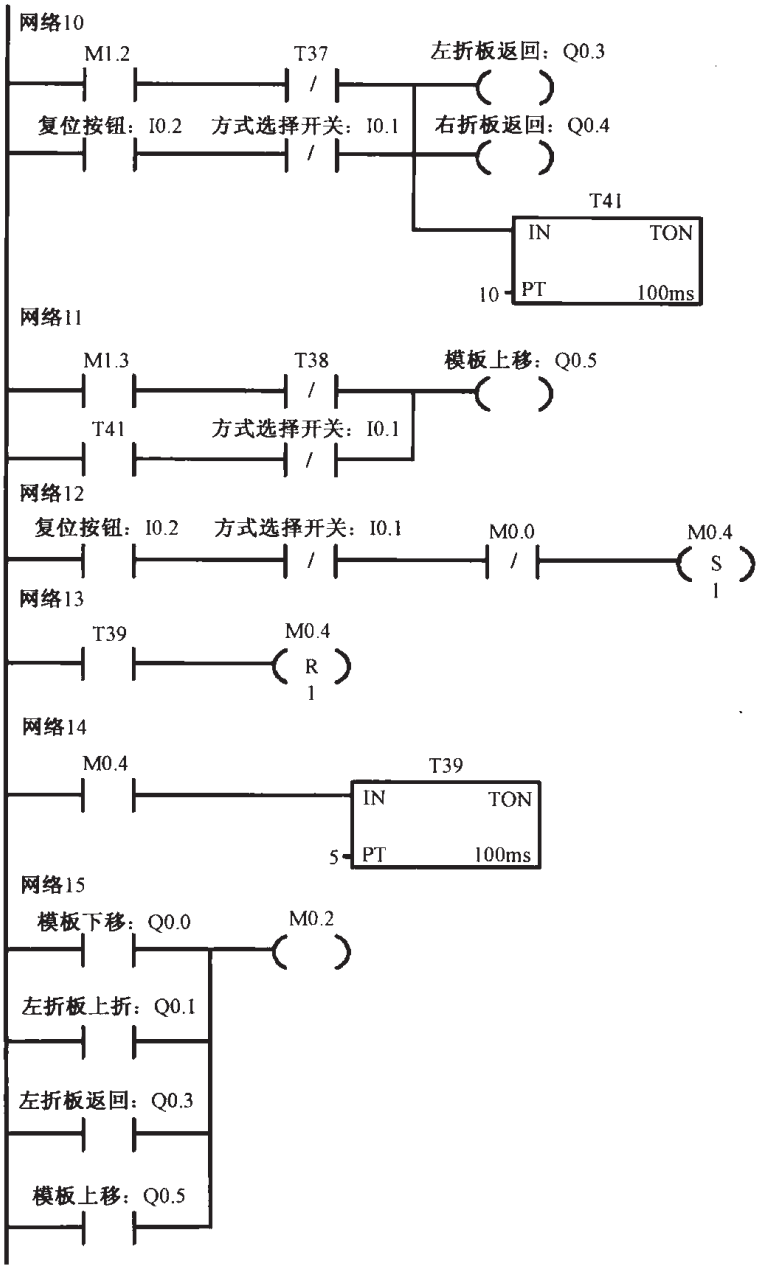


图 6-21　U 形折板机的 PLC 控制梯形图（续）

图 6-21 所示梯形图控制程序语句如下：

```
//U 形折板机控制程序
//启动后调用子程序
LD      I0.3
CALL    SBR0

//子程序 SBR0
// 折返延时
LD      M1.2
TON     T37, 30
// 上移延时
LD      M1.3
```

```
TON     T38, 30
// 保压延时
LD      I0.4
A       I0.5
TON     T40, 20
// 初始置位
LD      T38
O       T40
S       M0.0, 1
LD      M0.2
R       M0.0, 1
LD      I0.0
A       M0.0
LD      I0.6
A       M1.0
OLD
LD      T40
A       M1.1
OLD
LD      T37
A       M1.2
OLD
A       I0.1
LD      T38
A       M1.3
LD      I0.0
AN      I0.1
AN      M0.2
OLD
OLD
SHRB    M0.0, M1.0, 5
LD      I0.2
AN      I0.1
O       M1.4
R       M1.0, 5
LD      M1.0
AN      I0.6
=       Q0.0
LD      M1.1
LPS
AN      I0.4
=       Q0.1
LPP
```

```
AN      I0.5
=       Q0.2
LD      M1.2
AN      T37
LD      I0.2
AN      I0.1
OLD
=       Q0.3
=       Q0.4
TON     T41, 10

LD      M1.3
AN      T38
LD      T41
AN      I0.1
OLD
=       Q0.5
LD      I0.2
AN      I0.1
AN      M0.0
S       M0.4, 1
LD      T39
R       M0.4, 1

LD      M0.4
TON     T39, 5

LD      Q0.0
O       Q0.1
O       Q0.3
O       Q0.5
=       M0.2
```

实例分析

下面介绍各种操作方式时的控制功能以及误操作禁止的原理。

（1）复位操作

为了确保机器从正确的初始位置开始工作，开机时首先要进行复位操作，否则机器不能被启动。将方式选择开关拨在单步位，I0.1 为 OFF，自动运行被禁止。按住复位按钮 I0.2 不放，其复位过程是：首先，寄存器 M1.0～M1.4 被复位；接着 M0.4 被置位，T39 开始计时，0.5 s 后将 M0.0 置位，为板料加工做好准备。Q0.3 和 Q0.4 均为 ON，两个折板开始返回。为了避免模板和折板相互摩擦，要自折板开始返回 1s 后，再让模板开始上移，因此设置 T41 定时 1s。T41 定时 1 s 到，模板开始上移。当模板和折板都返回原位（目测）时，松开复位

按钮，复位过程结束。

（2）单周期方式的加工过程控制

模板和折板都返回原位时，将剪好的板料放在平台上。将方式开关拨在单周期位，常开触点 I0.1 闭合，使 SHRB 的移位脉冲输入端接通，而单步运行方式被禁止。按一下操作按钮 I0.0，第一次执行移位，M1.0 和 Q0.0 均为 ON，则模板开始下移；M0.2 为 ON 并将 M0.0 复位，使下一次移位时移位输入是“0”。模板下移到位压下限位开关 I0.6。其一，使 Q0.0 为 OFF、模板下移停止并压紧板料；其二，输入一个移位脉冲，使 M1.0 为 OFF、M1.1 为 ON，于是 Q0.1 和 Q0.2 为 ON，两折板开始上折。当两折板上折到位，压动限位开关 I0.4 和 I0.5，其一，使 Q0.1 和 Q0.2 均为 OFF，折板上折停止；其二，当 I0.4 和 I0.5 均 ON 时 T40 开始计时，进行 2s 保压。T40 计时到，输入一个移位脉冲，使 M1.1 为 OFF，M1.2 为 ON，则 Q0.3 和 Q0.4 均 ON，折板开始返回，T37 进行折板返回计时。T37 定时时间到，输入一个移位脉冲，使 M1.2 为 OFF、M1.3 为 ON，Q0.3 和 Q0.4 为 OFF，折板返回停止；Q0.5 为 ON 使模板开始上移，T38 开始模板上移计时。T38 计时到，输入一个移位脉冲，使 M1.3 为 OFF，M1.4 为 ON。其一，Q0.5 为 OFF，模板上移停止；其二，T38 为 ON，将 M0.0 置位，为下一次加工作好准备；其三，M1.4 为 ON 使移位寄存器复位。到此为止，加工一块板料的过程结束，下一块板料的加工与上述过程相同。

（3）单步操作

将方式开关拨在单步位，常开触点 I0.1 断开，自动方式被禁止，单步方式被允许。按一下操作按钮，I0.0 ON 一次，输入一个移位脉冲，M1.0 和 Q0.0 均为 ON。其一，模板开始下移；其二，M0.2 将 M0.0 复位，使下一次移位时输入的 是“0”。当模板下移到位压限位开关 I0.6 时，模板下移停止并压紧板料。再等一下操作按钮 I0.0，又输入一个移位脉冲，M1.0 为 OFF，M1.1 为 ON。使 Q0.1 和 Q0.2 均为 ON，两折板开始上折。折板上折到位压限位开关 I0.4 和 I0.5，则 Q0.1 和 Q0.2 均为 OFF，使折板上折停止，同时 T40 开始计时进行 2s 保压。2s 后再按一下操作按钮，又输入一个移位脉冲，M1.1 为 OFF，M1.2 为 ON。于是折板开始返回，T37 进行折板返回计时。T37 计时到，Q0.3 和 Q0.4 均为 OFF，折板返回停止。再按一下操作按钮，又输入一个移位脉冲，M1.2 为 OFF，M1.3 为 ON。于是模板开始上移，T38 开始模板上移计时。T38 计时到，其一，输入一个移位脉冲，使 M1.3 为 OFF， M1.4 为 ON，则 Q0.5 为 OFF，模板上移停止；其二，M0.4 为 ON 使移位寄存器复位；其三，T38 为 ON 将 M0.0 置位，为下一次加工做好准备。

（4）误操作时误动作的禁止

单步操作时误动作的禁止：当按过一次操作按钮，且正在执行某步加工的过程中又误按了一下操作按钮时，不会产生误动作。因为在每一步启动时按一次按钮 I0.0 后，对应各个步，分别使 Q0.0、Q0.1、Q0.3、Q0.5 为 ON，都能使 M0.2 为 ON，其常闭触点断开。所以在某步运行过程中误按操作按钮 I0.0 时，不会再输入移位脉冲，也就避免了误动作。另外，由手动开关产生的信号一般都可能产生抖动，例如，按一次按钮却连续发出几个脉冲信号。本例中，为了避免由这种现象造成的误动作，采取了如下措施：将常闭触点 M0.2 与 I0.0 串联，由于 I0.0 第一次 ON 就使 M0.2 为 ON，其常闭触点即断开，这样 I0.0 的后几次 ON 就不会起作用了。单周期操作时误动作的禁止：当第一次按过操作按钮 I0.0、设备启动后，Q0.0 为 ON，M0.2 为 ON，将 M0.0 复位。由于 I0.0 与 M0.0 的常开触点串联后连接在移位寄存器的移位脉

冲输入端，在整个加工过程中，由于 M0.0 的常开触点总是断开，也不会产生错误的移位脉冲输入，所以由误按操作按钮而产生的误动作不会出现。

实例 80：某型导弹测试架控制

实例说明

导弹在发射前必须进行性能测试，模拟导弹在飞行（飞机挂弹飞行）状态下各种姿态的性能是否符合要求，如上仰、俯冲、翻滚、转弯等，模拟导弹飞行状态由测试机架来完成。而现役的某型导弹的测试机架是纯机械装置，如图 6-22 所示。靠手动控制，操作强度大，模拟量不准确，可靠性差，且无法满足导弹测试系统自动化检测，因此，提出对其进行 PLC 控制的自动化改造。

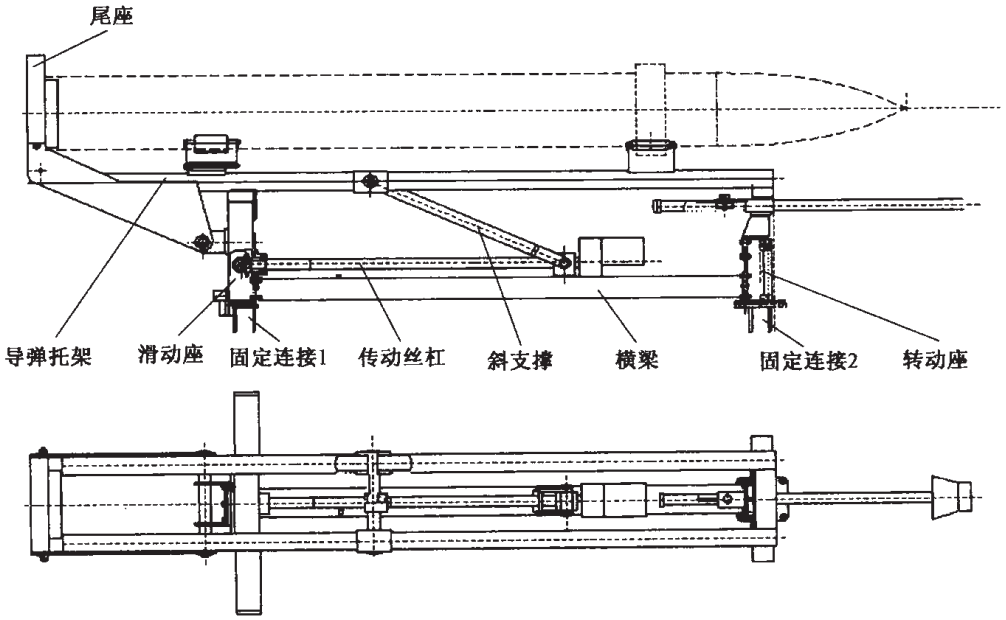


图 6-22　某型导弹测试系统机械装置

实例实现

1．明确控制要求

测试机架原为手动，现用 PLC 改造为自动控制运行，并可与其他系统衔接，因此，原测试机架保持不变，主要设计其控制系统，要求具有如下功能：装上导弹后的测试机架能自动俯仰 45°；导弹在机架上可自动产生一定的横滚角速度，并能自动翻滚 180°；导弹能水平摆动，且角加速度大于或等于 1 rad/s^2；测试机架的 PLC 能与上位 PC RS-232C 通信口通信。

2．电器传动控制系统设计

（1）控制系统组成及元件选择

系统由 PLC 模块、电源模块、现场 I/O 信号、步进电动机、驱动器、位置传感器、上位机接口等几部分组成。PLC 作为控制系统的核心，选用 S7-200 系列的 CPU224XP 型号，该

机有 14 输入/10 输出共 24 个数字量 I/O 端子，6 个独立的 30kHz 高速计数器，2 路独立的 20kHz 高速脉冲输出，具有 PID 控制器。两个 RS-485 通信/编程口，具有 PPI 通信协议、MPI 通信协议和自由方式通信能力，有较强的控制能力；步进电动机选择进口的百格拉三相混合步进电动机；驱动器选择和电动机配套的 WD3-007 型驱动器（接在 CPU224XP 选 24V 输入，接在 EM253 模块上的选 5 V 输入）。上位机要求 RS-232C 串口通信，而 CPU224XP 型 PLC 只有 RS-485 接口，所以使用 PC/PPI 电缆。整个控制系统要用 1 台 PLC 和两个 EM253 位控模块控制 3 台步进电动机实现 3 种运动，如图 6-23 所示。

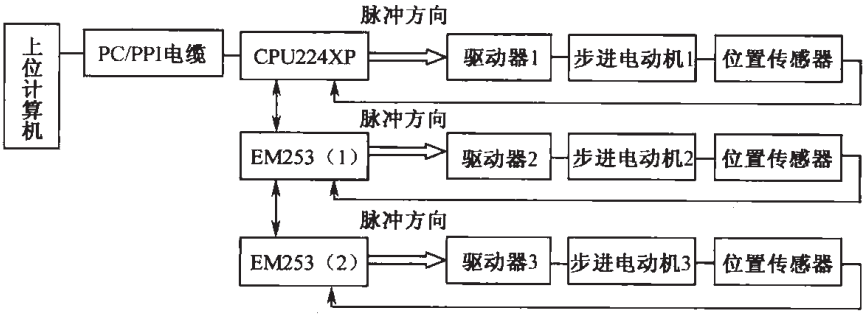


图 6-23　控制系统组成方框图

（2）PLC 硬件接线

每台 PLC 主控单元由现场按钮和限位开关等输入信号，脉冲、脉冲方向和状态指示灯等输出信号组成。根据控制要求列出输入、输出信号，并标出代号，如表 6-4 所示。分配每个 I/O 端子的地址，并画出 PLC 与现场器件的安装接线图，如图 6-24 所示。CPU224XP 的脉冲和脉冲方向输出端都需要串接一个 1.8 kΩ的限流电阻，以满足驱动器脉冲和脉冲方向输入端的电流范围：7 mA<I_{max}<25 mA，−25 mA<I_{min}<0.2 mA。

表 6-4　PLC 输入输出信号表

输入信号			输出信号		
代　号	地址号	名　称	代　号	地址号	名　称
SB0	I0.0	上仰 45°按钮	脉冲 0	Q0.0	俯仰运动脉冲
SB1	I0.1	下降 45°按钮	脉冲方向 0	Q0.1	俯仰运动方向
SB2	I0.2	俯仰急停按钮	L0	Q0.2	上升指示灯
SM0	I0.3	上仰限位开关	L1	Q0.3	下降指示灯
SM1	I0.4	下降限位开关	脉冲 1	2P1	翻滚运动脉冲
SB3	I0.5	顺时针翻滚按钮	脉冲方向 1	2P0	翻滚运动方向
SB4	I0.6	逆时针翻滚按钮	L2	Q0.4	顺时针翻滚指示灯
SB5	I0.7	翻滚急停按钮	L3	Q0.5	逆时针翻滚指示灯
SM2	2LMT-	翻滚限位开关 1	脉冲 2	1P1	水平摆动脉冲
SM3	2LMT+	翻滚限位开关 2	脉冲方向 2	1P0	水平摆动方向
SB6	I1.0	摆动启动按钮	L4	Q0.6	顺时针摆动指示灯
SB7	I1.1	摆动停止按钮	L5	Q0.7	逆时针摆动指示灯
SB8	I1.2	摆动急停按钮			
SM4	1LMT-	摆动限位开关 1			
SM5	1LMT+	摆动限位开关 2			

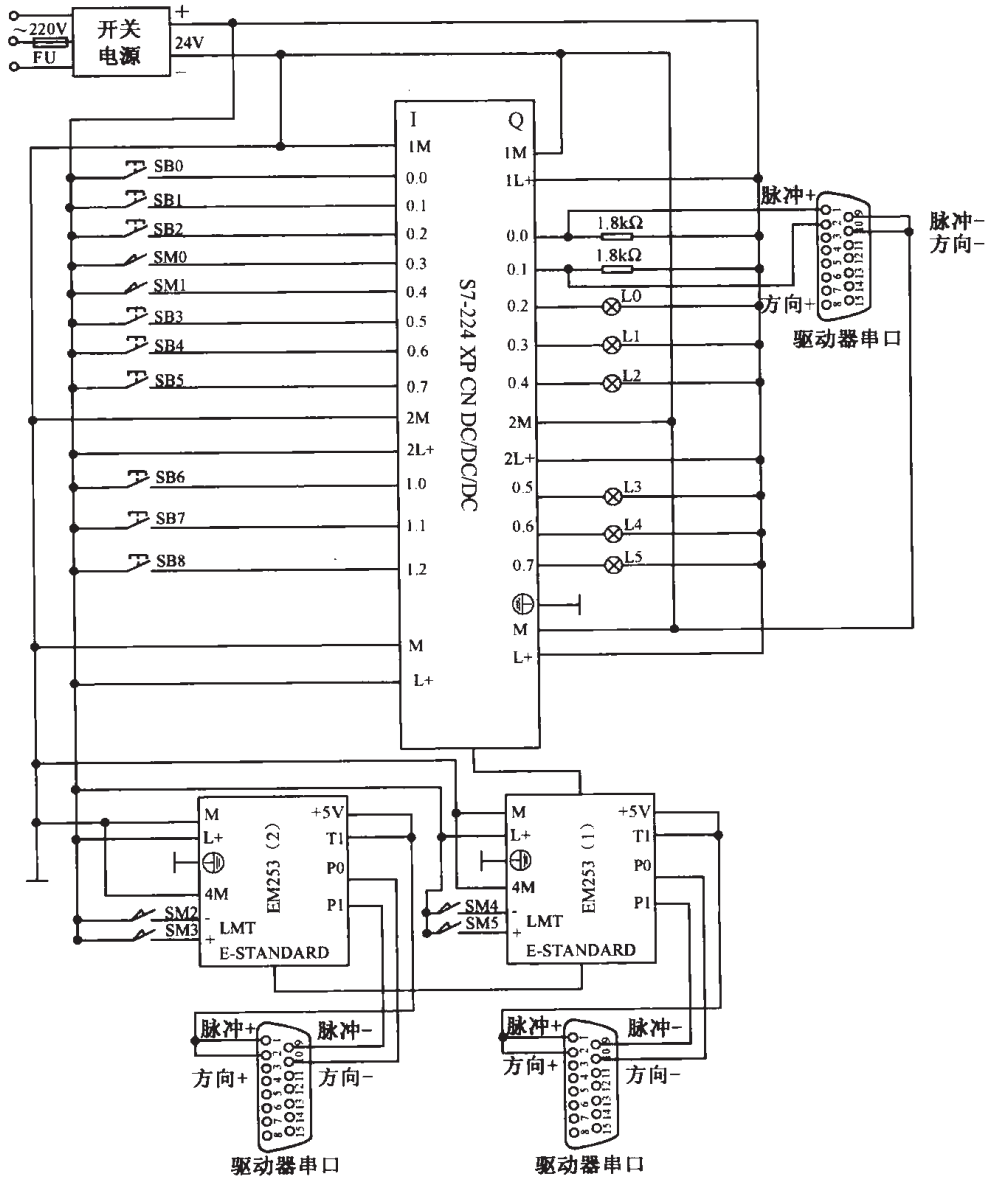


图 6-24 PLC 与现场器件安装接线图

3. PLC 控制程序

PLC 控制程序设计采用模块化设计，对每个主控单元分别写入不同的控制程序，实现不同的功能。现以 CPU224XP 为例，实现自俯仰 45°和翻滚 180°两种运动，运动程序流程如图 6-25 所示。在程序设计中，为了提高系统的可靠性，保证电动机运行到位且不超行程，采用双保险措施。正常情况下主要是以软件形式设定步进电动机的脉冲数来控制并判断电动机是否运行到位。当电动机运行途中出现断电或急停等特殊情况，在下次重新启动时，程序初始化并重新设定脉冲数，步进电动机若按照原设定的脉冲数运行，运动件必将超出行程，这时限位开关起作用，限制行程，从而保证运动的准确性，严格控制导弹俯仰 45°和翻滚 180°。该单元控制程序设计的关键是控制步进电动机，以自动俯仰 45°控制为例，运用 S7-200 CPU224XP 内置的 PTO 控制步进电动机，步进电动机控制程序由初始化、旋转方向、连锁、启动电动机（机架上升或下降）和停止电动机 5 部分组成。

① 程序初始化。在程序的第一个扫描周期（SM0.1=1），脉冲输出功能（PTO）选择参数，PTO 发生器控制电动机脉冲包络线如图 6-26 所示，输出波形包括三段：步进电动机加速（第一段）、步进电动机匀速（第二段）和步进电动机减速（第三段），其包络线表值如表 6-5 所示，启动和结束频率是 500 Hz，最大脉冲频率是 5000 Hz，目标脉冲数±170500。

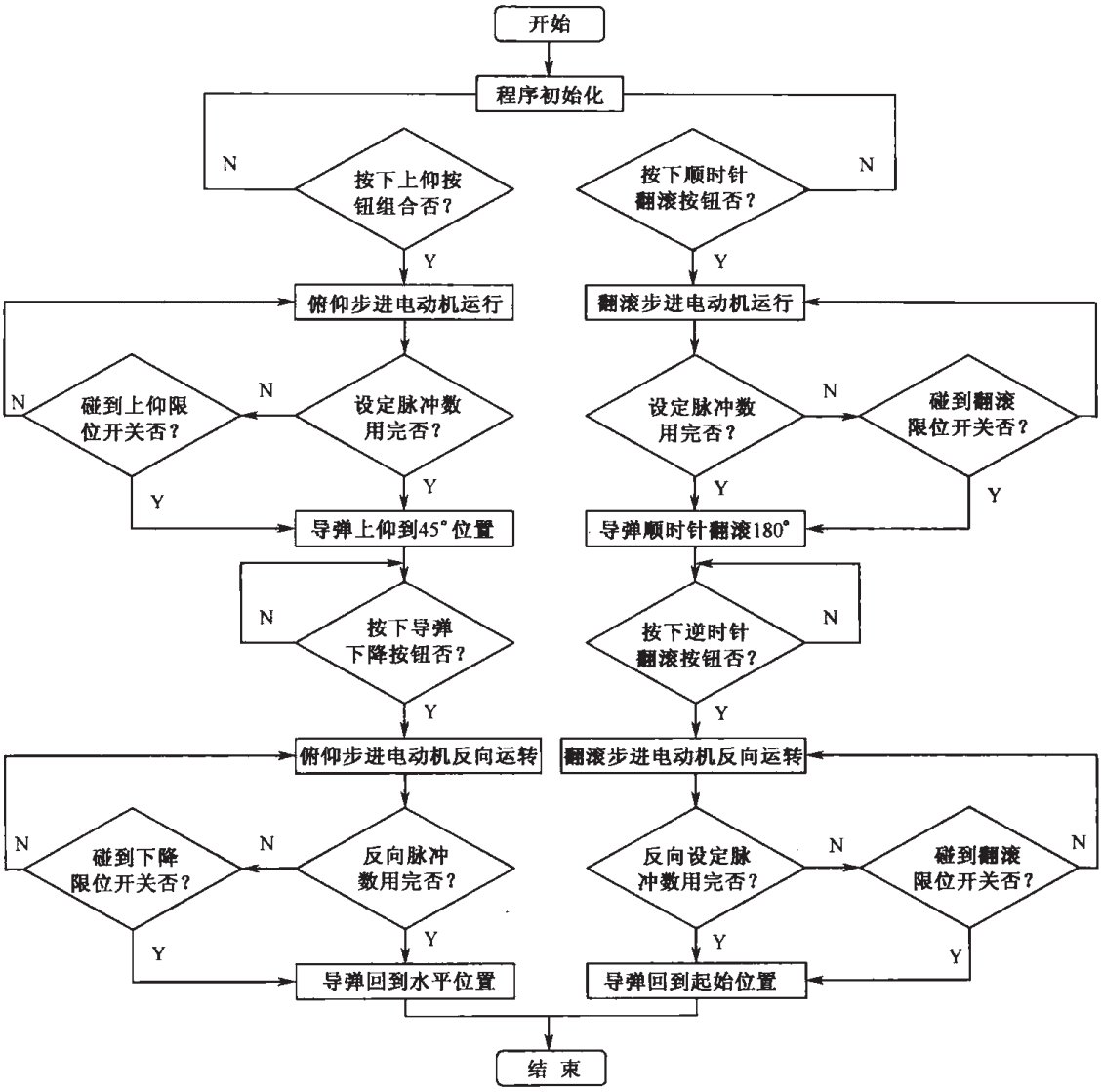


图 6-25　CPU224XP 程序流程图

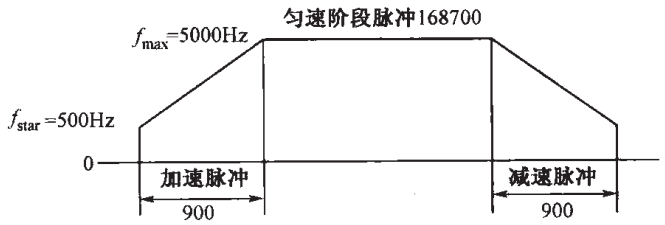


图 6-26　加减速包络曲线图

表 6-5 包络线表值

V 存储器地址	值	中 断 描 述	
VB500	3	总段数	
VW501	2000	初始周期	第一段
VW503	-2	周期增量段	
VD505	900	脉冲数	
VW509	200	初始周期	第二段
VW511	0	周期增量段	
VD513	168700	脉冲数	
VW517	200	初始周期	第三段
VW519	2	周期增量段	
VD521	900	脉冲数	

② 选择旋转方向。用接在输入端 I0.0 和 I0.1 的开关来选择转动方向。如果 I0.0=1，将输出 Q0.1 置成高电位，那么电动机顺时针转动。如果 I0.1=1，将输出 Q0.1 置成低电位，那么电动机逆时针转动。为保护电动机避免漏步，电动机转动方向的改变只能在电动机处于停止状态（M0.1=0）时进行。

③ 连锁。为保护人员和设备的安全，在按停止按钮（I0.2）之后，必须规定驱动器连锁（或称阻塞），将连锁标志 M0.2 置位（M0.2=1），立即关断驱动器。只有在 M0.2 复位（M0.2=0）后，才能重新启动电动机。当停止按钮松开后，为防止电动机的意外启动，只有在启动按钮（I0.0，I0.1）和停止按钮（I0.2）都松开后，才能将 M0.2 复位（M0.2=0），如果要再次启动电动机，则必须再发出一个启动信号。

④ 启动电动机。启动电动机执行上升和下降控制，启动电动机的条件有：按启动按钮 I0.0（上升）或 I0.1（下降）产生脉冲上升沿（从 0 升到 1）；无连锁，即连锁标志 M0.2=0；电动机处于停止状态，即操作标志 M0.1=0。如果同时具备这 3 个条件，则将 M0.1 置位（M0.1=1），控制器执行 PLS 指令，在输出端 Q0.0 输出脉冲，控制电动机转动，实现机架上升或下降。其他条件，已经在首次扫描（SM0.1=1）设置，脉冲输出基本数据置于相应的属于 PTO 的特殊存储字 SMB67 中。

⑤ 停止电动机。停止电动机的条件有：按停止按钮（I0.2）或限位开关（I0.3 或 I0.4）产生脉冲上升沿（从 0 升到 1）；电动机处于运转状态，即操作标志 M0.1=1；如果同时具备这两个条件，则将标志 M0.1 复位（M0.1=0），并中断输出端 Q0.0 的脉冲输出。在完整的脉冲序列输出后，中断程序 0 将标志 M0.1、M0.4、M0.5 复位，从而使电动机能够重新启动，指示灯熄灭。

步进电动机控制梯形图程序清单如图 6-27 所示。另两个方向的运动梯形图程序类似，在此不再介绍。

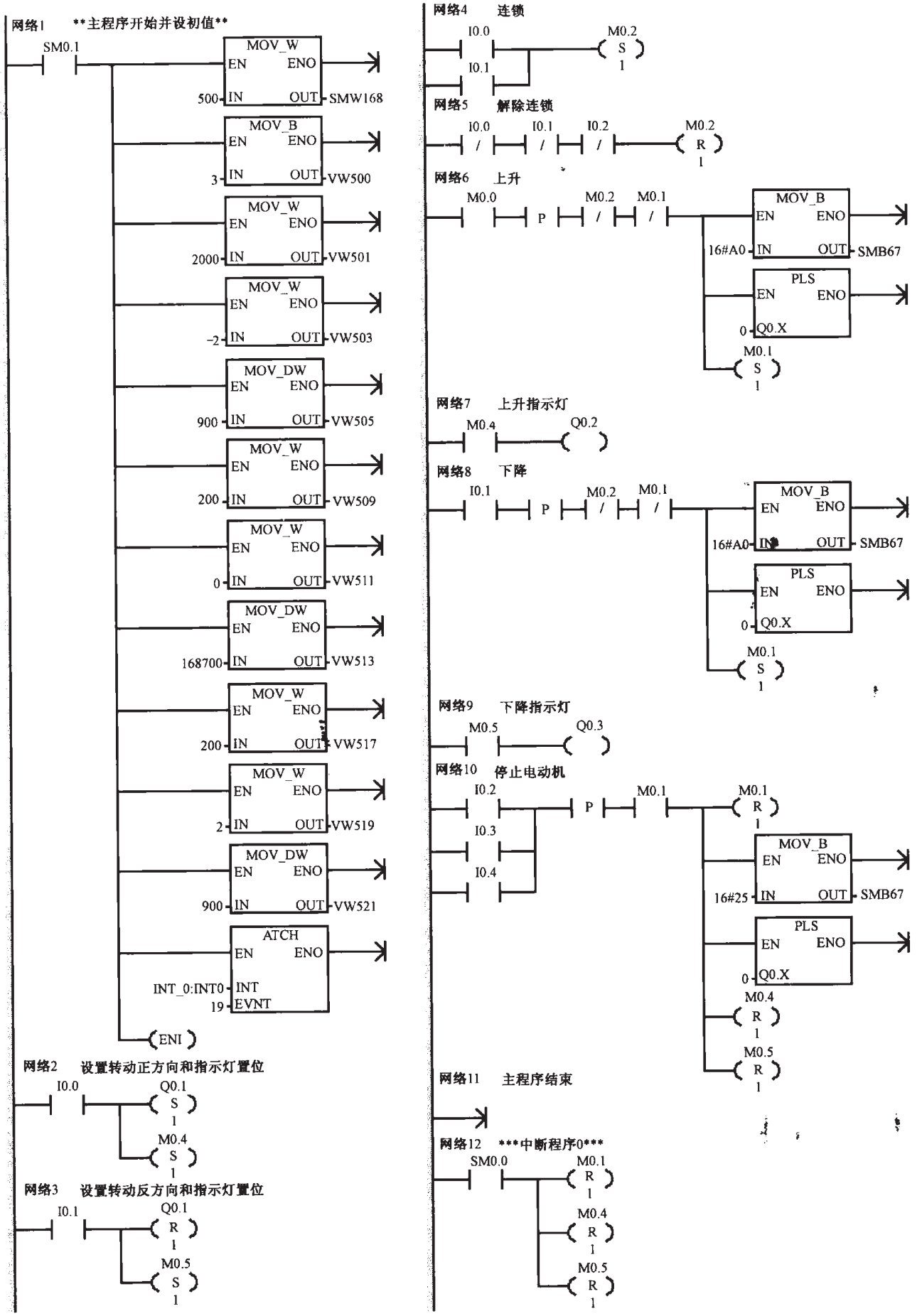


图 6-27　俯仰 45°运动的梯形图

实例分析

通过对现役某型导弹测试机架进行 PLC 控制改造，重点解决用 PLC 高速脉冲输出功能（PTO）对步进电动机控制的问题，经试验，整体性能稳定可靠、运动模拟准确、功能完备、操作方便和缩短了检测时间，提高了部队保障能力和反应能力。

思考题

1. 可编程控制器控制系统设计的原则有哪些？
2. PLC 软件设计方法主要有几种类型，试分别举例说明。
3. 移植设计法需要注意些什么问题？
4. 逻辑设计法的核心问题是什么？
5. 设计 PLC 控制汽车拐弯灯的梯形图。具体要求是：汽车驾驶台上有一个开关，有三个位置分别控制左闪灯亮、右闪灯亮和关灯。当开关扳到 S1 位置时，左闪灯亮（要求亮、灭时间各为 1 s）；当开关扳到 S2 位置时，右闪灯亮（要求亮、灭时间各为 1 s）；当开关扳到 S0 位置时，关闭左、右闪灯；如果司机开灯后忘了关灯，则过 1.5 min 后自动停止闪灯。

第 7 章　PLC 系统通信

- S7-200 PLC 通信部件介绍
- S7-200 PLC 的通信协议及指令
- PPI 通信实例
- MPI 通信实例
- PROFIBUS-DP 通信实例
- 工业以太网通信实例
- 自由口通信实例

随着工业生产规模的不断扩大，对生产管理的信息化、集成化需求的不断提高，PLC 控制系统也逐步从单机分散型控制向着多机协同的网络化控制系统发展，这就要求 PLC 系统具有灵活的通信能力。同时，伴随着通信技术的发展，PLC 系统通信的形式、结构、协议也是多种多样，并各有特点。下面以西门子 S7-200 PLC 为例对 PLC 系统通信的应用加以描述。

7.1 S7-200 PLC 通信部件介绍

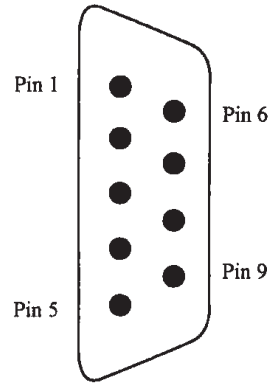
在本节中将介绍 S7-200 通信的有关部件，包括通信端口、PC / PPI 电缆、通信卡及 S7-200 通信扩展模块等。

7.1.1 通信端口

S7-200 CPU 上的通信端口采用的是 9 针 D 型 RS-485 串行接口，该端口也符合欧洲标准 EN50170 中 PROFIBUS 标准。RS-485 串行接口外形如图 7-1 所示。RS-485 串行接口引脚布置及说明如表 7-1 所示。

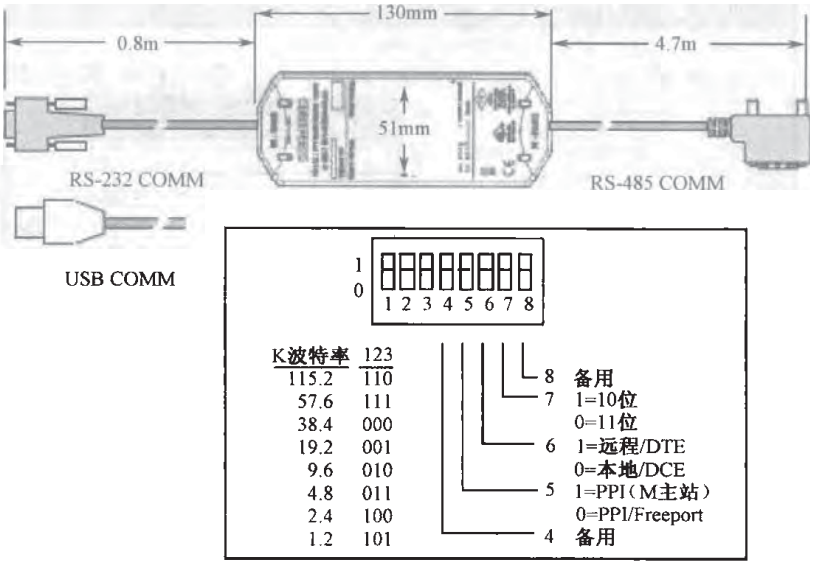
图 7-1 RS-485 串行接口外形

表 7-1 RS-485 串行接口各引脚名称

端 口	引 脚	名 称	端口 0/端口 1
	1	屏蔽	机壳地
	2	24 V 返回	逻辑地
	3	RS-485 信号 B	RS-485 信号 B
	4	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

7.1.2　PC/PPI 电缆

PC/PPI 电缆为多主站电缆，一般用于 PLC 与计算机通信，这是一种低成本的通信方式。根据与计算机接口方式的不同，PC/PPI 电缆有两种不同的形式，分别是 RS-232/PPI 多主站电缆和 USB/PPI 多主站电缆，电缆外形如图 7-2 所示。



Tx	颜　色	描　述
LED/Tx	绿色	RS-232/USB发送指示灯
Rx	绿色	RS-232/USB接收指示灯
PPI	绿色	RS-485 发送指示灯

图 7-2　PC/PPI 电缆外形图

1．PC/PPI 电缆的连接

图 7-3 表示了计算机与 PLC 之间的连接。将 PC/PPI 电缆有“PC”的 RS-232 端连接到计算机的 RS-232 通信接口，标有“PPI”的 RS-485 端连接到 CPU 模块的通信端口，拧紧两边螺钉即可。

图 7-3　通过 PC/PPI 电缆连接 PC 与 PLC

在 PC/PPI 电缆上有 8 个 DIP 开关，其中 1/2/3 号开关用于选择通信波特率。这里的选择应与编程软件中设置的波特率一致。一般选通信速率的默认值 9600bps。5 号开关为 PPI/自由口通信选择。6 号开关为远程/本地选择，7 号开关选择 10 位或 11 位 PPI 通信协议。

2. PC/PPI 电缆的通信设置

在 STEP 7-Micro/WIN 32 编程软件中选择“Communications”，双击“Set PG/PC Interface”，如图 7-4 所示。在“设置 PG/PC 接口”界面（如图 7-5 所示）中，双击 PC/PPI cable（PPI）图标，打开 PC/PPI 电缆属性设置窗口（如图 7-6 所示），选择通信速率（一般为 9.6 kbps）。

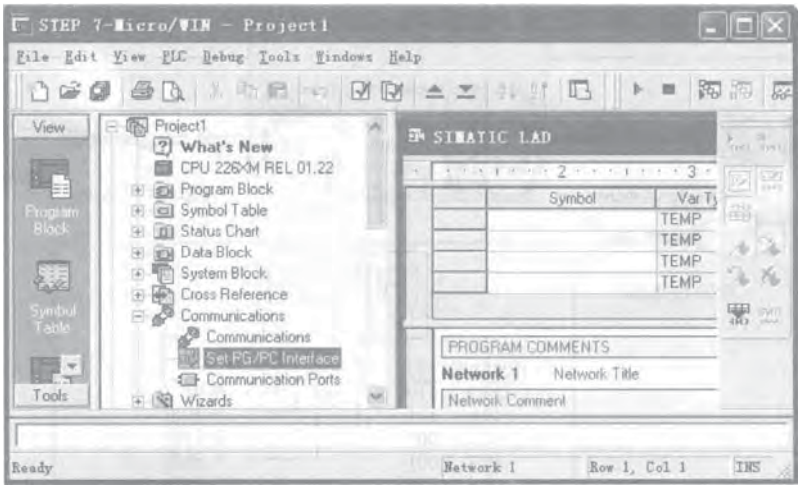


图 7-4　编程软件界面

图 7-5　PG/PC 接口选择界面

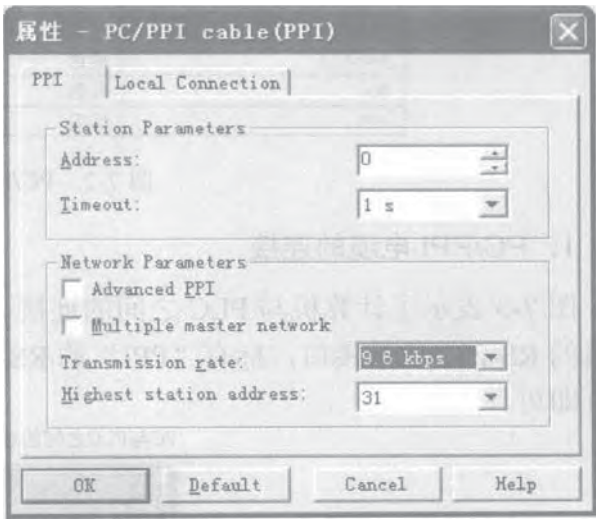


图 7-6　PC/PPI 属性设置界面

7.1.3　网络连接器

为了能够把多个设备很容易地连接到网络中，西门子公司提供两种网络连接器：一种标准网络连接器（引脚分配如表 7-1 所示）和一种带编程接口的连接器。后者允许在不影响现有网络连接的情况下，再连接一个编程站或者一个 HMI 设备到网络中。带编程接口的连接器将 S7-200 的所有信号（包括电源引脚）传到编程接口。这种连接器对于那些从 S7-200 取电源的设备（如 TD200）尤为有用。

两种连接器都有两组螺钉连接端子，可以用来连接输入连接电缆和输出连接电缆。两种连接器也都有网络偏置和终端匹配的选择开关，同时在终端位置的连接器要安装偏压和终端电阻。网络连接器如图 7-7 所示。

进行网络连接时，连接的设备应共享一个共同的参考点。参考点不同时，在连接电缆中会产生电流，这些电流会造成通信故障或损坏设备，需要将通信电缆所连接的设备进行隔离。

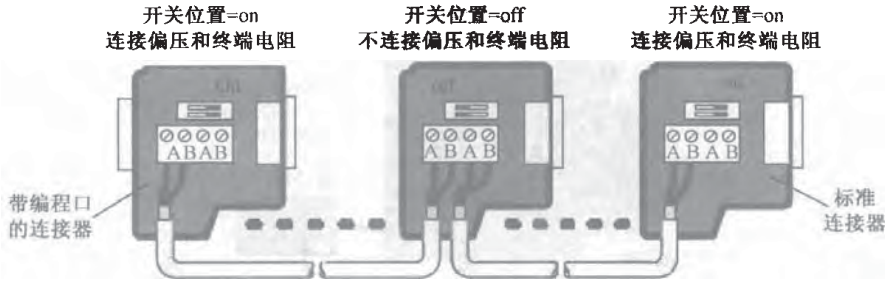


图 7-7　网络连接器

7.1.4　网络中继器

RS-485 中继器为网段提供偏压电阻和终端电阻。中继器有以下用途。

1．增加网络的长度

在网络中使用一个中继器可以使网络的通信距离扩展 50 m。如图 7-8 所示，如果在已连接的两个中继器之间没有其他节点，那么网络的长度将能达到波特率允许的最大值。在一个串联网络中，用户最多可以使用 9 个中继器，但是网络的总长度不能超过 9600 m。

图 7-8　带中继器的网络

2．为网络增加设备

在 9600 的波特率下，50 m 距离之内，一个网段最多可以连接 32 个设备。使用一个中继器允许用户在网络上再增加 32 个设备。

3．实现不同网段的电气隔离

如果不同的网段具有不同的地电位，将它们隔离会提高网络的通信质量。

一个中继器在网络中被算作网段的一个节点，但是它没有被指定的站地址。

7.1.5　EM277 PROFIBUS–DP 模块

EM277 PROFIBUS-DP 模块是专门用于 PROFIBUS-DP 协议通信的智能扩展模块。它的外形如图 7-9 所示。EM277 机壳上有一个 RS-485 接口，通过接口可将 S7-200 CPU 连接至网络，它支持 PROFIBUS-DP 和 MPI 从站协议。其他的地址选择开关可进行地址设置，地址范围为 0～99。

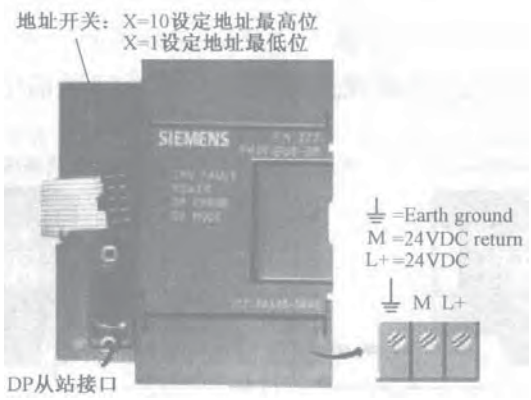


图 7-9 EM277 模块图

7.1.6 CP 243-1 和 CP 243-1 IT 模块

CP 243-1 和 CP 243-1 IT 都是一种通信处理器，设计用于在 S7-200 自动化系统中运行。它们可用于将 S7-200 系统连接到工业以太网（IE）中。通过它们可以使用 STEP 7 Micro/WIN 32，对 S7-200 进行远程组态、编程和诊断。而且，一台 S7-200 还可通过以太网与其他 S7-200、S7-300 或 S7-400 控制器进行通信。并可与 OPC 服务器进行通信。

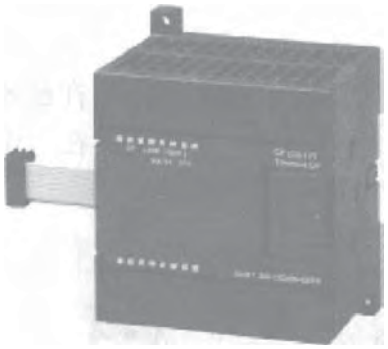


图 7-10 CP 243-1 和 CP 243-1 IT 模块外形图

另外，基于 CP 243-1 IT 的 IT 功能，可以实现监控，如果需要，还可通过 Web 浏览器，从一台联网的工控机中控制自动化系统，并将诊断报文通过 E-mail 在系统中发送。使用 IT 功能，可以非常容易地与其他计算机以及控制器系统交换全部文件。

CP 243-1 和 CP 243-1 IT 模块的外形是一致的，如图 7-10 所示。

7.2 S7-200 PLC 的通信协议及指令

西门子 S7-200 CPU 支持多种通信协议，根据所使用的机型，网络可以支持一个或多个协议。如点到点（Point-to-Point）接口协议（PPI）、多点（Multi-Point）接口协议（MPI）、自由通信接口协议、现场总线协议和工业以太网协议。在本节中将对这些网络协议进行概要介绍，如 PPI、MPI、自由口通信协议、现场总线、工业以太网等通信方式以及相关的程序指令。

7.2.1 PPI 协议

PPI 是一种主-从协议：主站器件发送请求到从站器件，从站器件响应这个请求，如图 7-11 所示。从站器件不发信息，只是等待主站的请求并对请求做出响应。主站靠一个由 PPI 协议管理的共享连接来与从站通信。PPI 并不限制与任意一个从站通信的主站数量，但是在一个网络中，主站的个数不能超过 32。

图 7-11 PPI 网络

如果在用户程序中使能 PPI 主站模式，S7-200 CPU 在运行模式下可以作主站。在使能

PPI 主站模式之后，可以使用网络读/写指令来读/写另外一个 S7-200。当 S7-200 作 PPI 主站时，它仍然可以作为从站响应其他主站的请求。

7.2.2　MPI 协议

MPI 允许主-主通信和主-从通信，如图 7-12 所示。选择何种方式依赖于设备类型。如果是 S7-300 CPU，由于所有的 S7-300 CPU 都必须是网络主站，所以应进行主/主通信方式。如果设备是 S7-200 CPU，那么就进行主/从通信方式，因为 S7-200 CPU 是从站。因此，与一个 S7-200 CPU 通信，STEP 7-Micro/WIN 应建立主-从连接。

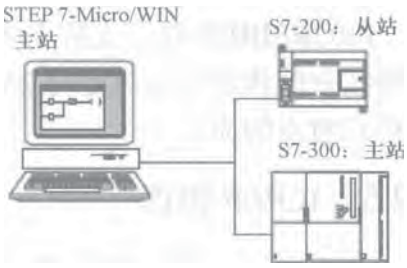


图 7-12　MPI 网络

MPI 可在任意两个网络设备之间建立单独的连接（由 MPI 协议管理）。这种连接可能是两个设备之间的非固定连接，且其他主站不能对其进行干涉。网络连接设备数量受 S7-200 CPU 或者 EM277 模块所支持的连接个数的限制。

7.2.3　自由口通信协议

自由口通信协议(Freeport Mode)方式是 S7-200 PLC 的一个很有特色的功能。S7-200 PLC 的自由通信，即用户自定义通信协议（如 ASCII 协议），数据传输率最高为 38.4 kbps。

自由口通信协议的应用，使可通信的范围大大增加，控制系统配置更加灵活、方便。应用此种方式，使 S7-200 PLC 可以使用任何公开的通信协议，并能与具有串口的外设智能设备和控制器进行通信。如打印机、条码阅读器、调制解调器、变频器和上位 PC 等。当然也可以用于两个 CPU 之间简单的数据交换。当外设具有 RS-485 接口时，可以通过双绞线进行连接，具有 RS-232 接口的外设也可以通过 PC/PPI 电缆连接起来进行自由口通信。

与外设连接后，用户程序可以通过使用发送中断、接收中断、发送指令（XMT）和接收指令（RCV）对通信口操作。在自由通信口模式下，通信协议完全由用户程序控制。另外，自由口通信模式只有在 CPU 处于 RUN 模式时才允许。当 CPU 处于 STOP 模式时，自由通信口停止，通信口转换成正常的 PPI 协议操作。

7.2.4　PROFIBUS 协议

PROFIBUS 是世界上第一个开放式现场总线标准，是用于车间级和现场级的国际标准，传输速率最大为 12 Mbps，响应时间的典型值为 1ms，使用屏蔽双绞线电缆（最长 9.6 km）或光缆（最长 90 km），最多可接 127 个从站。其应用领域覆盖了从机械加工、过程控制、电力、交通到楼宇自动化的各个领域。

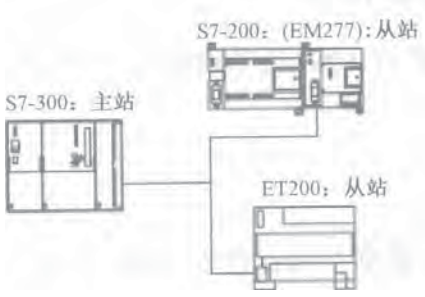


图 7-13　PROFIBUS 网络

在 S7-200 PLC 的 CPU 中，CPU22X 都可以通过增加 EM277 PROFIBUS-DP 扩展模块的方法支持 PROFIBUS-DP 网络协议。最高传输速率可达 12 Mbps。在采用 PROFIBUS 的系统中，对于不同厂家所生产的设备不需要对接口进行特别的处理和转换，就可以实现通信。

PROFIBUS 协议通常用于实现与分布式 I/O(远程 I/O)的高速通信。PROFIBUS 网络通常有一个主站和若干个 I/O 从站，如图 7-13 所示。主站能够控制总线，并通过配置可

以知道 I/O 从站的类型和站号。当主站获得总线控制权后，可以主动发送信息。从站可以接收信号并给予响应，但没有控制总线的权力。当主站发出请求时，从站回送给主站相应的信息。PRORFIBUS 除了支持主/从模式，还支持多主/多从的模式。对于多主站的模式，在主站之间按令牌传递顺序决定对总线的控制权。取得控制权的主站，可以向从站发送和获取信息，实现点对点的通信。

7.2.5 TCP/IP 协议

为了实现企业管理自动化与工业控制自动化的无缝接合，工业以太网成为了工业控制系统中一种新的工业通信网络。西门子公司也已将工业以太网运用于工业控制领域，并采用 ASI，PROFIBUS 和工业以太网构成了全方位的工厂生产管理监控系统。

通过以太网扩展模块（CP243-1）或互联网扩展模块（CP243-1 IT），S7-200 将能支持 TCP/IP 以太网通信。图 7-14 为西门子 PLC 通过以太网通信的系统图。

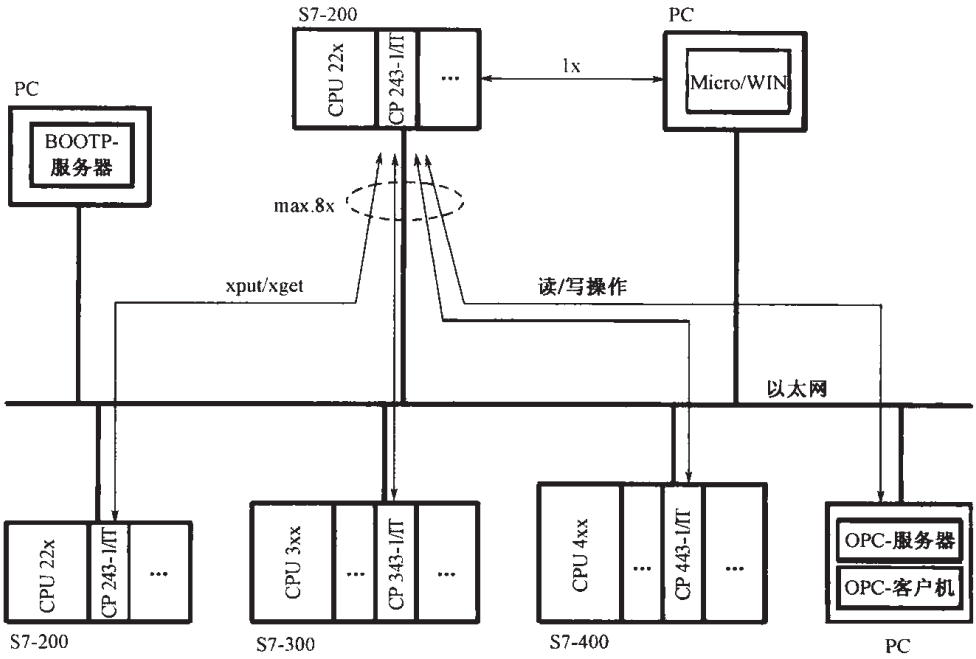


图 7-14 西门子 PLC 通过以太网通信系统图

7.2.6 通信指令

网络的通信功能是通过通信程序来实现的。因此，就需要了解 PLC 提供的通信指令。S7-200 PLC 提供的通信指令主要有：网络读与网络写指令、发送与接收指令、获取与设置通信口地址指令等，下面对各指令的格式、要求和用法分别予以介绍。

1. 网络读与网络写指令 NETR（Network Read）/NETW（Network Write）

网络读/网络写指令格式如图 7-15 所示。

TBL：缓冲区首址，操作数为字节。

PROT：操作端口，CPU226 为 0 或 1，其他机型只能为 0。

网络读 NETR 指令是通过端口（PROT）接收远程设备的数据并保存在表（TBL）中。可从远方站点最多读取 16 字节的信息。

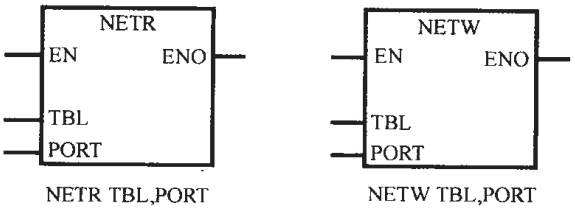


图 7-15　网络读/网络写指令 NETR/ NETW

网络写 NETW 指令是通过端口（PROT）向远程设备写入在表（TBL）中的数据。可向远方站点最多写入 16 字节的信息。

在程序中可以写任意多 NETR/NETW 指令，但在任意时刻最多只能有 8 个 NETR 或 8 个 NETW 指令、4 个 NETR 和 4 个 NETW 指令，或者 2 个 NETR 和 6 个 NETW 指令有效。

TBL 表的参数定义如表 7-2 所示。其中字节 0 的各标志位及错误码（4 位）的含义如表 7-3 所示。

表 7-2　TBL 表的参数定义

<table>
<tr><td>地　址</td><td colspan="5">定义与说明</td></tr>
<tr><td>字节 0</td><td>D</td><td>A</td><td>E</td><td>0</td><td>错误码</td></tr>
<tr><td>字节 1</td><td colspan="5">远程站点的地址（被访问的 PLC 地址）</td></tr>
<tr><td>字节 2</td><td colspan="5" rowspan="4">指向远程站点的数据区指针（双字）（指向远程 PLC 存储区中的数据的间接指针）</td></tr>
<tr><td>字节 3</td></tr>
<tr><td>字节 4</td></tr>
<tr><td>字节 5</td></tr>
<tr><td>字节 6</td><td colspan="5">数据长度（1～16 字节）（远程站点被访问的字节数）</td></tr>
<tr><td>字节 7</td><td colspan="2">数据字节 0</td><td colspan="3" rowspan="4">接收或发送数据区：保存数据的 1～16 字节，其长度在“数据长度”字节中定义。对于 NETR 指令，此数据区指执行 NETR 后存放从远程站点读取的数据区。对于 NETW 指令，此数据区指执行 NETW 前发送给远程站点的数据存储区</td></tr>
<tr><td>字节 8</td><td colspan="2">数据字节 1</td></tr>
<tr><td>…</td><td colspan="2">…</td></tr>
<tr><td>VB122</td><td colspan="2">数据字节 15</td></tr>
</table>

表 7-3　缓冲区首字节标志位的含义

标 志 位		定　义	说　明
D		操作已完成	0=未完成，1=功能完成
A		激活（操作已排队）	0=未激活，1=激活
E		错误	0=无错误，1=有错误
4 位错误代码	0	无错误	
	1	超时错误	远程站点无响应
	2	接收错误	有奇偶错误，帧或校验和出错
	3	离线错误	重复的站地址或无效的硬件引起冲突
	4	排队溢出错误	多于 8 条的 NETR/NETW 指令被激活
	5	违反通信协议	没有在 SMB30 中允许 PPI，就试图使用 NETR/NETW 指令
	6	非法参数	NETR/NETW 表中包含非法或无效的参数值
	7	没有资源	远程站点忙（正在进行上装或下载操作）
	8	第七层错误	违反应用协议
	9	信息错误	错误的数据地址或错误的数据长度

2. 发送与接收指令 XMT（Transmit）/RCV（Receive）

发送与接收指令如图 7-16 所示。

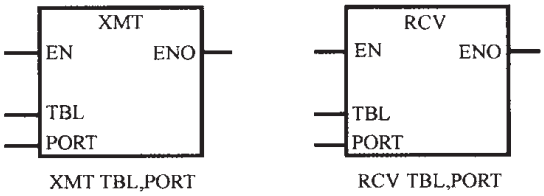


图 7-16 发送与接收指令

发送指令 XMT 启动自由端口模式下数据缓冲区（TBL）的数据发送，通过指定的通信端口（PORT），发送存储在数据缓冲区（TBL）中的信息。

XMT 指令可以方便地发送 1～255 字符，如果有中断程序连接到发送结束事件上，在发送完缓冲区的最后一个字符时，端口 0 会产生中断事件 9，端口 1 会产生中断事件 26。也可以监视发送完成状态位 SM4.5 和 SM4.6 的变化，而不是用中断进行发送。

TBL 指定的发送缓冲区的格式如图 7-17 所示，起始字符和结束字符是可选项，第一个字节“字符数”是要发送的字节数，它本身并不发送出去。

字符数	起始字符	数据区	结束字

图 7-17 TBL 数据缓冲区格式

如果将字符数设置为 0，然后执行 XMT 指令，以当前的波特率在线路上产生一个 16 位的间断（break）条件。发送间断语和发送任何其他信息一样，采用相同的处理方式。完成间断时产生一格 XMT 中断，SM4.5 或 SM4.6 反映 XMT 的当前状态。

实例 81：检测 XMT 指令对数据的发送

实例说明

本实例主要实现通过检测一些特殊存储器的状态来获知 XMT 指令的执行情况。通过本实例，认识通信指令 XMT 的使用方法。

实例实现

如图 7-18 所示，通过子程序对自由口通信进行初始化设置。其通信协议设置为：自由通信口模式，波特率为 9600 波特，无奇偶校验，每字符 8 位。然后定时器进行定时 1.5s，时间到后 PLC 开始发送数据，同时输出口 Q0.5 置 1，当数据发送完后发生中断事件 9，这样就会执行中断程序，使得输出口 Q0.5 产生周期为 1 的方波信号，同时中断程序与中断事件分离。如果在 Q0.5 的输出端接上灯泡，当发现灯泡点亮时，代表 PLC 开始发送数据，当发送完成后灯泡就会开始闪烁。

图 7-18 所对应的指令如下：

```
//主程序
LD      SM0.1          //首次扫描初接通
CALL    初始化：SBR_0   //调用初始化程序
```

```
LDN             T37
TON             T37,+15          //定时器定时 1.5s
LD              T37              //定时器定时到
XMT             VB20 , 0         //利用自由口发送 VB20 的数据
S               Q0.5 , 1         //将 Q0.5 置位
LD              SM0.0
ATCH            INT_0:INT0,9
//初始化子程序 SBR_0
LD              SM0.7            //PLC 在运行状态为 1，保证采取自由通信模式
MOVB            16#09, SMB30     //设置通信协议，自由通信口模式
                                 //波特率为 9600 波特，无奇偶校验，每字符 8 位
S               Q0.0 ,1          //将 Q0.0 置位
//中断程序 0
LD              SM0.5            //产生周期为 1s 的方波信号
R               Q0.5, 1          //使得 Q0.5 的输出为方波信号
DTCH            9                //与中断事件 9 分离
```

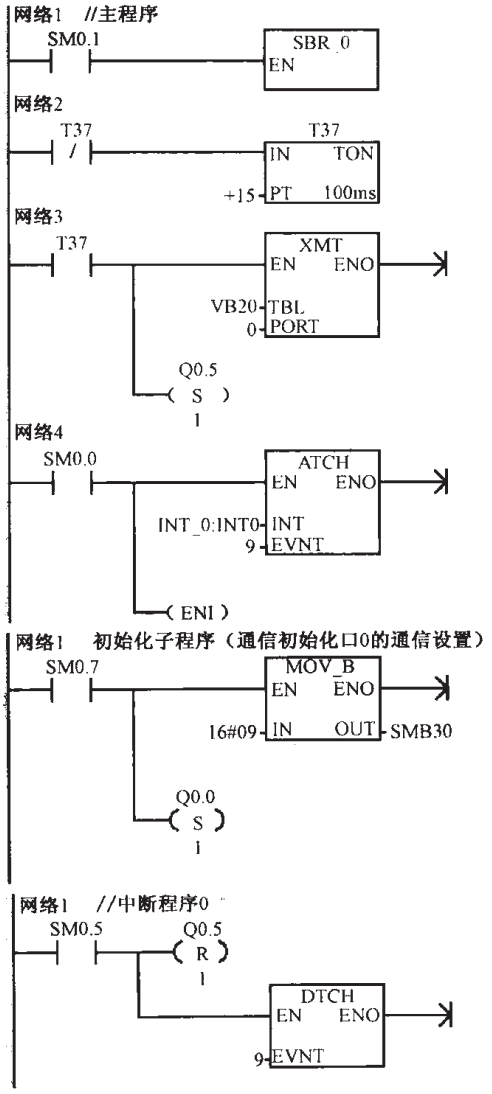


图 7-18　检测 XMT 指令发送数据程序

实例分析

从程序图中可以看出，将通信完成的中断事件与状态标志位相连接，即可对通信指令 XMT 的执行状态进行显示。实际应用中可通过类似方法，将 XMT 的执行状态通过中断事件与其他操作相连接，达到相应的控制目的。

在本章下面几节中，将通过实例对 S7-200 PLC 通信系统的应用加以说明。

7.3 PPI 通信实例

实例 82：两台 S7-200 实现 PPI 通信

实例说明

本实例主要实现两台 S7-200PLC 通过 PORT0 口互相进行 PPI 通信。通过本实例，了解 PPI 通信的应用。

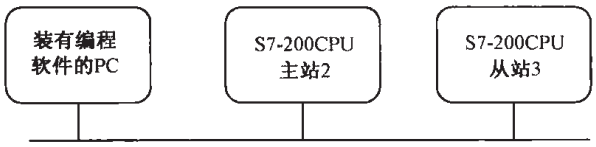


图 7-19　S7-200 CPU 之间的 PPI 通信网络

实例实现

图 7-19 是通信系统的网络配置图。系统将完成用甲机的 I0.0～I0.7 控制乙机的 Q0.0～Q0.7，用乙机的 I0.0～I0.7 控制甲机的 Q0.0～Q0.7。甲机为主站，站地址为 2；乙机为从站，站地址为 3，编程用的计算机站地址为 0。系统通信的实现过程如下。

1．端口设置

分别用 PC/PPI 电缆连接各个 PLC。打开 STEP 7-Micro/WIN 编程软件，如图 7-20 所示，选中“Communications”打开，双击其子项“Communication Ports”，打开通信口设置界面，如图 7-21 所示。在对甲机进行设置时，将“Port 0”口的“PLC Address”设置为 2，选择“Baud Rate”为 9.6 kbps。然后把设置好的参数下载到 CPU 中（通过单击  图标完成）。用同样方法设置乙机时，将“Port 0”口的“PLC Address”设置为 3，选择“Baud Rate”也为 9.6 kbps。

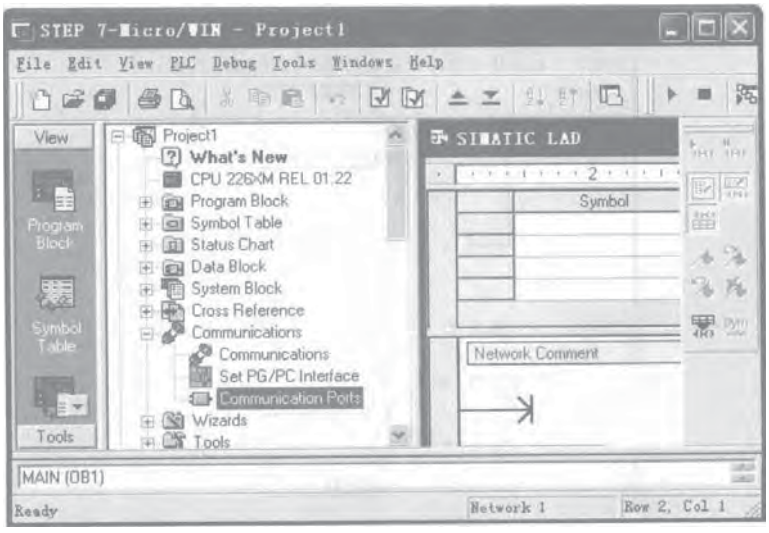


图 7-20　打开编程软件

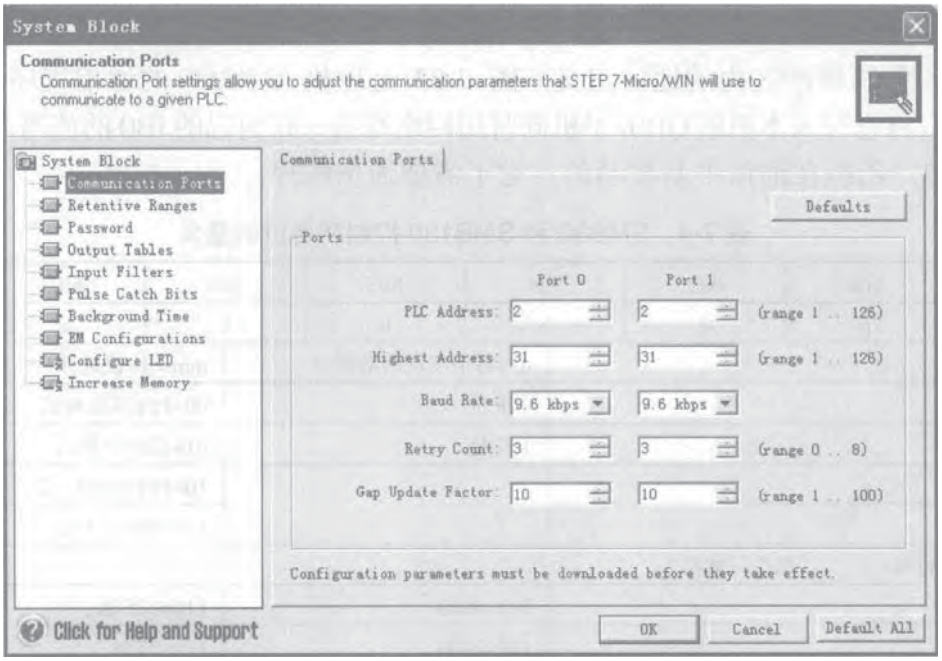


图 7-21　设置通信端口

2. 建立连接

连接好网线，双击“Communications”的子项“Communications”，打开通信连接界面，如图 7-22 所示。

双击通信刷新图标，编程软件将会显示出网站中站号为 2 和 3 的两个子站。双击某一个子站的图标，编程软件将和该子站建立连接，可以对它进行下载、上装和监视等通信操作。

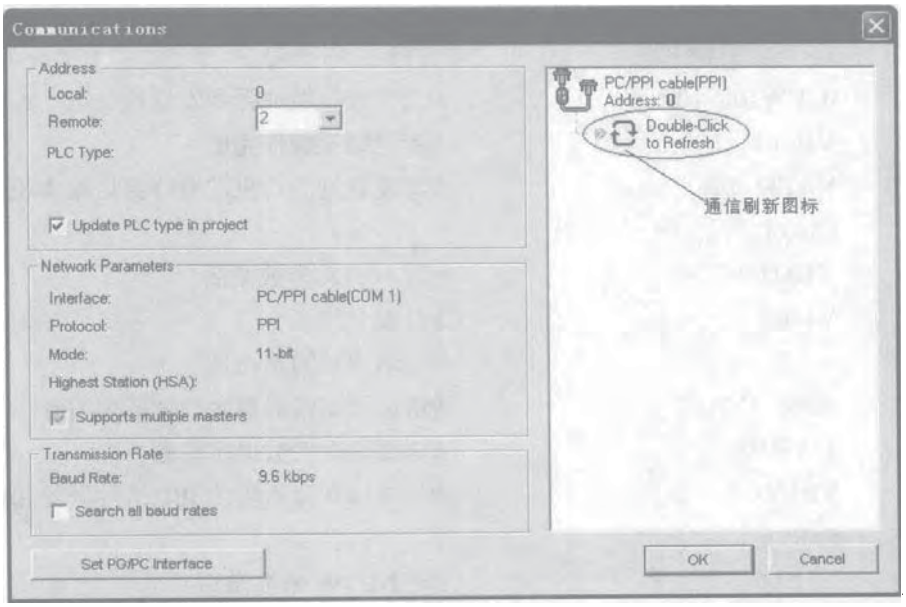


图 7-22　通信连接界面

3. 输入、编译通信程序

将编译通过的通信程序下载到站号为 2 的 CPU 模块中（该 CPU 为主站），并把两台 PLC 的工作方式开关置于 RUN 位置，分别改变两台 PLC 输入信号状态，可以观察到通信结果。

通信程序是用网络读/写指令完成的。其中 SMB30 是 S7-200 PLC PORT0 通信口的控制

字（SMB130 是 S7-200 PLC PORT1 通信口的控制字），各位表达的意义如表 7-4 所示。表 7-5 是甲机的网络读/写缓冲区内的地址定义，图 7-23 是甲机（主站）的通信程序。甲机读取乙机 IB0 的值后，将它写入本机的 QB0，甲机同时用网络写指令将自己的 IB0 的值写入乙机的 QB0。

在本例中，乙机在通信中是被动的，它不需要通信程序。

表 7-4 SMB30 和 SMB130 控制字各位的意义

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
p	p	d	b	b	b	m	m
pp：校验选择			d：每个字符的数据位		mm：协议选择		
00=不校验			0=8 位		00=PPI/从站模式		
01=偶校验			1=7 位		01=自由口模式		
10=不校验					10=PPI/主站模式		
11=奇校验					11=保留（未用）		
bbb：自由口波特率 （单位：波特）							
000=38400			011=4800		110=115.2k		
001=19200			100=2400		111=57.6k		
010=9600			101=1200		注：查看 CPU 版本		

表 7-5 缓冲区各字节的定义

字节意义	状态字节	远程站地址	远程站数据区指针	读写的数据长度	数据字节
NETR 缓冲区	VB100	VB101	VD102	VB106	VB107
NETW 缓冲区	VB110	VB111	VD112	VB116	VB117

如图 7-23 所示程序对应的语句为：

```
LD      SM0.1
MOVB    16#0A, SMB30          //PPI 主站模式
FILL    0, VW100, 10          //清空接收缓冲区和发送缓冲区
LD      V100.7                //若网络读操作完成
MOVB    VB107, QB0            //将读取到的乙机的 IB0 穿送给本机的 QB0
LDN     SM0.1
AN      V100.6                //若 NETR 未被激活
AN      V100.5                //且没有错误
MOVB    3, VB101              //送远程站的站地址
MOVD    &IB0, VD102           //送远程站的数据区指针的值 IB0
MOVB    1, VB106              //送要读取的数据字节数
NETR    VB100, 0              //从端口 0 读乙机的 IB0,缓冲区的起始地址为 VB100
LDN     SM0.1
AN      V110.6                //若 NETW 未被激活
AN      V110.5                //且没有错误
MOVB    3, VB111              //送远程站的站地址
MOVD    &QB0, VD112           //送远程站的数据区指针的值 QB0
MOVB    1, VB116              //送要写入的数据字节数
MOVB    IB0, VB117            //将本机的 IB0 的值写入发送数据缓冲区的数据区
NETW    VB110, 0              //从端口 0 写乙机的 QB0,缓冲区的起始地址为 VB110
```

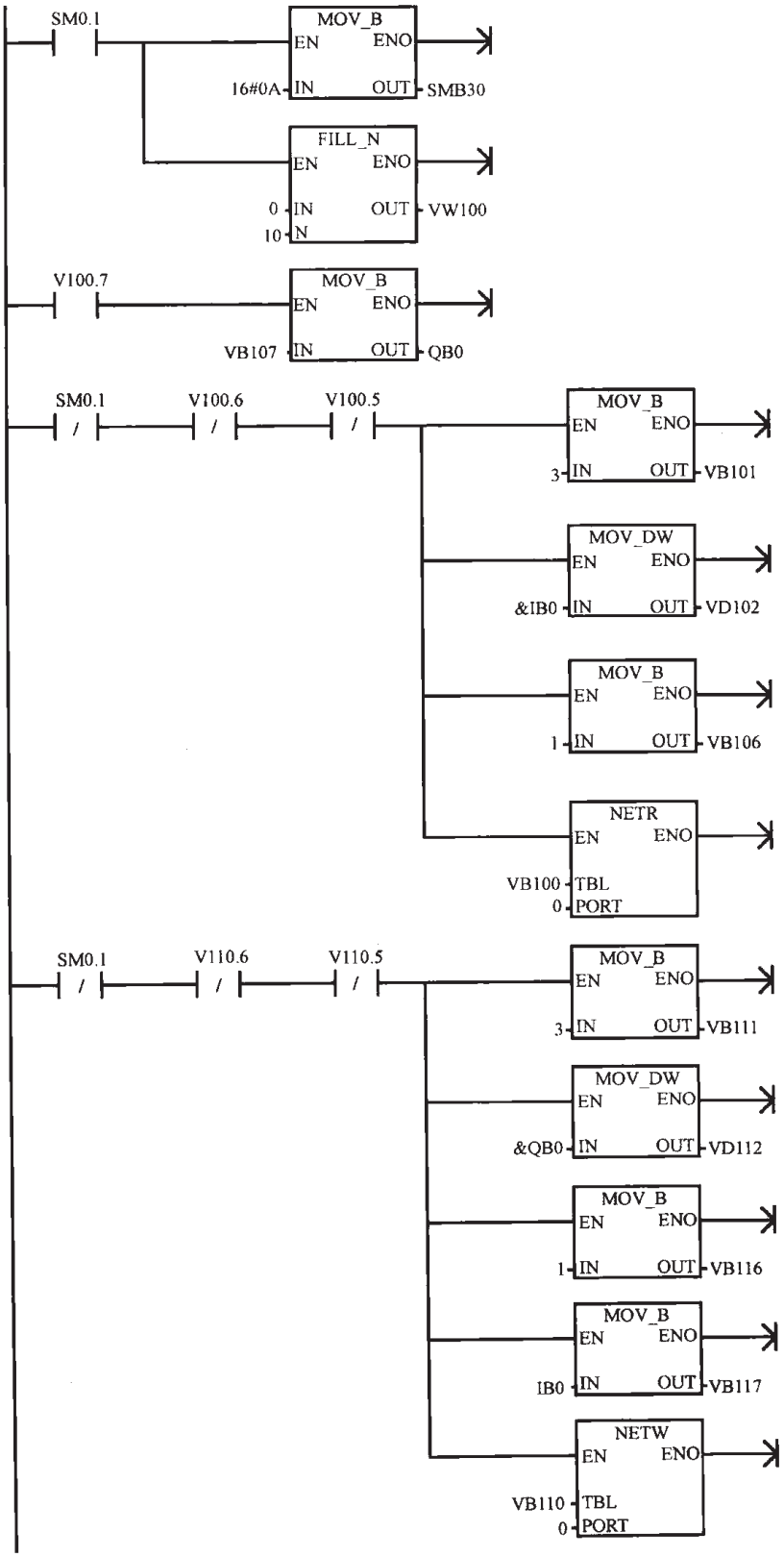


图 7-23　两台 S7-200 实现 PPI 通信

实例分析

PPI 通信的实现是非常容易的，在相应的通信口组态完成后，仅需在主站通过激活 NETR 和 NETW 指令就可以直接完成了，而从站无需任何程序。

实例 83：多台 S7-200PLC 实现 PPI 通信

实例说明

本实例主要通过三台 S7-200 PLC 的 PPI 通信，进一步说明 PPI 通信的使用过程。

实例实现

三台 S7-200 PLC 通过 PORT0 口进行通信，甲机为主站（站号为 2），乙机和丙机为从站（乙机站号为 3，丙机站号为 4）。在控制功能上实现乙机的 I0.0 启动丙机的电动机星形-三角形启动器，乙机 I0.1 停止丙机的电动机转动；丙机的 I0.0 启动乙机的电动机星形-三角形启动器，丙机 I0.1 停止乙机的电动机转动。PPI 通信程序是由甲机完成的。网络系统图如图 7-24 所示，乙机和丙机的 I/O 分配如表 7-6 所示。

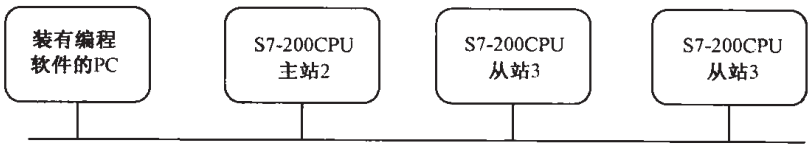


图 7-24　多从站 PPI 通信网络

表 7-6　乙机与丙机的 I/O 分配

乙机（从站，站号为 3）	丙机（从站，站号为 4）
I0.0 启动丙机电动机	I0.0 启动乙机电动机
I0.1 停止丙级电动机	I0.1 停止乙级电动机
Q0.0 星形	Q0.0 星形
Q0.1 三角形	Q0.1 三角形
Q0.2 主继电器	Q0.2 主继电器

本例中的端口设置与网络连接与实例 82 完全相同。PPI 通信程序在主站上完成（甲机）如图 7-25 所示，两个从站分别完成各自的星形-三角形启动，乙机程序如图 7-26 所示，丙机程序如图 7-27 所示。

图 7-25 的语句为：

```
LD      SM0.1
MOVB    16#0A, SMB30          //定义端口 0 为 PPI 主站
LDN     T37
TON     T37, 1                //利用定时器，每 100ms 读/写网络一次
LD      T37
MOVB    3, VB301
MOVD    &IB0, VD302
MOVB    1, VB306
NETR    VB300, 0              //读从站 3 数据，把 3 号站 IB0 读到主站 VB307 中
LD      T37
MOVB    4, VB401
MOVD    &VB10, VD402
MOVB    1, VB406
NETW    VB400, 0              //把主站 VB307 发送到从站 4 的 VB10 中
LD      T37
MOVB    4, VB501
```

```
MOVD    &IB0, VD502
MOVB    1, VB506
NETR    VB500, 0          //读从站 4 数据，把 4 号站 IB0 读到主站 VB507 中
LD      T37
MOVB    3, VB601
MOVD    &VB20, VD602
MOVB    1, VB606
NETW    VB600, 0          //把主站 VB507 发送到从站 3 的 VB20 中
```

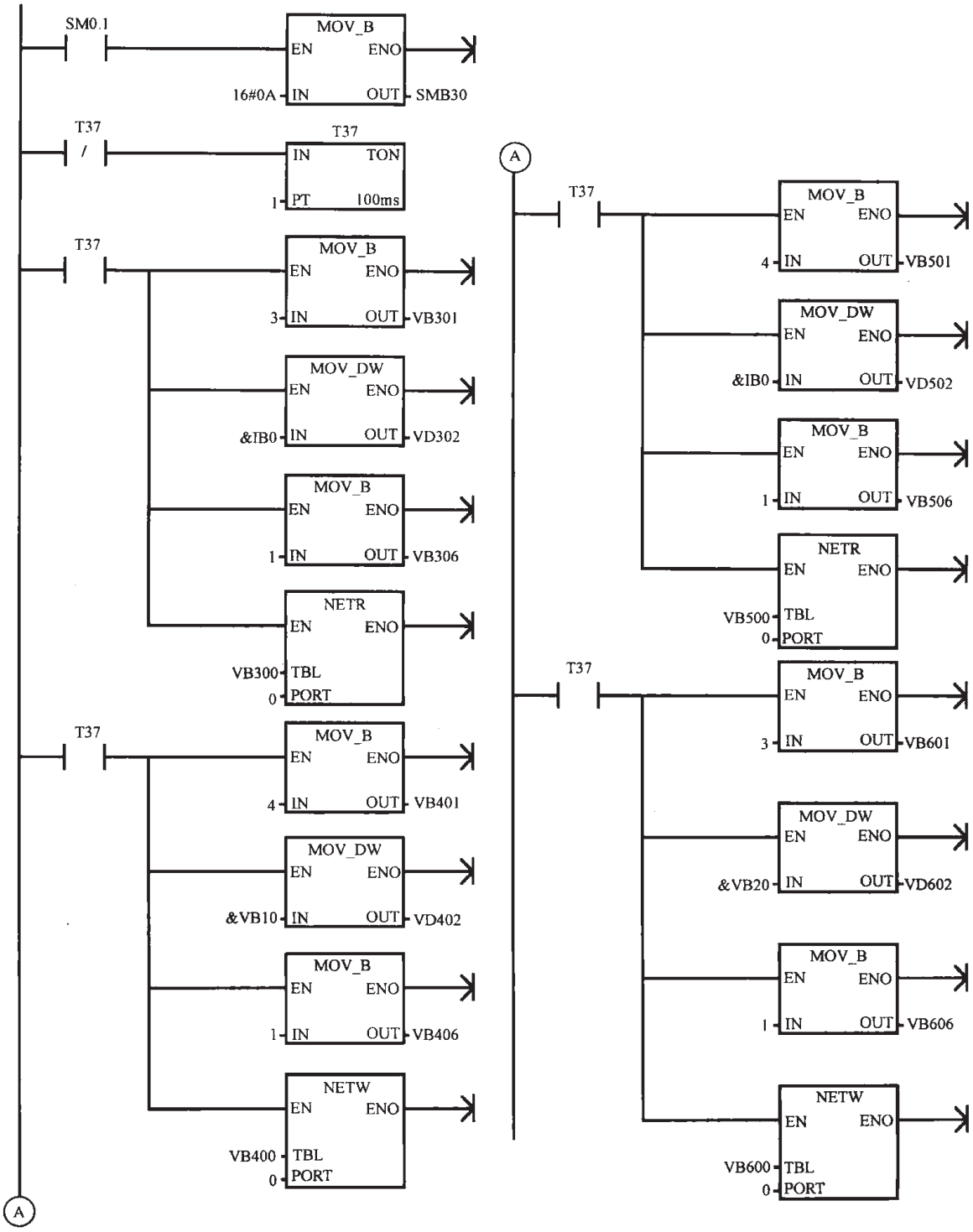


图 7-25　甲机通信程序

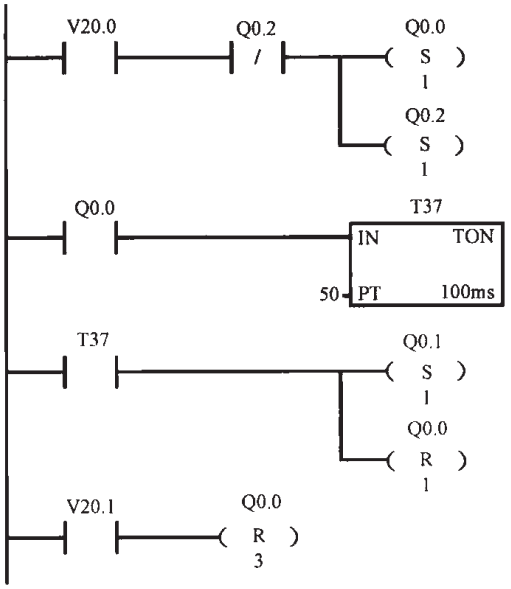


图 7-26 乙机通信程序

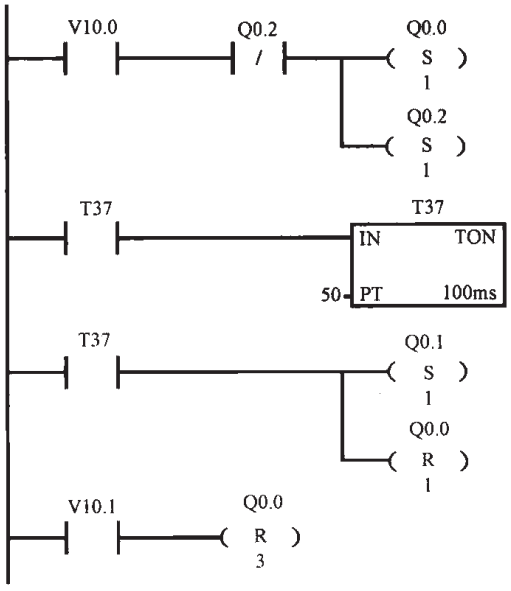


图 7-27 丙机通信程序

实例分析

PPI 的通信程序在主站完成，从站只是完成各自的功能程序，并被动地接受主站的管理。

7.4 MPI 通信实例

西门子 PLC S7-200/300/400CPU 上的 RS-485 接口不仅是编程接口，同时也是一个 MPI 的通信接口，不增加任何硬件就可以实现 PG/OP、全局数据通信及少量数据交换的 S7 通信等功能。MPI 网络的通信速率是 19.2 kbps～12 Mbps，最多支持连接 32 个节点，最大通信距离为 50 m。MPI 网络节点通常可以挂 S7 PLC、人机界面、编程设备、智能型 ET200S、RS485 中继器等网络元件。西门子 PLC 之间的 MPI 通信有三种方式：全局数据包通信方式、无组态连接通信方式、组态连接通信方式。其中，无组态连接通信方式又分为双边编程通信方式与单边编程通信方式。S7-200 PLC 可以进行单边编程通信方式。下面就对这几种 MPI 通信方式进行详细的说明。

实例 84：全局数据包通信方式

实例说明

通过两台 S7-300 PLC 实现全局数据包通信方式。这种通信方式只能在 S7-300 和 S7-400 之间进行。用户不需要编写通信程序，在硬件组态时设置好所有 MPI 通信 PLC 站上的发送区与接收区就可以了。

实例实现

1．S7-300 PLC 工作站组态

打开 STEP 7，分别组态两个 S7-300 PLC 通信站。首先在 STEP 下建立一个新项目，并保存到指定的文件夹，完成后的界面如图 7-28 所示。然后插入 S7-300 PLC 工作站 1，如图 7-29、图 7-30 所示。利用同样步骤插入工作站 2，如图 7-31 所示。

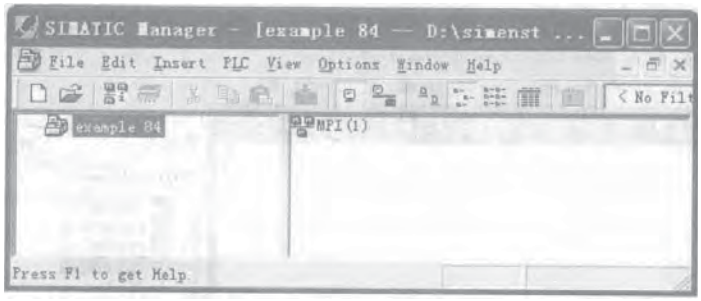


图 7-28　创建新项目

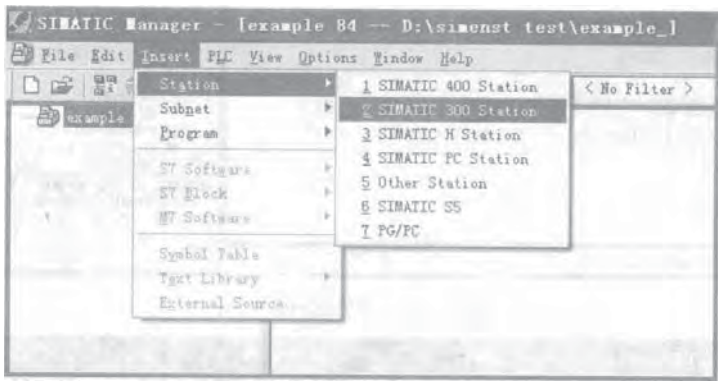


图 7-29　插入工作站

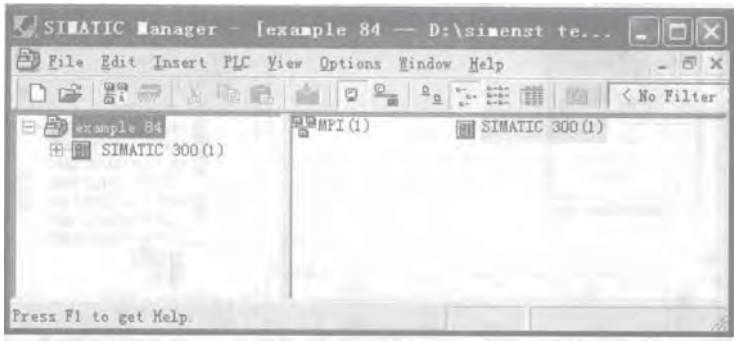


图 7-30　工作站 1

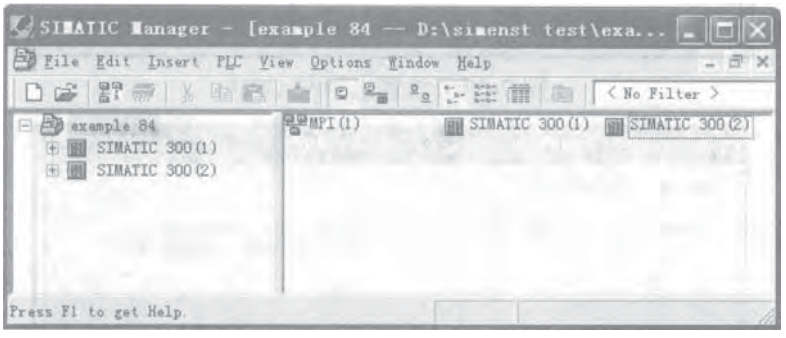


图 7-31　工作站 2

单击界面中左侧组态框中的 SIMATIC 300（1），在右侧组态框中就会出现 Hardware 图标，双击这一图标后出现的硬件配置界面如图 7-32 所示。打开这一界面右侧组态框中的 SIMATIC 300 图标，在下属菜单中，依次按照槽架、电源、CPU 的顺序放置模块，如图 7-33 所示。当选择 CPU 模块时会出现添加网络界面如图 7-34 所示，单击 OK 添加网络。完成后再依次放置好其他 I/O 模块。

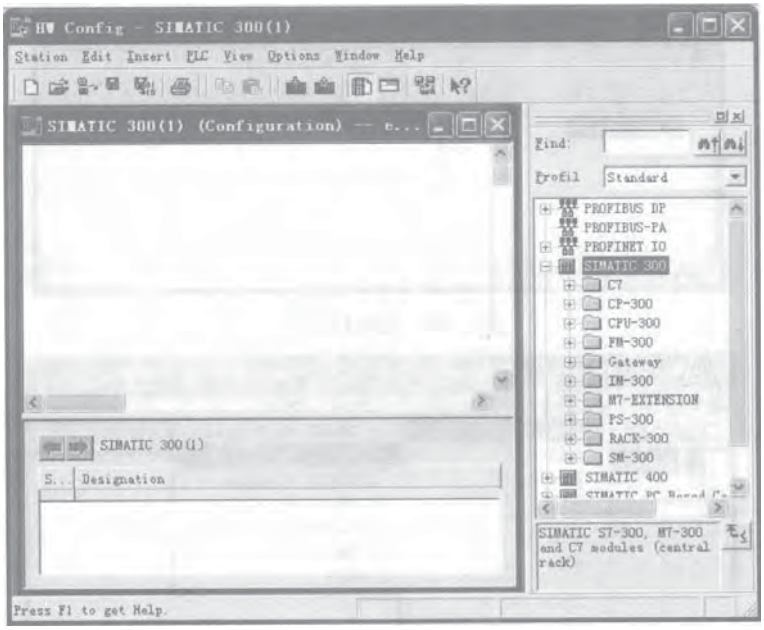


图 7-32 硬件配置界面

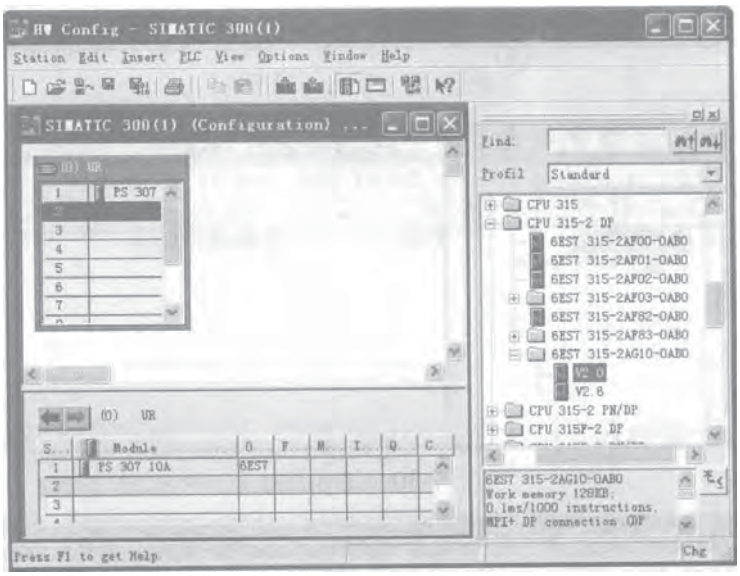


图 7-33 放置电源与 CPU

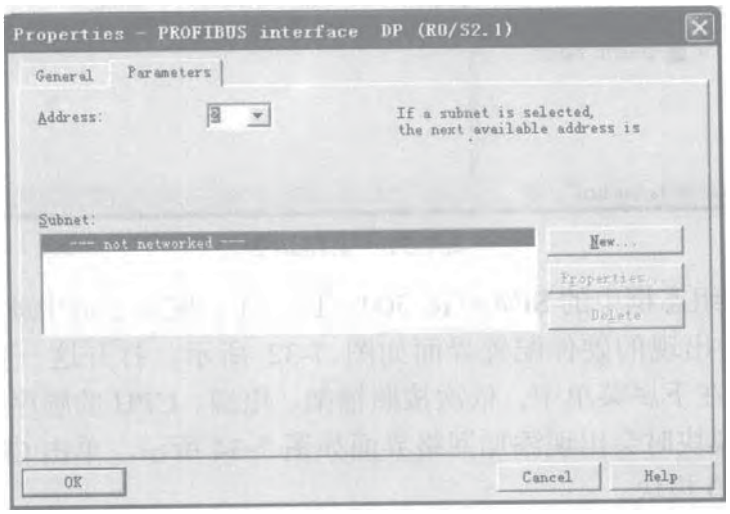


图 7-34 添加网络

2．MPI 通信网络组态

双击硬件配置界面左侧组态框中 2 号槽架位置（CPU 槽），如图 7-35 所示，打开 CPU 属性界面（如图 7-36 所示）。在接口框中（interface）单击属性（Properties）按钮，就可以看到在 MPI 属性窗口中已添加了 MPI(1)网络，设置本站地址为 2，如图 7-37 所示。单击图标，编译并保存后，单击下载图标，选中 2 号站，将程序下载。这样 2 号 MPI 站就组态完毕。另一个 MPI 站的组态过程与上述过程相同，不再重复。

下面组态 MPI 网络的 2 号站和 4 号站的发送与接收数据区。返回到 SIMATIC Manager 界面。双击右侧组态框中的 MPI(1)，在打开的窗口中可以看到 MPI 网络图（如图 7-38 所示）。

如图 7-39 所示的界面中，选中 MPI 网络主干线（可以看到网络线变粗），然后单击主菜单上“Options”选项，在下拉菜单中单击“Define Global Data”。

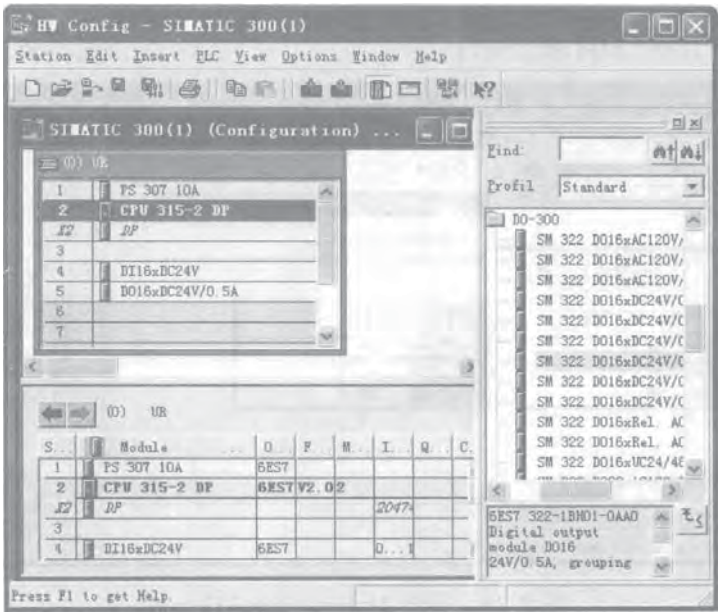


图 7-35　模块添加完成

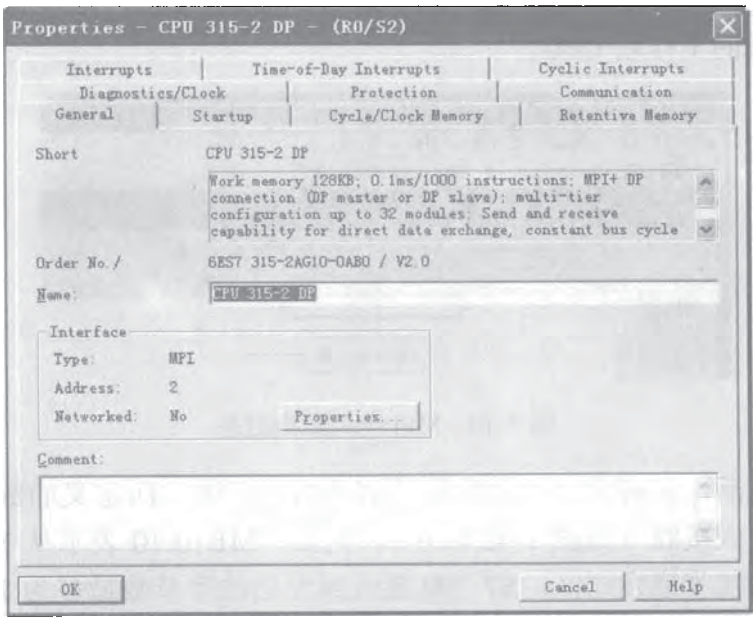


图 7-36　CPU 属性窗口

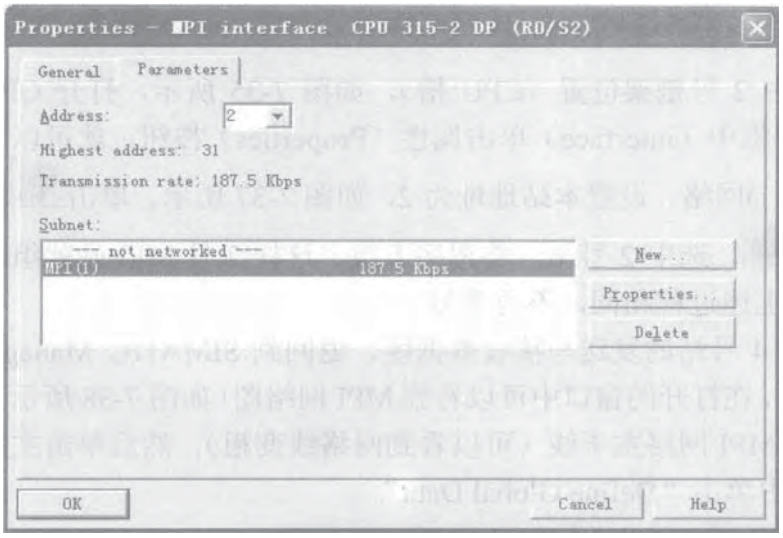


图 7-37 MPI 属性窗口

图 7-38 MPI 网络图

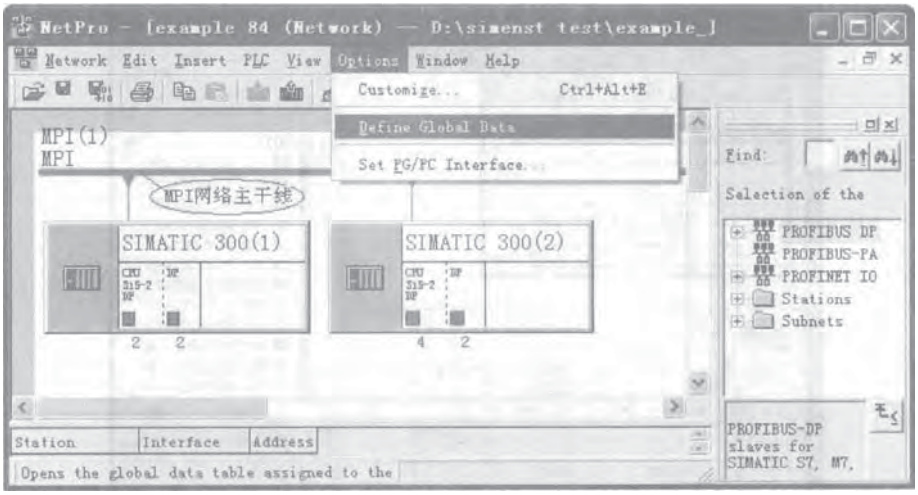


图 7-39 单击 Define Global Data

图 7-40 是 MPI 全局数据组态窗口，分别双击图示箭头位置，打开如图 7-41 所示界面，选择每个数据区所属的工作站 CPU。

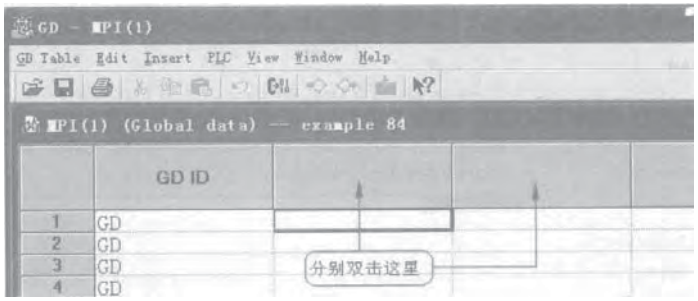


图 7-40 MPI 全局数据组态

工作站选择完成后，在每个站下面所属的表格内直接输入所定义的数据区，并通过右键的小菜单来定义发送或接收（如图 7-42 所示）。例如，MB10:10 表示从 MB10 开始共 10 字节的数据区作为全局接收或发送区。S7-300 接收或发送的字节数最多为 22 个，S7-400 最多是 54 个。发送区与相应的接收区的大小要一致。接收和发送的数据区可以为 D、B、M、I、Q。

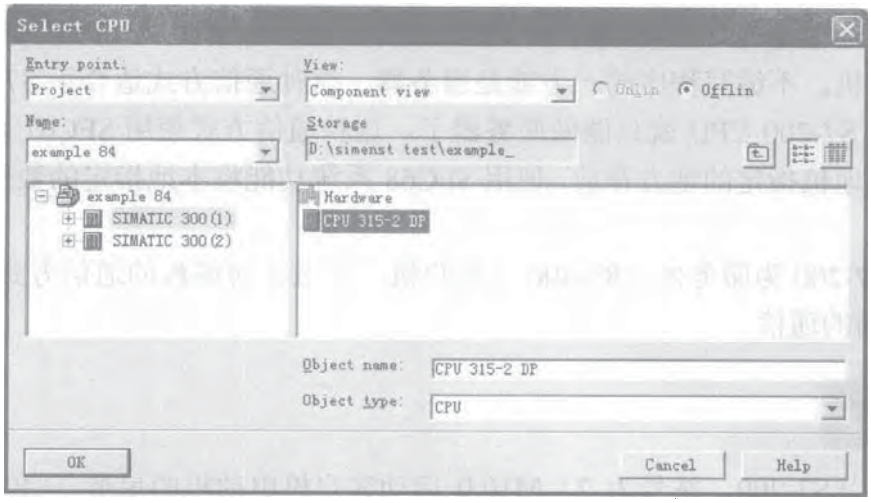


图 7-41　数据区 CPU 选择界面

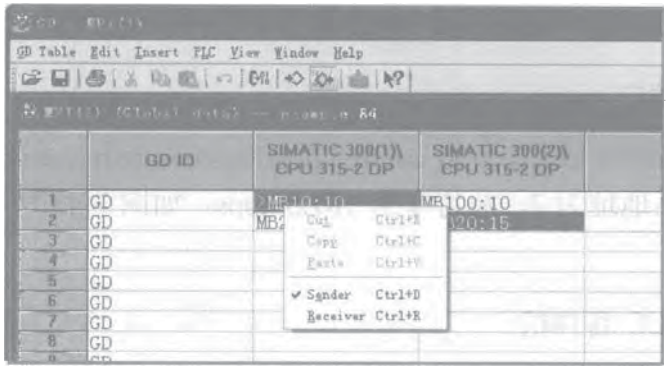


图 7-42　数据区 CPU 选择界面

GD-MPI(1)界面中的主菜单按钮，对全局数据区进行编译。编译完成后单击该界面下的图标，将组态好的数据区内容下载到对应的 CPU 中（最好选用单独下载的方式）。这样就建立了全局的 MPI 网络。

实例分析

在组态完成后进入使用时，每个站的 CPU 在各自发送区内写入信息，则其他站相应的接收区就可以得到这些信息。

实例 85：无组态连接通信方式

实例说明

无组态连接的 MPI 通信适合 S7-300、S7-400、S7-200 之间的通信，通过调用 SFC65、SFC66、SFC67、SFC68、SFC69 来实现。同时，无组态连接通信方式不能与全局数据包通信方式混合使用。无组态连接的 MPI 通信又分为：双边编程通信方式与单边编程通信方式。

双边编程通信方式，即本地站与远程站双方都要编写通信程序，发送方使用 SFC65 发送数据，接收方使用 SFC66 接收数据。这些系统功能只有 S7-300/400 才有，因此双边编程的通信方式只适用于 S7-300/400 之间的通信，不能与 S7-200 通信。

单边编程的通信方式只在一方编写程序，这就像客户机与服务器的访问模式，编写程序的一方就是客户机，不编写程序的一方就是服务器，这种通信方式适合于 S7-200/300/400 之间的通信，对于 S7-200 CPU 就只能做服务器了。这种通信方式使用 SFC67 来读取对方指定的地址数据到本地机指定的地方存放，使用 SFC68 系统功能将本地指定的数据发送到对方指定的数据区。

本实例以 S7-200 为服务器，S7-400 为客户机，通过单边编程的通信方式实现 S7-200 与 S7-400 CPU 之间的通信。

实例实现

实现服务器（S7-200，站号为 2）M10.0 启动客户机电动机的星形-三角形启动，M10.1 停止客户机电动机转动；反过来客户机 M10.2 启动服务器电动机的星形-三角形启动，M10.3 停止服务器电动机转动。

1．设置服务器的端口

打开 STEP7-Micro/WIN 软件，单击 communications/communication Ports 打开通信口设置界面，设置 PORT0 的站地址为 2，波特率为 187.5 kbps，如图 7-43 所示，把设置好的参数下载到 CPU 中。

2．组态 S7-400 PLC 客户机

打开 STEP 7 Manager，建立项目，插入 S7-400 站并组态硬件。硬件组态和 MPI 通信口参数设置方法与实例 84 相同。连接好网络之后，在 STEP 7 Manager 界面下，单击主菜单中的 PLC/Upload Station to PG，即可把 S7-200 站读取出来。在 S7-400 OB1 中编写梯形图程序如图 7-44 所示。

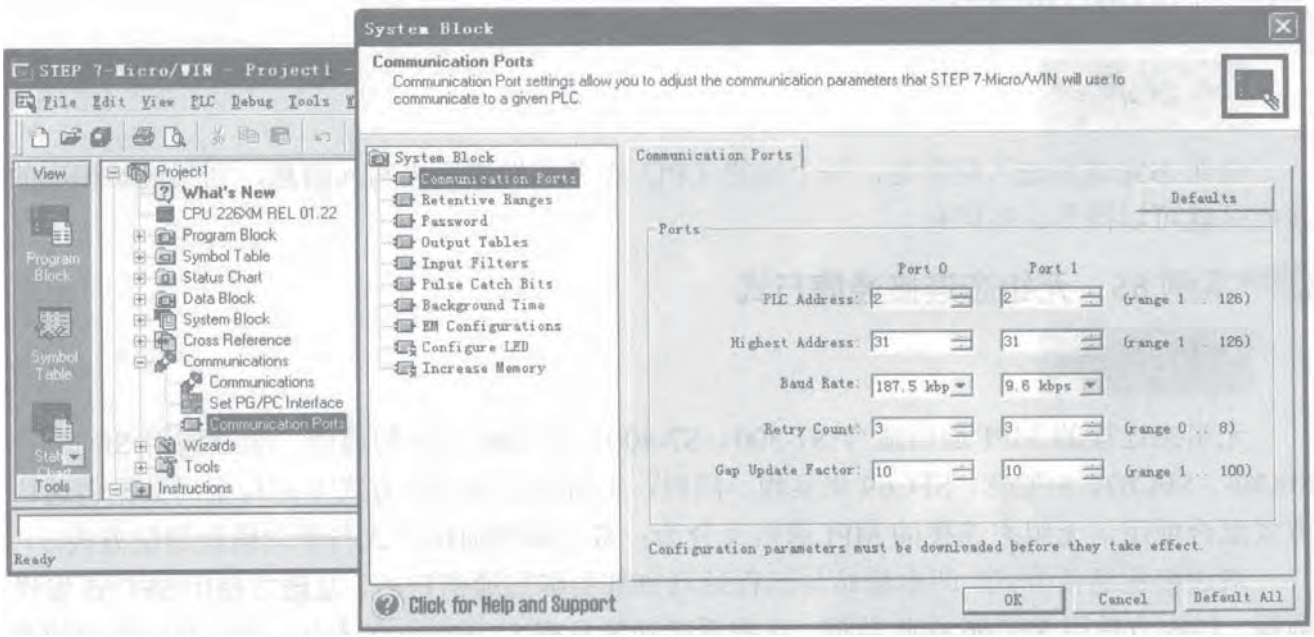


图 7-43　服务器通信端口设置

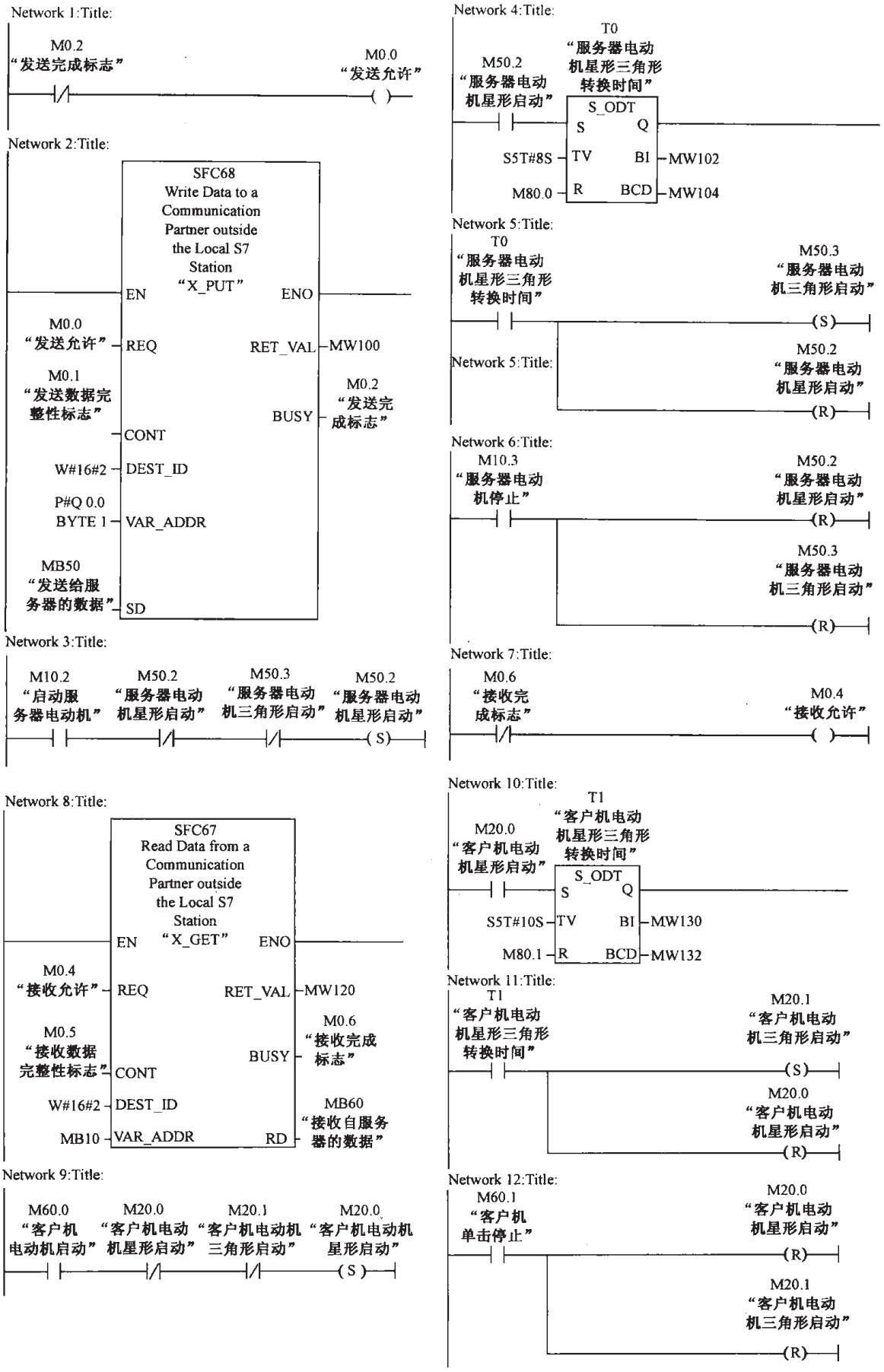

Network 1:Title:
M0.2
"发送完成标志"
M0.0
"发送允许"
Network 2:Title:
SFC68
Write Data to a Communication Partner outside the Local S7 Station
"X_PUT"
EN
ENO
M0.0
"发送允许"
REQ
RET_VAL
MW100
M0.1
"发送数据完整性标志"
BUSY
M0.2
"发送完成标志"
CONT
W#16#2
DEST_ID
P#Q 0.0
BYTE 1
VAR_ADDR
MB50
"发送给服务器的数据"
SD
Network 3:Title:
M10.2
"启动服务器电动机"
M50.2
"服务器电动机星形启动"
M50.3
"服务器电动机三角形启动"
M50.2
"服务器电动机星形启动"
(S)
Network 4:Title:
T0
"服务器电动机星形三角形转换时间"
M50.2
"服务器电动机星形启动"
S_ODT
S
Q
S5T#8S
TV
BI
MW102
M80.0
R
BCD
MW104
Network 5:Title:
T0
"服务器电动机星形三角形转换时间"
M50.3
"服务器电动机三角形启动"
(S)
Network 5:Title:
M50.2
"服务器电动机星形启动"
(R)
Network 6:Title:
M10.3
"服务器电动机停止"
M50.2
"服务器电动机星形启动"
(R)
M50.3
"服务器电动机三角形启动"
(R)
Network 7:Title:
M0.6
"接收完成标志"
M0.4
"接收允许"
Network 8:Title:
SFC67
Read Data from a Communication Partner outside the Local S7 Station
"X_GET"
EN
ENO
M0.4
"接收允许"
REQ
RET_VAL
MW120
M0.5
"接收数据完整性标志"
BUSY
M0.6
"接收完成标志"
CONT
W#16#2
DEST_ID
MB10
VAR_ADDR
RD
MB60
"接收自服务器的数据"
Network 9:Title:
M60.0
"客户机电动机启动"
M20.0
"客户机电动机星形启动"
M20.1
"客户机电动机三角形启动"
M20.0
"客户机电动机星形启动"
(S)
Network 10:Title:
T1
"客户机电动机星形三角形转换时间"
M20.0
"客户机电动机星形启动"
S_ODT
S
Q
S5T#10S
TV
BI
MW130
M80.1
R
BCD
MW132
Network 11:Title:
T1
"客户机电动机星形三角形转换时间"
M20.1
"客户机电动机三角形启动"
(S)
M20.0
"客户机电动机星形启动"
(R)
Network 12:Title:
M60.1
"客户机单击停止"
M20.0
"客户机电动机星形启动"
(R)
M20.1
"客户机电动机三角形启动"
(R)

图 7-44 梯形图程序

图 7-44 的语句表如下：

```
AN    "发送完成标志"                          M0.2
=     "发送允许"                              M0.0
A     "发送允许"                              M0.0
=     L     20.0
BLD   103
A     "发送数据完整性标志"                    M0.1
=     L     20.1
BLD   103
CALL  "X_PUT"                                 SFC68
REQ   :=L20.0
CONT  :=L20.1
DEST_ID :=W#16#2
VAR_ADDR:=P#Q 0.0 BYTE 1
SD    :="发送给服务器的数据"                  MB50
RET_VAL :=MW100
BUSY  :="发送完成标志"                        M0.2
NOP   0
A     "启动服务器电动机"                      M10.2
AN    "服务器电动机星形启动"                  M50.2
AN    "服务器电动机三角形启动"                M50.3
S     "服务器电动机星形启动"                  M50.2
A     "服务器电动机星形启动"                  M50.2
L     S5T#8S
SD    "服务器电动机星形三角形转换时间"        T0
A     M     80.0
R     "服务器电动机星形三角形转换时间"        T0
L     "服务器电动机星形三角形转换时间"        T0
T     MW    102
LC    "服务器电动机星形三角形转换时间"        T0
T     MW    104
NOP   0
A     "服务器电动机星形三角形转换时间"        T0
S     "服务器电动机三角形启动"                M50.3
R     "服务器电动机星形启动"                  M50.2
A     "服务器电动机停止"                      M10.3
R     "服务器电动机星形启动"                  M50.2
R     "服务器电动机三角形启动"                M50.3
AN    "接收完成标志"                          M0.6
=     "接收允许"                              M0.4
A     "接收允许"                              M0.4
=     L     20.0
BLD   103
```

```
A       "接收数据完整性标志"                    M0.5
=       L       20.1
BLD     103
CALL    "X_GET"                                 SFC67
 REQ    :=L20.0
 CONT   :=L20.1
 DEST_ID :=W#16#2
 VAR_ADDR:=MB10
 RET_VAL :=MW120
 BUSY   :="接收完成标志"                        M0.6
 RD     :="接收自服务器的数据"                  MB60
NOP     0
A       "客户机电动机启动"                      M60.0
AN      "客户机电动机星形启动"                  M20.0
AN      "客户机电动机三角形启动"                M20.1
S       "客户机电动机星形启动"                  M20.0
A       "客户机电动机星形启动"                  M20.0
L       S5T#10S
SD      "客户机电动机星三角形转换时间"          T1
A       M       80.1
R       "客户机电动机星形三角形转换时间"        T1
L       "客户机电动机星三角形转换时间"          T1
T       MW      130
LC      "客户机电动机星三角形转换时间"          T1
T       MW      132
NOP     0
A       "客户机电动机星形三角形转换时间"        T1
S       "客户机电动机三角形启动"                M20.1
R       "客户机电动机星形启动                   M20.0
A       "客户机单击停止"                        M60.1
R       "客户机电动机星形启动"                  M20.0
R       "客户机电动机三角形启动                 M20.1
```

7.5　PROFIBUS-DP 通信实例

S7-200 必须通过 EM277 模块才能连接到 PROFIBUS-DP 总线上。PROFIBUS-DP 总线的主站是 S7-300 或 S7-400。而 EM277 是以第三方设备的形式出现在 PROFIBUS-DP 网络中的。支持 PROFIBUS-DP 协议的第三方设备都会有 GSD 文件，通常以*.GSD 或*.GSE 文件名出现。组态时将此文件加入就可以设置第三方设备的通信接口了。EM277 的 GSD 文件为“siem089d.gsd”。下面的例子说明了 S7-200 PLC 通过 EM277 与 PROFIBUS-DP 的连接。

实例 86：以 EM277 为接口的 S7-200 与 Profibus-DP 的连接

实例说明

本实例通过硬件模块 EM277 实现 S7-200 PLC 与现场总线协议 Profibus-DP 的连接。

实例实现

Profibus-DP 通过 EM277 与 S7-200 连接的网络配置图如图 7-45 所示。

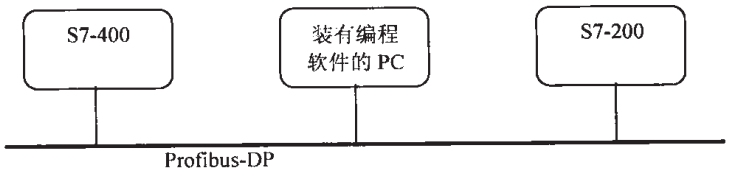


图 7-45　网络配置图

1．Profibus-DP 网络组态

在 STEP 7 下新建项目并组态 S7-400 主站（步骤同实例 83）。在配置 CPU 时，需新建网络。在如图 7-46 所示界面上单击“New...”按钮，出现如图 7-47 所示的窗口。

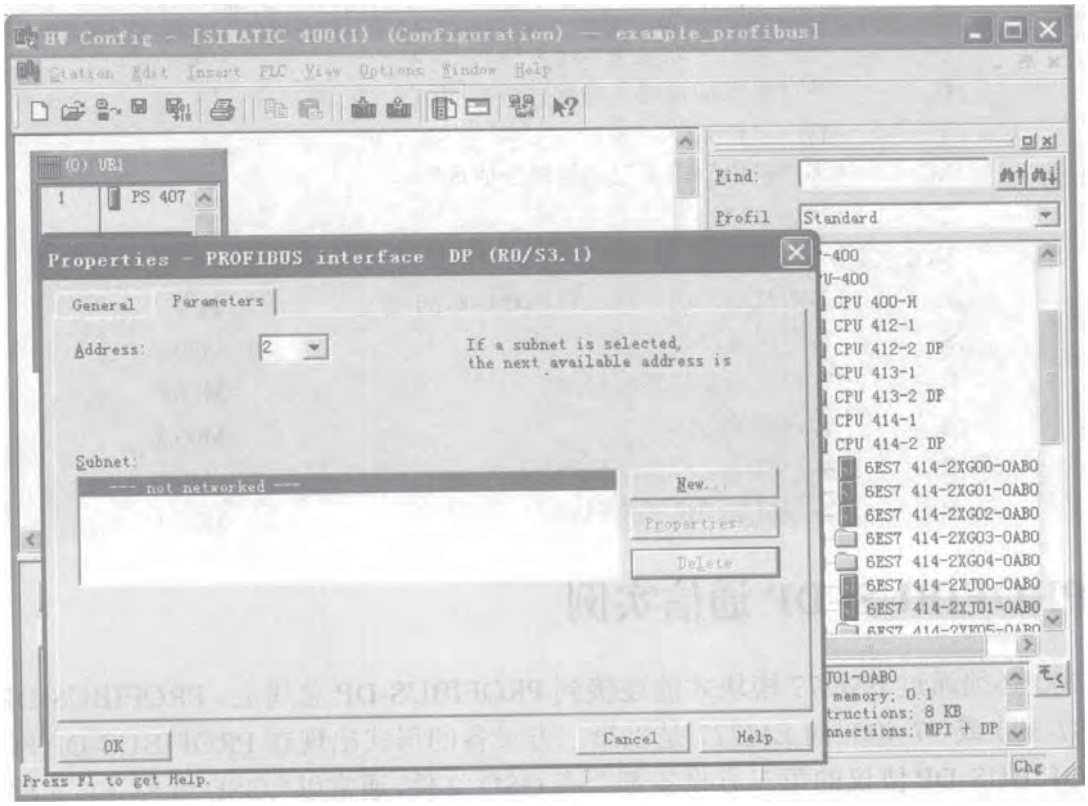


图 7-46　CPU 网络组态

单击左上方“Network Setting”选项，在新出现的窗口（如图 7-48 所示）上选择网络类型与传输速率。设置完毕后，单击“OK”按钮，返回如图 7-49 所示的界面。此时可以编译存盘。

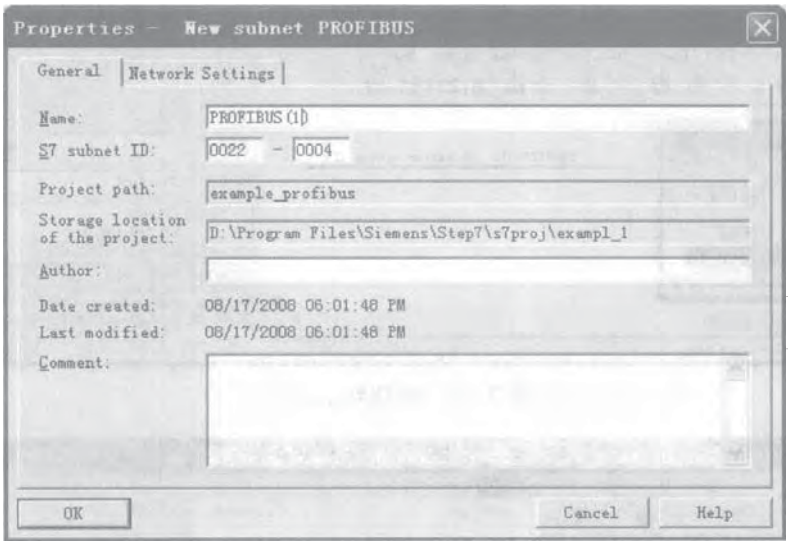


图 7-47　新建网络

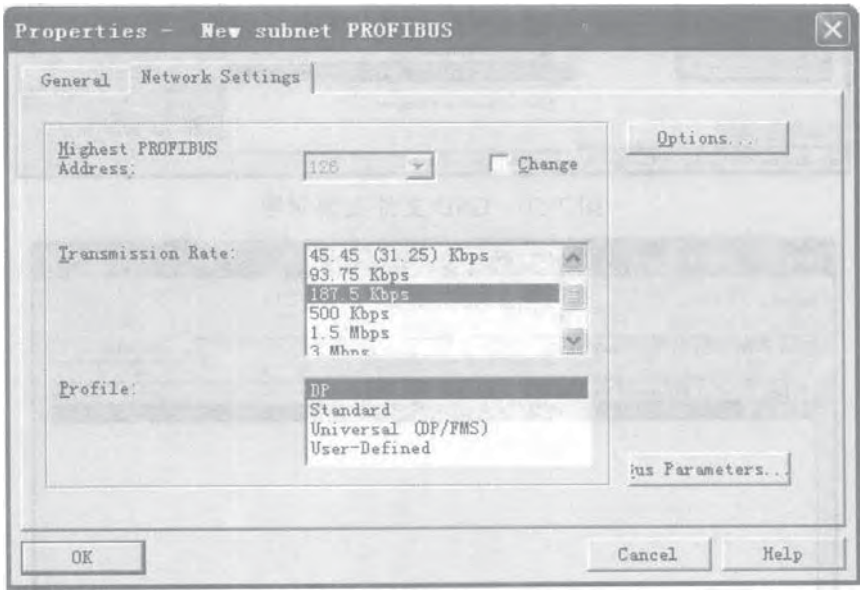


图 7-48　设置网络参数

2. 将 S7-200 PLC 总线通信模块 EM277 组态到网络中

将 S7-200 PLC 总线通信模块 EM277 组态到网络中，是通过安装“GSD”文件实现的。在硬件组态界面上单击主菜单“Options”选项（如图 7-50 所示），在下拉菜单中单击“Install GSD File”选项，打开文件安装界面（如图 7-51 所示），由“Browse”键找到安装目录，然后选中找到的文件，单击“install”进行安装。安装完成后又回到图 7-49 所示界面，用鼠标选中 Profibus-DP 干线，单击右键打开快捷菜单，选中“Insert Object”，在依次出现的选择框中分别选择“Additional Field Devices”，“PLC”，“EM 277 Profibus-DP”，如图 7-52 所示。在随后出现的 EM 277 Profibus-DP 属性设置界面中，选择地址为 3（这个站号与 EM277 的拨码开关设定地址要一致）。完成后，EM277 就挂在了 Profibus-DP 干线上（如图 7-53 所示）。

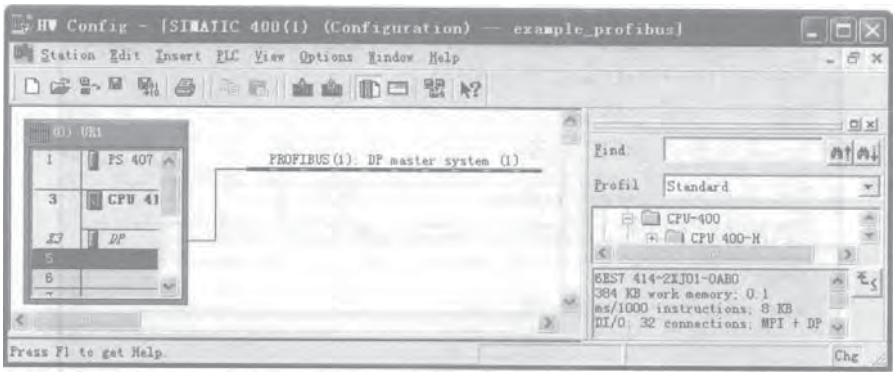


图 7-49　网络组态完成

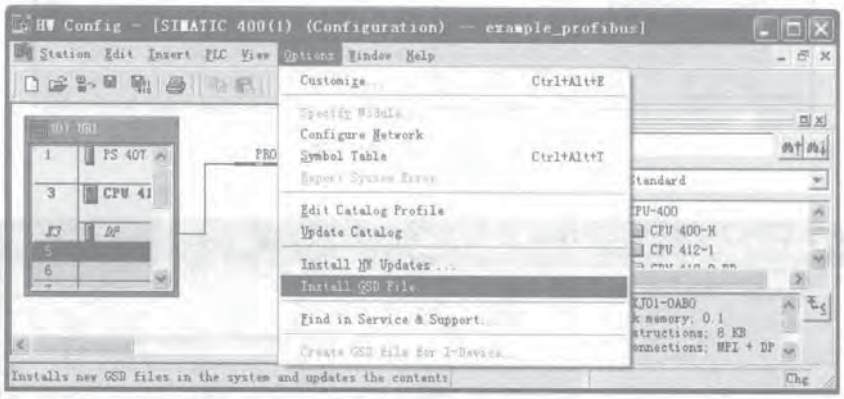


图 7-50　GSD 文件安装向导

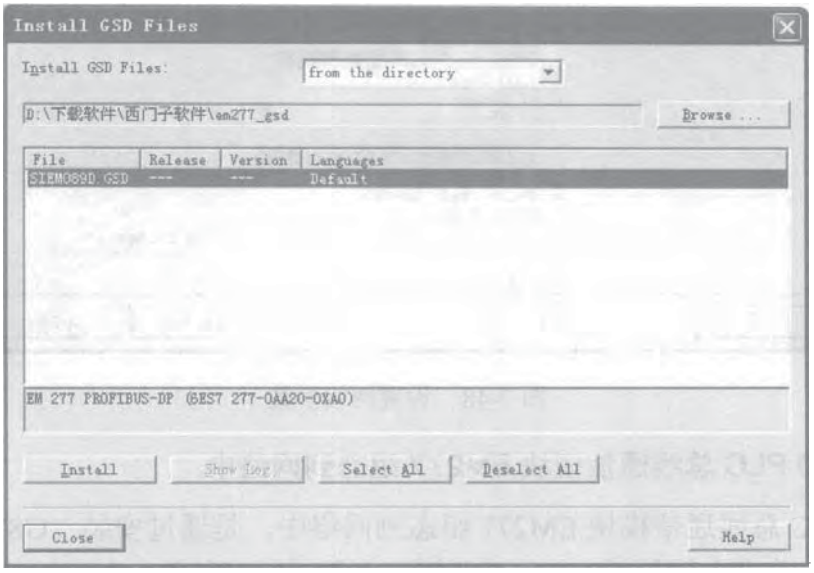


图 7-51　安装 GSD 文件

3．定义通信接口的大小

在硬件配置窗口的右侧，逐级选择“PROFIBUS-DP/Additional Field Devices/PLC/SIMATIC/ EM 277 Profibus-DP”，在下面出现的菜单中可以定义通信接口的大小。例如，选择接口大小为 2 字节输入/2 字节输出（2 Bytes Out/2 Bytes in），则 S7-400 对应的输入为 IB0 和 IB1 字节，输出为 QB0 和 QB1 字节。S7-200 对应的是 V 区的 4 个字节，其中前面两个字节为接收，后面两个字节为发送。（双击 EM 277 图形，打开 DP Slave 属性窗口，可以设置 S7-200 的 V 区偏移量，如为 100，则接收区的地址为 VB100 和 VB101，发送区的地址为 VB102

和 VB103）。数据之间的传递无需程序的干涉。写入相应字节的数据在对应站的字节中可直接得到。

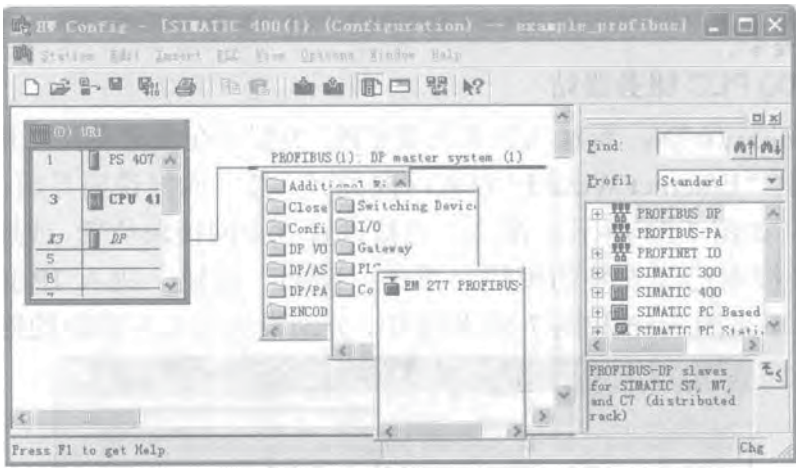


图 7-52　添加 EM277

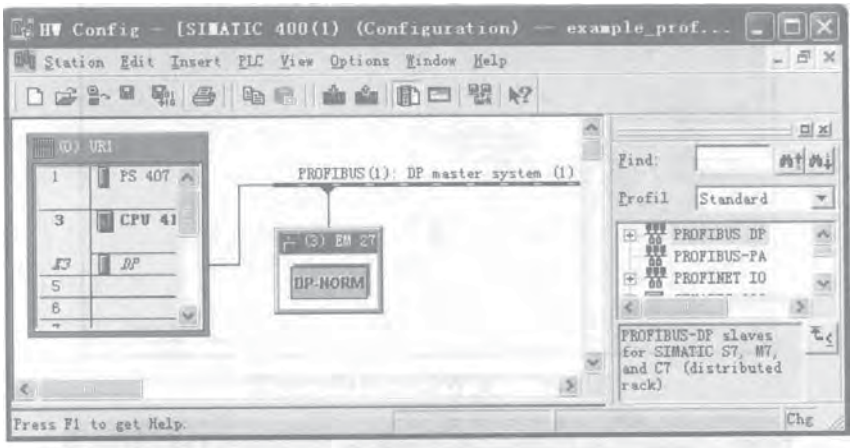


图 7-53　添加 EM277 的结果

实例分析

Profibus-DP 通信的实现是通过组态完成的，而 S7-200 PLC 必须通过 EM277 模块才能与 Profibus-DP 网络连接。

7.6　工业以太网通信实例

通过以太网模块（CP243-1 IT S7-200 以太网模块，CP343-1 以太网模块，CP443-1 IT S7-400 以太网模块）和 TP 电缆，就可以把 PLC 连接成以太网通信网络。下面分别以 S7-200 PLC 做服务器与做客户机为例，对以太网通信的设置与实现进行说明。

实例 87：S7-200 为服务器、S7-400 为客户机的以太网通信

实例说明

本实例以 S7-200 为服务器、S7-400 为客户机构建以太网通信系统。通过本实例，了解以太网通信的组态过程。

实例实现

S7-200 作为服务器的以太网通信，客户机采用 S7-400 PLC。编程时，计算机通过 PC/PPI 电缆分别与 PLC 连接。

1．组态 S7-200 PLC 服务器站

打开 STEP 7 Micro/WIN，新建项目并设置 CPU 类型。在主菜单上选取“Tool（工具）”，打开下拉菜单，选择“Ethernet Wizard（以太网向导）”，打开向导说明界面，直接单击“Next”按钮进入组态界面(如图 7-54 所示)，在这里直接设置以太网模块位置，或通过“Read Modules（读取模块）”直接搜索已经安装的模块。单击“Next”按钮，进入 IP 地址设置界面，如图 7-55 所示。在接下来图 7-56 和图 7-57 界面中，分别设置以太网模块的连接数和连接配置。

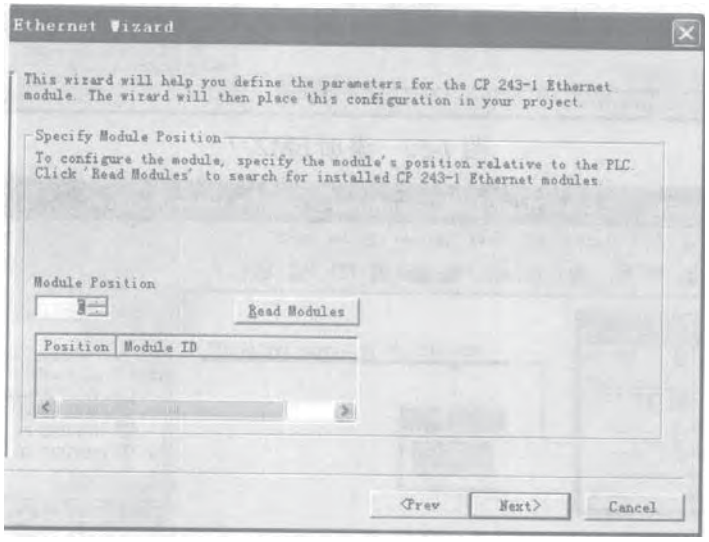


图 7-54　配置以太网模块位置

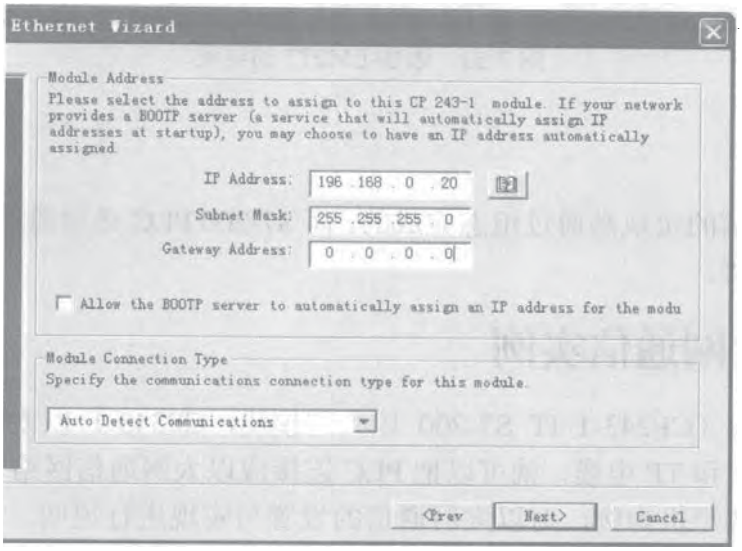


图 7-55　IP 地址设置

在连接配置界面中，首先要将连接选项选择为：“This is a Server Connection（此为服务器连接）”。然后设置远程属性 TASP，勾选“Accept all connection requests.（接受所有的连接请求）”和“Enable the Keep Alive function for this connection.（为此连接启用‘保持活动’功能）”。其中，TSAP 的含义如下：

TASP 由两个字节组成。第一个字节定义了连接数，对 Local TSAP（服务器）定义范围为：16#02、16#10～16#FE；对 Remote TSAP（客户机）定义范围为：16#02、16#03、16#10～16#FE。第二个字节定义了机架上的 CP 槽号，本例中服务器是 16#00，在 00 位置上，客户机是 16#03，在 03 号位置上。

单击“OK”按钮确认后，在接下来的界面中选择“Yes, generate CRC protection for this configuration in the data block.（是，为数据块中的此配置生成 CRC 保护）”。CRC 的含义是 Cycle Redundancy Check，即循环冗余码校验。

接下来配置以太网指令占用的系统 V 区资源，这些资源用户在编程时不能占用。如图 7-58 所示。最终以太网向导所自动生成的子程序与占用资源信息如图 7-59 所示。

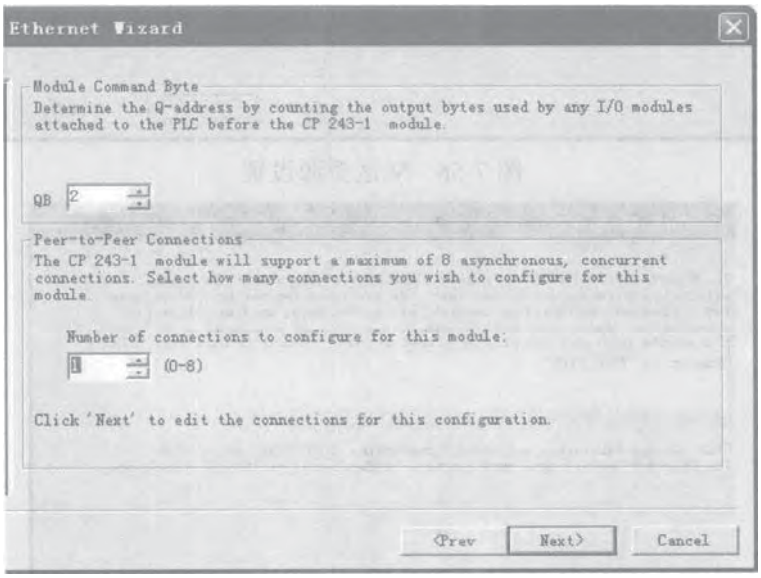


图 7-56　连接数设置

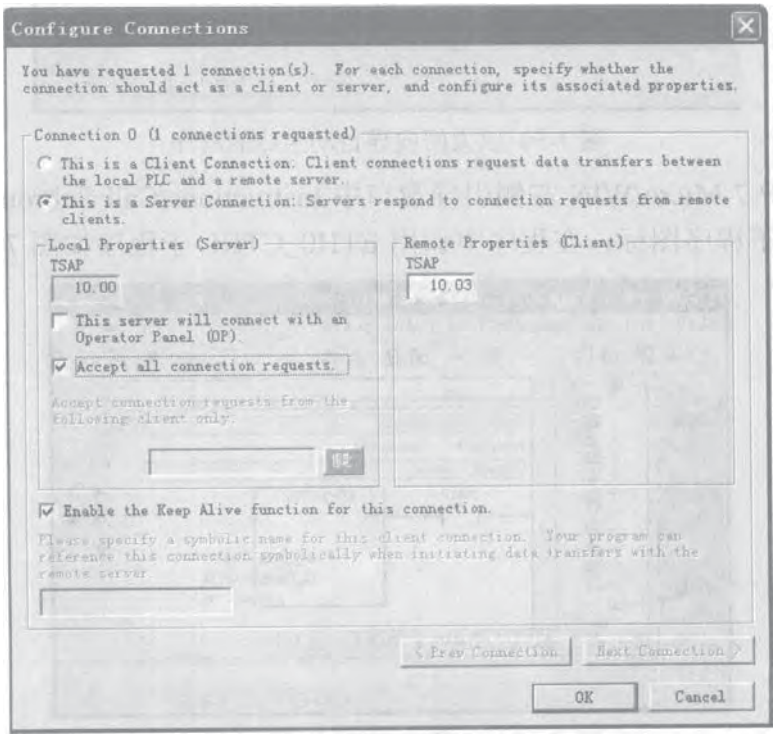


图 7-57　连接配置设置

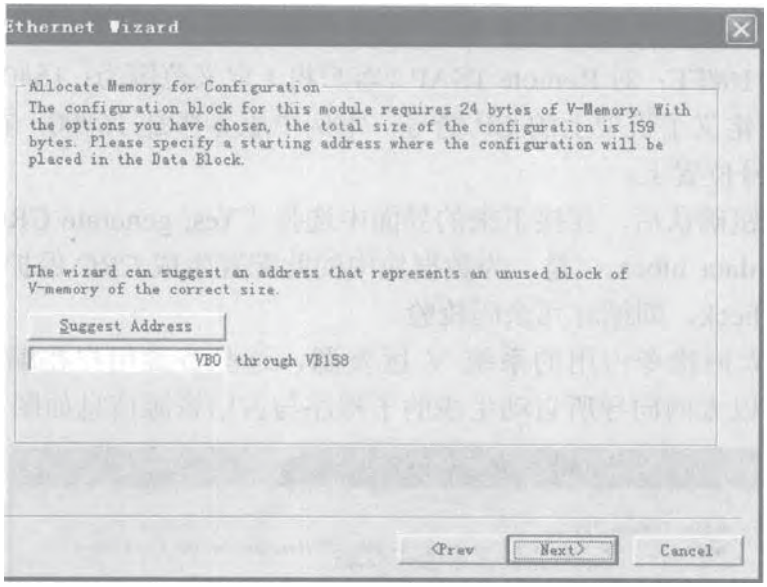


图 7-58 V 区资源设置

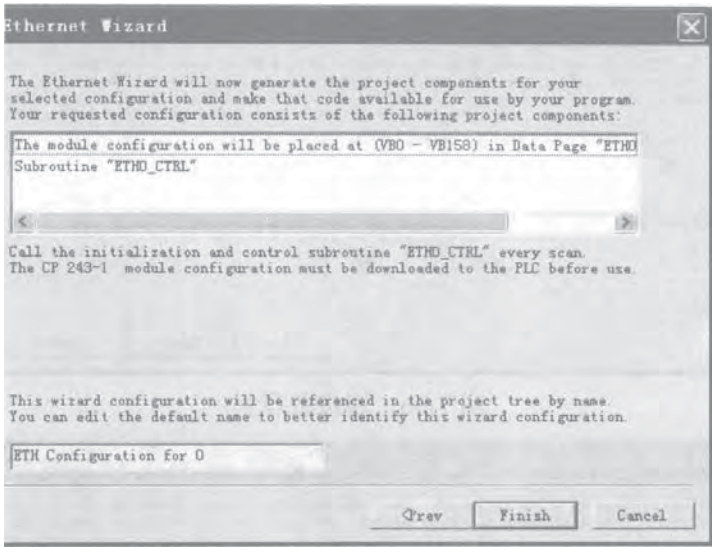


图 7-59 以太网向导自动生成的程序

完成后在 STEP 7-Micro/WIN 左侧引导窗口中 Instructions/Call Subroutines 菜单下就可以看到 ETH0_CTRL 子程序图标。在程序中调用 ETH0_CTRL 子程序如图 7-60 所示。

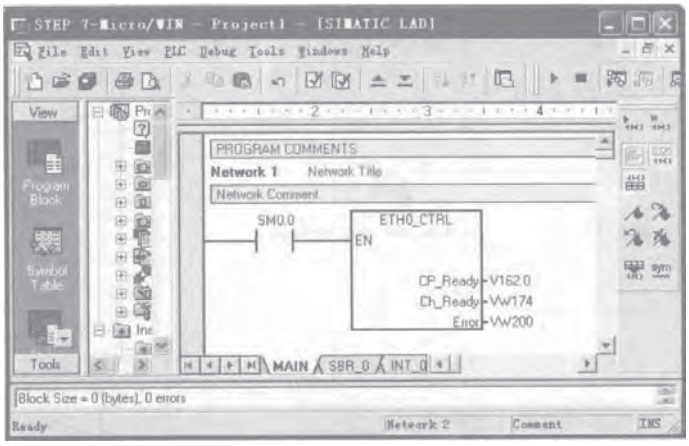


图 7-60 调用 ETH0_CTRL 子程序

其中 ETH0_CTRL 返回参数的含义如表 7-7 所示。

表 7-7　ETH0_CTRL 返回参数的含义

名　称	类　型	含　义
CP-Ready	BOOL	CP 243_1 IT 的状态 0：CP 没有准备就绪 1：CP 准备就绪
CH-Ready	WORD	每个通道或 IT 服务的状态： （第一个字节） 位 0 对应于通道 0；位 1 对应于通道 1；　位 2 对应于通道 2 位；3 对应于通道 3； 位 4 对应于通道 4；位 5 对应于通道 5；位 6 对应于通道 6；位 7 对应于通道 7； （第二个字节） 位 0 对应于 e-mail 服务；位 1 对应于 FTP 客户机服务； 位 2 对应于 FTP 服务器服务；位 3 对应于 HTTP 服务器服务；位 4～7：保留 0：通道或服务没有准备就绪；1：通道或服务准备就绪
错误	WORD	出错或报文代码 0*0000：没有错误 其他：错误代码 出错或报文代码最大可持续 60s

2．组态客户机 S7-400 PLC 工作站

打开 STEP 7 的 SIMATIC Manger 窗口，新建一个项目，然后插入并组态 S7-400 站。在硬件组态完成槽架、电源和 CPU 模块之后，插入 Ethernet 模块 CP443-1 IT，如图 7-61 所示。设置 IP 地址和子网掩码后，单击“NEW”按钮打开如图 7-62 所示窗口，单击“OK”按钮关闭窗口，然后编译保存已建立的以太网通信。

回到 SIMATIC Manger 窗口（如图 7-63 所示），双击“Ethernet(1)”图标，打开 Netpro 界面（如图 7-64 所示）。可以看到 Ethernet 网络干线图。用鼠标选中 S7-400 槽架中的 CPU 414-2DP，然后在下方表格的 Local ID 处单击鼠标右键，在出现的下拉菜单中选择“Insert New Connection”，出现如图 7-65 所示画面。选择“Unspecified”和“S7 connection”后，单击“Apply”按钮。

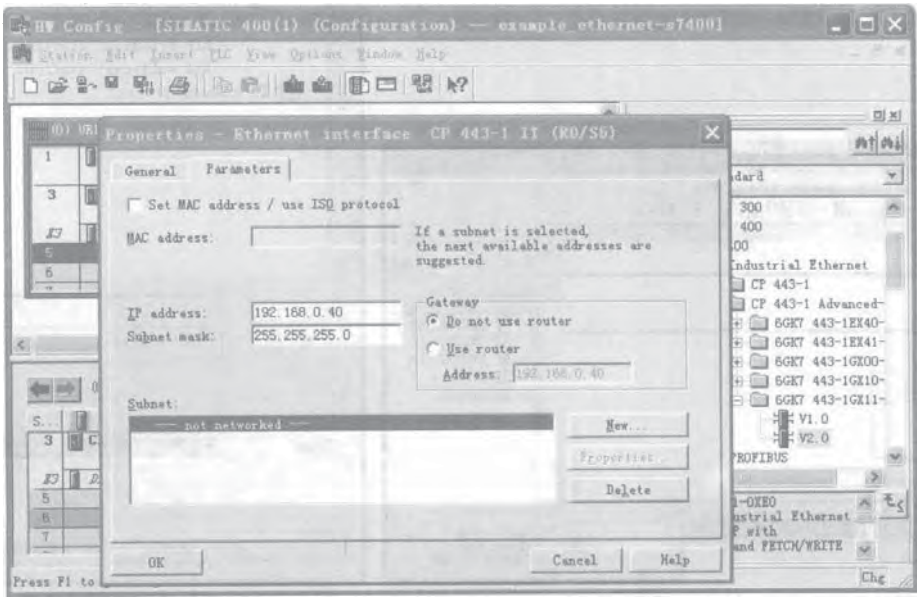


图 7-61　插入以太网模块

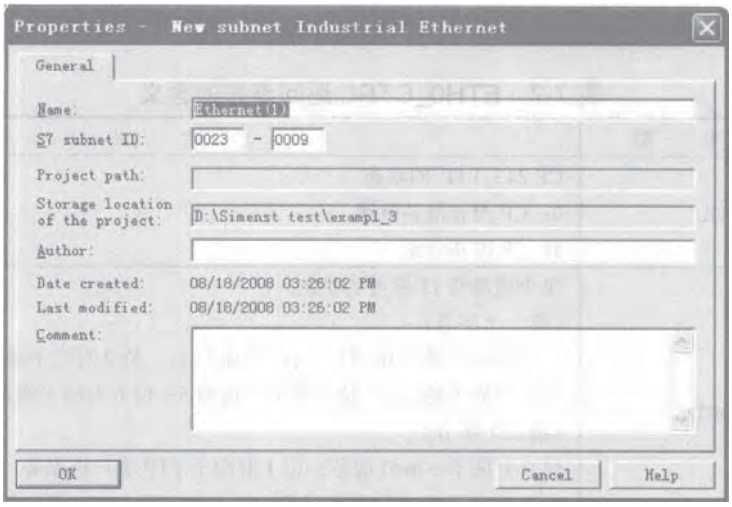


图 7-62 新建以太网连接

在图 7-66 中，在 Local ID 里写上“1”号连接，在 Partner（伙伴）地址里写上服务器的 IP 地址。然后单击“Address Details”，出现如图 7-67 所示的画面，设定 TSAP。根据网络的设定写出本地（Local）与伙伴（Partner）的 TSAP，然后单击“OK”按钮，编译保存。

图 7-63 添加了 Ethernet 的 SIMATIC Manger 界面

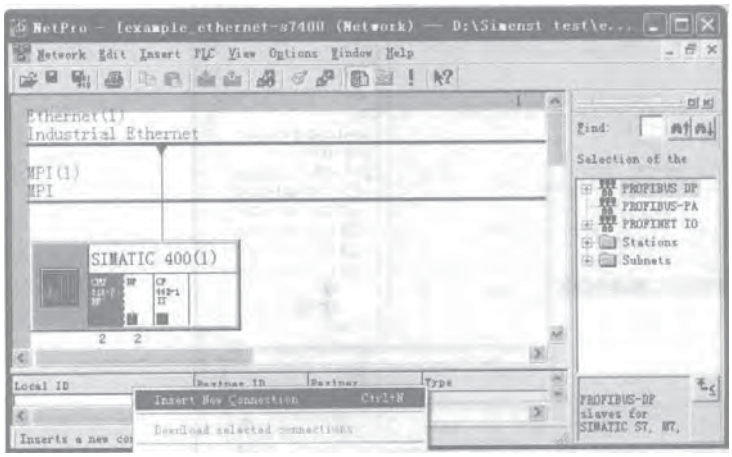


图 7-64 插入以太网连接

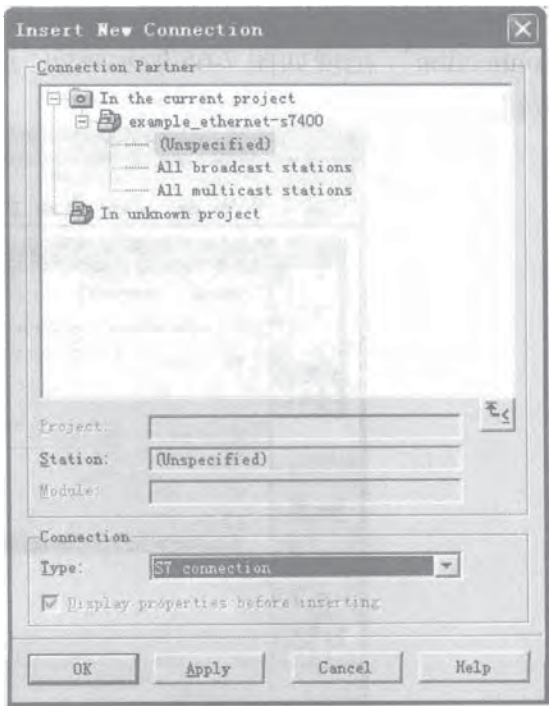


图 7-65 建立新连接

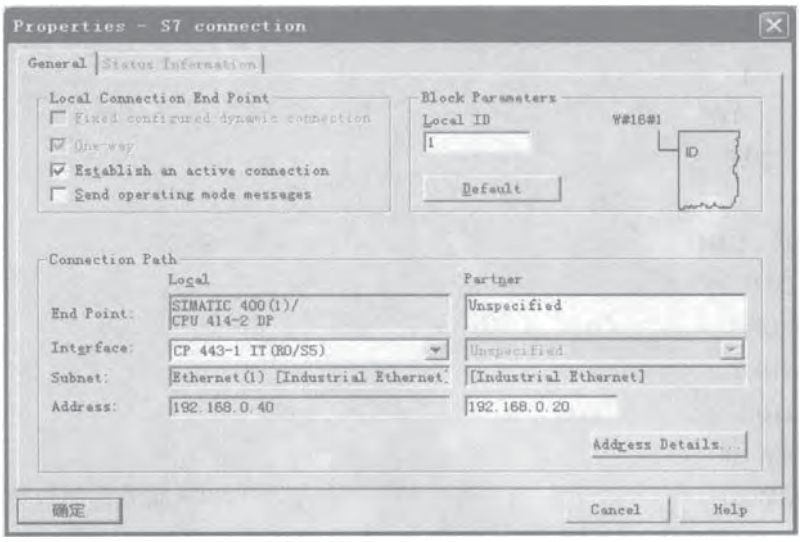


图 7-66　设定连接参数

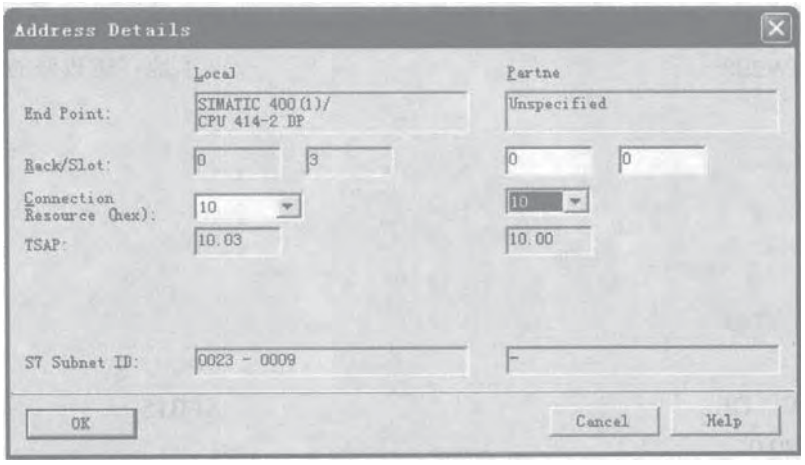


图 7-67　设定 TSAP

至此，S7-200 与 S7-400 的以太网连接组态相继完成，分别把组态下载到各自的 CPU，则 CP443-1 IT 与 CP243-1 IT 可建立起正常通信了。

虽然建立了通信连接，但网络上具体要发送或接收什么信息还没有表达出来，还要在客户机里编写程序把需要交换的信息编写出来，服务器不需要编写程序。

3. 编写客户机通信程序

在客户机 OBL 里调用 SFB14，实现的功能是把服务器的数据读到本地机来；调用 SFB15，实现的功能是把本地机的数据发送到服务器。梯形图程序如图 7-68 所示。

图 7-68 语句为：

```
AN    T     1
L     S5T#1S
SD    T     2
A     T     2
FP    M     0.1
=     M     0.0
A     T     2
```

```
L      S5T#1S
SD     T      1
A      M      0.0
=      L      20.0
BLD    103
CALL   "GET" , DB1                          SFB14
 REQ    :=L20.0
 ID     :=W#16#1
 NDR    :=M10.0
 ERROR :=M10.1
 STATUS:=MW100
 ADDR_1:=MW20                               //从服务器读取数据的地址
 ADDR_2:=
 ADDR_3:=
 ADDR_4:=
 RD_1   :=MW200                             //本地存储数据的地址
 RD_2   :=
 RD_3   :=
 RD_4   :=
NOP    0
A      T      1
=      L      20.0
BLD    103
CALL   "PUT" , DB2                          SFB15
 REQ    :=L20.0
 ID     :=W#16#1
 DONE   :=M10.3
 ERROR :=M10.4
 STATUS:=MW104
 ADDR_1:=MW30                               // 要将数据写入的服务器地址
 ADDR_2:=
 ADDR_3:=
 ADDR_4:=MW300                              //提供数据的本地地址
 SD_1   :=
 SD_2   :=
 SD_3   :=
 SD_4   :=
NOP    0
```

实例分析

服务器与客户机的以太网通信是分别组态完成的。通信的信息交换程序仅在客户机中编写。

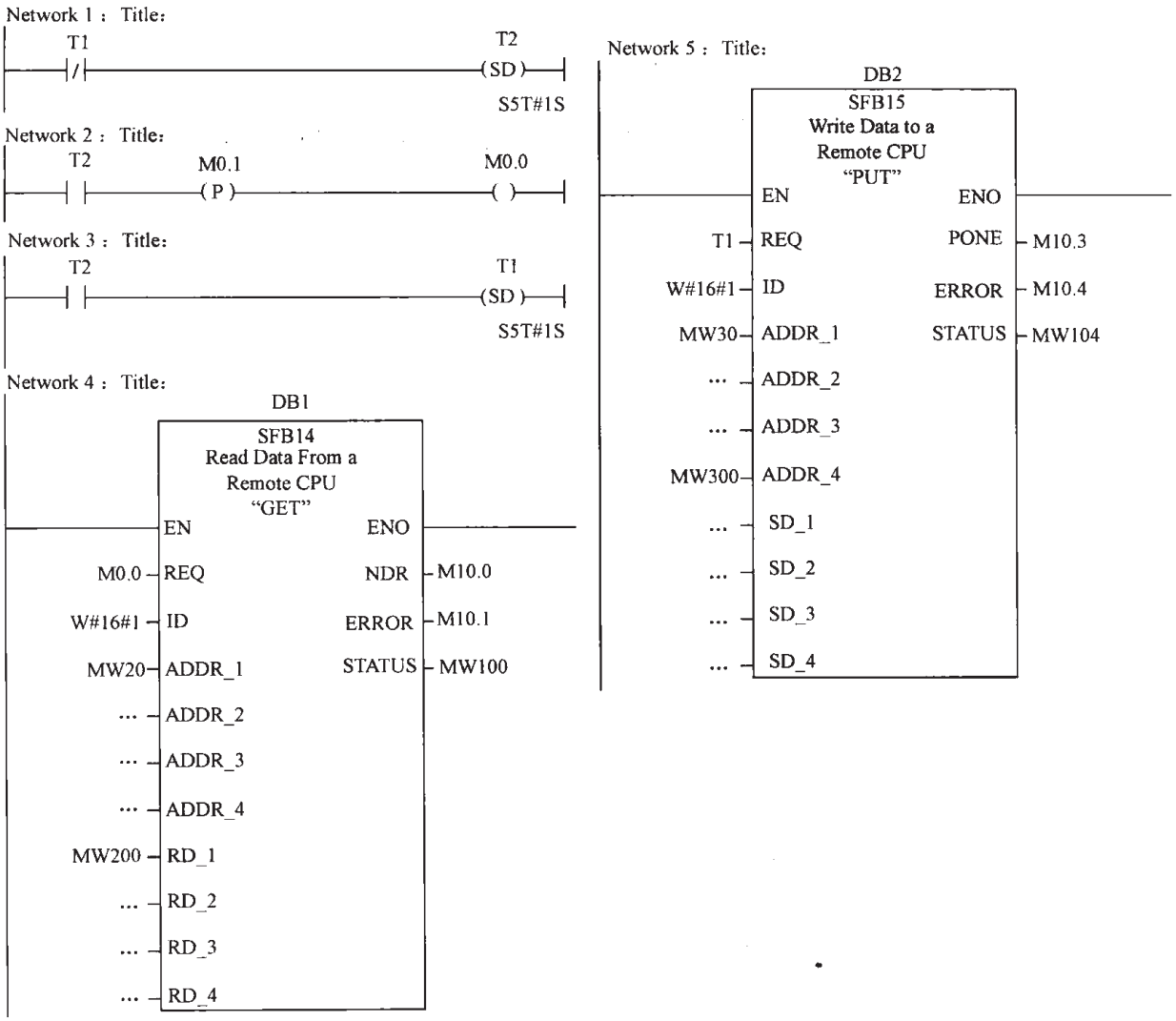


图 7-68　梯形图程序

实例 88：S7-200 为客户机、S7-400 为服务器的以太网通信

实例说明

本实例以 S7-200 为客户机、S7-400 为服务器构建以太网通信系统。

实例实现

1．S7-200 以太网卡的配置

S7-200 以太网卡配置的前面几步与例 87 相同。从配置连接开始（如图 7-57 所示），设置的步骤按照客户机进行，配置连接界面的变化如图 7-69 所示。在设置完服务器的 TSAP 后，单击“Data Transfers（数据传输）”按钮。在新出现的画面（如图 7-70 所示）中，单击“New Transfer”按钮，然后在弹出的对话框中选择“是”，打开数据传输配置界面。首先设置接收的数据（如图 7-71 所示），然后通过图 7-71 界面中的“New Transfer”按钮，再打开数据传输界面，设置发送的数据（如图 7-72 所示）。单击“OK”按钮关闭传输数据设置界面，在接下来的界面中选择“Yes, generate CRC protection for this configuration in the data block.”。在

图 7-73 所示界面中来配置以太网指令占用的系统 V 区资源。最后以太网向导自动生成的子程序与相关信息如图 7-74 所示。单击“Finish”按钮完成配置。

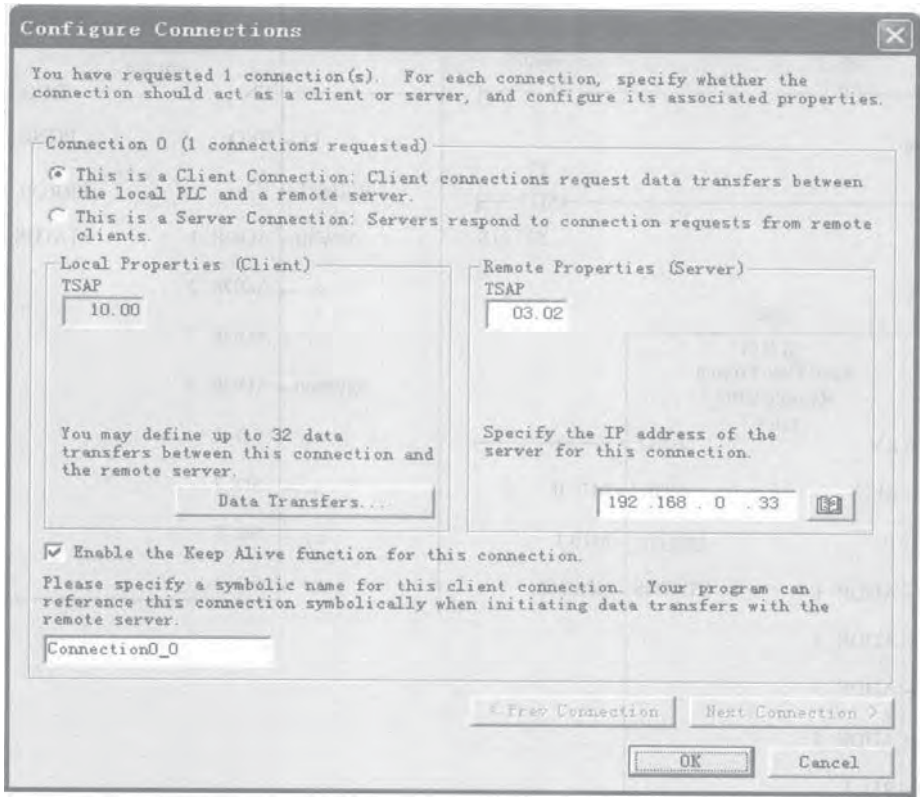


图 7-69 客户机设置

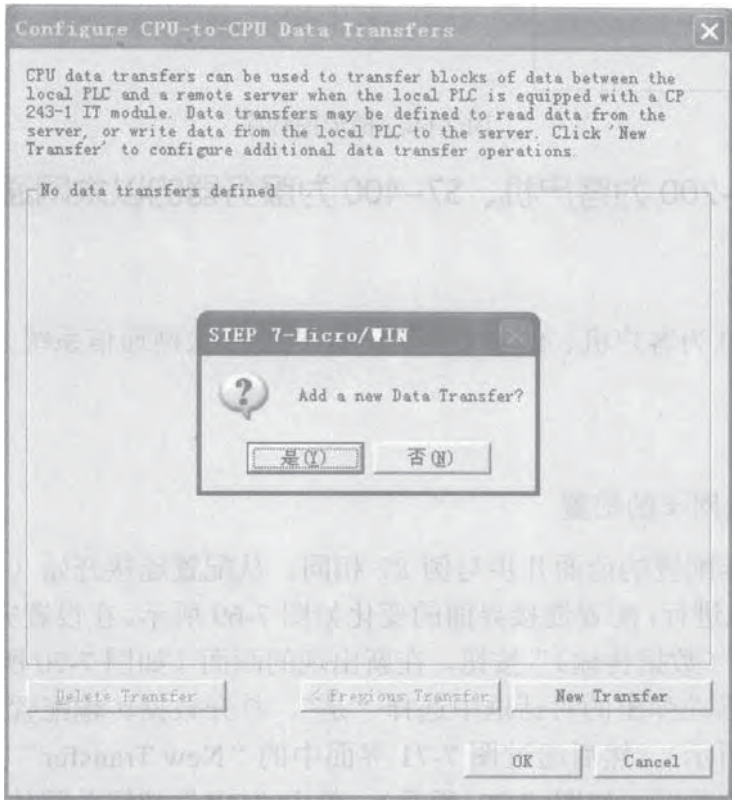


图 7-70 添加数据连接

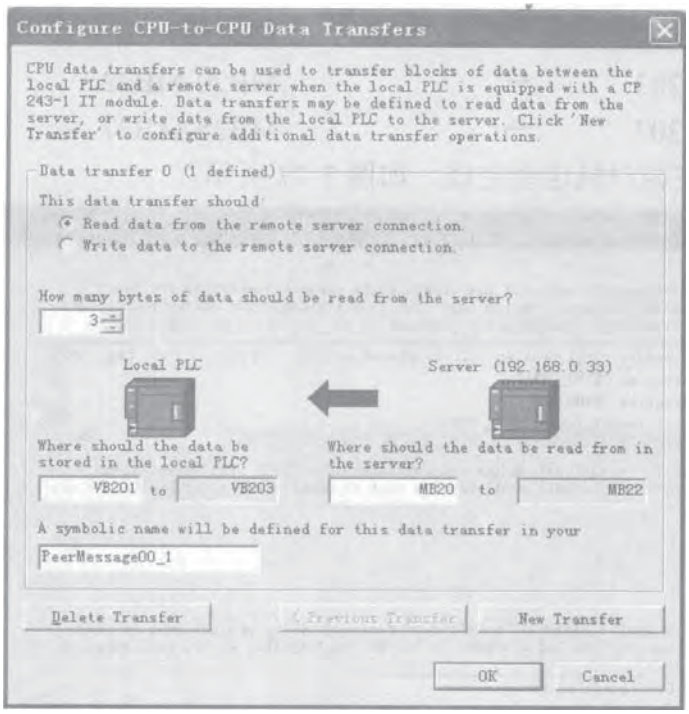


图 7-71　建立读连接

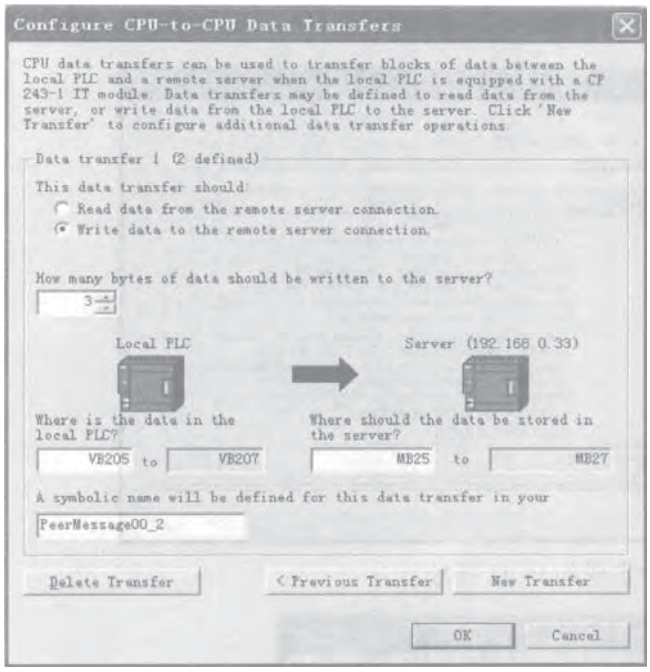


图 7-72　建立写连接

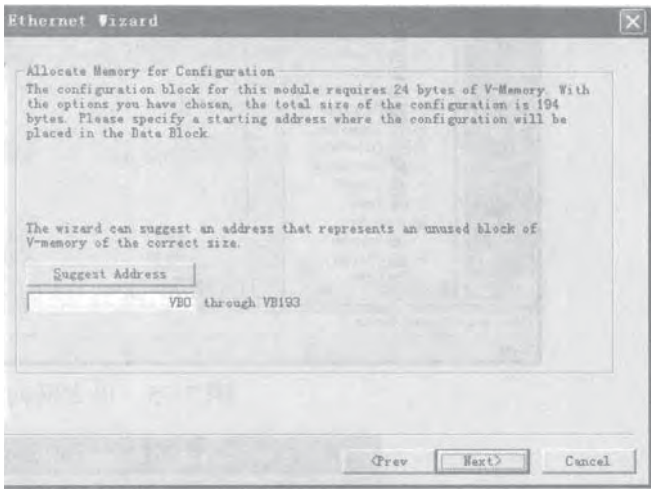


图 7-73　向导占用的 V 区

在 STEP 7-Micro/WIN 主界面下，单击最左侧“View”栏中“Symbol Table”，再单击界面下方的“POU Symbols”标签，出现如图 7-75 所示的画面，对以太网设置向导生成子程序的符号、地址和使用进行了说明。通过 ETH0_SYM 标签出现如图 7-76 所示的窗口，规定了 SBR2 的 Chan_ID 字节（VB164），以及在接收（PeerMessage00_1 数据传输）和发送（PeerMessage00_2 数据传输）信息时操作数 Data 的地址分别是 VB165 和 VB166。

客户机与服务器之间发送与接收数据区的对应关系为：

客户机		服务器
VB201-VB203	←	MB20-MB22
VB205-VB207	→	MB25-MB27

接收与发送程序在客户机这里完成，如图 7-77 所示。

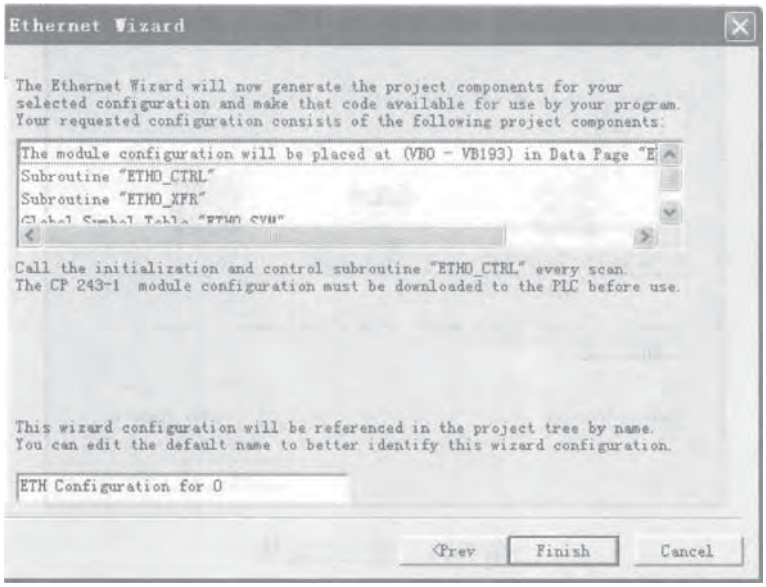


图 7-74　生成的子程序

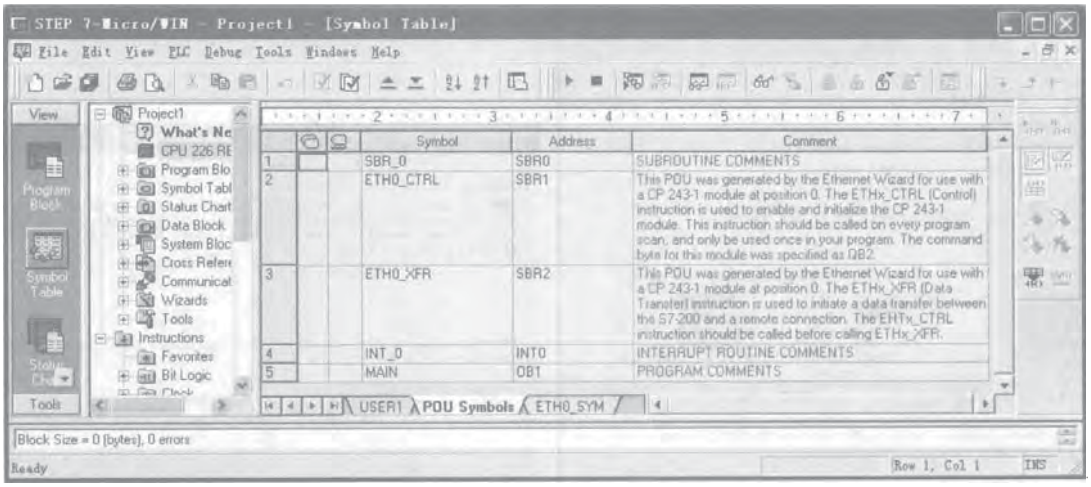


图 7-75　以太网向导生成的 POU 符号

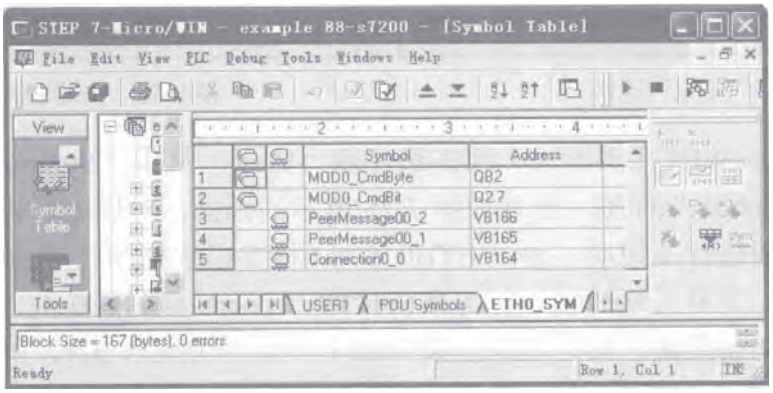


图 7-76　以太网向导生成的 ETH0_STM 符号

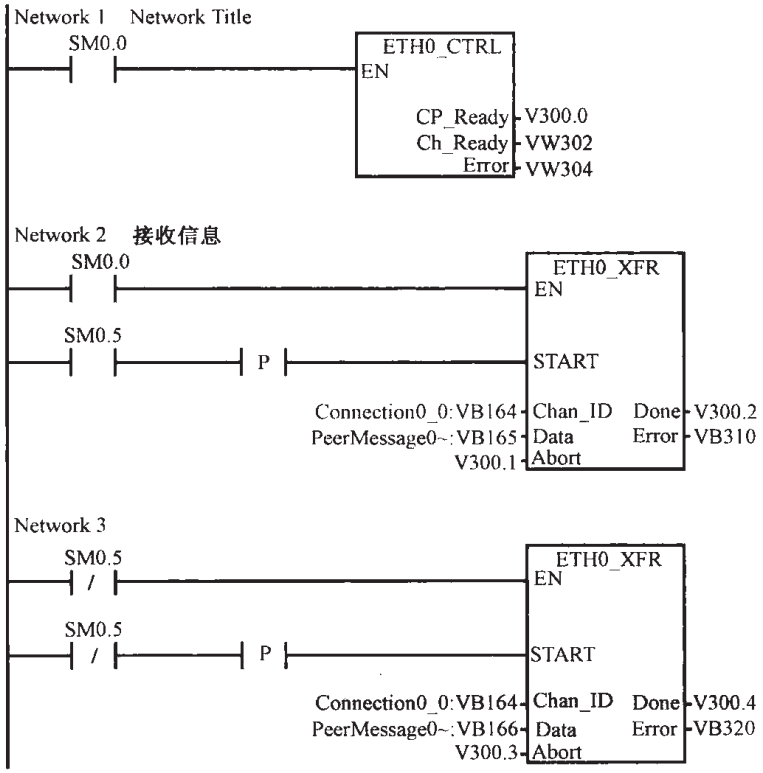


图 7-77　梯形图程序

图 7-77 的语句为：

```
LD      SM0.0
CALL    SBR1, V300.0, VW302, VW304
LD      SM0.0
=       L60.0
LD      SM0.5
EU
=       L63.7
LD      L60.0
CALL    SBR2, L63.7, VB164, VB165, V300.1, V300.2, VB310
LDN     SM0.5
=       L60.0
LDN     SM0.5
EU
=       L63.7
LD      L60.0
CALL    SBR2, L63.7, VB164, VB166, V300.3, V300.4, VB320
```

2．服务器 S7-300 PLC 的组态

服务器 S7-300 的组态是在 STEP 7 SIMATIC Manger 界面下完成的。首先进行 S7-300 的硬件组态，在完成槽架、电源、CPU 的组态之后，选择以太网通信模块 CP 343-1 IP，如图 7-78 所示。在出现的窗口中对 CP 343-1 IP 的以太网通信参数进行组态（如图 7-79 所示），并通过按钮“New”添加以太网网络（如图 7-80 所示）。单击“OK”按钮完成组态。服务器

方不需要编写程序，只需要把数据放入相应的存储区就可以了（本例中发送数据存储在 MB20-MB22，接收数据存储在 MB25-MB27）。

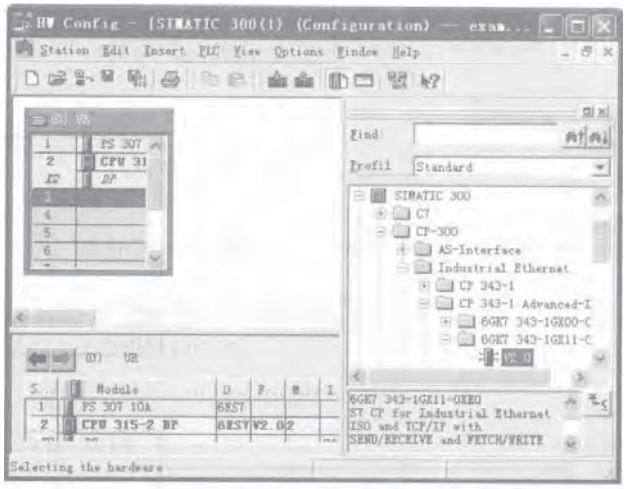


图 7-78　选择以太网通信模块 CP 343-1 IP

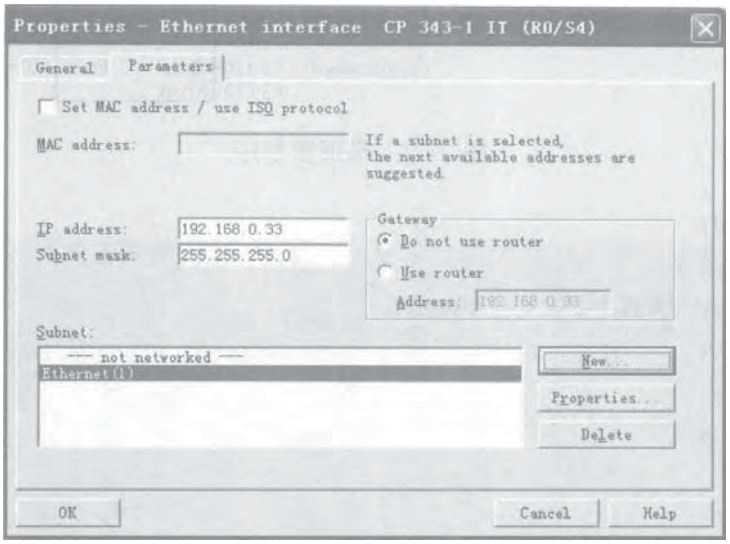


图 7-79　组态以太网通信参数

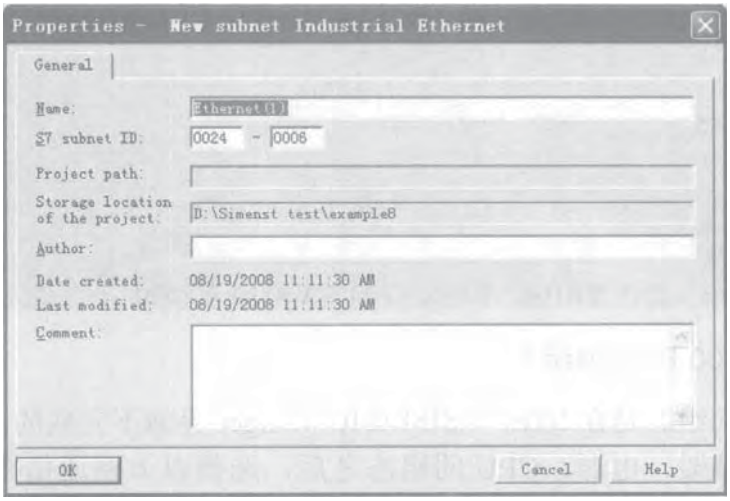


图 7-80　添加以太网网络

7.7　自由口通信实例

自由口通信为计算机或其他具有串行通信接口的设备与 S7-200 之间实现通信提供了一种廉价和灵活的方法。通过使用接收中断、发送中断、字符中断、发送指令（XMT）和接收指令（RCV），自由端口通信可以控制 S7-200 CPU 通信操作模式，即 CPU 的串行通信接口由用户程序控制。利用自由端口模式，可以实现用户定义的通信协议，连接多种智能设备。

当 CPU 处于 STOP 模式时，停止自由端口通信，通信口强制转换成其他协议模式（如 PPI 协议），从而保证了编程软件对可编程控制器的编程和控制功能。只有当 CPU 处于 RUN 模式时，才能使用自由端口模式。通过向控制字 SMB30（对于 PORT0）或 SMB130（PORT1）的协议位置 1，可以将通信端口设置为自由端口模式。同时也可以用 SM0.7 来控制自由端口模式的进入。当 SM0.7 为 1，且方式开关置 RUN 时，可选择自由端口模式；当 SM0.7 为 0，且方式开关置 TERM 时，应选 PC/PPI 协议模式。

下面通过几个例子来说明 S7-200 自由口通信的使用方法。

实例 89：利用 S7-200 的自由通信口收/发数据

实例说明

自由通信口可以自由通信，连接多种智能设备，当与各种智能设备连接好后，是如何实现与其他设备的数据交换的呢？也就是如何发数据，如何接收数据的呢？本实例将说明 S7-200 如何通过自由口通信实现数据的接收和发送。

实例实现

要实现自由口收发数据，仅仅只需将 PLC 的自由通信口与其他的智能设备通过 PC/PPI 电缆相连接即可。如果直接用 PC/PPI 电缆相连接后不能通信，很有可能是由于两台通信设备的数据传输方向相同，使得两者的数据接收线连在了一起，发送线也连接在一起（线 2 和线 3），在这种情况下需要在中间加上空调置解调式的适配器，使得通信设备的某一端的发送线和接收线互换，这样就可以实现两设备的通信了。

在利用自由口进行数据的收发时，会用到几个特殊的存储器，主要是 SMB30，SMB130，SMB2 和 SMB3：其中 SMB30，SMB130 是分别控制口 0 和口 1 两自由通信口的通信协议，其具体含义可以参考表 7-4；SMB2 为自由端口接收字符缓冲区，只能读取，在自由口通信方式下，SMB2 存储从口 0 或口 1 接收到的每一个字符。SMB2 和 SMB3 由 0 口和 1 口共用。当 0 口接收到字符，就会使得与该事件（中断事件 8）相连的中断程序执行，SMB2 包含 0 口接收到的字符，而 SMB3 包含该字符的校验状态。当 1 口接收到字符就使得与该事件（中断事件 25）相连的中断程序执行时，SMB2 包含 1 口接收到的字符，而 SMB3 包含该字符的校验状态。

本实例收/发数据都是从 0 口进行的，假定收/发数据的通信协议是：波特率 9600 波特，无奇偶校验，每字符 8 位。具体的收/发程序如图 7-81 所示。该程序实现的是，当 I0.1 口接通时，输出一个字符“A”，在接收到一个字符“A”后，Q0.1 接通。

图 7-81 的语句为：

```
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 ASII 字符
MOVB    16#41, VB101           //A 字符长度为 1 个字节。A=41H （十六进制）
LD      SM0.1                  // 第一次扫描（SM0.1=1）
ATCH    INT0, 8                //指定接收中断事件 8 调用中断程序 0
ENI                            // 允许中断
LD      I0.1                   // 输入 I0.1 启动发送
EU                             // 识别脉冲上升沿
XMT     VB100, 0               // 发送
// INT  0（中断程序）
LDB=    SMB2, 16#41            //字节 SMB2 中的接收字符合大写字母 A 作比较
S       Q0.1, 1                //如字符为 A，则置输出位 Q0.1 为 1
```

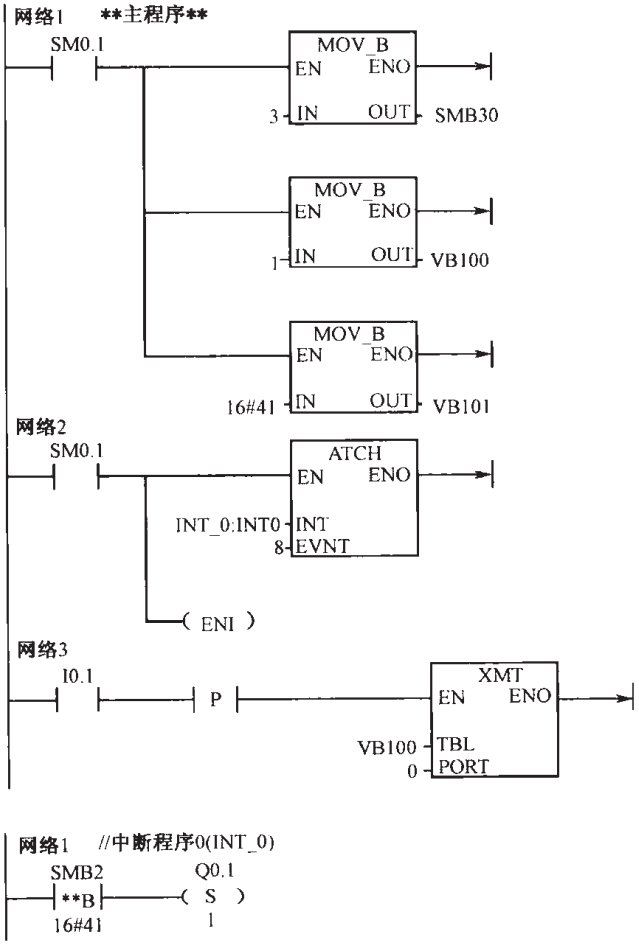


图 7-81　S7-200 自由口收/发数据程序

实例分析

在主程序中，I0.1 启动数据发送（XMT 指令）；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。

实例 90：利用 S7-200 的自由通信口发送数据

实例说明

利用 S7-200 CPU 自由通信口模式向打印机发送信息来控制打印机打印相应的文字。其主要功能是输入 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 O.x IS SET!"（其中 x 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 bps。

实例实现

S7-200 CPU 与并行口打印机的连接所需要的硬件有：

1 台 SIMATIC S7-200 CPU；

1 条 PC/PPI 电缆；

1 只 9 孔阴性插座到 25 针阳性插座的转换器；

1 台串行到并行的转换器；

1 台并行打印机。

其硬件连接如图 7-82 所示。因为 SIMATIC S7-200 CPU 和打印机都作为数据通信设备（DCE），所以两台设备的数据传输方向有可能会相同，因而需要添加空调置解调式的适配器。

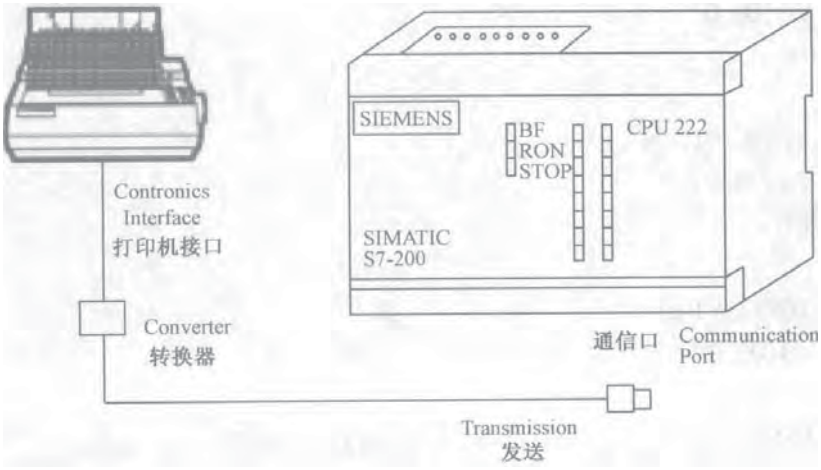


图 7-82　S7-200 与打印机的连接图

其主程序如图 7-83 所示，子程序如图 7-84 所示。

图 7-83 的语句为：

```
LD      SM0.1          // 第一次扫描标志：(SM0.1=1)
CALL    SBR0           // 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           // 启动打印输入 I0.0
EU                     // 识别脉冲上升沿
XMT     VB80, 0        // 发送 ASCII 码，并打印（VB100 中存放所发送的
                       // ASCII 码个数）
```

```
LD      I0.1                    // 启动打印输入 I0.1
EU                              // 识别脉冲上升沿
MOVB    16#31, VB109            // 把 1 的 ASCII 码#31 存入 VB109
XMT     VB100, 0                // 发送 ASCII 码，并打印（VB100 中存放所发送的
                                // ASCII 码个数）
LD      I0.2                    // 启动打印输入 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
EU
MOVB    16#33, VB109
XMT     VB100, 0
LD      I0.4
EU
MOVB    16#34, VB109
XMT     VB100, 0
LD      I0.5
EU
MOVB    16#35, VB109
XMT     VB100, 0
LD      I0.6
EU
MOVB    16#36, VB109
XMT     VB100, 0
LD      I0.7
EU
MOVB    16#37, VB109
XMT     VB100, 0
```

图 7-84 的语句为：

```
LD      SM0.0                   // 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 <SPACE>
MOVW    16#5337, VW89           // 字符 S7
MOVW    16#2D32, VW91           // 字符-2
MOVW    16#3030, VW93           // 字符 00
LD      SM0.0
MOVW    16#0D0A, VW95           //
MOVB    20, VB100               // 信息长度为 20 个 ASCII 码字符：INPUT 0.x IS SET!
```

```
MOVW    16#494E, VW101          // 字符 IN
MOVW    16#5055, VW103          // 字符 PU
MOVW    16#5420, VW105          // 字符 T  <SPACE>.
MOVW    16#302E, VW107          // 字符"0" "."
MOVB    16#20, VB110            // 有主程序装载 VB109(十六进制 31,32…,37)
MOVW    16#4953, VW111          // 字符 IS
MOVW    16#2053, VW113          // 字符 "<space>" and "S"
MOVW    16#4554, VW115          // 字符 ET
MOVW    16#2021, VW117          // 字符"<space>" and "!".
MOVW    16#0D0A, VW119          // 字符 CR and LF
```

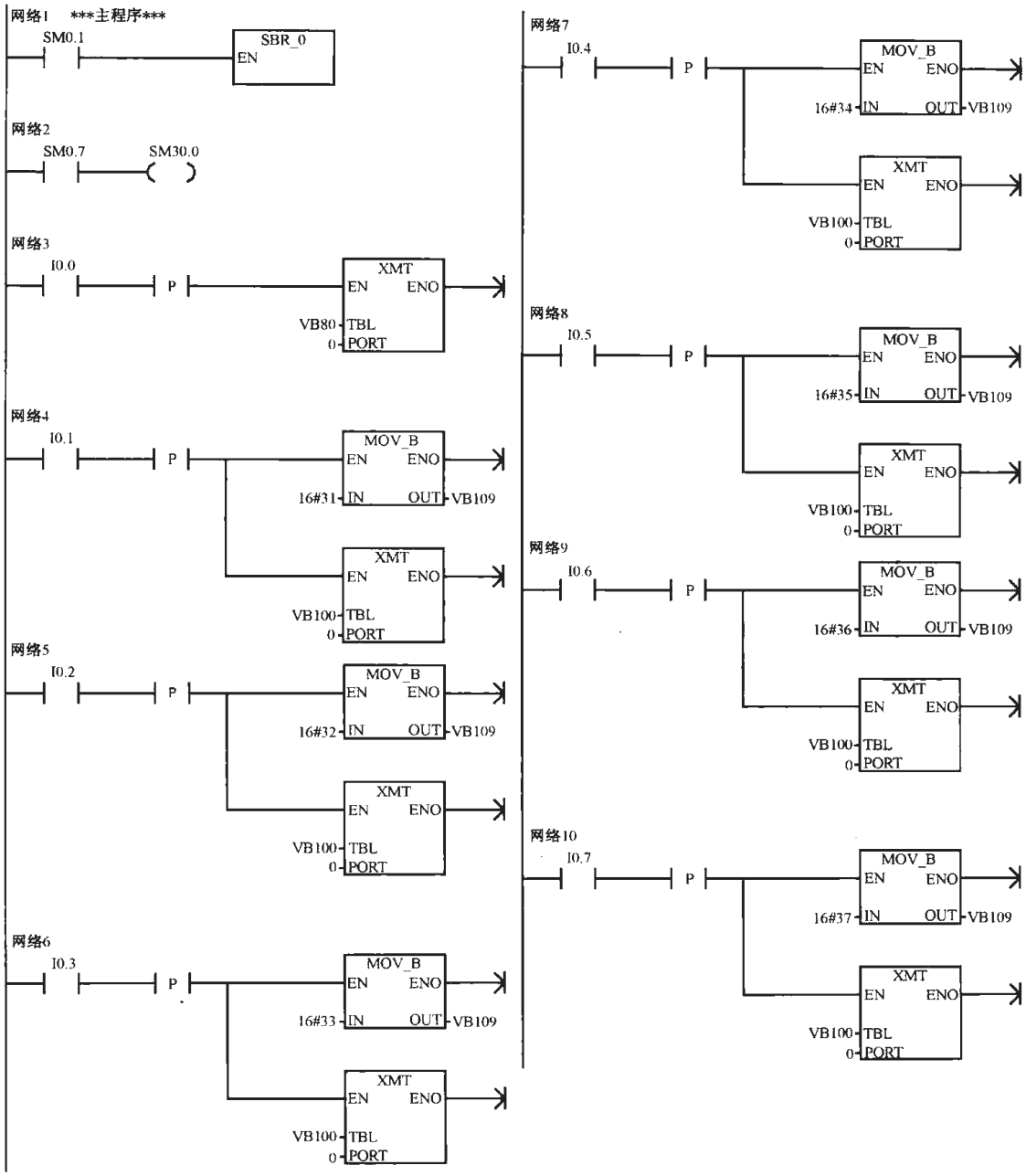


图 7-83　PLC 控制打印机主程序

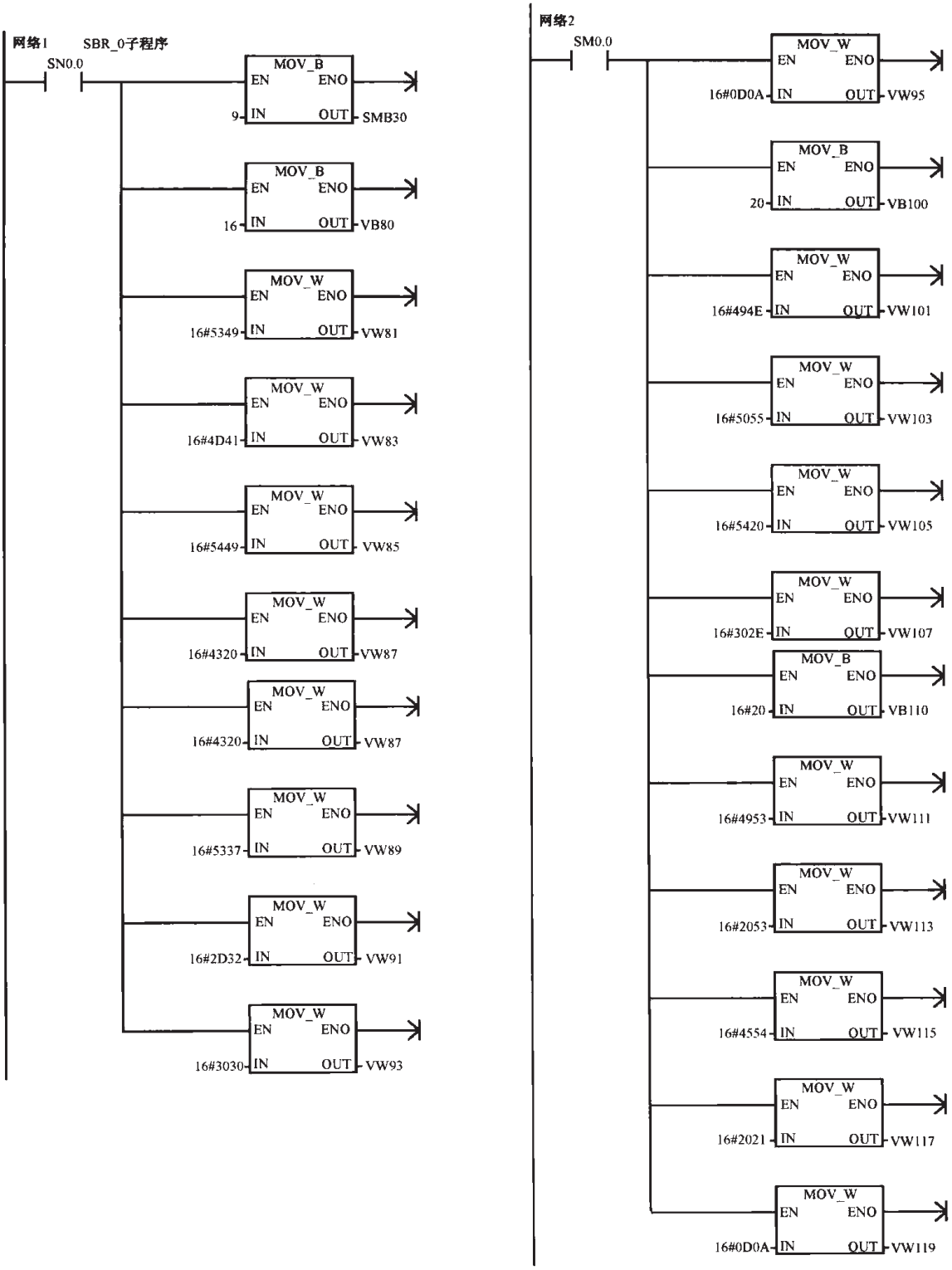


图 7-84　PLC 控制打印机子程序

实例分析

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

来完成：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 I0.7=1,则打印"INPUT 0.7 IS SET!"

实例 91：利用 S7-200 的自由通信口接收数据

实例说明

本例说明如何将 SIMATIC S7-200 与条形码阅读器配合使用。读入条形码的信息经解码器翻译为代码，在通过自由通信口模式把信息传入 SIMATIC S7-200 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代码。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。

实例实现

S7-200 CPU 与条形码阅读器的连接所需要的硬件有：

1 台 SIMATIC S7-200；

1 条 PC/PPI 电缆；

1 台合适的适配器（依据条形码解码器的接口类型，如 9 针阳性转换到 25 孔阴性的插座，线 2 和线 3 互换的空调制解调器）；

1 台条形码阅读器；

1 台条形码解码器（有时读码器与解码器是合一的）。

其硬件连接如图 7-85 所示。与实例 89 一样，SIMATIC S7-200 与条形码阅读器因为的数据传输方向也有可能会相同，因此也可能需要空调置解调式的适配器来解决。

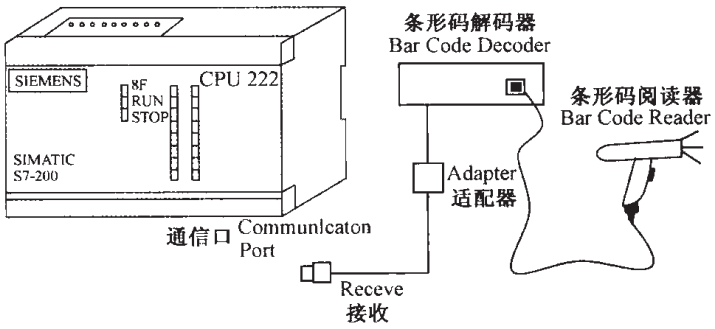


图 7-85　S7-200 与条形码阅读器的连接图

S7-212 将条形码读入的信息接收到缓冲区的程序梯形图如图 7-86 所示。

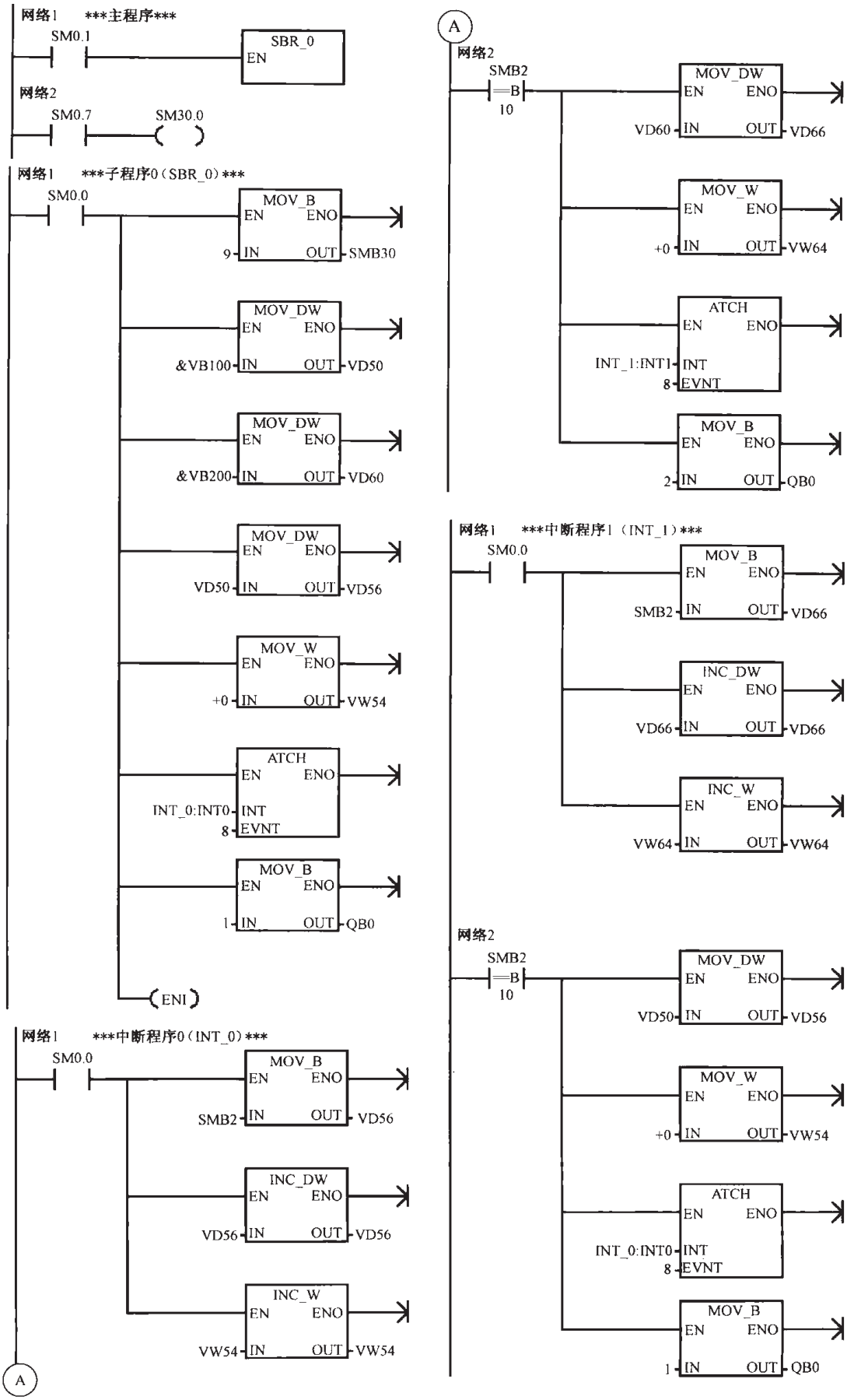


图 7-86 PLC 读取条形码信息程序

如图 7-86 所示的梯形图所对应的语句为：

```
//主程序
LD      SM0.1                    // 第一次扫描标志位 SM0.1=1
CALL    SBR_0:SBR0               // 调子程序 0
网络 2
LD      SM0.7                    // 若在 TERM 模式，则设置 PPI 协议
=       SM30.0                   // 若在 RUN 模式，则设置 Freeport 协议
//子程序 0（SBR_0）
LD      SM0.0
MOVB    9, SMB30                 // 9600 波特，无奇偶校验，每字节 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    +0, VW54                 // 清除缓冲区 0 的字符计数器
ATCH    INT_0:INT0, 8            // 中断程序 0 处理缓冲区 0 的接收
MOVB    1, QB0                   // 设置 Q0.0=1,Q0.1=0
ENI                              // 允许中断
//中断程序 0（INT_0）
LD      SM0.0
MOVB    SMB2, *VD56              // 字符装入缓冲区
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    INT_1:INT1, 8            // 中断程序 1 处理缓冲区 1 的接收
MOVB    2, QB0                   // 设置 Q0.0=0   Q0.1=1
//中断程序 1（INT_1）
LD      SM0.0                    // SM0.0=1.
MOVB    SMB2, *VD66              // 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, 10                 // 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    INT_0:INT0, 8            // 中断程序 0 处理缓冲区 0 的接收
MOVB    1, QB0                   // 设置 Q0.0=1   Q0.1=0
```

实例分析

程序中 MAIN（主程序）：初始化程序；SBR_0（子程序 0）：接收条形码；INT_0（中断程序 0）：缓冲区 0 接收；INT_1（中断程序 1）：缓冲区 1 接收，这样实现两个缓冲区轮流地存储每次新读入的条形码。

实例 92：利用 S7-200 的自由通信口控制调制解调器

实例说明

本例子将说明如何使用 SIMATIC S7-200 通过自由通信口模式控制海叶斯（Hayes）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拨号叫另一台 S7-212。由于海叶斯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 通过自由通信口控制调制解调器，然后通过调制解调器与其他的通信设备连接，从而使得 S7-200 实现用作通信从设备的需求。

实例实现

S7-200 与调制解调器的硬件连接与实例 90 的相似，只不过把实例 90 中的打印机换为调制解调器即可。因此，S7-200 与调制解调器的硬件设备有：

1 台 SIMATIC S7-200；

1 条 PC/PPI 电缆；

1 台合适的接口（取决于调制解调器的输出，多数情况下使用 9 针阳性转换到 25 针阴性的插座，发送线和接收线互换的空调制解调器）；

1 台海叶斯调制解调器。

利用 S7-200 的自由通信口控制调制解调器程序语句为（由于梯形图太长，在此仅仅给出语句）：

```
//主程序
LD      SM0.1                    // 如果第一次扫描，则
CALL    SBR_1:SBR1               // 对信息进行初始化
LDB=    0, MB0                   // 如果要发送信息，则
NOT
CALL    SBR_2:SBR2               // 调用子程序进行发送
LD      SM0.1                    // 如果第一次扫描，则
MOVB    16#1, MB0                // 对发送信息进行初始化
//子程序 1（SBR_1）
LD      SM0.0                    // SM0.0 总是 1
MOVW 16#012B, VW1000             // 在线换码序列字符（+）
MOVB    5, VB1002                // 设置挂起命令
MOVD    16#41544830, VD1003
MOVB    16#0D, VB1007            // 回车
MOVB    9, VB1010                // 设置拨号指令
MOVD    16#41544454, VD1011      // 用按钮拨号
MOVD    16#32363137, VD1015      // .2617: 一个调制解调器的电话号码
MOVB    16#0D, VB1019            // 回车，发送信息
LD      SM0.0
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<SPACE><SPACE>
MOVD    16#50484F4E, VD139      // 字符 PHON
MOVB    16#45, VB143            //字符 E
MOVB    16#20, VB144            //字符<SPACE>
MOVD    16#484F4D45, VD145      //字符 HOME
MOVW    16#0D0A, VW149
MOVB    16#69, SMB30            // 设置自由通信口：9600 波特，
 //每字符 8 位，无奇偶校验
MOVB    0, MB0                  // 显示空状态
MOVB    0, MB2                  // 清除错误响应
ATCH    INT_8:INT8, 8           // 指定接收中断事件 8 调用中断程序 8
ENI                             // 允许中断
R    T37, 1                     // 复位定时器 T37
//子程序 2（SBR_2）
LDB=    16#0, MB0               // 如果命令有效，则
NOT
TON T37,+600                    //激活定时器 37(100ms * 600=60s)
LD      T37                     // 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#02, 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#04, 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                   // 且发送以结束，则
MOVB    16#08, MB0              // 转到挂起状态
// 状态 4～8：挂起电话,挂起状态有以下 5 中状态(状态 4～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
LD   M0.4                           // 如果是发换码序列状态
AW=   +3, VW1008                    // 且计数器等于 3
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
A     SM4.5
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
LD    M0.5                          // 如果是第二次等待状态，且
A    SM4.5                          // 发送完毕
MOVB   16#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
//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
//中断程序 8（INT_8）
LDB=    16#0D, SMB2                 // 如果是 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#02, MB2                  // 置连接状态
R       T38, 1                      // 复位等待定时器 T38
CRETI                               // 返回
LD      SM0.0
MOVB    16#04, MB2                  // 否则，显示出错标志
```

实例分析

该程序实现的功能是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，SIMATIC S7-200 发送信息给海叶斯调制解调器，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。程序主要由四部分组成：主程序（MAIN），对程序进行初始化；子程序 1 (SBR_1)，对信息和自由通信口模式进行初始化；子程序 2（SBR_2）拨号、发送和切断；中断程序 8（INT_8），捕捉并处理来自调制解调器的响应。

实例 93：利用 S7-200 的自由通信口发送实时信息

实例说明

在 PLC 的控制中，需要将 PLC 控制系统的实时信息发送给上位机，以便让上位机实时显示或者处理，在实时发送的信息中，可能有的是静态信息，有的是切换信息，也有一些不断变化的信息，在这些信息中有的是整数，有的是十六进制数。本实例将叙述如何将这些多种多样的变化信息发送给上位机。

实例实现

实例给出了一个简单的“泵站”系统，假定 I0.0 为启动“主泵”的开关，I0.1 为紧急阀的开关，I0.2 为开或关主阀的开关（主阀控制液体流出），需要将三个状态变化中的任一信息通过自由口传送给上位机。I0.0 控制一个静态信息“Pump1 is on”的发送。I0.1 控制着信息“Valve open”或“Valve close”交替发送，I0.2 控制发送液体流出的时间，这个信息在 I0.2 接通时每秒变化一次。由于实例利用的是同一个自由口发送数据，因此不允许同时发送所有信息。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。发送的信息是多种多样的：一条静态信息，一条切换信息，以及一条不断变化的整数或十六进制数全包括在一起的信息。利用自由口发送实时信息的程序梯形图如图 7-87 所示。

对程序中所用的变量说明如表 7-8 所示。

表 7-8　由通信口发送实时信息程序变量说明

变　量	变 量 说 明
VW10	主计数存储器，用来显示液体流出时间（s）
VW20	第二计数存储器，来自 VW10 的复制，用在 SBR2 中进行 IBCD 转换且不擦除计数器里的值
VB80	存储值 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储 12 或 14，即发送缓冲区待发送的 ASCII 码（十六进制）字母数
VD101-VD109	信息: "Valve open"
VD101-VW113	信息:"Valve closed"
VB120	存储数值 28
VD121-VD133	信息: "Flow time in sec"
VB137	以十六进制形式存储“:”作为字段分界符
VB138-VB141	存储秒计数器的 ASCII 码（以 ASCII 的形式发送）
VB142	以十六进制存储“:”作为下一个字段分界符
VB143-VB146	存储秒计数器的 ASCII 码（以 ASCII 的形式发送）
VW147	回车换行，作为发送信息结束

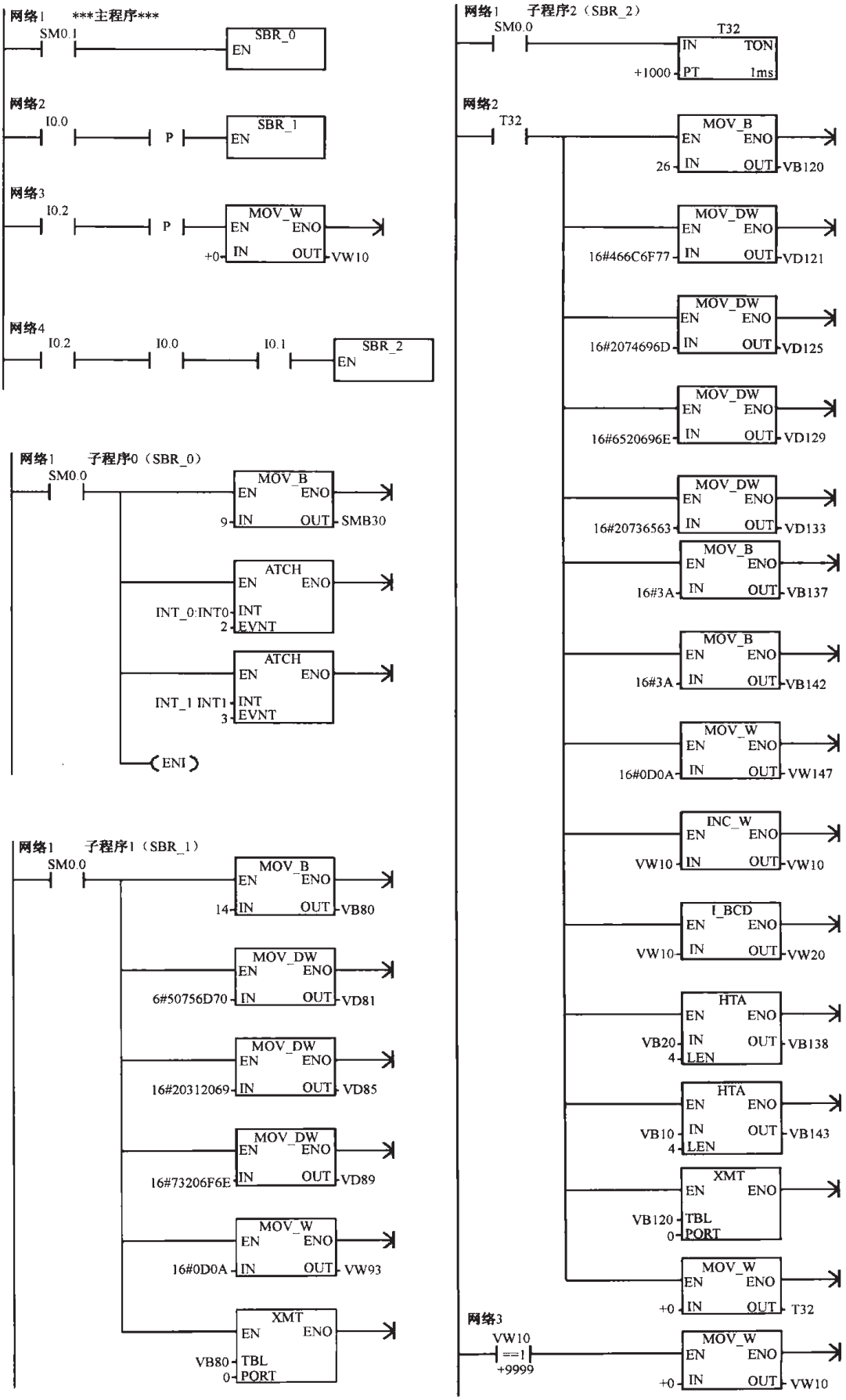


图 7-87 PLC 读取条形码信息程序

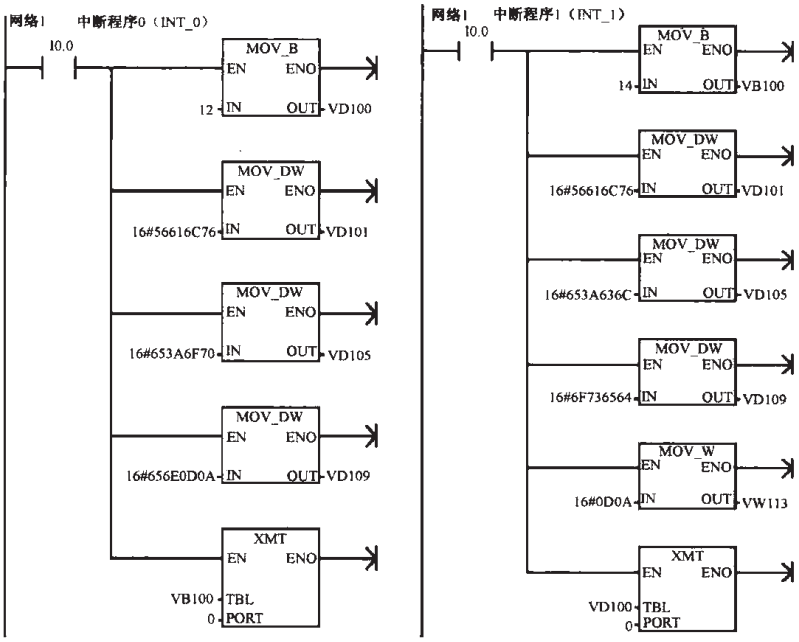


图 7-87　PLC 读取条形码信息程序（续）

图 7-87 的梯形图所对应的语句为：

```
//主程序
LD      SM0.1                  // 首次扫描
CALL    SBR_0:SBR0             // 调用 SBR0
LD      I0.0                   // 若主泵开关 I0.0=1
EU                             // 且上升沿
CALL    SBR_1:SBR1             // 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    SBR_2:SBR2             // 调用 SBR2
//子程序 0（SBR_0）
LD      SM0.0                  // SM0.0 总是 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    INT_0:INT0, 2          // 指定中断事件 2（I0.1 上升沿）调用中断程序 0
ATCH    INT_1:INT1, 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
//子程序 1（SBR_1）
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 i"
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）
//子程序 2（SBR_2）
LD      SM0.0                  // SM0.0 总为 1
TON     T32, +1000             // 启动定时器 T32（1000ms）
```

```
LD      T32                    // 1s 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#20736563, VD133     // " sec"
MOVB    16#3A, VB137           // ":"（字段分界符）
MOVB    16#3A, VB142           // ":"（字段分界符）
MOVW    16#0D0A, VW147         // 回车换行表示信息结束
INCW    VW10                   // VW10 增加 1
MOVW    VW10, VW20             // 把 VW10 复制到 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            // 若流出时间超出 9999 秒
MOVW    +0, VW10               // 则复位秒计数器
//中断程序 0（INT_0）
LD      I0.0                   //
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）
// 中断程序 1（INT_1）
LD      I0.0                   // SM0.0 总位 1
MOVB    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）
```

实例分析

在程序中所用到的中断事件为 2，事件 3 的两中断分别是 I0.1 上升沿和下降沿所产生的中断。如果本实例发送数据的对象是 PC，通过适当的编程，将能使 PC 实时显示 PLC 对“泵站”的控制状态。

思考题

1. S7-200 PLC 的网络通信模块有哪些？
2. S7-200 PLC 的网络通信硬件由哪些组成？
3. S7-200 PLC 网络通信协议有哪些？他们各有何特点？
4. MPI 通信有几种通信方式？
5. S7-200 如何完成 PPOFIBUS-DP 通信？
6. 自由口通信有什么特点，如何完成数据的收发？
7. S7-200 PLC 网络通信形式有哪几种？各有何特点？如何配置各通信参数？

第 8 章　PLC 与人机界面

- 西门子人机界面（HMI）概述
- WinCC flexible 组态软件的使用
- 操作元件的组态
- 显示元件的组态

当今世界，自动化控制水平越来越高，控制过程也越来越趋于复杂。那么作为控制系统的缔造者——人，将如何干预与监视这些控制过程呢？人机界面技术正是在这一需求下产生的。

人机界面（HMI）顾名思义就是人与控制过程交流信息的窗口，人可以通过这个窗口下达控制指令，也可以通过这个窗口观测被控系统的运行状态。人机界面把控制过程变得清楚与透明。

作为控制系统的重要核心部件，PLC 与人机界面的结合是必不可少的。过去采用按钮、开关和指示灯等作为人机界面装置，提供的信息少且操作困难，采用数码管和拨码开关，又会占用大量的 PLC I/O 资源。随着计算机技术的发展，人机界面装置逐步出现了屏幕显示的面板或触摸屏等形式，同时它们的防护等级与可靠性完全按照工业现场设计，从而成为了 PLC 系统的最佳拍档。

本章将通过实例对西门子人机界面系统加以描述。

8.1 西门子人机界面（HMI）概述

人机界面技术是在计算机技术的基础上发展起来的，因此人机界面包含硬件装置和软件两个方面，下面分别从这两个方面对西门子人机界面的情况加以介绍。

8.1.1 人机界面的硬件装置

1. 计算机

在环境条件较好的控制室内，可以直接采用计算机作为人机界面。

控制室是目前的网络化控制系统的一个重要节点，是它所管辖范围内设备信息的汇集处。控制室内的监控设备往往起到承上启下的作用，向上级管理网络发送现场信息，以及向现场设备传送上级下达的控制指令。作为集中监控的场所，控制室也是向操作人员表达系统信息的重要地点。计算机本身就具有良好的界面，同时具有容量大，操作简便等特点，大型的人机界面组态软件，如“intouch”、“IFix”、“组态王”等都是直接在计算机上运行的。西门子公司出品的大型组态软件“WinCC”也是专门在计算机上使用的。

2. 纯按键面板（Push Button Panel）

西门子纯按键面板 PP7、PP17 系列产品采用“即插即用”的安装方式，方便简单，只需要使用相应的安装开孔和总线电缆，大大降低了安装时间与成本。面板与控制系统的连接使用 PROFIBUS-DP 或 MPI 通信方式，并预装有带耐用多色表面 LED 的按键，所有的按键和数字量输入都可以单独组态为按钮或开关。

面板上的按钮可以通过组态与 PLC 中的位地址连接，从而可由这些按钮改变 PLC 内部位的状态，达到控制生产过程的目的。同时可以通过按钮上集成的 LED 显示 PLC 内部位的状态，以此来监控生产过程的运行情况。图 8-1 为 PP7-II 按键面板。按键面板的组态是通过背面的液晶显示器和 6 个按钮完成的（如图 8-1（b）所示），无需专门的组态软件。

（a）PP7-II　　（b）PP7背面

图 8-1　PP7-II 按键面板

3．微型面板（Micro Panel）

微型面板是专门为使用 SIMATIC S7-200 小型 PLC 的应用而定做的。它们带有文本显示屏或触摸屏。微型面板主要有OP73micro、TP 177micro、K-TP178 micro、TD400C 等系列产品。其中，K-TP178 micro微型面板是为中国用户量身定做的触摸屏，它专门与 S7-200 配合使用，屏幕为 5.7 英寸，蓝色 4 级灰度显示。面板采用 32 位 ARM7 处理器，性能优异，操作界面为触摸屏+按键方式，并具有操作声音提示功能，具有 1MB 超大存储空间，可组态 500 个画面和 2000 条报警信息。在操作安全方面具有强大的密码保护功能，能进行 50 个用户组管理。K-TP178 micro触摸屏具有较高的鲁棒性，能够防冲击和震动，并防水耐脏。图 8-2 为 TP 177micro 微型面板图。

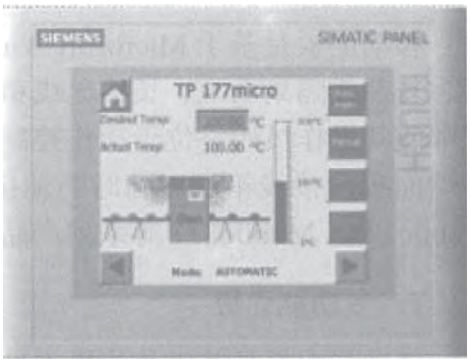


图 8-2　TP 177micro微型面板

图 8-3　OP177B PN/DP 面板

4．键控式面板

键控式面板将操作键与显示屏相结合，虽然还不是真正意义上的触摸屏，但键控式面板也具有十分强大的功能，以确保高效的控制和监视。基于文本的显示设备可直接显示状态信息；基于图形的显示设备可显示曲线及棒图。此外，OP 系列还免费提供参量管理、线性转换、可变限值变量、可装载固件、在线语言选择及更多功能。键控式面板主要有OP77B、OP177B、OP277、OP77A等产品。如图 8-3 所示为 OP177B 系列的 PN/DP 面板图。

5．触摸屏面板

触摸屏面板可以自定义显示屏上的图形式按钮，不再需要传统的按钮，表达更直观，前面板的防护等级为 IP65，可耐热、耐冷、防油污、防潮，图形功能非常强大。主要有 TP177A、TP177B、TP277 系列产品。

其中，TP177B 是通用型触摸面板，适用于 PROFIBUS-DP 或 ROFINET 的各种应用环

境，可纵向或横向安装，用户存储器容量为 2MB，有一个 RS-422/485 接口和内置 USB 接口，从而可以连接键盘、鼠标、打印机、读码器，并可下载项目文件。触摸屏内部装有标准多媒体卡（MMC）插槽，用于配方数据、归档、组态和系统数据的备份与恢复。TP177B 有 TP 177B DP（4 级灰度蓝色 SIN 显示屏）和 TP177B PN/DP（256 色彩色 SIN 显示屏）两种型号选择。TP177B PN/DP 支持 Sm@rtService 和 Sm@rtAccess（Sm@rtService 可通过以太网与 Intrantet/Internet 访问远程 HMI 设备，进行远程维护；Sm@rtAccess 用于不同 HMI 系统之间的客户机、服务器通信），如图 8-4 所示。

图 8-4　TP 177B PN/DP 面板

6．移动面板

移动面板是基于 Microsoft Windows CE 操作系统的移动 HMI 设备。在涉及大型生产工厂、复杂或隔离系统、长传送线和生产线以及材料处理的应用中，选择移动面板可使调试工程师或其操作员始终位于动作控制中心，及时观看工件或过程，从而可以快速响应操作要求。移动面板的接线箱可以随时随地接入，且能确保运行中无故障的热插拔。移动面板主要有 Mobile177，Mobile277 等系列产品，图 8-5 为移动面板图。

7．多功能面板

多功能面板是基于传统操作设备（例如操作员面板和 PLC）与 PC 之间的一种全新设备，它的操作系统为 Windows CE 5.0，具有内置的 IE 浏览器，可以访问 HTML 网页。多功能面板使用 PROFINET/Ethernet 联网即可以访问办公环境。归档和配方可以保存在上一级计算机中并进行集中管理，同时还可以进行数据交换。多功能面板可以使用网络打印机。MP277/MP377 是其主要系列产品，图 8-6 为 MP377 键控屏界面图。

图 8-5　移动面板

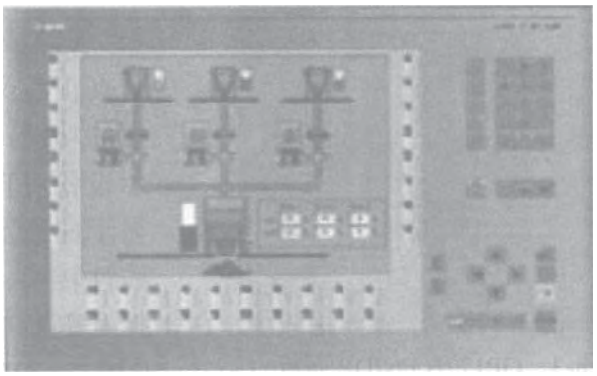


图 8-6　MP377 键控屏界面

8.1.2　人机界面的组态软件

1. SIMATIC WinCC

SIMATIC WinCC 是在计算机上使用的人机界面组态软件，是一种可扩展的过程可视化系统（SCADA，Supervisory Control And Data Acquisitio，数据采集与监视控制系统），能高

效控制自动化过程。SIMATIC WinCC 基于 Windows 平台，可实现完美的过程可视化，能为各种工业领域提供完备的操作和监视功能，涵盖从简单的单用户系统直到采用冗余服务器和远程 Web 客户端解决方案的分布式多用户系统。WinCC 的特点之一是其整体开放性。它可方便地与标准程序和用户程序组合在一起使用，建立人机界面，精确地满足实际需要。

WinCC 可以作为后台服务运行，可将 IO 服务器、数据归档服务器和 Web 服务器放置在安全数据中心。用户无需鼠标、键盘和显示器，甚至不需登录即可运行。集成 SIMATIC Logon，使用 Windows 用户验证机制，IE 的 Web 客户端，有效避免了针对 IE 的木马，增强系统安全性。支持 Windows Vista 及其界面风格，支持玻璃材质、渐进色、光影效果和阴影等效果。动画播放采用全新的动画播放控件（avi、gif 等），并支持.NET 和 XAML 控件，能够完美显示在线趋势、功能曲线、配方控件。

WinCC 系统最大提供 256KB 点外部 I/O，并支持超大系统。系统集成有第三方驱动，如 Modbus TCP、Ethernet/IP，可作为系统平台软件，集成其他子系统。

2. WinCC flexible

WinCC flexible 是在 Protool 人机界面组态软件的基础上发展而来的，并兼容 Protool，综合了 WinCC 的开放性和扩展性，以及 Protool 的易用性的新型人机界面组态软件。它支持多种语言，可以全球通用。除了用于 HMI 设备组态以外，WinCC flexible 高级版的运行软件还可以运行于 PC。WinCC flexible 具有开放的扩展功能，带有 Visual Basic 脚本功能，集成了 ActiveX 控件，可以将人机界面集成到 TCP/IP 网络。

WinCC flexible 软件由三个部分组成：工程系统、运行系统和选件。

（1）WinCC flexible 工程系统（简称 ES），是用于处理组态任务的软件，采用模块化设计。西门子为各种不同的 HMI 设备量身定做了不同价格和性能档次的版本，如微型版、压缩版、标准版和高级版，随着版本的逐步升高，软件的功能以及支持的设备范围也不断扩展。

（2）WinCC flexible 运行系统是用于过程可视化的软件，运行系统在过程模式下执行项目，实现与自动化系统之间的通信、图像在屏幕上的可视化、各种过程操作、记录过程值和报警事件。运行系统支持一定数量的过程变量（Powertags），该数量由许可证确定。

（3）WinCC flexible 选件可以扩展软件的标准功能，每个选件都需要一个许可证。

通过 WinCC flexible，才能使西门子 HMI 设备具有实际的功能，WinCC flexible 软件与 HMI 设备共同构成 HMI 控制系统，并完成下列任务：

- 过程可视化。过程显示在 HMI 设备上。HMI 设备上的画面可根据过程变化动态更新。这基于过程的变化。
- 操作员对过程的控制。操作员可以通过 GUI（图形用户界面）来控制过程。例如，操作员可以预置控件的参考数值或者启动电动机。
- 显示报警。过程的临界状态会自动触发报警。例如，当超出设定值时报警。
- 归档过程值和报警。HMI 系统可以记录报警和过程值。该功能使用户可以记录过程值序列，并检索以前的生产数据。
- 过程值和报警记录。HMI 系统可以输出报警和过程值报表。例如，用户可以在某一轮班结束时打印输出生产数据。

- 过程和设备的参数管理。HMI 系统可以将过程和设备的参数存储在配方中。例如，可以一次性将这些参数从 HMI 设备下载到 PLC，以便改变产品版本进行生产。

8.2 WinCC flexible 组态软件的使用

安装完 WinCC flexible 软件之后，会在 Windows 桌面上生成一个图标，下面以 TP170A 为例，来说明 WinCC flexible 软件创建项目的过程。

实例 94：WinCC flexible 组态项目的创建

实例说明

本实例通过 TP170A 画面组态项目的创建，说明在 WinCC flexible 软件中如何创建一个新的项目。

实例实现

1．使用项目向导创建新项目

双击图标，打开如图 8-7 所示的界面，选择“使用项目向导创建一个新项目”。出现如图 8-8 所示的界面，选择设备组态预定义场景，这里选择“小型设备”，单击界面下方的“下一步”按钮进入下一窗口（如图 8-9 所示）。

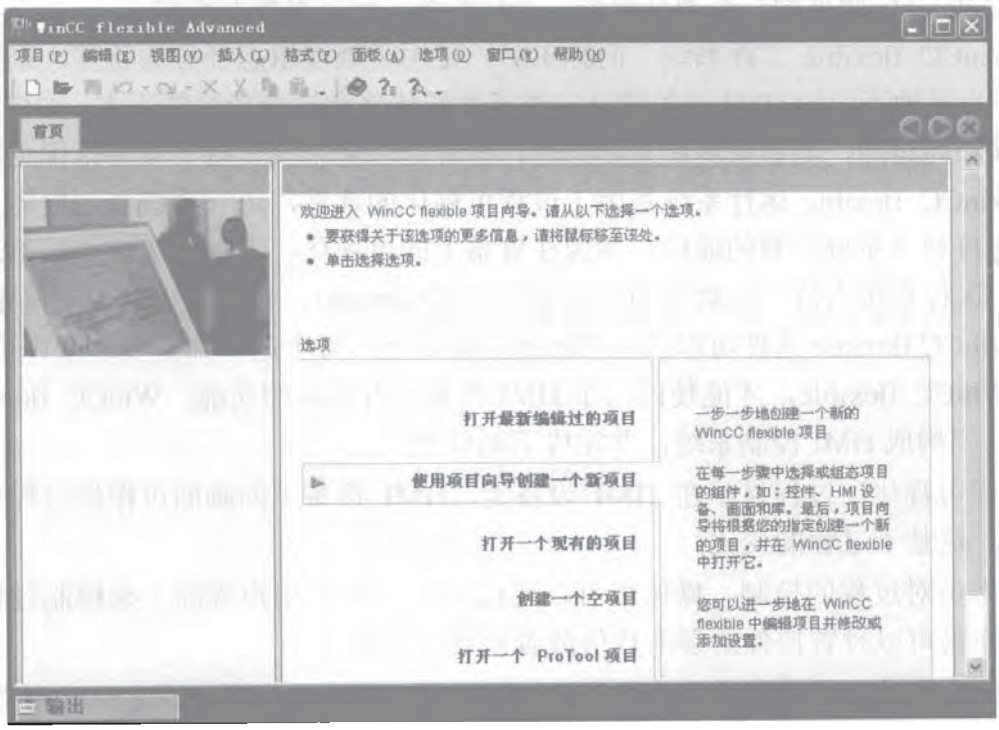


图 8-7 WinCC flexible 软件初始画面

图 8-8　预定义场景选择

在图 8-9 所示界面中，单击红色框所标识的按钮，则出现触摸屏选择窗口，如图 8-10 所示。在相应的菜单下选择 TP170A，单击“确定”按钮返回如图 8-9 所示界面。在连接选择的下拉菜单中选择连接；在控制器选择的下拉菜单中选择 SIMATIC S7-200。单击“下一步”按钮，打开组态画面模板（如图 8-11 所示）。

在组态画面模板中，选择相应的选项，并可添加公司标识图片。完成后单击“下一步”按钮，打开组态画面浏览界面（如图 8-12 所示），设置画面的层次结构。设置完成后单击“下一步”按钮，进入组态系统画面（如图 8-13 所示）。

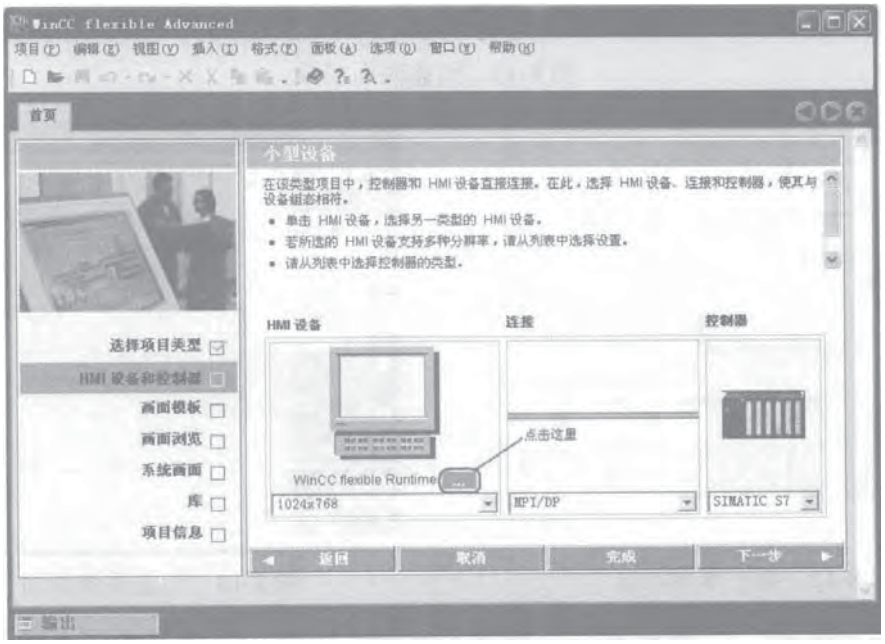


图 8-9　设备选择画面

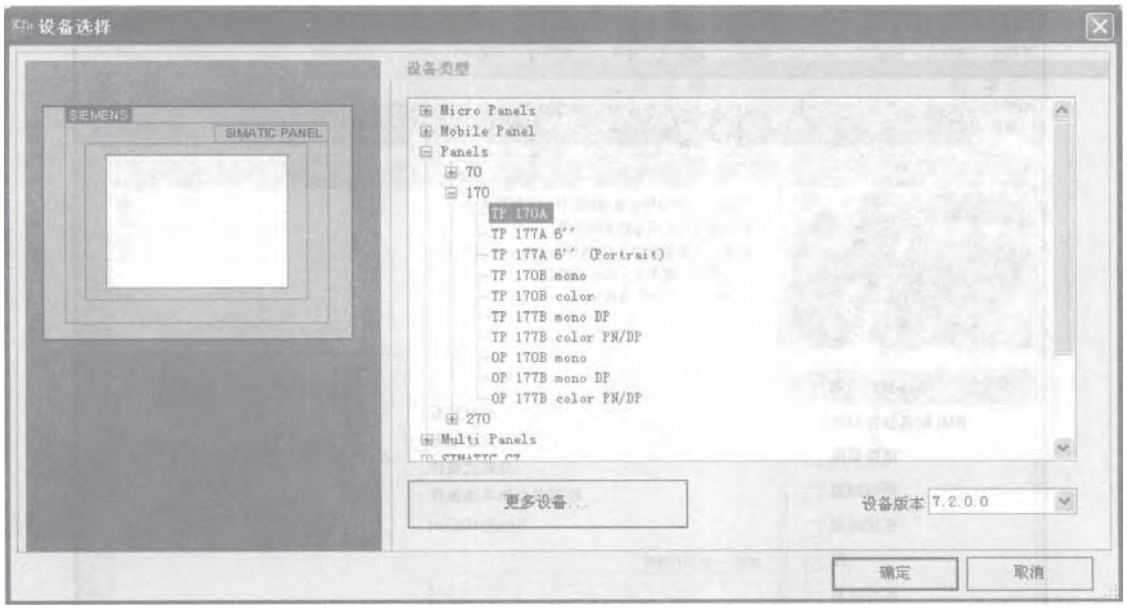


图 8-10 选择 HMI

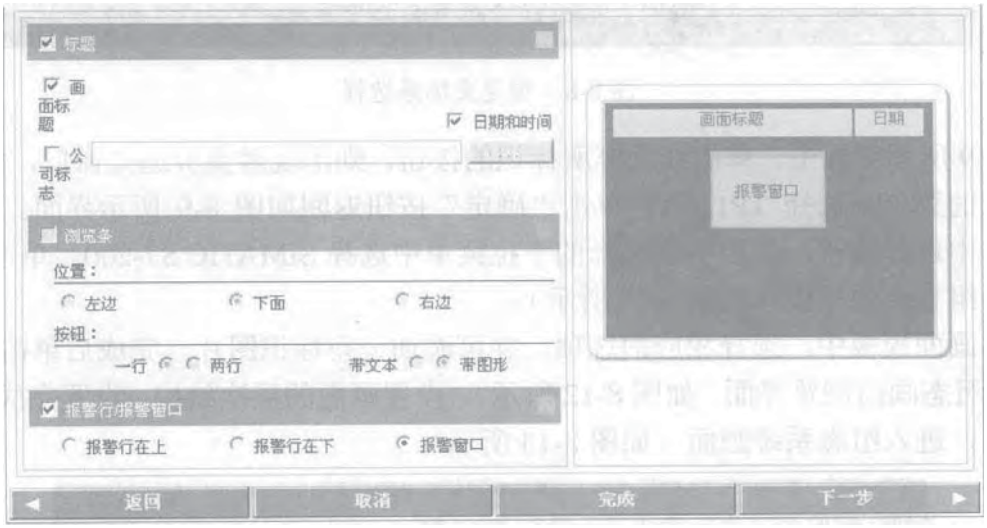


图 8-11 组态画面模板

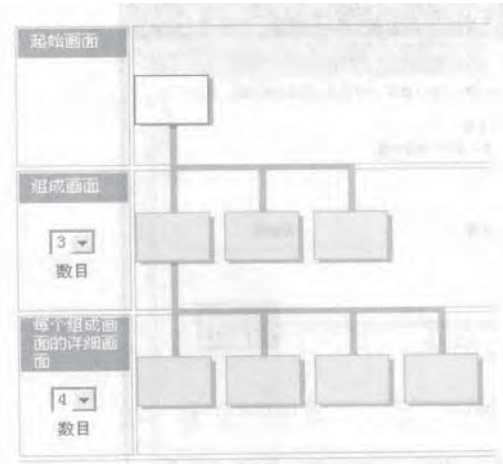


图 8-12 组态画面浏览

图 8-13 组态系统画面

在组态系统画面中，一般采用默认设置，然后直接进入库选择画面（如图 8-14 所示）。窗口左侧为“可用的库”，通过 按钮可以将“可用的库”中的选项添加到右侧“选择的库”

区域中，以便后续组态操作中使用。

图 8-14　库选择画面

库设置完成后，进入项目信息设置，可以输入项目的名称、项目作者、创建日期等内容。最后单击“完成”按钮，生成项目。软件直接打开用户组态界面，如图 8-15 所示。

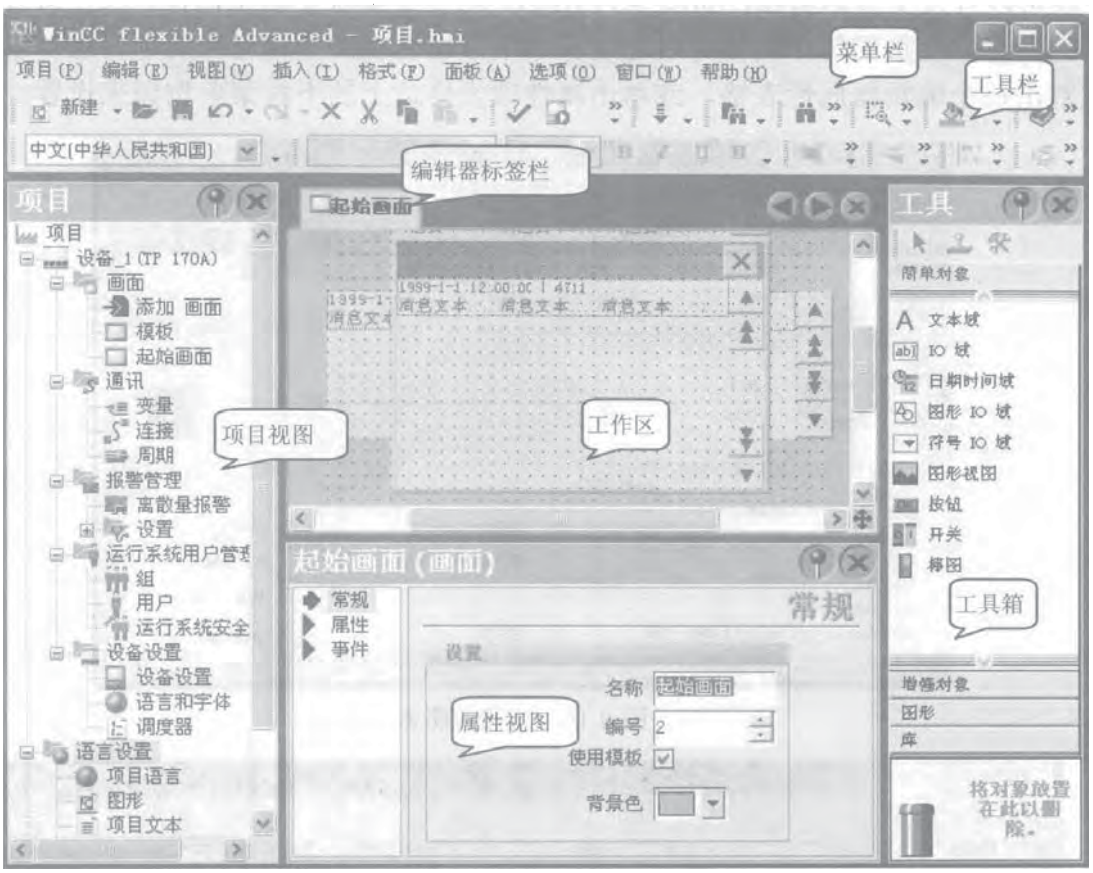


图 8-15　用户组态界面

在用户组态界面中，用户在工作区编辑项目对象，在属性视图区选取对象属性。使用界面右侧的工具箱，可以快速创建对象。不同型号的 HMI 设备，在工具箱中的可用对象是不同的。此时，可以对这一初步建立的项目进行保存。

2. 建立 HMI 设备与 PLC 之间的连接

单击界面左侧项目视图中“通信”文件夹下的“连接”图标，打开连接编辑器，如图 8-16 所示。连接表中自动出现与 S7-200 的连接，默认名为“连接_1”。连接表的下方是默认属性视图。图中的参数为默认值，是项目向导自动生成的，一般直接采用默认值，用户也可

以根据情况修改。

3. 变量的生成与组态

（1）变量的分类

变量分为外部变量和内部变量，每个变量都要给出一个符号名和数据类型。外部变量是 HMI 与 PLC 进行数据交换的纽带，是 PLC 中定义的存储单元的映像，其状态和数值随 PLC 程序的执行而改变。

内部变量存储在 HMI 中，与 PLC 没有连接关系，用于 HMI 设备内部计算或执行其他任务，且只有 HMI 可以访问。内部变量用名称来区分，没有地址。

（2）变量的生成与属性设置

变量的生成与设置是通过变量编辑器完成的。双击项目视图中“通信”文件夹下变量图标，就可以打开变量编辑器，如图 8-17 所示。双击变量表中的空白行，软件将会自动生成一个新的变量。变量的每个属性可以通过界面下方属性视图中相对应的项目进行设置；也可以通过双击每个编辑器属性表格，或单击属性项表格边上的下拉菜单按钮来设置。

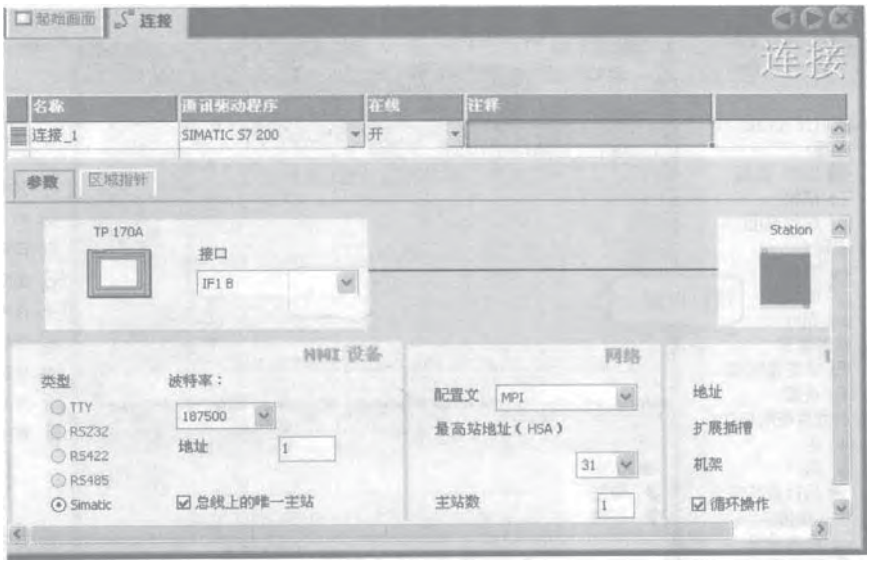


图 8-16　连接编辑器

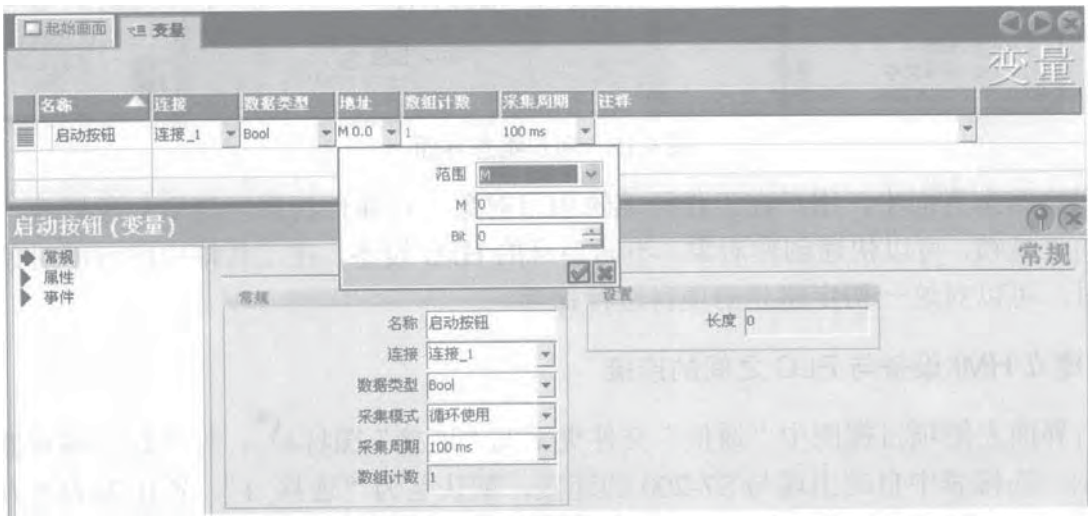


图 8-17　变量编辑器

4．画面的生成与组态

画面是操作人员在生产过程中所要面对的直接界面。HMI 组态的一项重要内容就是画面的制作。

画面是由静态元件和动态元件组成的。静态元件（如文本或图形对象）用于静态显示，在设备运行时，它们的状态不会发生变化，不需要变量与之连接，不能由 PLC 进行更新。

动态元件指的是用图形、字符、数字趋势图和棒图等画面元件来显示 PLC 或 HMI 设备存储区中变量的当前状态或当前值。因此，动态元件的状态受变量的控制，需要设置与它连接的变量。

在项目生成时，系统会自动生成一个初始画面（如图 8-15 所示的工作区）。通过画面编辑器下方的属性视图，可以对画面的属性进行设置。至此，项目创建完成，后续的组态工作可以在这个项目下继续进行。

每一个元件都有对应的属性视图，在属性视图中一般有以下项目：

“常规”项：用来设置元件最重要和基本的属性。

“属性”项：常用于静态设置，如文本的字体大小、对象的位置和访问权限等。

“动画”项：用于对象外观或位置的动态设置，要用变量接口来实现。

“事件”项：用于设置在特定事件发生时执行的系统函数。

实例分析

WinCC flexible 组态项目的创建是 HMI 设备软件开发的第一步，要完成 HMI 设备的确定，与 PLC 通信协议的确定以及变量和基本画面的确定。

8.3　操作元件的组态

实例 95：按钮的生成与组态

实例说明

HMI 上的按钮与接在 PLC 输入端的实际按钮的功能是相同的，用来给 PLC 提供开关量输入信号，属于 BOOL 量操作。但画面上的按钮元件不能与 PLC 的数字量输入（I0.0）相连接，而要与存储区位（如 M0.0）连接。本实例说明了如何组态 HMI 的按钮控件。

实例实现

1．按钮的生成

在如图 8-15 所示界面中，单击工具箱中的“简单对象”组，出现常见的画面元件图标。用鼠标单击其中的“按钮”图标后（图标位置出现选中条），将鼠标放在画面位置，此时鼠标变成“+”形。按住鼠标左键在画面上拖拉十字光标至合适位置，松开左键，就会在画面上生成按钮元件（如图 8-18 所示）。鼠标操作环绕在按钮周围的 8 个小正方形即可放大、缩小和移动按钮。

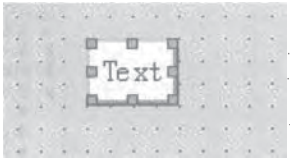


图 8-18　按钮的生成

2. 按钮的属性设置

在属性视图中，可对按钮的属性进行设置。选中左侧窗口中的“常规”选项，在右侧对话框中选择按钮模式为“文本”。在文本域的选项中，选择“文本”。在“OFF 状态文本”输入框中写入按钮在弹起状态时的显示内容。若勾选“ON 状态文本”，则可在相应的输入框中写入按钮在按下状态时的显示内容。如不勾选该项，则按钮在按下和弹起时显示同样的内容。这里按下时显示内容为“输入”，弹起时显示内容为“按钮”，如图 8-19 所示。

图 8-19 按钮常规属性组态

选中属性视图左侧窗口中的“属性”项，在其子项中选择“外观”，在右侧对话框中将按钮背景色修改为浅灰色，如图 8-20 所示，按钮表面颜色成为浅灰色。在其他下属子项中，还可以对按钮的布局、文本格式、名称等属性进行修改。

3. 按钮功能的设置

打开属性视图左侧窗口中的“事件”项，选中子项中的“按下”，单击右侧“函数列表”对话框中最上面一行，再单击该行右侧出现的图标，则会出现“系统函数”的树形列表结构。选择“编辑位”文件夹中的函数“Setbit”（置位），如图 8-21 所示。在表的第二行会出现两个分列，分别显示“变量（InOut）”和“<无值>”。双击“<无值>”栏，激活图标，在下拉菜单中，会出现变量表，如图 8-22 所示。选择按钮所要操作的变量（本例中为启动按钮）。在运行时，按下按钮将使 PLC 的启动按钮位（M0.0）置 1。

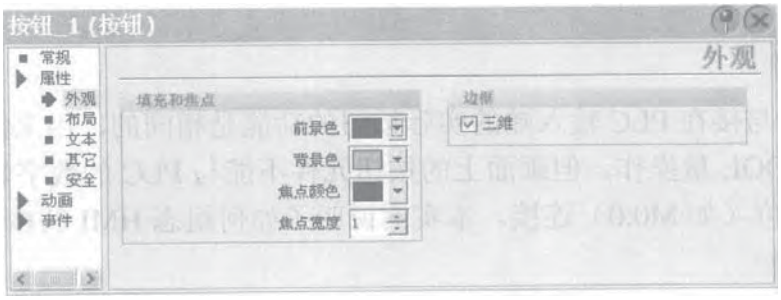


图 8-20 按钮外观组态

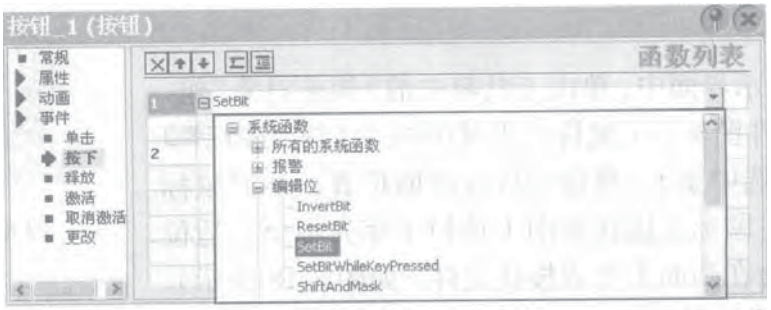
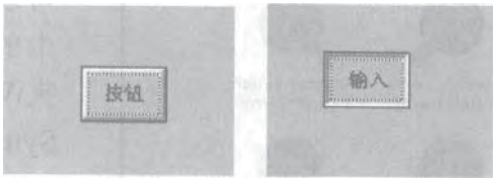


图 8-21 按钮按下时执行的函数

图 8-22　按钮按下时操作的变量

4．模拟运行

将项目存盘（存盘时，软件会自动对项目进行编辑），然后单击组态界面最上部工具栏中图标（使用仿真器启动系统运行）对组态结果进行仿真运行，图 8-23 是仿真界面下用鼠标操作按钮前后的状态，当松开鼠标时，按钮自动复位。



（a）操作前　　（b）按下后

图 8-23　按钮模拟运行

实例分析

按钮是 HMI 设备常用的 I/O 输入方式，采用 HMI 的组态按钮可以节省硬件的投入。

实例 96：开关的生成和组态

实例说明

开关也是一种 BOOL 量操作元件。与按钮不同的是，开关可以保持操作后的状态，不能自复位。本实例对开关进行生成与组态。

实例实现

选择工具箱中“简单对象”组中的开关图标，然后用鼠标在画面上拖拉生成开关元件（默认状态下为文本切换，图形与按钮形式一样）。在开关属性视图的“常规”对话框中，可以对开关的显示状态和操作变量进行设置，如图 8-24 所示。

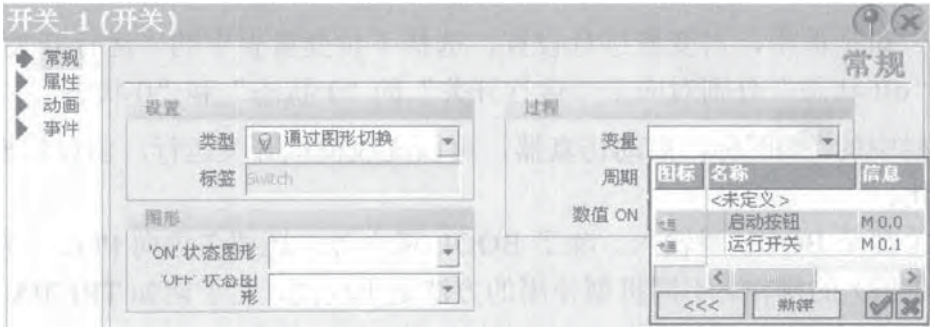


图 8-24　开关常规属性组态

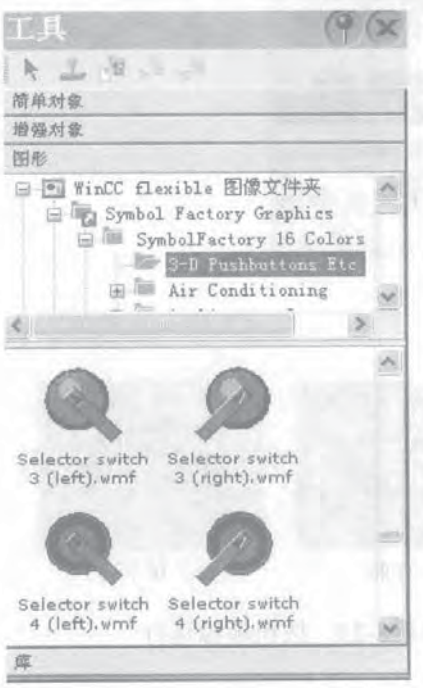


图 8-25　图形库

在“设置”对话框中，若选择“通过文本切换”，则在文本输入框中分别输入开关 “on 状态”和“off 状态”时的显示内容，这与按钮的操作类似。如选择“通过图形切换”（如图 8-24 所示），则在下面的“图形”输入框中，分别设置开关“on 状态图形”和“off 状态图形”的图形显示。这种开关显示更为直观与形象。

WinCC flexible 软件自带的图形库中有大量制作精美的工控元件图形，通过工具箱的“图形”组中的目录树，可以看到这些图形，并进行选择，如图 8-25 所示。图形库中的文件为 wmf 格式，存储在 WinCC flexible 安装文件夹\WinCC flexible Support\Gra-phics\SymbolFactory Graphics\SymbolFactory 16 Colors\3-D Pushbuttons Etc 中，并可由 Visio 软件进行修改。

单击“on 状态图形”选择框右侧的图标，在下拉图框中选择左上角的“用文件创建新图形”选项，则会打开 Graphics 文件夹。此时可以采用 Windows 文件夹操作，选择想要显示的图形，在这里选取的是 WinCC flexible 自带的开关图形 Selector switch2 (right).wmf，如图 8-26 所示。采用同样方法，在“off 状态图形”选择框中创建和选择图形 Selector switch2 (up).wmf。

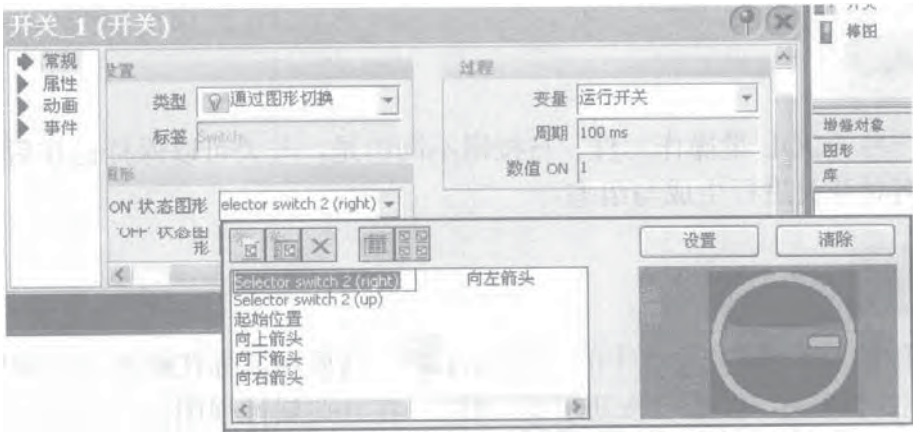


图 8-26　图形切换开关的组态

在“过程”对话框中，对变量进行设置，选择下拉变量表中的“运行开关”。则开关的“on 状态”和“off 状态”分别对应于“运行开关”的“1 状态”和“0 状态”。

通过工具栏中的图标，启动仿真器，可以离线模拟开关运行，由鼠标操作，可观察开关图形的变化。

按钮和开关都是 BOOL 量输入。除了 BOOL 量之外，还要考虑向 PLC 下发数值量（如控制温度的设定值）的操作。不同机型使用的方法是不同的，对于诸如 TP170A 型的低档机，对 PLC 数值量的写入是通过按钮或开关中事件函数 IncreaseValue 或 DecreaseSetVale 的设置来完成的。对于 MP 等高级机型来说可以通过滚动条完成数值变量的写入。

实例分析

除了按钮，开关对象的使用是另一种 I/O 输入命令选择。可根据实际应用灵活运用。

实例 97：滚动条的组态

实例说明

在 HMI 中数值变量的输入，也是一项重要的操作内容。滚动条控件用于操作员输入和监控变量的数值。操作员改变滑块位置输入数值，在滚动条的数字显示处显示给出的数值。本实例用于说明滚动条控件的组态过程。

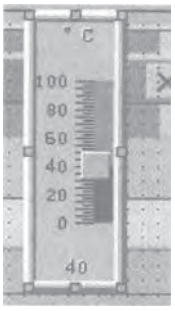


图 8-27　滚动条元件

实例实现

选中工具箱“增强对象”组中“滚动条”图标后，在画面内用鼠标竖向拖拉出滚动条元件，如图 8-27 所示。

在变量表中创建 WORD 型变量“温度设置”，在图 8-28 所示的滚动条常规属性设置窗口中，设置连接的变量为“温度设置”，设置滚动条最大值和最小值分别为 100 和 0。

在属性视图，“属性”项下面的“图样”窗口中，可以设置用滚动条输入物理量的单位，它将显示在滚动条的上方，如图 8-29 所示。

在“外观”窗口（如图 8-30 所示）中，背景填充样式可选“实心的”，如果选“透明的”，将隐藏背景和边框。刻度样式有 3 种选择，也可以选择隐藏。

在“布局”窗口（如图 8-31 所示）中，可以对部件的显示或隐藏进行选择。其中“标记标签”指刻度旁边的刻度数值显示。

在“边框”窗口中，用图形形象地说明了各边框参数的含义（如图 8-32 所示）。

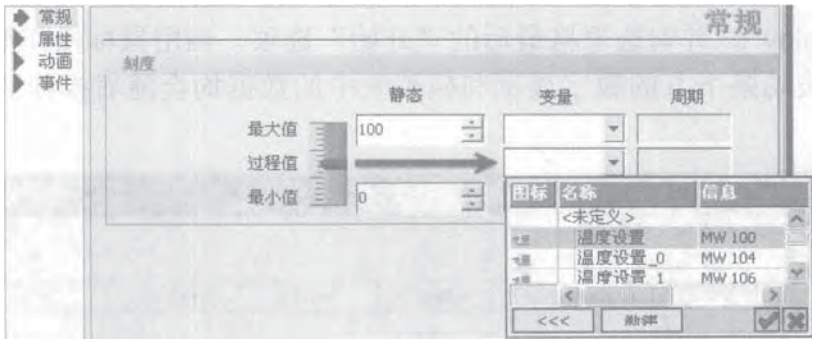


图 8-28　滚动条常规属性组态

图 8-29　滚动条的图样组态

图 8-30 滚动条的外观组态

图 8-31 滚动条的布局组态

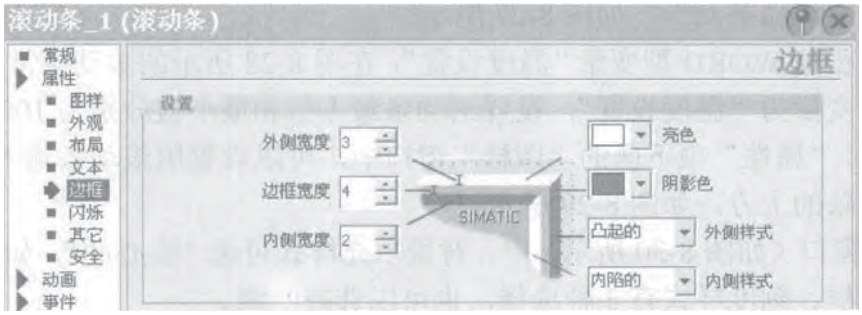


图 8-32 滚动条的边框组态

所有设置完成后存盘，并通过图标，对滚动条的操作运行进行离线模拟。此时，模拟器要使用由模拟器启动的模拟表。在模拟表中创建变量“温度设置”（如图 8-33 所示），“模拟”设定为“Display”，并勾选表格最后的“开始”选项。当用鼠标操作滚动条时，可以看到滚动条位置、滚动条下方的数字显示和模拟表中的数据均会随着操作同步发生变化，如图 8-34 所示。

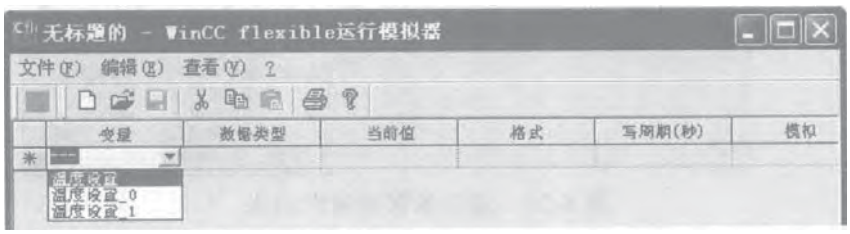


图 8-33 模拟运行表

实例分析

HMI 的滚动条对象，使 PLC 系统抛开了传统的电位器输入模式，极大地简化了模拟量命令输入的硬件组成，充分体现了 HMI 的优势。

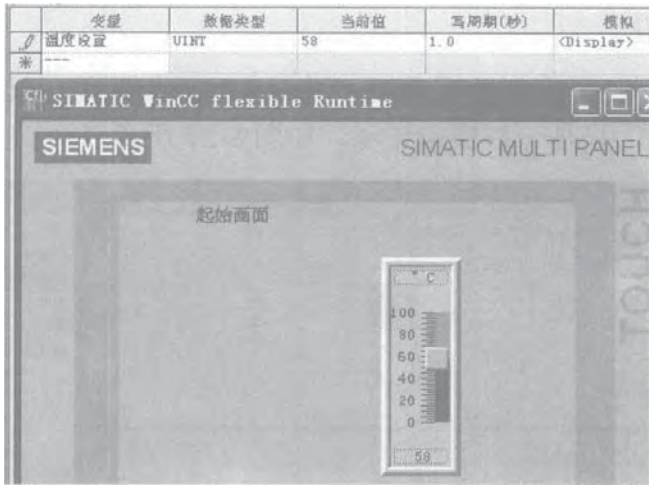


图 8-34　滚动条的模拟运行

8.4　显示元件的组态

实例 98：指示灯的组态

实例说明

HMI 不仅用于指令的下发，状态的显示也是 HMI 的主要功能之一。系统最基本的状态就是 I/O 状态，指示灯是控制系统常用的显示设备 I/O 状态的元件。相应的在 HMI 也有指示灯对象。本实例说明了 HMI 指示灯对象的组态过程。

实例实现

在工具箱内没有用于显示 ON/OFF 位变量的指示灯对象，一般采用对象库中的指示灯对象。

“库”可使用两种方法调入，一种是在项目生成时就将库直接组态进入系统，见第 8.2 节；另一种在项目生成后，由文件夹操作调入，具体操作为：

选中工具箱中的“库”组，在库工作区单击鼠标右键，弹出一个快捷菜单，选择“库…”后，在新出现的二级菜单中选择“打开”，出现文件夹操作对话框。按照“SIMATIC WinCC flexible\WinCC flexible Support\Libraries\System-Libraries”的次序打开文件夹，就可看到 WinCC flexible 软件自带的库文件，单击文件并打开，就可将库文件调入组态库组中。这里调入“Button_and_switches.wlf”库文件。

在工具箱中的“库”组中双击“Button_and_switches”文件夹，在下属的选项中选择“Indicator_switches”，如图 8-35 所示。由鼠标选择“PilotLight-1(en-Us)”图形之后，在画面中用鼠标拖拉出指示灯元件。

在画面下面属性视图的“常规”对话框中，默认出现“通过图形切换”格式，“on 状态图形”为“Signal1_on1”，“off 状态图形”为“Signal1_off1”。通过“on 状态图形”和“off 状态图形”操作框右侧的下拉菜单按钮可以看到这两种状态下的图形，如图 8-36 所示。在过程子项中，对指示灯连接的“变量”进行设置，这里设置为“运行开关”，以便于在模拟运

行时观察指示灯状态变化。

存盘并通过  图标打开模拟运行环境，通过鼠标操作开关，就可看到指示灯的变化。但这时没有与 PLC 连接，只是一种模拟显示，如图 8-37 所示。

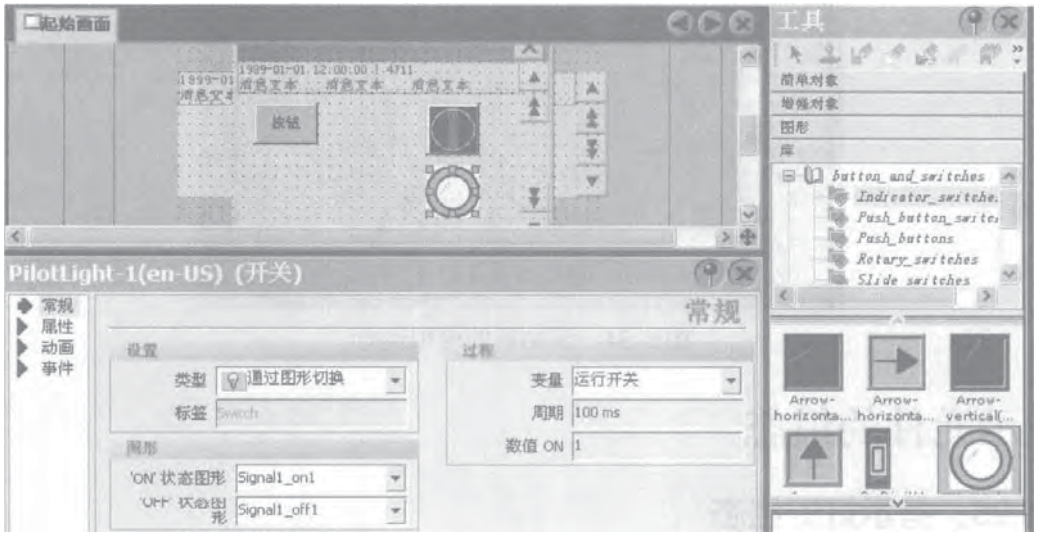
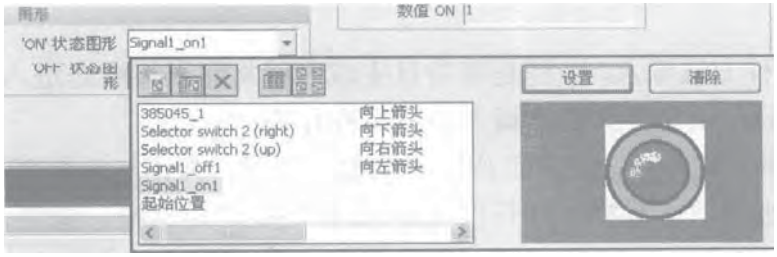


图 8-35　指示灯组态

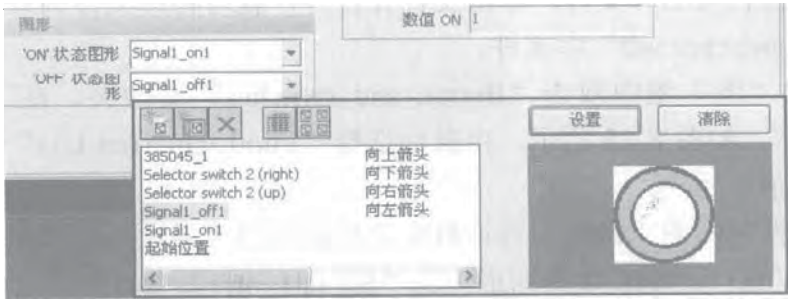
为了表明指示灯所对应的设备，可以用工具箱中的“文本域”在指示灯边上输入文字，予以表示。

鼠标单击选中工具箱中“文本域”图标A，然后在画面上拖拉出文本显示单元，默认文本为“Text”。双击生成的文本域，输入需要的文字，或在属性视图“常规”对话框中输入文字。

在属性视图中，可以设置文本的字体、大小、颜色、背景、填充样式、边框、垂直放置或水平放置等内容。添加了文本说明的指示灯如图 8-37 所示。



（a）指示灯“亮”显示图片



（b）指示灯“灭”显示图片

图 8-36　指示灯图形显示

实例分析

指示灯对象的使用，使 HMI 画面上可以直接显示设备的启停状态、开关的动作状态，以及现场一些二位传感器的运行状态等。相比于硬件的指示灯，不仅具有灵活的特点，同时也取消了大量的硬件投入。

（a）操作前　　（b）操作后

图 8-37　指示灯与开关配合运行及文本显示

实例 99：日期时间显示的组态

实例说明

在 HMI 上可以显示日期时间，本实例对日期时间对象的组态过程进行了详细的描述。

实例实现

在工具箱“简单对象”中有“日期时间域”选项，在高级 HMI 设备的“增强对象”中有“时钟”选项，可以通过这两个选项在画面中完成下列功能：

（1）输出日期和时间值。

（2）重新设置日期和时间。

将工具箱的“简单视图”组中的“日期时间域”图标和“复杂视图”中的“时钟”图标拖放到画面中，如图 8-38 所示。

在时钟属性视图的“属性”类的“外观”对话框中（如图 8-39 所示），可以设置钟面的颜色，钟面的填充样式可以选择“实心的”、“透明的”和“透明框”。时钟的指针可以选择填充色或“空心的”，可以改变各部分的颜色。

图 8-38　日期时间显示

图 8-39　时钟外观组态

在属性视图的“属性”类的“布局”对话框中，可以设置在改变时钟的尺寸时是否保持正方形状。

在时钟的属性视图的“常规”对话框中，可以设置钟面的背影图案，选择是否显示钟面。如果不激活“模拟显示”复选框，将采用数字显示方式，显示方式与简短格式的日期时间域相同，但是不显示秒的值。

对日期时间域，其类型如果组态为“输出”，只用于显示；如果组态为“输入/输出”，还可以作为输入域来修改当前的时间。可以选择只显示时间，或者只显示日期（如图 8-40 所示）。可以使用系统时间作为日期和时间的数据源。如果选择“使用变量”，日期和时间由一个 DATA_AND_TIME 类型的变量提供，该变量值可以来自 PLC。

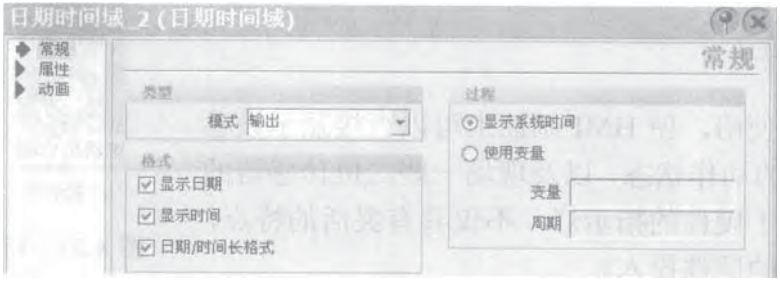


图 8-40　日期时间域常规组态

图 8-41　日期时间域输入键盘

图 8-38 左上角的日期时间以简短格式显示，左下角的日期时间以长格式显示。图中右下角的时间日期域的属性为“输入/输出”，其他的时间日期域的属性均为“输出”，利用时间日期域的输入功能，可以设置实时时钟的日期和时间。在模拟运行时单击右下角的时间日期域，在出现的字符键盘（如图 8-41 所示）中输入新的时间，单击 ↵ 键后返回画面，可以看到修改的效果。在计算机上模拟时，实际上修改的是计算机的系统时钟。

实例分析

HMI 的时间显示功能，弥补了 PLC 在这一方面的不足之处，使操作人员可以很直观的对 PLC 的内部时钟进行显示与控制。

实例 100：IO 域的组态

实例说明

输入域与输出域统称 IO 域，分为 3 种模式：

（1）输出域：只显示变量的数值。

（2）输入域：由操作员输入数字、字母或符号，并将数值传送到指定的 PLC 变量中。

（3）输入/输出域：同时具有输入和输出功能，操作员可以用它来修改变量的数值，并将修改后的数值显示出来。

3 种模式的选择，仅需在组态时选择 IO 域的相应属性就可以实现，本实例说明了 IO 域的组态过程。

实例实现

为组态 IO 域，首先在变量表中创建整型（Int）变量“变量_1”和“变量_2”，并创建 8 字节的字符型（String）变量“变量_3”。选中工具箱中“简单对象”的 IO 域图标 ab| ，在画面上创建 3 个 IO 域对象，如图 8-42 所示。分别在这 3 个 IO 域的属性视图“常规”对话框中，设置模式为“输入”、“输出”和“输入/输出”，设置输入和输出域的过程变量为“变量_1”，

格式类型为“十进制”，并经输出域的“移动小数点”设置为 2 位，设置输入/输出域的格式类型为“字符串”，过程变量为“变量_3”，“字符串域长度”设置为 8 个字符，如图 8-43 所示。

图 8-42　IO 域的组态

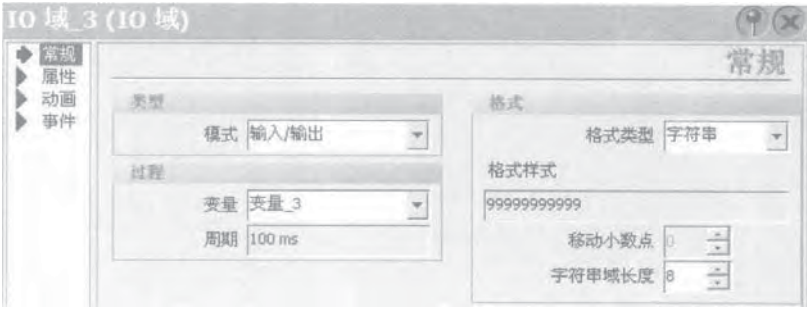


图 8-43　IO 域的常规属性组态

启动模拟运行后，用鼠标单击输入域，就会出现数值输入键盘（如图 8-44 所示），输入所需数据后，单击 ↵ 键，完成操作，如 1234。此时可看到输出域中显示的数字为 12.34（输出域与输入域是同一个变量，且输出数字的小数点后有 2 位）。用鼠标单击输入/输出域，则弹出字符输入键盘（如图 8-45 所示），输入内容后，同样通过 ↵ 键完成操作。

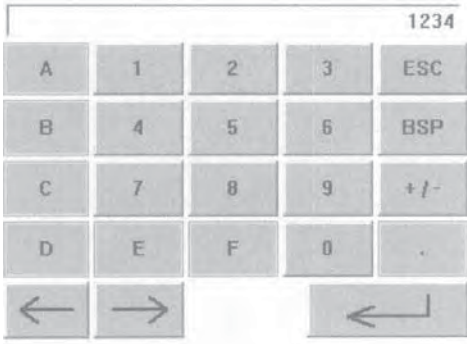


图 8-44　数字输入键盘

图 8-45　字符输入键盘

实例分析

由于 IO 域既可以组态为输出域，又可以组态为输入域，还可以组态为输入/输出域，因此在使用时非常灵活，可根据需要自由选择。

思考题

1. 什么是内部变量？什么是外部变量？如何在项目中添加变量？
2. 什么样的组态具有点动功能的按钮？
3. 怎样采用文本显示方式组态开关？
4. 怎样打开已有的库？怎样使用库中的对象？
5. 棒图有什么特点和作用？如何组态棒图？
6. 用输出域将变量中 5 位整数显示为 3 位整数和 2 位小数，如何实现？
7. 触摸屏组态完成后，如何实现模拟调试？

第 9 章　物料混合控制系统

- 物料混合控制系统简介
- PLC 程序设计
- 触摸屏画面设计

9.1 物料混合控制系统简介

9.1.1 系统工艺过程概述

在玻璃、水泥等流程性工业生产线上，生产原料往往是通过多种颗粒或粉末状的物料混合后，经过相应的处理，最终加工成为成品的。对物料的混合控制，是这类工业必不可少的重要环节。

现有一物料控制系统，要求将 3 种颗粒状物料按照一定的比例混合成加工原料。每种物料分别放置在一个料仓内，料仓下面有可以通过电磁阀控制开闭的插板阀，每种料的投放量通过插板阀打开的时间 T_L 进行控制，T_L 可由下式确定，其中物料投放流量由实验预先测定。

$$T_L = \frac{\text{所需投料量}}{\text{物料投放流量}}$$

将各个料仓的料投放到混合仓内。3 种料全部投放完毕后，混合仓内的搅拌器开始工作，将物料混合均匀，搅拌器的工作也是通过定时控制完成的。在混合仓底部也有可通过电磁阀控制的插板阀，用于将混合均匀的物料排放出去。物料系统工艺布局如图 9-1 所示。

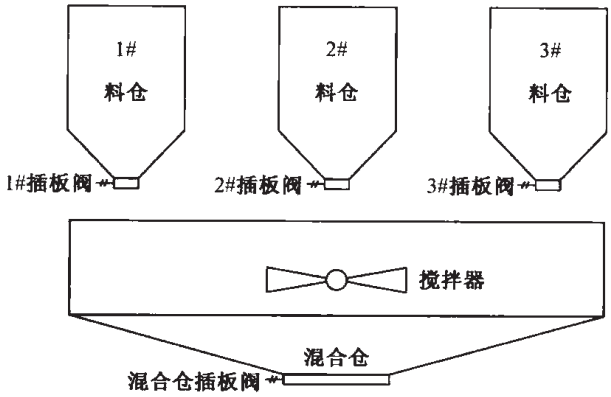


图 9-1 物料系统工艺布局示意图

系统采用 PLC 控制，并通过触摸屏设置参数与监控系统的运行状态。

9.1.2 PLC 系统选型

由于要使用配方功能，选用 S7-200 CPU226 机型。PLC 的 I/O 地址分配如表 9-1 所示。触摸屏与 PLC 的部分内部交换变量如表 9-2 所示。

表 9-1 PLC I/O 地址分配表

名 称	地址	名 称	地址	名 称	地址	名 称	地址
自动/手动开关	I0.0	混合仓放料按钮	I0.5	自动运行指示灯	Q0.0	1#仓放料	Q0.4
自动启动按钮	I0.1	1#仓放料按钮	I1.1	报警指示灯	Q0.1	2#仓放料	Q0.5
停止按钮	I0.2	2#仓放料按钮	I1.2	混合仓放料	Q0.2	1#仓放料	Q0.6
外部故障	I0.3	3#仓放料按钮	I1.3	搅拌器运行	Q0.3		
搅拌器按钮	I0.4						

表 9-2　PLC 内部地址分配表（部分）

名　称	地址	名　称	地址	名　称	地址	名　称	地址
混合仓放料按钮*	M0.0	1#料重量设置*	VD0	1#料重量显示*	VD240	混合仓放料指示*	M1.0
1#仓放料按钮*	M0.1	2#料重量设置*	VD4	2#料重量显示*	VD244	1#仓放料指示*	M1.1
2#仓放料按钮*	M0.2	3#料重量设置*	VD8	3#料重量显示*	VD248	2#仓放料指示*	M1.2
3#仓放料按钮*	M0.3	1#物料投放流量*	VD300	1#料放料时间	VD330	3#仓放料指示*	M1.3
搅拌器按钮*	M0.4	2#物料投放流量*	VD304	2#料放料时间	VD334	搅拌器运行指示*	M1.4
启动按钮*	M0.5	3#物料投放流量*	VD308	3#料放料时间	VD338	搅拌时间设定*	VD342
停止按钮*	M0.6	1#料计时器	T37	混合仓放料时间	VD346	搅拌计时器	T41
混合仓放料流量*	VD312	2#料计时器	T38	手动放料时间设置*	VD400	混合仓放料时间*	VD346
混合仓物料重量*	VD12	3#料计时器	T39	手动放料计时器	T40	混合仓放料计时器	T42

表 9-2 中带*地址是与触摸屏相连接的变量地址。在 STEP 7-Micro/WIN 中生成变量表，如图 9-2 所示。

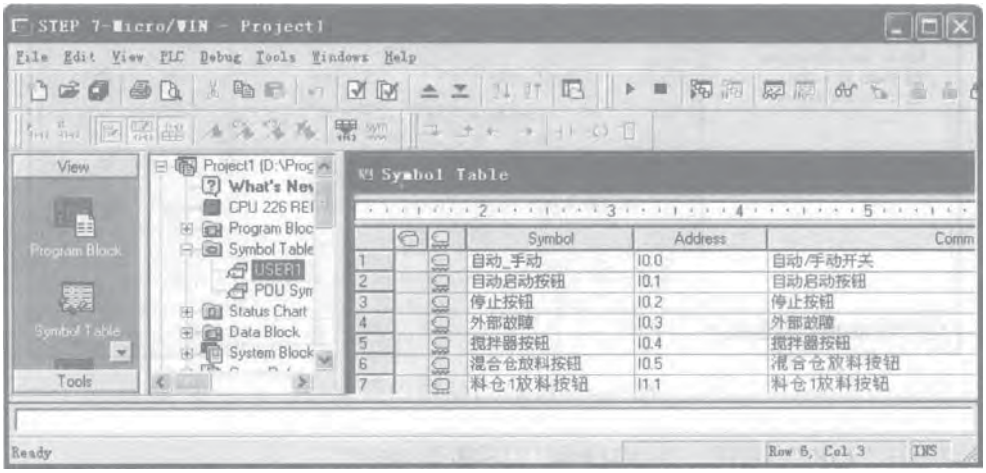


图 9-2　PLC 变量表

9.1.3　触摸屏选型

S7-200 PLC 可以与多种型号的触摸屏连接，在这里选用 TP170B 作为人机界面。采用第 8 章所述方法在 WinCC flexible 中建立触摸屏组态项目，并生成变量表如图 9-3 所示。

名称	数据类型	地址	数组计数	采集周期	注释
混合仓放料按钮	Bool	M 0.0	1	100 ms	
1# 仓放料按钮	Bool	M 0.1	1	100 ms	
2# 仓放料按钮	Bool	M 0.2	1	100 ms	
3# 仓放料按钮	Bool	M 0.3	1	100 ms	
搅拌器按钮	Bool	M 0.4	1	100 ms	
启动按钮	Bool	M 0.5	1	100 ms	
停止按钮	Bool	M 0.6	1	100 ms	
混合仓放料指示	Bool	M 1.0	1	100 ms	
1# 仓放料指示	Bool	M 1.1	1	100 ms	
2# 仓放料指示	Bool	M 1.2	1	100 ms	
3# 仓放料指示	Bool	M 1.3	1	100 ms	

图 9-3　触摸屏变量表

9.1.4 PLC 与触摸屏的连接

PLC 与触摸屏之间是以通信方式连接的。因此必须经过适当的设置，保证通信的顺利进行。连接设置分为 PLC 设置和触摸屏设置两个方面。

1. S7-200 PLC 的通信设置

在 STEP 7-Micro/WIN 对 S7-200 PLC 通信端口进行设置，将 port0 口的“PLC Address”设置成 2，将“Baud Rate”设置为 187.5kbps，如图 9-4 所示。

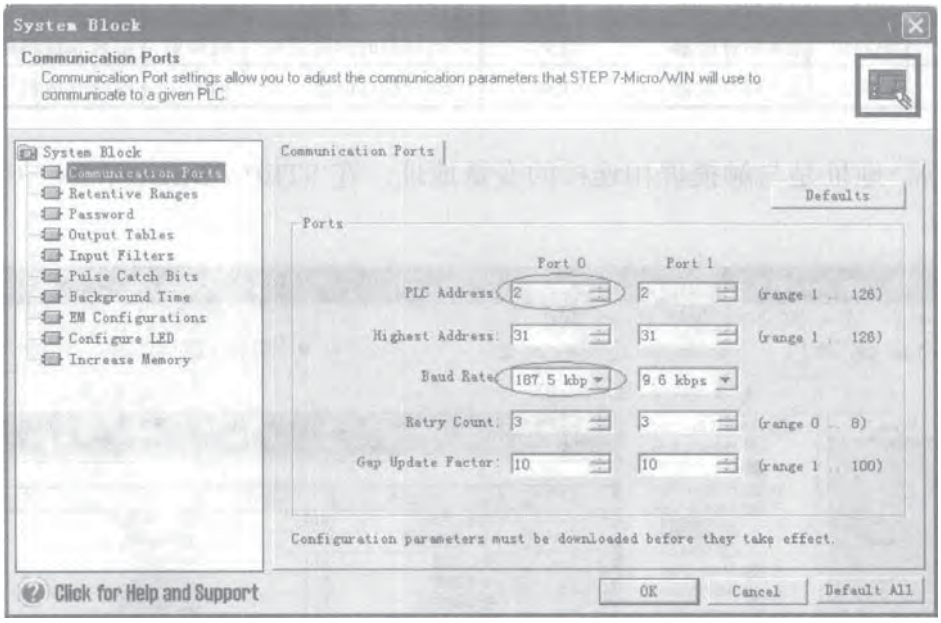


图 9-4 PLC 通信端口设置

完成后，单击“OK”按钮，并将此配置下载到 PLC。

2. 触摸屏的通信设置

在 WinCC flexible 项目菜单里双击“连接”图标，在新建的“连接_1”的通信驱动程序里选择“SIMATIC S7-200”，同时在“连接”表屏幕下方设定界面中将 TP170B 的地址设置为 1，波特率为 187500；网络配置为 MPI，并将 S7-200 的地址设定为 2（注意此参数一定要与上节 PLC 设定一致），如图 9-5 所示。

图 9-5 触摸屏通信端口设置

将此设定存盘并下载至 TP170B 即可与 S7-200 进行通信了。

9.2　PLC 程序设计

PLC 程序设计是非常细致的一项工作，本节仅给出程序设计的一些思路和部分程序。

1. 自动控制程序的设计

混合物料控制系统的自动控制程序属于典型的顺序控制程序，可以采用顺序控制设计法进行设计。在程序实现上可采用顺序功能图。

2. 混合仓物料总重量与单个物料放料量的计算

物料的投放是通过时间来控制的。由于每种物料的放料流量不同，各料仓的放料量可以通过自己的定时器当前值与放料流量的乘积来表示。混合仓的初始物料重量为零，3 个料仓按顺序依次放料，所以混合仓的物料重量按照前一物料重量加上当前放料重量计算得到。

3. 配方功能的实现

S7-200 的配方数据和实时采集数据保存在 64KB 或 256KB 的 EEPROM 存储卡中，存储卡插在 CPU 模块的插槽中。

用户程序可以调用由配方向导生成的读/写配方子程序，将指定的配方读入 CPU 模块中预设的存储区，或者将修改后的某一配方值写入存储卡中。操作人员可以通过触摸屏来选择需要的配方。

在 STEP-7 Micro/WIN 编程软件中（V4.0 以上版本），执行菜单命令“Tools（工具）”→“Recipe Wizard（配方向导）”，如图 9-6 所示。按照向导的提示依次向下执行。在图 9-7 所示的配方数据域定义窗口中，设置所涉及的数据项目。在本例中，分别是 3 个料仓的投料量配方值、搅拌时间设定值和混合仓放料时间设定值。在表中设置这几个数据项目的数据类型、默认值等内容。完成后单击“Next”按钮进入“生成与编辑配方”界面，如图 9-8 所示。

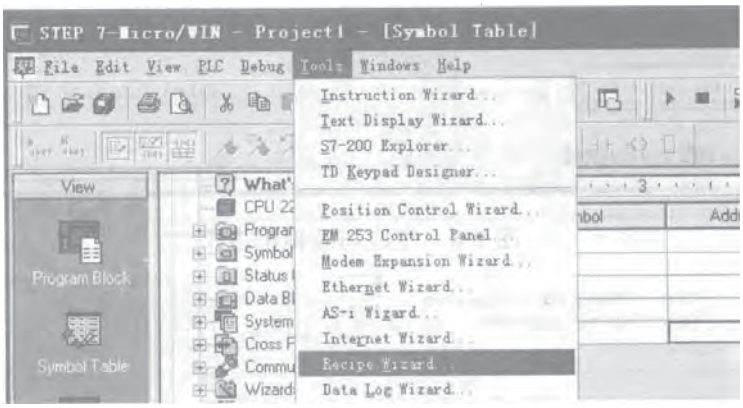


图 9-6　配方向导指令

在如图 9-8 所示的界面中，通过“Add Recipe(s)”操作键可以添加新的配方，并设置配方中各数据项的具体数值。在接下来的界面中设置数据项的存储位置，配方数据项只能存储于 V 型存储区中，这里设定好初始地址就可以了，如图 9-9 所示。组态完成后系统自动在 V

区中生成用于配方的符号表，如图 9-10 所示，这些地址在用户符号表中不能再出现。

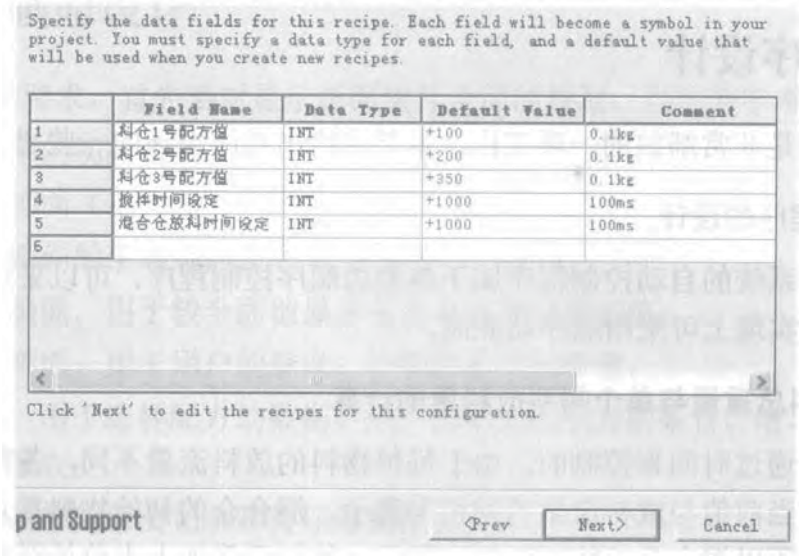


图 9-7　配方数据项设置

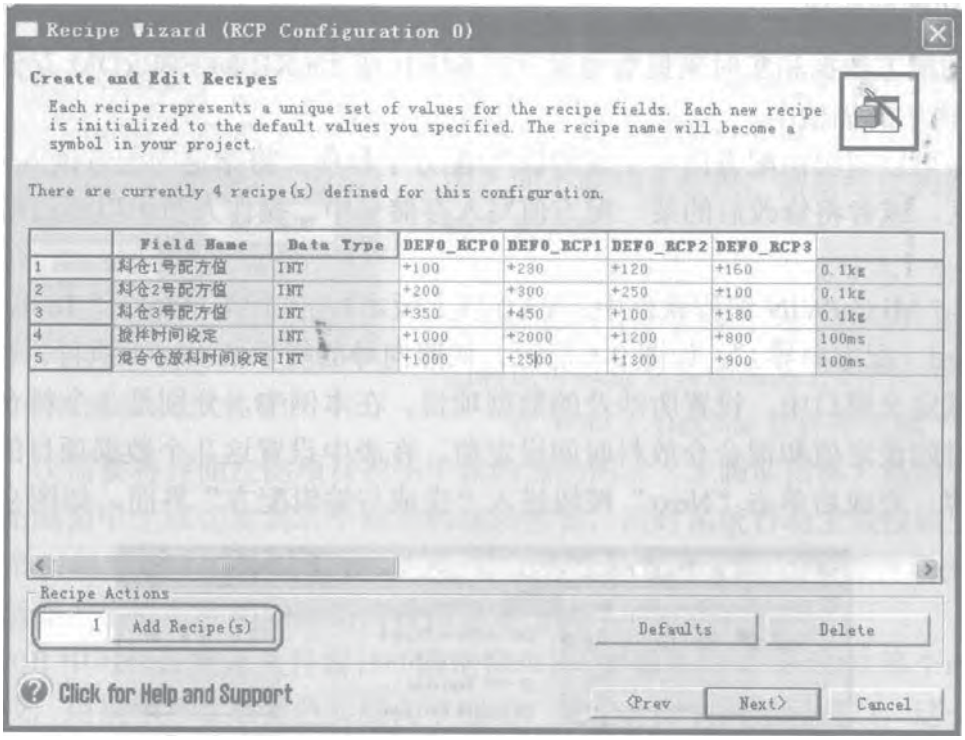


图 9-8　生成与编辑配方

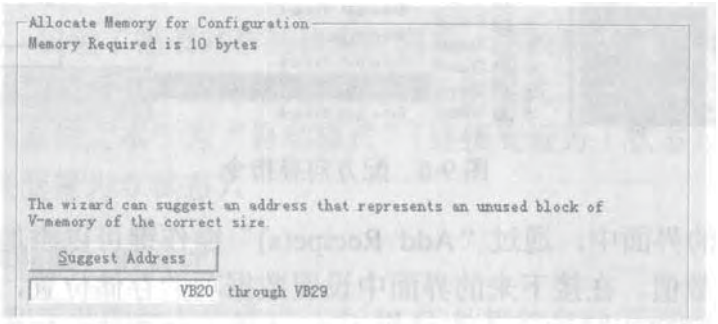


图 9-9　配方数据项地址设置

		Symbol	Address	Comment
1		DEF0_RCP0	0	
2		DEF0_RCP1	1	
3		DEF0_RCP2	2	
4		DEF0_RCP3	3	
5		料仓1号配方值	VW20	0.1kg
6		料仓2号配方值	VW22	0.1kg
7		料仓3号配方值	VW24	0.1kg
8		搅拌时间设定	VW26	100ms
9		混合仓放料时间设定	VW28	100ms

图 9-10　用于配方的符号表

配方向导设置完成后，在 STEP 7-Micro/WIN 软件主界面左侧的指令树“Instructions/ call Subroutines”文件夹下可以看到系统自动创建的读/写配方子程序图标。其中读配方指令 RCP0_Read 将某个配方从存储卡传送到预设的 V 存储区，指令框中 Rcp 引脚对应的输入值是配方的编号，数据类型为字型。写配方指令 RCP0_Write 用指定 V 区中预设的配方数据代替存储卡中某套配方中的数据。读、写配方子程序的使用如图 9-11 所示。

图 9-11　读/写配方程序

4. 程序示例——手动程序的设计

图 9-12 是 1 号料仓手动投料的部分程序。

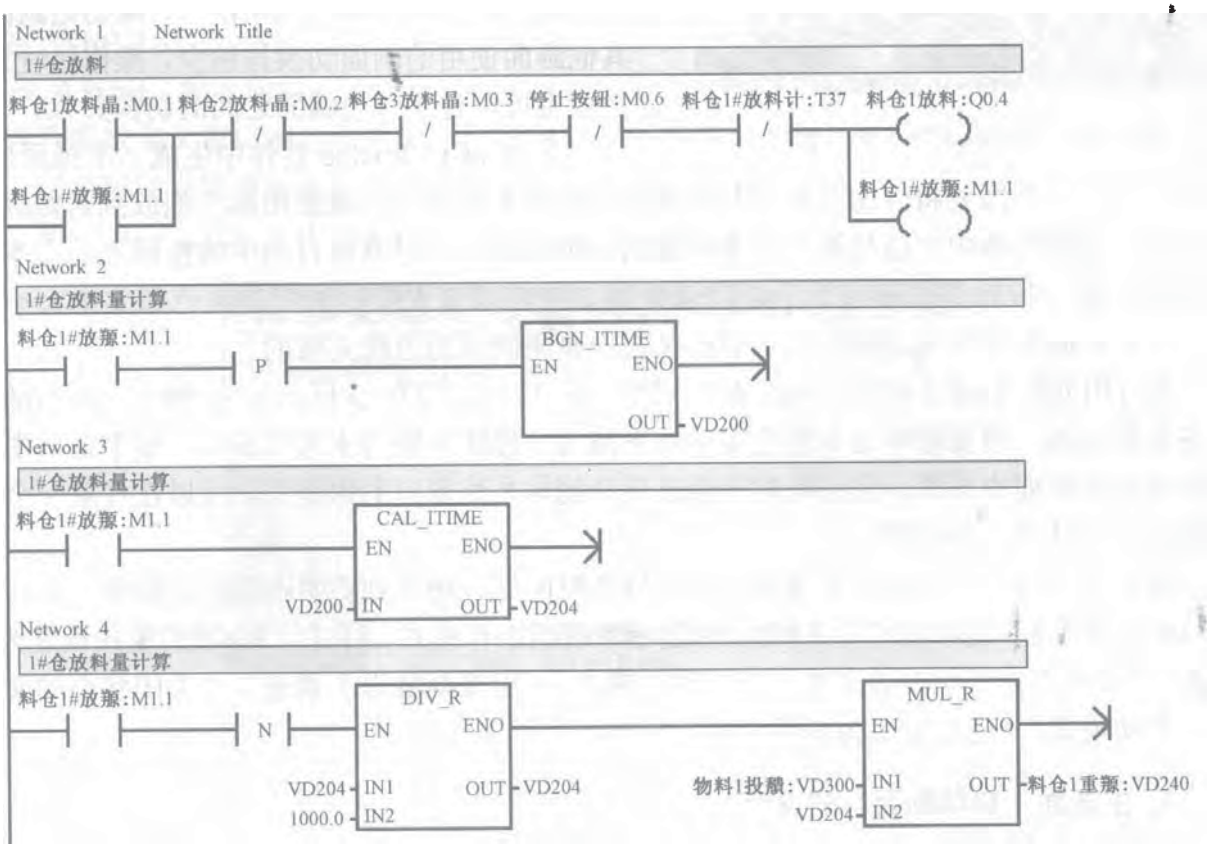


图 9-12　1#料仓手动投料程序

9.3 触摸屏画面设计

根据工艺操作要求，首先要对显示画面进行全面的规划，在本例中考虑设置如下画面：

- 开机时显示的初始画面；
- 自动运行画面（主画面）；
- 进行手动操作的手动画面；
- 设备状态画面，用于较全面地显示各设备主要变量的值；
- 用户管理画面，用于用户的登录、注销和用户的管理；
- 配方画面，用于选择配方的数据，用户可以修改配方的条目，增、减配方数据记录，打印配方报表；
- 用于查看报警的历史记录和打印报警报表。

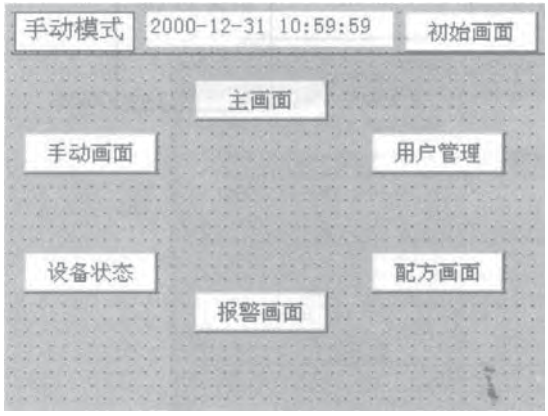


图 9-13 初始画面与永久性窗口

由于画面不多，以初始画面为中心，采用“单线联系”的星形切换方式，开机后显示初始画面，在初始画面中设置切换到其他画面的切换按钮（如图 9-13 所示），从初始画面可以切换到所有其他画面，其他画面只能返回到初始画面。其他画面之间不能相互切换，需要经过初始画面的“中转”来完成。

这种画面组织方式的层次少，除初始画面外，其他画面使用的画面切换按钮少，操作比较方便。当然也可以建立其他画面之间的切换关系。

在 WinCC flexible 软件中生成一个画面（如主画面）之后，仅需要将界面左侧项目视图中该画面的图标（主画面图标）拖放到初始画面，就可以在初始画面中生成切换到这个画面的切换按钮，同时系统自动生成按钮上的文本。可以用鼠标调节按钮的位置和大小，在属性视图中可以设置按钮的背景颜色和文本大小等属性。在其他画面上生成返回初始画面的按钮也是用同样的方法完成的。

在 TP170B 中可以设定永久性窗口（固定窗口）（如图 9-13 所示）。在每个画面顶部都有一条黑色实线，用鼠标将这条黑色实线向下拖动，黑线上面为永久性窗口，这个窗口中的对象将在所有画面中出现，且运行时不会出现分割永久性窗口的水平线。可以在任意一个画面中修改永久性窗口的对象。

在永久性窗口中，放置了需要共享的日期时间域、切换到初始画面的按钮和一个与变量“自动/手动开关”连接的符号 IO 域，以显示系统的工作模式。将符号 IO 域的模式设置为“双状态”，其中“打开状态的文本”为“自动模式”（连接变量为 1 状态），“关闭状态的文本”为“手动模式”（连接变量为 0 状态）。

1. 主画面（自动画面）的设计

控制系统有自动和手动两种工作模式，由 PLC 外接的自动/手动开关来选择。由于自动运行画面使用的最多、最频繁，因此将主画面用于监控自动工作模式的运行。

开机后进入初始画面，单击“主画面”按钮，进入主画面（如图 9-14 所示）。画面中给出了物料系统的示意图，用图形方式（白色和黑色）显示各仓插板阀的通断状态，用棒图动画方式显示混合仓内物料的高度。画面中还显示了系统工作过程中各种物料的重量，放料和搅拌的运行时间，并设置了启动和停止自动运行按钮。

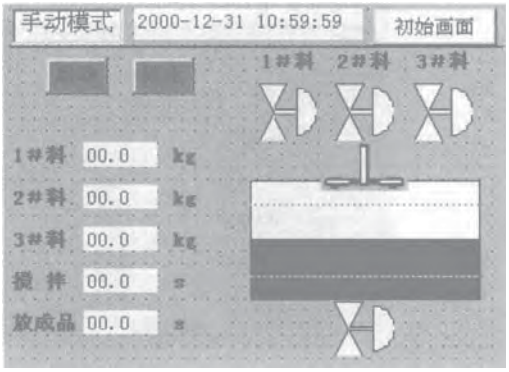


图 9-14　主画面

主画面中数据显示均采用 IO 域，且置为输出模式。阀的状态采用图形 IO 域，模式设置为“双状态”。显示用的图形为 WinCC flexible 软件提供的图形。为了表明阀门的开关状态，使用 Visio 软件对图形进行了处理，以使得开/关两种状态下，阀门的颜色发生变化。

启动自动运行的条件为：系统处于自动模式；各插板阀关闭，搅拌器电动机停止；混合仓中没有物料。

自动模式的工作过程为：单击主画面中的启动按钮或 PLC 外接按钮，1＃料仓插板阀接通（插板阀中间部分变为灰色）。进料达到配方设定值时，1＃仓停止放料，自动改为 2＃料仓放料，直到进完所有的物料，然后启动混合仓内的搅拌器，对混合仓内的物料进行搅拌。经过设定的搅拌时间之后，搅拌器停止，混合仓底部插板阀打开，放出混合好的物料。在到达排料设定时间之后，关闭插板阀，开始下一轮进料。

单击触摸屏上的“停止”按钮或 PLC 外部停止按钮后，系统进入正常停机过程，即将正在执行的物料排放过程全部完成之后，系统才真正停止工作。

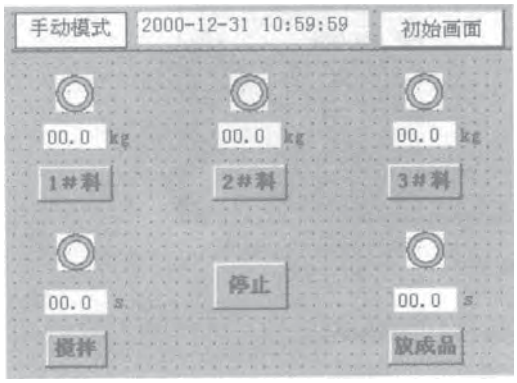


图 9-15　手动画面

2. 手动画面的设计

系统运行时在初始画面中单击“手动画面”按钮，就可以进入手动画面（如图 9-15 所示）。但要手动画面中的操作按钮起作用，必须将 PLC 外接的“自动/手动”开关放置在手动位置。此时，永久性窗口中符号 IO 域显示“手动模式”。

在手动模式下，手动画面上的按钮与 PLC 的外接手动操作按钮的作用是等价的。各设备的运行状态通过指示灯表示。画面上还有 IO 域显示手动操作过程中的设备参数值。

3. 配方画面的设计

配方画面与 S7-200 配方向导所生成的读/写配方子程序配合使用，可以很方便的对物料混合系统的物料配方进行显示与修改。配方画面如图 9-16 所示。配方编号通过符号 IO 域来选择。符号 IO 域为输入/输出模式，显示文本列表“配方编号选择”中的内容（如图 9-17 所示）。符号 IO 域的组态如图 9-18 所示。操作时，以下拉菜单选择的方式选择具体的配方编号，并将变量“配方编号”写入相应的值。例如，用符号 IO 域选择“1＃配方”，则“配方变量”

中的值为 0，与图 9-10 中 PLC 配方符号表相一致。

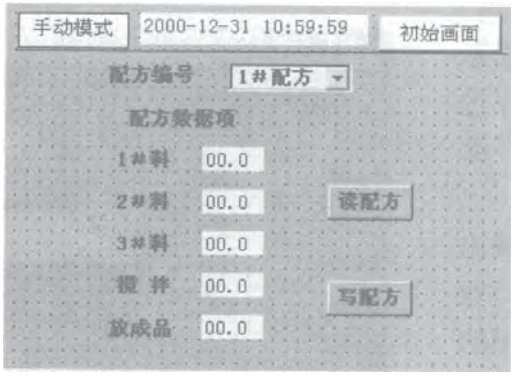


图 9-16　配方画面

图 9-17　文本列表

在画面中还将配方数据项以 IO 域的方式列出，IO 域的工作模式均为“输入/输出”模式。

用符号 IO 域选择好配方编号以后。单击“读配方”按钮，则 PLC 执行“RCP0_READ”命令，将 PLC 存储卡中被选中的配方参数读入 PLC 的 VW20～VW28 中，同时，通过通信，在“配方画面”中将这些参数显示出来。

若要改写某个配方，则通过符号 IO 域选择该配方编号，通过画面上配方数据项中各对应 IO 域写入各配方数据。这些数据直接由通信写入 PLC 存储区 VW20～VW28 中。单击“写配方”按钮，执行指令“RCP0_WRITE”，修改后的配方参数写入 PLC 存储卡内由变量“配方编号”指定的配方中。

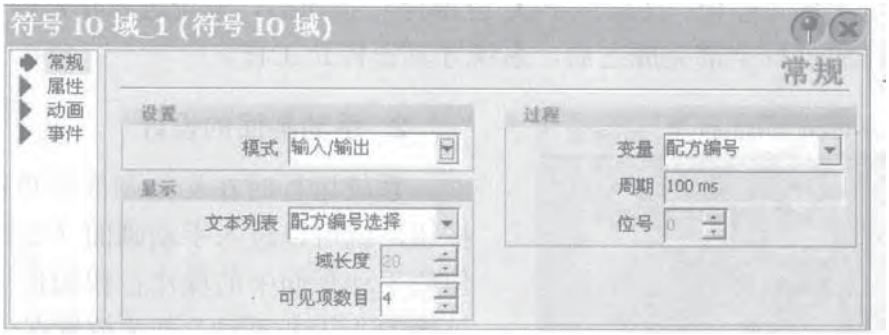


图 9-18　符号 IO 域组态

思考题

1. 怎样用 TP 170B 触摸屏和 S7-200 PLC 实现配方功能？
2. TP 170B 触摸屏和 S7-200 PLC 之间如何实现通信？
3. 如何实现画面上的永久性窗口，并对其中的对象进行修改？

附录A 特殊寄存器（SM）标志位

表A-1 状态位（SMB0）

SM位	描 述
SM0.0	CPU运行时，该位始终为1
SM0.1	该位在首次扫描时为1
SM0.2	若保持数据丢失，则该位在一个扫描周期中为1，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，该位将接通一个扫描周期
SM0.4	该位提供周期为1 min，占空比位30 s的时钟脉冲
SM0.5	该位提供周期为1 s，占空比为50%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）。在RUN位置时，该位可使自由端口通信方式有效；在TERM位置时，可与编程设备正常通信

表A-2 状态位（SMB1）

SM位	描 述
SM1.0	零标志，执行某些指令的结果为0时，该位置1
SM1.1	错误标志，执行某些指令的结果溢出或查处非法数值时，该位置1
SM1.2	负数标志，执行数学运算的结果为负数时，该位置1
SM1.3	试图除以0时．该位置1
SM1.4	当执行ATT（Add to Table）指令时，试图超出表范围时，该位置1
SM1.5	当执行LIFO或FIFO指令时，试图从空表中读数据时，该位置1
SM1.6	当试图把一个非BCD换为二进制数值时，该位置1
SM1.7	当ASCⅡ码不能转换为有效数字的十六进制数时，该位置1

表A-3 自由端口接收字符缓冲区（SMB2）

SM位	描 述
SMB2	在自由端口通信方式下，该字符存储从端口0或1接收到的每一个字符

表A-4 自由端口奇偶校验错误（SMB3）

SM位	描 述
SMB3.0	端口0或1的奇偶校验错误（0=无错；1=有错）
SMB3.1～SMB3.7	保留

注意：SMB2和SMB3与端口0和1公用，当端口0接收到的字符并使得与该事件（中断事件8）相连的中断程序执行时，SMB2包含端口0接收到的字符，而SMB3含该字符的

校验位状态。当端口 10 接收到的字符并使得与该事件（中断事件 25）相连的中断程序执行时，SMB2 包含端口 1 接收到的字符，而 SM3 包含该字符的校验位状态。

表 A-5 中断允许、队列溢出、发送空闲标志位（SMB4）

SM 位	描 述
SM4.0	当通信中断队列溢出时，该位置 1
SM4.1	当输入中断队列溢出时，该位置 1
SM4.2	当定时终端队列溢出时，该位置 1
SM4.3	当运行时刻，发现编程问题时，该位置 1
SM4.4	该位指示全局中断允许位，当允许中断时，该位置 1
SM4.5	当（端口 0）发送空闲时，该位置 1
SM4.6	当（端口 1）发送空闲时，该位置 1
SM4.7	发生强制置位时，该位置 1

注意：只有在终端程序里，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。

表 A-6 I/O 错误状态位（SMB5）

SM 位	描 述
SM5.0	当有 I/O 错误时，该位置 1
SM5.1	当有 I/O 总线上连接了过多的数字量 I/O 点时，该位置 1
SM5.2	当有 I/O 总线上连接了过多的模拟量 I/O 点时，该位置 1
SM5.3	当有 I/O 总线上连接了过多的智能量 I/O 点时，该位置 1
SM5.4～SM5.5	保留

表 A-7 CPU 识别（ID）寄存器（SMB6）

SM 位	描 述
格 式	MSB LSB 7 0 CPU ID 寄存器 x x x x r r r r
SM6.0～SM6.3	保留
SM6.4～SM6.7	xxxx=0000CPU222 0010CPU224 0110CPU221 1001CPU226/CPU226XM

表 A-8 I/O 模块识别和错误寄存器（SMB8～SMB21）

SM 位	描 述
格式	偶数字节：模块 ID 寄存器 MSB LSB 0 m t t a i i q q m：模块存在 0=有模块；1=无模块 tt：模块类型 00=非智能模块；01=智能模块；10、11=保留 a：I/O 类型 0=开关量；1=模拟量 ii：输入 00=无输入；01=2AI 或 8DI；10=4AI 或 16DI；11=8AI 或 32DI qq：输出 00=无输出；01=2AQ 或 8DQ；10=4AQ 或 16DQ；11=8AQ 或 32DQ 奇数字节：模块 ID 寄存器 MSB LSB 7 0 c 0 0 b r p f t c：配置错误 0=无错误；1=错误 b：总线错误或校验错误 r：超范围误差 p：无用户电源错误 f：熔断器错误 t：端子松动错误
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

表 A-9 扫描时间寄存器（SMW22～SMW26）

SM 位	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

表 A-10 模拟电位器（SMB28～SMB29）

SM 位	描 述（只读）
SMB28	存储模拟调节器 0 的输入值。在 STOP/RUN 方式下，每次扫描时更新该值
SMB29	存储模拟调节器 1 的输入值。在 STOP/RUN 方式下，每次扫描时更新该值

表 A-11 自由端口控制寄存器（SMB30～SMB130）

端口 0	端口 1	描　述
SMB30 格式	SMB130 格式	自由端口模式控制字节 MSB　　LSB 7 6 5 4 3 2 1 0 p p d b b b m m
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=自由端口协议 10=PPI/主站模式；11=保留（默认是 PPI/从站模式） 注意：当选择 mm=10（PPI 主战），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 2～7 位。
SM30.2～SM30.4	SM130.2～SM130.4	bbb：自由口波特率 000=38 400 bps，　001=19 200 bps 010=9 600 bps，　011=4 800 bps 100=2 400 bps，　101=1 200 bps 110=115.2 kbps，111=57.6 kbps
SM30.5	SM130.5	d：每个字符的数据位 0=8 位/字符；1=7 位/字符
SM30.6 和 SM30.7	SM130.6 和 SM130.7	pp：校验选择 00=不校验；01=偶校验；10=不校验；11=奇校验

表 A-12 永久存储器 E^2PROM 写控制（SMB31 和 SMW32）

SM 位	描　述（只读）
格式	SMB31：软件命令 MSB　　LSB 7 6 5 4 3 2 1 0 c 0 0 0 0 0 s s SMW32：V 存储器地址 MSB　　LSB 15 14 13 12 11 10 9 8　7 6 5 4 3 2 1 0 V 存储器地址
SMB31.0 和 SMB31.1	ss:被存储数据类型 00=字节；10=字；01=字节；11=双字
SMB31.7	c：存入永久存储，0=无执行存储器操作的请求；1=用户程序申请向永久存储器存储数据，每次存储操作完成后，S7-200 复位该位
SMW32	SMW32 总是所存数据的 V 存储器地址，该值是相对于 V0 的偏移量。当执行存储命令时，把该数据存到永久存储器中的相应位置

表 A-13 定时中断的时间间隔寄存器（SMB34 和 SMB35）

SM 位	描　述
SMB34	定义定时中断 0 时间间隔（1～255 ms，以 1 ms 为增量）
SMB35	定义定时中断 1 时间间隔（1～255 ms，以 1 ms 为增量）

附录 B 错误代码信息

表 B-1 致命错误代码和信息

错误代码	描 述
0000	无致命错误
0001	用户程序校验和错误
0002	编译后梯形图程序校验和错误
0003	扫描看门狗超时错误
0004	内部 E^2PROM 错误
0005	内部 E^2PROM 用户程序校验和错误
0006	内部 E^2PROM 配置参数（SDB0）错误
0007	内部 E^2PROM 强制数据校验和错误
0008	内部 E^2PROM 默认输出表值校验和错误
0009	内部 E^2PROM 用户数据 DB1 校验和错误
000A	存储器卡失灵
000B	存储器卡用户程序校验和错误
000C	存储器卡配置参数（SDB0）校验和错误
000D	存储器卡强制数据校验和错误
000E	存储器卡默认输出表值校验和错误
000F	存储器卡上用户数据 DB1 校验和错误
0010	内部软件错误
0011	比较节点间接寻址错误
0012	比较节点非法值错误
0013	存储器卡空或者 CPU 不识别该程序
0014	比较节点范围错误

表 B-2 运行程序错误

错误代码	描 述
0000	无错误
0001	执行 HDEF 之前，HSC 未允许
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，试图执行 ENI、DISI 或 FDEF 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，通信端口 0 由另一条 XMT/RCV 指令执行
000A	在同一 HSC 执行时又企图用 HDEF 指令再定义该 HSC
000B	通信端口 1 上同时执行 XMT/RCV 指令

续表

错误代码	描　述
000C	时钟存储卡不存在
000D	重新定义已经使用的脉冲输出
000E	PTO 个数设为 0
000F	比较指令中的非法字符
0010	在当前 PTO 操作模式中，命令未允许
0011	非法 PTO 命令代码
0012	非法 PTO 包络表
0013	非法 PID 回路参数表
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端口模式
009B	非法指针（字符串操作中起始位置指定为 0）
009F	无存储卡或存储卡无响应

表 B-3　编译规则错误（非致命）

错误代码	描　述
0080	程序太大无法编译：须缩短程序
0081	堆栈溢出：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指令：加 MEND 或删去不正确指令
0084	保留
0085	无 FOR 肠指令：加条 FOR 指令或删除 NEXT 指令
0086	无 NEXT 指令：加条 NEXT 指令或删除 FOR 指令
0087	无标号（ LBL、INT、SBR）：加上合适标号
0088	无 RET 或子程序中有不允许指令：加条 RET 或删除不正确指令
0089	无 RETI 或中断程序中有不允许指令：加条 RETI 或删除不正确指令
008A	保留
008B	从/向一个 SCR 段的非法跳转
008C	标号重复（LSL、INT、SBR）：重新命名标号
008D	非法标号（LBL、INT、SBR）：确保标号数在允许范围内
0090	非法参数：确认指令所允许的参数
0091	范围错误（带地址信息）：检查操作数范围
0092	指令计数域错误（带计数信息）：确认最大计数范围
0093	FOR/NEXT 嵌套层数超出范围
0095	无 LSCR 指令（装载 SCR）
0096	无 SCRE 指令（SCR 指令结束）或 SCRE 前面有不允许的指令
0097	用户程序包含非法字编码的和数字编码的 EU/ED 指令
0098	在运行模式进行非法编辑（试图编辑非数字编码的 EU/ED 指令）
0099	隐含网络段太多（HIDE 指令）
009B	非法指针（字符串操作中起始位置指定为 0）
009C	超出最大指令长度
009D	SDB0 中检测到非法参数
009E	PCALL 字符太多
009F～00FF	保留

附录 C S7-200 可编程控制器指令集

表 C-1 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	IN1，IN2	装载字节比较的结果，IN1（x:<，<=，=，>=，>，<>）IN2
ABx	IN1，IN2	“与”字节比较的结果，IN1（x:<，<=，=，>=，>，<>）IN2
OBx	IN1，IN2	“或”字节比较的结果，IN1（x:<，<=，=，>=，>，<>）IN2
LDWx	IN1，IN2	装载字比较的结果，IN1（x:<，<=，=，>=，>，<>）IN2
AWx	IN1，IN2	“与”字比较的结果，IN1（x:<，<=，=，>=，>，<>）IN2
OWx	IN1，IN2	“或”字比较的结果，IN1（x:<，<=，=，>=，>，<>）IN2
LDDx	IN1，IN2	装载双字比较的结果，IN1（x:<，<=，=，>=，>，<>）IN2
ADx	IN1，IN2	“与”双字比较的结果，IN1（x:<，<=，=，>=，>，<>）IN2
ODx	IN1，IN2	“或”双字比较的结果，IN1（x:<，<=，=，>=，>，<>）IN2
LDRx	IN1，IN2	装载实数比较的结果，IN1（x:<，<=，=，>=，>，<>）IN2
ARx	IN1，IN2	“与”实数的比较结果，IN1（x:<，<=，=，>=，>，<>）IN2
ORx	IN1，IN2	“或”实数的比较结果，IN1（x:<，<=，=，>=，>，<>）IN2
NOT		栈顶值取反
EU		检测上升沿
ED		检测下降沿
=	N	赋值
=I	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

表 C-2 传送、移位、循环、填充和逻辑操作指令

MOVB IN, OUT	字节传送
MOVW IN, OUT	字传送
MOVD IN, OUT	双字传送
MOVR IN, OUT	实数传送
BIR IN, OUT	立即读取物理输入点字节
BIW IN, OUT	立即写物理输出点字节
BMB IN, OUT, N	字节块传送
BMW IN, OUT, N	字块传送
BMD IN, OUT, N	双字块传送
SWAP IN	交换字节
SHRB DATA, S_BIT, N	移位寄存器
SRB OUT, N	字节右移 N 位
SRW OUT, N	字右移 N 位
SRD OUT, N	双字右移 N 位
SLB OUT, N	字节左移 N 位
SLW OUT, N	字左移 N 位
SLD OUT, N	双字左移 N 位
RRB OUT, N	字节循环右移 N 位
RRW OUT, N	字循环右移 N 位
RRD OUT, N	双字循环右移 N 位
RLB OUT, N	字节循环左移 N 位
RLW OUT, N	字循环左移 N 位
RLD OUT, N	双字循环左移 N 位
FILL IN, OUT, N	用指定的元素填充存储器空间
ALD	触点组串联
OLD	触点组并联
LPS	逻辑入堆栈
LRD	逻辑读栈
LPP	逻辑出栈
LDS	装载堆栈
AENO	对 ENO 进行“与”操作
ANDB IN1, OUT	字节逻辑“与”
ANDW IN1, OUT	字逻辑“与”
ANDD IN1, OUT	双字逻辑“与”
ORB IN1, OUT	字节逻辑“或”
ORW IN1, OUT	字逻辑“或”
ORD IN1, OUT	双字逻辑“或”
XORB IN1, OUT	字节逻辑异“或”
XORW IN1, OUT	字逻辑异“或”
XORD IN1, OUT	双字逻辑异“或”
INVB OUT	字节取反
INVW OUT	字取反
INVD OUT	双字取反

表C-3 表、查找、转换、中断、通信和高速指令

ATT TABLE，DATA	把数据加到表中
LIFO TABLE，DATA	从表中取数据，后入先出从表中取数据
FIFO TABLE，DATA	从表中取数据，先入后出从表中取数据
FND= TBL，PATRN，INDX	在表TBL中查找等于比较条件PATRN的数据
FND<> TBL，PATRN，INDX	在表TBL中查找不等于比较条件PATRN的数据
FND< TBL，PATRN，INDX	在表TBL中查找小于比较条件PATRN的数据
FND> TBL，PATRN，INDX	在表TBL中查找大于比较条件PATRN的数据
BCDI OUT	BCD码转换成整数
IBCD OUT	整数转换成BCD码
BTI IN，OUT	字节转换成整数
ITB IN，OUT	整数转换成字节
ITD IN，OUT	整数转换成双整数
DTI IN，OUT	双整数转换成整数
DTR IN，OUT	双整数转换成实数
TRUNC IN，OUT	实数截位取整为双整数
ROUND IN，OUT	实数四舍五入为双整数
ATH IN，OUT，LEN	ASCⅡ码转换成十六进制数
HTA IN，OUT，LEN	十六进制数转换成ASCⅡ码
ITA IN，OUT，LEN	整数转换成ASCⅡ码
DTA IN，OUT，LEN	双整数转换成ASCⅡ码
RTA IN，OUT，LEN	实数转换成ASCⅡ码
DECO IN，OUT	译码
ENCO IN，OUT	编码
SEG IN，OUT	七段译码
CRETI	从中断条件返回
ENI	允许中断
DISI	禁止中断
ATCH INT，EVENT	建立中断事件与中断程序的连接
DTCH EVENT	解除中断事件与中断程序的连接
XMT TABLE，PORT	自由端口发送信息
RCV TABLE，PORT	自由端口接收信息
NETR TABLE，POPT	网络读
NETR TABLE，POPT	网络写
GPA ADDR，POPT	获取端口地址
SPA ADDR，POPT	设置端口地址
HDEF HSC，Mode	定义高速计数器模式
HDEF N	激活高速计数器
PLS Q	脉冲输出

表C-4 数学、增减指令

+I IN1，OUT	整数加法：IN1+OUT→OUT
+D IN1，OUT	双整数加法：IN1+OUT→OUT
+R IN1，OUT	实数加法：IN1+OUT→OUT
-I IN2，OUT	整数减法：OUT-IN2→OUT
-D IN2，OUT	双整数减法：OUT-IN2→OUT
-R IN2，OUT	实数减法：OUT-IN2→OUT

续表

MUL IN1，OUT	整数完全乘法
*I IN1，OUT	整数乘法：IN1*OUT→OUT
*D IN1，OUT	双整数乘法：IN1*OUT→OUT
*R IN1，OUT	实数乘法：IN1*OUT→OUT
DIV IN2，OUT	整数完全除法
/I IN2，OUT	整数除法：OUT/IN2→OUT
/D IN2，OUT	双整数除法：OUT/IN2→OUT
/R IN2，OUT	实数除法：OUT/IN2→OUT
SQRT IN，OUT	平方根
LN IN，OUT	自然对数
EXP IN，OUT	自然指数
SIN IN，OUT	正弦
COS IN，OUT	余弦
TAN IN，OUT	正切
INCB OUT	字节加 1
INCW OUT	字加 1
INCD OUT	双字加 1
DE OUT	字节减 1
DECW OUT	字减 1
DECD OUT	双字减 1
PID Table，Loop	PID 回路
定时器和计数器	
TON Txxx，PT	接通延时定时器
TOF Txxx，PT	断开延时定时器
TONR Txxx，PT	有记忆接通延时定时器
CTU Cxxx，PV	增计数器
CTD Cxxx，PV	减计数器
CTUD Cxxx，PV	增/减计数器
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	定时器监视（看门狗）复位（300 ms）
JMP N	跳到定义的符号
LBL N	定义一个跳转的符号
CALL N[N1，…]	调用子程序[N1，…]
CRET	从子程序条件返回
FOR INDX，INIT NEXT FINAL	For/Next 循环
LSCR N	顺控继电器段的启动
SCRT N	顺控继电器段的转换
SCRE	顺控继电器段的结束

参 考 文 献

[1] 西门子（中国）有限公司自动化与驱动集团．SIMATIC S7-200 可编程控制器系统手册．2004

[2] 西门子（中国）有限公司自动化与驱动集团．SIMATIC S7-200CN 选型手册．2006

[3] 西门子（中国）有限公司自动化与驱动集团．SIMATIC S7-200 系列产品外形图．2007

[4] 西门子（中国）有限公司自动化与驱动集团．SIEMENS AG 定位模板快速入门．2005

[5] 西门子（中国）有限公司自动化与驱动集团．SIEMENS AG SIMATIC Panels 致力于满足于您的各种需求产品样本．2008

[6] 西门子（中国）有限公司自动化与驱动集团．SIEMENS AG WinCC flexible 2007 压缩版/标准版/高级版用户手册．2007

[7] 西门子（中国）有限公司自动化与驱动集团．SIEMENS AG WinCC flexible 2007 移植用户手册．2007

[8] 西门子（中国）有限公司自动化与驱动集团．SIEMENS AG WinCC flexible 2007 Runtime 用户手册．2007

[9] 西门子（中国）有限公司自动化与驱动集团．SIEMENS AG STEP 7 V5.4 编程参考手册．2006

[10] 西门子（中国）有限公司自动化与驱动集团．SIEMENS AG S7-200CN 可编程序控制器产品样本．2008

[11] 西门子（中国）有限公司自动化与驱动集团．SIEMENS AG S7-200 可编程控制器系统手册．2007

[12] 杨后川，张学民，陈勇编著．SIMATIC S7-200 可编程控制器原理与应用．2008

[13] 杨后川、胡进、陈勇、丁丽丽编著．机电一体化控制技术与系统．北京：蓝天出版社，2008

[14] 杨后川、梁炜编著．机床数控技术及应用．北京：北京大学出版社，2005

[15] 杨后川、高昆、陈勇．某型飞机起落架收放作动筒液压测试系统 PLC 控制设计．机电产品开发与创新 2007 第 2 期

[16] 杨后川、冯春晓、陈勇．实验用气动机械手 PLC 控制设计．机电产品开发与创新 2009 第 2 期

[17] 杨后川、夏成宝等．U 形折板机的 PLC 控制设计．机电产品开发与创新 2009 第 3 期

[18] 杨后川等．基于 FX_{2N}PLC 控制的实验用气动机械手设计．液压与气动 2009 第 2 期

[19] 高昆、丁丽丽、胡进．某型导弹测试机架的 PLC 控制．兵工自动化 2006 第 1 期

[20] 廖常初主编．S7-200PLC 编程及应用．北京：机械工业出版社，2007

[21] 吴中俊，黄永红主编．可编程控制器原理及应用．北京：机械工业出版社，2003

[22] 贾德胜编著．PLC 应用开发实用子程序．北京：人民邮电出版社，2006

[23] 刘洪涛，黄 海编著．PLC 应用开发从基础到实践．北京：电子工业出版社，2001

[24] 宫淑贞，王冬青，徐世许编著．可编程控制器原理及应用．北京：人民邮电出版社，2006

[25] 宋伯生编著．PLC 编程实用指南．北京：机械工业出版社，2007

[26] 汪晓平等编著．PLC 可编程控制器系统开发实例导航．北京：人民邮电出版社，2004

[27] 张运刚，宋小春编著．从入门到精通——西门子工业网络通信实战．北京：人民邮电出版社，2007

[28] 求是科技．PLC 应用开发技术与工程实践．北京：人民邮电出版社，2005

[29] 廖常初主编，陈晓英副主编．西门子人机界面（触摸屏）组态与应用技术(第 2 版)．.北京：机械工业出版社，2008

反侵权盗版声明

举报电话：（010）88254396；（010）88258888
传　　真：（010）88254397
E-mail：　dbqq@phei.com.cn
通信地址：北京市万寿路 173 信箱
　　电子工业出版社总编办公室
邮　　编：100036