MCU

51系列单片机设计实例

楼然苗 李光飞 编著

北京航空航天大学出版社
http://www.buaapress.com.cn

封面设计：艺铭设计

ISBN 7-81077-268-6

ISBN 7-81077-268-6

定价：29.50元（附光盘）

51 系列单片机设计实例

楼然苗　李光飞　编著

北京航空航天大学出版社
http://www.buaapress.com.cn

内 容 简 介

本书是为希望掌握单片机设计应用技术的电子爱好者而编著的。本书除简要地介绍 51 系列单片机的硬件资源及指令外，重点列举了 13 个实际应用设计实例。文中对实例的硬件电路原理、软件设计的思路及功能模块进行了详细的介绍，并给出了完整的源程序及注释，这对单片机初学者迅速理解单片机的设计应用原理具有很好的效果。读者可以参考给出的硬件电路及源程序进行实验设计练习，从而逐步掌握具体应用系统的设计方法。本书含有光盘 1 张，包含书中所有的应用源程序。

本书可作为单片机设计与应用技术人员的参考用书，也是电子设计爱好者自学单片机应用技术难得的学习用书。

图书在版编目(CIP)数据

51 系列单片机设计实例/楼然苗等编著. —北京：北京航空航天大学出版社，2003.3

ISBN 7-81077-268-6

Ⅰ. M… Ⅱ. 楼… Ⅲ. 单片微型计算机，51 系列—程序设计 Ⅳ. TP368.1

中国版本图书馆 CIP 数据核字(2002)第 092536 号

51 系列单片机设计实例

楼然苗 李光飞 编著

责任编辑 孔祥燮

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(100083) 发行部电话：(010)82317024 传真：(010)82328026

http://www.buaapress.com.cn

E-mail:bhpress@263.net

北京市云西华都印刷厂印装 各地书店经销

*

开本：787×1 092 印张：16.5 字数：416 千字

2003 年 3 月第 1 版 2003 年 3 月第 1 次印刷 印数：5 000 册

ISBN 7-81077-268-6 定价：29.50 元

前　言

单片机的实际应用一直是单片机初学者难以掌握的问题。本书结合13个作者编制的单片机应用实例设计程序，对单片机应用系统的设计方法进行了全面的介绍。文中的程序设计方法也许不是最佳方案，但对帮助单片机初学者掌握单片机的实际应用设计方法具有很大的作用。读者可以参考给出的硬件电路及源程序进行实验设计练习，从而逐步掌握具体应用系统的设计方法。

各部分的安排如下：

第1部分：51系列单片机原理。

第1章：绪论。了解单片机的发展史；理解单片机的应用模式；熟悉单片机的应用开发过程。

第2章：单片机基本结构与工作原理。理解内部结构和引脚功能；掌握RAM中SFR和数据区地址划分；掌握ROM中程序复位及中断入口地址；掌握4个输入输出口的特点；掌握所有SFR的意义及特点。

第3章：单片机的指令系统。了解什么是寻址方式和指令系统，掌握51系列的寻址方式、指令格式；掌握并学会111条指令的使用方法。

第4章：单片机基本单元结构与操作原理。掌握定时器和中断的基本结构及使用方法；理解串行口的基本结构及使用方法。

第5章：汇编语言程序设计基础。了解程序设计的一般规律；掌握不同程序结构的单片机汇编程序设计的基本方法；程序举例。

第2部分：51系列单片机设计应用程序实例。

本部分共13章，详细介绍了13个单片机实际应用设计实例，有详细的电路设计介绍、完整源程序及注释。

参加本书编著工作的成员有楼然苗、李光飞，其中第2部分中的实例1、实例2、实例4、实例7、实例8及实例9由李光飞老师编写，其余部分由楼然苗老师编写。

为方便读者边学习边实践，将本书第 6 章～第 18 章应用例子中的源程序放在了本书所附光盘中，分别对应于光盘中的实例 1～实例 13。读者可直接将光盘中的程序复制到自己的电脑上进行实验调试。

本书在出版编辑过程中得到了北航出版社的大力支持，在此表示衷心的感谢。同时对在编写第一部分内容时参考的多部单片机原理著作的作者表示深深的谢意。

作者

2002 年 12 月

目　录

第 1 部分　51 系列单片机原理

第 1 章　绪　论

第 2 章　单片机基本结构与工作原理

第 2 部分　51 系列单片机设计应用程序实例

第 1 部分

51 系列单片机原理

第1章　绪　论

1.1　嵌入式系统

1.1.1　什么是嵌入式系统

通常将满足海量高速数值计算的计算机称为通用计算机系统；而把面向工控领域对象，嵌入到工控应用系统中，实现嵌入式应用的计算机称之为嵌入式计算机系统，简称嵌入式系统。

嵌入式系统具有以下特点：

(1) 面对控制对象。如传感信号输入、人机交互操作、伺服驱动等。

(2) 嵌入到工控应用系统中的结构形态。

(3) 能在工业现场环境中可靠运行的品质。

(4) 突出控制功能。如对外部信息的捕捉、对控制对象实时控制和有突出控制功能的指令系统(I/O 控制、位操作和转移指令等)。

1.1.2　嵌入式系统的种类

嵌入式系统的种类可分为以下四种：

(1) 工控机。将通用计算机经机械后加固和电气加固改造后构成，其特点是软件丰富、体积大。

(2) 通用 CPU 模块。用 CPU 构成各种形式的主机板系统，一般用在大量数据处理的场合，体积较小。

(3) 嵌入式微处理器。在通用微处理器(MPU)的基核上，增添一些外围单元和接口构成单芯片形态的计算机系统，如 80386EX 就将定时器/计数器、DMA、中断系统、串行口、并行口和看门狗(WDT)等集成在一个芯片上。

(4) 单片机(微控制器)。单片机有惟一的专门为嵌入式应用系统设计的体系结构与指令系统，最能满足嵌入式应用要求。单片机是完全按嵌入式系统要求设计的单芯片形态应用系统，能满足面对控制对象、应用系统的嵌入、现场的可靠运行及非凡的控制品质等要求，是发展最快、品种最多、数量最大的嵌入式系统。

1.2　单片机的技术发展历史

1.2.1　单片机的发展阶段

单片机的发展可分为以下四个阶段。

(1) 第一代：单片机探索阶段。主要有通用 CPU 68XX 系列和专用 CPUMCS－48 系列。

(2) 第二代:单片机完善阶段。表现在:

① 面对对象,突出控制功能,专用 CPU 满足嵌入功能;

② 寻址范围为 16 位或 8 位;

③ 规范的总线结构,有 8 位数据线,16 位地址线及多功能异步串行口(UART);

④ 特殊功能寄存器(SFR)的集中管理模式;

⑤ 海量位地址空间,提供位寻址及位操作功能;

⑥ 指令系统突出控制功能。

(3) 第三代:微控制器形成阶段。这一阶段已形成系列产品:以 8051 系列为代表,如 8031、8032、8051 和 8052 等。

(4) 第四代:微控制器百花齐放。表现在:

① 电气商、半导体商广泛加入;

② 满足最低层电子技术的应用(玩具、小家电);

③ 大力发展专用型单片机;

④ 致力于提高单片机的综合品质。

1.2.2 单片机的发展方向

未来单片机技术的发展趋势可归结为以下 10 个方面:

(1) 主流型机发展趋势。8 位单片机为主流,少量 32 位机,16 位机可能被淘汰。

(2) 全盘 CMOS 化趋势。指在 HCMOS 基础上的 CMOS 化,CMOS 速度慢、功耗小,面 HCMOS 具有低功耗及低功耗管理技术等特点。

(3) RISC 体系结构的发展。早期 CISC 指令较复杂,指令代码周期数不统一,难以实现流水线(单周期指令仅为 1MIPS)。采用 RISC 体系结构可以精简指令系统,使其绝大部分为单周期指令,很容易实现流水线作业(单周期指令速度可达 12MIPS)

(4) 大力发展专用单片机。

(5) OTPROM、flashROM 成为主流供应状态。

(6) ISP 及基于 ISP 的开发环境。FlashROM 的应用推动了 ISP(系统可编程技术)的发展,这样就可实现目标程序的串行下载,PC 机可通过串行电缆对远程目标高度仿真、更新软件等。

(7) 单片机的软件嵌入。目前的单片机只提供程序空间,没有驻机软件。ROM 空间足够大后,可装入如平台软件、虚拟外设软件和用于系统诊断管理的软件等,以提高开发效率。

(8) 实现全面功耗管理。如采用:ID、PD 模式、双时钟模式、高速时钟/低速时钟模式和低电压节能技术。

(9) 推行串行扩展总线。如 I^2C 总线等。

(10) ASMIC 技术的发展。如以 MCU 为核心的专用集成电路(ASIC)。

1.3 单片机的应用模式

1.3.1 单片机应用系统的结构

单片机应用系统的结构分三个层次。

(1) 单片机:通常指应用系统主处理机,即所选择的单片机器件。

(2) 单片机系统:指按照单片机的技术要求和嵌入对象的资源要求而构成的基本系统,如时钟电路、复位电路和扩展存储器等与单片机构成了单片机系统。

(3) 单片机应用系统:指能满足嵌入对象要求的全部电路系统。在单片机系统的基础上加上面向对象的接口电路,如前向通道、后向通道、人机交互通道(键盘、显示器、打印机等)和串行通信口(RS232)以及应用程序等。

单片机应用系统三个层次的关系如图 1.1 所示。

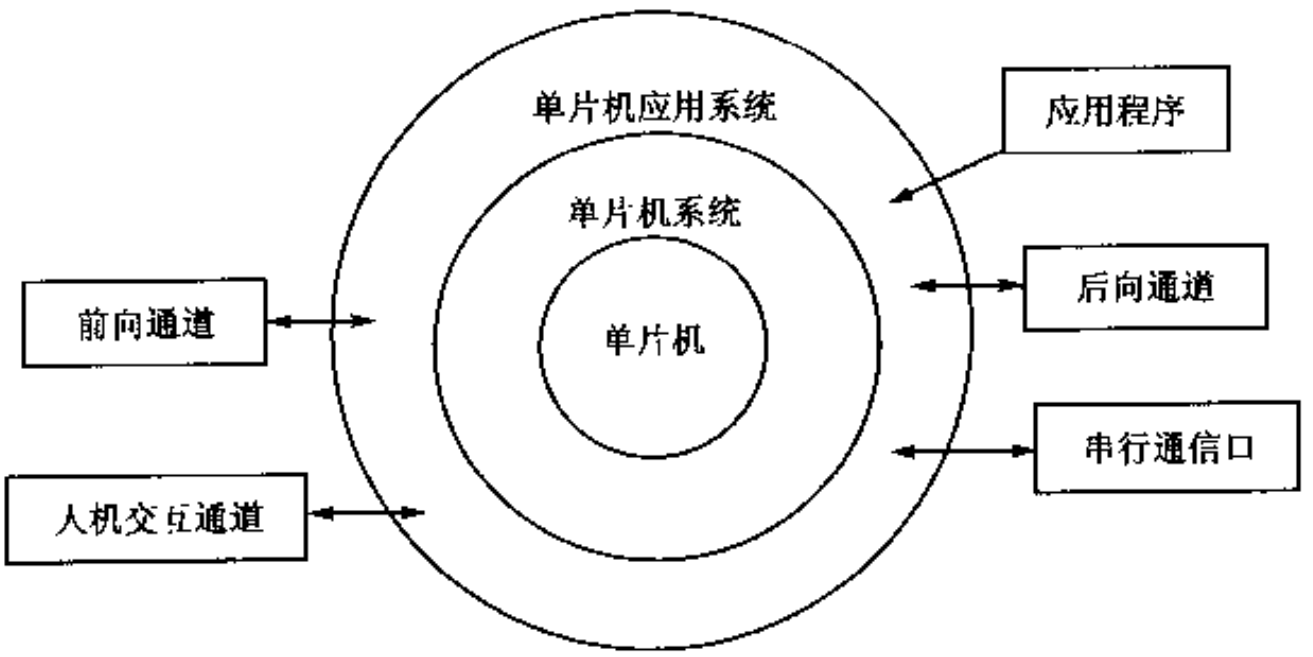


图 1.1 单片机应用系统三个层次的关系

1.3.2 单片机的种类

单片机可按应用领域、通用性、总线结构分类。

(1) 按应用领域可分为:家电类、工控类、通信类、个人信息终端等。

(2) 按通用性可分为:通用型和专用型(如计费率电表、电子记事簿)。

(3) 按总线结构可分为:总线型和非总线型。如 89C51 为总线型,有数据总线、地址总线及相应的控制线(WR、RD、EA、ALE 等);89C2051 等为非总线型,其外部引脚少,可使成本下降。

1.3.3 单片机的供应状态

按提供的存储器类型可分为以下五种状态。

(1) MASKROM 类:程序在芯片封装过程中用掩膜工艺制作到 ROM 区中,如 80C51,适合大批生产。

(2) EPROM 类:紫外线可擦写存储器类,如 87C51,价格较贵。

(3) ROMless 类:无 ROM 存储器,如 80C31,电路扩展复杂,较少用。

(4) OTPROM 类:可一次性写入程序。

(5) FlashROM(MTPROM)类:可多次编程写入的存储器,如 89C51、89C52,其成本低,开发调试方便,在恶劣环境下可靠性不及 OTPROM。

1.3.4 单片机的应用模式

单片机应用模式的分类如图 1.2 所示。各应用模式的结构如图 1.3、图 1.4、图 1.5 和图 1.6所示。

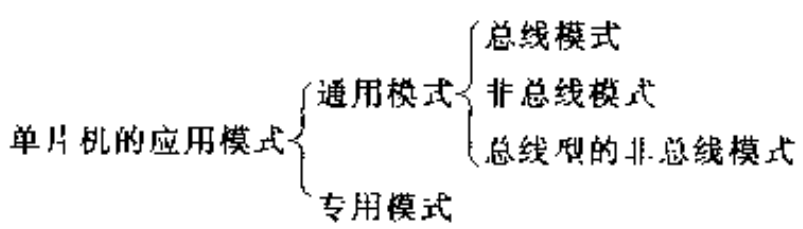


图 1.2　单片机应用模式的分类

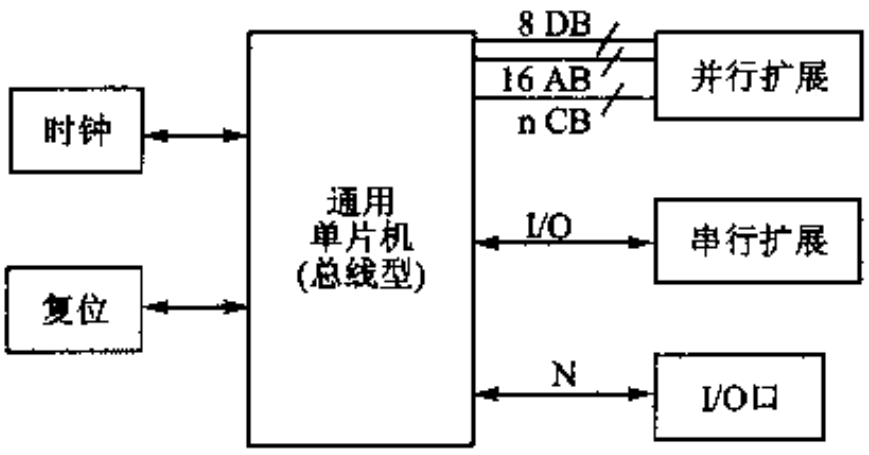


图 1.3　总线型的总线应用模式

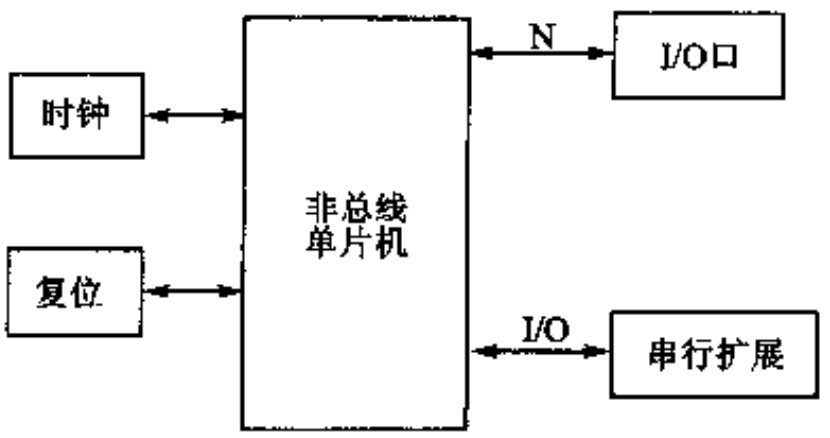


图 1.4　非总线型的应用模式

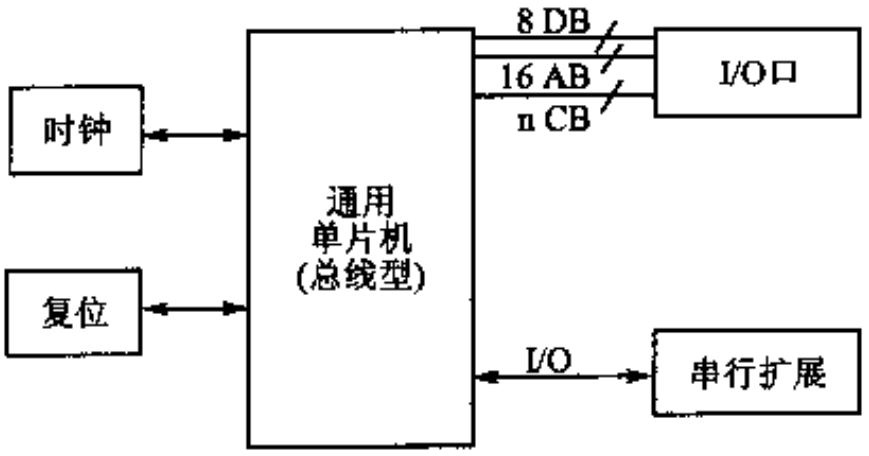


图 1.5　总线型的非总线应用模式

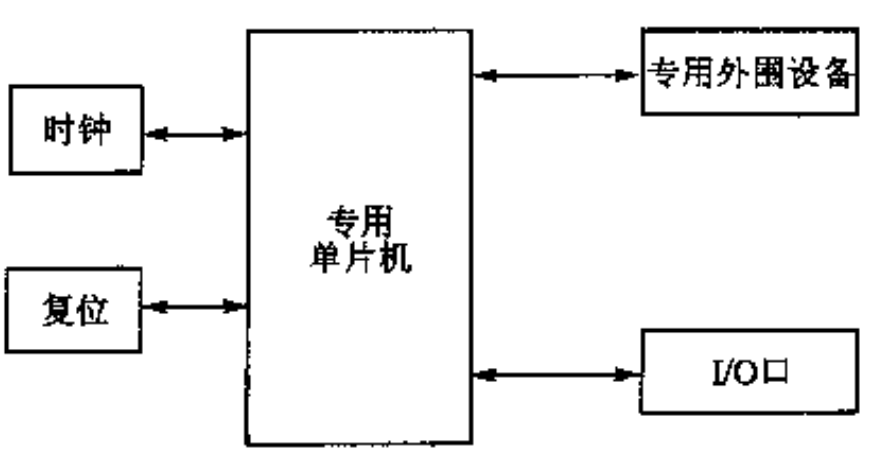


图 1.6　专用型的应用模式

1.4　单片机的应用开发过程

单片机的应用开发可分为以下五个过程。

(1) 硬件系统设计调试。如电路设计、PCB 印制板绘制等。

(2) 应用程序的设计。可使用如 Wave 等汇编工具软件进行源程序编写、编译调试等。

(3) 应用程序的仿真调试。指用仿真器对硬件进行在线调试或软件仿真调试,在调试中不断修改、完善硬件及软件。

(4) 单片机应用程序的烧写。用专用的单片机烧写器可将编译过的二进制源程序文件写入单片机(FlashROM)芯片内。

(5) 系统脱机运行检查。进行全面检查,针对出现的问题修正硬件、软件或总体设计方案。

思考与练习

1. 什么是嵌入式系统? 有哪些类型?

2. 通用计算机系统与一般嵌入式系统的主要区别在哪里?

3. 单片机的主要发展方向是什么?

4. 单片机的主要供应状态是指什么? 分几种供应状态? 在研制开发时主要用什么单片机?

5. 什么是总线型单片机? 什么是非总线型单片机? 什么是总线应用模式? 什么是非总线应用模式?

6. 简述单片机的开发过程。

第 2 章　单片机基本结构与工作原理

2.1　单片机的基本结构

典型系列单片机是由 CPU 系统、外围功能单元和归一化 I/O 端口三部分组成，如图 2.1 所示。

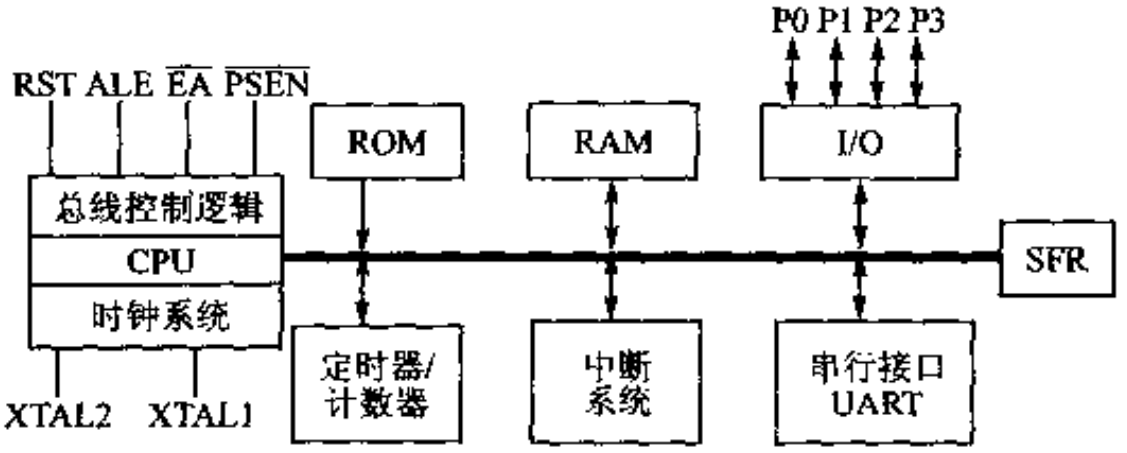


图 2.1　80C51 系列单片机的基本原理

1. CPU 系统

CPU 系统包括 CPU、时钟系统和总线控制逻辑三部分，其功能如下：

(1) CPU：包含运算器和控制器，专门为面向控制对象、嵌入式特点而设计，有突出控制功能的指令系统。

(2) 时钟系统：包含振荡器、外接谐振元件，可关闭振荡器或 CPU 时钟，其结构如图 2.2 所示。

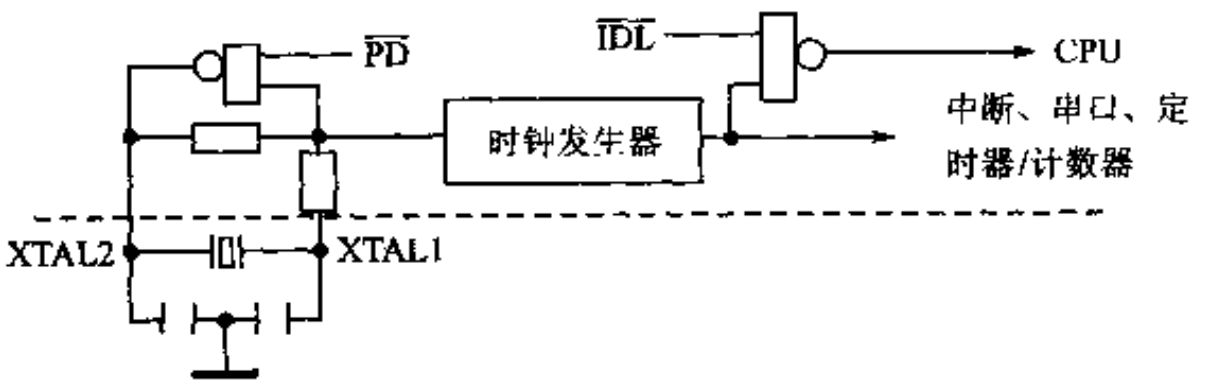


图 2.2　80C51 的时钟系统

(3) 总线控制逻辑：主要用于管理外部并行总线时序及系统的复位控制，外部引脚有 RST、ALE、EA、PSEN。

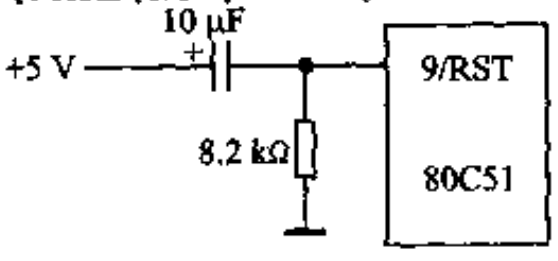


图 2.3　单片机的上电复位电路

RST：系统复位用。

ALE：数据(地址)复用控制。

EA：外部/内部程序存储器选择。

PSEN：外部程序存储器的取指控制。

单片机的上电复位电路如图 2.3 所示。

2. CPU 外围电路

CPU 外围电路包括 ROM、RAM、I/O 口和 SFR 四部分。

(1) ROM:程序存储器。地址范围为 0000H～FFFFH(64 KB)。按供应状态分:

80C51 为 ROMless，83C51 为 MaskROM，87C51 为 EPROM/OTPROM，89C51 为 flashROM。

(2) RAM:数据存储器。地址范围 00H～FFH(256 B)，是一个多用多功能数据存储器，有数据存储、通用工作寄存器、堆栈、位地址等空间。

(3) I/O 端口:80C51 系列单片机具有 4 个 8 位 I/O 端口，分别为 P0、P1、P2、P3。P0 为数据总线端口，P2、P0 组成 16 位地址总线，P1 为用户端口，P3 用于基本输入/输出端口以及并行扩展总线的读/写控制。P0、P2 可作用户 I/O 端口，P3 不作基本功能单元的输入/输出端口时，可作用户 I/O 端口。

(4) SFR:特殊功能寄存器。是单片机中的重要控制单元，CPU 对所有片内功能单元的操作都是通过访问 SFR 实现的。

3. 基本功能单元

80C51 系列单片机具有定时/计数器、中断系统和串行接口三个基本功能单元。

(1) 定时器/计数器:80C51 有 2 个 16 位定时器/计数器，定时时靠内部的分频时钟频率计数实现;作计数器时，对 P3.4 (T0) 或 P3.5 (T1) 端口的低电平脉冲计数。

(2) 中断系统:80C51 共有 5 个中断源，即 2 个外部中断源$\overline{\text{INT0}}$、$\overline{\text{INT1}}$、2 个定时器溢出中断(T0、T1)和 1 个串行中断。

(3) 串行接口 UART:一个带有移位寄存器工作方式的通用异步收发器，不仅可以作串行通信，还可用于移位寄存器方式的串行外围扩展。RXD(P3.0)脚为接收端口，TXD (P3.1) 脚为发送端口。

2.2　单片机内部资源的配置

单片机内部资源可按需要进行扩展与删减，单片机中许多型号系列是在基核的基础上扩展部分资源形成的。这些可扩展的资源有:

(1) 时钟系统的速度扩展，从 12 MHz～40 MHz。

(2) ROM 的容量扩展，从 8 KB、16 KB 到 64 KB。

(3) RAM 的容量扩展，从 256 B、512 B 到 1 024 B。

(4) I/O 口的数量扩展，从 4 个 I/O 口到 7 个 I/O 口。

(5) SFR 的功能扩展，如 ADC、PWM、WDT、模拟比较器等。

(6) 中断系统的中断源扩展。

(7) 定时器/计数器的数量扩展、功能扩展。

(8) 串行口的增强扩展。

(9) 电源供给系统的宽电压适应性扩展(从 2.7 V～6 V)。

为了满足小型廉价的要求，可将单片机的某些资源删减，某些功能加强，以达到不同场合使用要求。这些删减增加资源的内容有:

(1) 总线删减。如 89C1051、89C2051 删去并行总线，成为 20 脚封装。

(2) 功能删减。如 89C1051 只有 1 KB 的 ROM、64 B 的 RAM、1 个定时器/计数器，删除了串行口 UART 单元。

(3) 某些功能加强。如增加模拟比较器、计数器捕捉功能等。

2.3　单片机的外部特性

2.3.1　单片机的引脚分配及功能描述

1. 80C51 单片机不同封装的引脚分配图

80C51 系列的 DIP、LCC、QFP 封装引脚示意图如图 2.4 所示。

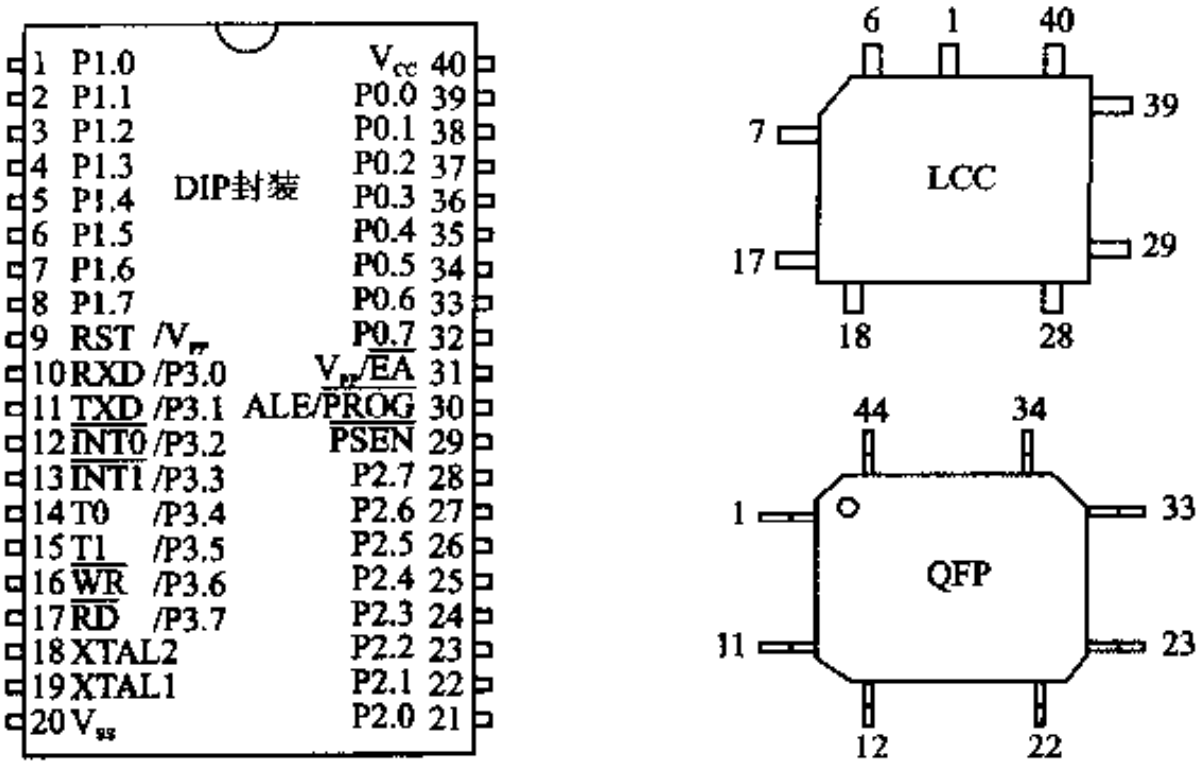


图 2.4　80C51 系列的 DIP、LCC、QFP 封装引脚示意图

2. 80C51 引脚功能描述

80C51 引脚功能描述如表 2.1 所列。

表 2.1　80C51 引脚功能描述

引脚标记	引脚编号			端口类别	引脚名称及功能描述
	DIP	LCC	QFP		
V_{SS}	20	22	16	I	地端：0V 基准
V_{CC}	40	44	38	I	电源端：正常操作、空闲、掉电状态的供电
P0.0～P0.7	39～32	43～36	37～30	I/O	P0 口：为开漏结构的准双向口，是 80C51 并行总线的数据总线和低 8 位地址总线；不作总线使用时，也可用作普通 I/O 口
P1.0～P1.7	1～8	2～9	40～44 1～3	I/O	P1 口：带内部上拉电阻的准双向口
P2.0～P2.7	21～28	24～31	18～25	I/O	P2 口：带内部上拉电阻的准双向口，是并行总线的高 8 位地址线，不作总线地址线时，也可用作普通 I/O 口

续表 2.1

引脚标记	引脚编号			端口类别	引脚名称及功能描述
	DIP	LCC	QFP		
P3.0～P3.7	10～17	11， 13～19	5， 7～13	I/O	P3 口：带内部上拉电阻的准双向口，具有复用功能，除作普通 I/O 口外，还可作以下用途： RXD：UART 的串行输入口，移位寄存器方式的数据端 TXD：UART 的串行输出口，移位寄存器方式的时钟端 INT0：外部中断 0 输入口 INT1：外部中断 1 输入口 T0：定时器/计数器 0 输入口 T1：定时器/计数器 1 输入口 WR：片外 RAM“写”控制信号 RD：片外 RAM“读”控制信号
RST/V_{PP}	0	10	4	I	复位端：高电平有效复位，在复位端上保持两个机器周期的高电平即可完成操作
ALE/$\overline{PROG}$	30	33	27	I/O	地址锁存允许/编程脉冲输入端：访问外部存储器时，提供 P0 口作为低 8 位地址的锁存信号；编程写入时，作为编程脉冲输入端；正常操作时，输出时钟振荡器的 6 分频频率信号
$\overline{PSEN}$	29	32	36	O	外部程序存储器选通信号：使用外部程序存储器时，作为外部程序存储器的取指控制端
$V_{PP}/\overline{EA}$	31	35	29	I	内外程序存储器选择/编程写入电源输入端：EA＝0 时选择访问外部程序存储器；编程写入时输入编程电压 V_{PP}
XTAL2	18	20	14	O	谐振器端口 2：时钟振荡器反相放大器输出端
XTAL1	19	21	15	I	谐振器端口 1：时钟振荡器反相放大器输入端

2.3.2　80C51 系列单片机引脚功能分类

(1) 基本引脚：电源 V_{cc}、V_{ss}，时钟 XTAL2、XTAL1 和复位 RST。

(2) 并行扩展总线：数据总线 P0 口，地址总线 P0 口(低 8 位)、P2 口(高 8 位)和控制总线 ALE、$\overline{PSEN}$、$\overline{EA}$。

(3) 串行通信总线：发送口 TXD 和接收口 RXD。

(4) I/O 端口：P1 口为普通 I/O 口，P3 口可复用作普通 I/O 口，P0、P2 口不作并行口时也可作普通 I/O 口。

2.3.3　单片机的引脚应用特性

1. 并行总线的构成

80C51 并行总线的构成如图 2.5 所示。

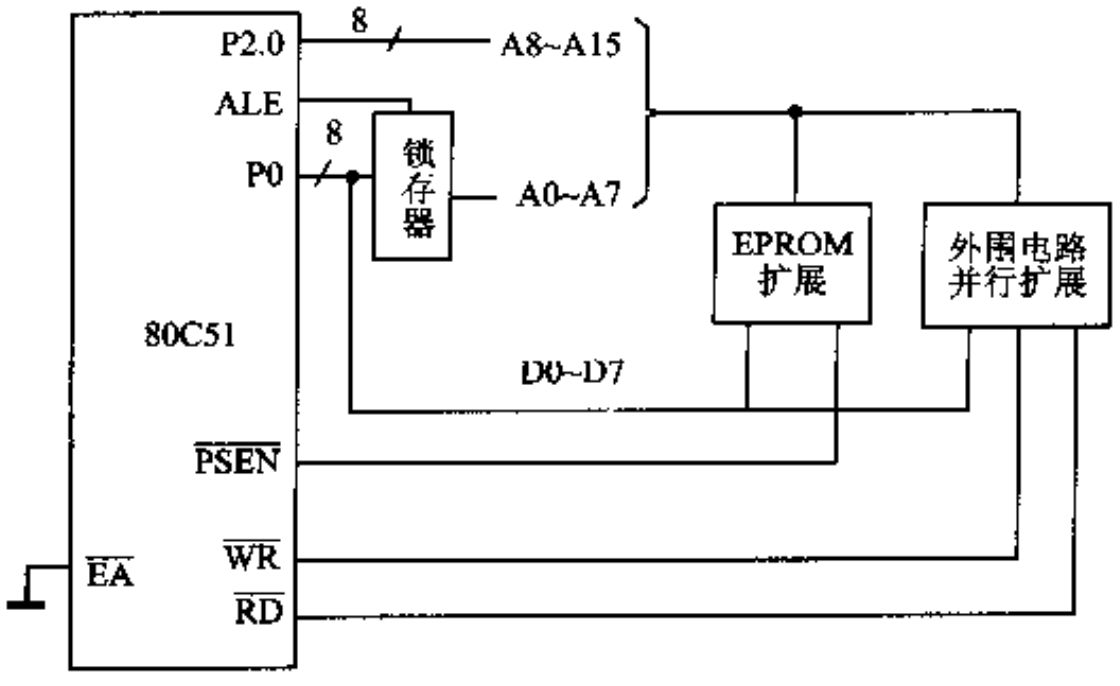


图 2.5　80C51 并行总线的构成

并行总线口特点：

(1) P0 口为地址/数据复用口。

(2) 两个独立的并行扩展空间。程序存储器使用$\overline{\text{PSEN}}$取指控制信号，数据采用$\overline{\text{WR}}$、$\overline{\text{RD}}$存取控制信号。

(3) 外围扩展统一编址。在 64 KB 的空间上，可扩展外数据存储器或其他外围器件。

2. 引脚复用特性

P3 口、P0 口、P2 口均可用作普通 I/O 口。

3. I/O 的驱动特性

由于采用 CMOS 电路，输入电流极微，通常不考虑 I/O 端口的扇出能力，当负载为 LED、继电器等功率驱动元件时才考虑驱动能力。

2.4　80C51 的 SFR 运行管理模式

2.4.1　80C51 的 SFR

1. SFR 清单

80C51 共有 21 个特殊功能寄存器，用于实现对片内 13 个电路单元的操作管理，其中 11 个可位寻址，10 个不可位寻址。表 2.2 列出了这些寄存器名及其功能特性。

表 2.2　80C51 中的 SFR

符号	寄存器名	位地址、位标记及位功能								直接地址	复位状态
		D7	D6	D5	D4	D3	D2	D1	D0		
(1)可位寻址 SFR(共 11 个)											
ACC	累加器	E7	E6	E5	E4	E3	E2	E1	E0	E0H	00H
		ACC. 7	ACC. 6	ACC . 5	ACC. 4	ACC. 3	ACC. 2	ACC. 1	ACC. 0		
B	B 寄存器	F7	F6	F5	F4	F3	F2	F1	F0	F0H	00H
		B. 7	B. 6	B. 5	B. 4	B. 3	B. 2	B. 1	B. 0		

续表 2.2

符号	寄存器名	位地址、位标记及位功能								直接地址	复位状态
		D7	D6	D5	D4	D3	D2	D1	D0		
PSW	程序状态字	D7	D6	D5	D4	D3	D2	D1	D0	D0H	00H
		CY	AC	F0	RS1	RS0	OV	— — —	P		
IP	中断优先权寄存器	BF	BE	BD	BC	BB	BA	B9	B8	B8H	×××00000B
		— — —	— — —	— — —	PS	PT1	PX1	PT0	PX0		
P3	P3 口	B7	B6	B5	B4	B3	B2	B1	B0	B0H	FFH
		P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0		
IE	中断允许寄存器	AF	AE	AD	AC	AB	AA	A9	A8	A8H	0××00000B
		EA	— — —	— — —	ES	ET1	EX1	ET0	EX0		
P2	P2 口	A7	A6	A5	A4	A3	A2	A1	A0	A0H	FFH
		P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0		
SCON	串行口控制寄存器	9F	9E	9D	9C	9B	9A	99	98	98H	00H
		SM0	SM1	SM2	REN	TB8	RB8	TI	RI		
P1	P1 口	97	96	95	94	93	92	91	90	90H	FFH
		P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0		
TCON	定时器控制寄存器	8F	8E	8D	8C	8B	8A	89	88	88H	00H
		TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0		
P0	P0 口	87	86	85	84	83	82	81	80	80H	FFH
		P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0		
(2)不可位寻址 SFR(共 10 个)											
SP	栈指示器									81H	07H
DPL	数据指针低 8 位									82H	00H
DPH	数据指针高 8 位									83H	00H
PCON	电源控制寄存器	SMOD	— — —	— — —	— — —	GF1	GF0	PD	IDL	87H	0×××0000B
TMOD	定时器方式寄存器	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	89H	00H
TL0	T0 寄存器低 8 位									8AH	00H
TL1	T1 寄存器低 8 位									8BH	00H
TH0	T0 寄存器高 8 位									8CH	00H
TH1	T1 寄存器高 8 位									8DH	00H
SBUF	串行口数据缓冲器									99H	××××××××B

2. SFR 的应用特性

(1) 可以对 SFR 进行编程操作。

(2) 对 SFR 编程时，必须了解该 SFR 的位定义、位地址、字节地址等情况。

(3) 应用时要区分控制位与标志位。

(4) 要了解标志位的清除特性(硬件自动清除或软件清除)。

2.4.2　80C51 中 SFR 的寻址方式

1. SFR 的直接寻址方式

在 80C51 片内 RAM 80H～FFH 地址上有 2 个物理空间，1 个是 SFR 的单元地址，另 1 个是高 128 B 的数据地址。采用直接寻址访问的是 SFR，而间接寻址则访问数据存储器。

2. SFR 的位寻址与字节寻址

在 80C51 中有许多 SFR 可位操作(直接地址为×0H 或×8H)，空出的 8 个地址号依次作为 8 个位地址。如 TCON 的直接地址为 88H，而 IT0 的位地址也是 88H，对 TCON 寻址使用直接寻址，而对 IT0 寻址则使用位寻址。

2.4.3　SFR 的复位状态

(1) I/O 端口均为 FFH 状态；

(2) 栈指示器 SP＝07H；

(3) 所有 SFR 有效位均为零；

(4) 复位时 RAM 中值不变，但上电复位时 RAM 中为随机数；

(5) SBUF 寄存器为随机数。

2.5　单片机 I/O 端口及应用特性

2.5.1　80C51 单片机 I/O 口电气结构

80C51 单片机的 P0、P1、P2 和 P3 的结构如图 2.6 所示。

其特点如下：

(1) 锁存器加引脚结构。

(2) I/O 复用结构：P0 口作并行扩展时为三态双向口；P3 口为功能复用 I/O 口，由内部控制端控制。

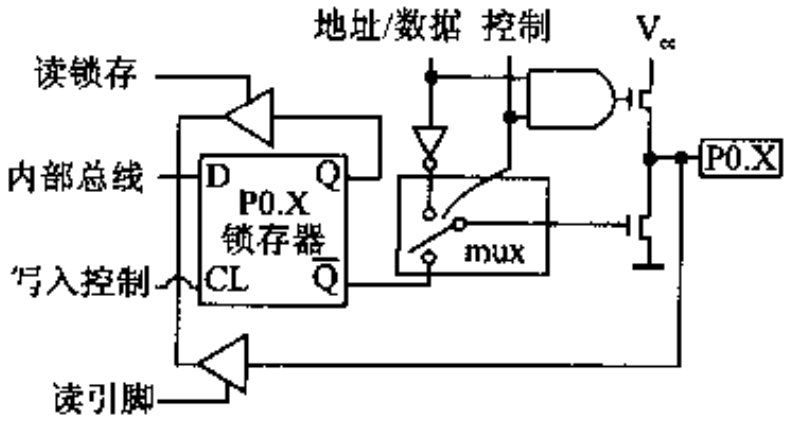
(3) 准双向口结构：P0～P3 口作普通 I/O 口使用时均为准双向口，典型结构如 P1 口。输入时读引脚，输出时为写锁存器。

2.5.2　I/O 端口应用特性

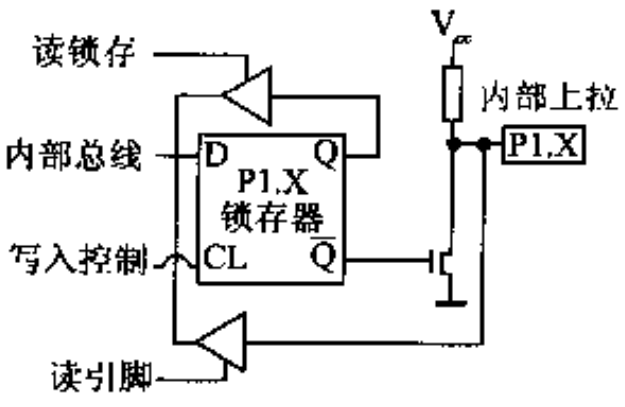
(1) 端口的自动识别：P0、P2 总线复用、P3 功能复用，内部资源自动选择。

(2) 端口锁存器的读、改、写操作：都是一些逻辑运算、置位/清除、条件转移等指令。

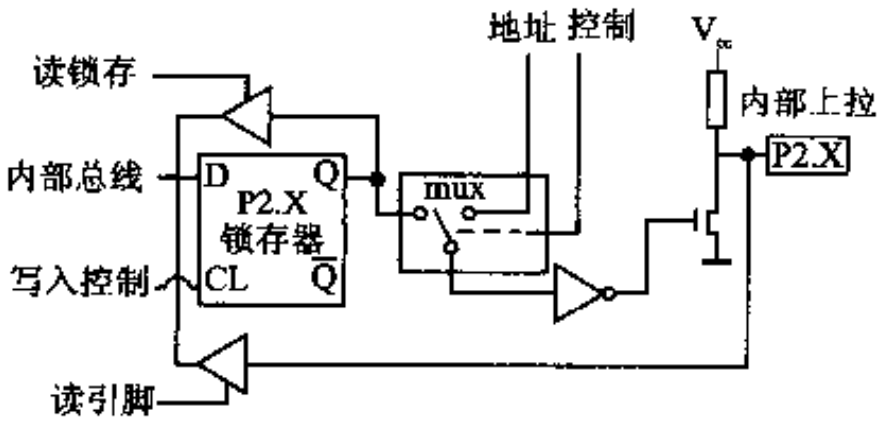
(3) 读引脚的操作指令：I/O 端口被指定为源操作数即为读引脚操作。例如，执行“MOV A,P1”时，P1 口的引脚状态传送到累加器中；而相对应的“MOV　P0，A”指令则是将



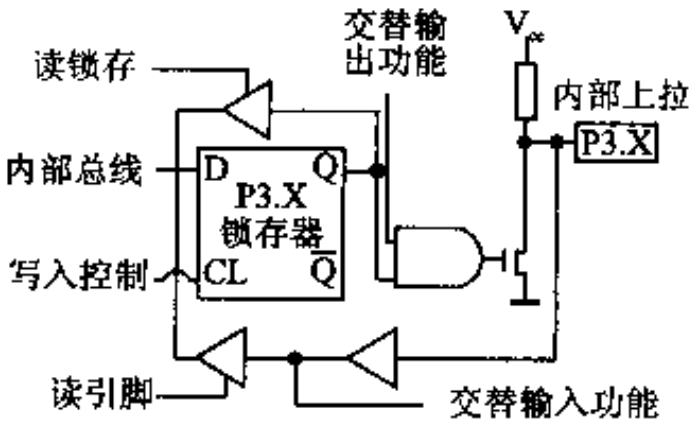
(a) P0口位



(b) P1口位



(c) P2口位



(d) P3口位

图 2.6　80C51 的 P0、P1、P2、P3 口的结构示意图

累加器的内容传送到 P1 口锁存器中。

(4) 准双向口的使用：端口作输入时，读入时应先对端口置“1”，然后再读引脚。

例如，将 P1 口的状态读入累加器 A 中，就需执行 2 条指令：

```
MOV    P1, #0FFH      ;P1 口置输入状态
MOV    A,  P1         ;将 P1 口读入 A 中
```

(5) P0 口作普通口使用：此时必须加上拉电阻。

(6) I/O 驱动特性：P0 口可驱动 8 个 LSTTL 输入端，P1～P3 口可驱动 4 个 LSTTL 输入端。

2.6　80C51 单片机存储器系统及操作方式

2.6.1　80C51 存储器的结构

程序存储器寻址范围为 64 KB(用 PC 或 DPTR)，片内数据存储器寻址范围为 256B，80H～FFH 只能间接寻址，片外数据存储器寻址范围为 64 KB(用 DPTR、P2、@Ri)。

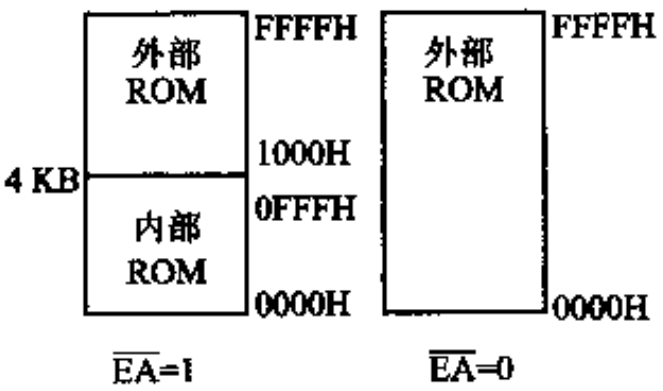


图 2.7　80C51 程序存储器系统结构

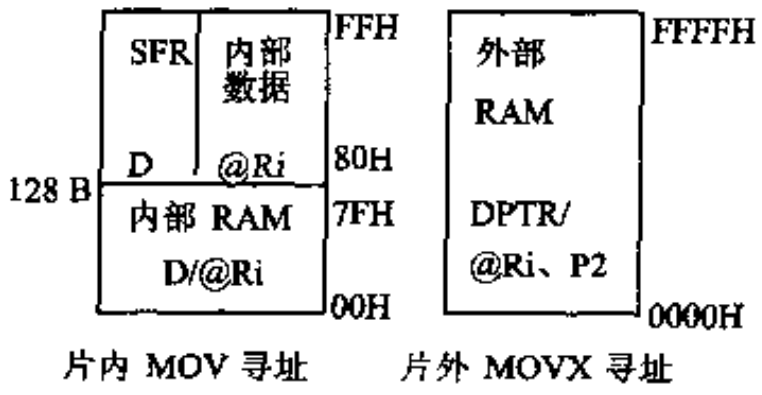


图 2.8　80C51 数据存储器系统结构

2.6.2　程序存储器及其操作

程序存储器用来存放应用程序和表格常数，设计中应根据要求选择容量，其最大容量为 64 KB。单片机复位时，PC 指针从 0000H 地址开始执行，应用程序的第一条指令的入口必须是 0000H。程序存储器中有一些固定的中断入口地址，这些入口地址不得安放其他程序，而应安放中断服务程序，这些入口地址如表 2.3 所列。

表 2.3　程序存储器的固定中断入口地址

ROM 地址	用　途	优先级
0000H	复位程序运行入口	高
0003H	外中断 0 入口地址(IE0)	
000BH	定时器 T0 溢出中断入口地址(TF0)	
0013H	外中断 1 入口地址(IE1)	
001BH	定时器 T1 溢出中断入口地址(TF1)	
0023H	串行口发送/接收中断入口地址(RI+TI)	
002BH	定时器 T2* 中断入口地址(TF2+EXF2)	低

程序存储器的操作有：

(1) 程序指令的自主操作：按 PC 指针顺序操作。

(2) 表格常数的查表操作：用 MOVC 指令。

2.6.3　数据存储器及其操作

1. 片内数据存储器的结构

数据存储器的结构如图 2.9 所示。

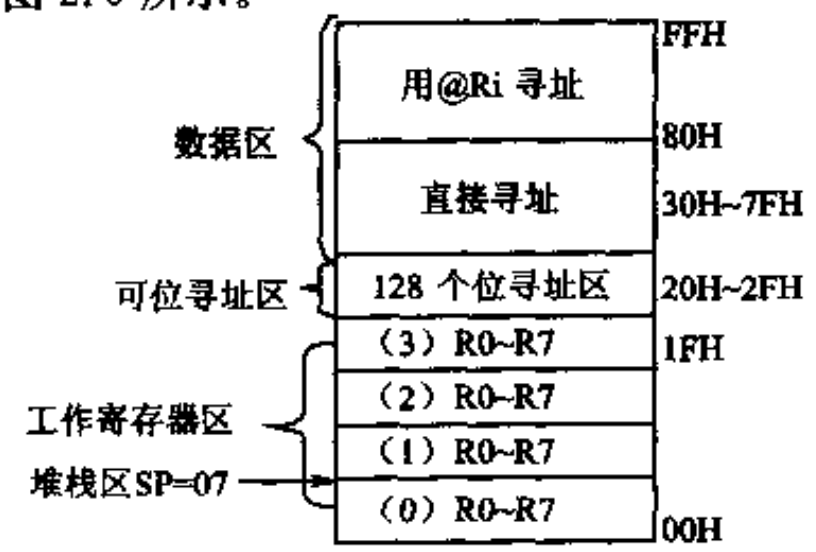


图 2.9　数据存储器的结构

2. 片内数据存储器的应用特性

(1) 复用特性:除工作寄存器、位寻址单元有固定空间外,其余没有使用的都可作数据缓冲区。

(2) 复位特性:复位时 SP=07H、PSW=00H,故栈底在 07H,工作寄存器为 0 组。

(3) 活动堆栈:程序运行中,SP 可随意设置。

3. 片内数据存储器的操作

(1) 直接寻址操作,如:

```
MOV     30H, #50H       ;30H←#50H
```

(2) 间接寻址操作,如:

```
MOV     R0,#30H         ;30H 赋给 R0
MOV     A,@R0           ;A←((R0))
```

(3) 位地址空间操作,如:

```
SETB    00H             ;20H 的 D0 位置 1
```

(4) 工作寄存器的选择操作,如:

```
MOV     PSW,#18H        ;RS1、RS0 置成 1 1
```

(5) 堆栈操作,如:

```
MOV     SP,#70H         ;栈底设在 70H
```

4. 片外数据存储器的操作

使用 MOVX 命令,只能与 A 交换数据。

(1) 读入数据

```
MOVX  A,  @TPDR
```

或

```
MOVX  A,  @Ri
```

(2) 写入数据

```
MOVX  @TPTR, A
```

或

```
MOVX  @Ri, A
```

例如:将片外 567FH 单元的数写入累加器 A 中,用 DPTR 指针操作为:

```
MOV   DPTR, #567FH
MOVX  A, @DPTR
```

用 R0 间接寻址操作为:

```
MOV    R0,  #7FH
MOV    P2,  #56H
MOVX   A,  @R0
```

思考与练习

1. 典型单片机由哪几部分组成？每部分的基本功能是什么？
2. 单片机的主要性能包括哪些？
3. 描述单片机的引脚功能。
4. 在 80C51 中，SFR 在内存里占什么空间？其寻址方式是怎样的？
5. 在 80C51 中，哪些内存空间可以位寻址？位地址范围是多少？
6. 在 80C51 的 80H～FFH 内分哪两个物理空间？如何来区别这两个空间？
7. 在程序存储器中，程序复位运行及中断入口的地址是在哪里？

第 3 章　单片机的指令系统

3.1　单片机指令系统基础

3.1.1　汇编指令格式

汇编指令是指令系统最基本的书写方式，由助记符、目的操作数、源操作数组成。格式如下：

(标号：)操作码助记符　　目的操作数，源操作数　(；注释)

标号可以是以英文字母开头的字母、数字和某些特殊符号的序列。某条指令一旦赋予标号，则在其他指令的操作数中即可引用该标号作为引用地址。

操作码助记符用来表达指令的操作功能。

操作数是指令操作所需的数据、地址或符号(标号)。通常右边操作数为源操作数，左边为目的操作数。例如：

```
MOV     A，#40H                 ；把数 40H 送入累加器 A 中
MOV     A，40H                  ；把 40H 中的数送入累加器 A 中
INC     A                       ；A 中的数加 1
CJNE    A，#40H，LOOP1          ；A 中数与数 40H 比较，不等时程序转到 LOOP1
DIV     AB                      ；A 中内容被 B 中内容除，商在 A 中，余数在 B 中。
```

3.1.2　指令代码格式

指令代码是程序指令的二进制数字表示方法。指令有单字节指令、双字节指令和三字节指令。第一个字节代码为操作码，表达了指令的操作功能，第二、三个字节则为操作数，可以是地址或立即数。

表 3.1 中列出了几种汇编指令与指令代码。

表 3.1　汇编指令与指令代码

代码字节	指令代码	汇编指令	指令周期
单字节	84	DIV　AB	四周期
单字节	A3	INC　TPTR	双周期
双字节	7440	MOV A，#40H	单周期
三字节	B440　rel	CJNE A，#40H，LOOP	双周期

3.1.3　汇编指令中的符号约定

汇编指令中的符号约定如下：

Rn(0～7)：当前选中的 8 个工作寄存器 R0～R7；

Ri(i=0,1)：当前选中的用于间接寻址的两个工作寄存器 R0、R1；

Direct：8 位直接地址，可以是 RAM 单元地址(00H～7FH)，或特殊功能寄存器(SFR)地址(80H～FFH)；

#data：8 位常数；

#data16：16 位常数；

addr16：16 位地址；

addr11：11 位地址；

rel：8 位偏移地址，表示相对跳转的偏移字节，按下一条指令的第一个字节计算，在－128～＋127 取值范围内；

DPTR：16 位数据指针；

bit：位地址，内部 RAM 20H～2F 中可寻址位和 SFR 中的可寻址位；

A：累加器；

B：B 寄存器，用于乘法等指令中；

C：进位标志或进位位，或位操作指令中的位累加器；

@：间接寻址寄存器的前缀；

/：位操作的取反前缀。

3.1.4　指令系统的寻址方式

指令系统的寻址方式有以下七种方式：

1. 寄存器寻址方式

(1) 单片机中的所有工作寄存器 R0～R7 及 SFR 都是可寻址寄存器，这些寄存器都以寄存器名作指令操作数。例如：

```
MOV     A,R0
MOV     SP,#70H
```

(2) 在寄存器寻址方式的操作指令中，寄存器内容作为操作数，可以是源操作数或目的操作数。例如：

```
MOV     R1,#10H
MOV     A,R1
```

2. 直接寻址方式

(1) 直接寻址的空间有片内数据存储器的直接地址 direct，其包括 00H～7FH 中的数据区及 80H～FFH 中的 SFR。

(2) 直接寻址方式的操作指令直接把地址作为操作数来运行，既可作为源操作数，也可作为目的操作数。例如：

```
MOV     50H,60H
MOV     DPH,40H
INC     60H
```

3. 间接寻址方式

(1) 间接寻址的地址空间有片内数据存储器的 00H～FFH 和片外数据存储器的

0000H～FFFFH。

(2) 间接寻址的寄存器有 Ri 和 DPTR，间接寻址时要在间接寻址寄存器标记前面加@符号。

(3) 间接寻址时，寄存器中的内容是操作数的地址。例如：

```
MOV     R0,#30H
MOV     A,@R0
MOV     DPTR,#0FFFH
MOVX    A,@DPTR
```

4. 位寻址方式

(1) 位寻址的位地址在 RAM 的 20H～2FH 单元的 128 个位和 SFR 中可位寻址的位单元。

(2) 进位位 C 作为位操作的位累加器。

(3) 在位寻址操作中，位单元可以使用地址编号或位地址名。例如：

```
SETB    TR0
CLR     00H
ANL     C,5FH     ;将 5FH 中的位状态与进位位 C 相与，结果在 C 中
```

5. 立即寻址方式

(1) 常数用来参与指令操作，一般用#标记作前缀。

(2) 立即数在寻址操作中只能作源操作数。例如：

```
MOV     A,#30H
MOV     DPTR,#2FFFH
ANL     A,#0F4H
```

6. 基址变址寻址方式

(1) 基址变址寻址方式是一种间接寻址方式，PC 和 DPTR 可作为基址地址，A 作为变量地址。

(2) 共有三条指令：

```
MOVC    A,@A+DPTR
MOVC    A,@A+PC
JMP     @A+DPTR
```

7. 相对寻址方式

(1) 相对寻址中，相对地址 rel 是一个 8 位的地址偏移量，是相对于转移指令下一条指令第一个代码的地址偏移量，为－128～＋127。

(2) 使用中应注意 rel 的范围不要超出。例如：

```
JZ      LOOP
DJNE    R0,DISPLAY
```

3.2　指令系统的分类与速解

3.2.1　指令的分类图解

按指令的操作功能，80C51 单片机的指令系统由数据传送、算术操作、逻辑操作、程序转移

和位操作指令组成，共有 111 条指令。

指令图解的标记符号如下：

箭头：单箭头表示操作数从源数到目的操作数；双箭头表示源操作数与目的操作数可互换；箭头上标有指令助记符。

圆框：为累加器 A 或位累加器 C。

矩形框：为指令操作数的空间。

虚线矩形框：为立即数 # data

1. 数据传送类指令(共 29 条)

(1) 程序存储器查表指令 MOVC(共 2 条)，如图 3.1 所示。

(2) 片外数据存储器数据传送指令 MOVX(共 4 条)，如图 3.2 所示。

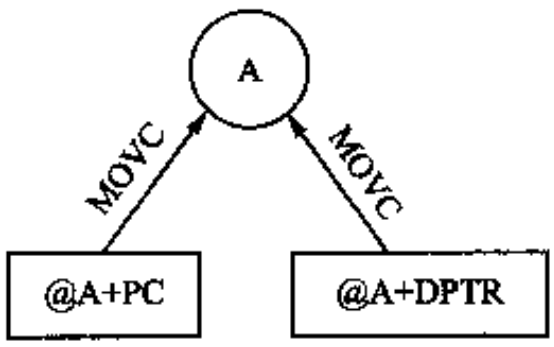


图 3.1 程序存储器查表指令

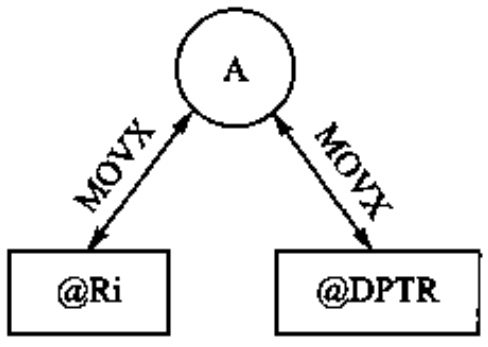


图 3.2 片外数据存储器数据传送指令

(3) 片内数据 RAM 及寄存器的数据传送指令 MOV、PUSH 和 POP(共 18 条)，如图 3.3 所示。

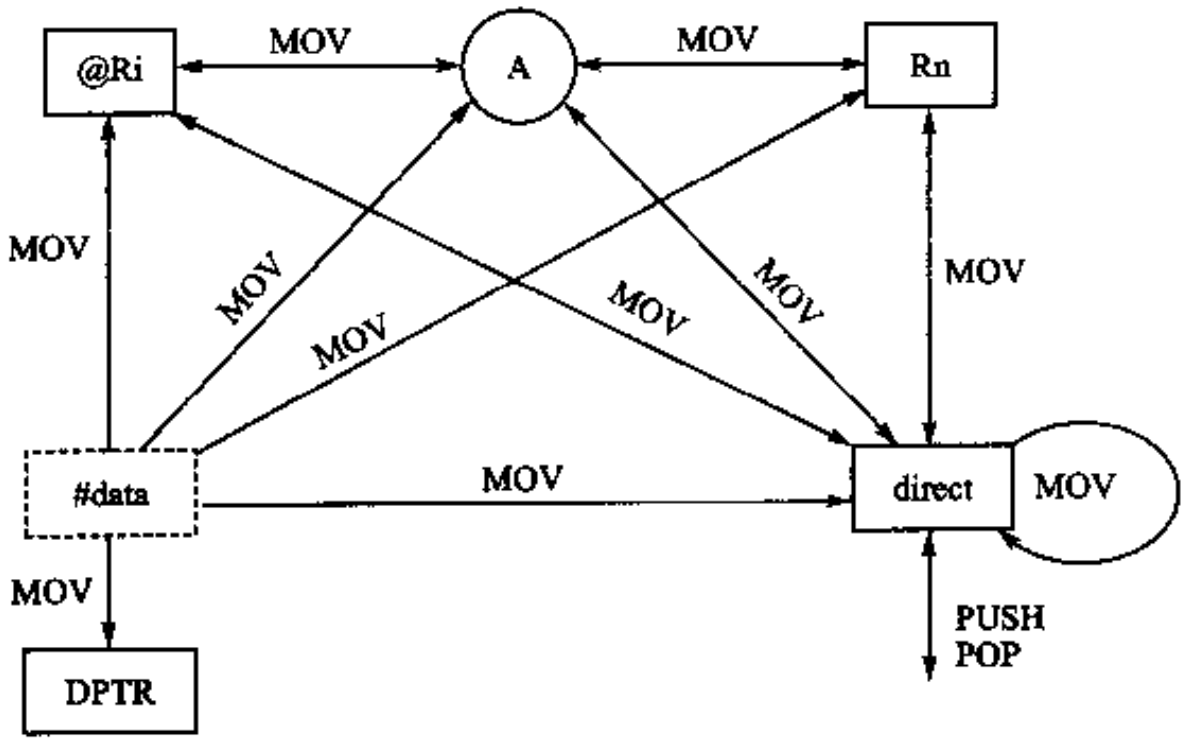


图 3.3 片内 RAM 及寄存器的数据传送指令

(4) 数据交换指令 XCH、XCHD 和 SWAP(共 5 条)，如图 3.4 所示。

2. 算术运算类指令(共 24 条)

算术运算类指令包括：ADD、ADDC、SUBB、MUL、DIV、INC、DEC 和 DA，如图 3.5 所示。

3. 逻辑运算类指令(共 24 条)

逻辑运算类指令包括：ANL、ORL、XRL、CLR、CPL、RR、RRC、RL 和 RLC，如图 3.6 所示。

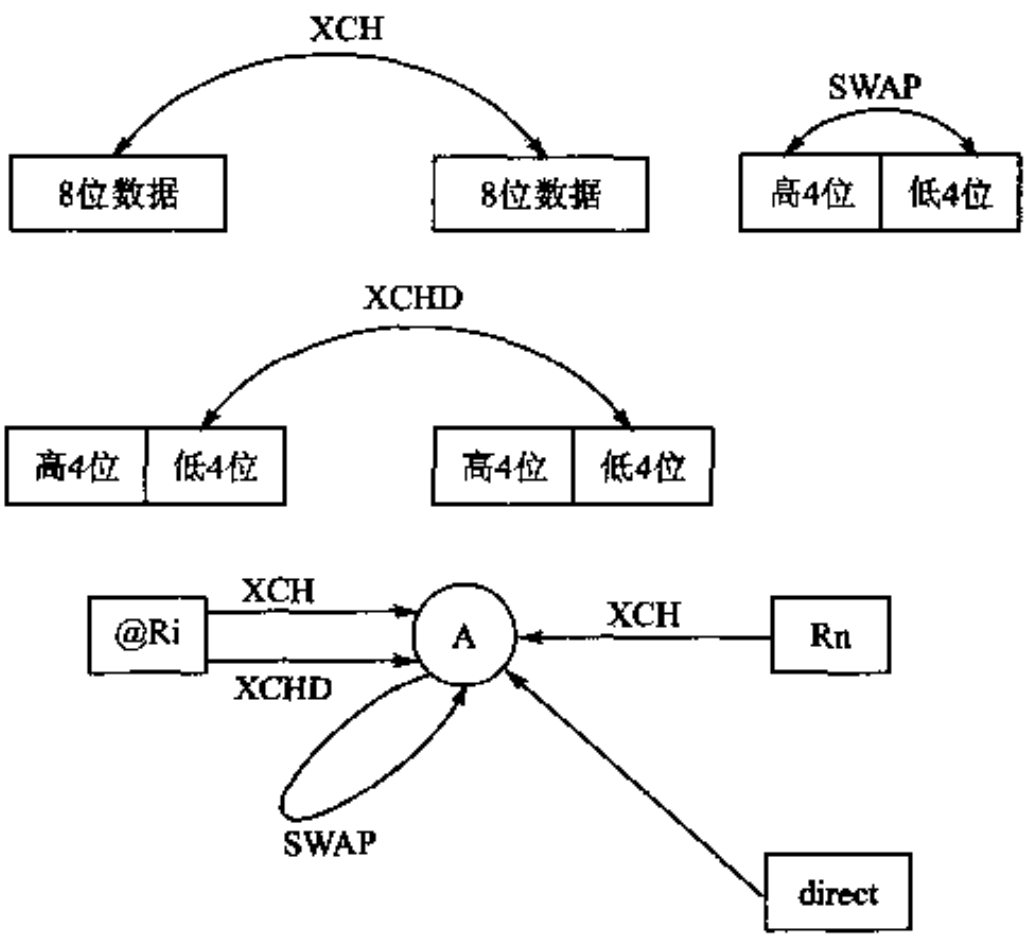


图 3.4　数据交换指令

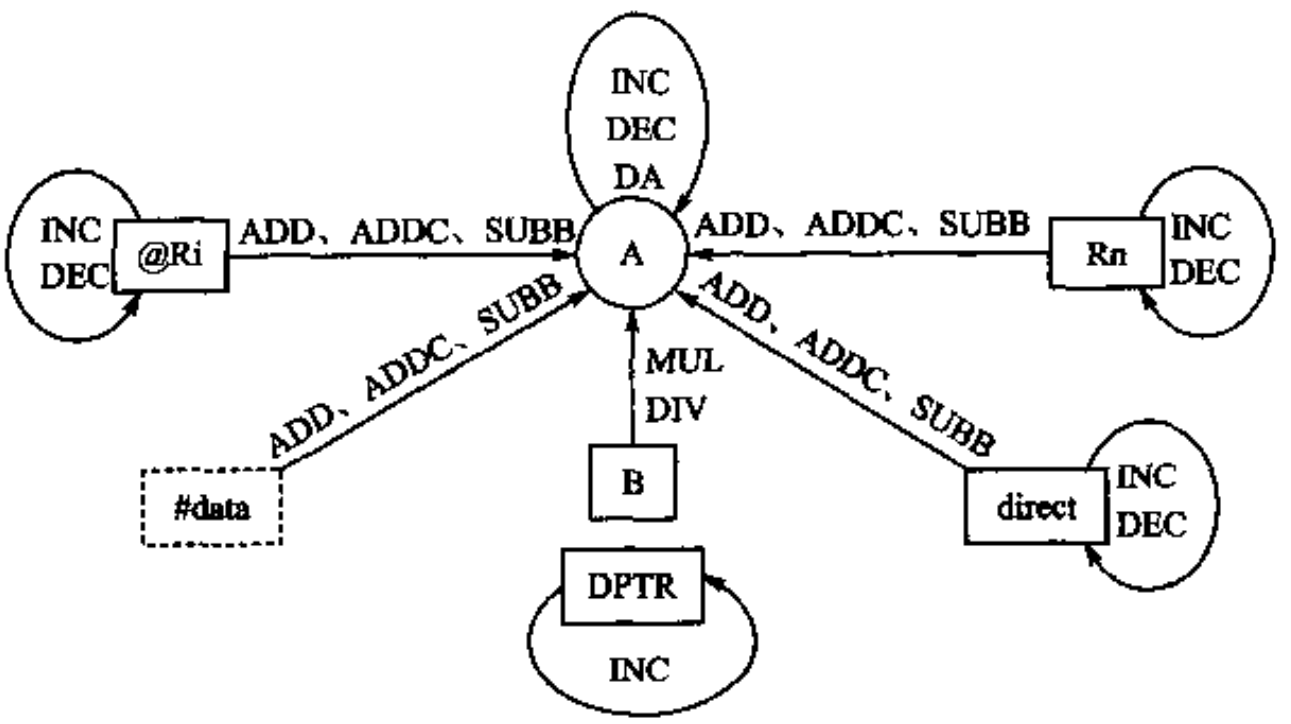


图 3.5　算术运算类指令

4. 转移操作类指令(共 17 条)

(1) 无条件转移类指令(共 9 条)：LJMP、AJMP、SJMP、LCALL、ACALL、JMP、RETI、RET 和 NOP。

(2) 条件转移类指令(共 8 条)：JZ、JNZ、DJNZ 和 CJNZ，如图 3.7 所示。

5. 布尔指令(共 17 条)

(1) 位操作指令(共 12 条)：MOV、ANL、ORL、CLR、SETB 和 CPL，如图 3.8 所示。

(2) 位条件转移指令(共 5 条)：JC、JNC、JB、JNB 和 JBC，如图 3.9 所示。

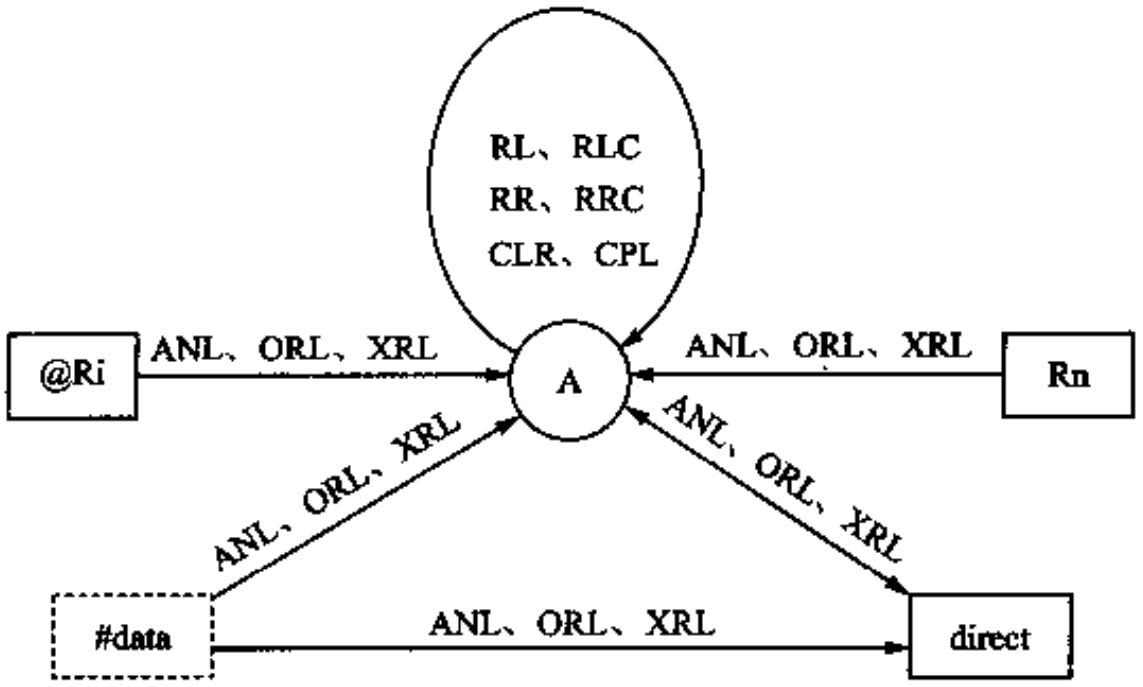


图 3.6　逻辑运算类指令

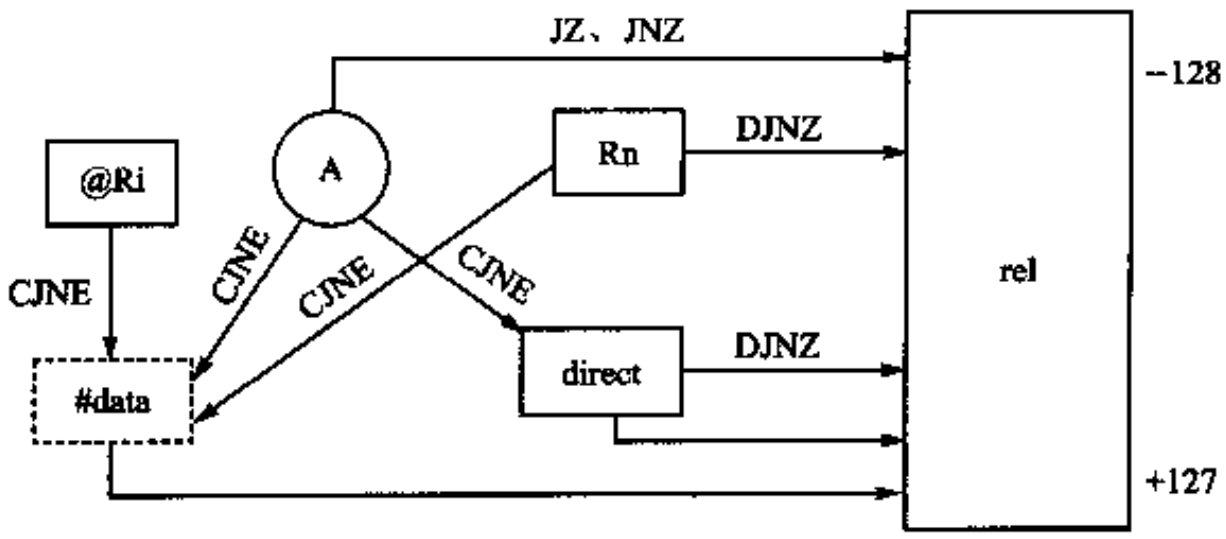


图 3.7　条件转换类指令

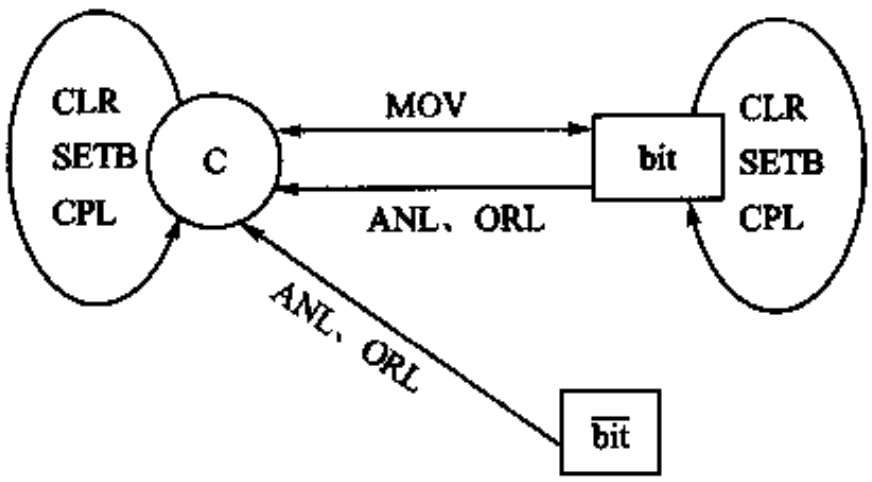


图 3.8　位操作指令

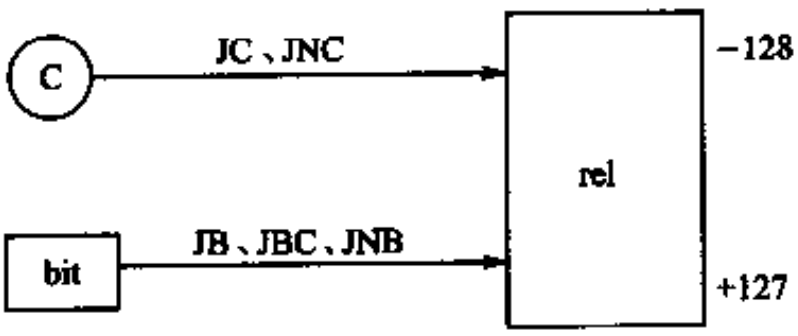


图 3.9　位条件转移指令

3.2.2　指令系统速解表

1. 数据传送指令(共 29 条)

汇编指令	操作说明	代码长度/B	指令周期	
			Tosc	Tm
(1) 程序存储器查表指令(共 2 条)				
MOVC　A,@A+DPTR	将以 DPTR 为基址,A 为偏移地址中的数送入 A 中	1	24	2
MOVC　A,@ A+PC	将以 PC 为基址,A 为偏移地址中的数送入 A 中	1	24	2
(2) 片外 RAM 传送指令(共 4 条)				
MOVX　A,@DPTR	将片外 RAM 中的 DPTR 地址中的数送入 A 中	1	24	2
MOVX　@DPTR,A	将 A 中的数送入片外 RAM 中的 DPTR 地址单元中	1	24	2
MOVX　A,@Ri	将片外 RAM 中@Ri 指示的地址中的数送入 A 中	1	24	2
MOVX　@Ri,A	将 A 中的数送入片外@Ri 指示的地址单元中	1	24	2
(3) 片内 RAM 及寄存器间数据传送指令(共 18 条)				
MOV　A,Rn	将 Rn 中的数送入 A 中	1	12	1
MOV　A,direct	将直接地址 direct 中的数送入 A 中	2	12	1
MOV　A,#data	将 8 位常数送入 A 中	2	12	1
MOV　A,@Ri	将 Ri 指示的地址中的数送入 A 中	1	12	1
MOV　Rn,direct	将直接地址 direct 中的数送入 Rn 中	2	24	2
MOV　Rn,#data	将立即数送入 Rn 中	2	12	1
MOV　Rn,A	将 A 中的数送入 Rn 中	1	12	1
MOV　direct,Rn	将 Rn 中的数送入 direct 中	2	24	2
MOV　direct,A	将 A 中的数送入 direct 中	2	12	1
MOV　direct,@Ri	将@Ri 指示单元中的数送入 direct 中	2	24	2
MOV　direct,#data	将立即数送入 direct 中	3	24	2
MOV　direct,direct	将一个 direct 中的数送入另一个 direct 中	3	24	2
MOV　@Ri,A	将 A 中的数送入 Ri 指示的地址中	1	12	1
MOV　@Ri,direct	将 direct 中的数送入 Ri 指示的地址中	2	24	2
MOV　@Ri,#data	将立即数送入 Ri 指示的地址中	2	12	1
MOV　DPTR,#data16	将 16 位立即数直接送入 DPTR 中	3	24	2
PUSH　direct	将 direct 中的数压入堆栈	2	24	2

续表

汇编指令		操作说明	代码长度/B	指令周期	
				Tosc	Tm
POP	direct	将堆栈中的数弹出到 direct 中	2	24	2
(4) 数据交换指令(共 5 条)					
XCH	A,Rn	A 中的数和 Rn 中的数全交换	1	12	1
XCH	A,direct	A 中的数和 direct 中的数全交换	2	12	1
XCH	A,@Ri	A 中的数和@Ri 中的数全交换	1	12	1
XCHD	A,@Ri	A 中的数和@Ri 中的数半交换	1	12	1
SWAP	A	A 中数自交换(高 4 位与低 4 位)	1	12	1

2. 算术运算类指令(共 24 条)

汇编指令		操作说明	代码长度/B	指令周期	
				Tosc	Tm
ADD	A,Rn	Rn 中与 A 中的数相加,结果在 A 中,影响 PSW 位的状态	1	12	1
ADD	A,direct	direct 中与 A 中的数相加,结果在 A 中,影响 PSW 位的状态	2	12	1
ADD	A,#data	立即数与 A 中的数相加,结果在 A 中,影响 PSW 位的状态	2	12	1
ADD	A,@Ri	@Ri 中与 A 中的数相加,结果在 A 中,影响 PSW 位的状态	1	12	1
ADDC	A,Rn	Rn 中与 A 中的数带进位加,结果在 A 中,影响 PSW 位的状态	1	12	1
ADDC	A,direct	direct 中与 A 中的数带进位加,结果在 A 中,影响 PSW 位的状态	2	12	1
ADDC	A,#data	立即数与 A 中的数带进位加,结果在 A 中,影响 PSW 位的状态	2	12	1
ADDC	A,@Ri	@Ri 中与 A 中的数带进位加,结果在 A 中,影响 PSW 位的状态	1	12	1
SUBB	A,Rn	Rn 中与 A 中的数带借位减,结果在 A 中,影响 PSW 位的状态	1	12	1
SUBB	A,direct	direct 中与 A 中的数带借位减,结果在 A 中,影响 PSW 位的状态	2	12	1
SUBB	A,#data	立即数与 A 中的数带借位减,结果在 A 中,影响 PSW 位的状态	2	12	1
SUBB	A,@Ri	@Ri 中与 A 中的数带借位减,结果在 A 中,影响 PSW 位的状态	1	12	1
INC	A	A 中的数加 1	1	12	1
INC	Rn	Rn 中的数加 1	1	12	1
INC	direct	direct 中的数加 1	2	12	1

续表

汇编指令		操作说明	代码长度/B	指令周期	
				Tosc	Tm
INC	@Ri	@Ri 中的数加 1	1	12	1
INC	DPTR	DPTR 中的数加 1	1	24	2
DEC	A	A 中的数减 1	1	12	1
DEC	Rn	Rn 中的数减 1	1	12	1
DEC	direct	direct 中的数减 1	2	12	1
DEC	@Ri	@Ri 中的数减 1	1	12	1
MUL	AB	A、B 中的两无符号数相乘，结果低 8 位在 A 中，高 8 位在 B 中	1	48	4
DIV	AB	A、B 中的两无符号数相除，商在 A 中，余数在 B 中	1	48	4
DA	A	十进制调整，对 BCD 码十进制加法运算结果调整	1	12	1

3. 逻辑运算类指令(共 24 条)

汇编指令		操作说明	代码长度/B	指令周期	
				Tosc	Tm
ANL	A,Rn	Rn 中与 A 中的数相“与”，结果在 A 中	1	12	1
ANL	A,direct	direct 中与 A 中的数相“与”，结果在 A 中	2	12	1
ANL	A,#data	立即数与 A 中的数相“与”，结果在 A 中	2	12	1
ANL	A,@Ri	@Ri 中与 A 中的数相“与”，结果在 A 中	1	12	1
ANL	direct,A	A 和 direct 中的数进行“与”操作，结果在 direct 中	2	12	1
ANL	direct,#data	常数和 direct 中的数进行“与”操作，结果在 direct 中	3	24	2
ORL	A,Rn	Rn 中和 A 中的数进行“或”操作，结果在 A 中	1	12	1
ORL	A,direct	direct 中和 A 中的数进行“或”操作，结果在 A 中	2	12	1
ORL	A,#data	立即数和 A 中的数进行“或”操作，结果在 A 中	2	12	1
ORL	A,@Ri	@Ri 中和 A 中的数进行“或”操作，结果在 A 中	1	12	1
ORL	direct,A	A 中和 direct 中的数进行“或”操作，结果在 direct 中	2	12	1
ORL	direct,#data	立即数和 direct 中的数进行“或”操作，结果在 direct 中	3	24	2
XRL	A,Rn	Rn 中和 A 中的数进行“异或”操作，结果在 A 中	1	12	1

续表

汇编指令		操作说明	代码长度/B	指令周期	
				Tosc	Tm
XRL	A,direct	direct 中和 A 中的数进行“异或”操作,结果在 A 中	2	12	1
XRL	A,#data	立即数和 A 中的数进行“异或”操作,结果在 A 中	2	12	1
XRL	A,@Ri	@Ri 中和 A 中的数进行“异或”操作,结果在 A 中	1	12	1
XRL	direct,A	A 中和 direct 中的数进行“异或”操作,结果在 direct 中	2	12	1
XRL	direct,#data	立即数和 direct 中的数进行“异或”操作,结果在 direct 中	3	24	2
RR	A	A 中的数循环右移(移向低位),D0 移入 D7	1	12	1
RRC	A	A 中的数带进位循环右移,D0 移入 C,C 移入 D7	1	12	1
RL	A	A 中的数循环左移(移向高位),D7 移入 D0	1	12	1
RLC	A	A 中的数带进位循环左移,D7 移入 C,C 移入 D0	1	12	1
CLR	A	A 中数清零	1	12	1
CPL	A	A 中数取反	1	12	1

4. 程序转移类指令(共 17 条)

汇编指令		操作说明	代码长度/B	指令周期	
				Tosc	Tm
(1) 无条件转移指令(共 9 条)					
LJMP	addr16	长转移,程序转到 addr16 指示的地址处	3	24	2
AJMP	addr11	短转移,程序转到 addr11 指示的地址处	2	24	2
SJMP	rel	相对转移,程序转到 rel 指示的地址处	2	24	2
LCALL	addr16	长调用,程序调用 addr16 处的子程序	3	24	2
ACALL	addr11	短调用,程序调用 addr11 处的子程序	2	24	2
JMP	@A+DPTR	程序散转,程序转到 DPTR 为基址,A 为偏移地址处	1	24	2
RETI		中断返回	1	24	2
RET		子程序返回	1	24	2
NOP		空操作	1	12	1
(2) 条件转移指令(共 8 条)					
JZ	rel	A 中的数为零,程序转到相对地址 rel 处	2	24	2

续表

汇编指令		操作说明	代码长度/B	指令周期	
				Tosc	Tm
JNZ	rel	A 中的数不为零，程序转到相对地址 rel 处	2	24	2
DJNZ	Rn,rel	Rn 中的数减 1 不为零，程序转到相对地址 rel 处	2	24	2
DJNZ	direct,rel	direct 中的数减 1 不为零，程序转到相对地址 rel 处	3	24	2
CJNE	A,#data,rel	#data 与 A 中的数不等转至 rel 处。C=1，data>(A)；C=0，data<(A)	3	24	2
CJNE	A,direct,rel	direct 与 A 中的数不等转至 rel 处。C=1，data>(A)；C=0，data<(A)	3	24	2
CJNE	Rn,#data,rel	#data 与 Rn 中的数不等转至 rel 处。C=1，data>(Rn)；C=0，data<(Rn)	3	24	2
CJNE	@Ri,#data,rel	#data 与@Ri 中的数不等转至 rel 处。C=1，data>(@Ri)；C=0，data<(@Ri)	3	24	2

5. 布尔指令(共 17 条)

汇编指令		操作说明	代码长度/B	指令周期	
				Tosc	Tm
(1) 位操作指令(共 12 条)					
MOV	C,bit	bit 中状态送入 C 中	2	12	1
MOV	bit,C	C 中状态送入 bit 中	2	24	2
ANL	C,bit	bit 中状态与 C 中状态相“与”，结果在 C 中	2	24	2
ANL	C,/bit	bit 中状态取反与 C 中状态相“与”，结果在 C 中	2	24	2
ORL	C,bit	bit 中状态与 C 中状态相“或”，结果在 C 中	2	24	2
ORL	C,/bit	bit 中状态取反与 C 中状态相“或”，结果在 C 中	2	24	2
CLR	C	C 中状态清零	1	12	1
SETB	C	C 中状态置 1	1	12	1
CPL	C	C 中状态取反	1	12	1
CLR	bit	bit 中状态清零	2	12	1
SETB	bit	bit 中状态置 1	2	12	1
CPL	bit	bit 中状态取反	2	12	1
(2) 位条件转移指令(共 5 条)					
JC	rel	进位位为零时，程序转至 rel	2	24	2
JNC	rel	进位位不为零时，程序转至 rel	2	24	2
JB	bit,rel	bit 状态为 1 时，程序转至 rel	3	24	2
JNB	bit,rel	bit 状态不为 1 时，程序转至 rel	3	24	2
JBC	bit,rel	bit 状态为 1 时，程序转至 rel，同时 bit 位清零	3	24	2

3.3 指令的应用例子

例 1 七段 LED 数码管显示程序

图 3.10 为一个采用 6 个七段 LED 数码管显示的时钟电路，其采用 AT89C2051 单片机最小化应用设计，LED 显示采用动态扫描方式实现，P1 口输出段码数据，P3.0～P3.5 口作扫描输出，P3.7 接按钮开关。为了提供 LED 数码管的驱动电流，用三极管 9012 作电源驱动输出。为了提高秒计时的精确性，采用 12MHz 晶振。

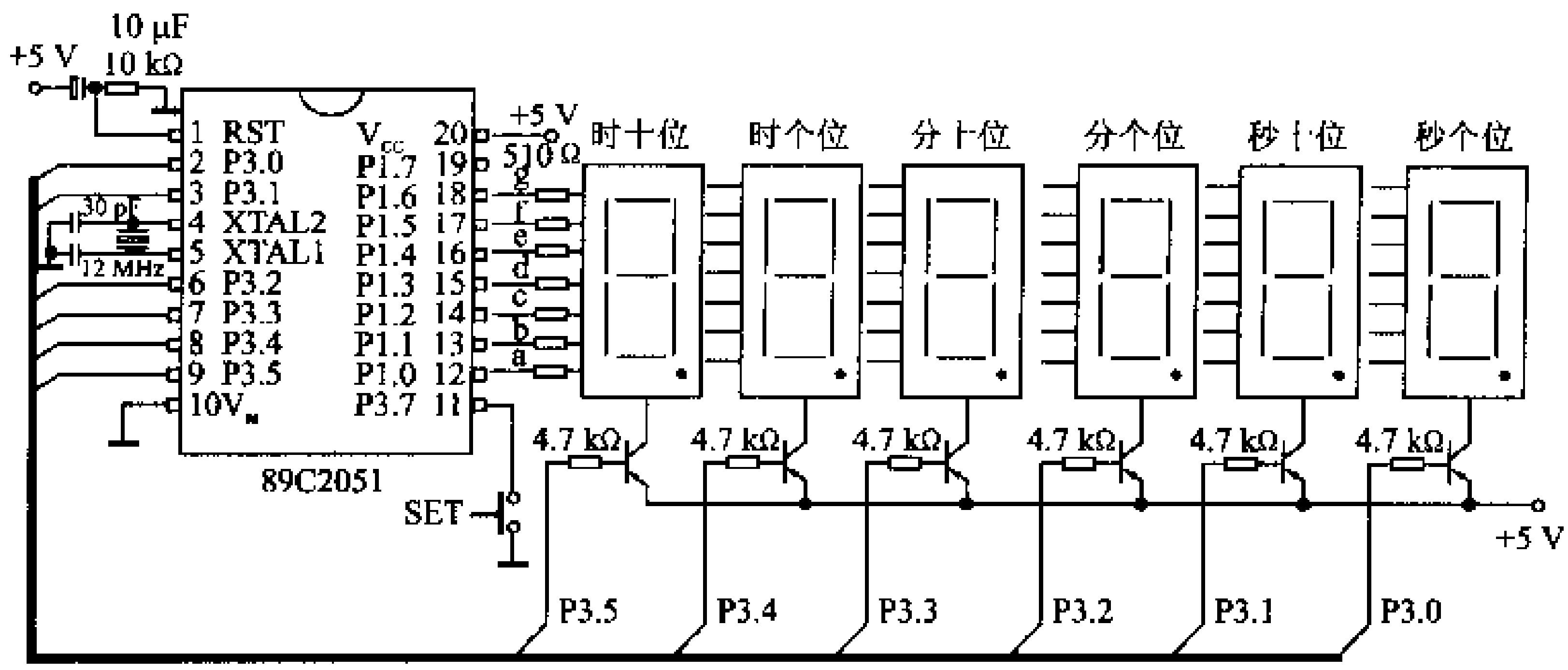


图 3.10 采用 89C2051 的六位时钟电路

数码管显示的数据存放在内存单元 70H～75H 中，其中 70H～71H 存放秒数据，72H～73H 存放分数据，74H～75H 存放时数据，每一地址单元内均为十进制 BCD 码。由于采用软件动态扫描实现数据显示功能，显示用十进制 BCD 码数据的对应段码存放在 ROM 表中。显示时，先取出 70H～75H 某一地址中的数据，然后查得对应的显示用段码从 P1 口输出。P3 口将对应的数码管选中，就能显示该地址单元的数据值。

以下是动态扫描法实现数据显示功能的程序：

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;              显示程序                 ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

DISPLAY: MOV R1,#70H          ;显示数据首址
         MOV R5,#0FEH         ;扫描端口初值
   PLAY: MOV A,R5             ;将 R5 中数据移入 A 中
         MOV P1,#0FFH         ;清原数据
         MOV P3,A             ;扫描端口赋值
         MOV A,@R1            ;取显示数据
         MOV DPTR,#TAB        ;段码表表址放入数据指针
         MOVC A,@A+DPTR       ;查段码
         MOV P1,A             ;段码数据放到 P1 口
```

```
            LCALL DL1MS            ;数据显示 1 ms
            INC R1                 ;存放显示数据地址加 1
            MOV A,R5               ;扫描端口值放入 A
            JNB ACC.5,ENDOUT       ;A 中值为 11011111(B)时结束
            RL A                   ;A 中数据循环左移一位
            MOV R5,A               ;A 中数据放回 R5 中
            AJMP PLAY              ;跳至 PLAY 循环
   ENDOUT:  MOV P3,#0FFH           ;退出时 P3 口复位
            MOV P1,#0FFH           ;退出时 P1 口复位
            RET                    ;子程序结束
      TAB:  DB 0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H,0FFH
;    共阳段码表  "0"  "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9" "熄灭符"
;
            ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            ;;           1 ms 延时程序            ;;
            ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
DL1MS:  MOV     R6,#14H      ; R6 赋初值 20(十进制)
DL1:    MOV     R7,#19H      ; R7 赋初值 25(十进制)
DL2:    DJNZ    R7,DL2       ; R7 减 1 不为零转 DL2
        DJNZ    R6,DL1       ; R6 减 1 不为零转 DL1
        RET                  ;子程序结束
```

思考与练习

1. 请区别汇编指令、指令代码、指令周期、指令长度。
2. 80C51 指令系统有哪些寻址方式，相应的空间在何处？
3. 片内 RAM20H～2FH 的 128 个位地址与直接地址 00H～7FH 形式完全相同，如何在指令中区分出位寻址操作和直接地址操作？
4. 什么是源操作数？什么是目的操作数？通常在指令中如何区别？
5. 查表指令是在什么空间上的寻址操作？
6. 80C51 中有 LJMP、LCALL，为何还设置了 AJMP、ACALL？
7. 查表指令中使用了基址加变址的寻址方式，请问 DPTR、PC 分别代表什么地址？
8. 比较不等转移指令 CJNE 有哪些扩展功能？

第 4 章　单片机基本单元结构与操作原理

4.1　定时器/计数器的基本结构与操作方式

4.1.1　定时器/计数器的基本组成

89C51 中有两个 16 位的加计数定时器/计数器 T0、T1，其组成如图 4.1 所示。

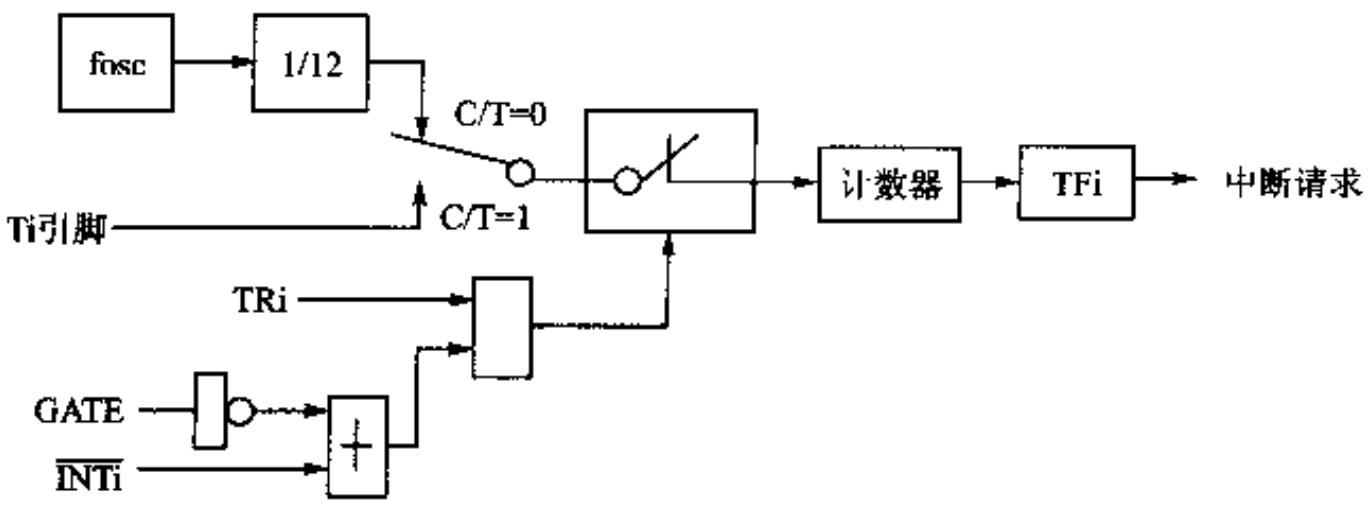


图 4.1　89C51 中定时器/计数器的基本组成

说明：

(1) 计数器由两个 8 位的加计数器 TLi 和 THi 组成，在不同的方式下，其组成结构不同。

(2) 计数输入可选择振荡器的 12 分频计数，也可从端口 Ti 对外部脉冲计数。

(3) 控制逻辑：当 GATE＝0 时，由 TRi 控制计数器的启停；当 GATE＝1，且 TRi＝1 时，计数器由外部引脚$\overline{\text{INTi}}$控制启停(高电平开启)。

(4) 计数器的溢出管理：当计数器溢出时，溢出中断请求标志位 TFi 置位，并请求中断，中断响应后 TFi 自动清零。

4.1.2　定时器/计数器的 SFR

参与定时器/计数器管理的 SFR 有方式寄存器 TMOD 和控制寄存器 TCON。

1. TMOD 方式寄存器

TMOD 方式寄存器的格式如下：

GATE	C/T	M1	M0	GATE	C/T	M1	M0
高 4 位 T1 控制用				低 4 位 T0 控制用			

说明：

TMOD 为不可位寻址 SFR，地址为 89H，其低 4 位控制 T0，高 4 位控制 T1，各位的意义如下：

M1、M0:方式控制。00 为方式 0,为 13 位计数器方式;01 为方式 1,为 16 位计数器方式;10 为方式 2,为 8 位自动重装初值方式;11 为方式 3,为两个 8 位计数器与波特率发生器工作方式。

C/T:计数/定时方式选择。C/T=1 时,对外部计数;C/T=0 时,对内部振荡器 12 分频计数。

GATE:控制方式选择。当 GATE=0 时,计数器由内部 TRi 控制启停;当 GATE=1 时,计数器由 TRi 和外部引脚$\overline{\text{INT}i}$一起控制。

2. TCON 控制寄存器

TCON 控制寄存器的格式如下:

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
用于定时器				用于外中断			

说明:

(1) TCON 是一个可位寻址的寄存器,字节地址为 88H。

(2) 高 4 位用于定时器控制,低 4 位用于外中断控制。

(3) 各位意义如下:

TF1:定时器/计数器 T1 溢出标志。溢出时自动置 1,中断响应后自动复位,也可用软件复位。

TR1:定时器/计数器 T1 运行控制位。TR1=0 时停止,TR1=1 时开启。

TF0:定时器/计数器 T0 溢出标志。溢出时自动置 1,中断响应后自动复位,也可用软件复位。

TR0:定时器/计数器 T0 运行控制位。TR0=0 时停止,TR0=1 时开启。

IE1:外中断 1 中断请求标志位。CPU 响应中断后自动复位。

IT1:外中断 1 触发类型选择位。IT1=0 时为电平触发,IT1=1 时为下降沿边沿触发。

IE0:外中断 0 中断请求标志位。CPU 响应中断后自动复位。

IT0:外中断 0 触发类型选择位。IT0=0 时为电平触发,IT0=1 时为下降沿边沿触发。

(4) 定时器/计数器 T0、T1 的数据寄存器为 TH0、TL0 和 TH1、TL1。T0 和 T1 各有一个 16 位的寄存器,由高 8 位和低 8 位组成,可以进行读写操作,复位时这四个寄存器全部清零。

4.1.3　定时器/计数器的工作方式

定时器/计数器的工作方式有以下四种:

1. 方式 0

当 TMOD 中的 M0=0,M1=0 时,为 13 位计数或定时方式,其中 TLi 使用低 5 位,其结构如图 4.2 所示。

2. 方式 1

当 TMOD 中的 M0=1,M1=0 时,为 16 位计数或定时方式,其结构如图 4.3 所示。

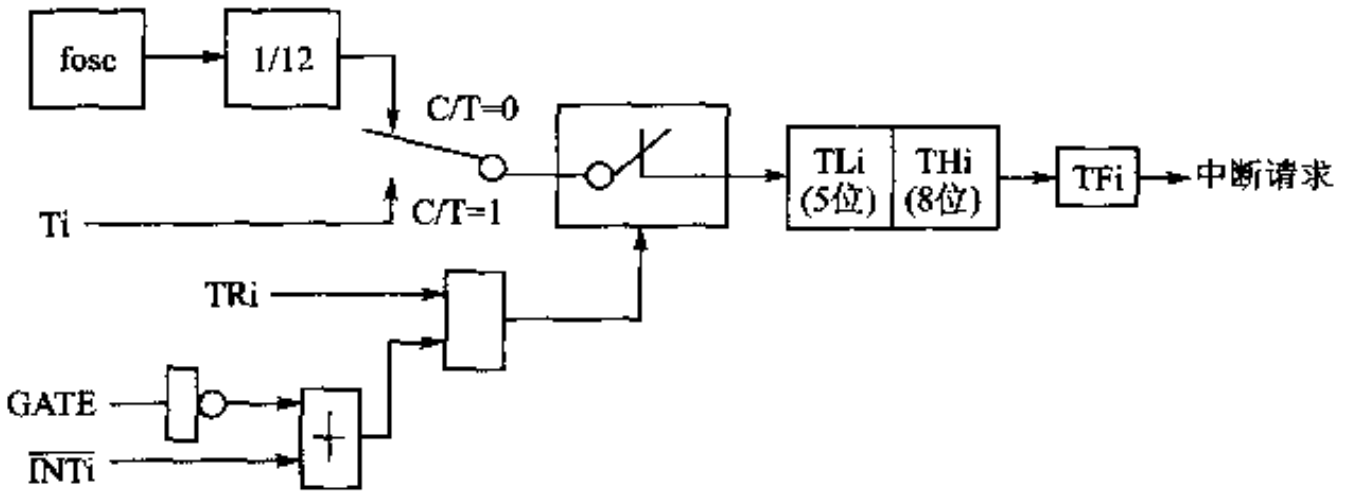


图 4.2 方式 0 时 T0、T1 的结构图

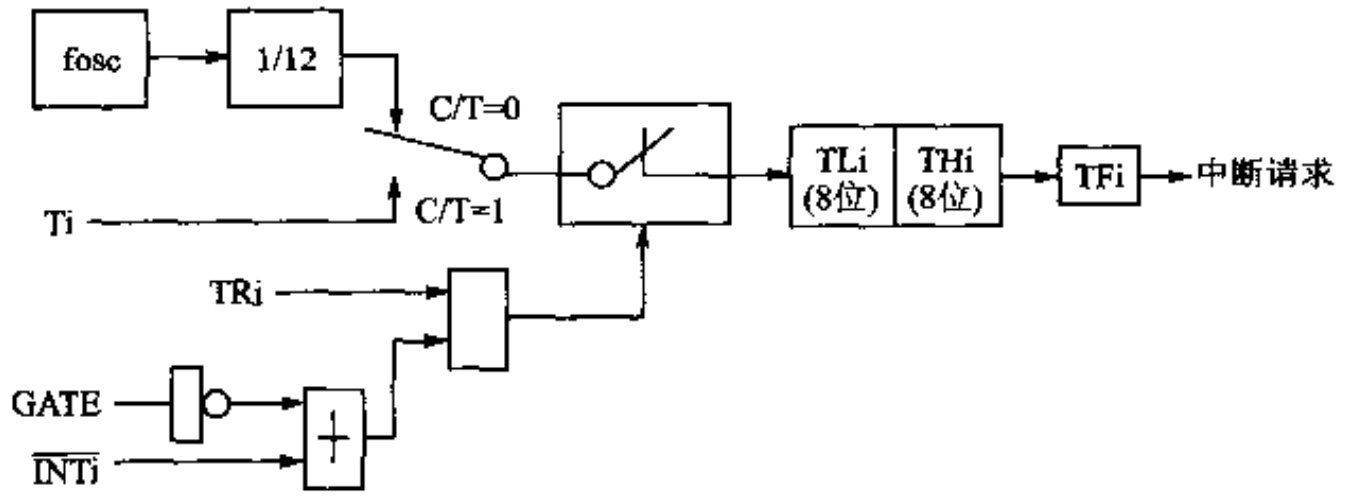


图 4.3 方式 1 时 T0、T1 的结构图

3. 方式 2

当 TMOD 中的 M0=0，M1=1 时，为 8 位自动重装初值计数或定时方式，其结构如图 4.4 所示。

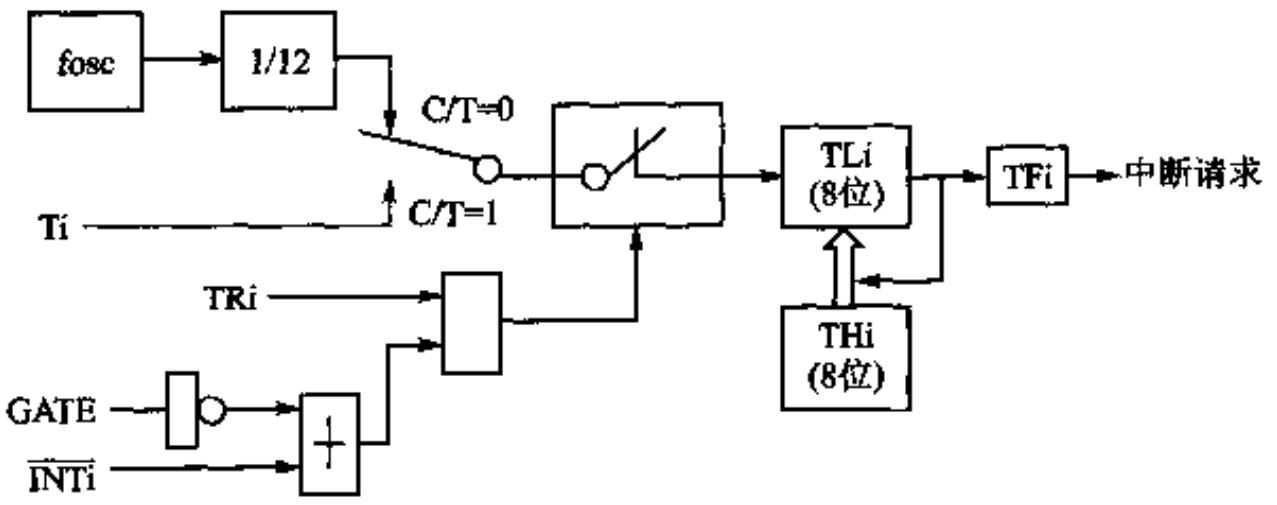


图 4.4 方式 2 时 T0、T1 的结构图

在方式 2 时，将 16 位计数器分成两个 8 位的计数器，THi 用来存放初值。当计数器溢出时，一方面将 TFi 置 1，申请中断；而另一方面自动将 THi 的值装入 TLi。

4. 方式 3

T0 为方式 3 时，T1 作为波特率发生器，其 TF1、TR1 资源出借给 T0 使用，而 T0 可以构成两个独立的结构，其中 TL0 构成一个完整的 8 位定时器/计数器，而 TH0 则是一个仅能对晶振频率 12 分频的定时器，其结构如图 4.5 所示。T1 作波特率发生器时，可以设置成方式 0、1 或 2，用在任何不需要中断控制的场合。一般 T1 作波特率发生器时，常设置成方式 2 的自动重装模式，其结构如图 4.6 所示。

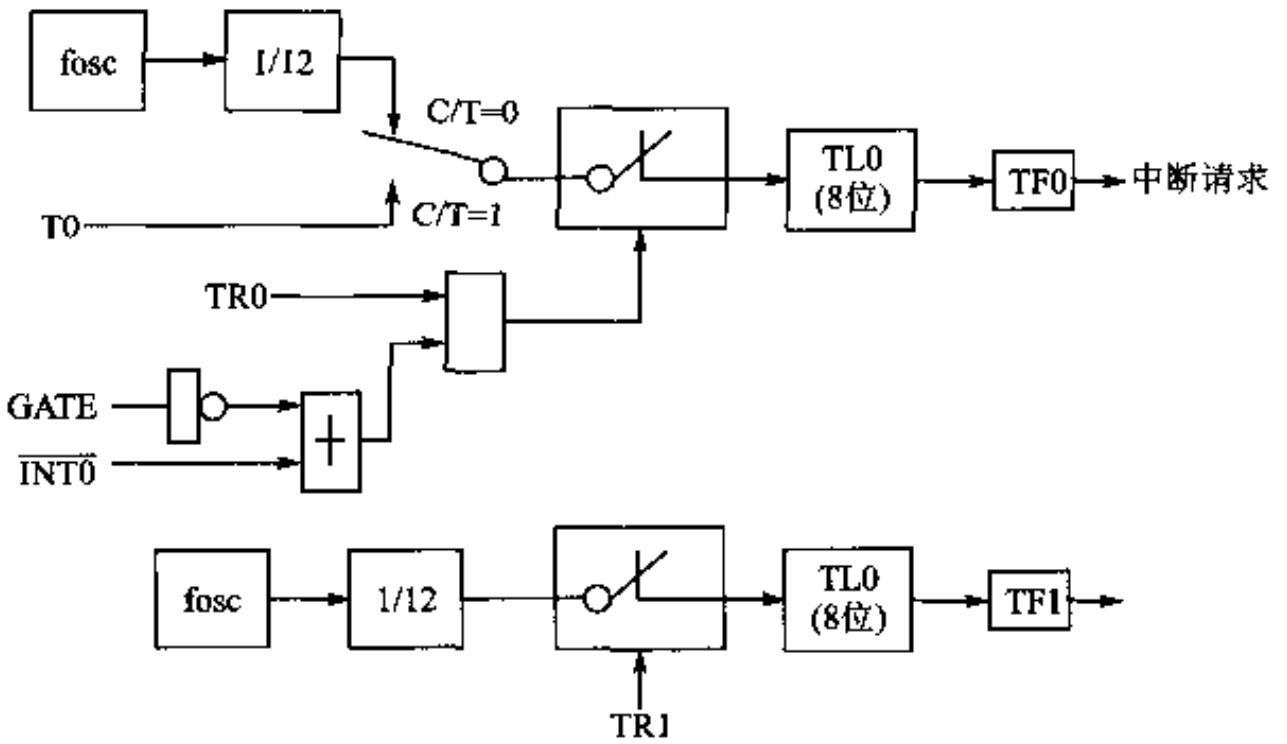


图 4.5　方式 3 时 T0 的结构图

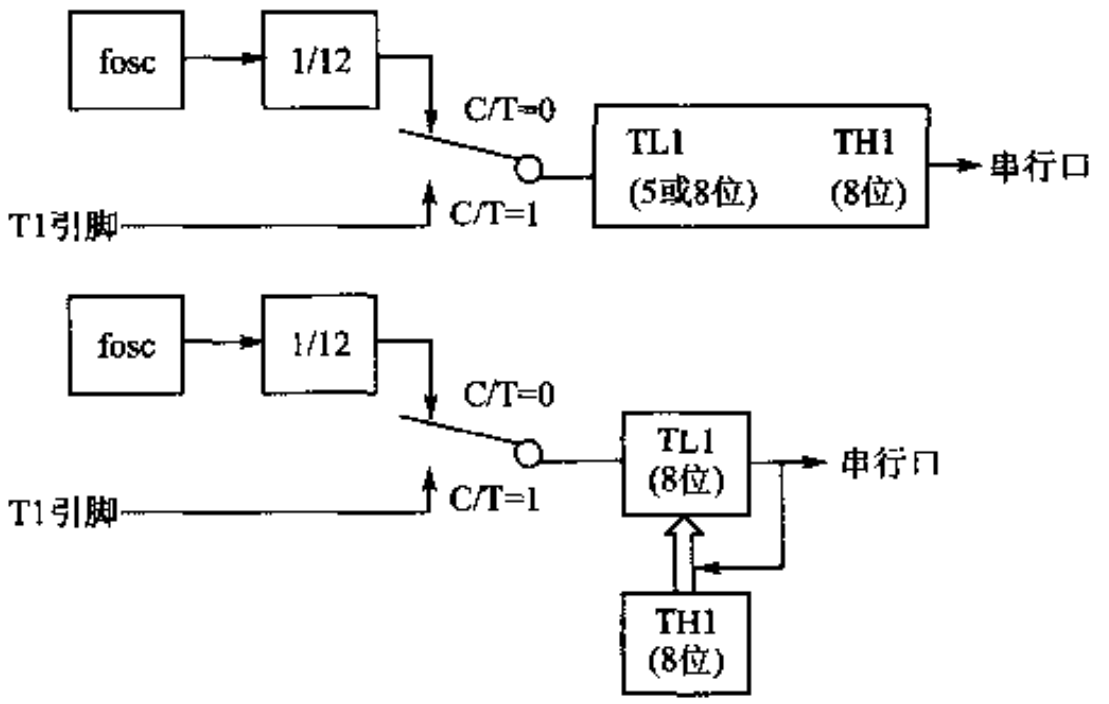


图 4.6　T0 为方式 3 时,T1 为波特率发生器时的 T1 结构图

4.1.4　定时器/计数器的编程和使用

1. 定时器/计数器溢出率的计算

由公式 $t = Tc \times (2^L - TC)$,先求出 t,然后再求溢出率。

式中:t 为定时时间(μs);

Tc 为机器周期,$Tc = 12 \div f_{osc}$;

L 为计数器位数,13 位时 $2^L = 8\ 192$,16 位时 $2^L = 65\ 536$,8 位时 $2^L = 256$;

TC 为定时器/计数器初值。

定时时间的倒数即为溢出率 $= 1/t$。

如:设晶振为 12 MHz,求定时器定时时间为 5 ms 时的初值。

(1) 采用 13 位计数器时

TC = 8 192 − 5 000 = 3 192 = 0C78H = 01100011 11000B

```
MOV     TH0,#63H
MOV     TL0,#18H
```

(2) 采用 16 位计数器时

TC＝65 536－5 000＝60 536＝EC78H

```
MOV     TH0,#0ECH
MOV     TL0,#78H
```

(3) 采用 8 位计数器时

12 MHz 时钟时，8 位计数器的最大定时时间为 256μs，一次定时 5 ms 不能达到要求，在中断程序中可采用多次溢出累加法。

2. 定时器/计数器的编程

定时器/计数器的编程步骤如下：

(1) 设置 TMOD 方式值，只能用字节寻址，如：

```
MOV     TMOD,#11H     ;两个 16 位定时器
MOV     TMOD,#22H     ;两个 8 位自动重装初值定时器
MOV     TMOD,#51H     ;T1 为 16 位计数器，T0 为 16 位定时器
```

(2) 将定时时间常数和初值放入 TH 和 TL，只能字节寻址，如：

```
MOV     TH0,#07H
MOV     TL0,#0FFH
MOV     TH1,#01H
MOV     TL1,#0F8H
```

(3) 定时器中断的开放与禁止，一般用位寻址，如：

```
SETB    EA
SETB    ET0
SETB    ET1
CLR     EA
CLR     ET0
CLR     ET1
```

(4) 启动或关闭定时计数器，一般用位寻址，如：

```
SETB    TR0
SETB    TR1
CLR     TR0
CLR     TR1
```

4.1.5　定时器应用举例

例 1　试设定定时器/计数器 T0 为计数方式 2，当 T0 引脚出现负跳变时，向 CPU 申请中断。

解：当 T0 引脚出现负跳变时，申请中断，可设初值为 0FFH，当第一个低电平来时，即发生溢出中断申请。程序如下：

```
ORG     0000H         ;主程序入口地址
LJMP    MAIN          ;跳至 MAIN 执行
ORG     00BH          ;定时器 T0 溢出中断服务程序入口地址
```

```
        LJMP    INTT0           ;跳至中断服务程序 INTT0 执行
MAIN:   MOV     TMOD,#06H       ;T0 为 8 位自动重装初值计数器
        MOV     TL0,#0FFH       ;初值为#0FF
        MOV     TH0,#0FFH;
        SETB    ET0             ;允许 T0 溢出中断
        SETB    EA              ;总中断允许开放
        SETB    TR0             ;开启定时器
        AJMP    $               ;等待
INTT0:  CLR     ET0             ;关定时器 T0 中断
        ⋮                       ;处理程序
        SETB    ET0             ;允许 T0 中断
        END                     ;程序结束
```

例 2　如图 4.7 所示，利用 T0 在 P1.0 端口产生 500 Hz 的方波对称脉冲(12 MHz 晶振)。

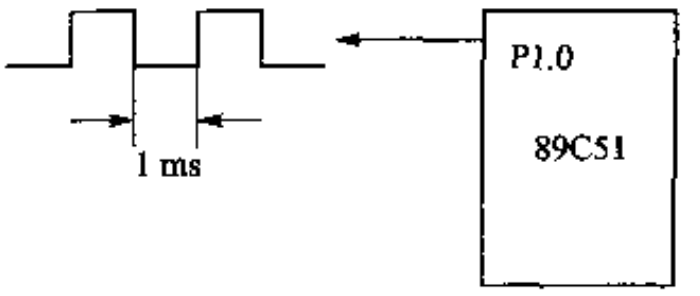


图 4.7　产生 500 Hz 的方波对称脉冲

解：设 T0 为 16 位定时器模式，利用查询法设计程序，溢出周期为 1 ms，则初值 TC＝65 536－1 000＝64 536＝FC18H。程序如下：

```
        ORG     0000H
        LJMP    MAIN
MAIN:   MOV     TMOD,#01H       ;设 T0 为 16 位定时器模式
        MOV     TL0,#18H        ;赋初值
        MOV     TH0,#0FCH       ;赋初值
        SETB    TR0             ;开启定时器
LOOP:   JBC     TF0,CPLP        ;TF0 为 1,转 CPLP 并将 TF0 清为 0
        AJMP    LOOP            ;TF0 为 0 则转 LOOP 循环等待
CPLP:   MOV     TL0,#18H        ;重装初值
        MOV     TH0,#0FCH       ;
        CPL     P1.0            ;P1.0 端口状态取反
        AJMP    LOOP            ;转 LOOP 再循环等待
        END                     ;结束
```

例 3　如图 4.8 所示，如果要在例 2 中产生周期为 3ms、占空比为 1∶2的脉冲波，应该怎样修改程序。

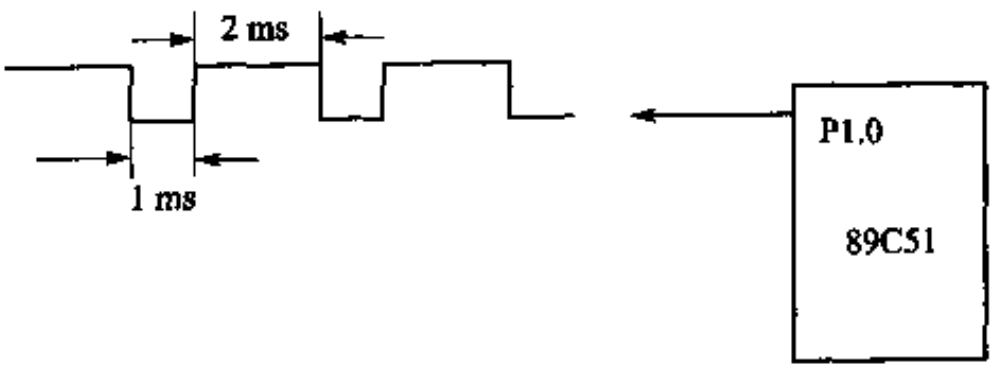


图 4.8　产生周期为 3ms，占空比为 1∶2 的脉冲波

解：可在程序中加入 P1.0 端口的状态判断，当 P1.0 为高电平时，需溢出二次才对端口取反。程序如下：

```
        ORG     0000H
        LJMP    MAIN
MAIN:   MOV     TMOD,#01H       ;T0 为 16 位定时模式
        MOV     TL0,#18H        ;定时器赋初值
        MOV     TH0,#0FCH       ;定时器赋初值
        MOV     R2,#02H         ;R2 赋初值
        SETB    TR0             ;开启定时器
LOOP:   JBC     TF0,CPLP        ;TF0 为 1(定时时间到),转 CPLP 并将 TF0 清为 0
        AJMP    LOOP            ;TF0 为 0 则转 LOOP 循环等待
CPLP:   MOV     TL0,#18H        ;定时器重装初值
        MOV     TH0,#0FCH       ;定时器重装初值
        JB      P1.0,CPLP1      ;P1.0 口为 1 则转 CPLP1
        CPL     P1.0            ;P1.0 口为 0,则取反(变 1)
        MOV     R2,#02H         ;R2 重赋初值
        AJMP    LOOP            ;转 LOOP 等待定时时间到
CPLP1:  DJNZ    R2,LOOP         ;2 ms 未到转 LOOP
        CPL     P1.0            ;2 ms 到对 P1.0 口取反(变为 0)
        AJMP    LOOP            ;转 LOOP 等待定时时间到
        END                     ;程序结束
```

4.2 中断系统的基本原理与操作方式

89C51 中断系统有五个中断源，其中有两个外部中断源、两个定时中断和一个串行中断，每个中断源都可以选择两个优先级。

4.2.1 中断系统的基本组成

(1) 中断：程序执行过程中，允许外部或内部事件通过硬件打断程序的执行，使其转向处理事件的中断服务程序中去，完成后继续执行原来的程序，这样的过程叫中断过程。

(2) 中断源：能产生中断的外部事件和内部事件叫中断源。

(3) 中断优先级：几个中断源同时申请中断时，CPU 必需区分哪个中断源更重要，从而确定优先处理哪个中断事件。89C51 中断优先级从高到低为 INT0、T0、INT1、T1、串口中断。

(4) 中断请求标志：当中断事件发生时，相应的中断请求标志 IE0、IE1、FT0、FT1、TI/RI 被置 1。

(5) 中断允许：有中断总允许 EA 和各中断源允许 EX0、ET0、EX1、ET1、ES，当被置 1 时开放中断。

图 4.9 为 89C51 中断系统结构示意图。

4.2.2 中断系统中的 SFR

与中断系统有关的 SFR 有 SCON、TCON、IE 和 IP。

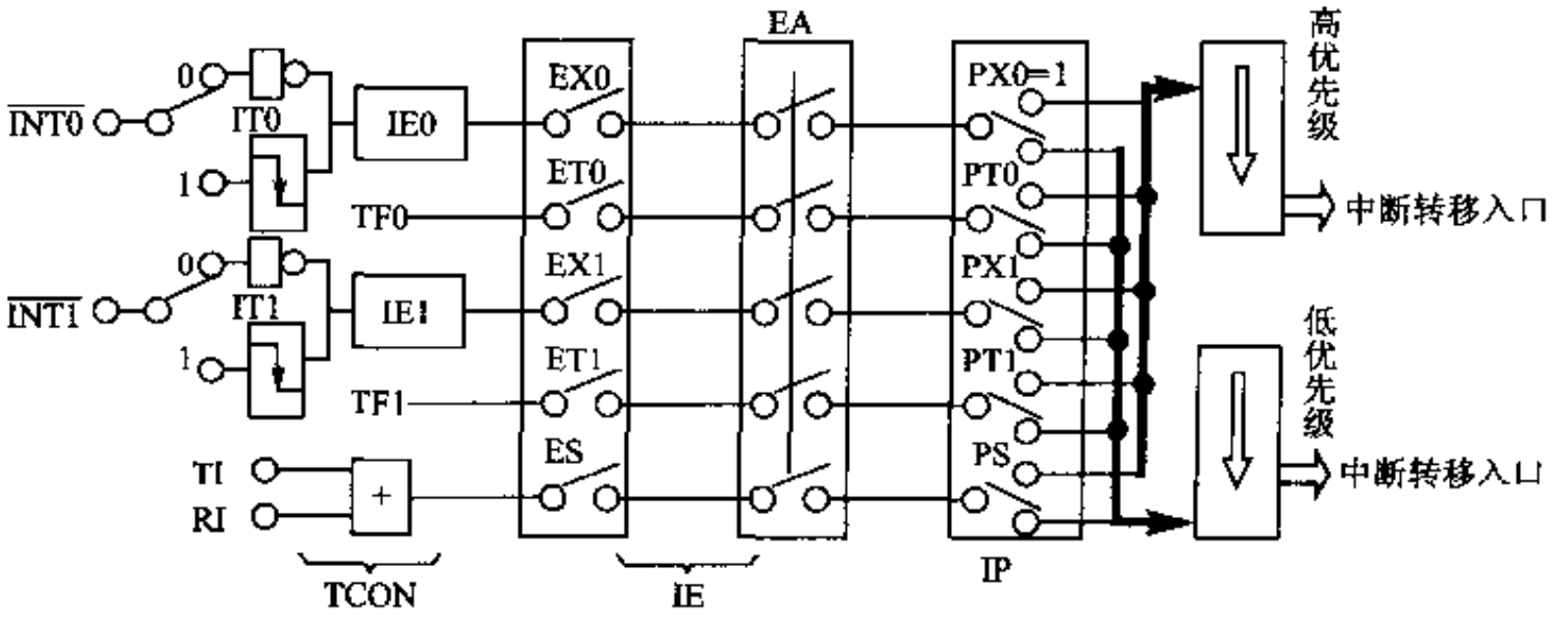


图 4.9　89C51 中断系统结构示意图

1. 串行口控制寄存器 SCON

SCON 为可位寻址寄存器，直接地址为 98H，其各位如下：

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

TI：发送中断标志。当发送数据完毕时，TI＝1，表示帧发送完毕，请求中断，也可供查询。TI 只能由程序清零。

RI：接收中断标志。当接收数据完毕时，TI＝1，表示接收完一帧数据，请求中断，也可供查询。RI 只能由程序清零。

2. TCON 控制寄存器

TCON 是一个可位寻址的寄存器，字节地址为 88H，高 4 位用于定时器控制，低 4 位用于外中断控制，其各位如下：

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
用于定时器				用于外中断			

各位意义如下：

TF1：定时器/计数器 T1 溢出标志。溢出时自动置 1，中断响应后自动复位，也可用软件复位。

TF0：定时器/计数器 T0 溢出标志。溢出时自动置 1，中断响应后自动复位，也可用软件复位。

IE1：外中断 1 中断请求标志位。CPU 响应中断后自动复位。

IT1：外中断 1 触发类型选择位。IT1＝0 时 为电平触发，IT1＝1 时为下降沿边沿触发。

IE0：外中断 0 中断请求标志位。CPU 响应中断后自动复位。

IT0：外中断 0 触发类型选择位。IT0＝0 时 为电平触发，IT0＝1 时为下降沿边沿触发。

3. 中断允许寄存器 IE

IE 为可位寻址寄存器，直接地址为 A8H，用于中断的开放与关闭，其各位如下：

EA			ES	ET1	EX1	ET0	EX0

各位意义如下：

EA：CPU 中断总允许控制。当 EA＝1 时，CPU 开放中断。

EX1：EX1＝1 时为允许外部中断 INT1 中断。

ET1：ET1＝1 时为允许定时器 T1 溢出中断。

EX0：EX0＝1 时为允许外部中断 INT0 中断。

ET0：ET0＝1 时为允许定时器 T0 溢出中断。

ES：ES＝1 时为允许串行口发送/接收中断。

4. 中断优先级管理寄存器 IP

IP 为可位寻址寄存器，直级地址为 B8H，用来设定优先级别。置 1 时为高优先级，清零时为低优先级，其各位如下：

			PS	PT1	PX1	PT0	PX0

各位意义如下：

PX0、PX1：外部中断源 INT0、INT1 优先级选择位。

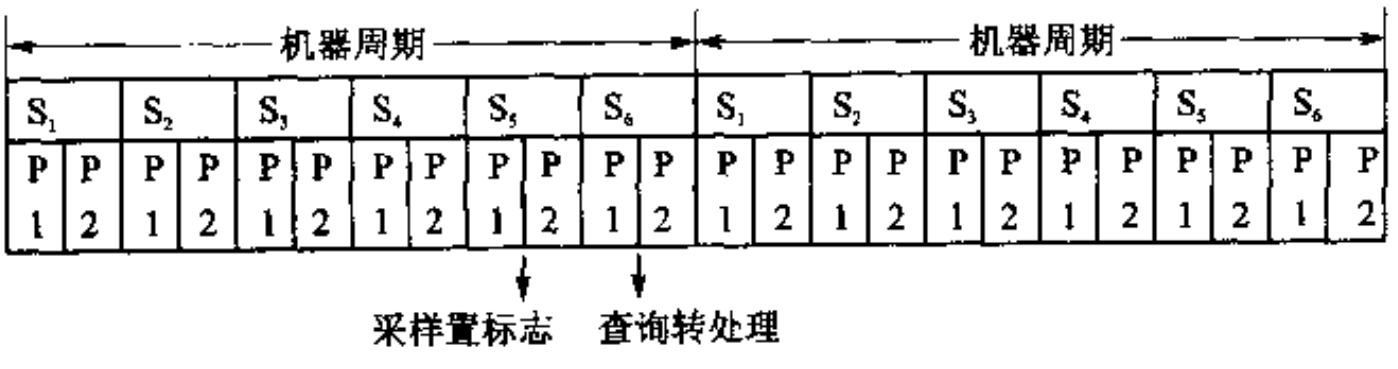
PT0、PT1：定时器/计数器溢出中断优先级选择位。

PS：串行口发送/接收中断优先级选择位。

4.2.3　中断响应的自主操作过程

1. CPU 的中断查询

CPU 的中断查询各位如下：



CPU 在每个机器周期的 S5P2 期间，各中断源被采样并设置相应的中断标志，在每个机器周期的 S6P2 状态中，按优先级顺序查询中断源的中断标志，并处理请求的中断源，且在下一个机器周期的 S1 状态中，响应最高级的中断请求。但以下情况除外：

(1) CPU 正在处理相同或更高级的中断源；

(2) 多机器周期指令中，还没有执行到最后一个机器周期；

(3) 正在执行中断系统的 SFR 操作，如 RETI 及访问 IE、IP 等的操作时，要延时一条指令。

2. 中断响应中的 CPU 自主操作

在中断响应中，CPU 要完成以下自主操作：

(1) 置位相应的优先级状态触发器，以标明响应所中断的优先级别；

(2) 中断源标志清零(TI、RI 除外)；

(3) 中断点地址装入堆栈保护(不保护 PSW)；

(4) 中断入口地址装入 PC，以便使程序转到中断入口地址处。

3. 中断返回时 CPU 的自主操作

CPU 执行到 RETI 中断返回指令时，产生以下自主操作：

(1) 优先级触发器清零；

(2) 断点地址装入 PC，以使程序返回到断点处。

4.2.4　应用练习

例 1　外部中断源的扩展方法

外部中断源的扩展接口电路如图 4.10 所示。

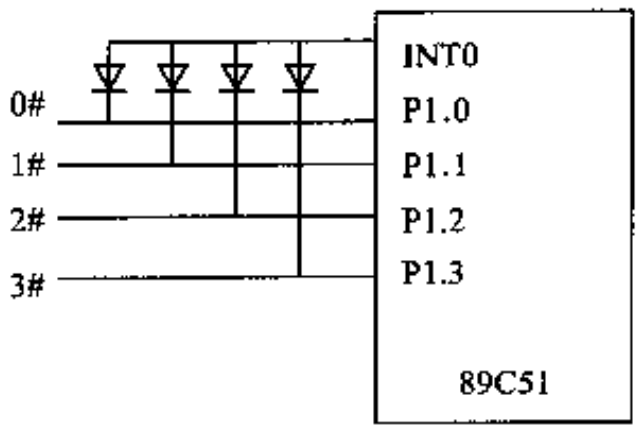


图 4.10　外部中断源的扩展接口电路

外部中断源的扩展程序如下：

```
ORG     0003H                ;外中断 0 入口地址
LJMP    INTEX0               ;跳到 INTEX0 执行
;
INTEX0: PUSH  PSW            ;PSW 入堆栈保护
        JNB   P1.0,INTFUN0   ;P1.0 为 0 转 INTFUN0
        JNB   P1.1,INTFUN1   ;P1.1 为 0 转 INTFUN1
        JNB   P1.2,INTFUN2   ;P1.2 为 0 转 INTFUN2
        JNB   P1.3,INTFUN3   ;P1.3 为 0 转 INTFUN3
INTOUT: POP   PSW            ;恢复 PSW
        RETI                 ;中断返回
INTFUN0: ……                  ;0# 中断处理程序
        ……
        LJMP  INTOUT
INTFUN1: ……                  ;1# 中断处理程序
        ……
        LJMP  INTOUT
INTFUN2: ……                  ;2# 中断处理程序
        ……
        LJMP  INTOUT
INTFUN3: ……                  ;3# 中断处理程序
        ……
        LJMP  INTOUT
        END                  ;程序结束
```

从程序可以看出，中断优先级是由查询的顺序决定的。

例 2　如图 4.11 所示，利用 T0 的溢出中断法，在 P1.0 端口产生 500 Hz 的方波对称脉冲。

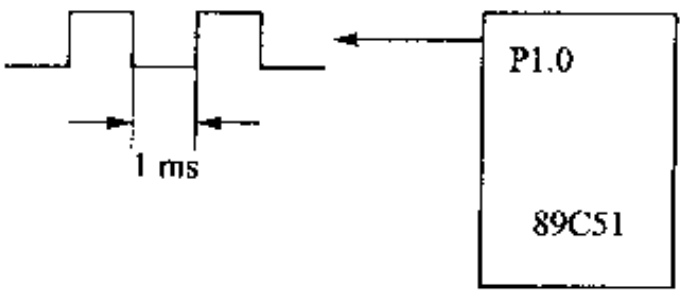


图 4.11　方波发生器接口电路

解：设 T0 为 16 位定时器模式，利用中断法设计程序，溢出周期为 1ms 时，初值为 TC＝65 536－1 000＝64 536＝FC18H(晶振为 12 MHz)。

程序如下：

```
        ORG   0000H          ;主程序执行入口地址
        LJMP  MAIN           ;跳至 MAIN 执行
        ORG   000BH          ;T0 溢出中断服务程序入口
        LJMP  INTT0          ;跳至 T0 溢出中断服务程序
MAIN:   MOV   TMOD,#01H      ;T0 为 16 位定时模式
        MOV   TL0,#18H       ;定时器装初值
        MOV   TH0,#0FCH      ;定时器装初值
        SETB  EA             ;开总中断允许
        SETB  ET0            ;开定时器 T0 中断允许
        SETB  TR0            ;开启定时器 T0
        SJMP  $              ;等待
INTT0:  CPL   P1.0           ;P1.0 取反
        MOV   TL0,#18H       ;重装初值
        MOV   TH0,#0FCH      ;重装初值
        RETI                 ;中断返回
        END                  ;结束
```

思　　考

1. 例 1、2 中程序执行后能否退出？

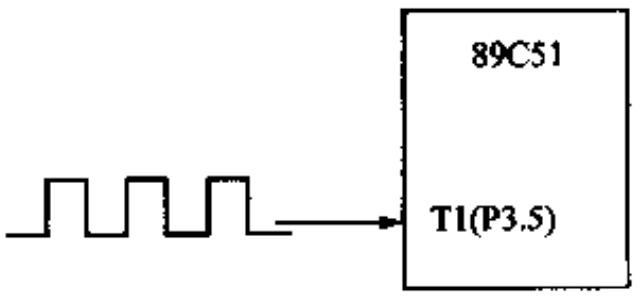


图 4.12　频率测量接口电路

2. 在例 1、2 程序中，CPU 有无空闲的时间？应怎样利用？

3. 为什么通常在中断入口处放一条转移指令？例 1、2 中能不能不用转移指令？

例 3　用定时器/计数器测量脉冲信号的频率

解：如图 4.12 所示，用 T1 作为计数器，T0 作 1 s 定时器，当 1 s 时间到时，将 T1 的计数值移入 70H、71H。T0 定时采用 50 ms，则初值为

65 536--50 000＝15 536＝3CB0H。

程序如下：

```
TESTF:    MOV  TMOD,#51H      ;T0 为 16 位定时器,T1 为 16 位计数器
          MOV R0,#14H         ;T0 溢出 20 次为 1 ms
          MOV TL1,#00H        ;清 T1 计数器
          MOV TH1,#00H        ;清 T1 计数器
          MOV TL0,#0B0H       ;T0 计数器赋初值
          MOV TH0,#3CH        ;T0 计数器赋初值
          SETBP3.5            ;P3.5 端口置输入状态
          SETBEA              ;开中断总允许
          SETBET0             ;允许 T0 溢出中断
          CLR  F0             ;清标志位 F0
LOOP:     JB   P3.5,LOOP      ;等待 P3.5 口低电平脉冲输入
          SETBTR0             ;1 ms 定时启动
          SETBTR1             ;脉冲计数开始
WAIT:     JB   F0,TESTEND     ;标志 F0 为 1 时测试结束
          SJMP WAIT           ;标志 F0 为 0 时等待
TESTEND:  RET                 ;测试结束
          ORG 000BH           ;T0 溢出中断服务程序入口地址
          LJMPINTT0           ;T0 溢出中断服务程序入口
INTT0:    DJNZ R0,INTOUT      ;T0 溢出不到 20 次转 INTOUT
          CLR  TR1            ;T0 溢出 20 次则关计数器 T1
          CLR  TR0            ;关定时器 T0
          CLR  EA             ;关总中断允许
          CLR  ET0            ;关 T0 中断允许
          MOV 70H,TH1         ;将脉冲个数计数值(高位)移入 70H 地址单元
          MOV 71H,TL1         ;将脉冲个数计数值(低位)移入 71H 地址单元
          SETBF0              ;将标志位 F0 置 1
          RETI                ;中断返回
INTOUT:   MOV TL0,#0B0H       ;T0 溢出不到 20 次重装初值
          MOV TH0,#3CH        ;T0 溢出不到 20 次重装初值
          RETI                ;中断返回
```

思　　考

1. 例 3 中 50 ms 的定时时间有误差吗？怎样修正？
2. CPU 有空闲吗？怎样利用？
3. 例 3 中频率计的测量范围是多少？如何扩展？扩展的限制是什么？

4.3　串行口的基本结构与操作方式

89C51 有一个全双工的串行接口，既可以作为串行异步通信(UART)接口，也可以作为同步移位寄存器方式下的串行扩展接口。UART 具有多机通信功能。

4.3.1　串行口的基本组成

串行口由发送控制、接收控制、波特率管理和发送/接收缓冲器 SBUF 组成，其示意图如图 4.13 所示。

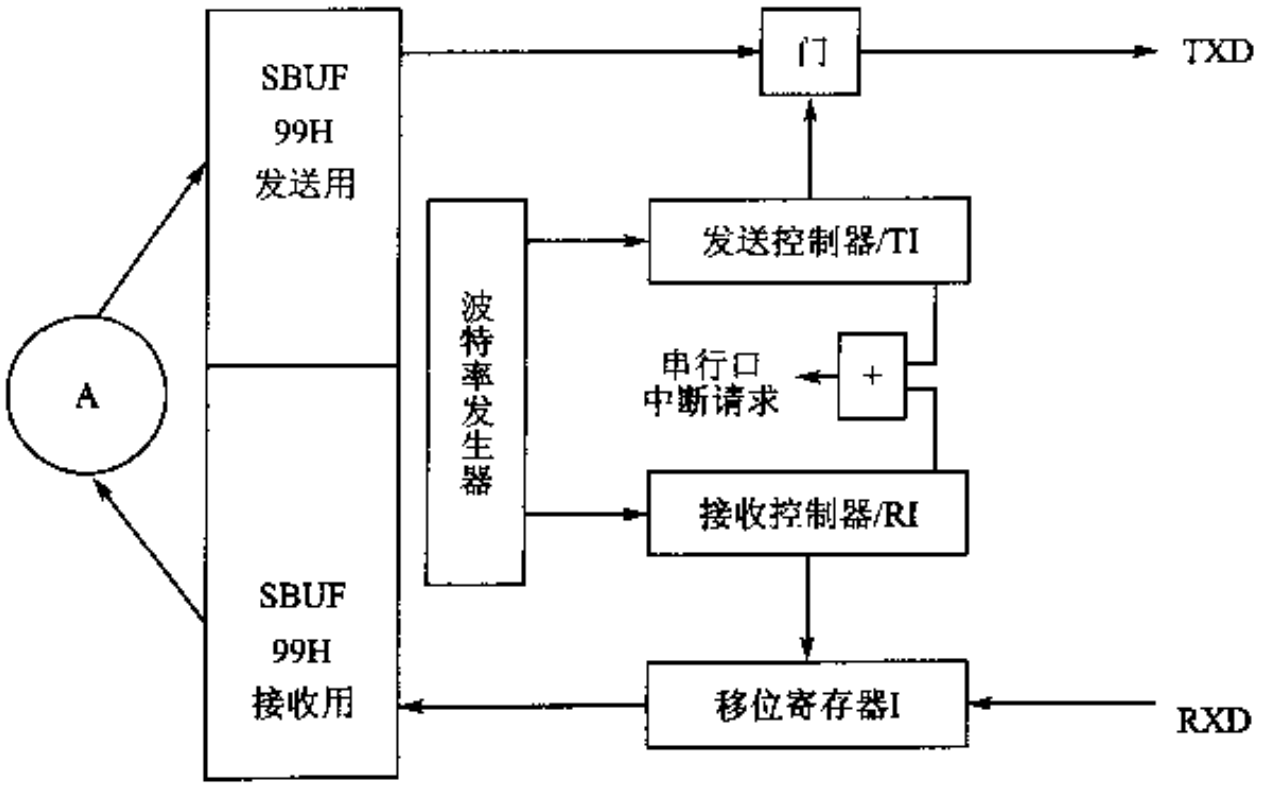


图 4.13　串行口的基本组成示意图

4.3.2　串行口的特殊功能寄存器

1. 发送/接收缓冲器 SBUF

SB UF 是两个各自独立的寄存器，直接地址为 99H，SBUF 只能与 A 进行数据交换。

2. 控制寄存器 SCON

SCON 为可位寻址寄存器，直接地址为 98H，其各位如下：

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

各位意义如下：

SM0、SM1：方式选择位。用来选择串行口的四种工作方式，其功能如表 4.1 所列：

表 4.1　方式选择位及功能

方式位		方　式	功　能	波特率
SM0	SM1			
0	0	0	同步移位寄存器方式	Fosc/12
0	1	1	8(10)位 UART 方式	须设置
1	0	2	9(11)位 UART 方式	Fosc/32 或 Fosc/64
1	1	3	9(11)位 UART 方式	须设置

SM2：多机通信控制位。在方式 2、方式 3 中用于多机通信控制。在方式 2、方式 3 的接收状态中，若 SM2＝1，接收到的第九位(RB8)为零时，舍弃接收到的数据，RI 清零；RB8 为 1 时，将接收到的数据送至 SBUF 中，并将 RI 置 1。当 SM2＝0 时，正常接收。

REN：允许接收位。REN＝1 允许接收，REN＝0 禁止接收。REN 由指令置位或清零。

TB8：第九位发送数据。多机通信(方式 2、方式 3)中，TB8 表明发送的是数据还是地址，TB8＝1 是地址，TB8＝0 是数据。TB8 由指令置位或清零。

RB8：多机通信(方式 2、方式 3)中用来存放接收到的第九位数据，用以表明接收数据的特征。

TI：发送中断标志。当发送数据完毕时，TI＝1，表示帧发送完毕，请求中断，也可供查询。TI 只能由程序清零。

RI：接收中断标志。当接收数据完毕时，RI＝1，表示接收完一帧数据，请求中断，也可供查询。RI 只能由程序清零。

3. 电源控制寄存器 PCON

串行口借用了电源控制寄存器 PCON 的最高位，PCON 为不可位寻址寄存器，直接地址为 87H，基功能位如下：

SMOD	—	—	—	DF1	GF0	PD	IDL

SMOD：波特率加倍位。当波特率由 T1 产生，且 SMOD＝1 时，在串行口的波特率提高一倍。

GF1、GF2：通用标志位。

PD：当 PD＝1 时，进入掉电工作模式。PD 只能由硬件复位。

IDL：当 IDL＝1 时，进入空闲工作模式。

4.3.3　串行口的工作方式

串行口分四种工作方式，由 SCON 中的 SM0、SM1 二位选择决定。

1. 方式 0

(1) 特　点

① 用作串行口扩展，具有固定的波特率，为 Fosf/12。

② 同步发送/接收，由 TXD 提供移位脉冲，RXD 用作数据输入/输出通道。

③ 发送/接收 8 位数据，低位在先。

(2) 发送操作

当执行一条“MOV　SBUF，A”指令时，启动发送操作，由 TXD 输出移位脉冲，由 RXD 串行发送 SBUF 中的数据。发送完 8 位数据后自动置 TI＝1，请求中断。要继续发送时，T1 必须由指令清零。

(3) 接收操作

在 RI＝0 条件下，置 REN＝1，启动一帧数据的接收，由 TXD 输出移位脉冲，由 RXD 接收串行数据到 A 中。接收完一帧自动置位 RI，请求中断。想继续接收时，要用指令清零 RI。

2. 方式 1

(1) 特　点

① 8 位 UART 接口。

② 帧结构为 10 位,包括起始位(为 0),8 位数据位,1 位停止位。

③ 波特率由指令设定,由 T1 的溢出率决定。

(2) 发送操作

当执行一条"MOV　SBUF,A"指令时,启动发送操作,A 中的数据从 TXD 端实现异步发送。发送完一帧数据后自动置 TI=1,请求中断。要继续发送时,TI 必须由指令清零。

(3) 接收操作

当置 REN=1 时,串行口采样 RXD,当采样到 1 至 0 的跳变时,确认串行数据帧的起始位,开始接收一帧数据,直到停止位到来时,把停止位送入 RB8 中。置位 RI 请求中断,CPU 取走数据后用指令清零 RI。

3. 方式 2 和方式 3

方式 2 和方式 3 具有多机通信功能,这两种方式除了波特率不同以外,其余完全相同。

(1) 特　点

① 9 位 UART 接口。

② 帧结构为 11 位,包括起始位(为 0)、8 位数据位、1 位可编程位 TB8/RB8 和停止位(为 1)。

③ 波特率在方式 2 时为固定 FOSC/32 或 FOSC/64,由 SMOD 位决定,当 SMOD=1 时,波特率为 FOSC/32;当 SMOD=0 时,波特率为 FOSC/64。方式 3 的溢出率由 T1 的溢出率决定。

(2) 发送操作

发送数据之前,由指令设置 TB8(如作为奇偶校对位或地址/数据位),将要发送的数据由 A 写入 SBUF 中启动发送操作。在发送中,内部逻辑会把 TB8 装入发送移位寄存器的第 9 位位置,然后发送一帧完整的数据,发送完毕后置位 TI。TI 须由指令清零。

(3) 接收操作

当置位 SEN 位且 RI=0 时,启动接收操作,帧结构上的第 9 位送入 RB8 中,对所接收的数据视 SM2 和 RB8 的状态决定是否会使 RI 置位。

当 SM2=0 时,RB8 不论什么状态 RI 都置 1,串行口都接收数据。

当 SM2=1 时,为多机通信方式,接收到的 RB8 为地址/数据标识位。

当 RB8=1 时,接收的信息为地址帧,此时置位 RI,串行口接收发送来的数据。

当 RB8=0 时,接收的信息为数据帧,若 SM2=1 时,RI 不会置位,此数据丢弃;若 SM2=0,则 SBUF 接收发送来的数据。

4.3.4　应用练习

例 1　串行通信中波特率的计算

方式 0 和方式 2 的波特率是不变的(FOSC/12、FOSC/32、FOS64);方式 1 和方式 3 的波特率由 T1 的溢出率决定:

$$\text{波特率}=(2^{\text{SMOD}}/32)*\text{T1 溢出率}$$

$$=(2^{\text{SMOD}}/32)*(F_{OSC}/12(256-X))$$

定时器 T1 产生的常用波特率如表 4.2 所列。

表 4.2　定时器 T1 产生的常用波特率

波特率	振荡时钟/MHz	SMOD	T1 在方式 2 的初值
62.5 K	12	1	FFH
19.2 K	11.059	1	FDH
9.6 K	11.059	0	FDH
4.8 K	11.059	0	FAH
2.4 K	11.059	0	F4H
1.2 K	11.059	0	E8H
137.5	11.986	0	1DH
110	6	0	72H

例 2　UART 方式 0 状态发送 N 个字节子程序

```
UARTOUT: MOV R0,#MTD        ;发送数据首址入 R0
         MOV R2,#N          ;发送字节个数入 R2
         MOV SCON,#00H      ;设串口为方式 0
SOUT:    MOV A,@R0          ;发送数据入 A
         CLR  TI            ;清发送标志 TI
         MOV SBUF,A         ;启动发送
WAITOUT: JNB  T1,WAITOUT    ;发送等待
         INC  R0            ;指向下一字节
         DJNZ R2,SOUT       ;N 个字节未发完,转 SOUT
         RET                ;N 个字节发完,结束
```

例 3　串行口方式 0 下接收子程序

```
UARTIN:  MOV R0,#MTD        ;接收数据存放首址入 R0
         MOV R2,#N          ;接收字节数入 R2
SIN:     CLR  RI            ;清接收标志
         MOV SCON,#10H      ;置串口为方式 0,REN=1
WAITIN:  JNB  RI,WAITIN     ;接收等待
         MOV A,SBUF         ;接收缓冲器数据入 A
         MOV @R0,A          ;将数据移入内存单元
         INC  R0            ;指向下一存储单元
         DJNZ R2,SIN        ;N 个数据接收未完,转 SIN
         RET                ;N 个数据接收完,结束
```

例 4　利用串口方式 1 实现一个数据块的发送,数据首址为 50H,发送数据长度为 10H,选定波特率为 1 200。

解：设 T1 为方式 2 自动重装初值模式,当时钟为 11.059 MHz 时,初值为 E8H。

```
TXD1:   MOV TMOD,#20H      ;T1 为 8 位自动重装初值模式
        MOV TL1,#0E8H      ;赋初值
        MOV TH1,#0E8H      ;赋初值
        CLR  ET1           ;关 T1 中断
        SETB TR1           ;开定时器 T1
        MOV SCON,#40H      ;串口初始化成方式 1
        MOV PCON,#00H      ;SMOD= 0,不加倍模式
        MOV R0,#50H        ;数据首址入 R0
        MOV R2,#10H        ;数据长度入 R2
TSTART: MOV A,@R0          ;取数据
        MOV SBUF,A         ;数据发送
WAIT:   JBC  TI,CONT       ;等待 TI 变 1 后转 CONT 并对 TI 清 0
        SJMP WAIT
CONT:   INC  R0            ;指向下一字节
        DJNZ R2,TSTART     ;数据未发完,转 TSTART
        RET                ;数据发完,结束
```

思考与练习

1. 在什么方式下需要设置串行口的波特率,如何设定?
2. 在例 4 中若采用末地址来表示时,怎样修改程序?
3. 设计一个对应例 4 的接收程序。
4. SMOD 对波特率有什么影响?
5. 在发送和接收中都有等待过程,在这些程序中是如何判断发送或接收结束的?
6. TB8 在 89C51 的什么地方?

第 5 章　汇编语言程序设计基础

5.1　汇编语言应用程序设计的一般格式

5.1.1　单片机汇编语言程序设计的基本步骤

单片机汇编语言程序设计的基本步骤如下：

(1) 设计任务的分析、确定算法或思路。

(2) 程序的总体设计并画出流程图。主程序流程图的实例如图 5.1 所示，中断服务程序流程图的实例如图 5.2 所示。

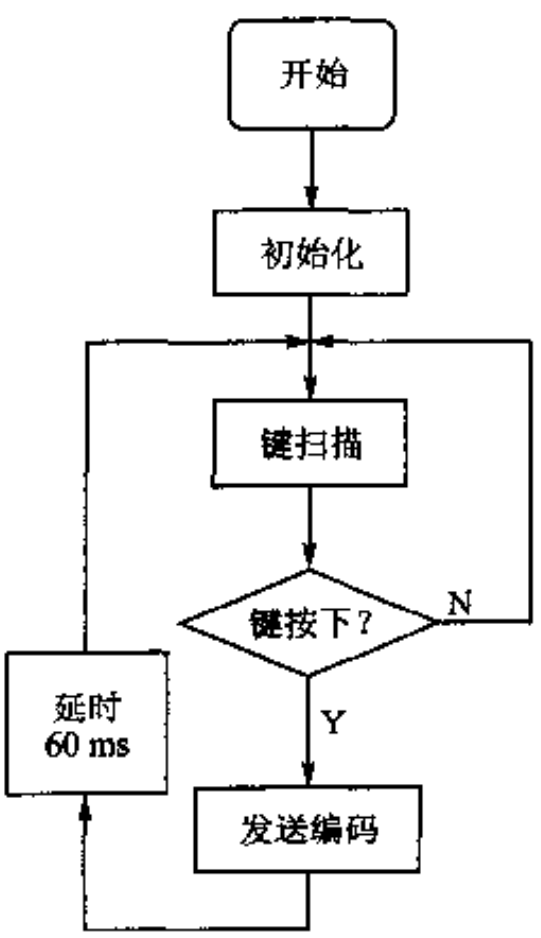


图 5.1　主程序流程图实例

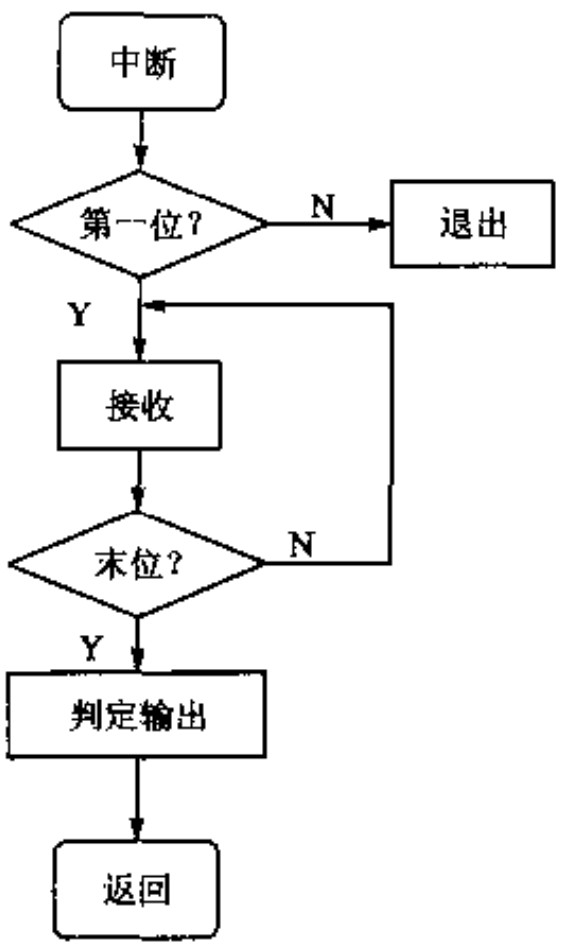
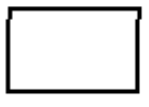
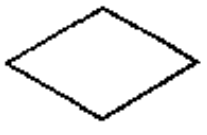


图 5.2　中断服务程序流程图的实例

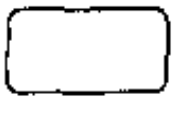
其中各图标的意义如下：



过程框：表示程序要做的事。



判断框：表示条件判断。



开始、结束框：表示流程的开始或终止。

程序流向：箭头所指表示程序的流向。

(3) 编写源程序。可在编译软件下编程(如 Wave),要求简练、层次清楚、字节数少和执行时间短等。

(4) 源程序的汇编与调试(在编译软件中进行)。

(5) 编写程序说明文件。

5.1.2 汇编语言程序的设计方法

(1) 汇编程序的基本结构总是由简单程序、分支程序、循环程序、查表程序、子程序、中断程序等结构化的程序模块有机组成的。

(2) 划分功能模块进行设计。

(3) 自上而下逐渐求精。

5.1.3 常用的伪指令

1. 标号等值伪指令——EQU

格式: (标号) EQU (表达式)。

例如:自行车里程车速计中的定义:

```
VSDA    EQU    P1.5     ;EEP 数据传送口
VSCL    EQU    P1.4     ;EEP 时钟传送口
SLA     EQU    50H      ;EEP 器件寻址字节存放单元
NUMBYT  EQU    51H      ;EEP 传送字节数存放单元
MTD     EQU    30H      ;EEP 发送数据缓冲单元
MRD     EQU    40H      ;EEP 读出数据存放单元
SLAW    EQU    0A0H     ;EEP 寻址字节写
SLAR    EQU    0A1H     ;EEP 寻址字节读
DPHH    EQU    62H      ;DPTR 计数扩展高 8 位
TH1H    EQU    6CH      ;定时器 T1 扩展计数单元
TH1HH   EQU    6DH      ;定时器 T1 扩展计数单元
```

2. 标号等值伪指令——DL

格式: (标号) DL (表达式)

DL 伪定义可以重复定义

3. 数据存储说明伪定义——DB

格式为: (标号) DB (表达式或数据串)

例如:

```
TAB:DB  00H,14H,45H,0FEH,56H
    DB  89H,0DFH
```

4. 数据伪定义——DW

格式: (标号) DW (双字节表达式或数据串)

例如:

```
TAB:DW  0013H,1456H,45DFH,0FE12H,5600H
```

5. 存储区说明伪指令——DS

格式：（标号）　DS　（表达式）

例如：

```
BASE: DS   0100H        ;从标号 BASE 开始空出 256 个单元
```

6. 程序起始地址伪定义——ORG

用来定义程序的起始地址。

例如：

```
ORG   0000H
LJMP  START
```

5.2　简单结构程序

简单结构程序又叫顺序程序，程序从第一条指令开始一直执行到最后一条，无分支，无循环。

例如：双字节加法程序，其程序如下：

```
;
;被加数在 addr1(低位)和 addr2(高位)中，加数在 addr3(低位)和 addr4(高位)中
;运算结果在 addr1 和 addr2 中
;
ADDR1    EQU  30H
ADDR2    EQU  31H
ADDR3    EQU  32H
ADDR4    EQU  33H
;
ADDST:   PUSH ACC
         MOV  R0,#addr1
         MOV  R1,#addr3
         MOV  A,@R0
         ADD  A,@R1
         MOV  @R0,A
         INC  R0
         INC  R1
         MOV  A,@R0
         ADDC A,@R1
         MOV  @R0,A
         POP  ACC
         RET
```

5.3 分支结构程序

1. 单分支结构程序

单分支结构程序只有一个入口，两个出口，根据条件的判断选择出口。例如：

```
START:  ACALL   CLEAR          ;调用初始化子程序
START1: MOV     P3,#0FFH       ;置 P3 口为输入状态
        JNB     P3.0,FUN0      ;P3.0 为 0 转 FUN0 执行
        LJMP    FUN1           ;P3.0 为 1 转 FUN1 执行
```

2. 多分支结构程序

多分支结构程序指一个入口，多个出口，根据条件选择执行一个程序。例如：键功能散转程序，其程序如下：

```
        MOV     DPTR,#KEYFUNTAB  ;装入键功能标号首址
        JMP     @A+DPTR          ;散转
KEYFUNTAB:LJMP  KEYFUN00         ;跳到 KEYFUN00
        LJMP    KEYFUN01         ;跳到 KEYFUN01
        LJMP    KEYFUN02         ;跳到 KEYFUN02
        ⋮
        RET
```

5.4 循环结构程序

循环结构程序用以控制一个程序多次重复执行，当条件满足时退出循环。循环结构程序由初始化、循环处理、判断、结束处理等组成。例如：采用 12MHZ 晶振的 513μs 延时程序，其程序如下：

```
;
DL513:  MOV     R2,#0FFH
DELAY1: DJNZ    R2,DELAY1
        RET
```

5.5 子程序结构程序

一些经常要用的程序一般设计成子程序，以便供其他程序经常调用。子程序必须具有程序标号，结束必须用 RET 指令，调用时用 LCALL、ACALL 等指令。例如：延时程序和显示程序等。

5.6 查表程序

查表程序用 MOVC 指令，用于访问(查)程序存储器中的固定数表，如用于七段 LED 数码

管显示的程序中就用到了查表指令，其程序如下：

```
;
DISPLAY: MOV R1,#70H          ;显示数据首址
         MOV R5,#0FEH         ;扫描端口初值
  PLAY: MOV A,R5              ;将 R5 中数据移入 A 中
         MOV P1,#0FFH         ;清原数据
         MOV P3,A             ;扫描端口值
         MOV A,@R1            ;取显示数据
         MOV DPTR,#TAB        ;段码表表址放入数据指针
         MOVC A,@A+DPTR       ;查段码
         MOV P1,A             ;段码数据放到 P1 口
         LCALL DL1MS          ;数据显示 1 ms
         INC R1               ;存放显示数据地址加 1
         MOV A,R5             ;扫描端口值放入 A
         JNB ACC.5,ENDOUT     ;A 中值为 11011111(B)时结束
         RL A                 ;A 中数据循环左移一位
         MOV R5,A             ;A 中数据放回 R5 中
         AJMP PLAY            ;跳至 PLAY 循环
ENDOUT: MOV P3,#0FFH          ;退出时 P3 口复回
         MOV P1,#0FFH         ;退出时 P1 口复回
         RET                  ;子程序结束
   TAB: DB 0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H,0FFH
;     共阳段码表     "0" "1" "2"  "3" "4" "5" "6" "7" "8" "9" "熄灭符"
```

5.7　查键程序

具有按键控制功能的单片机应用系统都有查键功能程序，有简单的顺序查键及复杂的行列式查键。

例 1　顺序查键程序

```
START:MOV    P3,#0FFH     ;置 P3 口为输入口
      JNB    P3.0,FUN0    ;P3.0 口为 0 转 FUN0
      JNB    P3.1,FUN1    ;P3.1 口为 0 转 FUN1
      JNB    P3.2,FUN2    ;P3.2 口为 0 转 FUN2
      JNB    P3.3,FUN3    ;P3.3 口为 0 转 FUN3
      AJMP   START        ;转 START 循环
```

例 2　32 键行列式查键程序(4×8)

32 键行列式查键电原理图如图 5.3 所示。

以下是 32 键行列式查键程序(4×8)：

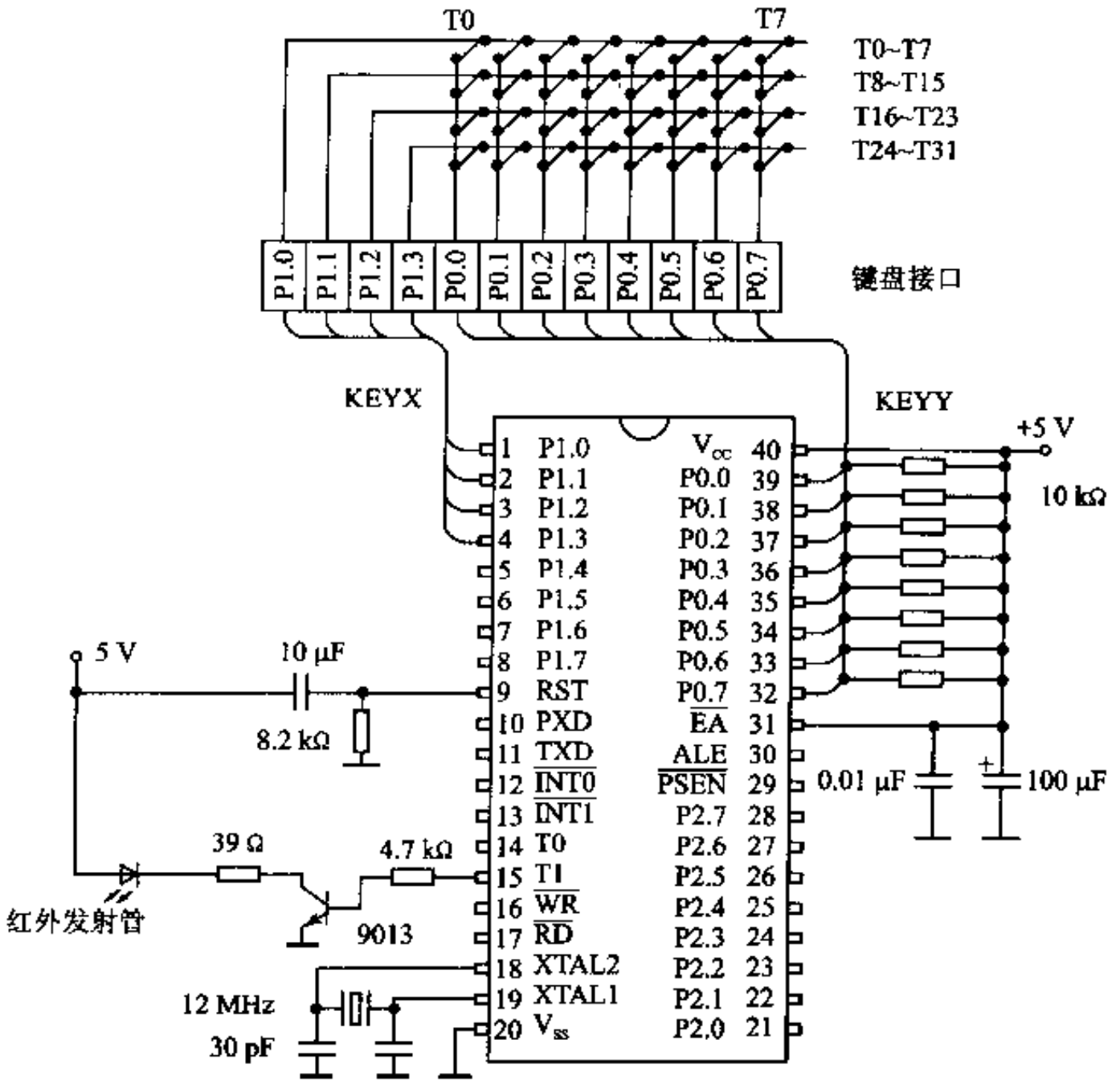


图 5.3　32 键行列式查键电原理图

```
;*****************************************
;*                                       *
;*      键盘工作子程序(4 × 8 阵列)       *
;*        出口为各键工作程序入口         *
;*                                       *
;*****************************************
KEYWORK:    MOV   KEYY,#0FFH        ;置列线输入
            CLR   KEYX0             ;行线(P1 口)全置 0
            CLR   KEYX1
            CLR   KEYX2
            CLR   KEYX3
            MOV   A,KEYY            ;读入 P0 口值
            MOV   B,A               ;KEYY 口值暂存 B 中
            CJNE  A,#0FFH,KEYHIT    ;不等于#0FFH,转 KEYHIT(有键按下)
KEYOUT:     RET                     ;没有键按下返回
;
KEYHIT:     LCALL DL10MS            ;延时去抖动
            MOV   A,KEYY            ;再读入 P0 口值至 A
            CJNE  A,B,KEYOUT        ;A 不等于 B(是干扰),子程序返回
            SETB  KEYX1             ;有键按下,找键号,开始查 0 行
```

```
        SETB    KEYX2
        SETB    KEYX3
        MOV     A,KEYY                  ;读入 P0 口值
        CJNE    A,#0FFH,KEYVAL0         ;P0 不等于#0FFH,按下键在第 0 行
        SETB    KEYX0                   ;不在第 0 行,开始查 1 行
        CLR     KEYX1
        MOV     A,KEYY                  ;读入 P0 口值
        CJNE    A,#0FFH,KEYVAL1         ;P0 口不等于#0FFH,按下键在第 1 行
        SETB    KEYX1                   ;不在第 1 行,开始查 2 行
        CLR     KEYX2
        MOV     A,KEYY                  ;读入 P0 口值
        CJNE    A,#0FFH,KEYVAL2         ;P0 口不等于#0FFH,按下键在第 2 行
        SETB    KEYX2                   ;不在第 2 行,开始查 3 行
        CLR     KEYX3
        MOV     A,KEYY                  ;读入 P0 口值
        CJNE    A,#0FFH,KEYVAL3         ;P0 口不等于#0FFH,按下键在第 3 行
        LJMP    KEYOUT                  ;不在第 3 行,子程序返回
;
KEYVAL0: MOV    R2,#00H                 ;按下键在第 0 行,R2 赋行号初值 0
        LJMP    KEYVAL4                 ;跳到 KEYVAL4
;
KEYVAL1: MOV    R2,#08H                 ;按下键在第 1 行,R2 赋行号初值 8
        LJMP    KEYVAL4                 ;跳到 KEYVAL4
;
KEYVAL2: MOV    R2,#10H                 ;按下键在第 2 行,R2 赋行号初值 16
        LJMP    KEYVAL4                 ;跳到 KEYVAL4
;
KEYVAL3: MOV    R2,#18H                 ;按下键在第 3 行,R2 赋行号初值 24
        LJMP    KEYVAL4                 ;跳到 KEYVAL4
;
KEYVAL4: MOV    DPTR,#KEYVALTAB         ;键值翻译成连续数字
        MOV     B,A                     ;P0 口值暂存 B 内
        CLR     A                       ;清 A
        MOV     R0,A                    ;清 R0
KEYVAL5: MOV    A,R0                    ;查列号开始,R0 数据放入 A
        SUBB    A,#08H                  ;A 中数减 8
        JNC     KEYOUT                  ;借位 C 为 0,查表出错,返回
        MOV     A,R0                    ,查表次数小于 8,继续查
        MOVC    A,@A+DPTR               ;查列号表
        INC     R0                      ;R0 加 1
        CJNE    A,B,KEYVAL5             ;查得值和 P0 口值不等,转 KEYVAL5 再查
        DEC     R0                      ;查得值和 P0 口值相等,R0 减 1
        MOV     A,R0                    ;放入 A(R0 中数值即为列号值)
```

```
                ADD     A,R2                ;与行号初值相加成为键号值(0～31)
                MOV     B,A                 ;键号乘 3 处理用于 JMP 散转指令
                RL      A                   ;键号乘 3 处理用于 JMP 散转指令
                ADD     A,B                 ;键号乘 3 处理用于 JMP 散转指令
                MOV     DPTR,#KEYFUNTAB     ;取散转功能程序(表)首址
                JMP     @A+DPTR             ;散转至对应功能程序标号
KEYFUNTAB:LJMP  KEYFUN00                    ;跳到键号 0 对应功能程序标号
                LJMP    KEYFUN01            ;跳到键号 1 对应功能程序标号
                LJMP    KEYFUN02            ;跳到键号 2 对应功能程序标号
                LJMP    KEYFUN03
                LJMP    KEYFUN04
                LJMP    KEYFUN05
                LJMP    KEYFUN06
                LJMP    KEYFUN07
                LJMP    KEYFUN08
                LJMP    KEYFUN09
                LJMP    KEYFUN10
                LJMP    KEYFUN11
                LJMP    KEYFUN12
                LJMP    KEYFUN13
                LJMP    KEYFUN14
                LJMP    KEYFUN15
                LJMP    KEYFUN16
                LJMP    KEYFUN17
                LJMP    KEYFUN18
                LJMP    KEYFUN19
                LJMP    KEYFUN20
                LJMP    KEYFUN21
                LJMP    KEYFUN22
                LJMP    KEYFUN23
                LJMP    KEYFUN24
                LJMP    KEYFUN25
                LJMP    KEYFUN26
                LJMP    KEYFUN27
                LJMP    KEYFUN28
                LJMP    KEYFUN29
                LJMP    KEYFUN30
                LJMP    KEYFUN31
                RET
;P0 口对应列号的 ROM 数值表
KEYVALTAB:DB  0FEH,0FDH,0FBH,0F7H,0EFH,0DFH,0BFH,7FH
;                      0    1     2    3     4     5     6    7
                    RET
```

```
;各按键功能程序
KEYFUN00:  RET                    ;键号 0 功能程序
KEYFUN01:  RET                    ;键号 1 功能程序
KEYFUN02:  RET                    ;键号 2 功能程序
KEYFUN03:  RET
KEYFUN04:  RET
KEYFUN05:  RET
KEYFUN06:  RET
KEYFUN07:  RET
KEYFUN08:  RET
KEYFUN09:  RET
KEYFUN10:  RET
KEYFUN11:  RET
KEYFUN12:  RET
KEYFUN13:  RET
KEYFUN14:  RET
KEYFUN15:  RET
KEYFUN16:  RET
KEYFUN17:  RET
KEYFUN18:  RET
KEYFUN19:  RET
KEYFUN20:  RET
KEYFUN21:  RET
KEYFUN22:  RET
KEYFUN23:  RET
KEYFUN24:  RET
KEYFUN25:  RET
KEYFUN26:  RET
KEYFUN27:  RET
KEYFUN28:  RET
KEYFUN29:  RET
KEYFUN30:  RET
KEYFUN31:  RET
RET
;
```

5.8　显示程序

LED 七段数码管显示电路如图 5.4 所示。

LED 七段数码管显示程序采用动态扫描法，先将要显示的数据通过查表得到段码数据，然后放入输出口，再将相应的数码管点亮，以此循环。以下是一个四位 LED 共阳数码管显示程序，用 P1 口及 P3 口作显示扫描口，数据在 P1 口输出，列扫描在 P3.0～P3.3 口。

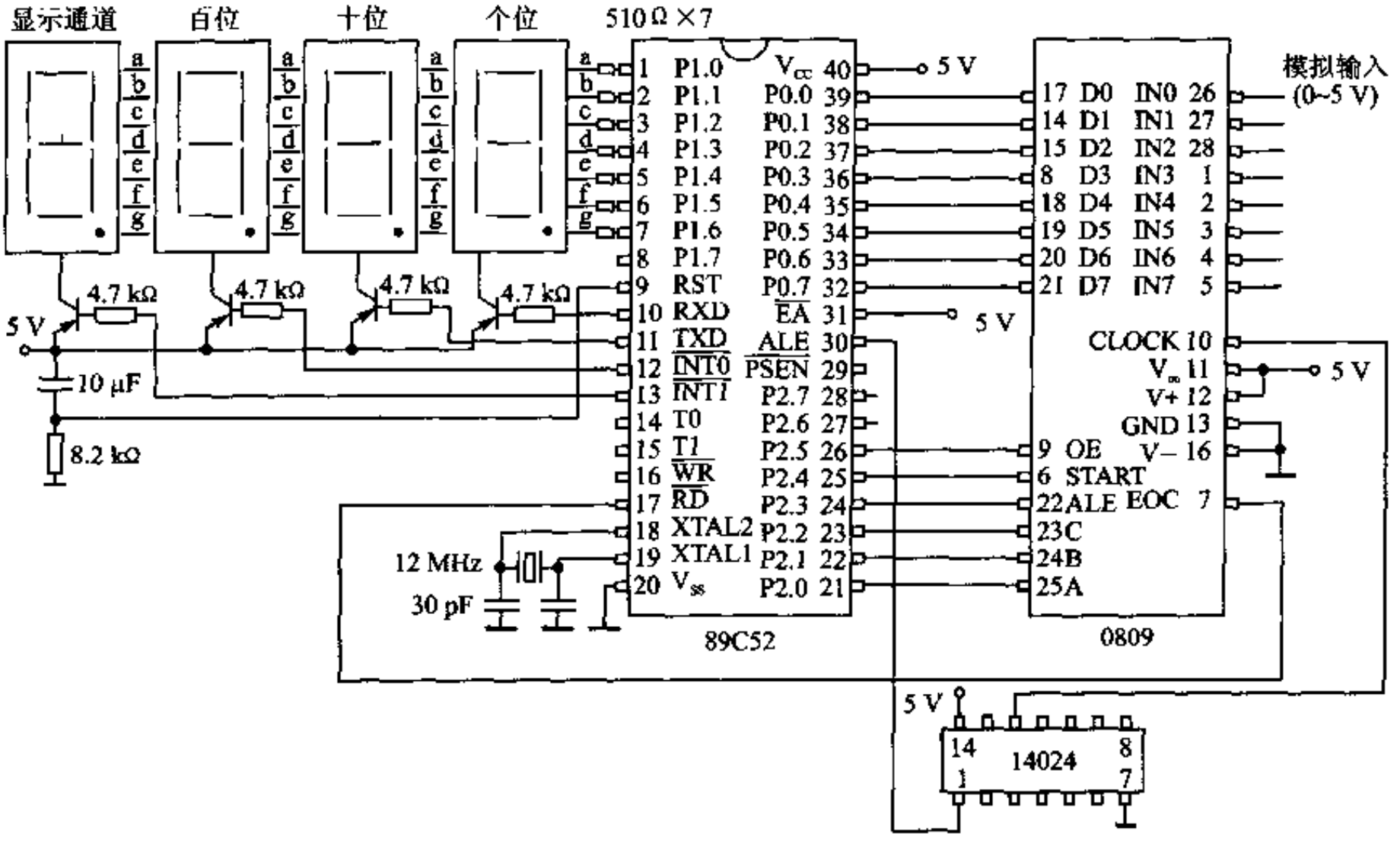


图 5.4　LED 七段数码管显示电原理图

```
;四位共阳数口码管显示子程序,显示内容在 78H~7BH
DISP:    MOV   R1,#78H          ;取显示数据首址
         MOV   R5,#0FEH         ;扫描用初值
PLAY:    MOV   P1,#0FFH         ;显示关闭
         MOV   A,R5             ;扫描控制值入 A
         ANL   P3,A             ;放入 P3 口
         MOV   A,@R1            ;取显示数据
         MOV   DPTR,#TAB        ;取表首地址
         MOVC  A,@A+DPTR        ;查显示用段码数据
         MOV   P1,A             ;段码数据放入 P1 口
         LCALL DL1MS            ;显示 1 ms
         INC   R1               ;显示数据地址加 1
         MOV   A,P3             ;读入 P3 端口值至 A
         JNB   ACC.3,ENDOUT     ;P3.3 为 0,结束
         RL    A                ;P3.3 不为 0,A 中数值左移一位
         MOV   R5,A             ;放回 R5 内暂存
         MOV   P3,#0FFH         ;关扫描显示
         AJMP  PLAY             ;跳回 PLAY 循环
ENDOUT:  MOV   P3,#0FFH         ;P3 口置 1,关显示
         MOV   P1,#0FFH         ;P1 口置 1,关显示
         RET                    ;子程序返回
TAB:     DB 0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H,0FFH  ;共阳段码表
;        显示数      "0"  "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "熄灭符"
```

思考与练习

1. 简述单片机程序设计的基本步骤。
2. 阅读“双字节加法程序”并给程序加上注释。
3. 试写一个正时时间为 515 μs 的延时用子程序(设晶振为 12 MHz)。
4. 在“32 键行列式查键程序”例子中为什么键号值在执行散转指令前要进行乘 3 处理。

第 2 部分

51 系列单片机

设计应用程序实例

第 6 章　实例 1　闪烁 LED 小灯的设计

本设计的闪烁小灯控制器，可使小灯轮流点亮、逐点点亮、间隔闪亮。如要控制交流彩灯，可在 P1 端口加接继电器或可控硅接口电路。本设计可应用在广告彩灯控制器、舞台灯光控制器等领域。

1. 系统硬件电路的设计

图 6.1 为闪烁小灯控制器的电路原理图，其中：单片机采用 AT89C2051，P1 口作 LED 发光管输出控制用，P3.0～P3.2 口为闪烁方式控制开关 K1、K2、K3 按键接口，P3.3 口的按键作备用，限流电阻为 310 Ω，发光管工作电流约为 10 mA，采用 12 MHz 晶振。

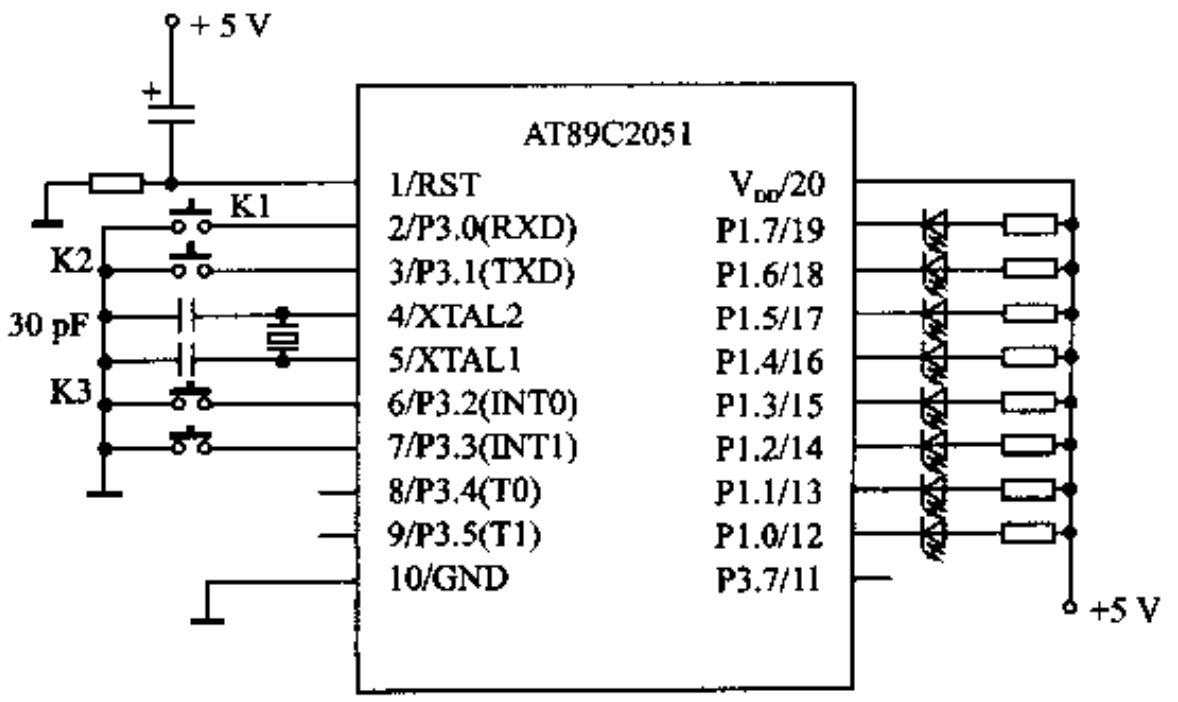


图 6.1　闪烁小灯电路

2. 系统主要程序的设计

(1) 主程序

通过扫描 P3.0～P3.2 口，判断是否有按键按下，然后在 20H 内存单元的低 3 位的对应位置 1 标志，确定应执行的闪烁功能。当 20H.0 为 1 时，发光管轮流点亮；当 20H.1 为 1 时，发光管逐点点亮；当 20H.2 为 1 时，发光管间隔闪亮。在主程序对 20H 的低 3 位进行位值判定后，转入相应的闪烁控制程序。上电初始化时，对 20H 的最低位置 1，系统进入轮流点亮方式。

(2) 键扫描子程序

因按键较少，采用直接端口扫描键开关，用软件延时消抖确认后，对 20H 内存单元相应的位置 1，并把其余位清零。

(3) 闪烁控制程序

闪烁控制程序用来控制 P1 口的发光管发光变化方式，其中：

执行功能程序 0(FUN0)时的 P1 口输出值变化为 11111110→延时→11111101→延时→11111011→延时→11110111→延时→11101111→延时→11011111→延时→10111111→延时→01111111→延时→结束转主程序。

执行功能程序 1(FUN1)时的 P1 口输出变化为 11111110→延时→11111100→延时→

11111000→延时→11110000→延时→11100000→延时→11000000→延时→10000000→延时→00000000→延时→结束转主程序。

执行功能程序 2(FUN2)时的 P1 口输出变化为 10101010→延时→01010101→延时→结束转主程序。

(4) 延时子程序

延时子程序有 10 ms 和 0.5 s 两个,用作键扫描消抖及发光管闪烁延时。发光管闪烁的快慢可由 R4 寄存器内的初值进行改变。

3. 主程序流程图

主程序流程图如图 6.2 所示。

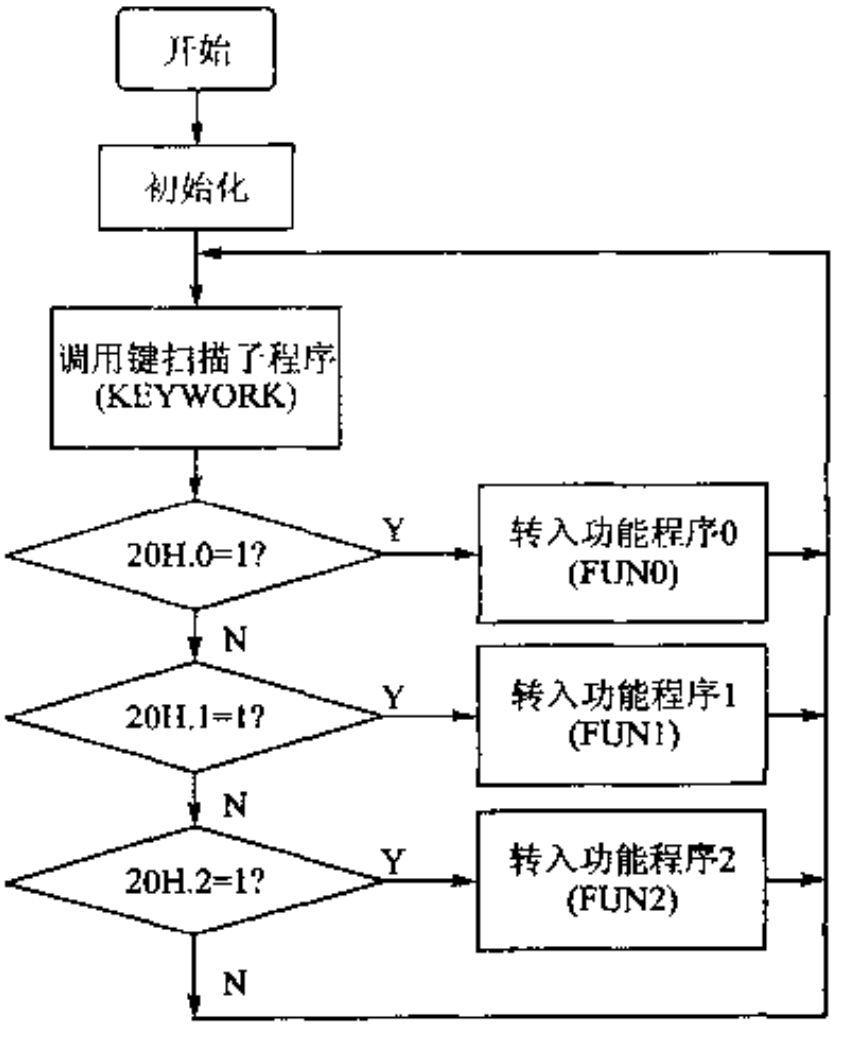


图 6.2　主程序流程图

本控制器在使用中,如要改变闪烁的方式,可按下相应的功能按键。当一个完整的闪烁循环结束后,即可转入新的闪烁方式。由于键扫描是在闪烁循环结束时进行,因此,功能开关按下的时间应较长才能被读入。改进的方法是把 DL05S 延时子程序用键扫描子程序来替代,这样,只要按下按键即可被键扫描程序读入。

以下是 LED 小灯闪烁控制器的完整源程序:

```
;************;
;   小灯控制程序    ;
;************;
;
;************;
;   中断入口程序    ;
;************;
;
```

```
ORG     0000H                   ;程序执行开始地址
LJMP    START                   ;跳至 START 执行
ORG     0003H                   ;外中断 0 中断入口地址
RETI                            ;中断返回(不开中断)
ORG     000BH                   ;定时器 T0 中断入口地址
RETI                            ;中断返回(不开中断)
ORG     0013H                   ;外中断 1 中断入口地址
RETI                            ;中断返回(不开中断)
ORG     001BH                   ;定时器 T1 中断入口地址
RETI                            ;中断返回(不开中断)
ORG     0023H                   ;串行口中断入口地址
RETI                            ;中断返回(不开中断)
;
;************;
;    初始化程序    ;
;************;
;
CLEAR:  MOV     20H,#00H        ;20H 单元内存清 0(闪烁标志清 0)
        SETB    00H             ;20H.0 位置 1(上电时自动执行闪烁功能 1)
        RET                     ;子程序返回
;
;************;
;     主 程 序    ;
;************;
;
START:  ACALL   CLEAR           ;调用初始化子程序
MAIN:   LCALL   KEYWORK         ;调用键扫描子程序
        JB      00H,FUN0        ;20H.0 位为 1 时执行 FUN0
        JB      01H,FUN1        ;20H.1 位为 1 时执行 FUN1
        JB      02H,FUN2        ;20H.2 位为 1 时执行 FUN2
        JB      03H,MAIN        ;备用
        AJMP    MAIN            ;返回主程序 MAIN
;
;************;
;     功能程序    ;
;************;
;第 1 种闪烁功能程序
FUN0: MOV   A,#0FEH ;累加器赋初值
FUN00:  MOV     P1,A            ;累加器值送至 P1 口
        LCALL   DL05S           ;延时
        JNB     ACC.7,MAIN      ;累加器最高位为 0 时转 MAIN
        RL      A               ;累加器 A 中数据循环左移 1 位
        AJMP    FUN00           ;转 FUN00 循环
```

```
;
;第 2 种闪烁功能程序
FUN1:     MOV     A,#0FEH          ;累加器赋初值
FUN11:    MOV     P1,A             ;累加器值送至 P1 口
          LCALL   DL05S            ;延时
          JZ      MAIN             ;A 为 0 转 MAIN
          RL      A                ;累加器 A 中数据循环左移 1 位
          ANL     A,P1             ;A 同 P1 口值相与
          AJMP    FUN11            ;转 FUN11 循环
;
;第 3 种闪烁功能程序
FUN2:     MOV     A,#0AAH          ;累加器赋初值
          MOV     P1,A             ;累加器值送至 P1 口
          LCALL   DL05S            ;延时
          CPL     A                ;A 中各位取反
          MOV     P1,A             ;累加器值送至 P1 口
          LCALL   DL05S            ;延时
          AJMP    MAIN             ;转 MAIN
;************;
;     扫键程序     ;
;************;
;
KEYWORK:  MOV     P3,#0FFH         ;置 P3 口为输入状态
          JNB     P3.0,KEY0        ;读 P3.0 口,若为 0 转 KEY0
          JNB     P3.1,KEY1        ;读 P3.1 口,若为 0 转 KEY1
          JNB     P3.2,KEY2        ;读 P3.2 口,若为 0 转 KEY2
          JNB     P3.3,KEY3        ;读 P3.3 口,若为 0 转 KEY3
          RET                      ;子程序返回
;
;闪烁功能 0 键处理程序
KEY0:     LCALL   DL10MS           ;延时 10 ms 消抖
          JB      P3.0,OUT0        ;P3.0 为 1,子程序返回(干扰)
          SETB    00H              ;20H.0 位置 1(执行闪烁功能 1 标志)
          CLR     01H              ;20H.1 位清 0
          CLR     02H              ;20H.2 位清 0
          CLR     03H              ;20H.3 位清 0
OUT0:     RET                      ;子程序返回
;
;闪烁功能 1 键处理程序
KEY1:     LCALL   DL10MS
          JB      P3.1,OUT1
          SETB    01H              ;20H.1 位置 1(执行闪烁功能 2 标志)
          CLR     00H
```

```
                CLR     02H
                CLR     03H
OUT1:           RET
;
;闪烁功能 2 键处理程序
KEY2:           LCALL   DL10MS
                JB      P3.2,OUT2
                SETB    02H             ;20H.2 位置 1(执行闪烁功能 3 标志)
                CLR     01H
                CLR     00H
                CLR     03H
OUT2:           RET
;
;闪烁功能(备用)键处理程序
KEY3:           LCALL   DL10MS
                JB      P3.3,OUT3
                SETB    03H             ;20H.3 位置 1(执行备用闪烁功能标志)
                CLR     01H
                CLR     02H
                CLR     00H
OUT3:           RET
;
;************;
;      延时程序      ;
;************;
;延时子程序,执行一次时间为 513 μs
DL512:          MOV     R2,#0FFH
LOOP1:          DJNZ    R2,LOOP1
                RET
;
;10 ms 延时子程序(调用 20 次 0.5 ms 延时子程序)
DL10MS:         MOV     R3,#14H
LOOP2:          LCALL   DL512
                DJNZ    R3,LOOP2
                RET
;
;延时子程序,改变 R4 寄存器初值可改变闪烁的快慢(时间为 25 ms×15)
DL05S:          MOV     R4,#0FH
LOOP3:          LCALL   DL25MS
                DJNZ    R4,LOOP3
                RET
;
;25ms 延时子程序,调用扫键子程序延时,可快速读出功能按键值
```

```
DL25MS:    MOV     R5,#0FFH
LOOP4:     LCALL   KEYWORK
           DJNZ    R5,LOOP4
           RET
END                                    ;程序结束
```

第 7 章　实例 2　数码管时钟电路的设计

LED 数码管时钟电路采用 24 h 计时方式，时、分、秒用六位数码管显示。该电路采用 AT89C2051 单片机，使用 3 V 电池供电，只使用一个按键开关即可进入调时、省电(不显示 LED 数码管)和正常显示三种状态。

1. 时钟电路硬件的设计

数码管时钟电路如图 7.1 所示，其采用 AT89C2051 单片机最小化应用设计，LED 显示采用动态扫描方式实现，P1 口输出段码数据，P3.0～P3.5 口作扫描输出，P3.7 接按钮开关。为了提供 LED 数码管的驱动电流，用三极管 9012 作电源驱动输出。为了提高秒计时的精确性，采用 12 MHz 晶振。

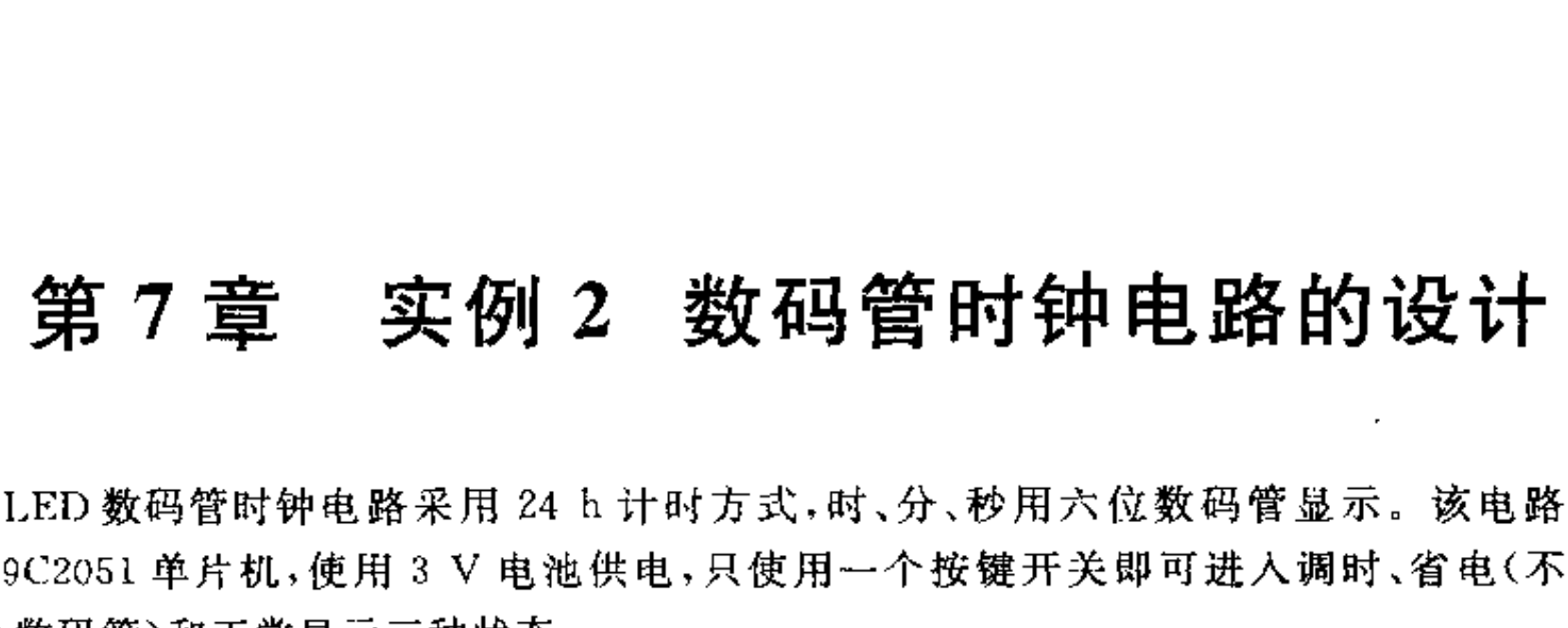


图 7.1　采用 89C2051 的六位时钟电路

2. 系统主要程序的设计

(1) 主程序

本设计中的计时采用定时器 T0 中断完成，其余状态循环调用显示子程序，当 P3.7 端口开关按下时，转入调时功能程序。系统主程序流程图如图 7.2 所示。

(2) 显示子程序

数码管显示的数据存放在内存单元 70H～75H 中，其中 70H～71H 存放秒数据，72H～73H 存放分数据，74H～75H 存放时数据，每一地址单元内均为十进制 BCD 码。由于采用软件动态扫描实现数据显示功能，显示用十进制 BCD 码数据的对应段码存放在 ROM 表中。显示时，先取出 70H～75H 某一地址中的数据，然后查得对应的显示用段码从 P1 口输出。P3 口将对应的数码管选中，就能显示该地址单元的数据值。

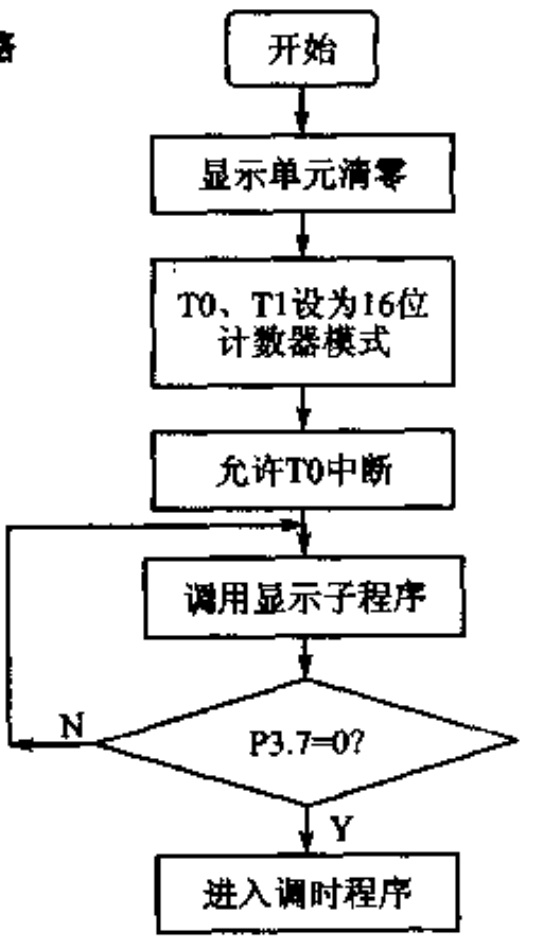


图 7.2　主程序流程图

(3) 定时器 T0 中断服务程序

定时器 T0 用于时间计时。定时溢出中断周期设为 50 ms,中断累计 20 次(即 1s)时对秒计数单元进行加 1 操作。时间计数单元地址分别在 70H～71H(s)、76H～77H(min)、78H～79H(h),7AH 单元内存放"熄灭符"数据(#0AH)。在计数单元中采用十进制 BCD 码计数,满 60 进位。T0 中断服务程序流程图如图 7.3 所示。

(4) T1 中断服务程序

T1 中断服务程序用于指示调整单元数字的亮闪。在时间调整状态下,每过 0.3 s,将对应单元的显示数据换成"熄灭符"数据(#0AH)。这样,在调整时间时,对应调整单元的显示数据就会间隔闪亮。

(5) 调时功能程序

调时功能程序的设计方法是:按下按键,若按下时间小于 1 s,进入省电状态(数码管不亮,时钟不停),否则进入调分状态,等待操作,此时计时器停止走动。当再次按下按钮时,若按下时间小于 0.5 s,则时间加 1 min;若按下时间大于 0.5 s,则进入小时调整状态。在小时调整状态下,当按键按下的时间大于 0.5 s 时退出调整状态,时钟继续走动。

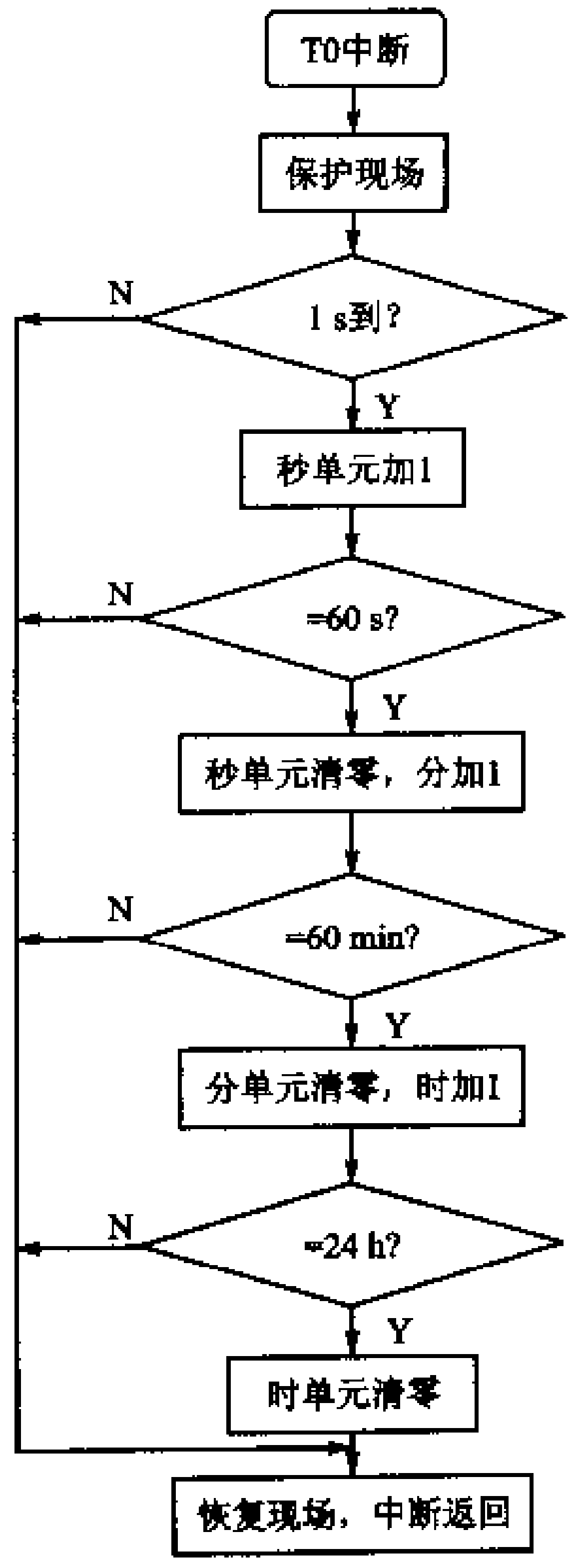


图 7.3　T0 中断服务程序

以下是 LED 数码管时钟电路的完整源程序:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;              AT89C2051 时钟程序          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;   定时器 T0、T1 溢出周期为 50 ms,T0 为秒计数用, T1 为调整时闪烁用,
;   P3.7 为调整按钮,P1 口为字符输出口,采用共阳显示管
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;           中断入口程序                  ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
            ORG 0000H                      ;程序执行开始地址
            LJMP      START                ;跳到标号 START 执行
            ORG 0003H                      ;外中断 0 中断程序入口
            RETI                           ;外中断 0 中断返回
            ORG 000BH                      ;定时器 T0 中断程序入口
            LJMP      INTT0                ;跳至 INTT0 执行
            ORG 0013H                      ;外中断 1 中断程序入口
```

```
            RETI                                ;外中断 1 中断返回
            ORG 001BH                           ;定时器 T1 中断程序入口
            LJMP      INTT1                     ;跳至 INTT1 执行
            ORG 0023H                           ;串行中断程序入口地址
            RETI                                ;串行中断程序返回
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;              主 程 序                 ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
      START:    MOV     R0,#70H             ;清 70H~7AH 共 11 个内存单元
                MOV     R7,#0BH             ;
  CLEARDISP:    MOV     @R0,#00H            ;
                INC     R0                  ;
                DJNZ    R7,CLEARDISP        ;
                MOV     20H,#00H            ;清 20H(标志用)
                MOV     7AH,#0AH            ;放入"熄灭符"数据
                MOV     TMOD,#11H           ;设 T0、T1 为 16 位定时器
                MOV     TL0,#0B0H           ;50 ms 定时初值(T0 计时用)
                MOV     TH0,#3CH            ;50 ms 定时初值
                MOV     TL1,#0B0H           ;50 ms 定时初值(T1 闪烁定时用)
                MOV     TH1,#3CH            ;50 ms 定时初值
                SETB    EA                  ;总中断开放
                SETB    ET0                 ;允许 T0 中断
                SETB    TR0                 ;开启 T0 定时器
                MOV     R4,#14H             ;1 s 定时用初值(50 ms * 20)
     START1:    LCALL   DISPLAY             ;调用显示子程序
                JNB     P3.7,SETMM1         ;P3.7 口为 0 时转时间调整程序
                SJMP    START1              ;P3.7 口为 1 时跳回 START1
    SETMM1:     LJMP    SETMM               ;转到时间调整程序 SETMM
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;              1 s计时程序              ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;T0 中断服务程序
      INTT0:    PUSH    ACC                 ;累加器入栈保护
                PUSH    PSW                 ;状态字入栈保护
                CLR     ET0                 ;关 T0 中断允许
                CLR     TR0                 ;关闭定时器 T0
                MOV     A,#0B7H             ;中断响应时间同步修正
                ADD     A,TL0               ;低 8 位初值修正
                MOV     TL0,A               ;重装初值(低 8 位修正值)
                MOV     A,#3CH              ;高 8 位初值修正
```

```
        ADDC    A,TH0           ;
        MOV     TH0,A           ;重装初值(高 8 位修正值)
        SETB    TR0             ;开启定时器 T0
        DJNZ    R4, OUTT0       ;20 次中断未到中断退出
ADDSS:  MOV     R4,#14H         ;20 次中断到(1 s)重赋初值
        MOV     R0,#71H         ;指向秒计时单元(71H~72H)
        ACALL   ADD1            ;调用加 1 程序(加 1 s 操作)
        MOV     A,R3            ;秒数据放入 A(R3 为 2 位十进制数组合)
        CLR     C               ;清进位标志
        CJNE    A,#60H,ADDMM    ;
ADDMM:  JC      OUTT0           ;小于 60 s 时中断退出
        ACALL   CLR0            ;大于或等于 60 s 时对秒计时单元清 0
        MOV     R0,#77H         ;指向分计时单元(76H~77H)
        ACALL   ADD1            ;分计时单元加 1 min
        MOV     A,R3            ;分数据放入 A
        CLR     C               ;清进位标志
        CJNE    A,#60H,ADDHH    ;
ADDHH:  JC      OUTT0           ;小于 60 min 时中断退出
        ACALL   CLR0            ;大于或等于 60 min 时分计时单元清 0
        MOV     R0,#79H         ;指向小时计时单元(78H~79H)
        ACALL   ADD1            ;小时计时单元加 1 h
        MOV     A,R3            ;时数据放入 A
        CLR     C               ;清进位标志
        CJNE    A,#24H,HOUR     ;
HOUR:   JC      OUTT0           ;小于 24 h 中断退出
        ACALL   CLR0            ;大于或等于 24 h 小时计时单元清 0
OUTT0:  MOV     72H,76H         ;中断退出时将分、时计时单元数据移
        MOV     73H,77H         ;入对应显示单元
        MOV     74H,78H         ;
        MOV     75H,79H         ;
        POP     PSW             ;恢复状态字(出栈)
        POP     ACC             ;恢复累加器
        SETB    ET0             ;开放 T0 中断
        RETI                    ;中断返回
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;              闪动调时程序                ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;T1 中断服务程序,用作时间调整时调整单元闪烁指示
INTT1:  PUSH    ACC             ;中断现场保护
        PUSH    PSW             ;
        MOV     TL1,    #0B0H   ;装定时器 T1 定时初值
        MOV     TH1,    #3CH    ;
```

```
            DJNZ    R2,INTT1OUT     ;0.3 s 未到退出中断(50 ms 中断 6 次)
            MOV     R2,#06H         ;重装 0.3 s 定时用初值
            CPL     02H             ;0.3 s 定时到对闪烁标志取反
            JB      02H,FLASH1      ;02H 位为 1 时显示单元"熄灭"
            MOV     72H,76H         ;02H 位为 0 时正常显示
            MOV     73H,77H         ;
            MOV     74H,78H         ;
            MOV     75H,79H         ;
INTT1OUT:   POP     PSW             ;恢复现场
            POP     ACC             ;
            RETI                    ;中断退出
FLASH1:     JB      01H,FLASH2      ;01H 位为 1 时,转小时熄灭控制
            MOV     72H,7AH         ;01H 位为 0 时,"熄灭符"数据放入分
            MOV     73H,7AH         ;显示单元(72H~73H),将不显示分数据
            MOV     74H,78H         ;
            MOV     75H,79H         ;
            AJMP    INTT1OUT        ;转中断退出
FLASH2:     MOV     72H,76H         ;01H 位为 1 时,"熄灭符"数据放入小时
            MOV     73H,77H         ;显示单元(74H~75H),小时数据将不显示
            MOV     74H,7AH         ;
            MOV     75H,7AH         ;
            AJMP    INTT1OUT        ;转中断退出
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;          加 1 子 程 序            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
      ADD1: MOV     A,@R0           ;取当前计时单元数据到 A
            DEC     R0              ;指向前一地址
            SWAP    A               ;A 中数据高 4 位与低 4 位交换
            ORL     A,@R0           ;前一地址中数据放入 A 中低 4 位
            ADD     A,#01H          ;A 加 1 操作
            DA      A               ;十进制调整
            MOV     R3,A            ;移入 R3 寄存器
            ANL     A,#0FH          ;高 4 位变 0
            MOV     @R0,A           ;放回前一地址单元
            MOV     A,R3            ;取回 R3 中暂存数据
            INC     R0              ;指向当前地址单元
            SWAP    A               ;A 中数据高 4 位与低 4 位交换
            ANL     A,#0FH          ;高 4 位变 0
            MOV     @R0,A           ;数据放入当削地址单元中
            RET                     ;子程序返回
            ;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;              清零程序            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;对计时单元复零用
CLR0:     CLR     A               ;清累加器
          MOV     @R0,A           ;清当前地址单元
          DEC     R0              ;指向前一地址
          MOV     @R0,A           ;前一地址单元清 0
          RET                     ;子程序返回
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;            时钟调整程序          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;当调时按键按下时进入此程序
SETMM:    cLR     ET0             ;关定时器 T0 中断
          CLR     TR0             ;关闭定时器 T0
          LCALL   DL1S            ;调用 1 s 延时程序
          JB      P3.7,CLOSEDIS   ;键按下时间小于 1 s,关闭显示(省电)
          MOV     R2,#06H         ;进入调时状态,赋闪烁定时初值
          SETB    ET1             ;允许 T1 中断
          SETB    TR1             ;开启定时器 T1
SET2:     JNB     P3.7,SET1       ;P3.7 口为 0(键未释放),等待
          SETB    00H             ;键释放,分调整闪烁标志置 1
SET4:     JB      P3.7,SET3       ;等待键按下
          LCALL   DL05S           ;有键按下,延时 0.5 s
          JNB     P3.7,SETHH      ;按下时间大于 0.5 s 转调小时状态
          MOV     R0,#77H         ;按下时间小于 0.5 s 加 1 min 操作
          LCALL   ADD1            ;调用加 1 子程序
          MOV     A,R3            ;取调整单元数据
          CLR     C               ;清进位标志
          CJNE    A,#60H,HHH      ;调整单元数据与 60 比较
HHH:      JC      SET4            ;调整单元数据小于 60 转 SET4 循环
          LCALL   CLR0            ;调整单元数据大于或等于 60 时清 0
          CLR     C               ;清进位标志
          AJMP    SET4            ;跳转到 SET4 循环
CLOSEDIS: SETB    ET0             ;省电(LED 不显示)状态,开 T0 中断
          SETB    TR0             ;开启 T0 定时器(开时钟)
CLOSE:    JB      P3.7,CLOSE      ;无按键按下,等待
          LCALL   DISPLAY         ;有键按下,调显示子程序延时消抖
          JB      P3.7,CLOSE      ;是干扰返回 CLOSE 等待
WAITH:    JNB     P3.7,WAITH      ;等待键释放
          LJMP    START1          ;返回主程序(LED 数据显示亮)
SETHH:    CLR     00H             ;分闪烁标志清除(进入调小时状态)
```

```
SETHH1:  JNB    P3.7,SET5       ;等待键释放
         SETB   01H             ;小时调整标志置1
  SET6:  JB     P3.7,SET7       ;等待按键按下
         LCALL  DL05S           ;有键按下延时0.5 s
         JNB    P3.7,SETOUT     ;按下时间大于0.5 s退出时间调整
         MOV    R0,#79H         ;按下时间小于0.5 s加1 h操作
         LCALL  ADD1            ;调加1子程序
         MOV    A,R3            ;
         CLR    C               ;
         CJNE   A,#24H,HOUU     ;计时单元数据与24比较
  HOUU:  JC     SET6            ;小于24转SET6循环
         LCALL  CLR0            ;大于或等于24时清0操作
         AJMP   SET6            ;跳转到SET6循环
SETOUT:  JNB    P3.7,SETOUT1    ;调时退出程序。等待键释放
         LCALL  DISPLAY         ;延时消抖
         JNB    P3.7,SETOUT     ;是抖动,返回SETOUT再等待
         CLR    01H             ;清调小时标志
         CLR    00H             ;清调分标志
         CLR    02H             ;清闪烁标志
         CLR    TR1             ;关闭定时器T1
         CLR    ET1             ;关定时器T1中断
         SETB   TR0             ;开启定时器T0
         SETB   ET0             ;开定时器T0中断(计时开始)
         LJMP   START1          ;跳回主程序
  SET1:  LCALL  DISPLAY         ;键释放等待时调用显示程序(调分)
         AJMP   SET2            ;防止键按下时无时钟显示
  SET3:  LCALL  DISPLAY         ;等待调分按键时时钟显示用
         AJMP   SET4
  SET5:  LCALL  DISPLAY         ;键释放等待时调用显示程序(调小时)
         AJMP   SETHH1          ;防止键按下时无时钟显示
  SET7:  LCALL  DISPLAY         ;等待调小时按键时时钟显示用
         AJMP   SET6
SETOUT1: LCALL  DISPLAY         ;退出时钟调整时键释放等待
         AJMP   SETOUT          ;防止键按下时无时钟显示
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;              显示程序                  ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; 显示数据在70H~75H单元内,用六位LED共阳数码管显示,P1口输出段码数据,P3口作
; 扫描控制,每个LED数码管亮1 ms时间再逐位循环。
DISPLAY: MOV    R1,#70H         ;指向显示数据首址
         MOV    R5,#0FEH        ;扫描控制字初值
  PLAY:  MOV    A,R5            ;扫描字放入A
```

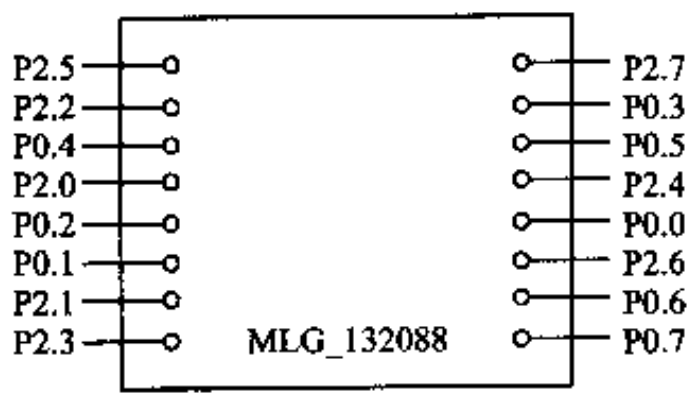
```
                MOV     P3,A              ;从 P3 口输出
                MOV     A,@R1             ;取显示数据到 A
                MOV     DPTR,#TAB         ;取段码表地址
                MOVC    A,@A+DPTR         ;查显示数据对应段码
                MOV     P1,A              ;段码放入 P1 口
                LCALL   DL1MS             ;显示 1 ms
                INC     R1                ;指向下一地址
                MOV     A,R5              ;扫描控制字放入 A
                JNB     ACC.5,ENDOUT      ;ACC.5=0 时一次显示结束
                RL      A                 ;A 中数据循环左移
                MOV     R5,A              ;放回 R5 内
                AJMP    PLAY              ;跳回 PLAY 循环
        ENDOUT: SETB    P3.5              ;一次显示结束,P3 口复位
                MOV     P1,#0FFH          ;P1 口复位
                RET                       ;子程序返回
         TAB: DB 0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H,0FFH
;共阳段码表          "0"  "1"  "2"  "3"  "4" "5" "6" "7"  "8" "9" "不亮"
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;              延时程序              ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;1 ms 延时程序,LED 显示程序用
         DL1MS: MOV     R6,#14H
         DL1:   MOV     R7,#19H
         DL2:   DJNZ    R7,DL2
                DJNZ    R6,DL1
                RET
;20 ms 延时程序,采用调用显示子程序以改善 LED 的显示闪烁现象
         DS20MS: ACALL  DISPLAY
                ACALL   DISPLAY
                ACALL   DISPLAY
                RET
;延时程序,用作按键时间的长短判断
         DL1S:  LCALL   DL05S
                LCALL   DL05S
                RET
         DL05S: MOV     R3,#20H           ;8 ms×32=0.256 s
         DL05S1: LCALL  DISPLAY
                DJNZ    R3,DL05S1
                RET
;
                END                       ;程序结束
```

第8章　实例3　8×8点阵LED字符显示器的设计

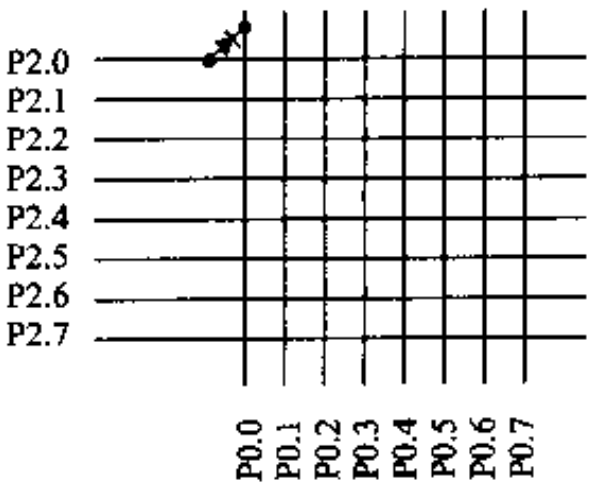
8×8点阵LED字符显示器能显示“电子设计”四个文字。显示方式可由K1、K2和K3选择，K1为逐字显示，K2为向上滚动显示，K3为向左滚动显示。

1. 系统硬件的设计

本字符显示器采用AT89C52单片机作控制器，12 MHz晶振，8×8点阵共阳LED显示器，其电路如图8.1所示。其中：P0作为字符数据输出口，P2为字符显示扫描输出口，第31脚(EA)接电源，P1.0～P1.2口分别接开关K1、K2、K3，改变电阻(270Ω)的大小可改变显示字符的亮度，驱动用9012三极管。



(a) 8×8点阵LED显示器引脚图



(b) 内部结构图

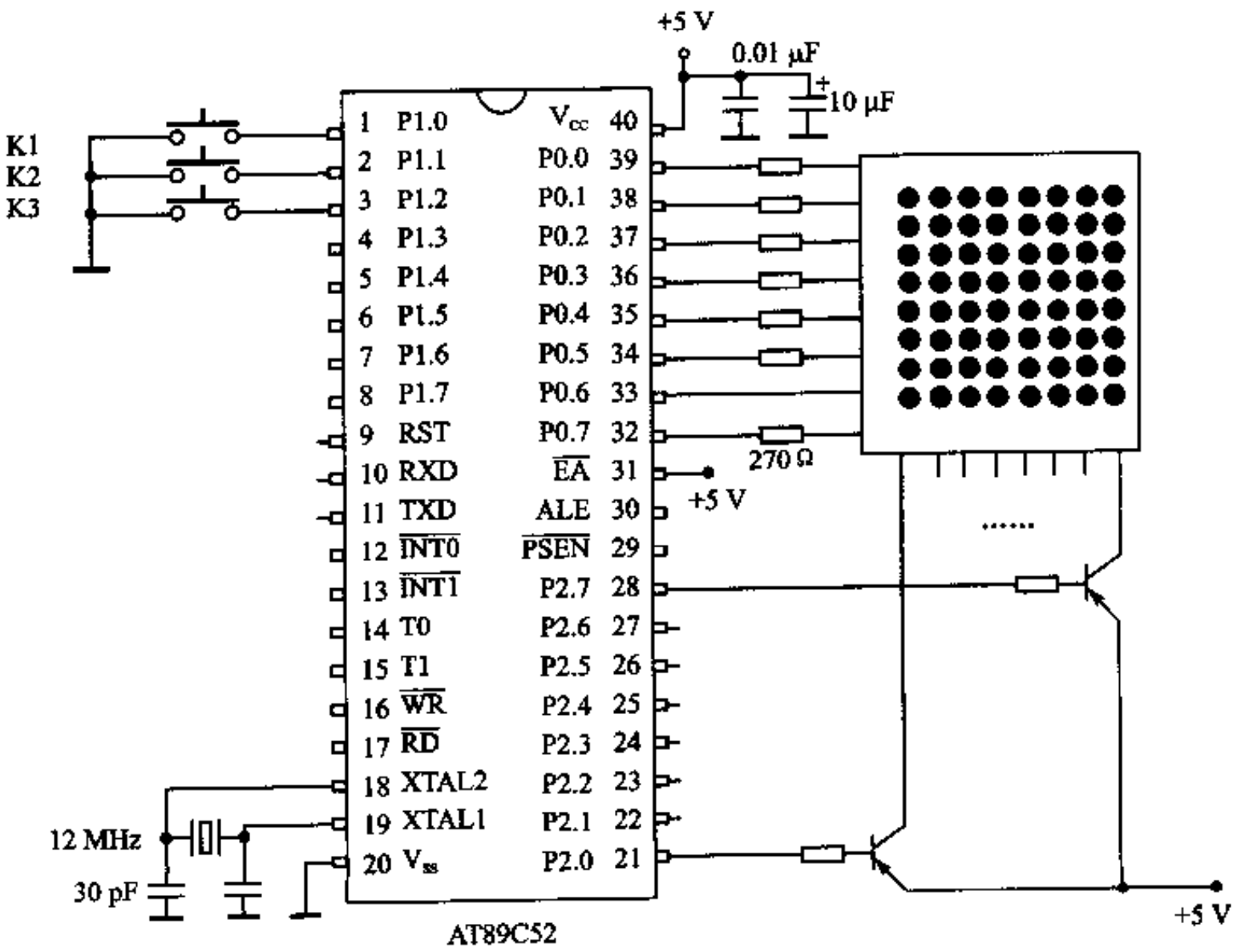


图 8.1　字符显示电路原理图

2. 系统主要程序的设计

(1) 主程序

主程序在刚上电时对系统进行初始化，然后读一次键开关状态，由键标志位值(00H、01H、02H)决定显示的方式。主程序流程图如 8.2 所示。

(2) 初始化程序

在系统初始化时，对四个端口进行复位，将显示用的字符数据从 ROM 表中装入内存单元 50H～6FH 中。“电子设计”中的每个文字占用 8 个地址单元。

(3) 显示程序

显示程序由显示主程序和显示子程序组成。显示主程序负责每次显示时的显示地址首址(在 B 寄存器中)、每个字的显示时间(由 30H 中的数据决定)和下一个显示地址的间隔(31H 中的数据决定)的处理。显示子程序则负责对指定 8 个地址单元的数据进行输出显示，显示一个完整文字的时间约为 8 ms。在显示子程序中，1 ms 延时程序是用调用键扫描子程序的方法实现的。图 8.3 为逐字显示及向上滚动显示方式时的显示控制程序流程图。

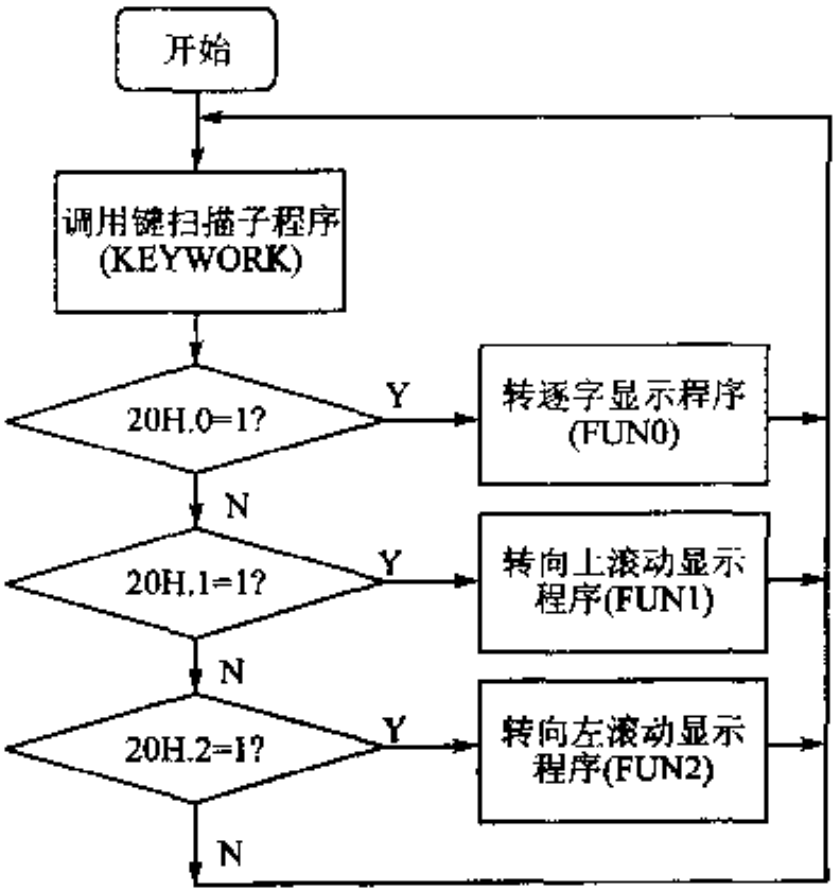


图 8.2 主程序流程图

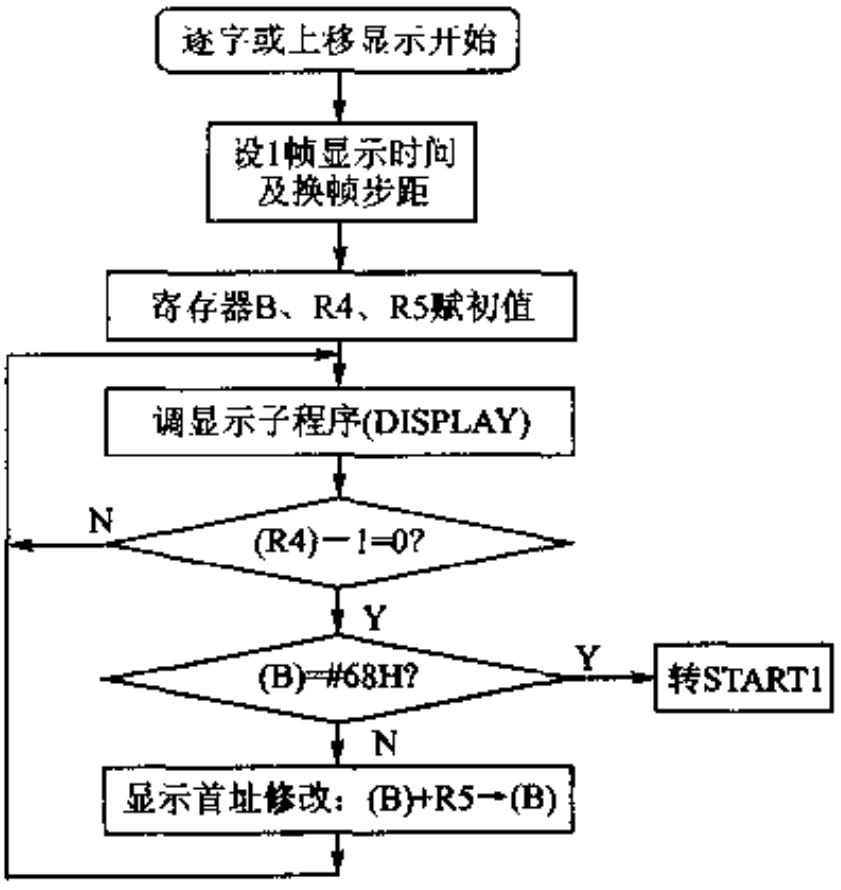


图 8.3 逐字显示及向上产动显示时的程序流程图

利用键扫描程序代替显示程序中的 1 ms 延时程序，既为了按键的快速响应，又可以提高动态显示的扫描频率，减少文字显示时的闪烁现象。对于多个文字的大屏幕显示，应该使用输出数据缓冲寄存器，才可以得到稳定的显示文字。

以下是 8×8 点阵 LED 字符显示器控制源程序：

```
;                    ************************
;                    *    电子屏字符显示器    *
;                    *      “电子设计”        *
;                    *      2001.10.23        *
;                    ************************
;          4 个显示字符数据表放在 50H～6FH 单元内，字符用 8×8 点阵，R4(30H)用于
;          控制显示静止字的时间，R5(31H)为静止字显示跳转地址步距，B 内放显示首址
```

```
;
;************;
;   中断入口程序   ;
;************;
;
ORG     0000H                   ;程序执行开始地址
LJMP    START                   ;跳至 START 执行
ORG     0003H                   ;外中断 0 中断入口地址
RETI                            ;中断返回(不开中断)
ORG     000BH                   ;定时器 T0 中断入口地址
RETI                            ;中断返回(不开中断)
ORG     0013H                   ;外中断 1 中断入口地址
RETI                            ;中断返回(不开中断)
ORG     001BH                   ;定时器 T1 中断入口地址
RETI                            ;中断返回(不开中断)
ORG     0023H                   ;串行口中断入口地址
RETI                            ;中断返回(不开中断)
ORG     002BH                   ;定时器 T2 中断入口地址
RETI                            ;中断返回(不开中断)
;
;************;
;   初始化程序   ;
;************;
CLEARMEN:   MOV     A,#0FFH     ;四端口置 1
            MOV     P1,A        ;
            MOV     P2,A        ;
            MOV     P3,A        ;
            MOV     P0,A        ;
            MOV     DPTR,#TAB   ;取"电子设计"字符表首址值
            CLR     A           ;
            MOV     21H,A       ;21H~24H 内存单元清 0
            MOV     22H,A       ;
            MOV     23H,A       ;
            MOV     24H,A       ;
            MOV     R3,A        ;R3 寄存器清 0
            MOV     R1,#50H     ;设字符表移入内存单元首址
            MOV     R2,#20H     ;设查表次数(32 次)
CLLOOP:     MOVC    A,@A+DPTR   ;查表将"电子设计"字符数据移入内存单元
            MOV     @R1,A       ;
            MOV     A,R3        ;
            INC     A           ;
            MOV     R3,A        ;
            INC     R1          ;
```

```
                DJNZ    R2,CLLOOP       ;查表 32 次,不到转 CLLOOP 再查
                RET                     ;子程序返回
;
;************;
;     主程序     ;
;************;
START:          MOV     20H,#00H        ;20H 内存单元清 0
                SETB    00H             ;20H.0 位置 1
START1:         LCALL   CLEARMEN        ;调用上电初始化子程序
                JB      00H,FUN0        ;20H.0 位为 1,执行 FUN0
                JB      01H,FUN1        ;20H.1 位为 1,执行 FUN1
                JB      02H,FUN2        ;20H.2 位为 1,执行 FUN2
                AJMP    START1          ;跳回 START1 循环
;
;************;
;   键扫描子程序   ;
;************;
KEYWORK:        MOV     P1,#0FFH        ;置输入状态
                JNB     P1.0,KEY1       ;P1.0 为 0(键按下)转 KEY1
                JNB     P1.1,KEY2       ;P1.1 为 0(键按下)转 KEY2
                JNB     P1.2,KEY3       ;P1.2 为 0(键按下)转 KEY3
KEYRET:         RET                     ;无键按下,子程序返回
;按键 1 功能处理
KEY1:           LCALL   DL10MS          ;延时 10 ms 消抖动
                JB      P1.0,KEYRET     ;是干扰转 KEYRET 结束
                SETB    00H             ;置逐字显示方式标志(20H.0=1)
                CLR     01H             ;
                CLR     02H             ;
                RET                     ;子程序返回
;按键 2 功能处理
KEY2:           LCALL   DL10MS          ;
                JB      P1.1,KEYRET     ;
                SETB    01H             ;置上移显示方式标志(20H.1=1)
                CLR     00H             ;
                CLR     02H             ;
                RET                     ;
;按键 3 功能处理
KEY3:           LCALL   DL10MS          ;
                JB      P1.2,KEYRET     ;
                SETB    02H             ;置左移显示方式标志(20H.2—1)
                CLR     01H             ;
                CLR     00H             ;
                RET
```

```
;
;逐字显示功能程序
FUN0:       MOV     30H,#80H        ;1 帧显示时间控制(约 1 s)
            MOV     31H,#08H        ;换帧跳转步距为 8
            LJMP    DISP1           ;转显示子程序 DISP1
; 上移显示功能程序
FUN1:       MOV     30H,#0AH        ;1 帧显示时间控制(约 80 ms)
            MOV     31H,#01H        ;换帧跳转步距为 1
            LJMP    DISP1           ;转显示子程序 DISP1
;左移显示功能程序
FUN2:       LJMP    DISP2           ;
;
;
;************;
;   显示控制程序   ;
;************;
DISP1:      MOV     B,#50H          ;显示数据首址
            MOV     R4,30H          ;放入 1 帧显示时间控制数据
            MOV     R5,31H          ;放入跳转步距控制数据
LOOP:       LCALL   DISPLAY         ;调用显示子程序一次
            DJNZ    R4,LOOP         ;1 帧显示时间未到再转 LOOP 循环
            MOV     R4,30H          ;1 帧显示时间到,重装初值
            MOV     A,B             ;
            CJNE    A,#68H,CONT     ;不是末地址转 CONT
            AJMP    START1          ;是末地址,一次显示结束跳回 START1
CONT:       ADD     A,R5            ;次帧扫描首址调整
            MOV     B,A             ;
            AJMP    LOOP            ;转 LOOP 进行次帧扫描
;
;显示子程序,字符数据从 P0 口输出,扫描控制字从 P2 口输出,显示 1 帧约需 8 ms
DISPLAY:    MOV     A,#0FFH         ;
            MOV     P0,A            ;关显示数据
            MOV     P2,A            ;关扫描
            MOV     R6,#0FEH        ;赋扫描字
            MOV     R0,B            ;赋显示数据首地址
            MOV     R7,#08H         ;一次扫描 8 行
DISLOOP:    MOV     A,@R0           ;取显示数据
            MOV     P0,A            ;放入 P0 口
            MOV     P2,R6           ;扫描输出(显示某一行)
            LCALL   DL1MS           ;亮 1ms
            INC     R0              ;指向下一行数据地址
            MOV     A,R6            ;扫描字移入 A
            RL      A               ;循环左移一位
            MOV     R6,A            ;放回 R6
```

```
                DJNZ    R7,DISLOOP      ;8 行扫描未完转 DISLOOP 继续
                RET                     ;8 行扫描结束
;
;左移显示控制程序
DISP2:          MOV     R5,#32          ;左移 32 次
DISP22:         LCALL   DISPP           ;调用左移显示控制子程序
                LCALL   MOVH            ;调用高位移出处理子程序 MOVH
                LCALL   MOVH1           ;调用高位移出处理子程序 MOVH1
                DJNZ    R5,DISP22       ;左移显示 32 次控制
                LJMP    START1          ;跳回主程序
;
;左移显示控制子程序
DISPP:          MOV     B,#50H          ;第一显示字符首址
                MOV     R4,#25H         ;1 帧显示时间控制
DISPP1:         LCALL   DISPLAY         ;调用显示子程序一次
                DJNZ    R4,DISPP1       ;1 帧显示时间不到转 DISPP 再循环
                RET
;
; 高位移出处理子程序,将"电子设计"四个字符数据的最高位移出至 21H~24H 单元内
MOVH:           MOV     R1,#21H         ;最高位移出存放单元首址
                MOV     R0,#50H         ;"电子设计"字符数据首址
                MOV     R2,#08H         ;每"字"移 8 次
MOV1:           MOV     A,@R0           ;取"电子设计"字符数据
                CLR     C               ;清进位 C
                RLC     A               ;带进位循环左移
                MOV     @R0,A           ;放回原单元
                MOV     A,@R1           ;存放单元数据入 A
                RRC     A               ;带进位循环右移
                MOV     @R1,A           ;放回存放单元
                INC     R0              ;字符数据地址加 1
                DJNZ    R2,MOV1         ;移 8 次未完转 MOV1 再移
                MOV     R2,#08H         ;8 次移完赋初值
                INC     R1              ;存放单元地址加 1
                MOV     A,R1            ;判断地址是否小于 25H
                SUBB    A,#25H          ;
                JZ      OUT             ;等于 25H 退出
                AJMP    MOV1            ;小于 25H 转 MOV1 继续
OUT:            RET                     ;子程序结束
;
;高位移出处理子程序
MOVH1:          MOV     A,21H           ;21H 与 22H、23H、24H 单元数据循交换
                XCH     A,24H           ;21H 与 24H 全交换
                XCH     A,23H           ;23H 与 24H 全交换
```

```
        XCH     A,22H           ;23H 与 22H 全交换
        MOV     21H,A           ;22H 与 21H 全交换
        MOV     R1,#21H         ;以下是重新组成显示字符数据表程序
        MOV     R0,#30H         ;将 21H～24H 的各位分别移入 50H～6FH 的低位
        MOV     R2,#08H         ;移位次数
MOV2:   MOV     A,@R0           ;取字符数据
        RR      A               ;右移
        MOV     @R0,A           ;放回原单元
        MOV     A,@R1           ;取原移出最高位存放单元数
        CLR     C               ;清 C
        RRC     A               ;带进位循环右移
        MOV     @R1,A           ;放回原单元
        MOV     A,@R0           ;取字符数据
        RLC     A               ;带进位循环左移
        MOV     @R0,A           ;放回字符数据
        INC     R0              ;字符数据地址加 1
        DJNZ    R2,MOV2         ;8 次未完转 MOV2 再继续
        MOV     R2,#08H         ;8 次完赋初值
        INC     R1              ;原移出最高位存放单元地址加 1
        MOV     A,R1            ;判断地址是否小于 25H
        SUBB    A,#25H          ;
        JZ      OUT             ;等于 25H 转 OUT 退出
        AJMP    MOV2            ;小于 25H 转 MOV2 继续
;
;1 ms 延时子程序,采用调用扫键子程序延时,可快速读出按钮的状态
DL1MS:  MOV     R3,#64H         ;100×(10+2) μs
LOOPK:  LCALL   KEYWORK
        DJNZ    R3,LOOPK
        RET
;
;0.5 ms 延时子程序
DL512:  MOV     R2,#0FFH
LOOP1:  DJNZ    R2,LOOP1
        RET
;
;10 ms 延时子程序
DL10MS: MOV     R3,#14H
LOOP2:  LCALL   DL512
        DJNZ    R3,LOOP2
        RET
;
;"电子设计"显示用 ROM 数据表
TAB:    DB      0EFH,83H,0ABH,83H,0ABH,83H,0EEH,0E0H      ;电
```

```
DB      0FFH,0C7H,0EFH,83H,0EFH,0EFH,0CFH,0EFH    ;子
DB      0B1H,0B5H,04H,0BFH,0B1H,0B5H,9BH,0A4H     ;设
DB      0BBH,0BBH,1BH,0A0H,0BBH,0BBH,9BH,0BBH     ;计
DB      00H,00H,00H,00H
END                                               ;程序结束
```

第 9 章　实例 4　8 路输入模拟信号数值显示电路的设计

本显示器可自动轮流显示 8 路输入模拟信号的数值，最小分辨率为 0.02 V，最大显示数值为 255(输入为 5 V 时)，模拟输入最大值为 5 V，可作为数字电压表用。

1. 系统硬件电路的设计

如图 9.1 所示，8 路输入模拟信号数值显示电路由 A/D 转换、数据处理及显示控制等组成。A/D 转换由集成电路 0809 完成。0809 具有 8 路模拟输入端口，地址线(23～25 脚)可决定对哪一路模拟输入作 A/D 转换。第 22 脚为地址锁存控制，当输入为高电平时，对地址信号进行锁存；6 脚为测试控制，当输入一个 2 μs 宽高电平脉冲时，就开始 A/D 转换；7 脚为 A/D 转换结束标志，当 A/D 转换结束时，7 脚输出高电平；9 脚为 A/D 转换数据输出允许控制，当 OE 脚为高电平时，A/D 转换数据从端口输出；10 脚为 0809 的时钟输入端，利用单片机 30 脚的六分频晶振信号再通过 14024 二分频得到。单片机的 P1、P3 端口作四位 LED 数码管显示控制，P0 端口作 A/D 转换数据读入用，P2 端口用作 0809 的 A/D 转换控制。

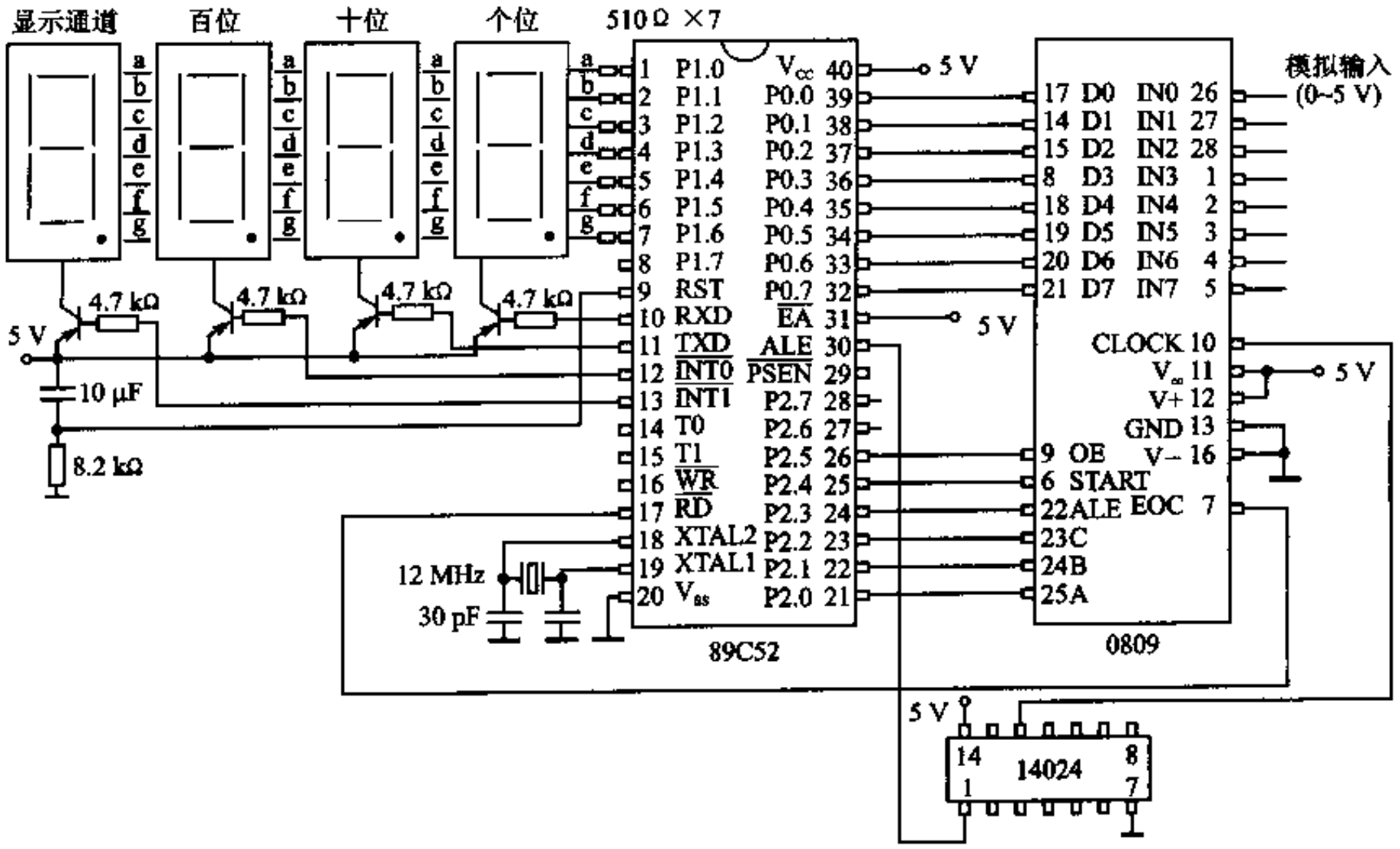


图 9.1　8 路输入模拟信号数值显示电路

2. 系统主要程序的设计

(1) 初始化程序

系统上电时，将 70H～77H 内存单元清零，P2 口置零。

(2) 主程序

在刚上电时，因 70H～77H 内存单元的数据为 0，则每一通道的数码管显示值都为 000。当进行一次测量后，将显示出每一通道的 A/D 转换值。每个通道的数据显示时间在 1 s 左右。主程序在调用显示程序和测试程之间循环，其流程图如 9.2 所示。

(3) 显示子程序

采用动态扫描法实现四位数码管的数值显示。测量所得的 A/D 转换数据放在 70H～77H 内存单元中。测量数据在显示时需经过转换成为十进制 BCD 码放在 78H～7BH 中，其中 7BH 存放通道标志数。寄存器 R3 用来作 8 路循环控制，R0 用作显示数据地址指针。

(4) 模数转换测量子程序

模数转换测量子程序是用来控制对 0809 8 路模拟输入电压的 A/D 转换，并将对应的数值移入 70H～77H 内存单元，其程序流程如图 9.3 所示。

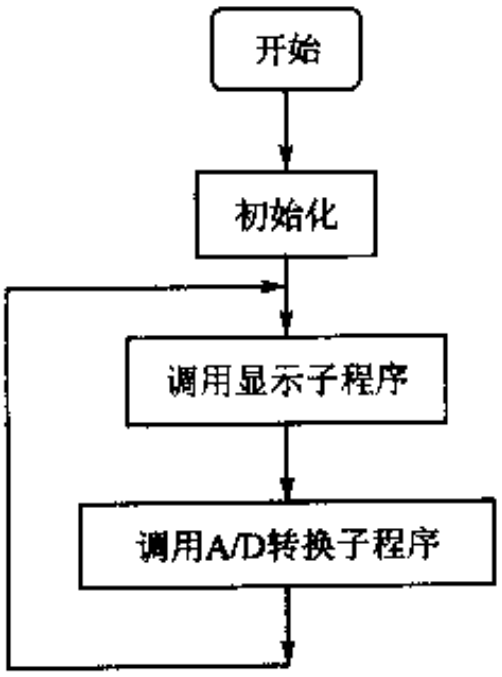


图 9.2　主程序流程图

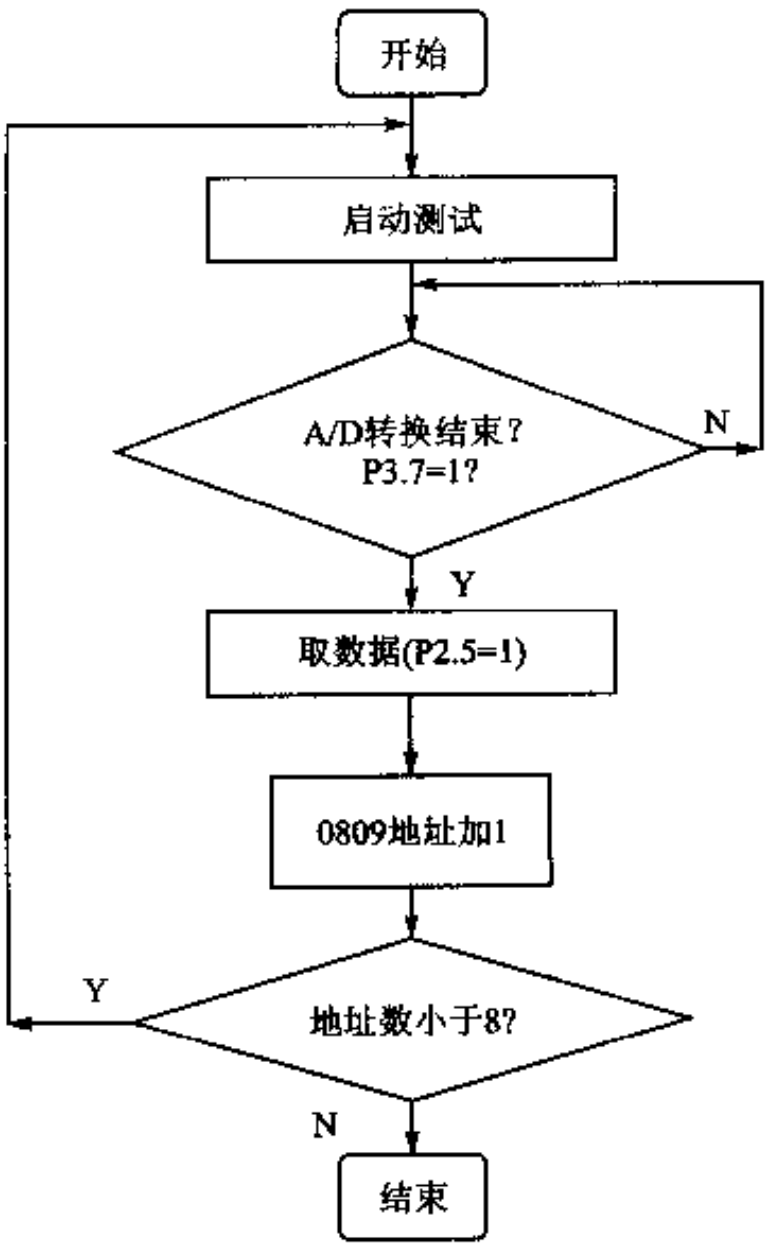


图 9.3　A/D 转换测量程序流程图

以下是 8 路输入模拟信号数值显示电路的控制源程序：

```
;**************************;
;      8 路模拟数据采集显示电路      ;
;          2001.10.08          ;
;**************************;

;
;70H~77H 存放采样值，78H~7BH 存放显示数据，依次为个位、十位、百位、通道标志
;
```

```
;*****************************************************
;*                                                   *
;*                  主程序和中断程序入口             *
;*                                                   *
;*****************************************************
        ORG     0000H           ;程序执行开始地址
        LJMP    START           ;跳至 START 执行
        ORG     0003H           ;外中断 0 中断入口地址
        RETI                    ;中断返回(不开中断)
        ORG     000BH           ;定时器 T0 中断入口地址
        RETI                    ;中断返回(不开中断)
        ORG     0013H           ;外中断 1 中断入口地址
        RETI                    ;中断返回(不开中断)
        ORG     001BH           ;定时器 T1 中断入口地址
        RETI                    ;中断返回(不开中断)
        ORG     0023H           ;串行口中断入口地址
        RETI                    ;中断返回(不开中断)
        ORG     002BH           ;定时器 T2 中断入口地址
        RETI                    ;中断返回(不开中断)
;
;*****************************************************
;*                                                   *
;*                初始化程序中的各变量               *
;*                                                   *
;*****************************************************
CLEARMEMIO: CLR     A           ;
        MOV     P2,A            ;P2 口置 0
        MOV     R0,#70H         ;内存循环清 0(70H～7BH)
        MOV     R2,#0CH         ;
LOOPMEM:    MOV     @R0,A       ;
        INC     R0              ;
        DJNZ    R2,LOOPMEM      ;
        MOV     A,#0FFH         ;
        MOV     P0,A            ;P0、P1、P3 端口置 1
        MOV     P1,A            ;
        MOV     P3,A            ;
        RET                     ;子程序返回
;
;*****************************************************
;*                                                   *
;*                    主 程 序                       *
;*                                                   *
;*****************************************************
```

```
START:    LCALL   CLEARMEM0       ;初始化
MAIN:     LCALL   DISPLAY         ;显示数据一次
          LCALL   TEST            ;测量一次
          AJMP    MAIN            ;返回 MAIN 循环
          NOP                     ;PC 值出错处理
          NOP                     ;空操作
          NOP                     ;空操作
          LJMP    START           ;重新复位起动
;
DISPLAY:  MOV     R3,#08H         ;8 路信号循环显示控制
          MOV     R0,#70H         ;显示数据初址(70H~77H)
          MOV     7BH,#00H        ;显示通道路数(0~7)
DISLOOP1: MOV     A,@R0           ;显示数据转为三位十进制 BCD 码存入
          MOV     B,#100          ;7AH、79H、78H 显示单元内
          DIV     AB              ;显示数据除 100
          MOV     7AH,A           ;商入 7AH
          MOV     A,#10           ;A 放入数 10
          XCH     A,B             ;余数与数 10 交换
          DIV     AB              ;余数除 10
          MOV     79H,A           ;商入 79H
          MOV     78H,B           ;余数入 78H
          MOV     R2,#0FFH        ;每路显示时间控制 4 ms×255
DISLOOP2: LCALL   DISP            ;调四位 LED 显示程序
          DJNZ    R2,DISLOOP2     ;每路显示时间控制
          INC     R0              ;显示下一路
          INC     7BH             ;通道显示数值加 1
          DJNZ    R3,DISLOOP1     ;8 路显示未完转 DISLOOP1 再循环
          RET                     ;8 路显示完子程序结束
;
;LED 共阳显示子程序,显示内容在 78H~7BH,数据在 P1 输出,列扫描在 P3.0~P3.3 口
DISP:     MOV     R1,#78H         ;赋显示数据单元首址
          MOV     R5,#0FEH        ;扫描字
PLAY:     MOV     P1,#0FFH        ;关显示
          MOV     A,R5            ;取扫描字
          ANL     P3,A            ;开显示
          MOV     A,@R1           ;取显示数据
          MOV     DPTR,#TAB       ;取段码表首址
          MOVC    A,@A+DPTR       ;查显示数据对应段码
          MOV     P1,             ;段码放入 P1 口
          LCALL   DL1MS           ;显示 1 ms
          INC     R1              ;指向下一地址
          MOV     A,P3            ;取 P3 口扫描字
          JNB     ACC.3,ENDOUT    ;四位显示完转 ENDOUT 结束
```

```
                RL      A                ;扫描字循环左移
                MOV     R5,A             ;扫描字放入 R5 暂存
                MOV     P3,#0FFH         ;显示暂停
                AJMP    PLAY             ;转 PLAY 循环
ENDOUT:         MOV     P3,#0FFH         ;显示结束,端口置 1
                MOV     P1,#0FFH         ;
                RET                      ;子程序返回
;
;LED 数码显示管用共阳段码表,分别对应 0～9,最后一个是"熄灭符"
TAB:            DB      0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H,0FFH
;
;1ms 延时子程序,LED 显示用
DL1ms:          MOV     R6,#14H
DL1:            MOV     R7,#19H
DL2:            DJNZ    R7,DL2
                DJNZ    R6,DL1
                RET
;
;*****************************************
;*                                       *
;*          模数转换测量子程序           *
;*                                       *
;*****************************************
TEST:           CLR     A                ;清累加器 A
                MOV     P2,A             ;清 P2 口
                MOV     R0,#70H          ;转换值存放首址
                MOV     R7,#08H          ;转换 8 次控制
                LCALL   TESTART          ;启动测试
WAIT:           JB      P3.7,MOVD        ;等 A/D 转换结束信号后转 MOVD
                AJMP    WAIT             ;P3.7 为 0 等待
;
TESTART:        SETB    P2.3             ;锁存测试通道地址
                NOP                      ;延时 2 μs
                NOP                      ;
                CLR     P2.3             ;测试通道地址锁存完毕
                SETB    P2.4             ;启动测试,发开始脉冲
                NOP                      ;延时 2 μs
                NOP                      ;
                CLR     P2.4             ;发启动脉冲完毕
                NOP                      ;延时 4 μs
                NOP                      ;
                NOP                      ;
                NOP                      ;
```

```
                RET                         ;子程序调用结束
;
;取 A/D 转换数据至 70H～77H 内存单元
MOVD:           SETB    P2.5                ;0809 输出允许
                MOV     A,P0                ;将 A/D 转换值移入 A
                MOV     @R0,A               ;放入内存单元
                CLR     P2.5                ;关闭 0809 输出
                INC     R0                  ;内存地址加 1
                MOV     A,P2                ;通道地址移入 A
                INC     A                   ;通道地址加 1
                MOV     P2,A                ;通道地址送 0809
                CLR     C                   ;清进位标志
                CJNE    A,#08H,TESTCON      ;通道地址不等于 8 转 TESTCONT 再测试
                JC      TESTCON             ;通道地址小于 8 转 TESTCONT 再测试
                CLR     A                   ;大于或等于 8,A/D 转换结束,恢复端口
                MOV     P2,A                ;P2 口置 0
                MOV     A,#0FFH             ;
                MOV     P0,A                ;P0 口置 1
                MOV     P1,A                ;P1 口置 1
                MOV     P3,A                ;P3 口置 1
                RET                         ;取 A/D 转换数据结束
TESTCON:        LCALL   TESTART             ;再发测试启动脉冲
                LJMP    WAIT                ;跳至 WAIT 等待 A/D 转换结束信号
;
                END                         ;程序结束
```

第 10 章　实例 5　单键学习型遥控器的设计

利用单键学习型遥控器可以学习任何遥控器的某个按键功能。单键学习型遥控器采用最小化应用模式设计，电路简单，可靠性高，尤其是通过大量不同遥控码的特征分析，在遥控码的读入时选择了最佳采样频率，使遥控码的学习成功率大大提高。此技术可应用于多媒体教室、家庭集中控制器等设备。使用时先按一下 K，待绿色指示灯亮后，用遥控器对准红外接收头，按某个功能按键，当绿灯灭且红灯亮时说明学习完成，按发射键即可进行遥控。

1. 系统硬件电路的设计

图 10.1 为单键学习型遥控器的电原理图，其中 P1.0 口接遥控码发射按键，P1.6 口用作状态指示，绿灯亮代表学习状态。P1.7 口用于指示控制键的操作，闪烁时表示遥控码正在发射之中。处在学习状态，绿灯灭表示码已读入。第 9 脚为单片机的复位脚，采用简单的 RC 上电复位电路；12 脚为中断输入口，用于工作方式的转换控制，当$\overline{\text{INT0}}$脚为低电平时，系统进入学习状态；14 脚用于红外线接收头的输出信号输入；15 脚作为遥控码的输出口，用于输出 40 kHz的遥控码；18、19 脚接 12 MHz 晶振。由于采用最小化应用系统，控制线$\overline{\text{PSEN}}$(片外取指控制)、ALE(地址锁存控制)不用，$\overline{\text{EA}}$(片外存储器选择)接高电平，使低 8 KB 的 E^2PROM 地址(0000H～1FFFH)指向片内。

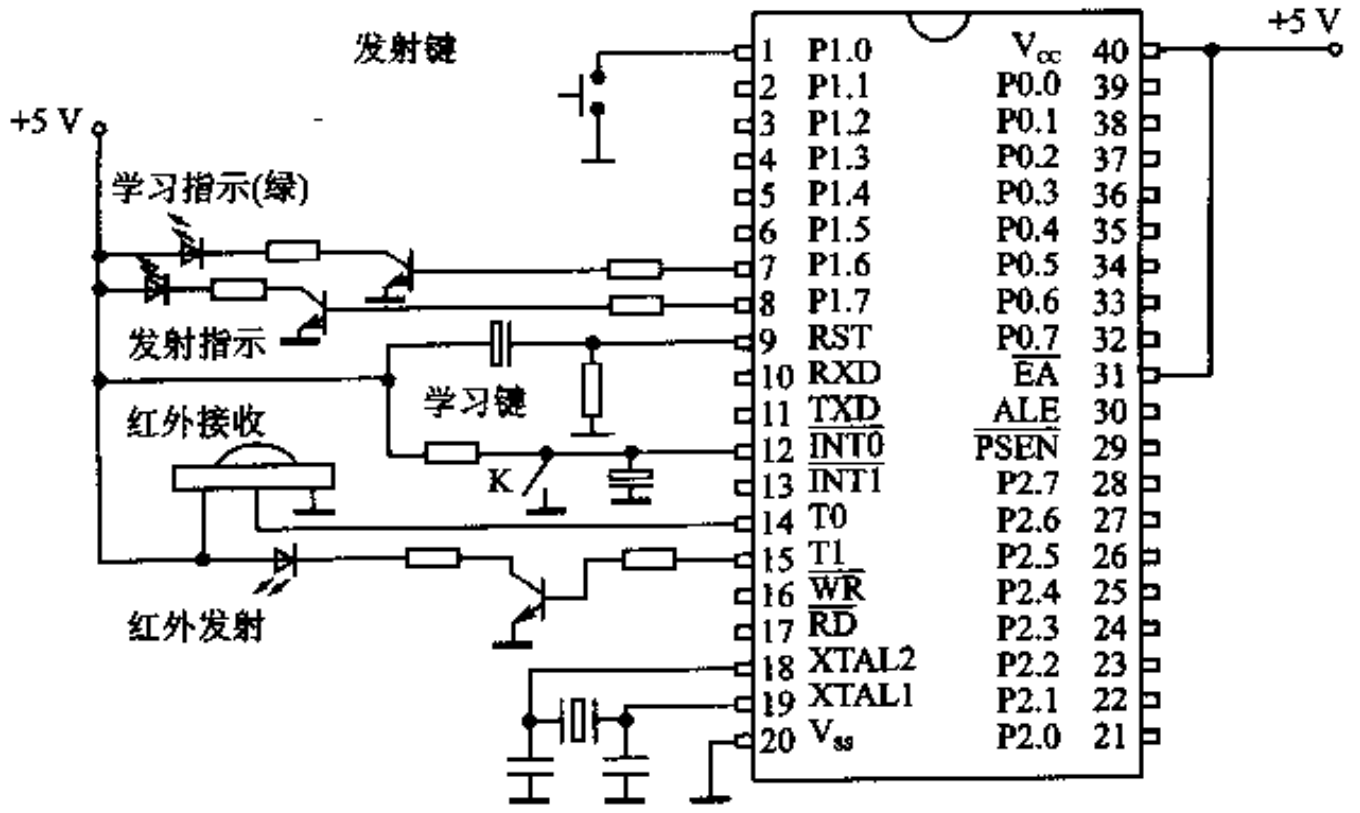


图 10.1　单键学习型遥控器电路原理图

2. 系统主要程序的设计

(1) 初始化程序

初始化程序内容包括 P0、P1、P3 口置位，P2 口清零，清 08H～6EH 共 103 个工作寄存器，设置堆栈基址(70H)，设置计数器计数模式、控制字，开外中断允许等。

(2) 遥控码读入处理程序

遥控码读入处理程序可以完成遥控码起始位的识别、脉宽计数功能，完成遥控码编码位的

宽度计数功能，完成结束位的认别功能，其流程图如图 10.2 所示。本程序模块在编程设计中非常重要，通过大量的不同种类的遥控码波形实验测试分析，遥控码的帧间歇位宽度均在 10 ms以上，起始位码宽度在 100 μs～20 ms 之间，编码位在 100 μs～5 ms 之间。

为确保所有遥控器学习的成功，采用以下设计方法：

寻找起始位方法：用 16 位 DPTR 计数器对高电平进行宽度计数，计数采样周期为 21 μs；当高电平结束时，如高 8 位计数器为非零，则说明高电平宽度超过 5.355 ms(255×21 μs)，接下来的低电平码就是起始位，否则重新开始。

读起始位方法：采用 16 位 DPTR 对低电平进行宽度计数(最大可读宽度为 1.376 s)，当高电平跳变时结束计数，并将 DPTR 的高 8 位、低 8 位分别存入 R_4、R_5 寄存器。

读遥控编码的方法：采用 DPTR 低 8 位计数器对码(高电平或低电平)进行宽度计数，电平跳变时结束计数，并将值存入规定的地址；在高电平码计数时，如 DPTR 高 8 位计数器为非零(宽度大于 5.355 ms)，则判定为结束帧间隔位，在相应存储单元写入数据 #00H 作为结束标志。

(3) 遥控码发送处理程序

遥控码发送处理程序利用计数器计数中断功能，实现 40 kHz 载波的发送，利用接收时接收的低电平位时间，控制载波的发送时间。

(4) 主程序

主程序在上电初始化后进行端口按键扫描，当确认有键按下时，将编码发出去，其流程图如图 10.3 所示。

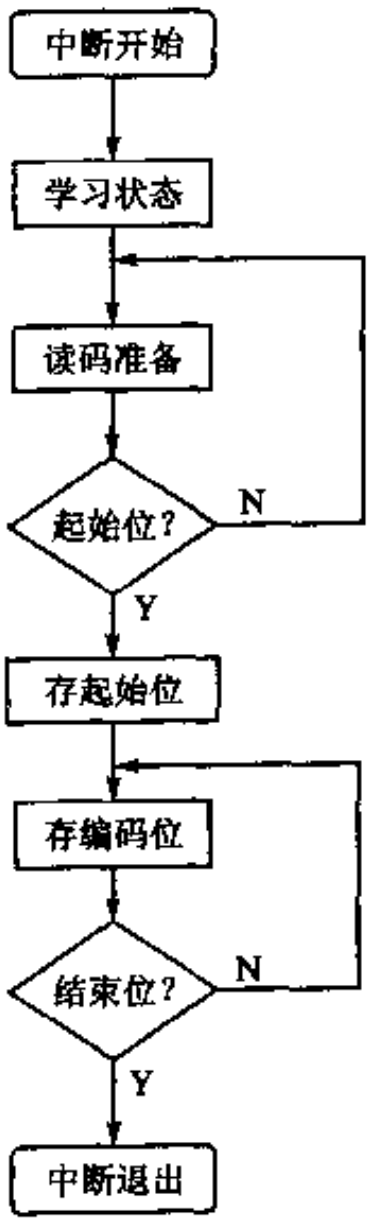


图 10.2　遥控码读入处理程序流程图

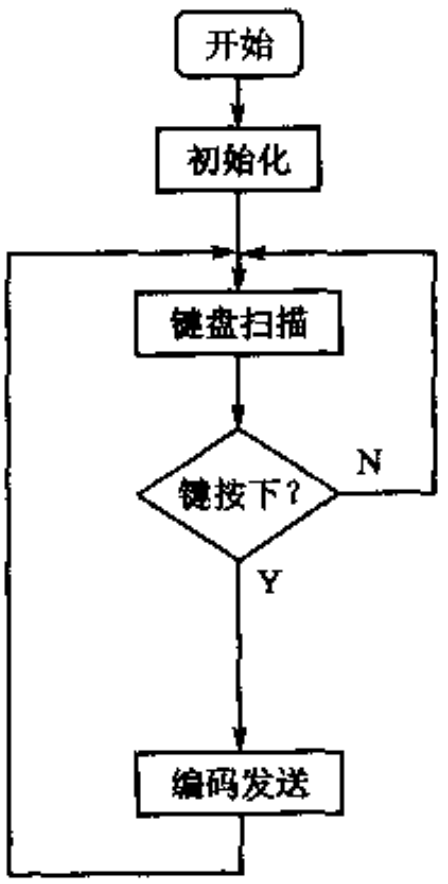


图 10.3　主程序流程图

(5) 延时程序

延时程序主要用于读键时消抖。

3. 电路主要性能指标

(1) 最大学习码长:206 b;

(2) 学习码识别范围:起始位:21 μs～1.376 s,编码位:21 μs～5.355 ms;

(3) 读码误差:±21 μs ;

(4) 帧间歇位识别范围:小于 1.37 s,大于 5.355 ms。

单键学习型遥控器的设计性能与软件的编写具有密切的关系,特别是码宽计数的采样周期及计数器采用 16 位还是 8 位,都关系到能否识别起始位及遥控码采样精度问题,所以码宽计数的采样周期等在编程时须经多次实验测试后才能决定。本设计中读码采样周期为 21 μs。

以下是单键学习型遥控器完整源程序:

```
;                  ************************************
;                  *          单键学习型遥控器          *
;                  *            2001.11.29              *
;                  ************************************
;
;
;*****************************************************************
;*                                                               *
;*                          P1.0  | 1        40 | VCC           *
;*                          P1.1  | 2        39 | P0.0          *
;*                          P1.2  | 3        38 | P0.1          *
;*                          P1.3  | 4        37 | P0.2          *
;*                          P1.4  | 5        36 | P0.3          *
;*                          P1.5  | 6        35 | P0.4          *
;*          STUDYLAMP       P1.6  | 7        34 | P0.5          *
;*              LAMP        P1.7  | 8        33 | P0.6          *
;*                          RST   | 9        32 | P0.7          *
;*              RXD         P3.0  | 10 MCS-51 31 | EA      VDD   *
;*              TXD         P3.1  | 11       30 | ALE           *
;*      STUDYKEY INT0       P3.2  | 12       29 | PSEN          *
;*              INT1        P3.3  | 13       28 | P2.7          *
;*          REMOTEIN T0     P3.4  | 14       27 | P2.6          *
;*          REMOTEOUT T1    P3.5  | 15       26 | P2.5          *
;*              WR          P3.6  | 16       25 | P2.4          *
;*              RD          P3.7  | 17       24 | P2.3          *
;*                          XTAL2 | 18       23 | P2.2          *
;*                          XTAL1 | 19       22 | P2.1          *
;*                          VSS   | 20       21 | P2.0          *
;*****************************************************************
;
```

```
SPBASE      EQU     70H     ;堆栈基址
IEVAL       EQU     00H     ;关所有中断
MEMBASE     EQU     08H     ;工作寄存器基址
MEMS        EQU     67H     ;工作寄存器个数
;
BITNMB      EQU     08H     ;一个字节包含8个位
;
KEYFUNFLAG  EQU     80H     ;键功能索引
KEYFUNNMB   EQU     81H     ;键功能号
KEYFUNRW    EQU     82H     ;遥控信号读/写标志
READFLAG    EQU     88H     ;读标记
WITERFLAG   EQU     99H     ;写标记
;
TMPHADDR    EQU     08H     ;读人高电平存放首址
TMPLADDR    EQU     90H     ;读人低电平存放首址
READTIME    EQU     00H     ;读人数据指令时间
LOWH        EQU     R4      ;起始位存放高地址
LOWL        EQU     R5      ;起始位存放低地址
STUDYLAMP   EQU     P1.6    ;学习指示灯
LAMP        EQU     P1.7    ;未定义指示灯
STUDYKEY    EQU     P3.2    ;学习键

REMOTEIN    EQU     P3.4    ;遥控输人
REMOTEOUT   EQU     P3.5    ;遥控输出
DELAYCONUT  EQU     30H     ;延时值
DELAYCONUT0 EQU     0FFH    ;延时值
T1COUNT     EQU     0F3H    ;T1计数初值
TMODVAL     EQU     22H     ;计数模式控制字
TCONVAL     EQU     41H     ;计数控制寄存器值
PCONVAL     EQU     00H     ;电源控制寄存器值
T2CONVAL    EQU     00H     ;T2控制寄存器值
SCONVAL     EQU     0F8H    ;串口控制寄存器值
IPVAL       EQU     01H     ;中断优先级控制值
;
;
;*************************************
;*                                   *
;*          主程序和中断程序人口     *
;*                                   *
;*************************************
            ORG     0000H           ;程序执行开始地址
            AJMP    START           ;跳至START执行
            ORG     0003H           ;外中断0中断人口地址
```

```
        AJMP    INTEX0          ;跳至 INTEX0 中断服务程序
        ORG     000BH           ;定时器 T0 中断入口地址
        RETI                    ;中断返回(不开中断)
        ORG     0013H           ;外中断 1 中断入口地址
        RETI                    ;中断返回(不开中断)
        ORG     001BH           ;定时器 T1 中断入口地址,
        AJMP    INTT1           ;跳至 INTT1 中断服务程序
        ORG     0023H           ;串行口中断入口地址
        RETI                    ;中断返回(不开中断)
        ORG     002BH           ;定时器 T2 中断入口地址
        RETI                    ;中断返回(不开中断)
;
;*****************************************
;*                                       *
;*          初始化程序中的各变量          *
;*                                       *
;*****************************************
CLEARMEMIO:  CLR    A               ;A 清 0
             DEC    A               ;A 为#FFH
             MOV    P0,A            ;P0 口置 1
             MOV    P3,A            ;P3 口置 1
             MOV    P1,A            ;P1 口置 1
             CLR    A               ;清 A(为 0)
             MOV    P2,A            ;P2 口为 0
             CLR    STUDYLAMP       ;关学习灯
             CLR    LAMP            ;关操作灯
             CLR    REMOTEOUT       ;关遥控码输出
             SETB   REMOTEIN        ;遥控接收为输入状态
             MOV    R0,#MEMBASE     ;清工作寄存器,从 08H 开始
             MOV    R1,#MEMS        ;清内存个数(为 103 个)
CLEARMEM:    MOV    @R0,A           ;清 0 开始
             INC    R0              ;地址加 1
             DJNZ   R1,CLEARMEM     ;未清完转 CLEARMEM 继续
;            MOV    R0,#KEYFUNRW
;            MOV    @R0,#READFLAG
             MOV    SP,#SPBASE      ;设堆栈基址(70H)
             MOV    IE,#IEVAL       ;关所有中断
             MOV    IP,#IPVAL       ;置优先级
             MOV    TMOD,#TMODVAL   ;置计数器模式(8 位自动重装初值
                                    模式)
             MOV    PCON,#PCONVAL   ;波特率不加倍
```

```
                MOV     SCON,#SCONVAL       ;串口中断不开
                MOV     TH1,#T1COUNT        ;T1 定时器初值(定时值为 13μs)
                MOV     TL1,#T1COUNT        ;T1 定时器初值
                SETB    EX0                 ;允许外中断 0 中断
                SETB    EA                  ;开总中断允许
                RET                         ;子程序结束
;
;*************************************
;*                                   *
;*            主   程   序           *
;*                                   *
;*************************************
START:          LCALL   CLEARMEMIO          ;调用上电初始化子程序
;主程序
MAIN:           LCALL   KEYWORK             ;调用读键子程序
                LJMP    MAIN                ;跳回 MAIN 循环
                NOP                         ;PC 值出错处理
                NOP                         ;空操作
                NOP                         ;
                LJMP    START               ;重新初始化
;
;*************************************
;*                                   *
;*          T1 中断服务程序          *
;*                                   *
;*************************************
INTT1:          CPL     REMOTEOUT           ;40 kHz 方波输出(红外线调制波)
                RETI                        ;中断返回
;
;*************************************
;*                                   *
;*            载波合成               *
;*                                   *
;*************************************
REMOTETX:       MOV     R0,#TMPHADDR        ;取遥控码高电平存放首址
                MOV     R1,#TMPLADDR        ;取遥控码低电平存放首址
                SETB    LAMP                ;开操作灯
                MOV     A,R4                ;起始位高 8 位放入 A
                MOV     R3,A                ;放入 R3 暂存
                JZ      LOWBACK             ;高 8 位为 0 转 LOWBACK 处理低 8 位
                CLR     A                   ;高 8 位非 0 处理
                DEC     A                   ;A 为 #FFH
LOWBACKTMP:     MOV     R2,A                ;起始位复原,R2 赋初值
```

```
LOWBACKTMP0:  SETB    TR1                ;开启 T1
              SETB    ET1                ;允许 T1 中断
              NOP                        ;用空操作延时
              NOP
              NOP
              NOP
              NOP
              NOP
              DJNZ    R2,LOWBACKTMP0     ;计数值每减 1 循环时间约为 21 μs
              DJNZ    R3,LOWBACKTMP      ;高 8 位计数值不为 0 转 LOW-
                                          BACKTMP
LOWBACK:      MOV     A,R5               ;起始位低 8 位处理
              MOV     R2,A               ;起始位低 8 位数放入 R2 暂存
LOWBACKTMP1:  SETB    TR1                ;开启 T1
              SETB    ET1                ;允许 T1 中断
              NOP                        ;用空操作延时
              NOP
              NOP
              NOP
              NOP
              NOP
              DJNZ    R2,LOWBACKTMP1     ;低 8 位计数值不为 0 转 LOW-
                                          BACKTMP1
TMP0:         MOV     A,@R0              ;高电平处理开始,取高电平数据
              MOV     R2,A               ;放入 R2
TMP1:         CLR     TR1                ;关闭 T1
              CLR     ET1                ;关闭 T1 中断允许(关 40 kHz 红
                                          外线)
              CLR     REMOTEOUT          ;关遥控输出
              NOP                        ;空操作延时
              NOP
              NOP
              NOP
              NOP
              NOP
              NOP
              NOP
              NOP
              NOP
              NOP
              NOP
              NOP
              NOP
              NOP
              NOP
```

```
                NOP
                DJNZ    R2,TMP1         ;R2 每减 1 循环时间约为 21 μs
                INC     R0              ;指向下一高电平数据地址
TMPP:           MOV     A,@R1           ;取低电平数据
                MOV     R2,A            ;放入 R2
TMP2:           SETB    TR1             ;低电平处理,开定时器 T1
                SETB    ET1             ;开 T1 中断
                NOP                     ;空操作延时
                NOP
                NOP
                NOP
                NOP
                NOP
                DJNZ    R2,TMP2         ;减 1 不为 0 转 TMP2 循环(周期为
                                         21 μs)
                INC     R1              ;指向下一低电平数据
                MOV     A,@R1           ;取数据
                JZ      OUT             ;为 0 转 OUT 退出
                AJMP    TMP0            ;不为 0 转 TMP0 执行
OUT:            CLR     TR1             ;退出程序,关 T1
                CLR     ET1             ;关 T1 中断
                CLR     LAMP            ;关操作灯
                CLR     REMOTEOUT       ;关遥控输出
                RET                     ;返回
;
;*************************************************
;*              遥控数据读取 INT0 中断程序          *
;*              高电平存 TMPHADDR 为首址 RAM        *
;*              低电平存 TMPLADDR 为首址 RAM        *
;*************************************************
INTEX0:         CLR     ET1             ;关 T1 中断允许
                CLR     TR1             ;关定时器 T1
                CLR     EX0             ;关外中断 0
                CLR     EA              ;关中断总允许
                SETB    STUDYLAMP       ;开学习状态指示灯
                CLR     LAMP            ;关操作灯
                MOV     R0,#TMPHADDR    ;高电平首址放入 R0(07H)
                MOV     R1,#TMPLADDR    ;低电平首址放入 R1(90H)
                CLR     A               ;A 清 0
                MOV     DPH,A           ;DPTR 寄存器清 0
                MOV     DPL,A           ;
READHEAD:       JNB     REMOTEIN,READDATA ;寻找起始位。当输入为 0 时转
                                           READDATA
```

```
                INC     DPTR                    ;输入为高电平时对 DPTR 循环计数
                NOP                             ;空操作延时
                NOP                             ;(循环周期约为 21 μs)
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                AJMP    READHEAD                ;跳回循环
;判断是不是遥控码起始位
READDATA:       CJNE    A,DPH,READDATA00        ;DPTR 高 8 位不为 0,是起始位
                MOV     DPH,A                   ;DPTR 高 8 位为 0,不是起始位
                MOV     DPL,A                   ;DPTR 清 0
                AJMP    READHEAD                ;重新寻找起始位
;
READDATA00:     CLR     A                       ;处理起始位开始,清 A
                MOV     DPH,A                   ;DPTR 计数器清 0
                MOV     DPL,A                   ;
READLOOP01:     JB      REMOTEIN,READDATA02     ;读起始位,高电平时转 READDA-
                                                 TA02
                INC     DPTR                    ;低电平时对 DPTR 循环计数
                NOP                             ;空操作延时
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
```

```
                NOP
                AJMP    READLOOP01          ;循环,周期约为 21 μs
;
READDATA02:     CLR     LAMP                ;关操作灯
                MOV     R4,DPH              ;存起始位(高 8 位入 R4)
                MOV     R5,DPL              ;存起始位(低 8 位入 R5)
                MOV     DPH,#00H            ;清 0
                MOV     DPL,#READTIME       ;放入校正值(本设计没校正,值为 0)
                AJMP    READLOOP1           ;转 READLOOP1 处理高电平程序
;
READDATA1:      SETB    LAMP                ;存高电平数据程序,开操作灯
                MOV     @R0,DPL             ;存入高电平数据(地址在 08H~
                                            6FH)
                INC     R0                  ;指向下一地址
                MOV     DPL,#READTIME       ;放入校正值(本设计没校正,值为 0)
                MOV     DPH,#00H            ;清 0
READLOOP0:      JB      REMOTEIN,READDATA2  ;读低电平程序,高电平时转
                                            READDATA2
                INC     DPTR                ;低电平时对 DPTR 循环计数
                NOP                         ;空操作延时,循环延时周期为 21 μs
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                AJMP    READLOOP0           ;延时循环控制
;
READDATA2:      CLR     LAMP                ;存低电平数据程序,关操作灯
                MOV     @R1,DPL             ;存低电平数据(地址在 90H~F7H)
                INC     R1                  ;地址加 1
                MOV     DPL,#READTIME       ;计数校正(本设计没校正,值为 0)
                MOV     DPH,#00H            ;清 0
READLOOP1:      JNB     REMOTEIN,READDATA3  ;读高电平程序,为 0 时转 READDA-
                                            TA3
```

```
                INC     DPTR                ;高电平时对 DPTR 循环计数
                NOP                         ;空操作延时,循环周期为 21 μs
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                AJMP    READLOOP1           ;循环控制
;
READDATA3:      CLR     A                   ;
                CJNE    A,DPH,READDATA4     ;DPH 不为 0 转 READDATA4(码读
                                            完)
                AJMP    READDATA1           ;转 READDATA1(存高电平数据)
READDATA4:      MOV     @R0,A               ;放结束标志数据
                MOV     @R1,A               ;放结束标志数据
;
                SETB    LAMP                ;开操作灯
                CLR     STUDYLAMP           ;关学习灯
                SETB    REMOTEIN            ;遥控输入状态
READEND:        JNB     STUDYKEY,READEND    ;等待键释放
                SETB    EX0                 ;开外中断
                SETB    EA                  ;开总中断允许
                RETI                        ;中断返回
;
;*************************************
;*                                   *
;*          键工作子程序             *
;*                                   *
;*************************************
KEYWORK:        SETB    P1.0                ;置 P1.0 口为输入状态
                JNB     P1.0,KEY0           ;为 0 转 KEY0
KEYOUT:         RET                         ;无键按下,返回
;
KEY0:           LCALL   DL10MS              ;延时去抖动
```

```
                JB      P1.0,KEYOUT          ;是干扰转 KEYOUT 返回
                LJMP    REMOTETX             ;有键按下,转 REMOTETX 发射遥
                                              控码
;
;*********************************************
;*                                           *
;*          延时程序(513 μs)                 *
;*                                           *
;*********************************************
DELAY:          MOV     R0,#DELAYCONUT0      ;(#0FFH)
DELAY1:         DJNZ    R0,DELAY1
                RET
;
;
;
;*********************************************
;*                                           *
;*          延时约 25 ms                     *
;*                                           *
;*********************************************
DL10MS:         MOV     R1,#DELAYCONUT       ;(#30H)
DL10MS1:        LCALL   DELAY
                DJNZ    R1,DL10MS1
                RET
;
                END                          ;程序结束
```

第 11 章　实例 6　15 路电器遥控器的设计

用单片机制作的 15 路电器遥控器，可以分别控制 15 个电器的电源开关，并且可对一路电灯进行亮度的遥控。该遥控器采用脉冲个数编码，4×8 键盘开关，可扩充到对 32 个电器的控制。

1. 系统硬件电路的设计

图 11.1 为该系统遥控发射器电原理图，其中 P1 口和 P0 口作键扫描端口，具有 32 个功能操作键；第 9 脚为单片机的复位脚，采用简单的 RC 上电复位电路；15 脚作为红外线遥控码的输出口，用于输出 40 kHz 载波编码；18、19 脚接 12 MHz 晶振。P0 口需接上拉电阻。

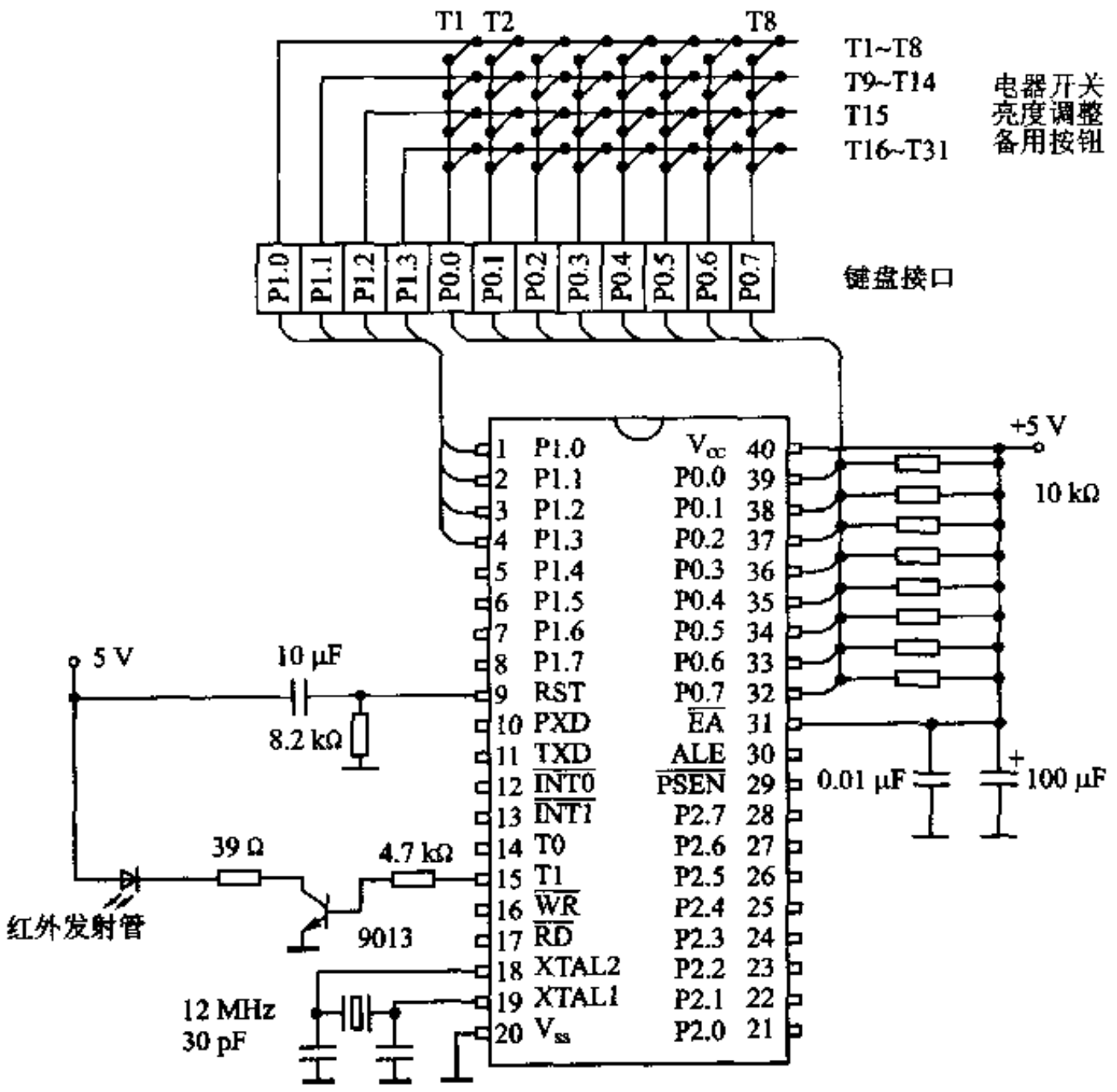


图 11.1　遥控发射器电原理图

图 11.2 为该系统遥控接收器电原理图，其中 P1.1～P1.2 口作为数码管的二进制数据输出，显示数字为 0～7，7 代表最亮，0 代表最暗，采用 4511 集成块硬件译码显示数值；P0.0～P0.7 以及 P2.0～P2.6 口作为 15 个电器的电源控制输出，接口可以用继电器或可控硅，在本电路中，P2.0 口控制一个电灯的亮灭；P2.7 口为可控硅调光灯的调光脉冲输出；第 10 脚 P3.0 口为 50 Hz 交流市电相位基准输入，第 12 脚为中断输入口；P3.1 口用于接收红外遥控码输入

信号。

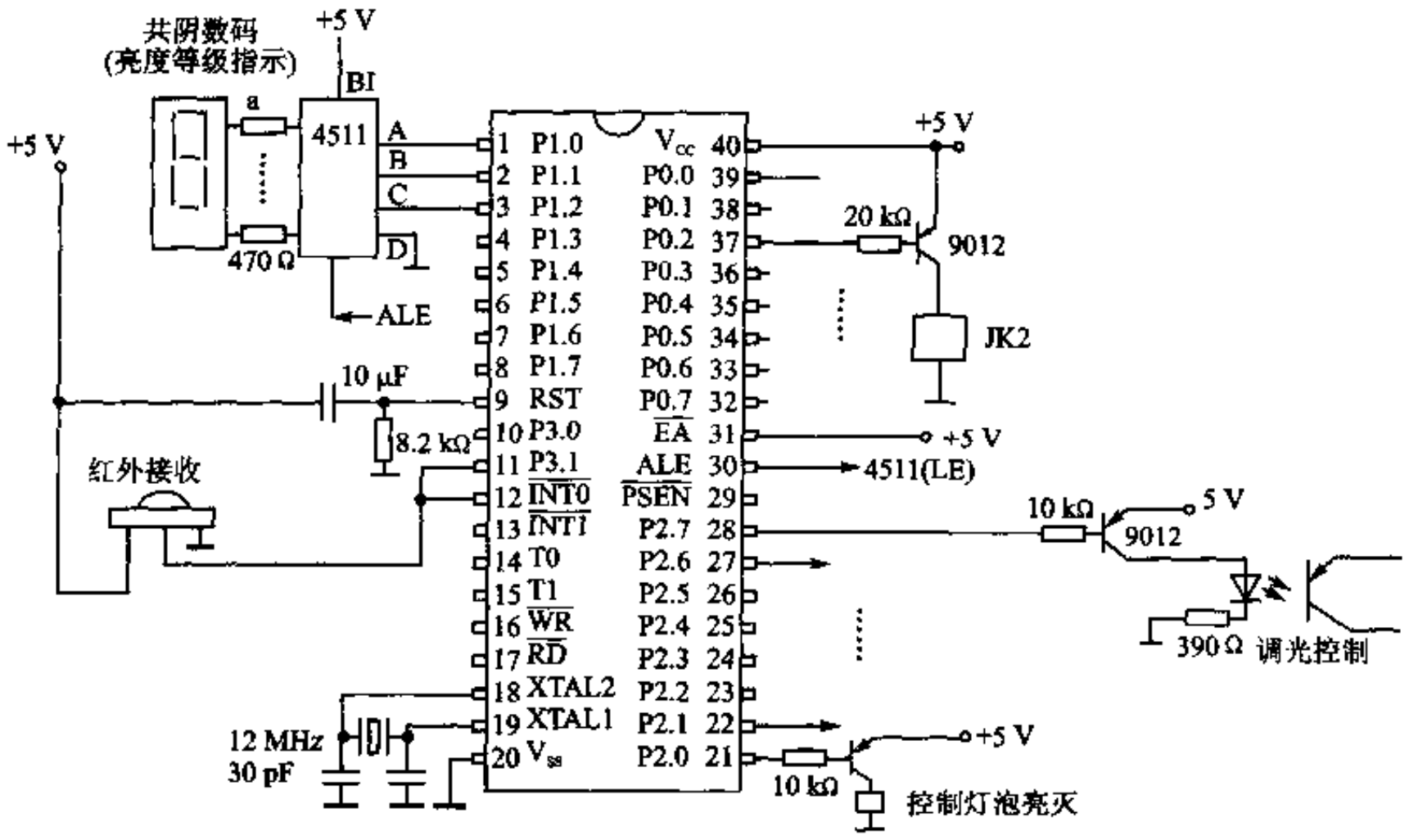


图 11.2 遥控接收器电原理图

2. 系统的功能实现方法

(1) 遥控码的编码格式

该遥控器采用脉冲个数编码,不同的脉冲个数代表不同的码,最小为 2 个脉冲,最大为 17 个脉冲。为了使接收可靠,第一位码宽为 3 ms,其余为 1 ms,遥控码数据帧间隔大于 10 ms,如图 11.3 所示。

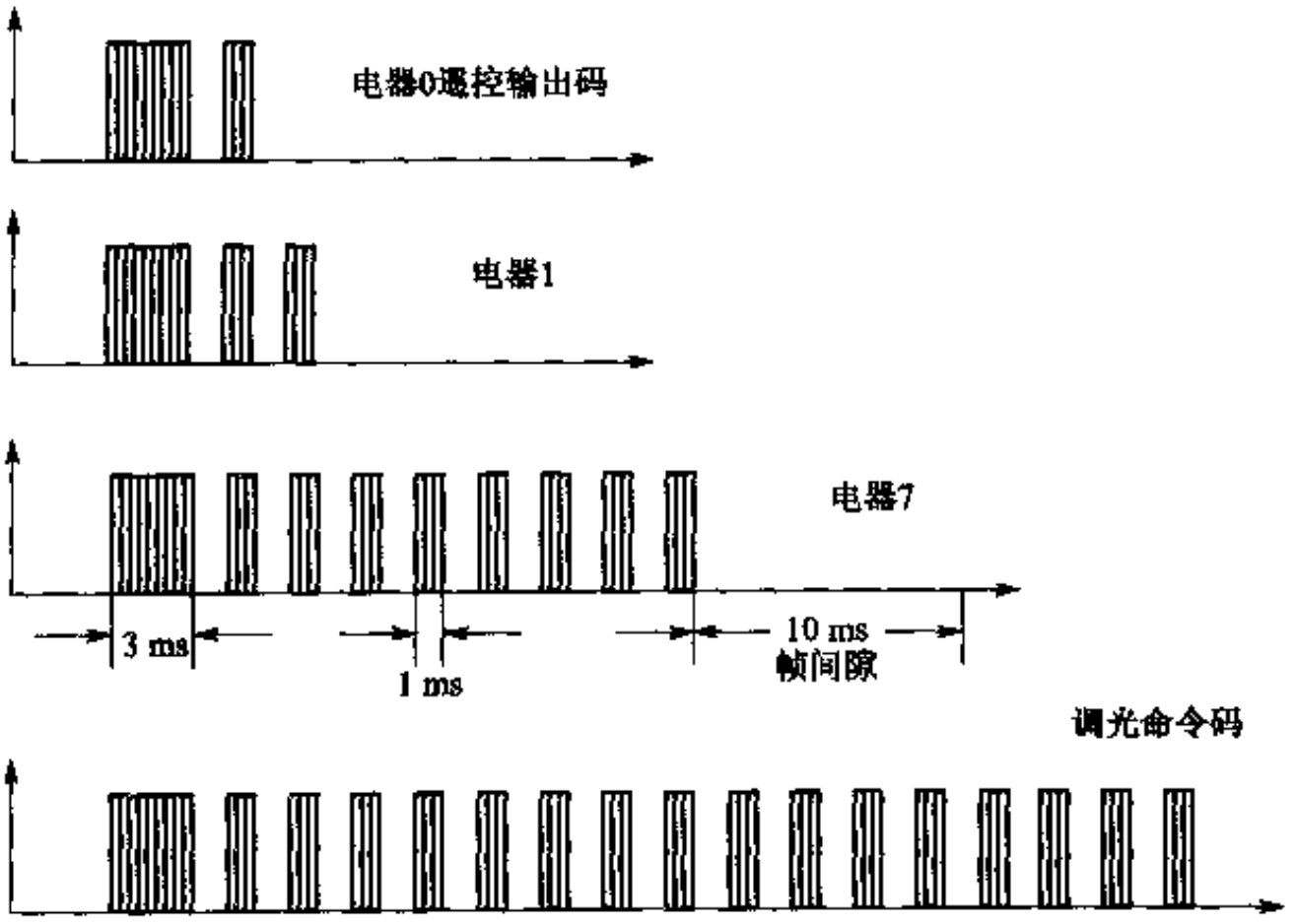


图 11.3 P3.5 端口输出编码波形图

(2) 遥控码的发射

当某个操作按键按下时，单片机先读出键值，然后根据键值设定遥控码的脉冲个数，再调制成 40 kHz 方波由红外线发光管发射出去。P3.5 端口的输出调制波如图 11.3 所示。

(3) 数据帧的接收处理

当红外线接收器输出脉冲帧数据时，第一位码的低电平将启动中断程序，实时接收数据帧。在数据帧接收时，将对第一位(起始位)码的码宽进行验证。若第一位低电平码的脉宽小于 2 ms，将作为错误码处理。当间隔位的高电平脉宽大于 3 ms 时，结束接收，然后根据累加器 A 中的脉冲个数，执行相应输出口的操作。图 11.4 为红外线接收器输出的一帧遥控码波形图。

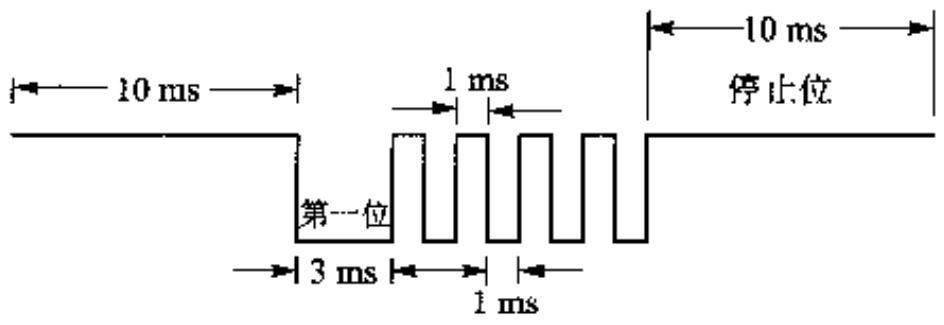


图 11.4　红外线接收器输出的一帧遥控码波形图

3. 遥控发射及接收控制程序流程图

遥控发射及接收控制流程图如图 11.5、图 11.6 所示。

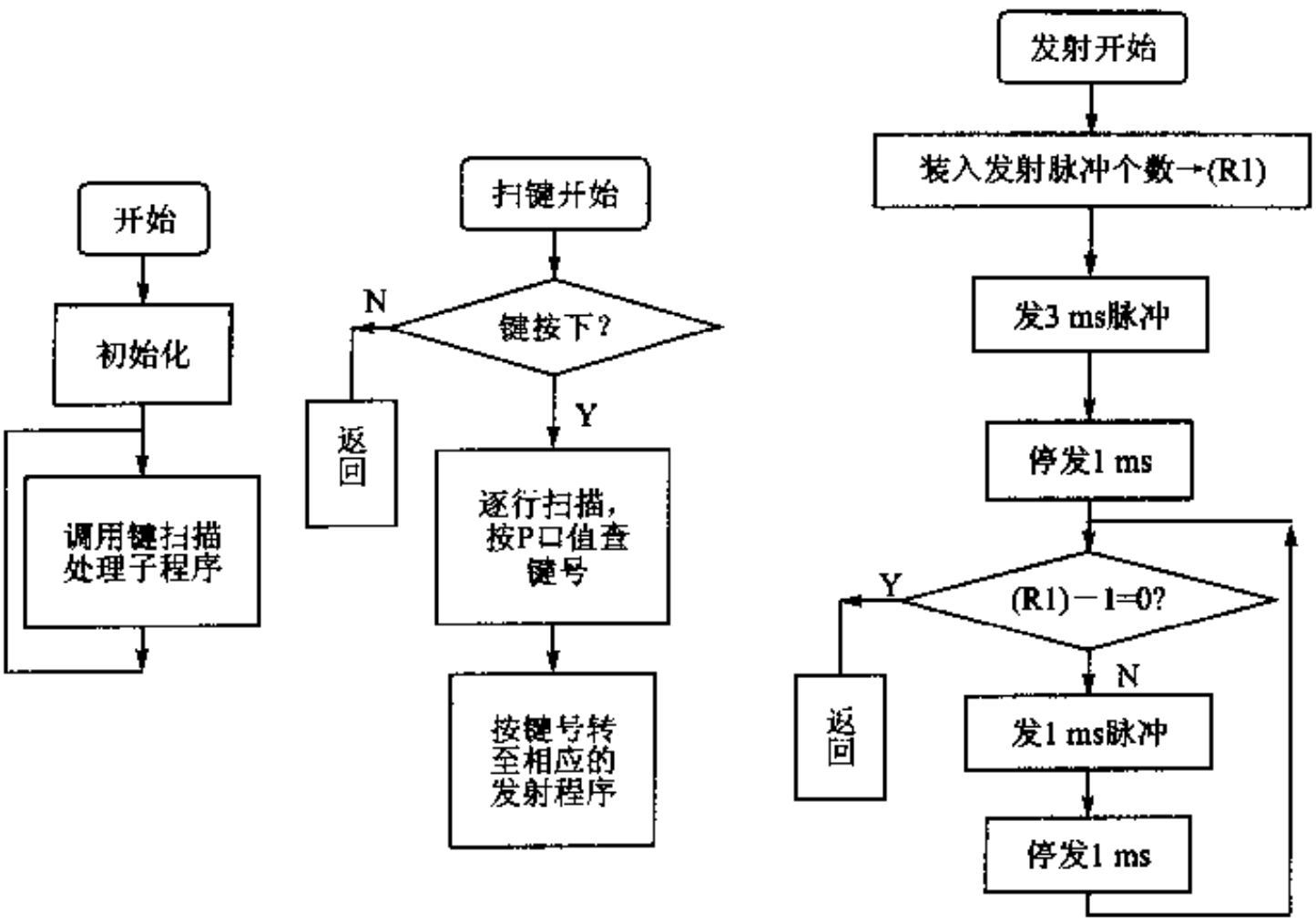


图 11.5　遥控发射器程序流程图

采用红外线遥控方式时，由于受遥控距离、角度等影响，使用效果不是很好，如采用调频或调幅发射接收编码，可提高遥控距离，并且没有角度影响。

以下是 15 路红外发射遥控器及接收器的控制源程序：

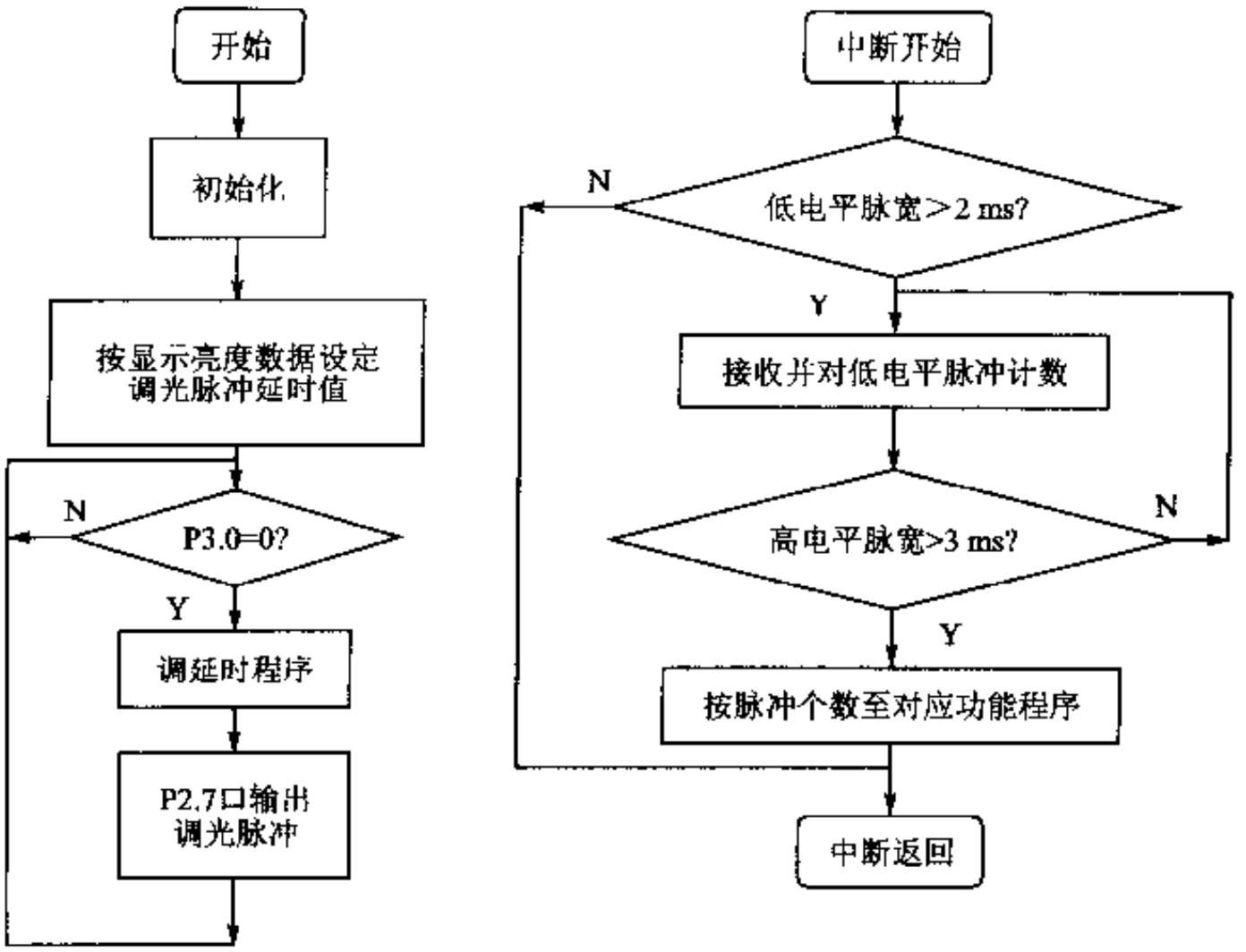


图 11.6 遥控接收器主程序、中断程序流程图

```
;*****************************************
;*                (SEND.ASM)             *
;*          15 路遥控发送控制器          *
;*              2001.7.18                *
;*****************************************
;*****************************************************************************
;*        KEYX0    P1.0 |  1        40 | VCC                    *
;*        KEYX1    P1.1 |  2        39 | P0.0      KEYY0        *
;*        KEYX2    P1.2 |  3        38 | P0.1      KEYY1        *
;*        KEYX3    P1.3 |  4        37 | P0.2      KEYY2        *
;*                 P1.4 |  5        36 | P0.3      KEYY3        *
;*                 P1.5 |  6        35 | P0.4      KEYY4        *
;*                 P1.6 |  7        34 | P0.5      KEYY5        *
;*                 P1.7 |  8        33 | P0.6      KEYY6        *
;*                 RST  |  9        32 | P0.7      KEYY7        *
;*                 P3.0 |  10  51单片机 31 | EA        VDD          *
;*                 P3.1 |  11       30 | ALE                    *
;*                 P3.2 |  12       29 | PSEN                   *
;*                 P3.3 |  13       28 | P2.7                   *
;*                 P3.4 |  14       27 | P2.6                   *
;*  REMOTEOUT TI   P3.5 |  15       26 | P2.5                   *
;*                 P3.6 |  16       25 | P2.4                   *
;*                 P3.7 |  17       24 | P2.3                   *
;*                 XTAL2|  18       23 | P2.2                   *
;*                 XTAL1|  19       22 | P2.1                   *
;*                 VSS  |  20       21 | P2.0                   *
```

```
;*                                                                    *
;**********************************************************************
;
;
;
;伪定义
        KEYX0       EQU     P1.0            ;P1.0～P1.3 位键盘行扫描
        KEYX1       EQU     P1.1            ;本系统采用 4×8 键盘阵列
        KEYX2       EQU     P1.2
        KEYX3       EQU     P1.3
        KEYY        EQU     P0              ;P0 口键盘列扫描
;
;
;
;**********************************************
;*                                            *
;*              主程序和中断程序入口          *
;*                                            *
;**********************************************
        ORG         0000H                   ;程序执行开始地址
                    AJMP    START           ;跳至 START 执行
        ORG         0003H                   ;外中断 0 中断入口地址
                    RETI                    ;中断返回(不开中断)
        ORG         000BH                   ;定时器 T0 中断入口地址
                    RETI                    ;中断返回(不开中断)
        ORG         0013H                   ;外中断 1 中断入口地址
                    RETI                    ;中断返回(不开中断)
        ORG         001BH                   ;定时器 T1 中断入口地址
                    LJMP    INTT1           ;跳至 INTT1 中断服务程序
        ORG         0023H                   ;串行口中断入口地址
                    RETI                    ;中断返回(不开中断)
        ORG         002BH                   ;定时器 T2 中断入口地址
                    RETI                    ;中断返回(不开中断)
;
;**********************************************
;*                                            *
;*              初 始 化 程 序                *
;*                                            *
;**********************************************
CLEARMEMI0:         CLR     A               ;A 清 0
                    DEC     A               ;A 为 #0FFH
```

```
                MOV     P0,A            ;P0～P3 口置 1
                MOV     P1,A            ;
                MOV     P2,A            ;
                MOV     P3,A            ;
                CLR     P3.5            ;关遥控输出
CLEARMEM:       MOV     SP,#70H         ;设堆栈基址为 70H
                MOV     IE,#00H         ;关所有中断
                MOV     IP,#01H         ;设优先级
                MOV     TMOD,#22H       ;8 位自动重装初值模式
                MOV     TH1,#0F3H       ;定时为 13 μs 初值
                MOV     TL1,#0F3H       ;
                SETB    EA              ;开总中断允许
                RET                     ;返回
;
;
;*****************************************
;*                                       *
;*              主 程 序                 *
;*                                       *
;*****************************************
START:          LCALL   CLEARMEMIO      ;调用初始化子程序
;
MAIN:           LCALL   KEYWORK         ;主体程序。调用查键子程序
                LJMP    MAIN            ;转 MAIN 循环
                NOP                     ;PC 值出错处理
                NOP
                NOP
                LJMP    START           ;重新初始化
;
;*****************************************
;*                                       *
;*          T1 中断服务程序              *
;*                                       *
;*****************************************
INTT1:          CPL     P3.5            ;40 kHz 红外线遥控信号产生
                RETI                    ;中断返回
;
;
;*****************************************
;*                                       *
;*      键盘工作子程序(4×8 阵列)         *
```

```
;*                    出口为各键工作程序入口                    *
;**********************************************************
KEYWORK:    MOV   KEYY,#0FFH        ;置列线输入状态
            CLR   KEYX0             ;行线(P1 口)全置 0
            CLR   KEYX1
            CLR   KEYX2
            CLR   KEYX3
            MOV   A,KEYY            ;读入 P0 口值
            MOV   B,A               ;KEYY 口值暂存 B 中
            CJNE  A,#0FFH,KEYHIT    ;不等于#0FFH,转 KEYHIT(有键按下)
KEYOUT:     RET                     ;没有键按下返回
;
;
KEYHIT:     LCALL DL10MS            ;延时去抖动
            MOV   A,KEYY            ;再读入 P0 口值至 A
            CJNE  A,B,KEYOUT        ;A 不等于 B(是干扰),子程序返回
            SETB  KEYX1             ;有键按下,找键号开始,查 0 行
            SETB  KEYX2
            SETB  KEYX3
            MOV   A,KEYY            ;读入 P0 口值
            CJNE  A,#0FFH,KEYVAL0   ;P0 口不等于#0FFH,按下键在第 0 行
            SETB  KEYX0             ;不在第 0 行,开始查 1 行
            CLR   KEYX1
            MOV   A,KEYY            ;读入 P0 口值
            CJNE  A,#0FFH,KEYVAL1   ;P0 口不等于#0FFH,按下键在第 1 行
            SETB  KEYX1             ;不在第 1 行,开始查 2 行
            CLR   KEYX2
            MOV   A,KEYY            ;读入 P0 口值
            CJNE  A,#0FFH,KEYVAL2   ;P0 口不等于#0FFH,按下键在第 2 行
            SETB  KEYX2             ;不在第 2 行,开始查 3 行
            CLR   KEYX3
            MOV   A,KEYY            ;读入 P0 口值
            CJNE  A,#0FFH,KEYVAL3   ;P0 口不等于#0FFH,按下键在第 3 行
            LJMP  KEYOUT            ;不在第 3 行,子程序返回
;
KEYVAL0:    MOV   R2,#00H           ;按下键在第 0 行,R2 赋行号初值 0
            LJMP  KEYVAL4           ;跳到 KEYVAL4
;
KEYVAL1:    MOV   R2,#08H           ;按下键在第 1 行,R2 赋行号初值 8
            LJMP  KEYVAL4           ;跳到 KEYVAL4
;
KEYVAL2:    MOV   R2,#10H           ;按下键在第 2 行,R2 赋行号初值 16
            LJMP  KEYVAL4           ;跳到 KEYVAL4
;
```

```
KEYVAL3:    MOV     R2,#18H              ;按下键在第 3 行,R2 赋行号初值 24
            LJMP    KEYVAL4              ;跳到 KEYVAL4
;
KEYVAL4:    MOV     DPTR,#KEYVALTAB      ;键值翻译成连续数字
            MOV     B,A                  ;P0 口值暂存 B 内
            CLR     A                    ;清 A
            MOV     R0,A                 ;清 R0
KEYVAL5:    MOV     A,R0                 ;查列号开始,R0 数据放入 A
            SUBB    A,#08H               ;A 中数减 8
            JNC     KEYOUT               ;借位 C 为 0,查表出错,返回
            MOV     A,R0                 ;查表次数小于 8,继续查,
            MOVC    A,@A+DPTR            ;查列号表
            INC     R0                   ;R0 加 1
            CJNE    A,B,KEYVAL5          ;查得值和 P0 口值不等,转 KEYVAL5
                                          再查
            DEC     R0                   ;查得值和 P0 口值相等,R0 减 1
            MOV     A,R0                 ;放入 A(R0 中数值即为列号值)
            ADD     A,R2                 ;与行号初值相加成为键号值(0~31)
            MOV     B,A                  ;键号乘 3 处理用于 JMP 散转指令
            RL      A                    ;键号乘 3 处理用于 JMP 散转指令
            ADD     A,B                  ;键号乘 3 处理用于 JMP 散转指令
            MOV     DPTR,#KEYFUNTAB      ;取散转功能程序(表)首址
            JMP     @A+DPTR              ;散转至对应功能程序标号
KEYFUNTAB:  LJMP    KEYFUN00             ;跳到键号 0 对应功能程序标号
            LJMP    KEYFUN01             ;跳到键号 1 对应功能程序标号
            LJMP    KEYFUN02             ;跳到键号 2 对应功能程序标号
            LJMP    KEYFUN03             ;跳到键号 3 对应功能程序标号
            LJMP    KEYFUN04             ;跳到键号 4 对应功能程序标号
            LJMP    KEYFUN05             ;跳到键号 5 对应功能程序标号
            LJMP    KEYFUN06             ;跳到键号 6 对应功能程序标号
            LJMP    KEYFUN07             ;跳到键号 7 对应功能程序标号
            LJMP    KEYFUN08             ;跳到键号 8 对应功能程序标号
            LJMP    KEYFUN09             ;跳到键号 9 对应功能程序标号
            LJMP    KEYFUN10             ;跳到键号 10 对应功能程序标号
            LJMP    KEYFUN11             ;跳到键号 11 对应功能程序标号
            LJMP    KEYFUN12             ;跳到键号 12 对应功能程序标号
            LJMP    KEYFUN13             ;跳到键号 13 对应功能程序标号
            LJMP    KEYFUN14             ;跳到键号 14 对应功能程序标号
            LJMP    KEYFUN15             ;跳到键号 15 对应功能程序标号
            LJMP    KEYFUN16             ;跳到键号 16 对应功能程序标号
            LJMP    KEYFUN17             ;跳到键号 17 对应功能程序标号
            LJMP    KEYFUN18             ;跳到键号 18 对应功能程序标号
            LJMP    KEYFUN19             ;跳到键号 19 对应功能程序标号
```

```
        LJMP    KEYFUN20        ;跳到键号 20 对应功能程序标号
        LJMP    KEYFUN21        ;跳到键号 21 对应功能程序标号
        LJMP    KEYFUN22        ;跳到键号 22 对应功能程序标号
        LJMP    KEYFUN23        ;跳到键号 23 对应功能程序标号
        LJMP    KEYFUN24        ;跳到键号 24 对应功能程序标号
        LJMP    KEYFUN25        ;跳到键号 25 对应功能程序标号
        LJMP    KEYFUN26        ;跳到键号 26 对应功能程序标号
        LJMP    KEYFUN27        ;跳到键号 27 对应功能程序标号
        LJMP    KEYFUN28        ;跳到键号 28 对应功能程序标号
        LJMP    KEYFUN29        ;跳到键号 29 对应功能程序标号
        LJMP    KEYFUN30        ;跳到键号 30 对应功能程序标号
        LJMP    KEYFUN31        ;跳到键号 31 对应功能程序标号
        RET
;列号对应数据表
KEYVALTAB:  DB   0FEH,0FDH,0FBH,0F7H,0EFH,0DFH,0BFH,7FH
;对应列号:          0    1    2    3    4    5    6    7
        RET
;
KEYFUN00:       MOV     A,#02H          ;发 2 个脉冲
                LJMP    REMOTE          ;转发送程序
                RET
;
KEYFUN01:       MOV     A,#03H          ;发 3 个脉冲
                LJMP    REMOTE          ;转发送程序
                RET
;
KEYFUN02:       MOV     A,#04H          ;发 4 个脉冲
                LJMP    REMOTE          ;转发送程序
                RET
;
KEYFUN03:       MOV     A,#05H          ;发 5 个脉冲
                LJMP    REMOTE          ;转发送程序
                RET
;
KEYFUN04:       MOV     A,#06H          ;发 6 个脉冲
                LJMP    REMOTE          ;转发送程序
                RET
;
KEYFUN05:       MOV     A,#07H          ;发 7 个脉冲
                LJMP    REMOTE          ;转发送程序
                RET
;
KEYFUN06:       MOV     A,#08H          ;发 8 个脉冲
```

```
                LJMP    REMOTE          ;转发送程序
                RET
;
KEYFUN07:       MOV     A,#09H          ;发 9 个脉冲
                LJMP    REMOTE          ;转发送程序
                RET
;
;
KEYFUN08:       MOV     A,#0AH          ;发 10 个脉冲
                LJMP    REMOTE          ;转发送程序
                RET
;
KEYFUN09:       MOV     A,#0BH          ;发 11 个脉冲
                LJMP    REMOTE          ;转发送程序
                RET
;
KEYFUN10:       MOV     A,#0CH          ;发 12 个脉冲
                LJMP    REMOTE          ;转发送程序
                RET
;
KEYFUN11:       MOV     A,#0DH          ;发 13 个脉冲
                LJMP    REMOTE          ;转发送程序
                RET
;
KEYFUN12:       MOV     A,#0EH          ;发 14 个脉冲
                LJMP    REMOTE          ;转发送程序
                RET
;
KEYFUN13:       MOV     A,#0FH          ;发 15 个脉冲
                LJMP    REMOTE          ;转发送程序
                RET
;
KEYFUN14:       MOV     A,#10H          ;发 16 个脉冲
                LJMP    REMOTE          ;转发送程序
                RET
;
KEYFUN15:       MOV     A,#11H          ;发 17 个脉冲
                LJMP    REMOTE          ;转发送程序
                RET
KEYFUN16:       RET                     ;备用功能
KEYFUN17:       RET                     ;备用功能
KEYFUN18:       RET                     ;备用功能
KEYFUN19:       RET                     ;备用功能
```

```
KEYFUN20:       RET                         ;备用功能
KEYFUN21:       RET                         ;备用功能
KEYFUN22:       RET
KEYFUN23:       RET
KEYFUN24:       RET
KEYFUN25:       RET
KEYFUN26:       RET
KEYFUN27:       RET
KEYFUN28:       RET
KEYFUN29:       RET
KEYFUN30:       RET
KEYFUN31:       RET                         ;备用功能
;
;
;*************************************************
;*                                               *
;*                 编 码 发 射 程 序             *
;*                                               *
;*************************************************
;
REMOTE:         MOV     R1,A                ;装入发射脉冲个数
                LJMP    OUT3                ;转第一个码发射处理
OUT:            MOV     R0,#55H             ;1 ms 宽低电平发射控制数据
OUT1:           SETB    ET1                 ;开 T1 中断
                SETB    TR1                 ;开启定时器 T1
                NOP                         ;延时
                NOP
                NOP
                NOP
                NOP
                NOP
                DJNZ    R0,OUT1             ;时间不到转 OUT1 再循环
                MOV     R0,#32H             ;1 ms 高电平间隙控制数据
OUT2:           CLR     TR1                 ;关定时器 T1
                CLR     ET1                 ;关 T1 中断
                CLR     P3.5                ;关脉冲输出
                NOP                         ;空操作延时
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
```

```
                NOP
                NOP
                NOP
                DJNZ    R0,OUT2          ;时间不到转 OUT2 再循环
                DJNZ    R1,OUT           ;脉冲未发完,转 OUT 再循环发射
                LCALL   DL500MS          ;
                RET                      ;
OUT3:           MOV     R0,#0FFH         ;装发射 3 ms 宽控制数据
                LJMP    OUT1             ;转 OUT1
;
;*************************************************
;*                                               *
;*                  延时 513 μs                  *
;*                                               *
;*************************************************
;
;513 μs 延时程序
DELAY:          MOV     R2,#0FFH
DELAY1:         DJNZ    R2,DELAY1
                RET
;
;
;
;*************************************************
;*                                               *
;*                  延时 10 ms                   *
;*                                               *
;*************************************************
;
;10 ms 延时程序
DL10MS:         MOV     R3,#14H
DL10MS1:        LCALL   DELAY
                DJNZ    R3,DL10MS1
                RET
;500 ms 延时程序
DL500MS:        MOV     R4,#32H
DL500MS1:       LCALL   DL10MS
                DJNZ    R4,DL500MS1
                RET
;
                END                      ;程序结束
```

```
;            ************************************
;            *          (INCEPT3. ASM)          *
;            *          15 路遥控接收板          *
;            *          2001. 7. 18             *
;            *                                  *
;*************************************************
;
;*                                                              *
;*      A ←— P1.0   | 1               40 | Vcc                  *
;*      B ←— P1.1   | 2               39 | P0.0  LED0           *
;*      C ←— P1.2   | 3               38 | P0.1  LED1           *
;*           P1.3   | 4               37 | P0.2  LED2           *
;*           P1.4   | 5               36 | P0.3  LED3           *
;*           P1.5   | 6               35 | P0.4  LED4           *
;*           P1.6   | 7               34 | P0.5  LED5           *
;*           P1.7   | 8               33 | P0.6  LED6           *
;*   100 Hz  RST    | 9               32 | P0.7  LED7           *
;*  ——————→  P3.0   | 10    51单片机   31 | EA    VDD            *
;*       ┌→  P3.1   | 11              30 | ALE                  *
;* REMOTEIN→ P3.2   | 12              29 | PSEN                 *
;*           P3.3   | 13              28 | P2.7  ———→ 调光脉冲   *
;*           P3.4   | 14              27 | P2.6  LED8           *
;*           P3.5   | 15              26 | P2.5  LED9           *
;*           P3.6   | 16              25 | P2.4  LED10          *
;*           P3.7   | 17              24 | P2.3  LED11          *
;*           XTAL2  | 18              23 | P2.2  LED12          *
;*           XTAL1  | 19              22 | P2.1  LED13          *
;*           Vss    | 20              21 | P2.0  LEV14(灯泡)     *
;*                                                              *
;*                                                              *
;*                                                              *
;****************************************************************
;注:P3.0 为 100 Hz 的交流电源过零点相位参考输入
;
;
;**************************************
;*                                    *
;*          主程序和中断程序入口          *
;*                                    *
;**************************************
        ORG     0000H                   ;程序开始地址
                LJMP    START           ;跳至 START 执行
        ORG     0003H                   ;外中断 0 中断入口
                LJMP    INTEX0          ;跳至 INTEX0 执行中断服务程序
        ORG     000BH                   ;定时器 T0 中断入口地址
                RETI                    ;中断返回(不开中断)
        ORG     0013H                   ;外中断 1 中断入口地址
```

```
                RETI            ;中断返回(不开中断)
        ORG     001BH           ;定时器 T1 中断入口地址
                RETI            ;中断返回(不开中断)
        ORG     0023H           ;串行口中断入口地址
                RETI            ;中断返回(不开中断)
        ORG     002BH           ;定时器 T2 中断入口地址
                RETI            ;中断返回(不开中断)
;
;*************************************
;*                                   *
;*          初始化程序               *
;*                                   *
;*************************************
CLEARMEMIO:     CLR     A               ;
                DEC     A               ;A 为#0FFH
                MOV     P0,A            ;P1～P3 口置 1
                MOV     P1,A            ;
                MOV     P2,A            ;
                MOV     P3,A            ;
CLEARMEM:       MOV     IE,#00H         ;关所有中断
                SETB    EX0             ;开外中断
                SETB    EA              ;总中断允许
                RET                     ;子程序返回
;
;*************************************
;*                                   *
;*          主 程 序                 *
;*                                   *
;*************************************
START:          LCALL   CLEARMEMIO      ;上电初始化
                LCALL   LOOP            ;调用调光控制程序
;
MAIN:           JB      P3.0,MAIN       ;50 Hz 交流电未过零转 MAIN
                LCALL   DLX             ;过零点时调用延时子程序(延时可变)
                CLR     P2.7            ;发调光脉冲
                LCALL   DELAY           ;持续 512 μs
                SETB    P2.7            ;关调光脉冲
                LJMP    MAIN            ;转 MAIN 循环
                NOP                     ;PC 值出错处理
                NOP
                LJMP    START           ;出错时重新初始化
;*************************************
;*          遥控接收程序             *
```

```
;**********************************
;采用中断接收
INTEX0:     CLR     EX0             ;关外中断
            JNB     P3.1,READ1      ;P3.1 口为低电平转 READ1
READOUTT0:  SETB    EX0             ;P3.1 口为高电平开中断(系干扰)
            RETI                    ;退出中断
;
READ1:      CLR     A               ;清 A
            MOV     DPH,A           ;清 DPTR
            MOV     DPL,A           ;
HARD1:      JB      P3.1,HARD11     ;P3.1 变高电平转 HARD11
            INC     DPTR            ;用 DPTR 对低电平计数
            NOP                     ;1 μs 延时
            NOP
            AJMP    HARD1           ;转 HARD1 循环(循环周期为 8 μs)
HARD11:     MOV     A,DPH           ;DPTR 高 8 位放入 A
            JZ      READOUTT0       ;为 0(脉宽小于 8 μs×255==2 ms)退出
            CLR     A               ;不为 0,说明是第一个宽脉冲(3 ms)
READ11:     INC     A               ;脉冲个数计 1
READ12:     JNB     P3.1,READ12     ;低电平时等待
            MOV     R1,#06H         ;高电平宽度判断定时值
READ13:     JNB     P3.1,READ11     ;变低电平时转 READ11 脉冲计数
            LCALL   DELAY           ;延时(512 μs)
            DJNZ    R1,READ13       ;6 次延时不到转 READ13 再延时
            DEC     A               ;超过 3 ms 判为结束,减 1
            DEC     A               ;减 1
            JZ      FUN0            ;为 0 执行 FUN0(2 个脉冲)
            DEC     A               ;减 1
            JZ      FUN1            ;为 0 执行 FUN1(3 个脉冲)
            DEC     A               ;
            JZ      FUN2            ;为 0 执行 FUN2(4 个脉冲)
            DEC     A               ;
            JZ      FUN3            ;为 0 执行 FUN3(5 个脉冲)
            DEC     A               ;
            JZ      FUN4            ;为 0 执行 FUN4(6 个脉冲)
            DEC     A               ;
            JZ      FUN5            ;为 0 执行 FUN5(7 个脉冲)
            DEC     A               ;
            JZ      FUN6            ;为 0 执行 FUN6(8 个脉冲)
            DEC     A               ;
            JZ      FUN7            ;为 0 执行 FUN7(9 个脉冲)
            DEC     A               ;
            JZ      FUN8            ;为 0 执行 FUN8(10 个脉冲)
```

```
        DEC     A                ;
        JZ      FUN9             ;为 0 执行 FUN9(11 个脉冲)
        DEC     A                ;
        JZ      FUN10            ;为 0 执行 FUN10(12 个脉冲)
        DEC     A                ;
        JZ      FUN11            ;为 0 执行 FUN11(13 个脉冲)
        DEC     A                ;
        JZ      FUN12            ;为 0 执行 FUN12(14 个脉冲)
        DEC     A                ;
        JZ      FUN13            ;为 0 执行 FUN13(15 个脉冲)
        DEC     A                ;
        JZ      FUN14            ;为 0 执行 FUN14(16 个脉冲)
        DEC     A                ;
        JZ      FUN15            ;为 0 执行 FUN15(17 个脉冲)
        NOP                      ;
        NOP                      ;
        LJMP    READOUTT0        ;出错退出
;
FUN0:   CPL     P0.0             ;P0 口各端口开关输出控制
        LJMP    READOUTT0        ;转中断退出
FUN1:   CPL     P0.1             ;
        LJMP    READOUTT0        ;
FUN2:   CPL     P0.2
        LJMP    READOUTT0
FUN3:   CPL     P0.3
        LJMP    READOUTT0
FUN4:   CPL     P0.4
        LJMP    READOUTT0
FUN5:   CPL     P0.5
        LJMP    READOUTT0
FUN6:   CPL     P0.6
        LJMP    READOUTT0
FUN7:   CPL     P0.7
        LJMP    READOUTT0
FUN8:   CPL     P2.6             ;P2 口各端口开关输出控制
        LJMP    READOUTT0        ;转中断退出
FUN9:   CPL     P2.5
        LJMP    READOUTT0
FUN10:  CPL     P2.4
        LJMP    READOUTT0
FUN11:  CPL     P2.3
        LJMP    READOUTT0
FUN12:  CPL     P2.2
```

```
                LJMP    READOUTT0
FUN13:          CPL     P2.1
                LJMP    READOUTT0
FUN14:          CPL     P2.0            ;P2.0 口开关控制
                LJMP    READOUTT0       ;转中断退出
FUN15:          DEC     P1              ;P1 口值减 1
                MOV     A,P1            ;移入 A
                CJNE    A,#0F7H,OUTT0   ;不等转 OUTT0(显示值小于 7)
                CLR     A               ;相等清 A
                DEC     A               ;A 为#0FFH
                MOV     P1,A            ;放回 P1(显示值为 7)
OUTT0:          LCALL   LOOP            ;亮度调整
                LJMP    READOUTT0       ;中断退出
;
;*************************************
;*          延时程序(513 μs)          *
;*************************************
;
DELAY:          MOV     R0,#0FFH
DELAY1:         DJNZ    R0,DELAY1
                RET
;
;*************************************
;*            延时 10 ms             *
;*************************************
;
DL10MS:         MOV     R1,#14H
DL10MS1:        LCALL   DELAY
                DJNZ    R1,DL10MS1
                RET
;
;*************************************
;*          调光延时时间控制          *
;*************************************
;
DLX:            MOV     R2,B            ;置延时初值
DLX1:           LCALL   DELAY           ;调 512 μs 延时子程序
                DJNZ    R2,DLX1         ;循环控制
                RET                     ;返回
;
;*************************************
;*          调光控制程序              *
;*************************************
```

```
;根据数码管指示值设置调光脉冲延时值
LOOP:   MOV   A,P1          ;读入 P1 口值
        SUBB  A,#0FFH       ;比较
        JZ    LOOP7         ;值为#0FFH(显示 7)时转 LOOP7
        MOV   A,P1          ;
        SUBB  A,#0FEH       ;
        JZ    LOOP6         ;值为#0FEH(显示 6)时转 LOOP6
        MOV   A,P1          ;
        SUBB  A,#0FDH       ;
        JZ    LOOP5         ;值为#0FDH(显示 5)时转 LOOP5
        MOV   A,P1          ;
        SUBB  A,#0FCH       ;
        JZ    LOOP4         ;值为#0FCH(显示 4)时转 LOOP4
        MOV   A,P1          ;
        SUBB  A,#0FBH       ;
        JZ    LOOP3         ;值为#0FBH(显示 3)时转 LOOP3
        MOV   A,P1          ;
        SUBB  A,#0FAH       ;
        JZ    LOOP2         ;值为#0FAH(显示 2)时转 LOOP2
        MOV   A,P1          ;
        SUBB  A,#0F9H       ;
        JZ    LOOP1         ;值为#0F9H(显示 1)时转 LOOP1
        MOV   A,P1          ;
        SUBB  A,#0F8H       ;
        JZ    LOOP0         ;值为#0F8H(显示 0)时转 LOOP0
        RET                 ;返回
;
LOOP7:  MOV   B,#01H        ;设置延时值#01H(最亮)
        RET                 ;返回
LOOP6:  MOV   B,#02H        ;设置延时值#02H(次亮)
        RET                 ;返回
LOOP5:  MOV   B,#04H        ;
        RET
LOOP4:  MOV   B,#06H        ;
        RET
LOOP3:  MOV   B,#08H        ;
        RET
LOOP2:  MOV   B,#0AH        ;
        RET
LOOP1:  MOV   B,#0CH        ;设置延时值#0CH(次暗)
        RET                 ;返回
LOOP0:  MOV   B,#0DH        ;设置延时值#0DH(最暗)
        RET                 ;返回
;
        END                 ;程序结束
```

第 12 章　实例 7　自行车里程/速度计的设计

自行车里程/速度计能自动显示自行车行驶的总里程数及行车速度，具有超速信响提醒功能，里程数据自动记忆，也可应用于电动自行车、摩托车、汽车等机动车仪表上。

1. 系统硬件电路的设计

自行车里程/速度计采用 AT89C52 单片机作控制，速度及里程传感器采用霍尔元件，其电气原理如图 12.1 所示。P0 口和 P2 口用于七段 LED 显示器的段码及扫描输出，在显示里程时，第三位小数点用 17 脚 P3.7 口控制点亮。P1.0 口和 P1.1 口分别用于显示里程状态和速度状态。P1.2、P1.3、P1.6 和 P1.7 口分别用于设置轮圈的大小。P3.0 口的开关用于确定显示的方式，当开关闭合时，显示速度；打开时显示里程。第 12 脚外中断 0 用于对轮子圈数的计数输入，轮子每转一圈，霍尔传感器输出一个低电平脉冲。13 脚外中断 1 用于控制定时器 T1 的启停，当输入为 0 时关闭定时器。此控制信号是将轮子圈数的计数脉冲经二分频后形成(见图 12.2)，这样，每次定时器 T1 的开启时间刚好为转 1 圈的时间，根据轮子的周长就可以计算出自行车的速度。P1.4 口和 P1.5 口用于 EEPROM 存储器 24C01 的存取控制。11 脚输出用于速度超速时的报警。

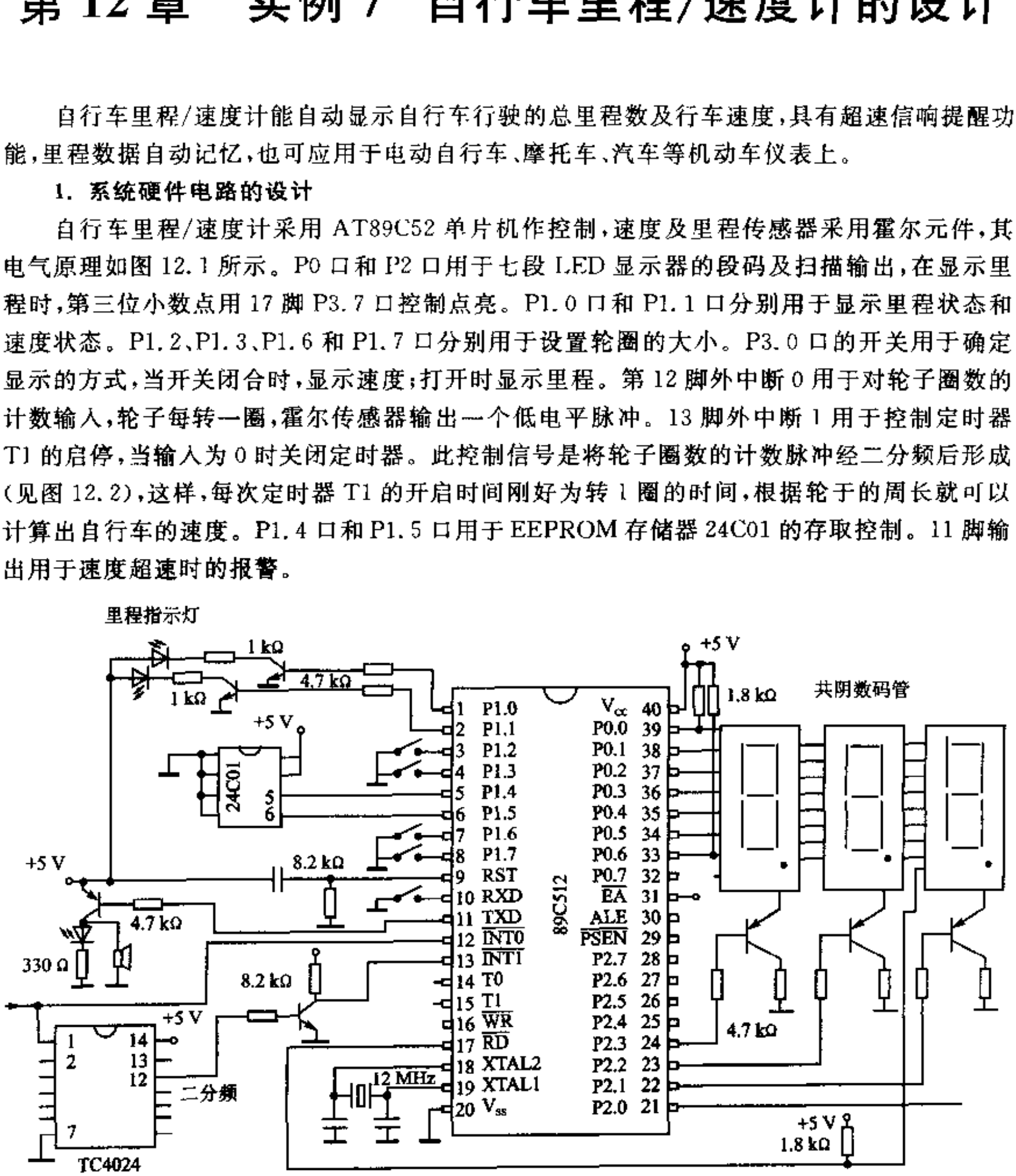


图 12.1　自行车里程/速度计电气原理图

2. 系统内存的规划

由于本系统处理功能较多，因而一部分内存单元用于特定的用处。其主要内存单元及用处如下：

50H：EEPROM 器件寻址字节存放单元；

51H：EEPROM 传送字节数存放单元；

30H：EEPROM 发送数据缓冲单元；

40H：EEPROM 读出数据存放单元；

0A0H：EEPROM 寻址字节写；

0A1H：EEPROM 寻址字节读；

62H：DPTR 计数扩展高 8 位；

6CH：定时器 T1 计数扩展高 8 位；

6DH：定时器 T1 计数扩展高 8～16 位；

60H、61H、62H：里程计数单元；

68H、69H、6AH、6BH：存放自行车每圈时间数；

70H、71H、72H、73H：显示 BCD 码数据存放用；

11H～15H：存放被除数；

16H～19H：存放除数。

3. 系统主要程序的设计

(1) 初始化程序

在本系统初始化程序中，主要完成以下工作：将 T1 设为外部控制定时器方式；外中断 0 及外中断 1 设为边沿触发方式；将部分内存单元清零；设置轮子周长值；开中断及定时器；将 EEPROM 中的数据调入内存等。

(2) 轮圈设置出错处理程序

P1.2、P1.3、P1.6、P1.7 端口的开关用于设定轮子的周长，当没有设定时（至少让一个开关闭合），能从 P3.1 口输出一个周期为 0.5 s 的方波信号，用作发光管闪烁及信响器提醒。

(3) 主程序

主程序根据 P3.0 口的开关状态选择里程显示或速度显示，其流程图如图 12.3 所示。

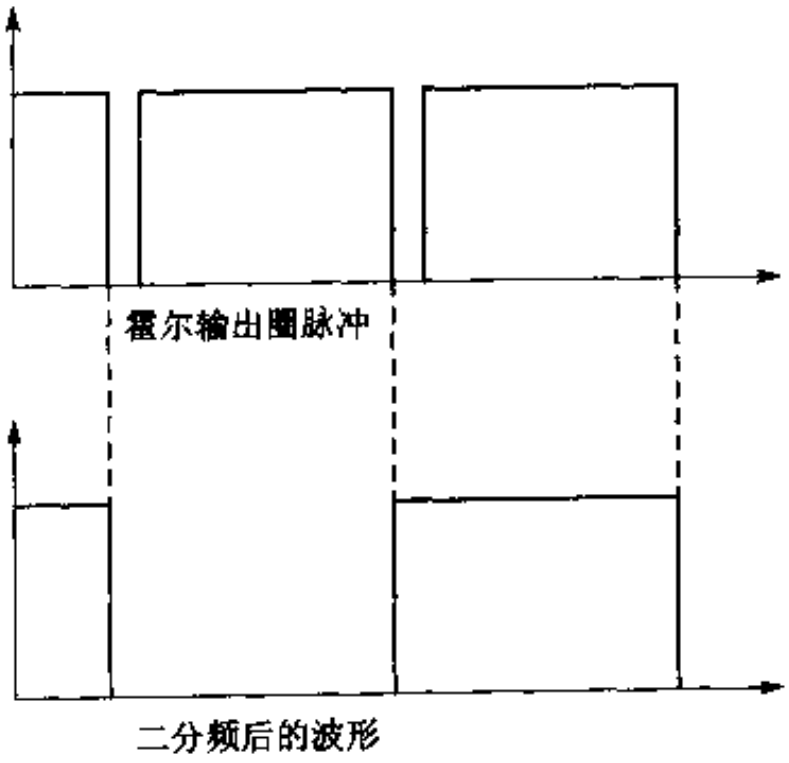


图 12.2　单片机 11、12 脚的输入波形

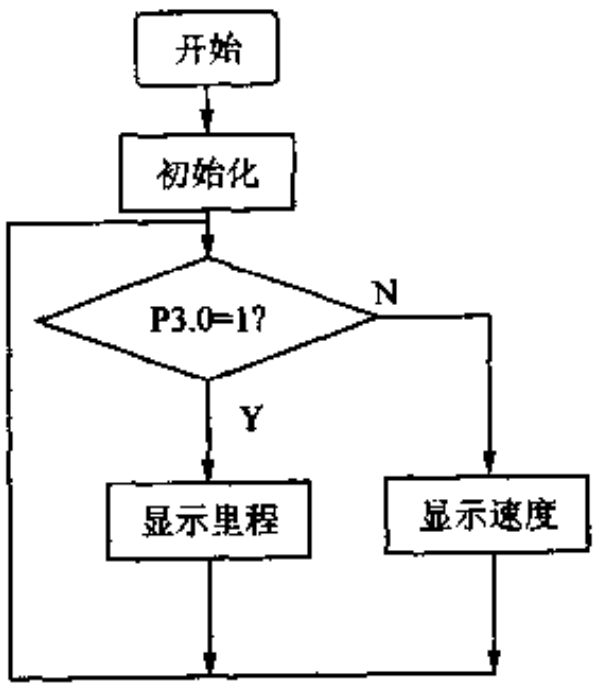


图 12.3　主程序流程图

(4)里程计数程序(外中断 0 服务程序)

外中断 0 服务程序用于对 12 脚输入的圈脉冲进行计数，为十六进制计数器。60H 为低位，62H 为高位。每计数一次后，对里程数据进行一次存储操作。

(5)外中断 1 服务程序

外中断 1 服务程序用于处理轮子转动一圈后的计时数据。当标志位(00H)为 1 时，说明计数器溢出，放入最大时间值(为 # 0FFH)；当标志位为 0 时，将计数单元(TL1、TH1、6CH、6DH)的值放入 68H～6BH 单元。

(6)EEPROM 存取程序

本系统使用归一化 I²C 串口存取子程序，使用一条数据线和时钟线，采用 ATMEL 公司的 24C01 串口存储器，应用简单方便。

(7)显示子程序

当显示里程时，先要将圈数计数器中的数据进行运算，求出总里程。当要显示速度时，要将轮子的周长和转一圈的时间数相除，然后换算成 km/h 单位。最后放入 70H～73H，进行数据的显示。

以下是控制系统完整源程序：

```
                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                    ;;     SPEED/MILE FOR BIKE    PROGRAM      ;;
                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;        60H,61H,62H 作里程计数单元,6CH,6DH 作 T1 计数扩充单元,
;        68H,69H,6AH,6BH 存放自行车每圈时间数,70H,71H,72H,73H
;        作显示 BCD 码存放数用,11H～15H 存放被除数,16H～19H 存放除数
;
;
;定义
            VSDA      EQU     P1.5        ; EEPROM 数据传送口
            VSCL      EQU     P1.4        ; EEPROM 时钟传送口
            SLA       EQU     50H         ; EEPROM 器件寻址字节存放单元
            NUMBYT    EQU     51H         ; EEPROM 传送字节数存放单元
            MTD       EQU     30H         ; EEPROM 发送数据缓冲单元
            MRD       EQU     40H         ; EEPROM 读出数据存放单元
            SLAW      EQU     0A0H        ; EEPROM 寻址字节写
            SLAR      EQU     0A1H        ; EEPROM 寻址字节读
            DPHH      EQU     62H         ; DPTR 计数扩展高 8 位
            TH1H      EQU     6CH         ;定时器 T1 扩展高 8 位
            TH1HH     EQU     6DH         ;定时器 T1 扩展高 8～16 位
;
;;;;;;;;;;;;;;;;;;;;
;; PROGRAM INPUT ;;
;;;;;;;;;;;;;;;;;;;;
```

```
;
        ORG     0000H           ;程序执行开始地址
        LJMP    START           ;跳至 START
        ORG     0003H           ;外中断 0 中断程序入口
        LJMP    INTEX0          ;跳至 INTEX0 中断服务程序
        ORG     000BH           ;定时器 T0 中断程序入口
        RETI                    ;中断返回
        ORG     0013H           ;外中断 1 中断入口
        LJMP    INTEX1          ;跳至 INTEX1 中断服务程序
        ORG     001BH           ;定时器 T1 中断程序入口
        LJMP    INTT1           ;跳至 INTT1 中断服务程序
        ORG     0023H           ;串口中断入口地址
        RETI                    ;中断返回
        ORG     002BH           ;定时器 T2 中断入口地址
        RETI                    ;中断返回
;
;
;;;;;;;;;;;;;;;;;;;;
;; PROGRAM CLEAR ;;
;;;;;;;;;;;;;;;;;;;;
;上电初始化程序
CLEARMEN:   MOV     TMOD,#90H   ;T1 为 16 位外部控制定时器
            MOV     SP,#75H     ;堆栈在 75H 开始
            SETB    PX0         ;外中断 0 优先级为 1
            SETB    IT0         ;外中断 0 用边沿触发
            SETB    IT1         ;外中断 1 用边沿触发
            CLR     A           ;清 A
            MOV     20H,A       ;清内存中特定单元
            MOV     6CH,A       ;
            MOV     6DH,A       ;
            MOV     70H,A       ;
            MOV     71H,A       ;
            MOV     72H,A       ;
            MOV     73H,A       ;
            MOV     60H,A       ;
            MOV     61H,A       ;
            MOV     62H,A       ;
            MOV     63H,A       ;清内存中特定单元
            DEC     A           ;A 为#0FFH
            MOV     68H,A       ;内存置数据#0FFH
            MOV     69H,A       ;内存置数据#0FFH
            MOV     6AH,A       ;内存置数据#0FFH
            MOV     6BH,A       ;内存置数据#0FFH
```

```
        MOV     P1, A           ;P1 口置 1
CLEAR1: JB      P1.2,KEY1       ;根据 P1.2,P1.3,P1.6,P1.7 设置状态,
                                ;在 21H 地址单元赋自行车周长值
        MOV     21H,#0FH        ;22 英寸自行车周长值
        LJMP    CLEAR2          ;转 CLEAR2
KEY1:   JB      P1.3,KEY2       ;
        MOV     21H,#12H        ;24 英寸自行车周长值
        LJMP    CLEAR2          ;转 CLEAR2
KEY2:   JB      P1.6,KEY3       ;
        MOV     21H,#14H        ;26 英寸自行车周长值
        LJMP    CLEAR2          ;转 CLEAR2
KEY3:   JB      P1.7,ERR        ;4 个开关都没合上,转出错处理
        MOV     21H,#19H        ;28 英寸自行车周长值
CLEAR2: SETB    TR1             ;开定时器 T1
        SETB    EA              ;开中断允许
        SETB    EX0             ;开外中断 0
        SETB    ET1             ;开定时中断 T1
        SETB    P3.1            ;关报警器
        LCALL   V1ICREAD        ;将 EEPROM 中原里程数据调入内存
        RET                     ;子程序返回
ERR:    CPL     P3.1            ;轮周长设置出错,LED 灯闪烁提醒
        LCALL   DL5S            ;延时
        LJMP    CLEAR1          ;重新初始化,等待轮周长设置开关合上
;
;;;;;;;;;;;;;;;;;;;;
;; PROGRAM START ;;
;;;;;;;;;;;;;;;;;;;;
;
START:  LCALL   CLEARMEN        ;上电初始化
START1: JB      P3.0,DISPLAYS   ;P3.0=1,显示里程
        LCALL   DISPLAYV        ;显示速度
START2: SJMP    START1          ;转 START1 循环
;
;;;;;;;;;;;;;;;;;;;;
;;INTEX0 PROGRAM ;;
;;;;;;;;;;;;;;;;;;;;
;里程计数程序,用外中断 0 实现,计数用 60H～62H 内存单元。
INTEX0: PUSH    ACC             ;累加器堆栈保护
        PUSH    PSW             ;状态字堆栈保护
        INC     60H             ;圈加 1
        CLR     A               ;清 A
        CJNE    A,60H,INTEX0OUT ;计数没溢出转 INTEX0OUT
        INC     61H             ;溢出进位(61H 加 1)
```

```
                CJNE    A,61H,INTEX0OUT     ;计数没溢出转 INTEX0OUT
                INC     62H                 ;溢出进位(62H 加 1)
INTEX0OUT:      LCALL   VIICWRITE           ;里程数据存入 EEPROM
                SETB    EX1                 ;开外中断 1
                POP     PSW                 ;状态字恢复
                POP     ACC                 ;累加器恢复
                RETI                        ;中断返回
;;;;;;;;;;;;;;;;;;;;
;;INTEX1 PROGRAM ;;
;;;;;;;;;;;;;;;;;;;;
;每转 1 圈时间计数处理程序,每圈时间放在 68H~6BH 单元中。
INTEX1:         PUSH    ACC                 ;堆栈保护
                PUSH    PSW                 ;
                CLR     EX1                 ;关外中断 1
                JNB     00H,INTEX11         ;溢出标志为 0 转 INTEX11
                MOV     TL1,#0FFH           ;溢出时计时单元赋#0FFH(显示速度为零)
                MOV     TH1,#0FFH           ;
                MOV     6CH,#0FFH           ;
                MOV     6DH,#0FFH           ;
INTEX11:        MOV     68H,TL1             ;将时间计数值移入暂存单元 68H~6BH
                MOV     69H,TH1             ;
                MOV     6AH,6CH             ;
                MOV     6BH,6DH             ;
                CLR     A                   ;清 A
                MOV     TL1,A               ;计时单元置 0
                MOV     TH1,A               ;
                MOV     6CH,A               ;
                MOV     6DH,A               ;
                CLR     00H                 ;清溢出标志
                POP     PSW                 ;堆栈恢复
                POP     ACC                 ;
                RETI                        ;中断返回
;
;;;;;;;;;;;;;;;;;;;;
;;INTT1  PROGRAM ;;
;;;;;;;;;;;;;;;;;;;;
;
; T1 计数器中断服务程序。(计数器 T1 由外中断 1 输入控制,当为高电平时计时开始)
INTT1:          PUSH    ACC                 ;堆栈保护
                PUSH    PSW                 ;
                INC     6CH                 ;6CH 计时单元加 1
                MOV     A,6CH               ;移入 A
                JNZ     INTT11              ;不等于 0 转 INTT11
```

```
                INC     6DH             ;进位,6DH 单元加 1
                MOV     A,6DH           ;移入 A
                JNZ     INTT11          ;不等于 0 转 INTT11
                SETB    00H             ;计时器溢出,置溢出标志
INTT11;         POP     PSW             ;恢复堆栈
                POP     ACC             ;
                RETI                    ;中断返回
;
;;;;;;;;;;;;;;;;;;;
;;   DISPLAY S   ;;
;;;;;;;;;;;;;;;;;;;
;
;里程显示控制程序
DISPLAYS:       SETB    P1.0            ;点亮 LED1(显示里程状态)
                CLR     P1.1            ;关闭速度指示灯
                SETB    P3.7            ;显示小数点(最小显示为 0.1 km)
                LCALL   SSS             ;将圈数转为公里数
                LCALL   DISPLAY         ;显示公里数据
                LJMP    START1          ;跳回 START1
;
;
;;;;;;;;;;;;;;;;;;;
;;   DISPLAY V   ;;
;;;;;;;;;;;;;;;;;;;
;速度显示控制程序
DISPLAYV:       CLR     P1.0            ;关闭 LED1(里程)灯
                SETB    P1.1            ;点亮 LED2(显示时速状态)
                CLR     P3.7            ;关小数点显示
                LCALL   VVV             ;每圈时间换算为 km/h 程序
                MOV     A,71H           ;将十位数(BCD 码)值移入 A
                SUBB    A,#04H          ;与预定报警值比较
                JNC     WARING          ;时速超过 40 时报警
                SETB    P3.1            ;关报警灯
V1:             LCALL   DISPLAY         ;显亮一次(为了改善闪烁)
                RET                     ;子程序返回
WARING:         CLR     P3.1            ;报警灯 LED3 点亮(并鸣叫)
                AJMP    V1              ;转 V1 退出
;
;;;;;;;;;;;;;;;;;;;
;;   VIICWRITE   ;;
;;;;;;;;;;;;;;;;;;;
;归一化 EEPROM 存入程序(12 MHz 时钟),存入数在 50H 起单元
VIICWRITE:      ACALL   WMOV9
```

```
        MOV     SLA,#SLAW
        MOV     NUMBYT,#09H
        LCALL   WRNBYT
        RET
WMOV9:  MOV     5FH,#50H
        MOV     R0,#MTD
        MOV     R1,#5FH
        MOV     R2,#09H
WMOV:   MOV     A,@R1
        MOV     @R0,A
        INC     R0
        INC     R1
        DJNZ    R2,WMOV
        RET
;
;;;;;;;;;;;;;;;;;;;;
;;VIICREAD      ;;
;;;;;;;;;;;;;;;;;;;;
;归一化 EEPROM 读出程序(12 MHz 时钟),读出数放入 60H~67H 单元
VIICREAD:   MOV     MTD,#50H        ;
            MOV     SLA,#SLAW
            MOV     NUMBYT,#01H
            LCALL   WRNBYT
            MOV     SLA,#SLAR
            MOV     NUMBYT,#08H
            LCALL   RDNBYT
            ACALL   RMOV8
            RET
RMOV8:      MOV     R0,#MRD
            MOV     R1,#60H
            MOV     R2,#08H
RMOV:       MOV     A,@R0
            MOV     @R1,A
            INC     R0
            INC     R1
            DJNZ    R2,RMOV
            RET
;
;;;;;;;;;;;;;;;;;;;;
;;VIIC PROGRAM ;;
;;;;;;;;;;;;;;;;;;;;
; I²C 串行归一化存储子程序
STA:        SETB    VSDA
```

```
        SETB    VSCL
        NOP
        NOP
        NOP
        NOP
        CLR     VSDA
        NOP
        CLR     VSDA
        NOP
        NOP
        NOP
        NOP
        CLR     VSCL
        RET
STOP:   CLR     VSDA
        SETB    VSCL
        NOP
        NOP
        NOP
        NOP
        SETB    VSDA
        NOP
        NOP
        NOP
        NOP
        CLR     VSDA
        CLR     VSCL
        RET
MACK:   CLR     VSDA
        SETB    VSCL
        NOP
        NOP
        NOP
        NOP
        CLR     VSCL
        SETB    VSDA
        RET
MNACK:  SETB    VSDA
        SETB    VSCL
        NOP
        NOP
        NOP
        NOP
```

```
        CLR     VSCL
        CLR     VSDA
        RET
CACK:   SETB    VSDA
        SETB    VSCL
        CLR     F0
        MOV     C,VSDA
        JNC     CEND
        SETB    F0
CEND:   CLR     VSCL
        RET
WRBYT:  MOV     R0,#08H
WLP:    RLC     A
        JC      WR1
        AJMP    WR0
WLP1:   DJNZ    R0,WLP
        RET
WR1:    SETB    VSDA
        SETB    VSCL
        NOP
        NOP
        NOP
        NOP
        CLR     VSCL
        CLR     VSDA
        AJMP    WLP1
WR0:    CLR     VSDA
        SETB    VSCL
        NOP
        NOP
        NOP
        NOP
        CLR     VSCL
        AJMP    WLP1
RDBYT:  MOV     R0,#08H
RLP:    SETB    VSDA
        SETB    VSCL
        MOV     C,VSDA
        MOV     A,R2
        RLC     A
        MOV     R2,A
        CLR     VSCL
        DJNZ    R0,RLP
```

```
                RET
WRNBYT:         MOV     R3,NUMBYT
                LCALL   STA
                MOV     A,SLA
                LCALL   WRBYT
                LCALL   CACK
                JB      F0,WRNBYT
                MOV     R1,#MTD
WRDA:           MOV     A,@R1
                LCALL   WRBYT
                LCALL   CACK
                JB      F0,WRNBYT
                INC     R1
                DJNZ    R3,WRDA
                LCALL   STOP
                RET
RDNBYT:         MOV     R3,NUMBYT
                LCALL   STA
                MOV     A,SLA
                LCALL   WRBYT
                LCALL   CACK
                JB      F0,RDNBYT
RDN:            MOV     R1,#MRD
RDN1:           LCALL   RDBYT
                MOV     @R1,A
                DJNZ    R3,ACK
                LCALL   MNACK
                LCALL   STOP
                RET
ACK:            LCALL   MACK
                INC     R1
                SJMP    RDN1
;
;;;;;;;;;;;;;;;;;;;;
;;DISPLAY PROGRAM ;;
;;;;;;;;;;;;;;;;;;;;
;显示程序。显示 BCD 码在 70H～73H 单元内,采用共阴 LED 数码管
DISPLAY:        MOV     R1,#70H         ;显示单元首址
                MOV     R2,#0FEH        ;扫描字
PLAY:           MOV     A,R2            ;扫描字入 A
                MOV     P2,A            ;放到端口
                MOV     A,@R1           ;取显示数据
                ANL     A,#0FH          ;去掉高 4 位(为 0)
```

```
                MOV     DPTR,#TAB       ;取段码表首址
                MOVC    A,@A+DPTR       ;查显示数据对应段码
                MOV     P0,A            ;段码输出
                LCALL   DL1MS           ;点亮 1 ms
                INC     R1              ;指向下一显示数据地址
                MOV     A,R2            ;取扫描字
                JNB     ACC.3,ENDOUT    ;已扫描到第 4 位,转 ENDOUT 退出
                RL      A               ;循环左移
                MOV     R2,A            ;放回 R2
                AJMP    PLAY            ;转 PLAY 循环
ENDOUT:         SETB    P2.0            ;关扫描
                SETB    P2.1            ;关扫描
                SETB    P2.2            ;关扫描
                SETB    P2.3            ;关扫描
                RET                     ;扫描结束
;共阴段码表(可显示 0~F)
TAB:            DB      3FH,06H,5BH,4FH,66H,6DH,7DH,07H
                DB      7FH,6FH,77H,7CH,39H,5EH,79H,71H
                DB      00H
;
;;;;;;;;;;;;;;;;;;;;
;;SSS PROGRAM    ;;
;;;;;;;;;;;;;;;;;;;;
;里程处理程序,将自行车圈数据换算成公里数
SSS:            MOV     19H,#64H        ;除数最高位赋值
                MOV     18H,#00H        ;除数赋值
                MOV     17H,#00H        ;除数赋值
                MOV     16H,#00H        ;除数赋值
                MOV     11H,#00H        ;被除数赋值
                MOV     12H,#00H        ;被除数赋值
                MOV     13H,62H         ;被除数赋值
                MOV     14H,61H         ;被除数赋值
                MOV     15H,60H         ;被除数赋值
                LCALL   DIVST           ;调除法程序
                LCALL   BCDST           ;调二进制转 BCD 码程序
                MOV     A,25H           ;结果处理,将 25H 数移入 A
                ANL     A,#0FH          ;将高 4 位置为 0
                MOV     70H,A           ;放入 70H 单元
                MOV     A,25H           ;25H 数移入 A
                SWAP    A               ;高低 4 位交换
                ANL     A,#0FH          ;将高 4 位置为 0
                MOV     71H,A           ;放入 71H 单元
                MOV     A,24H           ;24H 数移入 A
```

```
        ANL     A,#0FH          ;将高 4 位置为 0
        MOV     72H,A           ;放入 72H 单元
        MOV     A,24H           ;24H 数移入 A
        SWAP    A               ;高低 4 位交换
        ANL     A,#0FH          ;将高 4 位置为 0
        MOV     73H,A           ;放入 73H 单元
        RET                     ;子程序返回
;
;
;;;;;;;;;;;;;;;;;;;;
;;VVV  PROGRAM  ;;
;;;;;;;;;;;;;;;;;;;;
;时速处理程序,最大显示速度为 99 km/h(用作自行车)
VVV:    MOV     18H,68H         ;除数赋值
        MOV     17H,69H         ;除数赋值
        MOV     16H,6AH         ;除数赋值
        MOV     11H,#00H        ;被除数赋值
        MOV     12H,#00H        ;被除数赋值
        MOV     13H,#36H        ;被除数赋值
        MOV     14H,#0EEH       ;被除数赋值
        MOV     15H,#80H        ;被除数赋值
        LCALL   DIVST           ;调除法程序
        MOV     14H,#00H        ;舍去一位
        LCALL   BCDST           ;二进制转 BCD 码程序
        MOV     A,25H           ;以下将速度值放入显示单元
        ANL     A,#0FH          ;高 4 位为 0
        MOV     70H,A           ;放入 70H 内
        MOV     A,25H           ;再取数
        SWAP    A               ;高低 4 位交换
        ANL     A,#0FH          ;高 4 位为 0
        MOV     71H,A           ;放入 71H 内
        MOV     72H,#00H        ;72H 为 0(高 2 位 LED 显示 0)
        MOV     73H,#00H        ;73H 为 0
        RET                     ;子程序返回
;
;
;;;;;;;;;;;;;;;;;
;    DL1MS    ;;
;;;;;;;;;;;;;;;;;
;1 ms 延时程序,LED 点亮用
DL1MS:  MOV     R6,#14H
DL1:    MOV     R7,#19H
DL2:    DJNZ    R7,DL2
```

```
                DJNZ    R6,DL1
                RET
;
;;;;;;;;;;;;;;;;;
;    DL0.5 s    ;;
;;;;;;;;;;;;;;;;;
;出错闪烁用延时(255 ms)
DL5S:           MOV     R5,#0FFH
DL3:            LCALL   DL1MS
                DJNZ    R5,DL3
                RET
;;;;;;;;;;;;;;;;;
;   除法子程序   ;;
;;;;;;;;;;;;;;;;;
;除法子程序,用作四位除法.除数在 16H~19H.被除数在 11H~15H
DIVST:          CLR     C                    ;运算开始
                MOV     A,13H
                SUBB    A,18H
                MOV     A,12H
                SUBB    A,17H
                MOV     A,11H
                SUBB    A,16H
                JNC     LOOP4
                MOV     B,#10H
NDIV1:          CLR     C
                MOV     A,15H
                RLC     A
                MOV     15H,A
                MOV     A,14H
                RLC     A
                MOV     14H,A
                MOV     A,13H
                RLC     A
                MOV     13H,A
                MOV     A,12H
                RLC     A
                MOV     12H,A
                MOV     A,11H
                RLC     A
                MOV     11H,A
                MOV     F0,C
                CLR     C
                MOV     A,13H
```

```
                SUBB    A,18H
                MOV     1AH,A
                MOV     A,12H
                SUBB    A,17H
                MOV     19H,A
                MOV     A,11H
                SUBB    A,16H
                JB      F0,NDIV2
                JC      NDIV3
NDIV2:          MOV     11H,A
                MOV     A,19H
                MOV     12H,A
                MOV     A,1AH
                MOV     13H,A
                INC     15H
NDIV3:          DJNZ    B,NDIV1
                CLR     F0
DIVEND:         RET
LOOP4:          SETB    F0
                SJMP    DIVEND
;
;;;;;;;;;;;;;;;;;;;;
;  BCD 码转换程序   ;
;;;;;;;;;;;;;;;;;;;;
;将 14H、15H 单元内数据转换成十进制 BCD 码放在 24H、25H 单元内。
BCDST:          MOV     R7,#10H
                CLR     C
                MOV     25H,#00H
                MOV     24H,#00H
KKK:            MOV     A,15H
                RLC     A
                MOV     15H,A
                MOV     A,14H
                RLC     A
                MOV     14H,A
                MOV     A,25H
                ADDC    A,25H
                DA      A
                MOV     25H,A
                MOV     A,24H
                ADDC    A,24H
                DA      A
                MOV     24H,A
```

```
        DJNZ    R7,KKK
        RET
;
        END                         ;程序结束
```

第 13 章　实例 8　自动往返行驶小汽车的设计

本设计的小汽车能在如图 13.1 所示的跑道上自动往返行驶。车子从起跑线出发后到达终点线停车 10 s，然后返回到起点停止。在限速区行驶时间要求大于 8 s，终点线停车与最后停车时要求车子中心点与黑线的误差尽量小。车子能自动记录时间及里程并在车上显示。跑道宽为 0.5 m，两侧挡板高度大于 0.2 m，跑道表面贴有白纸，在 B、C、D、E、F 和 G 处画有 2 cm 宽的黑线。

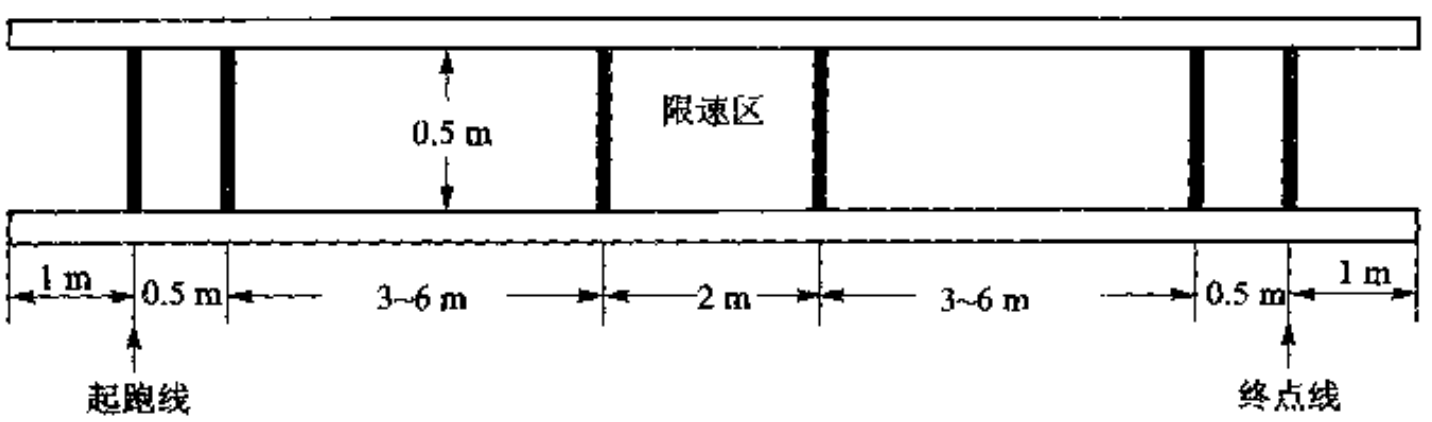


图 13.1　跑道顶视图

1. 系统硬件电路的设计

自动往返行驶小汽车的控制系统采用 AT89C52 单片机；显示系统采用 3 位 LED 数码管显示里程数，4 位 LED 数码管显示一次往返的时间；电机正反转采用桥式驱动控制，2 挡电压调速；里程记录采用霍尔传感器；跑道标志线采用光敏管检测并使用软件整形消抖措施；采用 4 个靠轮解决小汽车与挡板的碰擦问题；单片机、电机采用独立稳压电源供电。

(1) 电机驱动电路

本系统的电机驱动电路采用两对互补三极管，利用单片机 16、17 脚电位的高低去控制三极管的截止和导通状态，从而实现小汽车驱动电机的正反转功能。为了防止电机转动时对单片机的干扰影响，提高单片机的稳定性，本电路在电机的两端加了抗干扰电容，其电路如图 13.2 所示。

(2) 电机调速电路

电机驱动电压由 AT89C52 单片机的 P1.7 和 P1.6 分别控制。当 P1.7 为 0，P1.6 为 1 时，电机驱动电压为 +7.5 V，小车进入高速行驶状态；当 P1.7 为 1，P1.6 为 0 时，电机驱动电压为 +4.3 V，小车进入低速行驶状态。当 P1.0 为高电位时，电机供电三极管 D880 截止，关闭电机电源实现停车功能；当 P1.0 为 0 时，D880 输出电机驱动电压，小车按单片机的指令执行各种功能。电机调速控制电路如图 13.3 所示。

(3) 传感脉冲检测电路

检测电路由霍尔元件里程检测、跑道标志光电管检测两部分组成，其电路图如图 13.4、13.5 所示。

用于里程累计的脉冲信号由霍尔元件检测。霍尔元件安装在后左轮，车轮每转一周就由

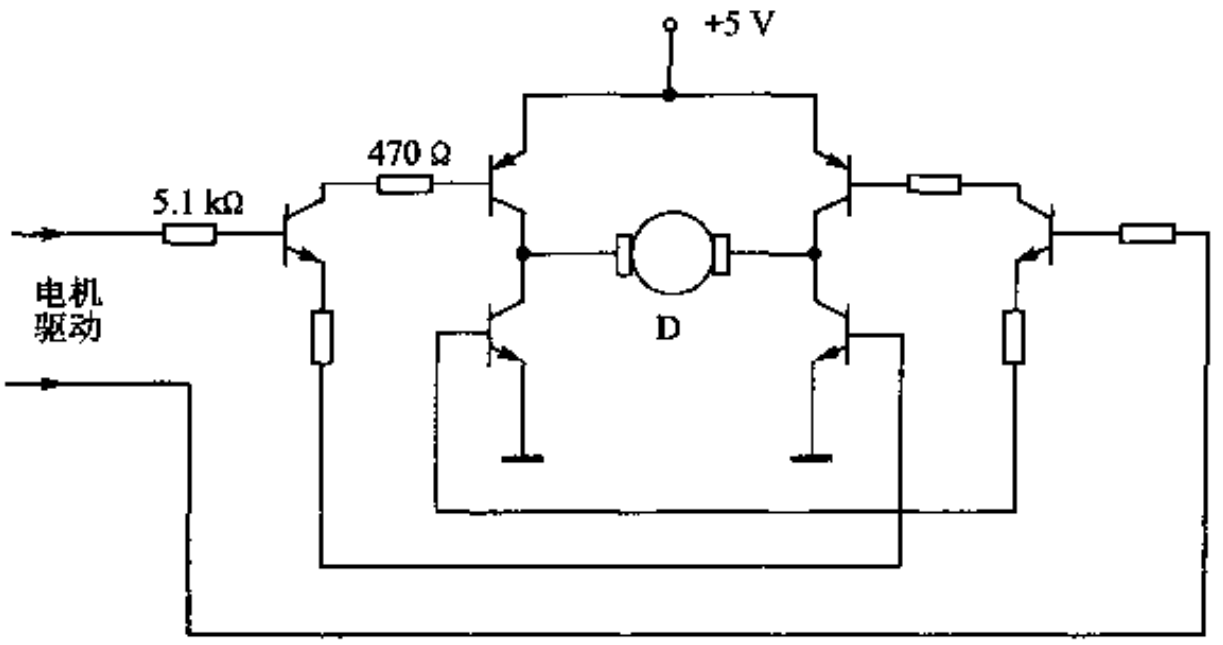


图 13.2　电机驱动电路

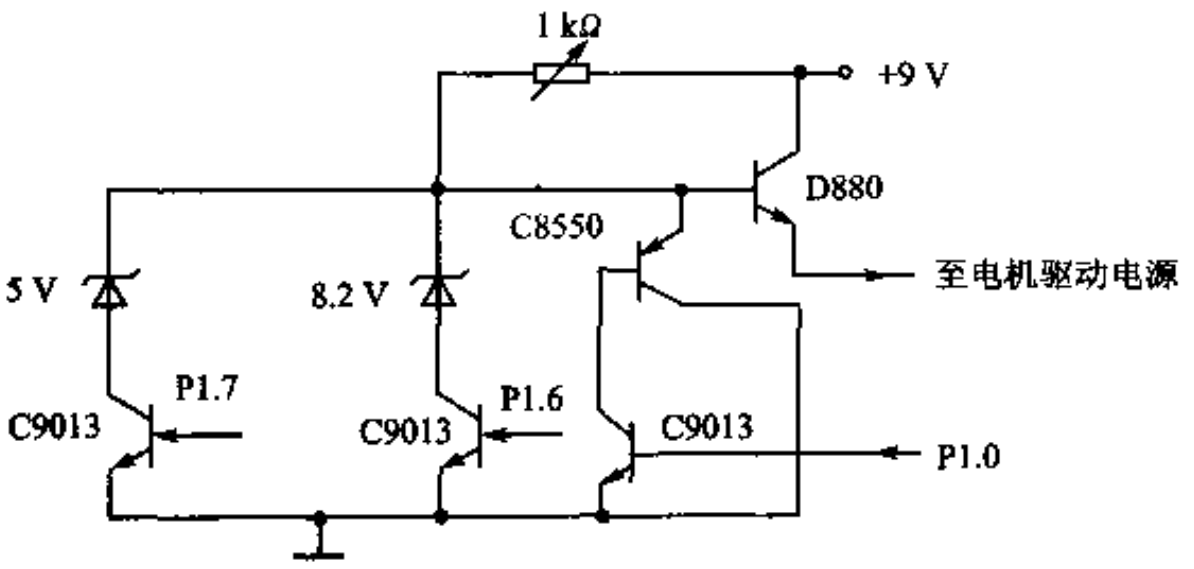


图 13.3　电机调速控制电路

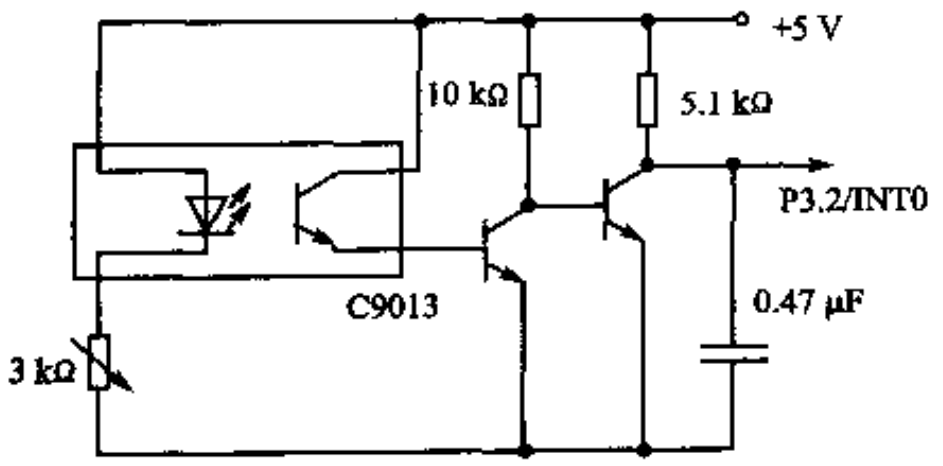


图 13.4　跑道标志检测电路

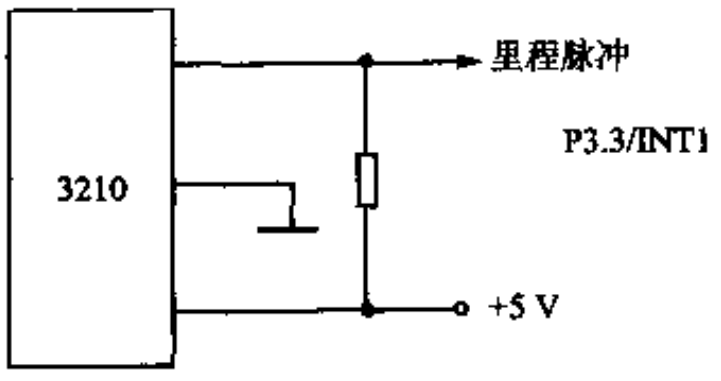


图 13.5　圈脉冲检测电路

其产生一个低电平脉冲，使单片机外部中断 1 产生中断，从而使里程脉冲数累计一次。根据本小车轮子的周长，每转 6 周为 1 m，所以每累计 6 个脉冲就是 1 m。

用于检测跑道标志的脉冲信号由光敏二极管、发光二极管电路组成。当小车在白纸上时，输出为高电平；当遇到黑条时，输出低电平脉冲，作中断计数判断用。

(4) LED 显示电路

对多位 LED 显示器采用动态扫描的方法进行显示。系统采用七位数码管显示，前三位数码管显示的数值表示里程，显示的范围为 1～999 m；后四位数码管的数值表示一次往返过程中所需的总时间，其中前两位表示分，后两位表示秒。往返到起点时显示的数值是一次往返的总时间。P0 口作段码数据输出，P2 口为扫描输出口。

2. 系统内存资源的分配

20H～24H 内存单元作为里程计数用(23H 为跑道条数存放单元)，采用十进制计数，最大计数值为 999 m。70H～73H 为时间计数单元，采用十进制 BCD 码计数，最大记录时间为 59 min59 s。显示数据在 70H～76H 单元中，其中 74H～76H 单元内为里程显示数据。为了标志是终点停车还是起点停车，用位地址 30H(即 35H.0 位)的位值作为判断标志。

3. 系统主要程序的设计

(1) 初始化程序

主要完成 70H～76H、20H～23H 等单元的清零，设置 T1 为 16 位定时模式，开放 T1、外中断 0、外中断 1 的中断等。

(2)主程序

主要完成初始化工作，设定小汽车的初始运行状态，最后循环调用显示程序，其程序流程图如图 13.6 所示。

(3) 外中断 0 服务程序

其任务是根据小车到达黑线的位置控制小车的运行状态，其程序流程图如图 13.7 所示。

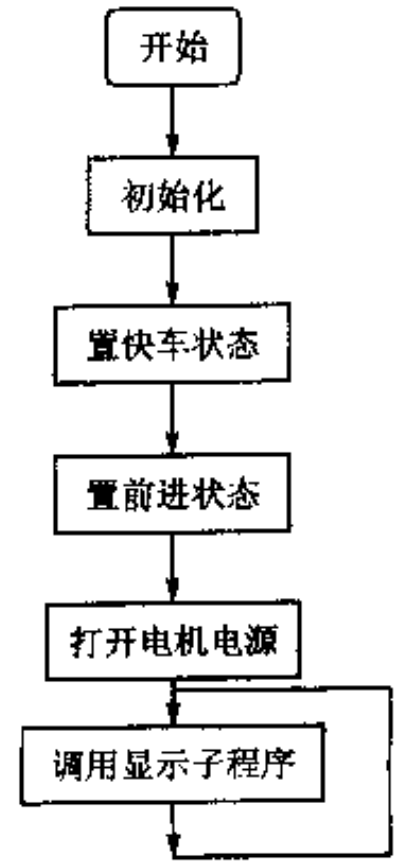


图 13.6　主程序流程图

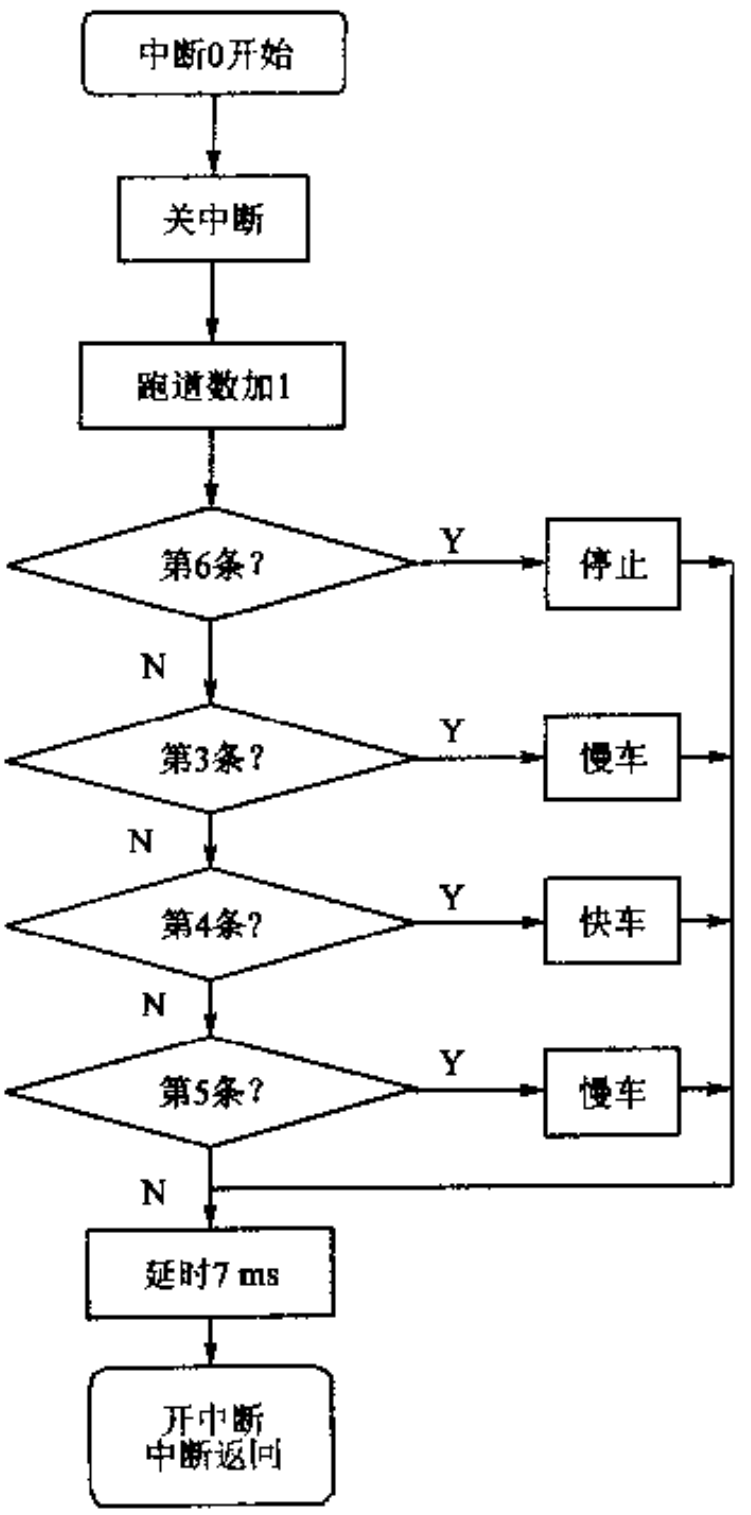


图 13.7　小车状态控制流程图

(4) 慢车子程序

慢车子程序执行时,先对电机进行反相驱动约 150 ms,使其刹车,然后改供低电压,使电机慢速转动。

(5) 停止子程序

当执行停车时,先对黑线道计数器单元 23H 清零,再反相驱动刹车后关电机电源,然后延时 10 s,对 30H 位取反,以判断是中途停车还是起点停车。若是起点停车,计时单元清零。最后,设小车为初始运动状态。

(6) 里程计数处理程序(外中断 1 服务程序)

里程计数器采用十进制计数,最大计数值为 999 m。当退出时,将计数值移入 74H~76H 显示数据存储单元。

(7) 计时程序

与本书第 7 章实例 2 的时钟程序相同。

(8) 延时程序

延时程序均采用调用显示子程序的方法,以改善 LED 显示的效果。

以下是自动往返小汽车控制系统完整源程序:

```
;**************************************
;*                                    *
;*          自 动 往 返 小 汽 车      *
;*              2001 年 9 月          *
;*                                    *
;**************************************
;
;
;
;**************************************
;*                                    *
;*          主程序和中断程序入口      *
;*                                    *
;**************************************
;
        ORG     0000H           ;程序执行起始地址
        LJMP    START           ;跳至 START
        ORG     0003H           ;外中断 0 入口
        LJMP    INTEX0          ;跳至 INTEX0 中断服务程序
        ORG     000BH           ;定时器 T0 中断入口
        RETI                    ;中断返回
        ORG     0013H           ;外中断 1 入口
        LJMP    INTEX1          ;跳至 INTEX1 中断服务程序
        ORG     001BH           ;定时器 T1 中断入口
        LJMP    INTT1           ;跳至 INTT1 中断服务程序
        ORG     0023H           ;串口中断入口
        RETI                    ;中断返回
```

```
                ORG     002BH                   ;定时器 T2 中断入口
                RETI                            ;中断返回
;
;*********************************
;*                               *
;*          初始化程序           *
;*                               *
;*********************************
CLEARMEMIO:     MOV     R0,#70H                 ;清 70H~76H 显示单元
                MOV     R7,  #07H               ;循环次数
ML0:            MOV     @R0, #00H               ;清 0
                INC     R0                      ;下一地址
                DJNZ    R7,  ML0                ;未完再循环
                MOV     TMOD,#10H               ;T1 为 16 位定时器
                MOV     R4,#14H                 ;1 s 定时用(50 ms 20 次)
                MOV     TL1,#0B0H               ;50 ms 定时用初值
                MOV     TH1,#3CH                ;
                MOV     20H,#00H                ;清 0 操作
                MOV     21H,#00H                ;
                MOV     22H,#00H                ;
                MOV     23H,#00H                ;
                MOV     24H,#00H                ;
                CLR     30H                     ;清停车标志
                SETB    ET1                     ;开 T1 中断
                SETB    EX1                     ;开外中断 1
                SETB    IT1                     ;外中断 1 采用边沿触发
                SETB    IT0                     ;外中断 0 优先级为 1(最高)
                SETB    EX0                     ;开外中断 0
                SETB    EA                      ;开总中断允许
                SETB    TR1                     ;开启定时器 T1
                RET                             ;子程序返回
;
;*************************************************
;*                                               *
;*                  主  程  序                   *
;*                                               *
;*************************************************
;
START:          LCALL   CLEARMEMIO              ;上电初始化
                SETB    P1.6                    ;选择 7.5 V 输出
                CLR     P1.7                    ;选择 7.5 V 输出
                SETB    P3.7                    ;前进状态
                CLR     P3.6                    ;前进状态
```

```
                CLR     P1.0            ;电机供电开始
MAIN:           LCALL   DISP            ;LED 显示一次
                LJMP    MAIN            ;转 MAIN 循环
                NOP                     ;PC 值出错处理
                NOP
                LJMP    START           ;重新初始化
;
;*****************************************************
;*                                                   *
;*          外中断 0 服务程序,用作跑道位置处理       *
;*              23H 作跑道计数器                     *
;*****************************************************
INTEX0:         PUSH    ACC             ;堆栈保护
                PUSH    PSW             ;
                CLR     EX0             ;关中断
                LCALL   DISP            ;LED 显示一次(延时抗干扰)
                JB      P3.2,IN0RET     ;P3.2 为 1 退出(干扰)
                INC     23H             ;跑道计数器加 1
                MOV     A,23H           ;数据入 A
                CJNE    A,#06H,JUDGE1   ;不是第 6 道转 JUDGE1
                LCALL   STOP            ;是第 6 道,停车
                LJMP    IN0RET          ;转中断退出
JUDGE1:         CJNE    A,#03H,JUDGE2   ;不是第 3 道转 JUDGE2
                LCALL   STOPSLOW        ;是第 3 道,变慢车
                LJMP    IN0RET          ;转中断退出
JUDGE2:         CJNE    A,#04H,JUDGE3   ;不是第 4 道转 JUDGE3
                LCALL   FAST            ;是第 4 道,变快车
                LJMP    IN0RET          ;转中断退出
JUDGE3:         CJNE    A,#05H,IN0RET   ;不是第 5 道转 IN0RET 退出
                LCALL   STOPSLOW        ;是第 5 道,变慢车
IN0RET:         CLR     IE0             ;清外中断 0 中断标志
                POP     PSW             ;恢复现场
                POP     ACC             ;
                LCALL   DL7MS           ;延时 7 ms(抗干扰)
                SETB    EX0             ;开外中断 0
                RETI                    ;中断返回
;
;*****************************************************
;*                   慢车控制子程序                  *
;*****************************************************
STOPSLOW:       CLR     P1.6            ;关 7.5 V 电源
                CPL     P3.6            ;反向驱动(刹车)
                CPL     P3.7            ;反向驱动
```

```
                LCALL   DS50MS          ;刹车时间(可根据试车情况调整)
                LCALL   DS50MS          ;
                LCALL   DS50MS          ;
                CPL     P3.6            ;正向驱动
                CPL     P3.7            ;正向驱动
                SETB    P1.7            ;开 4.3 V 电源
                RET                     ;返回
;
;*************************************
;*              快车控制子程序              *
;*************************************
FAST:           CLR     P1.7            ;关 4.3 V 电源
                SETB    P1.6            ;开 7.5 V 电源
                RET                     ;返回
;
;*************************************
;*              停车控制程序              *
;*************************************
STOP:           MOV     23H,#00H        ;跑道计数单元清 0
                CPL     P3.6            ;反向驱动(刹车)
                CPL     P3.7            ;反向驱动(刹车)
                LCALL   DS50MS          ;刹车时间
                LCALL   DS50MS          ;刹车时间(可调整)
                SETB    P1.0            ;关电机电源
                SETB    PT1             ;定时器 T1 为高优先级
                LCALL   DS10S           ;停车 10s
                CLR     PT1             ;恢复 T1 为低优先级
                SETB    P1.6            ;开 7.5 V 电源(高速)
                CLR     P1.7            ;关 4.3 V
                CLR     P1.0            ;电机电源开
                CPL     30H             ;停车点位置判断标志取反
                JB      30H,STREN       ;为 1(中途停车)转 STREN
                LCALL   CLR00           ;是终点,调复 0 程序
STREN:          RET                     ;返回
;
;*************************************
;*              计时清 0 程序              *
;*************************************
CLR00:          MOV     70H,#00H        ;计时单元清 0
                MOV     71H,#00H        ;计时单元清 0
                MOV     72H,#00H        ;计时单元清 0
                MOV     73H,#00H        ;计时单元清 0
                RET                     ;返回
```

```
;
;*****************************************
;*                                       *
;*          外中断 1 程序,里程计数用      *
;*          20H、21H、22H、24H 作计数器    *
;*****************************************
INTEX1:     PUSH    ACC             ;堆栈保护
            PUSH    PSW             ;
            CLR     EX1             ;关外中断 1
            INC     20H             ;圈加 1
  LLLL:     MOV     A,20H           ;判断是否满 6 圈
            CJNE    A,#06H,LLL      ;不满 6 圈转 LLL 退出
            MOV     20H,#00H        ;满 6 圈清 0 进位(6 圈为 1m)
            INC     21H             ;上位加 1
            MOV     A,21H           ;判断是否满 10
            CJNE    A,#0AH,LLL      ;不满 10 转 LLL
            MOV     21H,#00H        ;满 10 清 0 进 1 位
            INC     22H             ;高位加 1
            MOV     A,22H           ;判断是否满 10
            CJNE    A,#0AH,LLL      ;不满 10 转 LLL
            MOV     22H,#00H        ;满 10 清 0 进 1 位
            INC     24H             ;高位加 1
            MOV     A,24H           ;判断是否满 10
            CJNE    A,#0AH,LLL      ;不满 10 转 LLL
            MOV     24H,#00H        ;满 10 清 0
LLL:        MOV     74H,21H         ;将里程数移入显示单元(个位)
            MOV     75H,22H         ;将里程数移入显示单元(十位)
            MOV     76H,24H         ;将里程数移入显示单元(百位)
INTRET:     POP     PSW             ;恢复堆栈
            POP     ACC             ;
            SETB    EX1             ;开外中断 1
            RETI                    ;中断返回
;
;*****************************************
;*                时间计时器程序          *
;*                                       *
;*          (T1 定时中断服务程序)         *
;*****************************************
INTT1:      PUSH    ACC             ;堆栈保护
            PUSH    PSW             ;
            MOV     TL1,#0B0H       ;赋 50 ms 定时初值
            MOV     TH1,#3CH        ;
            DEC     R4              ;减 1
```

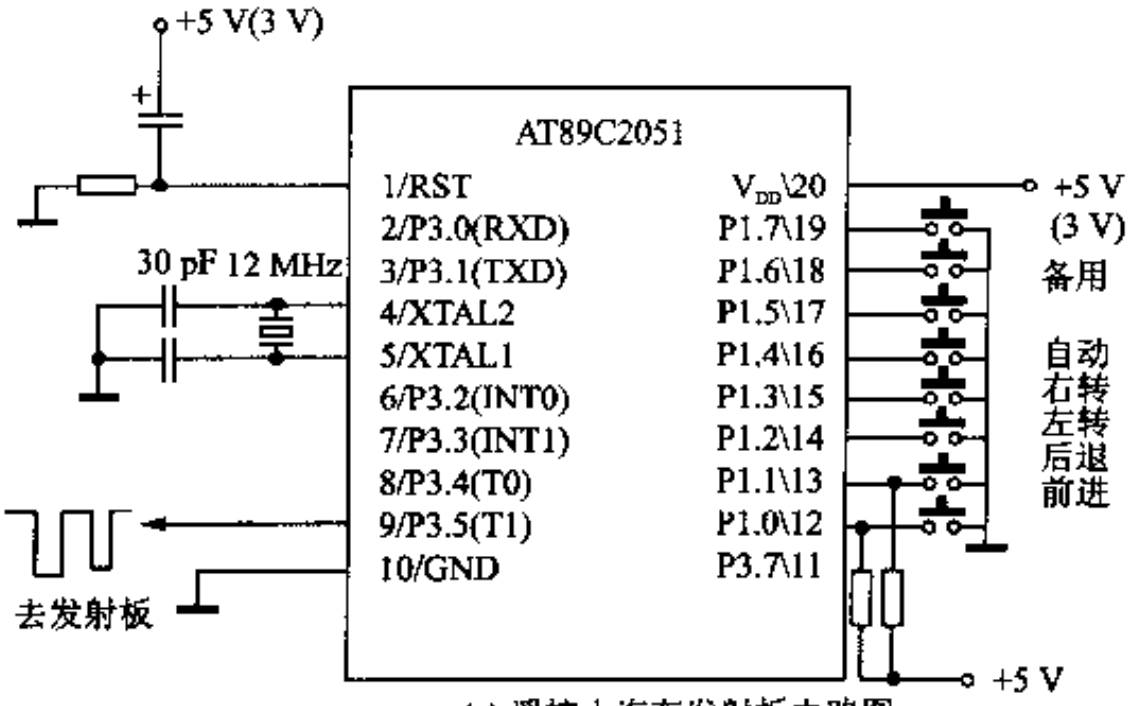
```
        MOV     A,R4            ;
        JNZ     RETT0           ;不为 0 转 RETT0
        MOV     R4,#14H         ;为 0(1 s 到)重赋初值
        MOV     R0,#71H         ;地址指向 71H
        ACALL   ADD1            ;加 1 s 操作
        MOV     A,R3            ;
        CLR     C               ;
        CJNE    A,#60H,CC       ;是否为 60 s?
CC:     JC      RETT0           ;小于 60 转 RETT0
        ACALL   CLR0            ;大于或等于 60 清 0
        MOV     R0,#73H         ;指向分计时地址单元
        ACALL   ADD1            ;分加 1
        MOV     A,R3            ;
        CLR     C               ;
        CJNE    A,#60H,CCC      ;是否为 60 min?
CCC:    JC      RETT0           ;小于 60 转 RETT0
        ACALL   CLR0            ;大于或等于 60 计时单元清 0
RETT0:  POP     PSW             ;恢复堆栈
        POP     ACC             ;
        RETI                    ;中断返回
;
;;;;;;;;;;;;;;;;
;  加 1 操作程序  ;
;;;;;;;;;;;;;;;;
ADD1:   MOV     A,@R0           ;取计数值
        DEC     R0              ;指向低一个地址
        SWAP    A               ;计数值高低 4 位交换
        ORL     A,@R0           ;相或组合成一个数据
        ADD     A,#01H          ;加 1
        DA      A               ;十进制调整
        MOV     R3,A            ;暂存 R3 内
        ANL     A,#0FH          ;高 4 位变 0
        MOV     @R0,A           ;放回低地址
        MOV     A,R3            ;取回 R3 内数据
        INC     R0              ;地址加 1
        SWAP    A               ;高低 4 位交换
        ANL     A,#0FH          ;高 4 位为 0
        MOV     @R0,A           ;放回原地址
        RET                     ;返回
;
;;;;;;;;;;;;;;
;  清 0 程序  ;
;;;;;;;;;;;;;;
```

```
CLR0:       CLR     A                   ;清 A
            MOV     @R0,A               ;对应地址单元清 0
            DEC     R0                  ;指向低一地址
            MOV     @R0,A               ;清 0
            RET                         ;返回
;
;
            ;;;;;;;;;;;;;
            ; 显示程序  ;
            ;;;;;;;;;;;;;
DISP:       MOV     R1,#70H             ;显示数据首址
            MOV     R5,#0FEH            ;扫描字
PLAY:       MOV     A,R5                ;扫描字入 A
            MOV     P2,A                ;从 P2 口输出
            MOV     A,@R1               ;取显示数据
            MOV     DPTR,#TAB           ;取段码表首址
            MOVC    A,@A+DPTR           ;查数据对应段码
            MOV     P0,A                ;段码从 P0 口输出
            LCALL   DL1MS               ;点亮 1 ms
            INC     R1                  ;指向下显示数地址
            MOV     A,R5                ;扫描字入 A
            JNB     ACC.6,ENDOUT        ;ACC.6=0 转 ENDOUT 结束
            RL      A                   ;循环左移
            MOV     R5,A                ;放回 A
            AJMP    PLAY                ;转 PLAY 再显示
ENDOUT:     MOV     P2,#0FFH            ;显示结束处理。P2 口置 1
            RET                         ;子程序结束
;
;LED 共阴段码表(0～9)
TAB:        DB 3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH
;
;1ms 延时程序
DL1MS:      MOV     R6,#14H
DL1:        MOV     R7,#19H
DL2:        DJNZ    R7,DL2
            DJNZ    R6,DL1
            RET
;
;延时程序,用调用显示程序实现,可使 LED 显示稳定
DS50MS:     LCALL   DISP                ;(7 ms)
            LCALL   DISP
            LCALL   DISP
```

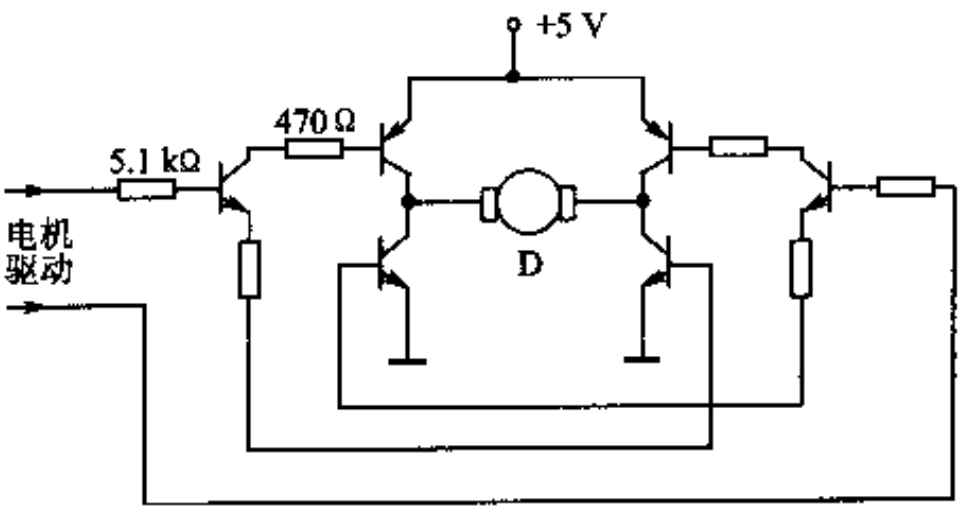
```
DS20MS:     LCALL   DISP
            LCALL   DISP
            LCALL   DISP
            RET
;
;10 s 延时程序,用调用显示程序实现,可使 LED 显示不熄灭
DS10S:      MOV     R2,#08H             ;(8×11×16×7=9 956 ms)
TI0:        MOV     R0,#0B0H
TI1:        LCALL   DISP
            DJNZ    R0,TI1
            DJNZ    R2,TI0
            RET
;7 ms 延时程序,跑道计数器抗干扰用
DL7MS:      SETB    PX1             ;外中断 1 置高优先级
            MOV     R2,#0EH         ;赋定时值
DL11:       LCALL   DISP            ;调用显示程序
            DJNZ    R2,DL11         ;循环
            CLR     PX1             ;外中断 1 恢复低优先级
            RET                     ;返回
;
            END                     ;程序结束
```

第 14 章　实例 9　遥控小汽车的设计

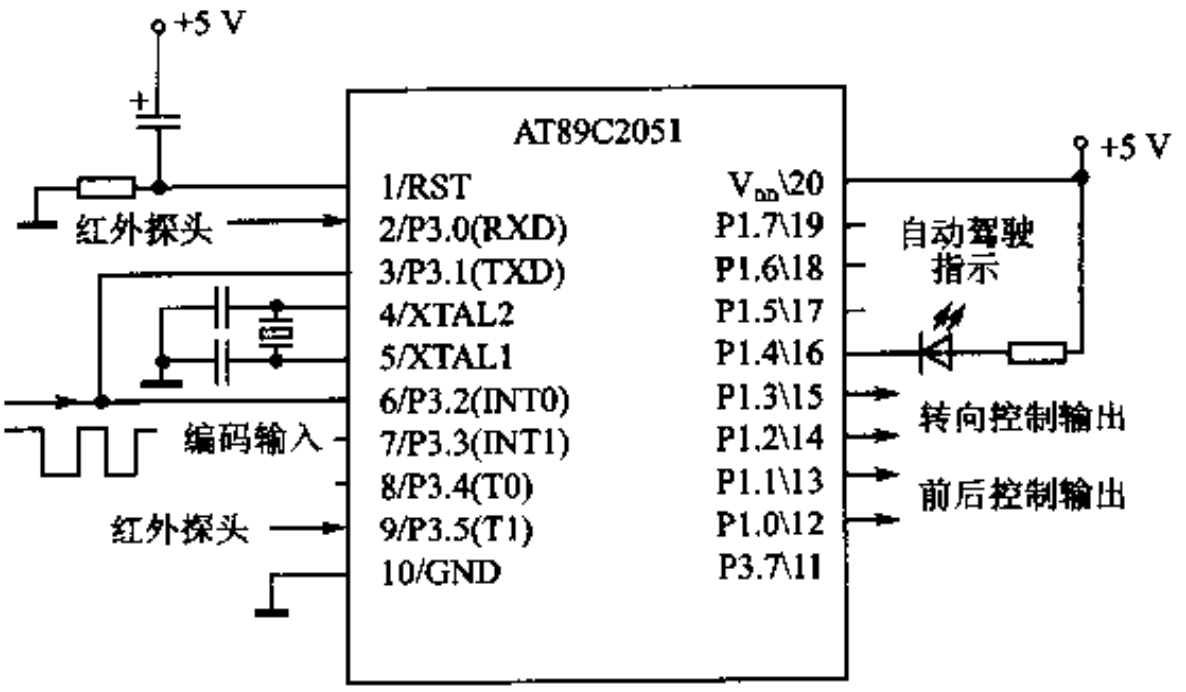
本设计采用 AT89C2051 作遥控发射器及接收处理器，汽车前进、后退与转向分别用两个电机，采用桥式开关电路驱动电机。无线传送用接收/发射模块完成。小汽车能前进、后退、左转、右转及自动驾驶(碰到障碍时能自动改变行驶方向)。遥控小汽车的电路原理如图 14.1 所示。



(a) 遥控小汽车发射板电路图



(b) 电机驱动电路



(c) 遥控小汽车接收板电路图

图 14.1　遥控小汽车硬件原理图

1. 系统硬件电路的设计

(1) 发射电路板

发射电路板共设有 5 个按键开关，分别作为汽车左转弯、右转弯、前进、后退及自动驾驶控制用。P1.0 和 P1.1 口作为输入口时应接上拉电阻，遥控编码从 P3.5 脚输出至无线发射模块。该板采用脉冲个数编码以区别不同的按键功能，具体定义如下：

前进键：按下发 2 个脉冲，释放发 8 个脉冲。

后退键：按下发 3 个脉冲，释放发 8 个脉冲。

左转键：按下发 4 个脉冲，释放发 7 个脉冲。

右转键：按下发 5 个脉冲，释放发 7 个脉冲。

自动键：按下发 6 个脉冲。

有关编码的格式要求可参看设计实例 6 中的介绍。

(2) 接收电路

无线接收模块输出的编码脉冲从 P3.2、P3.1 口输入，采用中断接收方式处理脉冲编码。电机采用桥式驱动，P1.0、P1.1 口作前后驱动电机控制用，P1.2、P1.3 口作转向电机控制用。P3.0 接前障碍红外线探测头，P3.5 接后障碍红外线探测头。P1.4 口接一个 LED 发光管用作自动驾驶指示。

2. 系统主要程序的设计

(1) 发射板控制程序的设计

发射板控制程序由主程序和键扫描子程序组成，在主程序中，采用调用键扫描子程序来完成各个按键的功能，其键扫描功能子程序流程图如图 14.2 所示。

(2) 接收处理程序的设计

① 初始化程序：对转向电机、前后驱动电机上电时设为停止状态，开中断允许等。

② 主程序：根据标志位 00H 的值判断进入自动驾驶或手动控制状态，其程序流程图如图 14.3所示。

③ 中断接收程序：对第一位脉冲的宽度进行验证，然后进行计数，根据脉冲的个数进行相应的控制操作，其程序流程图如图 14.4 所示。

以下是遥控小汽车发射及接收完整源程序：

```
;                    ********************************************
;                    *                                          *
;                    *                遥控编码控制器            *
;                    *                采用 89C2051              *
;                    *                                          *
;                    ********************************************
;
;P1.0 口按钮为前进，P1.1 口按钮为后退，P1.2 口按钮为左转弯，P1.3 口按钮为右转弯，
;P1.4 口按钮为自动驾驶，P3.5 口为编码输出
;
;*******************************************
;*                                         *
;*              主程序和中断程序入口       *
```

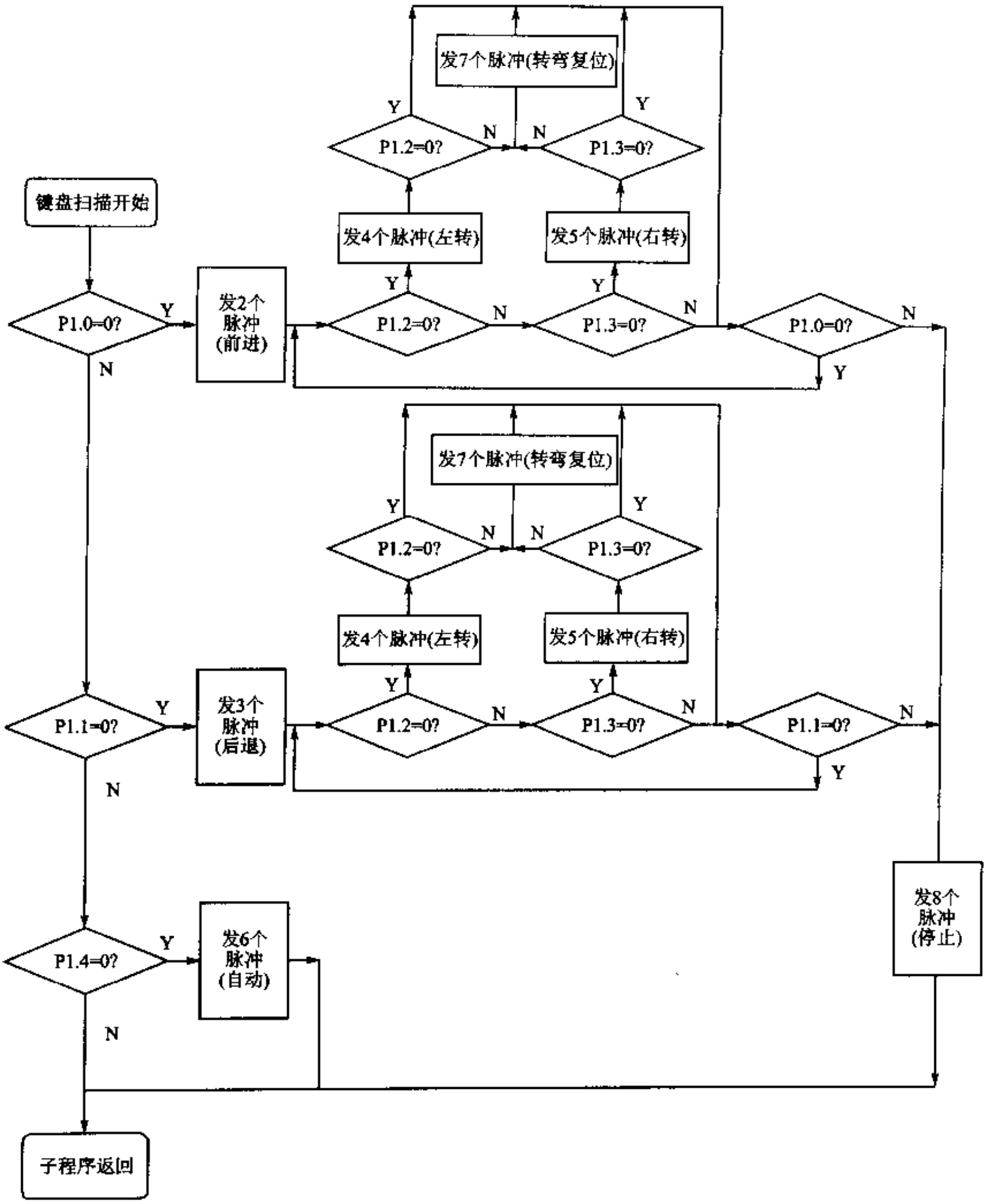


图 14.2　遥控小汽车发射程序流程图

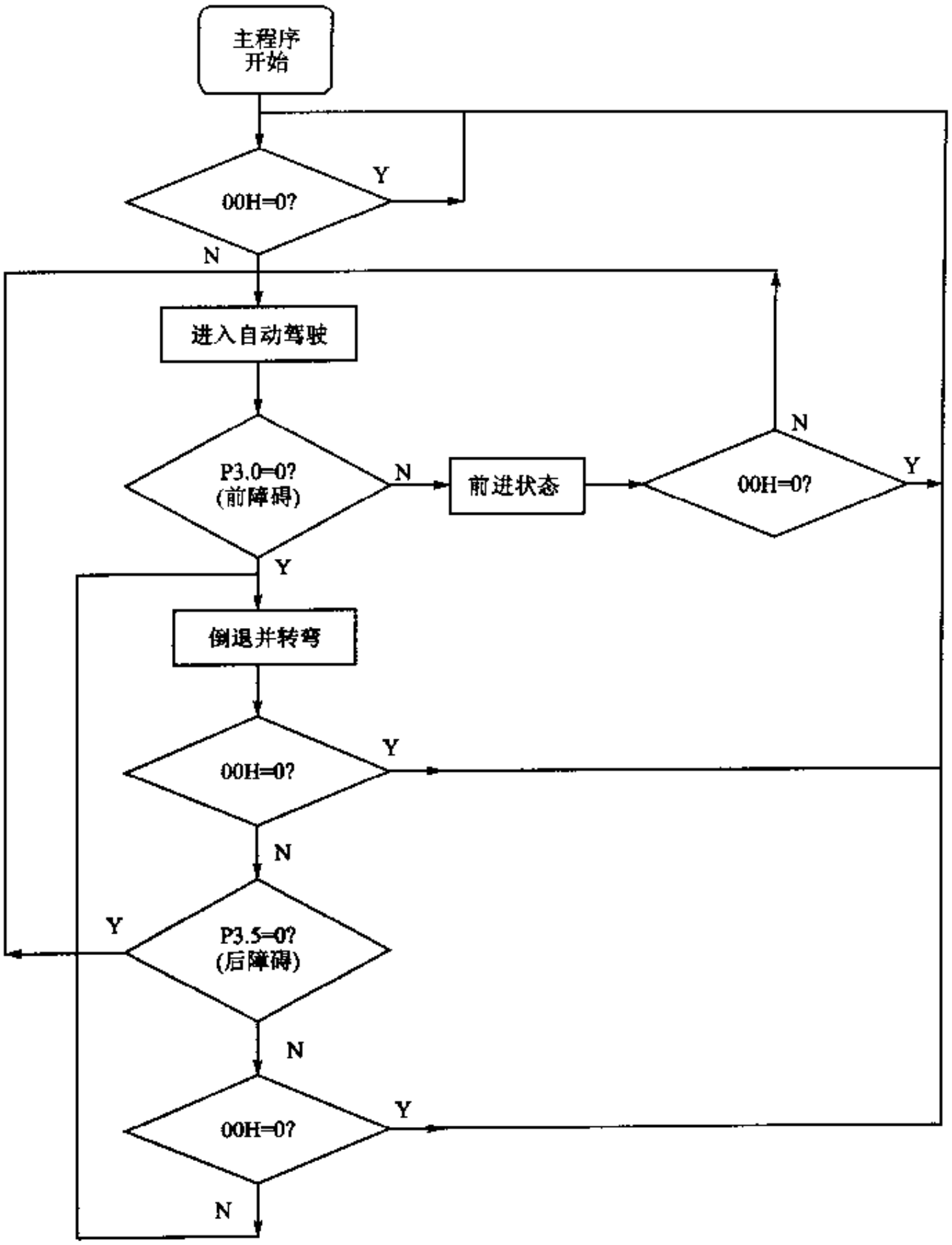


图 14.3　遥控小汽车接收板主程序流程图

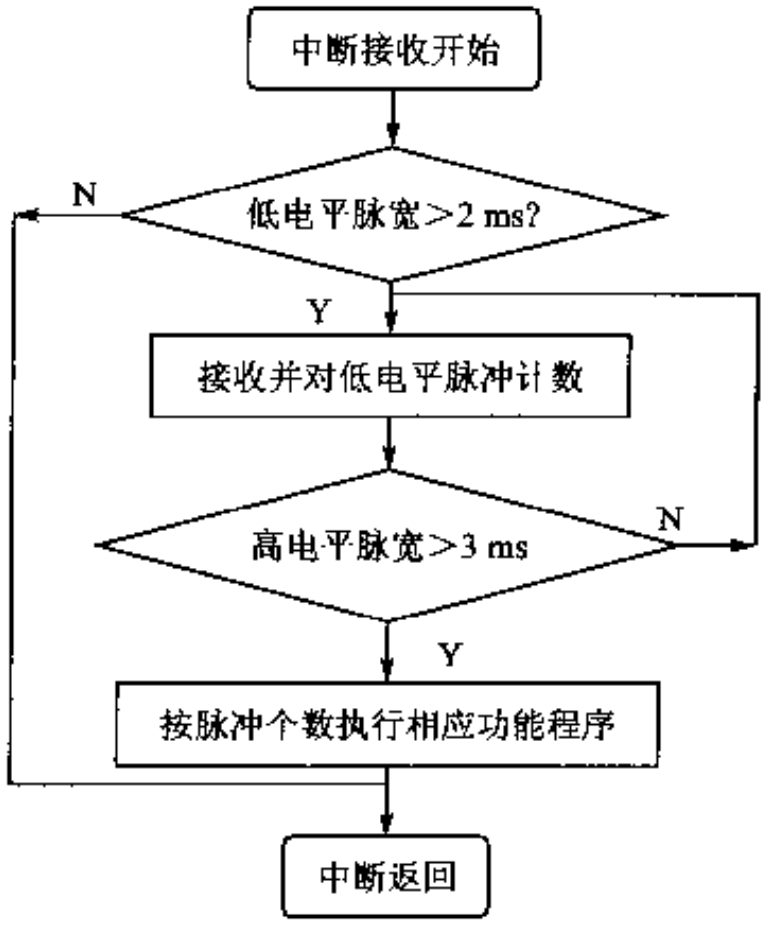


图 14.4　遥控小汽车中断接收程序流程图

```
;*                                                    *
;*****************************************************
        ORG     0000H                   ;程序执行开始地址
        LJMP    START                   ;跳至 START 执行
        ORG     0003H                   ;外中断 0 中断入口地址
        RETI                            ;中断返回(不开中断)
        ORG     000BH                   ;定时器 T0 中断入口地址
        RETI                            ;中断返回(不开中断)
        ORG     0013H                   ;外中断 1 中断入口地址
        RETI                            ;中断返回(不开中断)
        ORG     001BH                   ;定时器 T1 中断入口地址
        RETI                            ;中断返回(不开中断)
        ORG     0023H                   ;串行口中断入口地址
        RETI                            ;中断返回(不开中断)
        ORG     002BH                   ;定时器 T2 中断入口地址
        RETI                            ;中断返回(不开中断)
;
;*****************************************************
;*                                                   *
;*              初始化程序中的各变量                 *
;*                                                   *
;*****************************************************
CLEARMEMIO:  CLR     A                  ;清 A
             DEC     A                  ;A 为 #0FFH
             MOV     P1,A               ;P1 口置 1
             MOV     P3,A               ;P3 口置 1
```

```
                MOV     IE,#00H          ;关所有中断
                RET                      ;子程序返回
;
;
;
;*************************************
;*                                   *
;*            主　程　序             *
;*                                   *
;*************************************
START:          LCALL   CLEARMEMIO       ;调用初始化
;主体程序
MAIN:           LCALL   KEYWORK          ;调用查键子程序
                LJMP    MAIN             ;转 MAIN 循环
                NOP                      ;PC 值出错处理
                NOP
                NOP
                LJMP    START            ;转 START 重新启动
;
;*************************
;*      键盘工作子程序   *
;*************************
KEYWORK:        MOV     P1,#0FFH         ;置输入状态
                JNB     P1.0,KEY0        ;读 P1.0 口,为 0 转 KEY0
                JNB     P1.1,KEY1        ;读 P1.1 口,为 0 转 KEY1
                JNB     P1.4,KEY4        ;读 P1.4 口,为 0 转 KEY4
KEYOUT:         RET                      ;没键按下,退出
;
KEY0:           LCALL   DL10MS           ;延时 10 ms 消抖
                JB      P1.0,KEYOUT      ;干扰,退出
                LJMP    KEYFUN00         ;转 KEYFUN00
KEY1:           LCALL   DL10MS           ;延时 10 ms 消抖
                JB      P1.1,KEYOUT      ;干扰,退出
                LJMP    KEYFUN01         ;转 KEYFUN01
KEY2:           LCALL   DL10MS           ;延时 10 ms 消抖
                JB      P1.2,KEYOUT      ;干扰,退出
                LJMP    KEYFUN02         ;转 KEYFUN02
KEY3:           LCALL   DL10MS           ;延时 10 ms 消抖
                JB      P1.3,KEYOUT      ;干扰,退出
                LJMP    KEYFUN03         ;转 KEYFUN03
KEY4:           LCALL   DL10MS           ;延时 10 ms 消抖
                JB      P1.4,KEYOUT      ;干扰,退出
```

```
                LJMP    KEYFUN04        ;转 KEYFUN04
KEYWORK1:       JNB     P1.2,KEY2       ;P1.2 口为 0 转 KEY2
                JNB     P1.3,KEY3       ;P1.3 口为 0 转 KEY3
                LCALL   KEYFUN05        ;调转弯复位子程序
                RET                     ;返回
;
KEYFUN00:       MOV     A,#02H          ;发 2 个脉冲赋值
                LCALL   REMOTE          ;发射 2 个脉冲(前进)
WAIT0:          LCALL   KEYWORK1        ;调一次转弯查键子程序
                JNB     P1.0,WAIT0      ;等待按键释放
                LCALL   KEYFUN06        ;释放时发 8 个脉冲(停止)
                RET                     ;返回
;
KEYFUN01:       MOV     A,#03H          ;发 3 个脉冲赋值
                LCALL   REMOTE          ;发射 3 个脉冲(前进)
WAIT1:          LCALL   KEYWORK1        ;调一次转弯查键子程序
                JNB     P1.1,WAIT1      ;等待按键释放
                LCALL   KEYFUN06        ;释放时发 8 个脉冲(停止)
                RET                     ;返回
;
KEYFUN02:       MOV     A,#04H          ;发 4 个脉冲赋值
                LCALL   REMOTE          ;发射 4 个脉冲(左转)
                JNB     P1.2,KEYOUT1    ;键按下不放,转 KEYOUT1 返回
                LCALL   DL10MS          ;左转键放开,延时 10 ms
                JNB     P1.2,KEYOUT1    ;为 0(干扰),转 KEYOUT1 返回
                LCALL   KEYFUN05        ;左转键放开发 7 个脉冲(转弯复位)
                RET                     ;返回
;
KEYFUN03:       MOV     A,#05H          ;发 5 个脉冲赋值
                LCALL   REMOTE          ;发射 5 个脉冲(右转)
                JNB     P1.3,KEYOUT1    ;键按下不放,转 KEYOUT1 返回
                LCALL   DL10MS          ;右转键放开,延时 10 ms
                JNB     P1.3,KEYOUT1    ;为 0(干扰),转 KEYOUT1 返回
                LCALL   KEYFUN05        ;右转键放开发 7 个脉冲(转弯复位)
                RET                     ;返回
KEYOUT1:        LJMP    KEYOUT          ;跳至 KEYOUT 返回
;
KEYFUN04:       JNB     P1.4,KEYFUN04   ;等待按键释放
                LCALL   DL10MS          ;延时消抖动
                JNB     P1.4,KEYFUN04   ;P1.4 为 0,系干扰,转 KEYFUN04 再等待
                MOV     A,#06H          ;按键释放,发 6 个脉冲(进入自动驾驶)
                LCALL   REMOTE          ;发射一次
                RET                     ;返回
```

```
;
KEYFUN05:   MOV     A,#07H          ;发 7 个脉冲(转弯复位)
            LCALL   REMOTE          ;发射一次
            RET                     ;返回
KEYFUN06:   MOV     A,#08H          ;发 8 个脉冲(停车)
            LCALL   REMOTE          ;发射一次
            RET                     ;返回
;
;
;*************************************************
;*                                               *
;*                  编码发送程序                 *
;*                                               *
;*************************************************
;按 A 中数值发射脉冲
REMOTE:     MOV     R1,A            ;发射脉冲数入 A
            LJMP    OUT3            ;第一个脉冲处理
OUT:        MOV     R0,#55H         ;1 ms 脉宽定时值
OUT1:       CLR     P3.5            ;发低电平脉宽
            NOP                     ;延时循环(周期约为 21 μs)
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            DJNZ    R0,OUT1         ;定时时间未到,转 OUT1 循环
            MOV     R0,#55H         ;1 ms 脉宽定时值
OUT2:       SETB    P3.5            ;发高电平脉宽
            NOP                     ;延时循环(周期约为 21 μs)
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            DJNZ    R0,OUT2         ;定时时间未到,转 OUT2 循环
            DJNZ    R1,OUT          ;脉冲数未发完转 OUT 再循环
            LCALL   DL10MS          ;脉冲发完延时 10 ms
```

```
                RET                             ;返回
OUT3:           MOV     R0,#0FFH                ;3 ms 脉宽定时值
                LJMP    OUT1                    ;转 OUT1 红外线发射
;
;
;*****************************************
;*                                       *
;*          延时程序(513 μs)             *
;*                                       *
;*****************************************
DELAY:          MOV     R2,#0FFH
DELAY1:         DJNZ    R2,DELAY1
                RET
;
;*****************************************
;*                                       *
;*          延时 10 ms 程序              *
;*          (消按键抖动用)               *
;*****************************************
DL10MS:         MOV     R3,#14H
DL10MS1:        LCALL   DELAY
                DJNZ    R3,DL10MS1
                RET
DL500MS:        MOV     R4,#32H
DL500MS1:       LCALL   DL10MS
                DJNZ    R4,DL500MS1
                RET
;
;
                END                             ;程序结束

;               ***************************************
;               *                                     *
;               *          遥控接收解码               *
;               *          采用 AT89C2051             *
;               *                                     *
;               ***************************************
;
; P1.0～P1.1 口为电机前后驱动;P1.2～P1.3 口为左右转弯驱动;P1.4 口为自动驾驶指示,
; 编码输入为 P3.1 和 P3.2(INT0)口;P3.0 和 P3.5 分别为前后红外线探头,在自动驾驶时
; 用以控制小汽车前进和倒退
```

```
;*************************************
;*                                   *
;*          主程序和中断程序入口     *
;*                                   *
;*************************************
            ORG     0000H           ;程序执行开始地址
            LJMP    START           ;跳至 START 执行
            ORG     0003H           ;外中断 0 中断入口地址
            LJMP    INTEX0          ;跳至 INTEX0 中断服务程序
            ORG     000BH           ;定时器 T0 中断入口地址
            RETI                    ;中断返回(不开中断)
            ORG     0013H           ;外中断 1 中断入口地址
            RETI                    ;中断返回(不开中断)
            ORG     001BH           ;定时器 T1 中断入口地址
            RETI                    ;中断返回(不开中断)
            ORG     0023H           ;串行口中断入口地址
            RETI                    ;中断返回(不开中断)
            ORG     002BH           ;定时器 T2 中断入口地址
            RETI                    ;中断返回(不开中断)
;
;*************************************
;*                                   *
;*          初始化程序               *
;*                                   *
;*************************************
CLEARMEMIO: CLR     A               ;清 A
            MOV     P1,A            ;P1 口置 0
            DEC     A               ;A 为#0FFH
            MOV     P3,A            ;P3 口置 1
            SETB    P1.4            ;关自动驾驶指示灯
            CLR     00H             ;自动驾驶标志位清 0
CLEARMEM:   MOV     IE,#00H         ;关所有中断
            SETB    EX0             ;开外中断
            SETB    EA              ;开总中断允许
            RET                     ;返回
;
;*************************************
;*                                   *
;*              主 程 序             *
;*                                   *
;*************************************
START:      LCALL   CLEARMEMIO      ;上电初始化
MAIN:       JB      00H,AUTO2       ;00H 位标志为 1 转自动驾驶
```

```
                LJMP    MAIN            ;主程序循环
                NOP                     ;PC 值出错处理
                NOP
                LJMP    START           ;重新上电启动
AUTO2:          LJMP    AUTO            ;至自动驾驶程序
;**************************************
;*                                    *
;*          中断接收程序              *
;*                                    *
;**************************************
INTEX0:         CLR     EX0             ;关中断
                JNB     P3.1,READ1      ;P3.1 为 0 转 READ1
READOUTT0:      SETB    EX0             ;干扰,中断退出
                RETI                    ;中断返回
;
READ1:          CLR     A               ;清 A
                MOV     DPH,A           ;清 DPTR
                MOV     DPL,A           ;
HARD1:          JB      P3.1,HARD11     ;P3.1 为高电平时转 HARD11
                INC     DPTR            ;低电平脉冲宽度时间计数
                NOP                     ;空操作延时
                NOP
                AJMP    HARD1           ;跳回循环(周期约为 8 μs)
HARD11:         MOV     A,DPH           ;取 DPTR 高 8 位计数值
                JZ      READOUTT0       ;若为 0,则脉宽不足 3 ms 退出中断
                CLR     A               ;脉宽大于 2 ms,是第一个脉冲
READ11:         INC     A               ;脉冲计数加 1
READ12:         JNB     P3.1,READ12     ;低电平等待
                MOV     R1,#06H         ;高电平脉宽判断用
READ13:         JNB     P3.1,READ11     ;低电平转 READ11(脉冲计数)
                LCALL   DELAY           ;延时 513 μs
                DJNZ    R1,READ13       ;高电平脉宽小于 3 ms,转 READ13 循环
                DEC     A               ;高电平脉宽大于 3 ms,计数结束,A 减 1
                DEC     A               ;再减 1
                JZ      FUN0            ;收到 2 个脉冲,执行前进功能
                DEC     A
                JZ      FUN1            ;收到 3 个脉冲,执行后退功能
                DEC     A
                JZ      FUN2            ;收到 4 个脉冲,执行左转功能
                DEC     A
                JZ      FUN3            ;收到 5 个脉冲,执行右转功能
                DEC     A
                JZ      FUN4            ;收到 6 个脉冲,执行自动驾驶功能
```

```
        DEC     A
        JZ      FUN5        ;收到 7 个脉冲,执行转弯复位功能
        DEC     A
        JZ      FUN6        ;收到 8 个脉冲,执行停车功能
        NOP                 ;出错处理
        NOP                 ;
        LJMP    READOUTT0   ;退出
FUN0:   SETB    P1.0        ;前进状态
        LJMP    READOUTT0   ;
FUN1:   SETB    P1.1        ;后退状态
        LJMP    READOUTT0   ;
FUN2:   SETB    P1.2        ;左转弯状态
        LJMP    READOUTT0   ;
FUN3:   SETB    P1.3        ;右转弯状态
        LJMP    READOUTT0   ;
FUN4:   CPL     00H         ;自动驾驶标志取反
        JB      00H,AUTO1   ;00H=1 时,进入自动驾驶
        CLR     P1.0        ;00H=0 时,进入遥控驾驶,停车操作
        CLR     P1.1        ;停车操作
        CLR     P1.2        ;停车操作
        CLR     P1.3        ;停车操作
        SETB    P1.4        ;关自动驾驶灯
        LJMP    READOUTT0   ;中断退出
AUTO1:  CLR     P1.4        ;进入自动驾驶,开自动驾驶指示灯
        SETB    EX0         ;开中断
        RETI                ;中断返回
AUTO:   JNB     P3.0,BL     ;自动驾驶控制程序,前面有障碍转 BL
        SETB    P1.0        ;前面无障碍,前进状态
        CLR     P1.1        ;前面无障碍,前进状态
        CLR     P1.2        ;前面无障碍,前进状态
        CLR     P1.3        ;前面无障碍,前进状态
        JNB     00H,OUT1    ;标志为 0 转 OUT1(转遥控)
        AJMP    AUTO        ;为 1 转 AUTO 循环
BL:     CLR     P1.0        ;前面有障碍,后退并左转处理
        SETB    P1.1        ;置后退状态
        SETB    P1.2        ;左转状态
        CLR     P1.3        ;
        JNB     00H,OUT1    ;标志为 0 转 OUT1(转遥控)
        JNB     P3.5,AUTO   ;后面有障碍转 AUTO(前进处理)
        AJMP    BL          ;后无障碍转 BL,继续后退循环
OUT1:   CLR     P1.0        ;自动驾驶退出程序,关闭驱动及转向电机
        CLR     P1.1        ;
        CLR     P1.2        ;
```

```
        CLR     P1.3        ;
        SETB    P1.4        ;关自动驾驶指示灯
        LJMP    MAIN        ;转回主程序
FUN5:   CLR     P1.2        ;转弯停止程序
        CLR     P1.3        ;
        LJMP    READOUTT0   ;退出中断
FUN6:   CLR     P1.0        ;遥控停车程序,关闭驱动及转向电机
        CLR     P1.1        ;
        CLR     P1.2        ;
        CLR     P1.3        ;
        LJMP    READOUTT0   ;中断退出
;*************************************
;*                                   *
;*          延时程序(513 μs)         *
;*                                   *
;*************************************
DELAY:  MOV     R0,#0FFH    ;
DELAY1: DJNZ    R0,DELAY1   ;
        RET                 ;
;
;
        END                 ;程序结束
```

第 15 章　实例 10　汽车行驶信息发送与接收器的设计

汽车超车及转弯时经常容易出交通意外，特别是雨雾天气，转向及刹车指示灯能见度减小，使贺驶员不能及时了解前后车的行车意向。在汽车上设计一种汽车间行驶信息红外线自动接收发送电路，能自动显示前后车的转向、刹车情况并用声响提醒，对提高汽车行车安全具有一定的意义。本设计选择了汽车左转弯、右转弯和刹车 3 种汽车行驶状态作为发送信息，接收信息有前车左转弯、前车右转弯、前车刹车和后车超车 4 种显示信息。在汽车行驶中，当前车转弯、刹车或后车超车时都能在本汽车上显示并发出声响提醒。

汽车行驶信息接发器电路如图 15.1 所示。

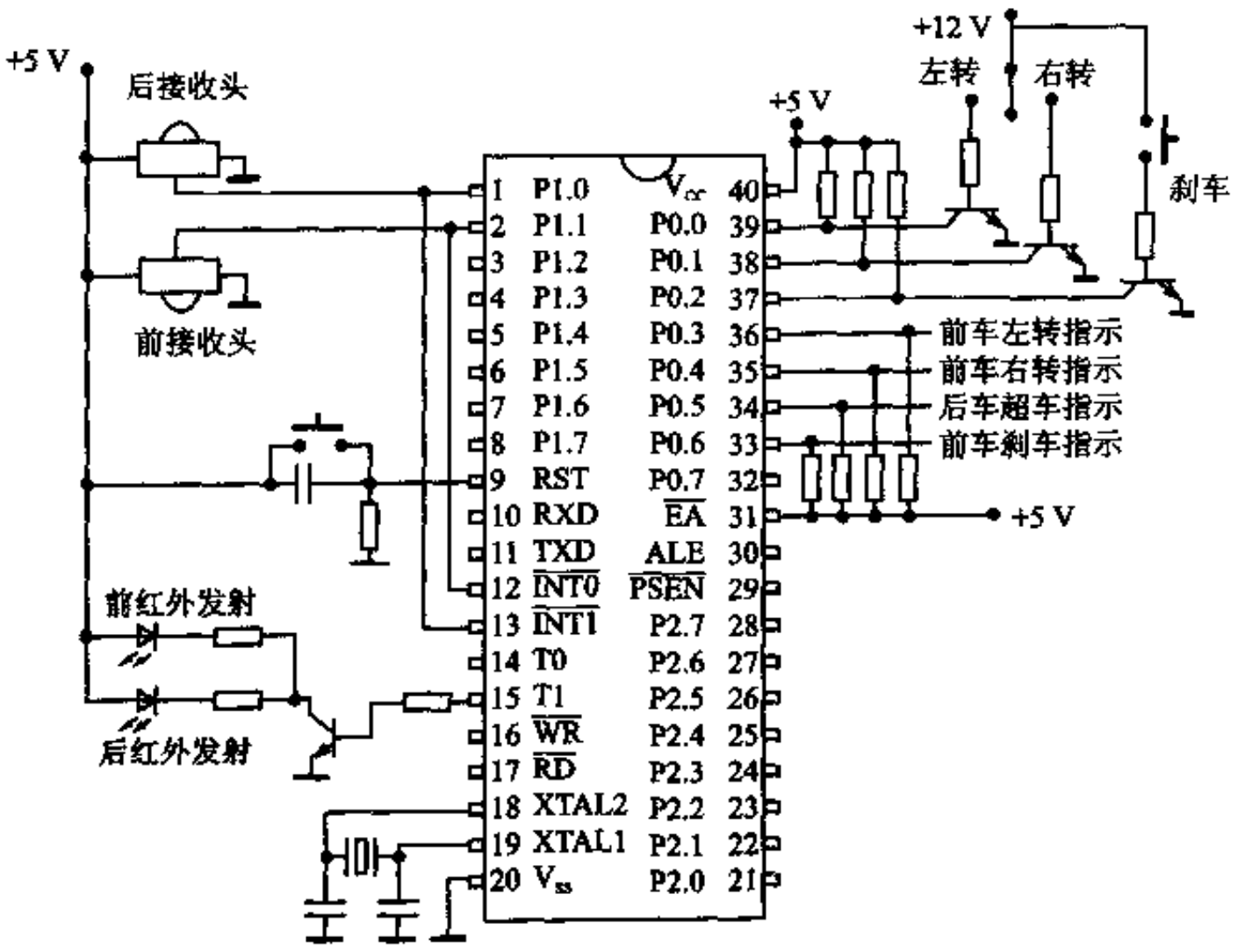


图 15.1　汽车行驶信息接发器电路图(接发一体板)

1. 系统硬件电路的设计

(1) 输入系统接口电路

输入接口电路有处理红外线接收的红外接收器及用于转向、刹车信号输入的电平转换电路。红外接收器采用通用远红外接收解调一体化成品，其器件为三引脚结构，安装使用方便，其信号脚可直接接单片机 P1.0、P1.1 及中断输入端口。P1.0 用于接收后车的行驶信息信号，P1.1 用于接收前车的行驶信息信号。电平转换接口电路采用 9013 三极管。当转向开关或刹车开关闭合时，其 P0.0、P0.1、P0.2 三端口相应的电平变为零。P0.0 为左转弯输入，P0.1为右转弯输入，P0.2 为刹车。

(2) 输出电路

单片机从 P0.3～P0.6 输出前后车的行驶信息提示信号，其中 P0.3 用于前车左转弯指示及提醒，P0.4 用于前车右转弯指示及提醒，P0.5 用于后车超车指示及提醒，P0.6 用于前车刹车指示及提醒。当某一输出端口为低电平时，相应的字符灯点亮并发出声响提醒。本车行驶信息的发送是从 P3.5(T1)输出，是一组调制频率为 40 kHz 的方波脉冲(见图 15.2)，通过三极管放大，由安装在汽车前后位置的红外线发射管发出。

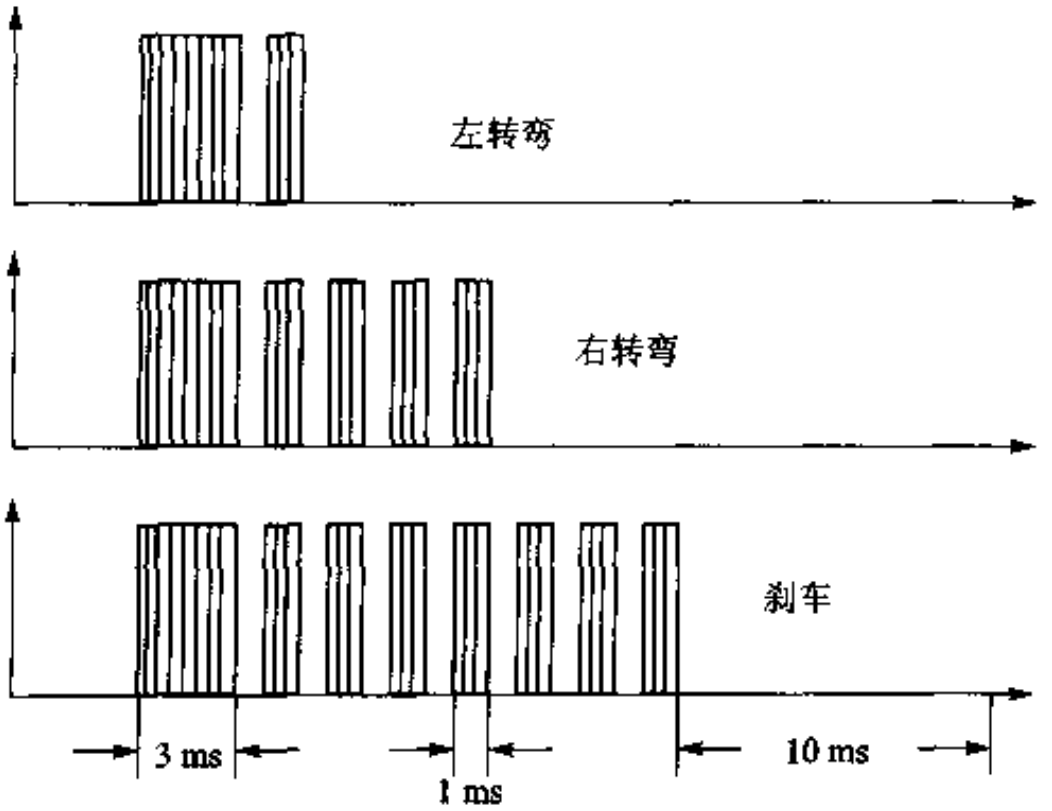


图 15.2　P3.5 端口输出的编码波形图

(3) 基本系统组成

本系统采用 12 MHz 晶振时钟频率，30 pF 的瓷片补偿电容，上电复位采用最简单的 RC 电路，片外存储器选择脚(31 脚$\overline{EA}$)接正电源。

2. 数据帧的编码格式及发送/接收过程

(1) 编码的格式

本系统采用脉冲个数编码，分别代表左转弯、右转弯、刹车 3 种状态，其中左转弯为 2 个脉冲，右转弯为 5 个脉冲，刹车为 8 个脉冲。为了增加接收的可靠性，第一位码宽为 3 ms，其余为 1 ms，数据帧间隔大于 10 ms，如图 15.3 所示。

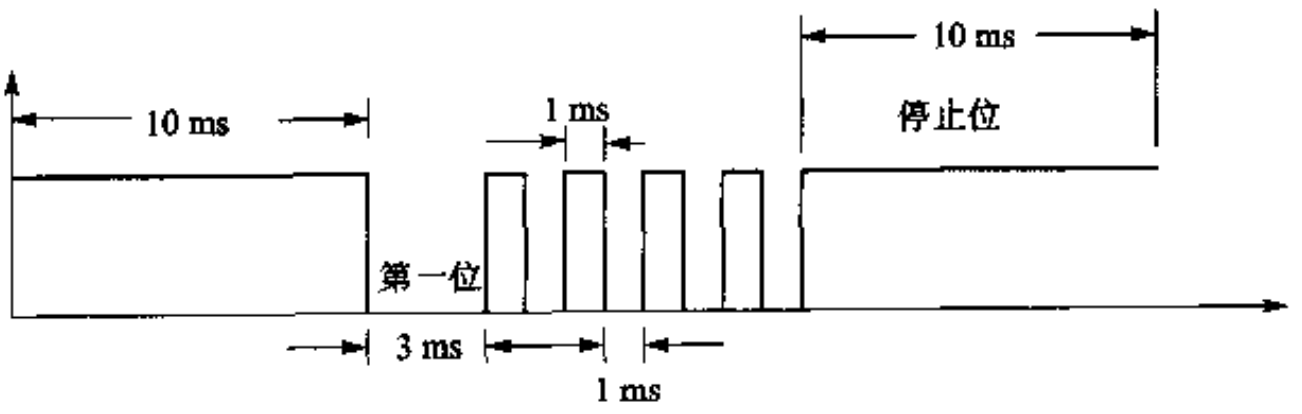


图 15.3　红外线接收器输出的一帧数据波形

(2) 数据帧的发送过程

当单片机检测到 P0.0～P0.2 端口为低电平时，先设置发送标志，然后依次发送数据帧。发送数据时，中断将被禁止。当刹车(转向)开关闭合时，数据帧将被重复连续地发射，直到开

关松开为止。

(3) 数据帧的接收过程

当红外线接收器输出数据帧脉冲时，第一位码的低电平将启动中断程序，实时接收数据帧。在数据帧接收时，中断将被关断，并且对第一位(起始位)码的码宽进行验证。若第一位低电平码的脉宽小于 2 ms，将作误帧处理。当间隔位的高电平脉宽大于 3 ms 时，结束接收，然后根据累加器 A 中的脉冲个数，使相应的输出口(P0.3～P0.6)为低电平，驱动显示及信响电路。

3. 系统主要程序的设计

(1) 初始化程序

将 P0～P3 端口置输入状态，堆栈指针设于 70H 处，定时器 T1 设为 8 位自动重装初值模式，定时时间为 13 μs，用于 40 kHz 的红外线信号调制。

(2) 主程序

顺序检测 P0.0～P0.2 端口，若某端口为低电平，则转发送程序。结束后延时 60 ms，再转检测程序循环。主程序流程图如图 15.4 所示。

(3) 中断接收程序

外中断 0 接收前车信息码，外中断 1 接收后车信息码。当外中断允许并且红外线接收头输出脉冲编码时，中断程序实时接收编码并且对脉冲个数进行计数，根据接收脉冲个数分别控制 P0.3～P0.6 显示端口。中断接收程序流程图如图 15.5 所示。

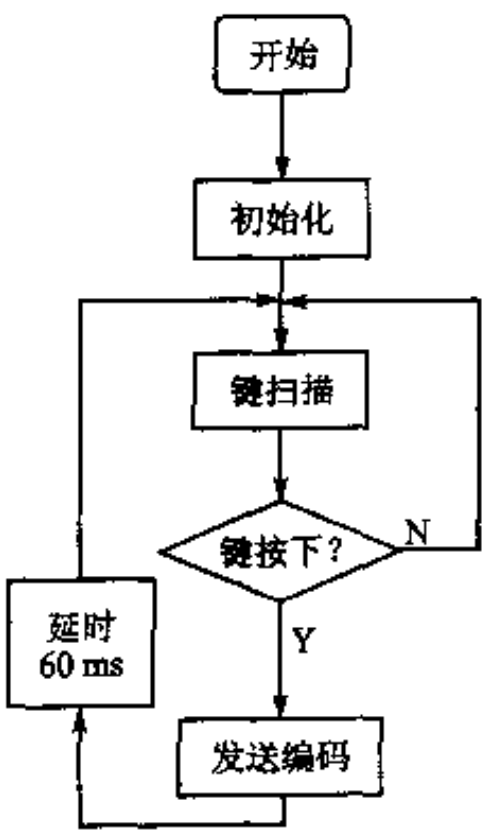


图 15.4　主程序流程图

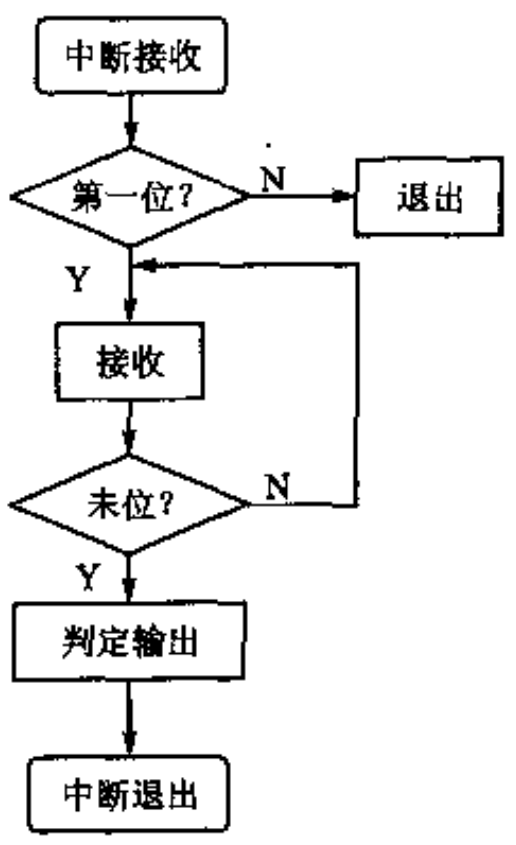


图 15.5　中断接收程序流程图

(4) 信息码发送程序

将扫键程序中的脉冲个数设定值调入寄存器，控制发射脉冲个数，其中第一个脉冲宽度为 3 ms，其余为 1 ms，发送完成后延时 10 ms 作结束标志。当发送编码时，开启定时中断 T1，以实现 40 kHz 的编码调制。

4. 系统调试中的问题及解决办法

(1) 接收前后车信息的概率比问题

在本系统中，前车信息采用外中断 0 接收，后车信息采用外中断 1 接收。由于外中断 0 的

优先级高于外中断1，在同时有前后车发出信息的条件下，前车接收信息的概率要大大高于后车，因此，在两个中断程序设计中，每个中断程序执行一次，均需让出中断优先权，这样使接收前后车的概率基本相同。

（2）本车信息的发送与前后车信息接收的时间分配问题

本系统中单片机发送一次信息时，时间最短为15 ms(发左转弯信号)，最长需48 ms(同时发刹车及右转弯信号)。当汽车又要接收又要发送信息时，起始位的发出时机是相当重要的。如果在发本车行驶信息时，前后车刚好发出起始位，由于中断的屏蔽将错过时机。为了解决这一问题，通过在信息发送后插入延时时间，以增加接收概率。通过实验的接/发测试及延时时间的调整，证明在延时时间为60 ms时，接收和发送都能满足使用的要求。延时时间太短，接收性能低；延时时间太长，发送率变低。当延时时间为60 ms时，发送与接收的时间比例在1∶4～4∶5之间。

当采用一块单片机分时完成接收和发送功能时，接收的成功率会受到一定的影响；如果采用两片单片机分别负责发送和接收工作，则能达到100%实时信息的接收。

以下是汽车行驶信息接发器的控制源程序：

```
;            ************************************
;            *          汽车通信控制程序          *
;            *            LOU RANMIAO           *
;            *            2001.4.24             *
;            ************************************
;*****************************************************
;*
;*   后车信号接收 ─┬─→ P1.0 │1          40│ Vcc                    *
;*   前车信号接收 ─┼┬→ P1.1 │2          39│ P0.0 ←─ 左转开关(L)      *
;*               ││   P1.2 │3          38│ P0.1 ←─ 右转开关(L)      *
;*               ││   P1.3 │4          37│ P0.2 ←─ 刹车开关(L)      *
;*               ││   P1.4 │5          36│ P0.3 ─→ 前车左转指示      *
;*               ││   P1.5 │6          35│ P0.4 ─→ 前车右转指示      *
;*               ││   P1.6 │7          34│ P0.5 ─→ 后车超车指示      *
;*               ││   P1.7 │8          33│ P0.6 ─→ 前车刹车指示      *
;*               ││   RST  │9          32│ P0.7                    *
;*               ││   P3.0 │10 51单片机 31│ EA       VDD            *
;*               ││   P3.1 │11         30│ ALE                     *
;*       INTEX0  │└─→ P3.2 │12         29│ PSEN                    *
;*       INTEX1  └──→ P3.3 │13         28│ P2.7                    *
;*                    P3.4 │14         27│ P2.6                    *
;*   remoteout←────── P3.5 │15         26│ P2.5                    *
;*                    P3.6 │16         25│ P2.4                    *
;*                    P3.7 │17         24│ P2.3                    *
;*                   XTAL2 │18         23│ P2.2                    *
;*                   XTAL1 │19         22│ P2.1                    *
;*                     Vss │20         21│ P2.0                    *
;*                                                                 *
;*****************************************************
```

```
;
;                                    接发器控制程序
;*****************************************************
;*                                                   *
;*                    主程序和中断程序入口           *
;*                                                   *
;*****************************************************
    ORG         0000H                   ;程序开始地址
                LJMP    START           ;转 START
    ORG         0003H                   ;外中断 0 中断入口
                LJMP    INTEX0          ;转 INTEX0
    ORG         000BH                   ;定时器 T0 中断入口
                RETI                    ;返回
    ORG         0013H                   ;外中断 1 入口地址
                LJMP    INTEX1          ;转 INTEX1
    ORG         001BH                   ;定时器 T1 中断入口
                LJMP    INTT1           ;转 INTT1
    ORG         0023H                   ;串行口中断入口
                RETI                    ;中断返回
    ORG         002BH                   ;定时器 T2 中断入口
                RETI                    ;中断返回
;
;*****************************************************
;*                                                   *
;*                  初始化程序                       *
;*                                                   *
;*****************************************************
CLEARMEMIO:  CLR        A               ;清 A
             DEC        A               ;A 为#0FFH
             MOV        P1,A            ;端口置 1
             MOV        P2,A            ;端口置 1
             MOV        P3,A            ;端口置 1
             CLR        P3.5            ;关遥控输出
CLEARMEM:    MOV        SP,#70H         ;设堆栈基址为 70H
             MOV        IE,#00H         ;关所有中断
             MOV        IP,#01H         ;外中断 0 为高优先级
             MOV        TMOD,#22H       ;8 位自动重装初值定时器
             MOV        TH1,#0F3H       ;置 13 μs 定时器初值
             MOV        TL1,#0F3H       ;
             SETB       EX0             ;允许外中断 0 中断
             SETB       EX1             ;允许外中断 1 中断
             CLR        ET1             ;关定时器 T1 中断
             SETB       EA              ;开总中断允许
```

```
                RET                         ;子程序返回
;
;*************************************************
;*                                           *
;*                  主 程 序                 *
;*                                           *
;*************************************************
START:          LCALL   CLEARMEMIO          ;上电初始化
;
MAIN:           LJMP    KEYWORK             ;跳到查键程序
                NOP                         ;PC 值出错处理
                NOP                         ;
                LJMP    START               ;重新初始化启动
;
;*************************************************
;*                                           *
;*              T1 中断服务程序              *
;*                                           *
;*************************************************
INTT1:          CPL     P3.5                ;产生 40 kHz 信号,用作红外线发射
                RETI                        ;中断返回
;
;
;*************************************************
;*                                           *
;*              扫键程序(主程序)             *
;*                                           *
;*************************************************
;
KEYWORK:        SETB    P0.2                ;置输入状态
                SETB    P0.0                ;置输入状态
                SETB    P0.1                ;置输入状态
                CLR     00H                 ;清 00H 刹车标志位
                JNB     P0.2,KEY3           ;查刹车输入,为 0 转 KEY3
KEY4:           JNB     P0.0,KEY0           ;查左转输入,为 0 转 KEY0
                JNB     P0.1,KEY1           ;查右转输入,为 0 转 KEY1
KEY5:           JB      00H,KEY2            ;标志为 1 转 KEY2
                SETB    EA                  ;标志为 0,开总中断允许
                SETB    EX1                 ;开外中断 1
                SETB    EX0                 ;开外中断 0
                LCALL   DL10MS              ;延时 60 ms
                LCALL   DL10MS              ;
                LCALL   DL10MS              ;
```

```
                LCALL   DL10MS          ;
                LCALL   DL10MS          ;
                LCALL   DL10MS          ;
KEY6:           SETB    P0.3            ;关前车左转提示输出
                SETB    P0.4            ;关前车右转提示输出
                SETB    P0.5            ;关后车超车提示输出
                SETB    P0.6            ;关前车刹车提示输出
                LJMP    KEYWORK         ;跳回 KEYWORK 循环
;
KEY3:           LCALL   DELAY           ;延时消抖动
                JB      P0.2,KEY5       ;是干扰转 KEY5
                SETB    00H             ;刹车标志置 1
                LJMP    KEY4            ;转 KEY4 查左右转弯按键
;
KEY0:           LCALL   DELAY           ;延时消抖动
                JB      P0.0,KEY5       ;是干扰转 KEY5
                MOV     A,#02H          ;发 2 个脉冲
                LJMP    REMOTE          ;跳到发射程序
;
KEY1:           LCALL   DELAY           ;延时消抖动
                JB      P0.1,KEY5       ;是干扰转 KEY5
                MOV     A,#05H          ;发 5 个脉冲
                LJMP    REMOTE          ;跳到发射程序
KEY2:           CLR     00H             ;清 00H 标志
                MOV     A,#08H          ;发 8 个脉冲
                LJMP    REMOTE          ;跳到发射程序

;*************************************
;*          前车信息接收程序(外中断 0)          *
;*************************************
;从 P1.1 口接收脉冲
INTEX0:         PUSH    ACC             ;现场保护
                PUSH    PSW             ;
                CLR     EX0             ;关外中断 0
                CLR     EX1             ;关外中断 1
                CLR     EA              ;关总中断允许
                JNB     P1.1,READ1      ;P1.1 为 0 转 READ1
READOUT0:       POP     PSW             ;是干扰,中断退出
                POP     ACC             ;
                RETI    ;
;
READ1:          CLR     A               ;清 A
                MOV     DPH,A           ;清 DPTR
```

```
        MOV     DPL,A           ;
HARD1:  JB      P1.1,HARD11     ;8×255=2.04 ms,>2.04 ms 判定是起始位
        INC     DPTR            ;低电平计数(周期为 8 μs)
        NOP                     ;
        NOP
        AJMP    HARD1           ;低电平循环计数
HARD11: MOV     A,DPH           ;
        JZ      READOUT0        ;高 8 位为 0,小于 2.04 ms,退出
        CLR     A               ;>2.04 ms 判定是起始位
READ11: INC     A               ;脉冲数加 1
READ12: JNB     P1.1,READ12     ;低电平等待
        MOV     R1,#0AH         ;高电平脉宽判断用
READ13: JNB     P1.1,READ11     ;变低电平转 READ11
        LCALL   DELAY           ;延时 512 μs
        DJNZ    R1,READ13       ;延时小于 10 次转 READ13 循环
        DEC     A               ;高电平宽大于 5 ms 停止接收,A 减 1
        DEC     A               ;再减 1
        JZ      FLT             ;是 2 个脉冲,执行 FLT
        DEC     A               ;
        DEC     A               ;
        DEC     A               ;
        JZ      FRT             ;是 5 个脉冲,执行 FRT
        DEC     A               ;
        DEC     A               ;
        DEC     A               ;
        JZ      STOP            ;是 8 个脉冲,执行 STOP
        CLR     PX0             ;外中断 0 置低优先级
        SETB    PX1             ;外中断 1 置高优先级
        LJMP    READOUT0        ;转中断退出
;
;*************************************
;*          后车信息接收程序(外中断 1)          *
;*************************************
;接收程序原理同外中断 0,从 P1.0 口接收脉冲
INTEX1: PUSH    ACC             ;
        PUSH    PSW             ;
        CLR     EX1             ;
        CLR     EX0             ;
        CLR     EA              ;
        JNB     P1.0,READ2      ;
READOUT1: POP   PSW             ;
        POP     ACC             ;
        RETI    ;
```

```
READ2:    CLR     A                ;
          MOV     DPH,A            ;
          MOV     DPL,A            ;
HARD2:    JB      P1.0,HARD21      ;
          INC     DPTR             ;
          NOP                      ;
          NOP                      ;
          AJMP    HARD2            ;
HARD21:   MOV     A,DPH            ;
          JZ      READOUT1         ;
          CLR     A                ;
READ21:   INC     A                ;
READ22:   JNB     P1.0,READ22      ;
          MOV     R1,#0AH          ;
READ23:   JNB     P1.0,READ21      ;
          LCALL   DELAY            ;
          DJNZ    R1,READ23        ;
          DEC     A                ;
          DEC     A                ;
          JZ      BLT              ;是 2 个脉冲,后车超车转 BLT
          CLR     PX1              ;外中断 0 与外中断 1 交换中断优先级
          SETB    PX0              ;
          LJMP    READOUT1         ;转中断退出
;
FLT:      CLR     P0.3             ;前车左转弯,P0.3 为 0
          LJMP    READOUT0         ;转外中断 0 中断退出
FRT:      CLR     P0.4             ;前车右转弯,P0.4 为 0
          LJMP    READOUT0         ;转外中断 0 中断退出
BLT:      CLR     P0.5             ;后车超车,P0.5 为 0
          LJMP    READOUT1         ;转外中断 1 中断退出
STOP:     CLR     P0.6             ;前车刹车,P0.6 为 0
          LJMP    READOUT0         ;转外中断 0 中断退出
;
;*************************************
;*                                   *
;*          载波发送程序             *
;*                                   *
;*************************************
;A 中数据为发射的脉冲个数
REMOTE:   CLR     EX0              ;关外中断 0
          CLR     EX1              ;关外中断 1
          MOV     R1,A             ;发射脉冲个数入 R1
          LJMP    OUT3             ;第一位脉冲处理
```

```
OUT:    MOV     R0,#02H         ;1 ms 脉冲控制
OUT1:   SETB    ET1             ;开定时 T1 中断(40 kHz 红外调制用)
        SETB    TR1             ;开启 T1
        LCALL   DELAY           ;延时 513 μs  6(2)×0.5=3 ms(1 ms)
        DJNZ    R0,OUT1         ;总延时值不到转 OUT1 再循环
        MOV     R0,#02H         ;赋 1 ms 脉宽定时值
OUT2:   CLR     TR1             ;关 T1
        CLR     ET1             ;关 T1 中断
        CLR     P3.5            ;关红外线输出
        LCALL   DELAY           ;延时 513 μs
        DJNZ    R0,OUT2         ;总延时(1 ms)不到转 OUT2 再延时
        DJNZ    R1,OUT          ;脉冲未发完,转 OUT 再发
        LCALL   DL10MS          ;脉冲发完延时 10 ms(帧间隔)
        LJMP    KEY5            ;脉冲发送结束跳到 KEY5
OUT3:   MOV     R0,#06H         ;3 ms 脉宽控制
        LJMP    OUT1            ;转 OUT1 红外线发射
;
;
;*****************************************
;*                                       *
;*          延时 249 μs×2+3= 501 μs      *
;*                                       *
;*****************************************
DELAY:  MOV     R7,#0F9H
DELAY1: DJNZ    R7,DELAY1
        RET
;
;*****************************************
;*                                       *
;*              延时 10 ms               *
;*                                       *
;*****************************************
DL10MS: MOV     R6,#14H
DL10MS1:LCALL   DELAY
        DJNZ    R6,DL10MS1
        RET
;
;
        END                             ;程序结束
```

第 16 章　实例 11　数控调频发射台的设计

本数控调频发射台可在 80.0～109.9 MHz 范围内任意设置发射频率，可预置 11 个频道，发射频率最小调整值为 0.1 MHz，具有单声道/立体声控制可广泛应用于学校无线广播、电视现场导播、汽车航行、无线演说等场所。

1. 系统硬件电路的设计

(1) 单片机控制部分

单片机采用 AT89C52，其采用最小化应用系统设计。P0 口和 P2 口作为共阳 LED 数码管驱动用。P1 口作为 16 键的键盘接口，其中 T0～T3 分别为百位、十位、个位、小数位的频率操作键。百位数只能是 0 或 1。当百位数为 0 时，十位数为 8 或 9；当百位数为 1 时，十位数只能为 0。个位及小数位为 0～9 之中的任意数。T4～T14 为发射频率预置键，T15 为单声道/立体声控制键。P3.0、P3.1 和 P3.2 作为与 BH1415 的通信端口，用于传送发射频率控制数据，P3.3 用于立体声发射批示。本电路采用 12 MHz 晶振，模拟串口通信。单片机控制部分电路如图 16.1 所示。

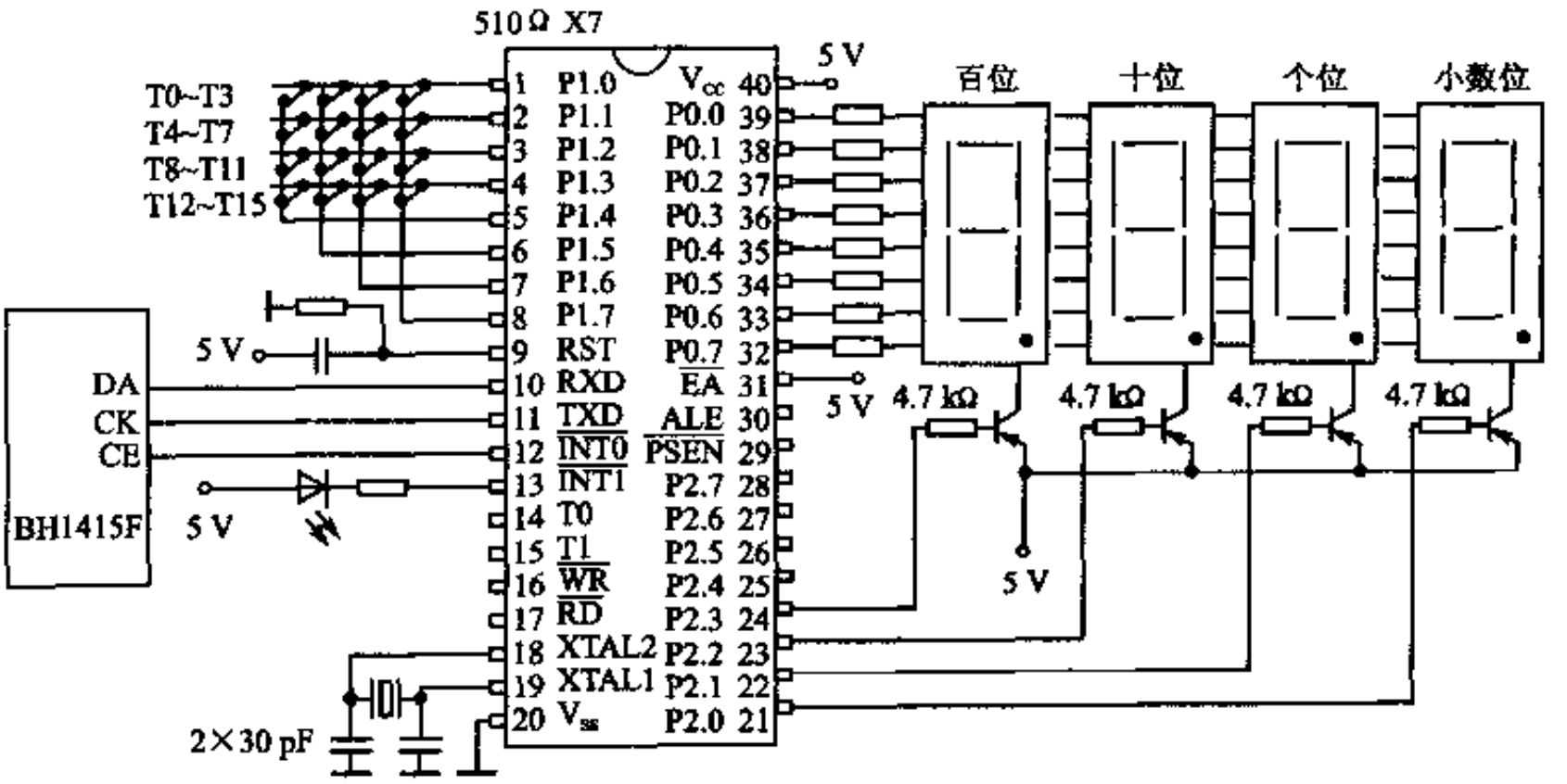


图 16.1　单片机控制电原理图

(2) 调频调制发射部分

本系统调频调制发射部分采用 Rohm 公司最新生产的调频发射专用集成电路 BH1415F，内含立体声信号调制、调频广播信号发射电路。BH1415F 内有前置补偿电路、限制器电路和低通滤波电路等，因此，系统具有良好的音色，内置 PLL 系统调频发射电路，传输频率非常稳定。调频发射频率可用单片机通过串行口直接控制。BH1415F 各引脚的功能如表 16.1 所列，应用电路如图 16.2 所示。从 11 脚输出的调频调制信号，经高频放大后由天线发射输出，后级高频放大器的功率可根据接收的距离范围考虑。

表 16.1　BH1415F 引脚功能表

引　脚	脚描述	直流 /V
1	右声道输入端:通过电容器与右声道音频信号相连	1/2Vcc
2	左音源输入端:通过电容器与左声道音频信号相连	
2,21	时间常数端:其连接一个电容,时间常数:$\tau \approx 22.7\ k\Omega F$	
3,20	LPF 时间常数端:其连接一个 150 pF 电容,时间常数为 15 kHz LPF	$1/2V_{cc}$
4	滤波器端:声频部分滤波器参考电压	$1/2V_{cc}$
5	立体声复合信号输出端:连接到调频调制器	$1/2V_{cc}$
6	接地端	GND
7	PLL 相位检波器输出端:连接到 PLL LPF 电路	—
8	电源供给端	V_{cc}
9	射频振荡器端:连接振荡时间常数,是振荡器基端	$4/7V_{cc}$
10	射频地端	GND
11	射频发送输出端	$V_{cc}-1.9$
12	PLL 电源供给端	V_{cc}
13,14	X'tal 振荡器端:连接一个 7.6 MHz 晶振	—
15	芯片授权端:连续输入高电平数据	
16	时钟输入端:带数据和同步的时钟在序列数据输入	
17	数据输入端	
18	静音端:$0.8V_{cc} \leqslant$ Pin18:Mute ON,$0.2V_{cc} \geqslant$ Pin18:Mute OFF	
19	控制信号调节端	$1/2V_{cc}$

BH1415F 的频率控制码为 16 位,其数据传送格式如图 16.3 所示,其中 D0～D10 为频率控制数据,其值乘 0.1 即为 BH1415F 的输出频率(单位:MHz);D11～D15 为控制位。D11(MONO)为单声道/立体声控制位,0 时为单声道发射模式,1 时为立体声发射模式。D12(PD0)和 D13(PD1)位用于相位控制,通常为 0,当分别为 01 或 10 时可使发射频率在最低和最高处。D14(T0)和 D15(T1)位用于测试模式控制,通常为 00,当为 10 时为测试模式。

(3) 电源系统

由于采用单片机控制的数字调频台功耗很小,可用 7805 三端稳压块分别对单片机和 BH1415F 电路单独供电,电源变压器功率大于 10 W 即可。集成块电源脚应就近接 0.1 μF 的瓷片电容。

2. 内存单元的使用要求

26H～29H 用来存放显示小数位、个位、十位、百位的 BCD 码数据。24H～25H 用来存放频率控制数据(十六进制)。21H 用来存放频率控制字节低 8 位数据。22H 用来存放频率控制字节高 8 位数据。23H 用来存放键扫描时 P1 端口的值。

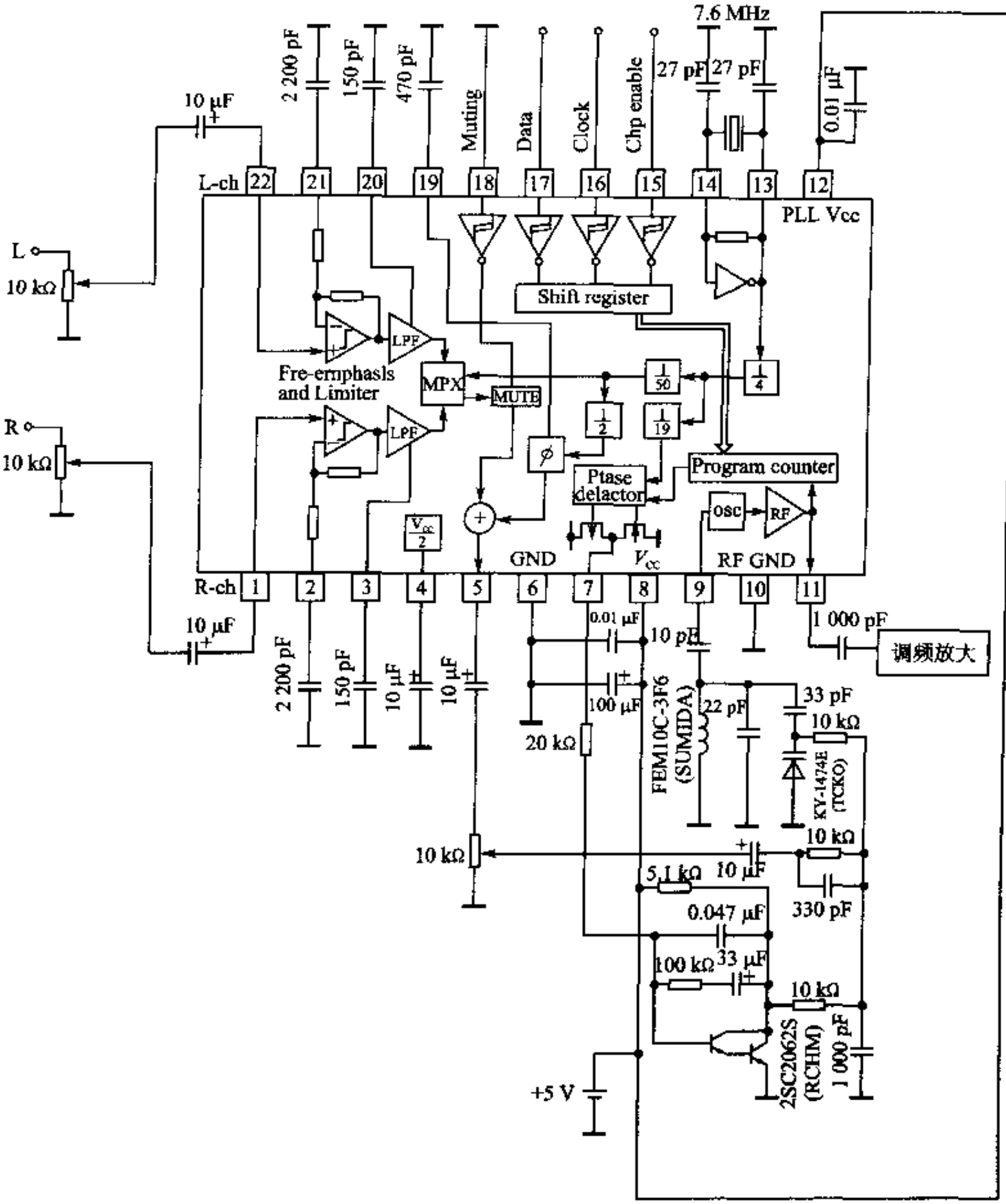


图 16.2　BH1415F 电原理图

3. 系统主要程序的设计

(1) 键盘扫描程序

本程序采用 4×4 行列式查询法，其方法是对 P1.0～P1.3 行线口分别置 0，然后读入 P1 口高 4 位的值。若不为 1111 则说明有键按下，根据读入的 P1 口值与键号表进行查表对照，从而取得按键的键号值。键盘扫描程序流程图如图 16.4 所示。

(2) 显示程序

本程序采用动态扫描法显示 4 位频率数字值。

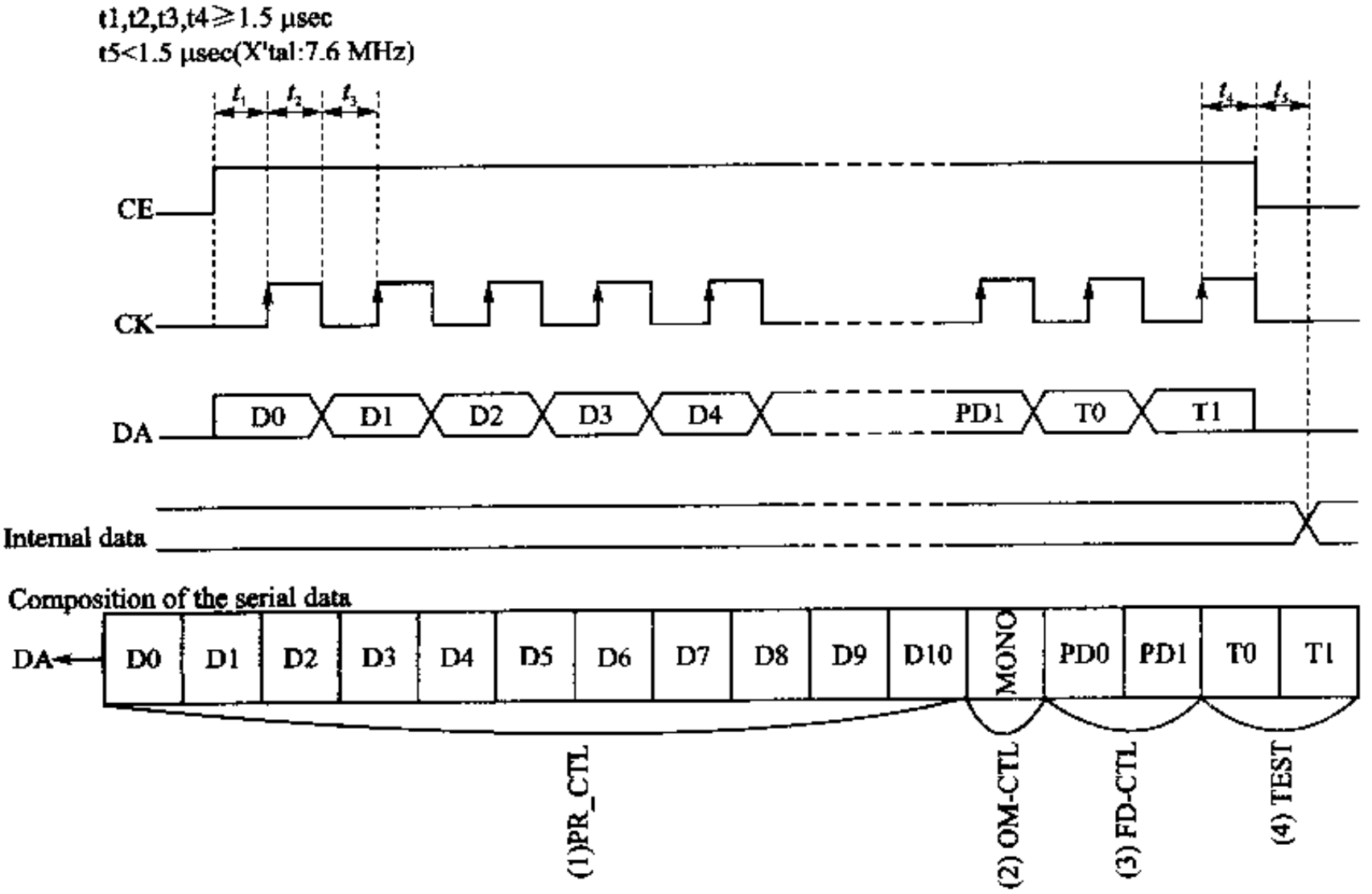


图 16.3　BH1415F 的数据传送格式

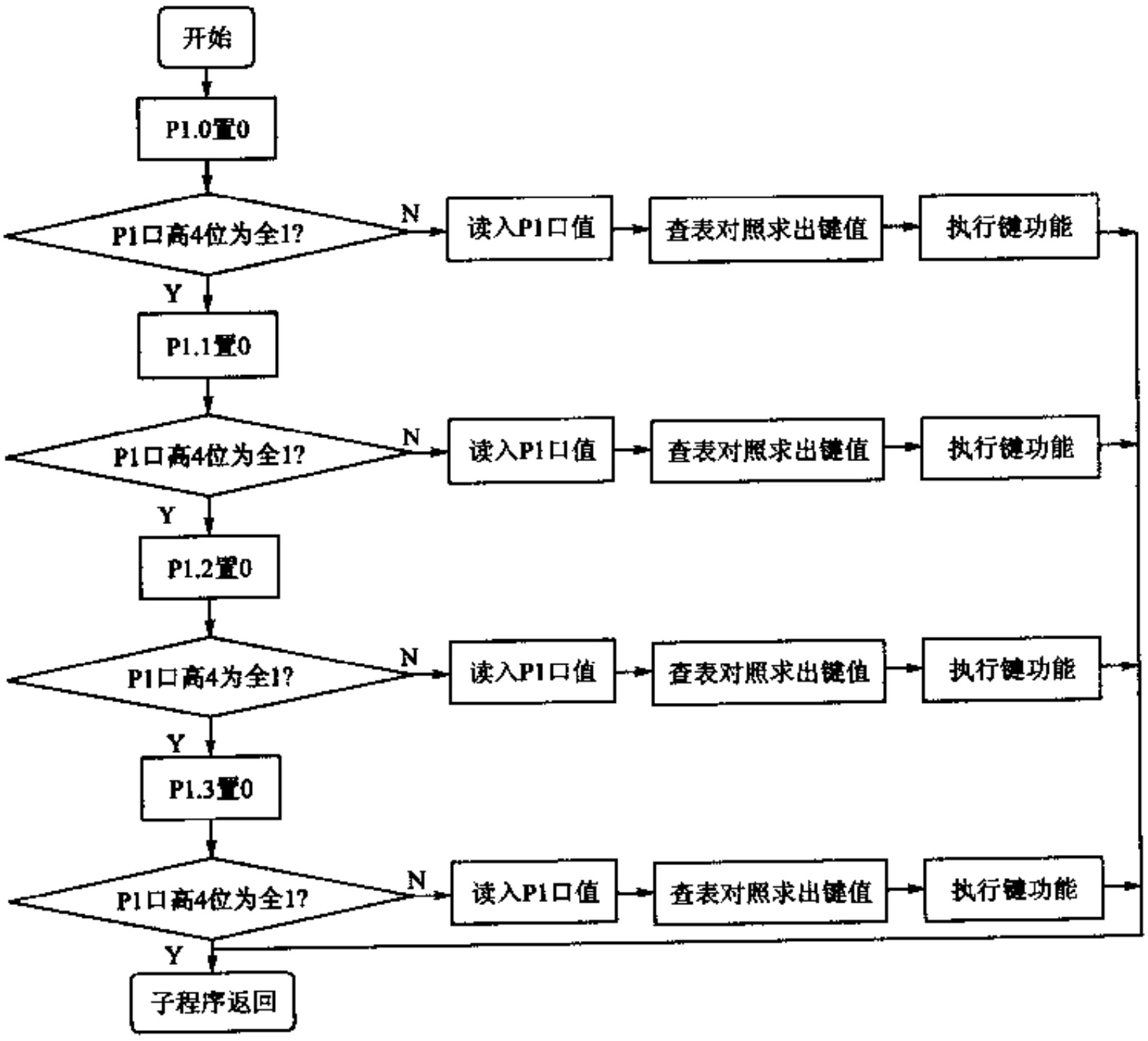
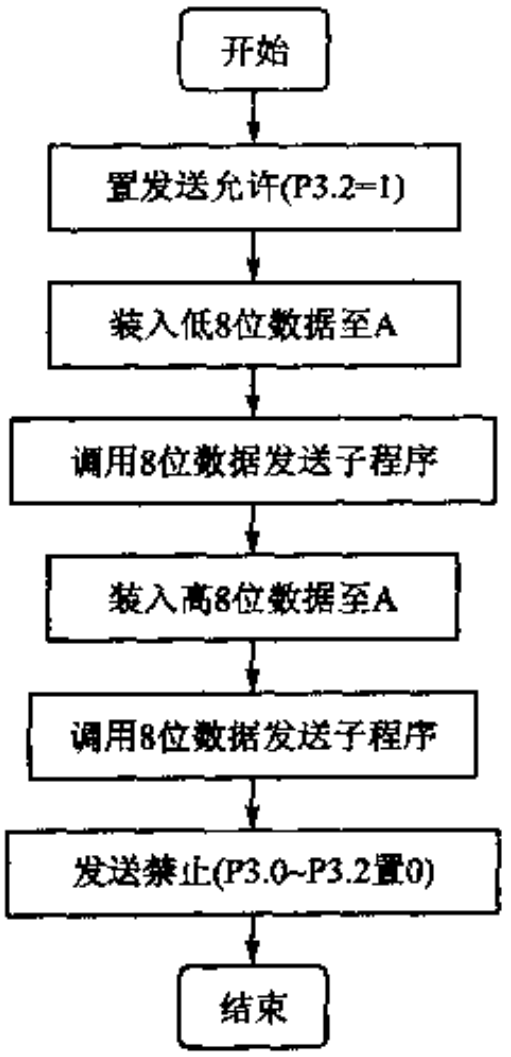


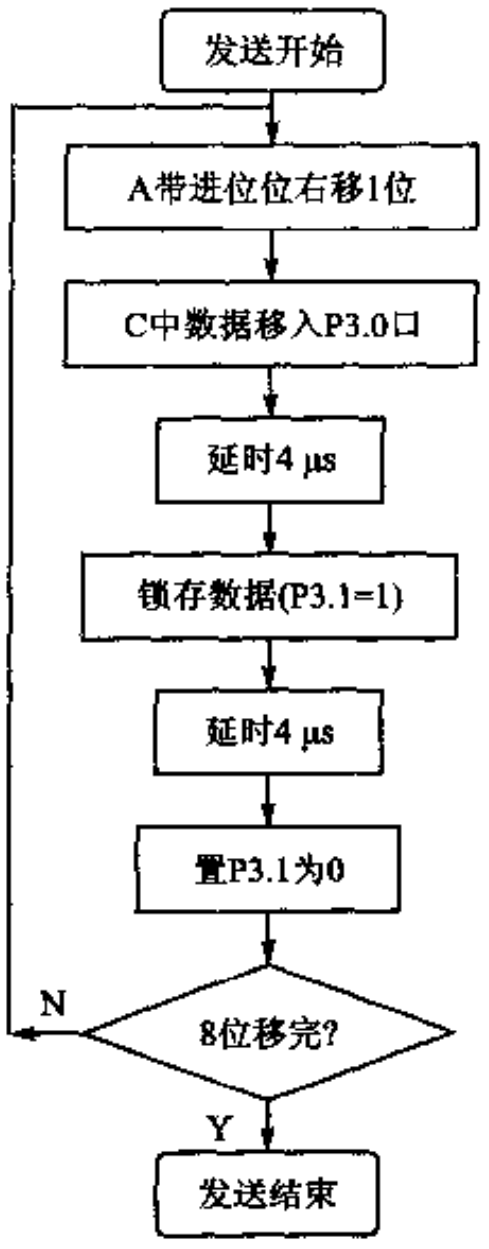
图 16.4　4×4 行列式 16 键扫描程序流程图

(3) 串行通信程序

本程序由十进制 BCD 码转十六进制程序、16 位频率控制字节合成程序和模拟异步串行发送程序组成。模拟异步串行发送程序是根据 BH1415F 的传送要求编写的，其发送子程序流程图如图 16.5 所示。



(a) 16位频率数据发送程序流程



(b) 8位数据发送子程序流程图

图 16.5　频率数据发送程序流程图

以下是数控调频台控制器完整源程序：

```
;*****************************;
;          数控调频台控制器          ;
;*****************************;

;
;26H~29H 存放显示小数位、个位、十位、百位 BCD 码数，24H~25H 存放频率控制数据(十六进制)
;
CONBITL   EQU   21H          ;频率控制字节低 8 位
CONBITH   EQU   22H          ;频率控制字节高 8 位
KEYWORD EQU   23H            ;存放键扫描时 P1 口值
;
;
ORG       0000H              ;程序开始地址
LJMP      START              ;转 START 执行
```

```
        ORG     0003H           ;
        RETI                    ;不用中断程序
        ORG     000BH           ;
        RETI                    ;不用中断程序
        ORG     0013H           ;
        RETI                    ;不用中断程序
        ORG     001BH           ;
        RETI                    ;不用中断程序
        ORG     0023H           ;
        RETI                    ;不用中断程序
        ORG     002BH           ;
        RETI                    ;不用中断程序
;
;初始化程序
CLEARMEN:  MOV     R0,#20H          ;20H～29H 循环清 0
        MOV     R1,#0AH          ;
CLEARLOOP: MOV     @R0,#00H         ;
        INC     R0               ;
        DJNZ    R1,CLEARLOOP     ;
        MOV     P0,#0FFH         ;4 端口置 1
        MOV     P1,#0FFH         ;
        MOV     P2,#0FFH         ;
        MOV     P3,#0FFH         ;
        CLR     P3.0             ;BH1415F 禁止操作
        CLR     P3.1             ;
        CLR     P3.2             ;
        LCALL   KEYFUN15         ;置立体声发射方式,开立体声发射指示灯
CLEAR1:    MOV     PCON,#00H        ;控制寄存器清 0
        MOV     29H,#00H         ;置初始值为 88 MHz(显示为 088.0)
        MOV     28H,#08H         ;
        MOV     27H,#08H         ;
        MOV     26H,#00H         ;
        LCALL   DISPUPDAT        ;写入 BH1415F 芯片(修改发送频率)
        RET                      ;子程序返回
;
;主程序
START:  LCALL   CLEARMEN         ;上电初始化
MAIN:   LCALL   KEYWORK          ;调查键子程序
        LCALL   DISPLAY          ;LED 显示一次
        AJMP    MAIN             ;转 MAIN 循环
        NOP                      ;PC 出错处理
        NOP                      ;
        AJMP    START            ;重新初始化
```

```
;
;4×4 行列扫描查键子程序
KEYWORK:  MOV     P1,#0FFH          ;置 P1 口为输入状态
          CLR     P1.0              ;扫描第 1 行(第 1 行为 0)
          MOV     A,P1              ;读入 P1 口值
          ANL     A,#0F0H           ;低 4 位为 0
          CJNE    A,#0F0H,KEYCON    ;高 4 位不为全 1(有键按下)转 KEYCOON
          SETB    P1.0              ;扫描第 2 行(第 2 行为 0)
          CLR     P1.1              ;
          MOV     A,P1              ;读入 P1 口值
          ANL     A,#0F0H           ;低 4 位为 0
          CJNE    A,#0F0H,KEYCON    ;高 4 位不为全 1(有键按下)转 KEYCOON
          SETB    P1.1              ;扫描第 3 行(第 3 行为 0)
          CLR     P1.2              ;
          MOV     A,P1              ;读入 P1 口值
          ANL     A,#0F0H           ;低 4 位为 0
          CJNE    A,#0F0H,KEYCON    ;高 4 位不为全 1(有键按下)转 KEYCOON
          SETB    P1.2              ;扫描第 4 行(第 4 行为 0)
          CLR     P1.3              ;
          MOV     A,P1              ;读入 P1 口值
          ANL     A,#0F0H           ;低 4 位为 0
          CJNE    A,#0F0H,KEYCON    ;高 4 位不为全 1(有键按下)转 KEYCOON
          SETB    P1.3              ;结束行扫描
          RET                       ;子程序返回
KEYCON:   LCALL   DL10MS            ;消抖处理
          MOV     A,P1              ;再读入 P1 口值
          ANL     A,#0F0H           ;低 4 位为 0
          CJNE    A,#0F0H,KEYCHE    ;高 4 位不为全 1,确有键按下,转 KEYCHE
KEYOUT:   RET                       ;干扰,子程序返回
KEYCHE:   MOV     A,P1              ;读 P1 口值
          MOV     KEYWORD,A         ;放入 23H 暂存
CJLOOP:   LCALL   DISPLAY           ;调显示子程序
          MOV     A,P1              ;读 P1 口值
          ANL     A,#0F0H           ;低 4 位为 0
          CJNE    A,#0F0H,CJLOOP    ;高 4 位为全 1(键还按着),转 CJLOOP 等待释放
          MOV     R7,#00H           ;键释放,置 R7 初值为#00H(查表次数)
          MOV     DPTR,#KEYTAB      ;取键值表首址
CHEKEYLOOP: MOV   A,R7              ;查表次数入 A
          MOVC    A,@A+DPTR         ;查表
          XRL     A,KEYWORD         ;查表值与 P1 口读入值比较
          JZ      KEYOK             ;为 0(相等)转 KEYOK
          INC     R7                ;不等,查表次数加 1
          CJNE    R7,#10H,CHEKEYLOOP
```

```
                                                ;查表次数不超过 16 次转 CHEKEYLOOP 再查
                RET                             ;16 次到,退出
;
KEYOK:          MOV     A,R7                    ;查表次数入 A(即键号值)
                MOV     B,A                     ;放入 B
                RL      A                       ;左移
                ADD     A,B                     ;相加(键号乘 3 处理 JMP 3 字节指令)
                MOV     DPTR,#KEYFUNTAB         ;取键功能散转表首址
                JMP     @A+DPTR                 ;查表
KEYFUNTAB:      LJMP    KEYFUN00                ;键功能散转表。跳至 0 号键功能程序
                LJMP    KEYFUN01                ;跳至 01 号键功能程序
                LJMP    KEYFUN02                ;跳至 02 号键功能程序
                LJMP    KEYFUN03
                LJMP    KEYFUN04
                LJMP    KEYFUN05
                LJMP    KEYFUN06
                LJMP    KEYFUN07
                LJMP    KEYFUN08
                LJMP    KEYFUN09
                LJMP    KEYFUN10
                LJMP    KEYFUN11
                LJMP    KEYFUN12
                LJMP    KEYFUN13
                LJMP    KEYFUN14
                LJMP    KEYFUN15                ;跳至 15 号键功能程序
                RET                             ;散转出错返回
;
;键号对应 P1 口数值表(同时按下两键为无效操作)
KEYTAB:         DB      0EEH,0DEH,0BEH,7EH,0EDH,0DDH,0BDH,7DH
                DB      0EBH,0DBH,0BBH,7BH,0E7H,0D7H,0B7H,77H,0FFH,0FFH
;
;0 号键功能程序
KEYFUN00:       INC     29H                     ;百位数加 1
                MOV     A,29H                   ;入 A
                CLR     C                       ;清进位标志
                CJNE    A,#02H,FUN00            ;
FUN00:          JC      FUN00OUT                ;百位小于 2 转 FUNOO0UT
                MOV     29H,#00H                ;大于等于 2 清为 0(百位只能是 0 或 1)
FUN00OUT:       MOV     A,29H                   ;判断百位是 0 还是 1
                XRL     A,#01H                  ;
                JNZ     F00OUT1                 ;若百位为 0 转 FOOOUT1
                MOV     28H,#00H                ;若百位为 1,十位为 0
                AJMP    F00OUT                  ;
```

```
F00OUT1:    MOV     28H,#08H            ;若百位为 0,十位数改为 8
F00OUT:     LCALL   DISPUPDAT           ;写入控制芯片(修改发射频率)
            RET                         ;返回
;
;01 号键功能程序
KEYFUN01:   INC     28H                 ;十位数加 1
            MOV     A,28H               ;入 A
            CLR     C                   ;清进位标志
            CJNE    A,#0AH,FUN01        ;判断是否小于 10
FUN01:      JC      FUN01OUT            ;十位数小于 10 转 FUN01OUT
            MOV     28H,#00H            ;十位数大于或等于 10 清为 0
FUN01OUT:   MOV     A,29H               ;判断百位数是 0 不是 1
            XRL     A,#01H              ;
            JNZ     F01OUT              ;
            MOV     28H,#00H            ;百位数为 1 时,十位数为 0
            AJMP    F001OUT             ;
F01OUT:     MOV     A,28H               ;百位为 0 时,十位数只能是 8 或 9
            XRL     A,#08H              ;判断是不是 8
            JZ      F001OUT             ;十位数是 8 转 F001OUT
            MOV     A,28H               ;
            XRL     A,#09H              ;判断是不是 9
            JZ      F001OUT             ;十位数是 9 转 F001OUT
            MOV     28H,#08H            ;不是 8 也不是 9,十位赋值为 8
F001OUT:    LCALL   DISPUPDAT           ;写入控制芯片(修改发射频率)
            RET                         ;返回
;
;02 号键功能程序
KEYFUN02:   INC     27H                 ;个位数加 1
            MOV     A,27H               ;
            CLR     C                   ;
            CJNE    A,#0AH,FUN02        ;判断是否小于 10
FUN02:      JC      FUN02OUT            ;小于 10 转 FUN02OUT
            MOV     27H,#00H            ;大于或等于 10 清为 0
FUN02OUT:   LCALL   DISPUPDAT           ;写入控制芯片(修改发射频率)
            RET                         ;
;
;03 号键功能程序
KEYFUN03:   INC     26H                 ;个位数加 1
            MOV     A,26H               ;
            CLR     C                   ;
            CJNE    A,#0AH,FUN03        ;判断是否小于 10
FUN03:      JC      FUN03OUT            ;小于 10 转 FUN03OUT
            MOV     26H,#00H            ;大于或等于 10 清为 0
```

```
FUN03OUT:   LCALL   DISPUPDAT          ;写入控制芯片(修改发射频率)
            RET                        ;返回
;
;04 号键功能程序(频率预置键)
KEYFUN04:   MOV     29H,#01H           ;预置 109.0 MHz 发射频率
            MOV     28H,#00H
            MOV     27H,#09H
            MOV     26H,#00H
            LCALL   DISPUPDAT          ;写入控制芯片(修改发射频率)
            RET
;
;05 号键功能程序(频率预置键)
KEYFUN05:   MOV     29H,#01H           ;预置 108.0 MHz 发射频率
            MOV     28H,#00H
            MOV     27H,#08H
            MOV     26H,#00H
            LCALL   DISPUPDAT          ;写入控制芯片(修改发射频率)
            RET
;
;06 号键功能程序(频率预置键)
KEYFUN06:   MOV     29H,#01H           ;预置 105.0 MHz 发射频率
            MOV     28H,#00H
            MOV     27H,#05H
            MOV     26H,#00H
            LCALL   DISPUPDAT          ;写入控制芯片(修改发射频率)
            RET
;
;07 号键功能程序(频率预置键)
KEYFUN07:   MOV     29H,#01H           ;预置 100.0 MHz 发射频率
            MOV     28H,#00H
            MOV     27H,#00H
            MOV     26H,#00H
            LCALL   DISPUPDAT          ;写入控制芯片(修改发射频率)
            RET
;
;08 号键功能程序(频率预置键)
KEYFUN08:   MOV     29H,#00H           ;预置 98.0 MHz 发射频率
            MOV     28H,#09H
            MOV     27H,#08H
            MOV     26H,#00H
            LCALL   DISPUPDAT          ;写入控制芯片(修改发射频率)
            RET
;
```

```
;09 号键功能程序(频率预置键)
KEYFUN09:   MOV     29H,#00H        ;预置 96.0 MHz 发射频率
            MOV     28H,#09H
            MOV     27H,#06H
            MOV     26H,#00H
            LCALL   DISPUPDAT       ;写入控制芯片(修改发射频率)
            RET
;
;10 号键功能程序(频率预置键)
KEYFUN10:   MOV     29H,#00H        ;预置 94.0 MHz 发射频率
            MOV     28H,#09H
            MOV     27H,#04H
            MOV     26H,#00H
            LCALL   DISPUPDAT       ;写入控制芯片(修改发射频率)
            RET
;
;11 号键功能程序(频率预置键)
KEYFUN11:   MOV     29H,#00H        ;预置 92.0 MHz 发射频率
            MOV     28H,#09H
            MOV     27H,#02H
            MOV     26H,#00H
            LCALL   DISPUPDAT       ;写入控制芯片(修改发射频率)
            RET
;
;12 号键功能程序(频率预置键)
KEYFUN12:   MOV     29H,#00H        ;预置 90.0 MHz 发射频率
            MOV     28H,#09H
            MOV     27H,#00H
            MOV     26H,#00H
            LCALL   DISPUPDAT       ;写入控制芯片(修改发射频率)
            RET
;
;13 号键功能程序(频率预置键)
KEYFUN13:   MOV     29H,#00H        ;预置 88.0 MHz 发射频率
            MOV     28H,#08H
            MOV     27H,#08H
            MOV     26H,#00H
            LCALL   DISPUPDAT       ;写入控制芯片(修改发射频率)
            RET
;
;14 号键功能程序(频率预置键)                ;预置 87.0 MHz 发射频率
KEYFUN14:   MOV     29H,#00H
            MOV     28H,#08H
```

```
        MOV     27H,#07H
        MOV     26H,#08H
        LCALL   DISPUPDAT           ;写入控制芯片(修改发射频率)
        RET
;
;15 号键功能程序(立体声/单声道设置键)
KEYFUN15:   CPL     03H             ;立体/单声标志取反
        JNB     03H,MONO            ;为 0 转单声道 MONO
        CLR     P3.3                ;为 1 开立体声指示灯
        LCALL   PUTBIT              ;发送控制字至 BH1415F
        RET                         ;返回
MONO:   SETB    P3.3                ;关立体声指示灯
        LCALL   PUTBIT              ;发控制字至 BH1415F
        RET                         ;返回
;
;将 BCD 码转为十六进制数,与 5 位控制码合成操作码,写入控制芯片
DISPUPDAT:  LCALL   BCDB            ;调 BCD 码转为十六进制数程序
        LCALL   CONCOMMAND          ;调 5 位控制码合成操作码程序
        LCALL   PUTBIT              ;发控制字至 BH1415F
        RET                         ;返回
;
;将 BCD 码转为十六进制数程序
BCDB:   MOV     CONBITL,#00H        ;控制字清 0
        MOV     CONBITH,#00H        ;控制字清 0
        MOV     CONBITL,26H         ;小数位数放入控制字低 8 位
        MOV     A,27H               ;个位数乘 10 操作
        MOV     B,#10               ;
        LCALL   MULLOOP             ;调乘法子程序
        MOV     A,28H               ;十位数乘 100 操作
        MOV     B,#100              ;
        LCALL   MULLOOP             ;调乘法子程序
        MOV     A,29H               ;
        JNZ     ADD3E8              ;百位数为 1 转 ADD3E8(加 1000 操作)
        RET                         ;百位数为 0 退出
ADD3E8:     CLR     C               ;清进位档标志
        MOV     A,#0E8H             ;低 8 位加法
        ADD     A,CONBITL           ;累加
        MOV     CONBITL,A           ;放回 CONBITL
        MOV     A,#03H              ;高 8 位加法
        ADDC    A,CONBITH           ;控制字高 8 位处理
        MOV     CONBITH,A           ;放回 CONBITH
        RET                         ;返回
;
```

```
;乘法及累加处理程序(将四位显示的十进制 BCD 码转为 1 个二进制数)
MULLOOP:  MUL    AB              ;乘法
          CLR    C               ;清进位标志
          ADD    A,CONBITL       ;积低 8 位与 CONBITL 相加
          MOV    CONBITL,A       ;放回 CONBITL
          MOV    A,CONBITH       ;
          ADDC   A,B             ;积高 8 位与 CONBITH 带进位累加
          MOV    CONBITH,A       ;放回 CONBITH
          RET                    ;返回
;
;频率控制数据与 5 位控制码合成 BH1415F 控制字
CONCOMMAND: ANL  CONBITH,#07H    ;高 4 位为 0
          MOV    A,20H           ;控制字放入 A
          ORL    A,CONBITH       ;合成控制字
          MOV    CONBITH,A       ;放回 CONBITH
          RET                    ;返回
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;              显示程序                ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;共阳 LED 显示,P0 口输出段码,P2 口输出扫描字
DISPLAY:  MOV    R1,#26H         ;显示首址
          MOV    R5,#0FEH        ;设扫描字
PLAY:     MOV    A,R5            ;放入 A
          MOV    P2,A            ;P2 口输出
          MOV    A,@R1           ;取显示数据
          MOV    DPTR,#TAB       ;取段码表首址
          MOVC   A,@A+DPTR       ;查段码
          MOV    P0,A            ;从 P0 输出
          MOV    A,R5            ;读入扫描字
          JB     ACC.1,PLAY1     ;不是十位(LED),不显示小数点
          CLR    P0.7            ;是十位,显示小数点
PLAY1:    LCALL  DL1MS           ;点亮 1 ms
          INC    R1              ;指向下一显示数据
          JNB    ACC.3,ENDOUT    ;是第 4 位 LED,退出
          RL     A               ;不是,左移一位
          MOV    R5,A            ;放回 R5
          SETB   P0.7            ;关小数点
          AJMP   PLAY            ;转 PLAY 循环
ENDOUT:   MOV    P2,#0FFH        ;显示结束,关显示输出口
          MOV    P0,#0FFH        ;
          RET                    ;返回
;
```

```
;0～9 共阳段码表
TAB: DB   0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H,0FFH,0FFH
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;              发送控制字节子程序        ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
PUTBIT:      MOV      A,CONBITL          ;低 8 位控制字入 A
             SETB     P3.2               ;BH1415F 使能(允许写)
             LCALL    PUT                ;发送 8 位
             MOV      A,CONBITH          ;高 8 位控制字入 A
             LCALL    PUT                ;发送 8 位
             CLR      P3.2               ; BH1415F 写禁止
             CLR      P3.0               ;复位
             CLR      P3.1               ;复位
             RET                         ;返回
;
;字节发送子程序
PUT:         MOV      R3,#8              ;发送 8 位控制
             CLR      C                  ;清 C
PUT1:        RRC      A                  ;带进位位右移(先发低位)
             MOV      P3.0,C             ;低位送至 P3.0 口
             NOP                         ;延时 4 μs
             NOP                         ;
             NOP                         ;
             NOP                         ;
             SETB     P3.1               ;锁存数据(上升沿时锁存数据)
             NOP                         ;延时 4 μs
             NOP                         ;
             NOP                         ;
             NOP                         ;
             CLR      P3.1               ;
             DJNZ     R3,PUT1            ;8 位未发完转 PUT1 再发
             RET                         ;8 位发完结束
;
;513 μs 延时子程序
DL513:       MOV      R3,#0FFH
DL513LOOP:   DJNZ     R3,DL513LOOP
             RET
;
;1 ms 延时子程序(LED 点亮用)
DL1MS:       MOV      R4,#02H
DL1MSLOOP:   LCALL DL513
```

```
                DJNZ        R4,DL1MSLOOP
                RET
;
;10 ms 延时子程序(消抖动用)
DL10MS:         MOV         R6,#0AH
DL10MSLOOP:     LCALL DL1MS
                DJNZ        R6,DL10MSLOOP
                RET
;
;
END                                          ;程序结束
```

第 17 章　实例 12　可在线修改程序的单片机 W78E516B 设计实例

1. W78E516B 的功能特点

W78E516B 是华邦公司 2000 年发布的一种可用于在线编程的 8 位单片机，其指令设置同 8052 标准完全兼容。W78E516B 包含一个 64 KB 的主闪存 EPROM 和一个特有的 4 KB 的附加闪存 EPROM，其中 64 KB EPROM 中的内容可被装在 4 KB 附加 EPROM 中的装载程序更新；具有 512 B 的 RAM；4 个 8 位的双向可位寻址 I/O 端口；1 个附加的 4 位 P4 端口；3 个 16 位的定时/计数器；1 个串行口；6 个向量二级中断结构；2 种可用软件选择的低功耗模式。

2. W78E516B 中的几个特殊寄存器

(1) CHPENR(F6H)：CHPCON 寄存器写操作允许控制，其通常是只读的。当用软件连续写 2 个特殊值 87H 及 59H 到 CHPENR 寄存器时，CHPCON 寄存器即可允许写操作；当写入除 87H 和 59H 以外的数据到 CHPENR 寄存器时，CHPCON 寄存器又恢复只读功能。

(2) CHPCON(BFH)：在线系统编程控制寄存器，其位功能定义如表 17.1 所列。

表 17.1　CHPCON(BFH)寄存器的位功能定义

位	名　称	功　能
7	SWRESET (F04K 模式)	当这位为 1 时，并且 FBOOTSL 和 FPROGEN 同时为 1 时，它将使 CPU 执行复位，重新引导系统。通过读这位逻辑 1 就能确定是否进入 F04KBOOT 模式
6	--	备用
5	—	备用
4	ENAUXRAM	=1 时，允许片内 AUX－RAM =0 时，不允许片内 AUX－RAM
3	0	必须为 0
2	0	必须为 0
1	FBOOTSL	程序安装选择 0：安装程序在 64 KB APROM，4 KB LDROM 禁止重复编程 1：安装程序在 4 KB LDROM，64 KB APROM 禁止重复编程
0	FPROGEN	Flash EPROM 编程允许 =1：允许，在进入空闲模式并通过中断唤醒后，CPU 进入在线系统编程模式。在线系统编程模式其间，擦除操作、编程和读在线设备进入空闲模式时执行 =0：禁止，片内 Flash 存储器只能读，在线系统编程被禁止

(3) SFRAH(C5H)：在线系统编程中，片内 Flash EPROM 实际地址的高 8 位。

(4) SFRAL(C4H)：在线系统编程中，片内 Flash EPROM 实际地址的低 8 位。

(5) SFRFD(C6H)：在线编程模式中用于片内 Flash EPROM 的编程数据。

(6) SFRCN(C7H)：在线 Flash EPROM 编程模式的控制字节，其位功能定义如表 17.2 所列，其工作模式如表 17.3 所列。

表 17.2　SFRCN(C7H)寄存器的位功能定义

位	名　称	功　能
7		备用
6	WFWIN	用于在系统编程中片内 Flash EPROM 空间选择 =0：选择 64 KB Flash EPROM 空间 =1：选择 4 KB Flash EPROM 空间
5	OEN	允许 Flash EPROM 输出
4	CEN	允许 Flash EPROM 集成电路片
3	CTRL3	快闪控制信号
2	CTRL2	
1	CTRL1	
0	CTRL0	

表 17.3　SFRCN 的工作模式

模　式	WFWIN	CTRL (3～0)	OEN	CEN	SFRAH， SFRAL	SFRFD
擦 64 KB APROM	0	0010	1	0	×	×
64 KB APROM 编程	0	0001	1	0	地址输入	数据输入
读 64 KB APROM	0	0000	0	0	地址输入	数据输出
擦 4 KB LDROM	1	0010	1	0	×	×
4K LDROM 编程	1	0001	1	0	地址输入	数据输入
读 4 KB LDROM	1	0000	0	0	地址输入	数据输出

3. W78E516B 在线编程的规则

W78E516B 在线编程流程图如图 17.1、图 17.2 和图 17.3 所示。

4. 应用系统硬件电路的设计

在线编程应用系统的硬件电路如图 17.4 所示。系统可对字符显示器实施在线修改而不必拆动硬件电路；采用串口连接实现计算机与单片机的通信，串口电平转换器采用 MAX232，计算机使用超级终端通信口；传送文件(.bin)采用 xmodem 协议；单片机的晶振频率应低于 40 MHz。由于串口用于在线编程调试，若应用系统需用串口可采用虚拟 I^2C 总线，具体的应用系统硬件可按实现功能设计。

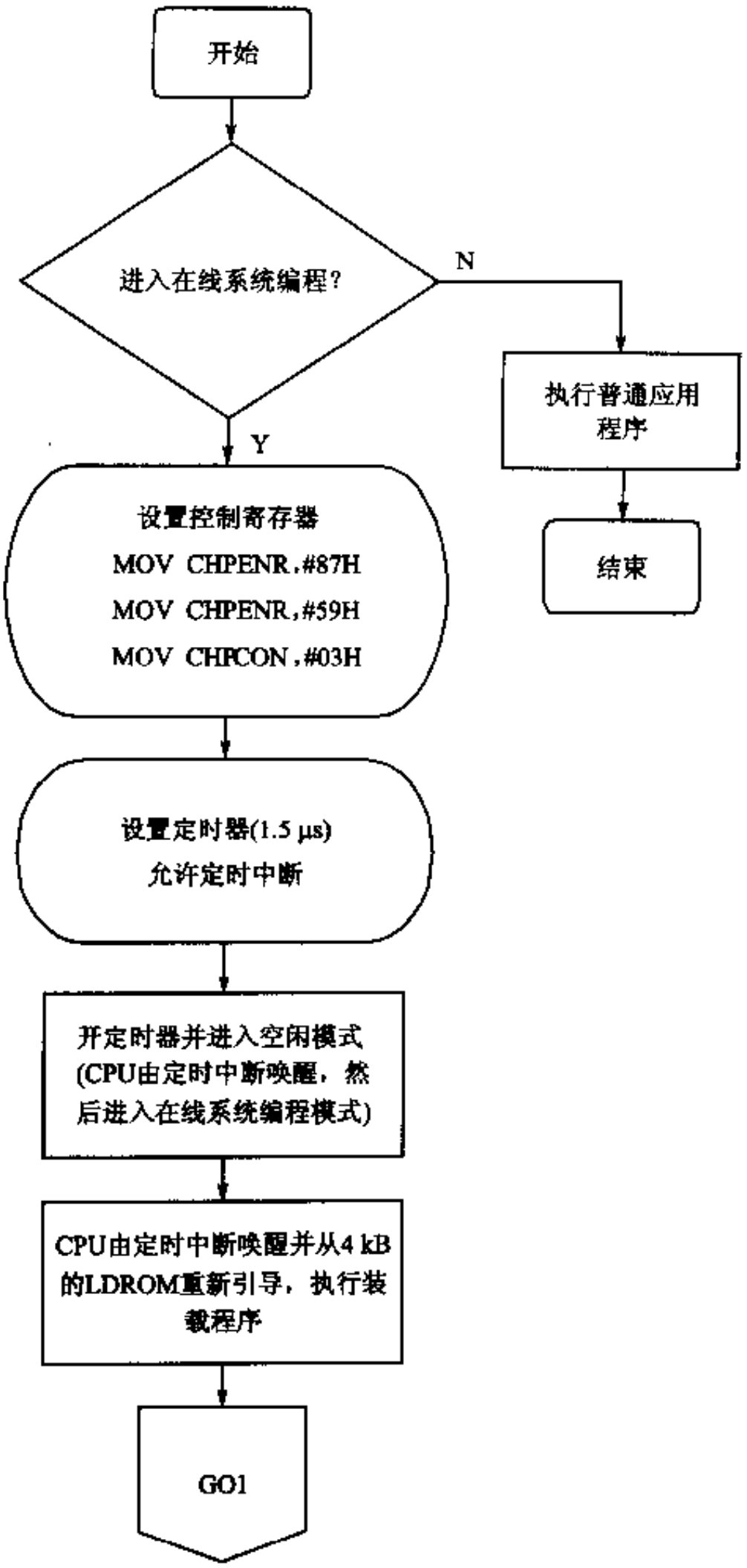


图 17.1 进入在线系统编程模式的流程图(在 64 KB APROM 中)

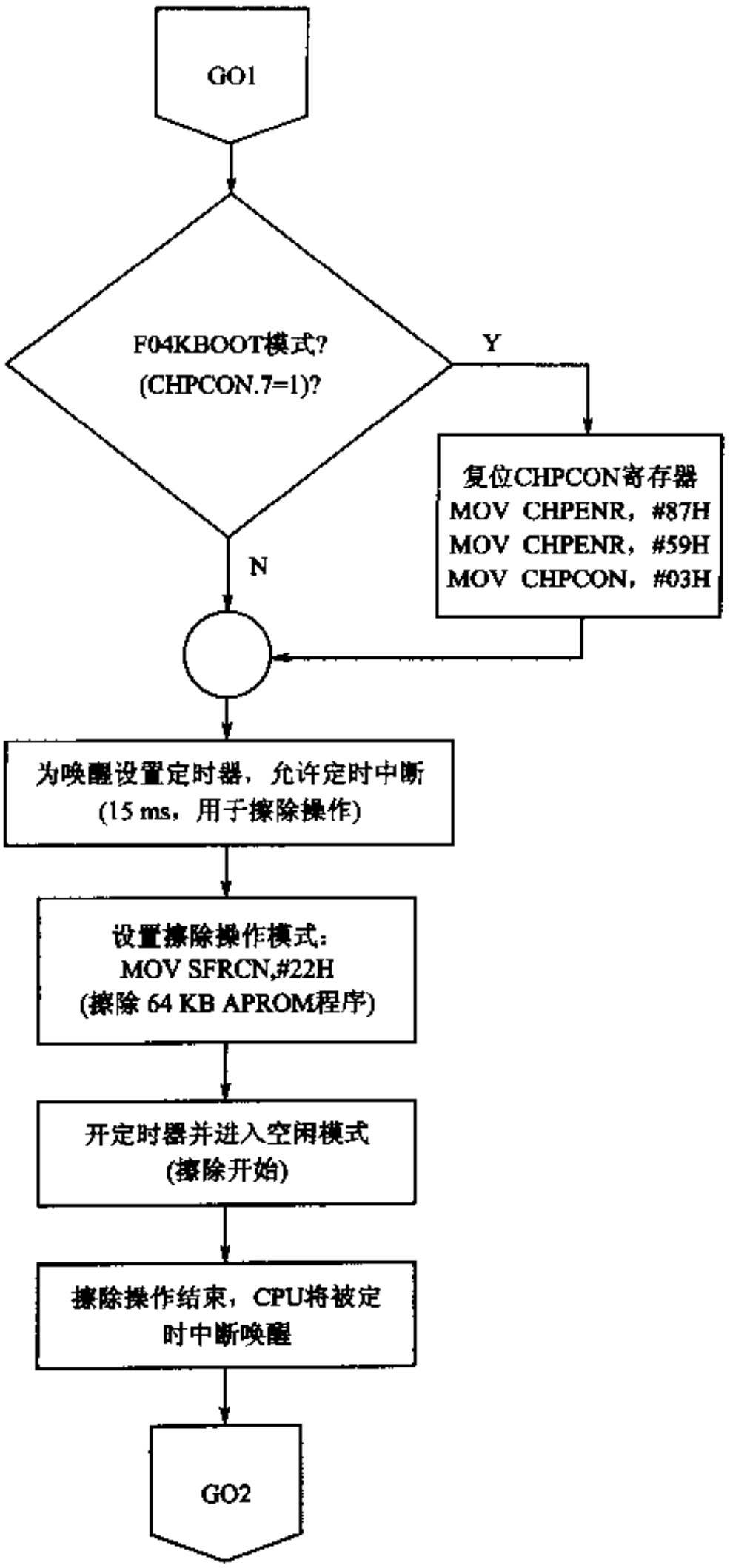


图 17.2　更新 64 KB APROM 程序的流程图(在 4 KB LDROM 中)

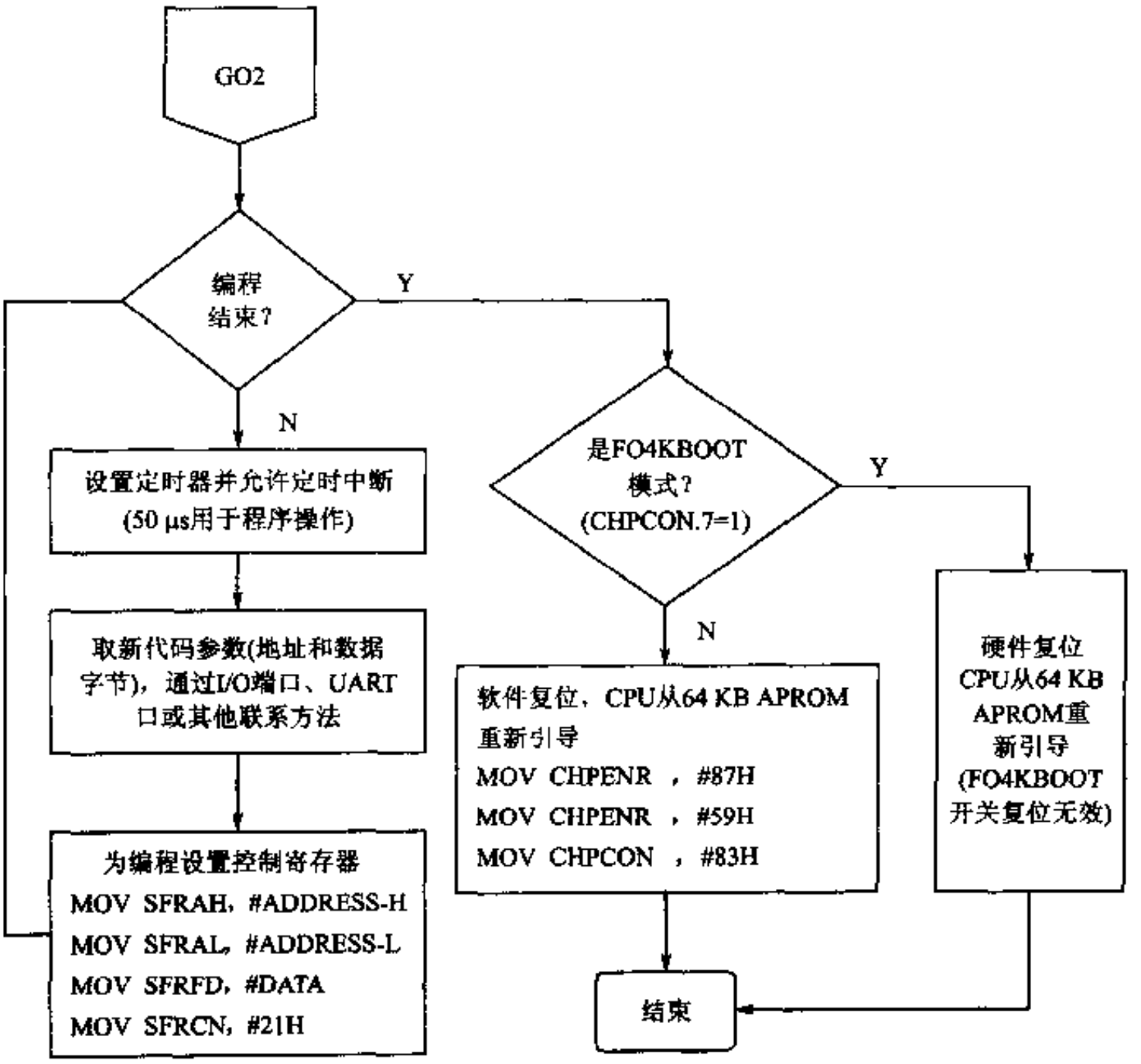


图 17.3　更新 64 KB APROM 程序的流程图(在 4 KB LDROM 中)

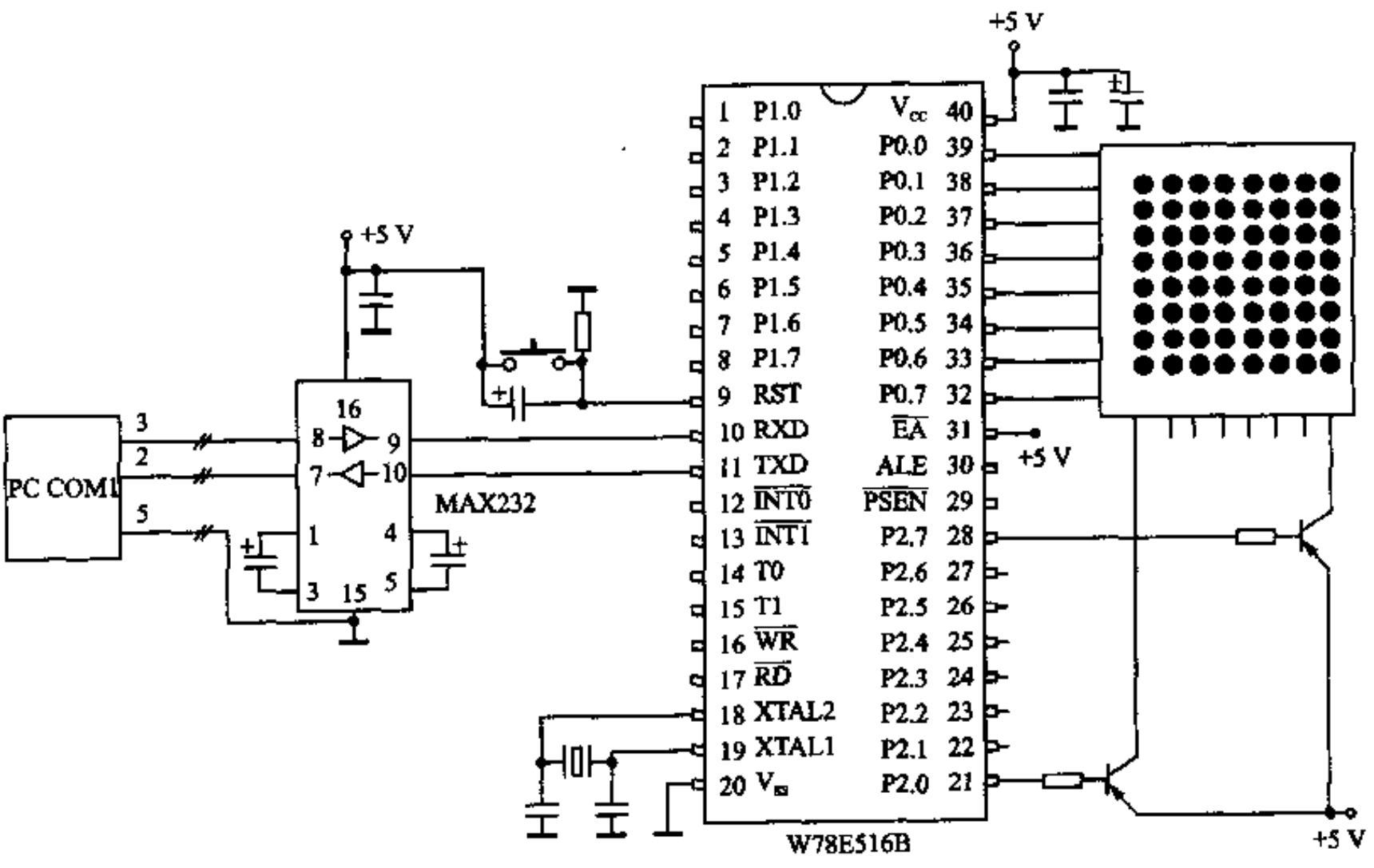


图 17.4　可在线编程的应用系统硬件原理图

5. 应用系统的程序设计

在 W78E516B 单片机应用系统中，为了实现从应用模式到在线编程模式的相互转换，在 APROMR 和 LDROM 中都需设计相应的控制程序，应用程序装入 0000H～FFFFH 地址处，装载程序在 10000H～10FFFH 处。

在应用系统设计开发时，先用烧录器写入装载程序，然后即可用计算机的超级终端对应用系统进行应用程序的写入及修改。其主要控制程序有：

(1) 装载程序

① 命令获取程序模块：对接收到的字符命令进行比较、出错处理，并转入相应的功能模块。

② 功能执行程序模块：有显示帮助信息功能；显示内存数据功能；设置波特率功能；擦除 64 KB APROM 功能；APROM 加密功能；mcu 信息显示功能；查空功能；程序代码更新功能和复位功能等。

③ 串口中断服务程序模块：负责接收和发送字符。

④ xmodem 协议接收程序：在计算机发送程序代码(updat)时，以 xmodem 协议接收。

(2) 应用程序中在线编程控制程序

为了在应用系统工作时能进入在线编程模式，在应用系统的应用程序中应由控制程序来执行这一功能，通常放在串口中断服务程序中。此程序负责接收口令代码并与相应设置的代码进行比较，相符时则进入在系统编程模式。

以下是可以在线修改程序的 8×8 点阵字符显示器源程序，包括装载程序和应用程序。装载程序在 4 KB ROM 中，应用程序在 64 KB ROM 中。

装载程序

```
;       ****************************************
;       *                                      *
;       *          LOAD 程序下载器             *
;       *                                      *
;       *      创建：2001.10.20 陈　斌         *
;       *      修改：2002.03.15 楼然苗         *
;       *                                      *
;       *      CPU：  W78E516B                 *
;       *                                      *
;       ****************************************
;
```

```
;*****************************************************************************
;*                                                                           *
;*                          P1.0    1          40    Vcc                     *
;*                          P1.1    2          39    P0.0                    *
;*                          P1.2    3          38    P0.1                    *
;*                          P1.3    4          37    P0.2                    *
;*                          P1.4    5          36    P0.3                    *
;*                          P1.5    6          35    P0.4                    *
;*                          P1.6    7          34    P0.5                    *
;*                          P1.7    8          33    P0.6                    *
;*                          RST     9          32    P0.7                    *
;*              RXD         P3.0    10  51单片机  31    EA        VDD       *
;*              TXD         P3.1    11         30    ALE                     *
;*              INT0        P3.2    12         29    PSEN                    *
;*              INT1        P3.3    13         28    P2.7                    *
;*    MCUDOG    T0          P3.4    14         27    P2.6                    *
;*              T1          P3.5    15         26    P2.5                    *
;*              WR          P3.6    16         25    P2.4                    *
;*              RD          P3.7    17         24    P2.3                    *
;*                          XTAL2   18         23    P2.2                    *
;*                          XTAL1   19         22    P2.1                    *
;*                          Vss     20         21    P2.0                    *
;*****************************************************************************
;
;
;*********************************************
;*                                           *
;*              变量定义                     *
;*                                           *
;*********************************************
;
;常数定义
                RXBUFFSTART     EQU   040H    ;接收缓冲区队列首址 (170 B)
                XBUFFSTART      EQU   050H    ;发送缓冲区队列首址 (170 B)
                RXBUFFEND       EQU   0F0H    ;接收缓冲区队列末址
                COMMANDLEN      EQU   008H    ;命令字符最大长度
;
;编程延时
                P15MS           EQU   0C9H    ;11.0592 MHz 12.000 MHz 14.318 MHz
                P1US            EQU   0FEH    ;
                P50US           EQU   0D1H    ;
;备用
;               P15MS           EQU   0B7H    ; 18.4320 MHz 24.000 MHz
;
```

```
;           P1US            EQU   0FDH      ;
;           P50US           EQU   0C2H      ;
;
;流控字符
            XON             EQU   011H      ;继续传送
            XOFF            EQU   013H      ;停止传送
;
;控制键
            CTRLC           EQU   003H      ;CTRL_C 键
            ESC             EQU   027H      ;ESC 键
            BACKSPACE       EQU   008H      ;退后删除键
;XMODEM 协议关键字
;发送端→接收端
            SOH             EQU   001H      ;每个包的开始
            EOT             EQU   004H      ;通知接收端结束
;接收端→发送端
            ACK             EQU   006H      ;包成功接收
            NAK             EQU   015H      ;包出错
            CAN             EQU   018H      ;结束传送操作
            CRC             EQU   043H      ;C
;
            BLKSIZE         EQU   080H      ;每包的数据大小
            XMODEMMXAERROR  EQU   01EH      ;最大错误次数
;芯片引脚定义
;WATCH DOG
            MCUDOG          EQU   P3.3      ;看门狗清零
;内部寄存器定义
            T2CON           EQU   0C8H      ;T2 控制寄存器
            T2MOD           EQU   0C9H      ;
            TL2             EQU   0CCH      ;T2 计数寄存器低字节
            TH2             EQU   0CDH      ;T2 计数寄存器高字节
            TR2             EQU   0CAH      ;T2 启动位
            RCAP2L          EQU   0CAH      ;T2 计数重载寄存器低字节
            RCAP2H          EQU   0CBH      ;T2 计数重载寄存器高字节
            CHPCON          EQU   0BFH      ;在系统编程控制寄存器
            CHPENR          EQU   0F6H      ;编程状态下 MTP ROM 的控制字节寄存器
            SFRAL           EQU   0C4H      ;编程状态下的目标低地址
            SFRAH           EQU   0C5H      ;编程状态下的目标高地址
            SFRFD           EQU   0C6H      ;编程状态下 MTP ROM 的编程数据
            SFRCN           EQU   0C7H      ;
;
;控制标志位定义
            TIME10MS        EQU   01H       ;10 ms 标记
```

```
TIME500MS       EQU   02H     ;500 ms 标记
KEYFUNFLAG      EQU   03H     ;
TXOKFLAG        EQU   04H     ;发送 OK 标记
RXCOMMFLAG      EQU   05H     ;接收命令标标志,1 时为 XMODEM 接收
                              方式
COMMSPACEFLAG   EQU   06H     ;命令删除 OK 标记
RXCOMMOKFLAG    EQU   07H     ;
FBOOTFLAG       EQU   08H     ;
;
;全局变量定义
;串口
RXBUFFTAIL      EQU   3BH     ;接收缓冲区尾指针
RXBUFFHEAD      EQU   3AH     ;接收缓冲区头指针
RXBUFFLEN       EQU   39H     ;命令队列长度计数
;
TIMECOUNT       EQU   38H     ;500 ms 时间计数
;
;CRC
CRCLO           EQU   37H     ;
CRCHI           EQU   36H     ;
BLKCHK          EQU   35H     ;
TIMELO          EQU   34H     ;定时器 T0 初值存放地址
TIMEHI          EQU   33H     ;定时器 T0 初值存放地址
DPLO            EQU   32H     ;DPTR 低 8 位初值存放地址
DPHI            EQU   31H     ;DPTR 高 8 位初值存放地址
;
;
ORG     0000H                 ;主程序人口
LJMP    START                 ;转 START
;
;
;*************************************************
;*                                               *
;*                  中断向量入口                 *
;*                                               *
;*************************************************
ORG     0003H                 ;中断入口表
RETI
;
ORG     000BH                 ;定时器 T0 中断入口
LJMP    INT_T0                ;至 T0 中断服务程序
;
ORG     0013H                 ;
```

```
                RETI
;
                ORG     001BH                   ;定时器 T1 中断入口
                LJMP    INT_T1                  ;至 T1 中断服务程序
;
                ORG     0023H                   ;串口中断程序入口
                LJMP    INT_SIO                 ;跳至串行中断服务程序
;
                ORG     002BH                   ;
                RETI
;
                ORG     0080H                   ;以下程序从 0080H 开始
;*************************************************
;*                                               *
;*                  初始化程序                   *
;*                                               *
;*************************************************
START:          MOV     R0,#0F0H                ;清
                CLR     A                       ;00 到
CLRAM:          MOV     @R0,A                   ;F0 末
                DJNZ    R0,CLRAM                ;RAM
;
                MOV     P1,#0FFH                ;置 P1 为 1
                MOV     P2,#0FFH                ;置 P2 为 1
                MOV     P3,#0FFH                ;置 P3 为 1
                MOV     DPL,#00H                ;清 DPTR
                MOV     DPH,#00H
                MOV     PSW,#00H                ;设第一组寄存器
                MOV     SP,#0F1H                ;设置堆栈指针
                MOV     SCON,#01010000B         ;串口工作方式 1(8 BIT UART)允许接收
                MOV     TMOD,#00010001B         ;定时器工作方式 1(16 BIT CONUTER)
                MOV     TH1,#0DBH               ;10 ms 定时参数
                MOV     TL1,#0FFH               ;
                MOV     T2CON,#00110000B        ;T2CON
                MOV     A,#01H                  ;设置波特率 0:38 400 1:19 200
                LCALL   INITBAUD                ;           2:9 600  3: 4 800
                SETB    ES                      ;允许串口中断
                SETB    ET0                     ;允许定时器 0 中断
                SETB    ET1                     ;允许定时器 1 中断
                MOV     IP,#00H                 ;低优先级
                SETB    TR1                     ;启动定时计数器 1
                SETB    TR2                     ;启动定时计数器 2
                SETB    REN                     ;启动串口接收中断
```

```
        CLR     TI                      ;清串口发送中断标志位
        CLR     RI                      ;清串口接收中断标志位
        SETB    EA                      ;开放所有中断
;
        MOV     RXBUFFHEAD,#RXBUFFSTART ;接收缓冲器头指针(#40H)
        MOV     RXBUFFTAIL,#RXBUFFSTART ;接收缓冲器尾指针(#40H)
        CLR     RXCOMMFLAG              ;接收命令标志清 0
        MOV     DPTR,#LOGOTBL           ;在终端上显示 LOGO 字符
        LCALL   DISPINFO                ;调显示表格内容子程序
        MOV     DPTR,#ROMMARKTBL        ;在终端上显示 ROM>字符
        LCALL   DISPINFO                ;调显示表格内容子程序
;
;*************************************************
;*                                               *
;*                  主程序开始                   *
;*                                               *
;*************************************************
MAIN:   LCALL   WATDOG                  ;清看门狗
        LCALL   GETCOMMAND              ;调获取命令子程序
MAINOUT: LJMP   MAIN                    ;循环
        NOP                             ;PC 出错处理
        NOP
        NOP
        LJMP    START                   ;重新初始化
;
;主要处理程序
;
;*************************************************
;*                                               *
;*                  看门狗                       *
;*                                               *
;*************************************************
WATDOG: CPL     MCUDOG                  ;清看门狗(P3.3)
        RET
;
;
;*************************************************
;*                                               *
;*                获取命令程序                   *
;*                                               *
;*************************************************
GETCOMMAND: LCALL PUTCOMMSPACE          ;刷新命令符
        MOV     A,RXBUFFTAIL            ;取尾指针
```

```
                XRL     A,#RXBUFFSTART          ;是否等于头指针
                JZ      GETCOMMOUT              ;接收缓冲区空不处理,退出
                JNB     RXCOMMOKFLAG,GETCOMMOUT;命令接收 OK 标志为 0 不处理
                MOV     A,RXBUFFLEN             ;取命令长度
                CLR     C                       ;清 C
                SUBB    A,#COMMANDLEN           ;与最大命令长度(#08H)比较
                JNC     GETCOMMERROR            ;命令字符超长出错(≥8)
                CLR     A                       ;
                MOV     R4,A                    ;清查表计数器
                MOV     R5,#RXBUFFSTART         ;置接收缓冲区首址
                MOV     DPTR,#COMMANDTBL        ;置命令表首址
GETCOMMCOMP:    MOV     A,R4                    ;
                MOVC    A,@A+DPTR               ;查表
                INC     R4                      ;查表计数器加 1
                MOV     R6,A                    ;查表值暂存到 R6
                MOV     A,R4                    ;
                JNZ     GETCOMMOVER             ;查表计数器是否溢出
                INC     DPH                     ;查表计数器溢出处理
GETCOMMOVER:    MOV     A,R5                    ;
                INC     R5                      ;接收缓冲区地址加 1
                MOV     R0,A                    ;
                MOV     A,@R0                   ;读接收缓冲区数据
                MOV     R7,A                    ;暂存到 R7
                MOV     A,R6                    ;查表值是否为#0FFH
                XRL     A,#0FFH                 ;
                JZ      GETCOMMERROR            ;是#0FFH 出错处理
                MOV     A,R6                    ;
                XRL     A,#0AH                  ;是否回车符
                JZ      GETCOMMCOMPEND          ;是回车符,比较成功
                MOV     A,R7                    ;接收值与查表值比较
                XRL     A,R6                    ;
                JZ      GETCOMMCOMP             ;相等,去下一个比较
                MOV     A,R6                    ;不相等,看是否为小写
                ADD     A,#20H                  ;小写处理
                XRL     A,R7                    ;与接收值再比较
                JZ      GETCOMMCOMP             ;相等去下一个比较
                MOV     R5,#RXBUFFSTART         ;不相等,重置接收缓冲器首址
                MOV     A,R4                    ;查表次数计数器值调整
                MOV     B,#COMMANDLEN           ;
                DIV     AB                      ;除 8,A 中为刚才查命令表的行数
                MOV     B,#COMMANDLEN           ;
                INC     A                       ;指向下一行
                MUL     AB                      ;查表次数调整(乘 8)
```

```
                MOV     R4,A                    ;查表次数放回 A
                LJMP    GETCOMMCOMP             ;比较下一个命令表
;
;命令比较成功处理
GETCOMMCOMPEND: CLR  A                          ;初始化
                MOV RXBUFFLEN,A                 ;长度清 0
                MOV RXBUFFTAIL,#RXBUFFSTART     ;尾指针为初值(40H)
                CLR  RXCOMMOKFLAG               ;清接收命令 OK 标志
                MOV A,R4                        ;查表次数移入 A
                MOV B,#COMMANDLEN               ;
                DIV  AB                         ;除 8
                MOV B,A                         ;乘 3 处理
                RL   A                          ;
                ADD  A,B                        ;
                MOV DPTR,#COMMFUNTBL            ;取命令功能表首址
                JMP  @A+DPTR                    ;功能散转处理
GETCOMMOUT: RET                                 ;结束
;
;错误命令处理,显示 Bad command
GETCOMMERROR:MOV    DPTR,#COMMERRORTBL          ;取出错提示符表首址
                LCALL   DISPINFO                ;显示 Bad command
                CLR     A                       ;初始化
                MOV     RXBUFFLEN,A             ;清长度计数器
                MOV     RXBUFFTAIL,#RXBUFFSTART ;尾指针为初值(40H)
                CLR     RXCOMMOKFLAG            ;清接收命令 OK 标志
                LJMP    DISPROM                 ;显示光标符
                RET                             ;结束
;
;命令功能表
COMMFUNTBL: LJMP    HELPFUN                     ;帮助功能
                LJMP    DISPROM                 ;回车显示光标符
                LJMP    RAMFUN                  ;显示 RAM 数据功能
                LJMP    SECFUN                  ;校验 ROM 数据
                LJMP    SETBFUN                 ;设置串口波特率
                LJMP    INFOFUN                 ;查看 MCU 资源
                LJMP    ERASEFUN                ;擦除 ROM
                LJMP    BLANKFUN                ;查空
                LJMP    RESETFUN                ;MCU 复位
                LJMP    UPDATFUN                ;数据刷新
                RET                             ;返回
;
;
;*************************************************
```

```
;*                                                    *
;*              定时器 0 中断服务程序                 *
;*                                                    *
;******************************************************
INT_T0:     CLR     TR0                 ;关 T0
            MOV     TL0,TIMELO          ;重装初值
            MOV     TH0,TIMEHI          ;重装初值
            RETI                        ;中断返回
;
;
;******************************************************
;*                                                    *
;*              定时器 1 中断服务程序                 *
;*                                                    *
;******************************************************
INT_T1:     PUSH    PSW                 ;堆栈保护
            PUSH    ACC                 ;
            MOV     TH1,#0DBH           ;重置计数器初值
            MOV     TL1,#0FFH           ;
INTKEY:     CPL     TIME10MS            ;10 ms 标志
            INC     TIMECOUNT           ;10 ms 数器加 1
            MOV     A,TIMECOUNT         ;
            CJNE    A,#10H,INTT0OUT     ;10 ms 中断不满 16 次退出
            MOV     TIMECOUNT,#00H      ;满 16 次复 0
            CPL     TIME500MS           ;500 ms 标志
;
INTT0OUT:   POP     ACC                 ;出栈
            POP     PSW                 ;
            RETI                        ;中断返回
;
;
;=====================================================
;/////////////////////////////////////////////////////
;/////////////////////////////////////////////////////
;=====================================================
; 命令解释器功能处理子程序
;

;******************************************************
;*                                                    *
;*                  显示帮助信息                      *
;*    显示帮助信息表                                  *
;******************************************************
```

```
HELPFUN:    MOV     DPTR,#HELPTBL          ;取命令帮助表首址
            LCALL   DISPINFO               ;显示帮助信息表
            LCALL   INITCOMMAND            ;调初始串口数据程序
            LJMP    DISPROM                ;转回车显示光标符程序
;
;
;*************************************************
;*                                               *
;*                  RAM  功能处理                *
;*   显示 RAM                                    *
;*************************************************
RAMFUN:     CLR     A                      ;初始化
            MOV     R0,A                   ;
            MOV     R1,A                   ;
            MOV     R2,A                   ;
            MOV     B,#01H                 ;
            LCALL   DISPROMRAM             ;调用显示 RAM 子程序
            LCALL   INITCOMMAND            ;串口初始化
            LJMP    DISPROM                ;转回车显示光标符程序
;
;
;*************************************************
;*                                               *
;*                  SEC  功能处理                *
;*   校验 ROM 数据                               *
;*************************************************
SECFUN:     MOV     DPTR,#SECTBL           ;
            LCALL   DISPINFO               ;显示"Are you security(Y/N)"
            MOV     DPTR,#INPUTTBL         ;
            LCALL   DISPINFO               ;显示"Input:"
SECLOOP:    MOV     RXBUFFTAIL,#RXBUFFSTART  ;尾指针初值(40H)
            LCALL   READWAIT               ;等待输入 Y/N
            MOV     R0,#RXBUFFSTART        ;缓冲器首址
            MOV     R4,#59H                ;是否 Y 判断
            LCALL   COMPASC                ;单个字符判断比较(大小写)
            JZ      SECCAPS                ;相等转 SECCAPS
            MOV     R4,#4EH                ;是否 N 判断
            LCALL   COMPASC                ;单个字符判断比较(大小写)
            JNZ     SECERROR               ;不是 N 转 SECERROR
            LJMP    SECOUT                 ;
;
SECCAPS:    MOV     DPTR,#SECOKTBL         ;
            LCALL   DISPINFO               ;显示"Security ok !!!"
```

```
SECOUT:     LCALL   INITCOMMAND            ;初始化串口
            LJMP    DISPROM                ;回车显示光标符
;
SECERROR:   MOV     DPTR,#SETERRORTBL      ;Y/N 输入出错处理
            LCALL   DISPINFO               ;显示"Security Error !!!"
            LJMP    SECLOOP                ;转 SECLOOP 再等待输入
;
;
;*********************************************
;*                                           *
;*                SETB  功能处理             *
;*    波特率设置                             *
;*********************************************
SETBFUN:    MOV     DPTR,#SETBFUNTBL       ;
            LCALL   DISPINFO               ;显示 "0) 38 400 1) 19 200
                                           ;2)9 600 3)4 800 4)Exit Input:"
            MOV     DPTR,#INPUTTBL         ;
            LCALL   DISPINFO               ;显示"Input:"
SETBLOOP:   LCALL   READWAIT               ;等待操作
            MOV     R0,#RXBUFFSTART        ;缓冲器首址
            MOV     A,@R0                  ;读缓冲器数据
            MOV     R7,A                   ;暂存 R7
            DEC     R7                     ;减 1
SETBLOOP1:  INC     R7                     ;加 1
            MOV     RXBUFFTAIL,#RXBUFFSTART  ;尾指针初始值
            LCALL   WATDOG                 ;调看门狗
            XRL     A,R7                   ;
            JNZ     SETBLOOP1              ;是否有输入,无转 SETBLOOP1 等待
            MOV     A,@R0                  ;有输入处理
            SUBB    A,#30H                 ;
            MOV     B,A                    ;
            MOV     A,B                    ;
            XRL     A,#04H                 ;是否 Exit
            JZ      SETBERROR              ;是 4 转 SETBERROR 退出
            MOV     A,B                    ;
            CLR     C                      ;
            SUBB    A,#05H                 ;大于 5 重输
            JNC     SETBRELOAD             ;输入数据大于 5 转 SETBRELOAD 处理
            MOV     A,B                    ;小于 5 设置 Baud rate
            LCALL   INITBAUD               ;调设置波特率子程序
SETBERROR:  LCALL   INITCOMMAND            ;退出处理,串口初始化
            LJMP    DISPROM                ;回车显示光标符
;
```

```
SETBRELOAD:  MOV     RXBUFFTAIL,#RXBUFFSTART   ;尾指针初始值
             MOV     DPTR,#SETERRORTBL         ;
             LCALL   DISPINFO                  ;显示"Error !!! Input:"
             LJMP    SETBLOOP                  ;转 SETBLOOP 等待输入
;
;
;*************************************************
;*                                               *
;*                  显示 MCU 信息                *
;*                                               *
;*************************************************
INFOFUN:     MOV     DPTR,#SYSINFOTBL          ;显示"Winbond W78E516B 512B/64KB"
             LCALL   DISPINFO                  ;          "Ver 1.00"
             CLR     TR2                       ;关定时器 T2
             MOV     A,RCAP2L                  ;读 RCAP2L(定时器初值)
             SETB    TR2                       ;开定时器 T2
             MOV     R2,A                      ;R2 暂存
             CLR     A                         ;
             MOV     R3,A                      ;
             MOV     DPTR,#BAUDTBL             ;查询 Baud rate
GETBAUD:     MOV     A,R3                      ;
             MOVC    A,@A+DPTR                 ;查波特率初值表
             INC     A                         ;
             JZ      INFOFUNOUT                ;是否结束
             DEC     A                         ;
             INC     R3                        ;
             XRL     A,R2                      ;比较
             JNZ     GETBAUD                   ;不相等转 GETBAUD 再查
             DEC     R3                        ;得到 Baud rate(0～3)
             MOV     A,R3                      ;
             MOV     B,#18H                    ;
             MOV     DPTR,#BAUDINFOTBL         ;波特率显示表首址
             LCALL   DISPMUL                   ;显示查表
INFOFUNOUT:  LJMP    DISPROM                   ;回车显示光标符,如"Baud rate 9600"
;
;
;
;*************************************************
;*                                               *
;*                RESET 功能处理                 *
;*   系统复位重启动                              *
;*************************************************
```

```
RESETFUN:   MOV     DPTR,#WARNINGTBL        ;
            LCALL   DISPINFO                ;显示"Are you sure(Y/N)"
            MOV     DPTR,#INPUTTBL          ;
            LCALL   DISPINFO                ;显示"Input:"
RESETLOOP:  MOV     RXBUFFTAIL,#RXBUFFSTART ;
            LCALL   READWAIT                ;
            MOV     R0,#RXBUFFSTART         ;
            MOV     R4,#59H                 ;是否 Y
            LCALL   COMPASC                 ;单字符比较(连大小写)
            JZ      RESETCAPS               ;转复位处理
            MOV     R4,#4EH                 ;是否 N
            LCALL   COMPASC                 ;单字符比较(连大小写)
            JNZ     RESETERROR              ;不是 N,出错处理
            LCALL   INITCOMMAND             ;N 重新串口初始化
            LJMP    DISPROM                 ;回车显示光标符
;
RESETCAPS:  LCALL   PUTENTER                ;显示回车
            LCALL   PUTENTER                ;发回车符
            MOV     CHPENR,#87H             ;软件复位,系统重新启动
            MOV     CHPENR,#59H
            MOV     CHPCON,#03H
            MOV     CHPENR,#87H
            MOV     CHPENR,#59H
            MOV     CHPCON,#83H
            RET
;
RESETERROR: MOV     DPTR,#SETERRORTBL       ;
            LCALL   DISPINFO                ;显示"Error !!! Input"
            LJMP    RESETLOOP               ;等待重新输入
;
;
;*************************************************
;*                                               *
;*                  ERASE 功能处理               *
;*   擦除 ROM 数据                               *
;*************************************************
ERASEFUN:   MOV     CHPENR,#87H             ;ROM 擦除操作模式
            MOV     CHPENR,#59H             ;
            MOV     CHPCON,#03H             ;
            MOV     CHPENR,#00H             ;
            LCALL   ERASECHIP               ;擦除芯片
            LCALL   INITCOMMAND             ;串口初始化
            LJMP    DISPROM                 ;回车显示光标符
```

```
;
;
;*************************************************
;*                                               *
;*                  BLANK 功能处理               *
;* 查空操作程序                                  *
;*************************************************
BLANKFUN:   MOV     CHPENR,#87H          ;启动查空操作模式
            MOV     CHPENR,#59H          ;
            MOV     CHPCON,#03H          ;
            MOV     CHPENR,#00H          ;
            LCALL   BLANKCHIP            ;查空操作
            LCALL   INITCOMMAND          ;串口初始化
            LJMP    DISPROM              ;回车显示光标符
;
;
;*************************************************
;*                                               *
;*                  UPDAT 功能处理               *
;*                                               *
;*************************************************
UPDATFUN:   MOV     CHPENR,#87H          ;起动 IN SYSTEM PROGRAMMING 模式
            MOV     CHPENR,#59H          ;
            MOV     CHPCON,#03H          ;
            MOV     CHPENR,#00H          ;
            LCALL   ERASECHIP            ;擦除芯片
            LCALL   BLANKCHIP            ;检查芯片
            JZ      UPDATOUT             ;为 0 转出错处理
            LCALL   XMODEMDOWN           ;下载数据
            JZ      UPDATOUT             ;为 0 转出错处理
            MOV     DPTR,#UPDATAOKTBL    ;
            LCALL   DISPINFO             ;显示"Updata ok !!!"
            MOV     DPTR,#VERIFYOKTBL    ;
            LCALL   DISPINFO             ;显示"Verify ok !!!"
            LCALL   PUTENTER             ;发回车
            LJMP    DISPROM              ;显示光标符
;
UPDATOUT:   MOV     DPTR,#VERIFYERRORTBL ;
            LCALL   DISPINFO             ;显示"Verify Error !!!"
            MOV     DPTR,#UPDATAERRORTBL ;
            LCALL   DISPINFO             ;显示"Updata Error !!!"
            LCALL   PUTENTER             ;发回车
            LJMP    DISPROM              ;显示光标符
```

```
;
;显示" ROM>"光标信息程序
DISPROM:    MOV     DPTR,#ROMMARKTBL    ;
            LCALL   DISPINFO            ;显示" ROM>"
            RET
;
;
;=============================================================
;//////////////////////////////////////////////////////////////
;//////////////////////////////////////////////////////////////
;=============================================================
; 串口操作子程序
;                                          Osc Freq
; 计算公式:  (RCAP2H,RCAP2L) = 65 536 - ---------
;                                        32×Baud
; DEMO:
; FREQUENCY:   12 MHz
; BAUD RATE:   9600 Baud
;
;                                  12×1 000 000
; (RCAP2H,RCAP2L) = 65 536 - ----------
;                                   32×9 600
;                 = 65 497 = FFD9H
;
;不同波特率设置对应初值表
;           DB      0F7H,0EEH,0DCH,0B8H,0FFH        ; 11.059 2 MHz
BAUDTBL:    DB      0F6H,0ECH,0D9H,0B2H,0FFH        ; 12.000 0 MHz
;           DB      0F4H,0E8H,0D1H,0A3H,0FFH        ; 14.318 0 MHz
;           DB      0F1H,0E2H,0C4H,087H,0FFH        ; 18.432 0 MHz
;           DB      0ECH,0D9H,0B2H,064H,0FFH        ; 24.000 0 MHz
;                   38 400 19 200 9 600 4 800  END
;           RET
;*********************************************
;*                                           *
;*                设置波特率                 *
;*                                           *
;*********************************************
; A 内容 0: 38 400     1: 19 200     2: 9 600     3: 4 800
;设置串口波特率
INITBAUD:   MOV     DPTR,#BAUDTBL       ;赋波特率初值表首址
            CLR     TR2                 ;关 T2
            MOVC    A,@A+DPTR           ;查表
            MOV     TL2,A               ;查表值移入 TL2(低 8 位)
```

```
                MOV     RCAP2L,A                ;移入初值寄存器 RCAP2L
                CLR     A                       ;
                DEC     A                       ;
                MOV     TH2,A                   ;高 8 位初值为 #0FFH
                MOV     RCAP2H,A                ;
                SETB    TR2                     ;开启 T2(T2 为波特率用)
                RET
;
;
;*****************************************************
;*                                                   *
;*                串口中断服务程序                   *
;*                                                   *
;*****************************************************
;串口中断处理,处理串口发送及接收数据。
;
INT_SIO:        PUSH    PSW                     ;堆栈保护
                PUSH    ACC
                MOV     PSW,#04H                ;设第二组寄存器
                JBC     RI,RXINFO               ;RI=1,去接收中断程序 RI=0
                JNB     TI,INTSIOOUT            ;TI 为 0,退出中断
                CLR     TI                      ;TI 为 1,清发送中断
                LJMP    TXINFO                  ;转发送中断程序
;
;串口接收中断程序
RXINFO:         JB      RXCOMMFLAG,RXXMODEM     ;接收命令标志为 1,XMODEM 接收
RXCOMMAND:      MOV     R0,RXBUFFTAIL           ;接收命令,尾指针入 R0
                CLR     RXCOMMOKFLAG            ;清接收命令 OK 标志
                MOV     A,SBUF                  ;接收数据
                XRL     A,#BACKSPACE            ;删除处理(#008H)
                JNZ     RXCOMMAND1              ;不是删除键,转 RXCOMMAND1
                SETB    COMMSPACEFLAG           ;设需要删除标记
                DEC     RXBUFFLEN               ;接收队列调整。队列长度减 1
                MOV     A,RXBUFFLEN             ;放入 A
                INC     A                       ;加 1
                JNZ     RXCOMMANDB1             ;队列长度不为 0,转 RXCOMMANDB1
                CLR     COMMSPACEFLAG           ;删除标志清 0
                INC     RXBUFFLEN               ;队列长度加 1
RXCOMMANDB1:    DEC     RXBUFFTAIL              ;尾指针减 1
                MOV     A,RXBUFFTAIL            ;
                INC     A                       ;
                XRL     A,#RXBUFFSTART          ;尾指针是不是等于头指针
                JNZ     INTSIOOUT               ;不相等退出
```

```
                CLR     COMMSPACEFLAG           ;清删除标志
                INC     RXBUFFTAIL              ;尾指针加 1
                LJMP    INTSIOOUT               ;转退出
;
;
RXCOMMAND1: INC     RXBUFFLEN               ;命令长度计数
                INC     RXBUFFTAIL              ;接收缓冲区计数
                MOV     A,RXBUFFLEN             ;
                XRL     A,#90H                  ;命令长度限制
                JZ      RXCOMMERROR             ;命令长度超过 90H 作出错处理
                MOV     A,SBUF                  ;
                MOV     @R0,A                   ;存命令字符
                XRL     A,#0DH                  ;是否结束判断
                JZ      RXCOMMEND               ;是回车符,接收命令结束
RXCOMMAND2: MOV     A,SBUF                  ;回显收到的字符
                CLR     TXOKFLAG                ;清发送 OK 标志
                MOV     SBUF,A                  ;回发接收到的字符数据
INTSIOOUT:      POP     ACC                     ;堆栈恢复
                POP     PSW                     ;
                RETI    ;中断退出
;
;
RXCOMMERROR:CLR     A                       ;出错处理程序
                MOV     RXBUFFLEN,A             ;初始化,命令长度为 0
                MOV     RXBUFFTAIL,#RXBUFFSTART  ;尾指针指向队列首址(40H)
                LJMP    INTSIOOUT               ;转中断退出
;
RXCOMMEND:  SETB    RXCOMMOKFLAG            ;接收成功,标志为 1
                LJMP    INTSIOOUT               ;转中断退出
;
;XMODEM 协议接收程序
RXXMODEM:       MOV     R0,RXBUFFTAIL           ;接收缓冲器放尾指针(地址)
                INC     RXBUFFTAIL              ;接收队列处理。尾地址加 1
                INC     RXBUFFLEN               ;队列长度加 1
                MOV     A,SBUF                  ;接收数据
                MOV     @R0,A                   ;放入接收缓冲器
                MOV     A,BLKCHK                ;求队列总长度
                ADD     A,#BLKSIZE              ;(#128)
                ADD     A,#XBUFFSTART           ;(#40H)
                ADD     A,#04H                  ;
                XRL     A,RXBUFFTAIL            ;与尾地址比较
                JNZ     INTSIOOUT               ;队列未满转中断退出
                MOV     RXBUFFTAIL,#XBUFFSTART  ;写队列满
```

```
                LJMP    INTSIOOUT               ;转中断退出
;
;
TXINFO:         SETB    TXOKFLAG                ;发送成功,发送成功标志置 1
                LJMP    INTSIOOUT               ;中断退出
;
;
;*****************************************
;*                                       *
;*          把表的内容写到串口           *
;*                                       *
;*****************************************
;DPTR 内容是表的首址
;显示表格内容
DISPINFO:       CLR     A                       ;
                MOV     R5,A                    ;清 R5
READDISPCODE:   MOV     A,R5                    ;计数值入 A
                MOVC    A,@A+DPTR               ;查表
                MOV     R4,A                    ;查表数放入 R4
                INC     R5                      ;计数器加 1
                MOV     A,R5                    ;放回 A
                JNZ     READOVER                ;溢出处理。不溢出转 READOVER
                INC     DPH                     ;溢出 DPH 加 1
READOVER:       MOV     A,R4                    ;
                INC     A                       ;查得数加 1
                JZ      DISPINFOOUT             ;读到结束符结束(为 0 退出)
                CLR     TXOKFLAG                ;清发送 OK 标志
                DEC     A                       ;A 减 1
                LCALL   PUTCHAR                 ;发送 1 个字符
                LJMP    READDISPCODE            ;循环
DISPINFOOUT:    RET                             ;子程序结束
;
;
;*****************************************
;*                                       *
;*          刷新命令符                   *
;*                                       *
;*****************************************
PUTCOMMSPACE:   JNB     COMMSPACEFLAG,PUTCOMOUT ;命令刷新标志不为 1 退出
                CLR     COMMSPACEFLAG           ;清命令刷新标志
                MOV     A,#BACKSPACE            ;置光标到前一字符
                LCALL   PUTCHAR                 ;发送后退键数据
                MOV     A,#20H                  ;用空格覆盖
```

```
                LCALL   PUTCHAR                 ;发空格数据
                MOV     A,#BACKSPACE            ;调整光标位置
                LCALL   PUTCHAR                 ;发后退键数据
PUTCOMOUT:      RET                             ;结束
;
;
;*********************************************
;*                                           *
;*              发送一个字符程序             *
;*                                           *
;*********************************************
;A 内容待发字符
PUTCHAR:        CLR     TXOKFLAG                ;清发送 OK 标志
                MOV     SBUF,A                  ;发送 1 个字节
                JNB     TXOKFLAG,$              ;发送 OK 标志不为 1 等待
                RET                             ;结束
;
;
;*********************************************
;*                                           *
;*              发送字符串程序               *
;*                                           *
;*********************************************
;A 内容待发字符首址
;B 内容待发字符个数
PUTSTRING:      MOV     R0,A                    ;赋首址
                MOV     R4,B                    ;发送字符个数
PUTSTRLOOP:     MOV     A,@R0                   ;发送数据入 A
                INC     R0                      ;下一地址
                LCALL   PUTCHAR                 ;发送一个字符
                DJNZ    R4,PUTSTRLOOP           ;未发完转 PUTSTRLOOP 循环
                RET
;
;
;*********************************************
;*                                           *
;*              接收一个字符程序             *
;*                                           *
;*********************************************
;A 内容接收字符
;B 内容超时次数
GETCHAR:        MOV     A,RXBUFFLEN             ;接收字符长度入 A
                JNZ     GETCHAROUT              ;不为 0 转 GETCHAROUT
```

```
                JB      TIME500MS,$             ;500 ms 标志为 1 等待
                LCALL   WATDOG                  ;调看门狗
                JNB     TIME500MS,$             ;500 ms 标志为 0 等待
                DEC     B                       ;超时计数
                MOV     A,B                     ;
                JNZ     GETCHAR                 ;未超时转 GETCHAR 再等
                RET                             ;超时结束
;
GETCHAROUT:     MOV     R0,RXBUFFHEAD           ;接收缓冲器首值人 R0
                INC     RXBUFFHEAD              ;缓冲器地址加 1
                DEC     RXBUFFLEN               ;长度计数器减 1
                MOV     A,#BLKSIZE              ;(#80H)
                ADD     A,BLKCHK                ;(校验和)
                ADD     A,#54H                  ;
                XRL     A,RXBUFFHEAD            ;是否相等
                JNZ     GETCHAREND              ;不相等转 GETCHAREND
                MOV     RXBUFFHEAD,#XBUFFSTART  ;初始化
GETCHAREND:     MOV     A,@R0                   ;取字符数据
                RET                             ;退出
;
;
;*********************************************
;*                                           *
;*                显示回车                   *
;*                                           *
;*********************************************
;向串口发送回车
PUTENTER:       MOV     A,#0DH                  ;回车键码
                LCALL   PUTCHAR                 ;发回车键
                MOV     A,#0AH                  ;回车键码
                LCALL   PUTCHAR                 ;发回车键
                RET
;
;
;*********************************************
;*                                           *
;*                显示查表                   *
;*                                           *
;*********************************************
;DPRT 显示信息首址
;A 内容数据
;B 内容长度
DISPMUL:        MUL     AB                      ;偏移地址
```

```
                LCALL   DPTRADDC                ;DPTR 计数器调整
                LCALL   DISPINFO                ;显示表格
                RET
;
;
;
;*****************************************
;*                                       *
;*              显示存储器 ADDER          *
;*                                       *
;*****************************************
;R2 内容高地址          R1 内容低地址
;显示成 XXXX:
DISPADD:        LCALL   PUTENTER                ;显示回车
                MOV     A,R2                    ;
                LCALL   DISPHEX                 ;十六进制格式显示
                MOV     A,R1                    ;
                LCALL   DISPHEX                 ;十六进制格式显示
                MOV     A,#3AH                  ;显示":"
                LCALL   PUTCHAR                 ;发送":"
                MOV     A,#20H                  ;
                LCALL   PUTCHAR                 ;发空格
                MOV     A,#20H                  ;
                LCALL   PUTCHAR                 ;发空格
                RET
;
;
;*****************************************
;*                                       *
;*              显示 BIN                 *
;*                                       *
;*****************************************
;A 内容是 BCD 码
;显示成 XXXXXXXXB 二进制格式
DISPBIN:        MOV     R3,#08H                 ;字节计数
BINRL:          RLC     A                       ;得到每一位
                JC      ONE                     ;
                MOV     A,#30H                  ;
                LCALL   PUTCHAR                 ;发送 0
                LJMP    BINRL1                  ;
;
ONE:            MOV     A,#31H                  ;
                LCALL   PUTCHAR                 ;发送 1
```

```
BINRL1:     DJNZ    R3,BINRL            ;是否结束
            MOV     A,#42H              ;
            LCALL   PUTCHAR             ;发送 B
            RET
;
;
;*********************************************
;*                                           *
;*          单字节十六进制转十进制           *
;*                                           *
;*********************************************
;A:待转换的单字节十六进制数
;A:高四位的 ASCII 码          R4:低四位的 ASCII 码
HEX2ASC:    MOV     R4,A                ;暂存待转换的单字节十六进制数
            LCALL   HEX2ASC1            ;转换低 4 位
            XCH     A,R4                ;存放低 4 位的 ASCII 码
            SWAP    A                   ;准备转换高 4 位
HEX2ASC1:   ANL     A,#0FH              ;将累加器的低 4 位转换成 ASCII 码
            ADD     A,#90H              ;
            DA      A                   ;
            ADDC    A,#40H              ;
            DA      A                   ;
            CLR     C                   ;
            RET     ;
;
;
;*********************************************
;*                                           *
;*              显示 HEX                     *
;*                                           *
;*********************************************
;A 内容是 BCD 码
;显示成 XX 十六进制格式
DISPHEX:    LCALL   HEX2ASC             ;发高 4 位的 ASCII 码
            LCALL   PUTCHAR             ;发送
            MOV     A,R4                ;发低 4 位的 ASCII 码
            LCALL   PUTCHAR             ;发送
            RET                         ;
;
;
;*********************************************
;*                                           *
```

```
;*                        显示 RAMASC                          *
;*                                                             *
;*************************************************************
;显示 RAM 或 ROM 内容
;R0 内容是 R2R1 地址的内容
;ASC 方式
;格式：0123456789ABCDEF
DISPRAMASC:   MOV     R6,#08H                ;
              MOV     R0,A                   ;
RAMSPACE:     MOV     A,#20H                 ;显示 8 个空格
              LCALL   PUTCHAR                ;
              DJNZ    R6,RAMSPACE            ;
              MOV     A,R0                   ;
              SUBB    A,#10H                 ;指针减一列长度
              MOV     R0,A                   ;
              MOV     R6,#10H                ;一列计数
RAMASC:       MOV     A,B                    ;
              JZ      ROMASC                 ;ROM RAM 处理
              MOV     A,@R0                  ;
              LJMP    ROMRAMASC              ;
;
ROMASC:       MOV     A,R0                   ;
              MOVC    A,@A+DPTR              ;
;             .
ROMRAMASC:    MOV     R4,A                   ;
              CLR     C                      ;
              SUBB    A,#2EH                 ;过滤小于 #2EH ASC
              MOV     A,R4                   ;
              JNC     RAMASC1                ;
RAMASC0:      MOV     A,#2EH                 ;
              LJMP    RAMASC2                ;
;
RAMASC1:      CLR     C                      ;
              MOV     A,R4                   ;
              SUBB    A,#7FH                 ;过滤大于 #7FH ASC
              MOV     A,R4                   ;
              JNC     RAMASC0                ;
RAMASC2:      LCALL   PUTCHAR                ;显示 ASC
              INC     R0                     ;
              DJNZ    R6,RAMASC              ;
              RET
;
;
```

```
;*********************************************
;*                                           *
;*              显示 RAM ROM                 *
;*                                           *
;*********************************************
;显示 RAM 或 ROM 内容
;R2 内容高位地址 R1 内容低位地址
;R0 内容是 R2R1 地址的内容
;HEX 方式
;格式：XXXX：10 11 12 13 14 15 16 17—18 19 1A 1B 1C 1D 1E 1F
DISPROMRAM: LCALL   DISPADD                ;显示地址
            MOV     R6,#10H                ;一行计数
READRAMLOOP:MOV     A,B                    ;
            JZ      READROM                ;
            MOV     A,@R0                  ;
            LJMP    READRAMROMEND          ;
;
READROM:    MOV     A,R0                   ;
            MOVC    A,@A+DPTR              ;
READRAMROMEND: INC  R0                     ;
            LCALL   DISPHEX                ;显示 HEX
            MOV     A,R0                   ;
            MOV     A,R6                   ;
            XRL     A,#09H                 ;
            JNZ     RAMROM16               ;8 列处理
            MOV     A,#2DH                 ;显示 —
            LCALL   PUTCHAR                ;
            LJMP    RAMROM8                ;
;
RAMROM16:   MOV     A,#20H                 ;显示空格
            LCALL   PUTCHAR                ;
RAMROM8:    DJNZ    R6,READRAMLOOP         ;
            MOV     A,R0                   ;
            LCALL   DISPRAMASC             ;
            LCALL   WATDOG                 ;
            MOV     A,R1                   ;
            ADD     A,#10H                 ;
            MOV     R1,A                   ;
            JNZ     DISPROMRAM             ;是否结束
            INC     DPH                    ;
            MOV     A,DPH                  ;
            MOV     R2,A                   ;
            RET
```

```
;
;
;*********************************************
;*                                           *
;*            初始化串口(COMMAND)            *
;*                                           *
;*********************************************
;初始化串口数据
INITCOMMAND:CLR     A                         ;
            MOV     RXBUFFLEN,A               ;初始化
            CLR     RXCOMMOKFLAG              ;接收命令 OK 标志清 0
            MOV     RXBUFFTAIL,#RXBUFFSTART   ;尾指针复位
            CLR     RXCOMMFLAG                ;启动 COMMAND 接收
            LCALL   PUTENTER                  ;显示回车
            RET
;
;
;*********************************************
;*                                           *
;*                读  等  待                 *
;*                                           *
;*********************************************
;死读缓冲区一直到有数据
READWAIT:   MOV     RXBUFFTAIL,#RXBUFFSTART   ;尾指针复位
            CLR     A                         ;
            MOV     A,RXBUFFLEN               ;长度计数清 0
READWAIT1:  LCALL   WATDOG                    ;看门狗
            MOV     A,RXBUFFTAIL              ;
            XRL     A,#RXBUFFSTART            ;
            JZ      READWAIT1                 ;接收缓冲区空循环等
READLOOP:   LCALL   WATDOG                    ;
            LCALL   PUTCOMMSPACE              ;刷新命令符
            JNB     RXCOMMOKFLAG,READLOOP     ;命令接收不成功不处理
            MOV     R0,#RXBUFFSTART           ;
            RET                               ;
;
;
;*********************************************
;*                                           *
;*              比较单个字符                 *
;*                                           *
;*********************************************
;R0 内容为地址
```

```
;R4 内容为源
;A 内容为零:成功;非零:失败
COMPASC:    MOV     A,@R0                   ;
            XRL     A,R4                    ;是否大写
            JZ      COMPASCOUT              ;
            MOV     A,R4                    ;是否小写
            ADD     A,#20H                  ;
            MOV     R4,A                    ;
            MOV     A,@R0                   ;
            XRL     A,R4                    ;
COMPASCOUT: RET
;
;
;*****************************************
;*                                       *
;*              是否数字                 *
;*                                       *
;*****************************************
;R0 内容为地址
;R4 内容为源
;A 内容为零:成功;非零:失败
ISDIGIT:    MOV     A,@R0                   ;
            CLR     C                       ;
            SUBB    A,#30H                  ;
            JC      ISDIGITOUT              ;小于#30H 非数字
            MOV     A,@R0                   ;
            CLR     C                       ;
            SUBB    A,#3AH                  ;
            JNC     ISDIGITOUT              ;大于#3AH 非数字
            CLR     A                       ;是数字
            RET                             ;
;
ISDIGITOUT: CLR     A                       ;不是数字
            DEC     A                       ;
            RET                             ;
;
;
;*********************************************
;*                                           *
;*                DPTR ADDC                  *
;*   DPTR 调整                               *
;*********************************************
;DPTR=DPTR + A
```

```
;IF DPTR > 0FFFFH      CY = 1
DPTRADDC:   PUSH    ACC             ;栈保护
            ADD     A,DPL           ;DPL + ACC
            MOV     DPL,A           ;
            MOV     A,DPH           ;
            ADDC    A,#000H         ;DPH = DPH + CY
            MOV     DPH,A           ;
            POP     ACC             ;恢复栈
            RET
;
;
;*****************************************
;*                                       *
;*              PAUSE                    *
;*                                       *
;*****************************************
MYPAUSE:    JNB     TIME500MS,$     ;低电平等 500 ms
            LCALL   WATDOG          ;调看门狗
            JB      TIME500MS,$     ;高电平等 500 ms
            DEC     A               ;
            JNZ     MYPAUSE         ;不为 0 再转 MYPAUSE 等
            RET                     ;结束
;
;
;=========================================
;/////////////////////////////////////////
;/////////////////////////////////////////
;=========================================
;XMODEM 子程序
;
;*****************************************
;*                                       *
;*          XMODEM DOWNLOAD              *
;*                                       *
;*****************************************
;以 XMODEM 协议接收数据
XMODEMDOWN:LCALL    PUTENTER            ;
           CLR      A                   ;
           MOV      DPLO,A              ;
           MOV      DPHI,A              ;
           MOV      R7,#XMODEMMXAERROR  ;最大错误次数
           MOV      RXBUFFTAIL,#XBUFFSTART ;
```

```
                MOV     RXBUFFHEAD,#XBUFFSTART;
                CLR     RXCOMMOKFLAG            ;
                CLR     A                       ;
                MOV     RXBUFFLEN,A             ;
                INC     A                       ;
                MOV     BLKCHK,A                ;1: CRC          0: CheckSum
                MOV     R5,A                    ;存包号
                MOV     R6,#CRC                 ;发送字符 C 请求字符 CRC
                SETB    RXCOMMFLAG              ;起动 XMODEM 协议接收
XDOWNLOOP:      MOV     A,R6                    ;
                LCALL   PUTCHAR                 ;
                MOV     B,#05H                  ;TIMEOUT
                LCALL   GETCHAR                 ;读入字符
                MOV     R3,A                    ;
                MOV     A,B                     ;
                JNZ     XDOWNREAD               ;是否超时
                MOV     A,R7                    ;
                CLR     C                       ;
                SUBB    A,#0AH                  ;
                JNC     XDOWNLOOPEND            ;继续发送 CRC 字符
                MOV     R6,#NAK                 ;
                CLR     A                       ;
                MOV     BLKCHK,A                ;发送 CheckSum 字符
XDOWNLOOPEND:   DJNZ    R7,XDOWNLOOP            ;
XDOWNCAN:       MOV     A,#CAN                  ;错误太多中断
                LCALL   PUTCHAR                 ;
                CLR     A                       ;
                LJMP    XDOWNEND2               ;

;
XDOWNEND1:      LCALL   PUTENTER                ;
                CLR     A                       ;
                DEC     A                       ;
XDOWNEND2:      CLR     RXCOMMFLAG              ;启动 COMMAND 接收
                MOV     RXBUFFTAIL,#RXBUFFSTART ;
                MOV     RXBUFFHEAD,#RXBUFFSTART ;
                MOV     RXBUFFLEN,#00H          ;
                RET
;
XDOWNEND:       MOV     A,#ACK                  ;
                LCALL   PUTCHAR                 ;
                LJMP    XDOWNEND1               ;
;
XDOWNREAD:      MOV     R6,#NAK                 ;
```

```
        MOV     A,R3                    ;
        XRL     A,#SOH                  ;是否包头
        JZ      XDOWNPACK               ;
        MOV     A,R3                    ;
        XRL     A,#EOT                  ;是否包结束
        JZ      XDOWNEND                ;
        MOV     A,R3                    ;
        XRL     A,#CAN                  ;是否发送端中断
        JZ      XDOWNCAN                ;
        MOV     A,#05H                  ;
        LCALL   MYPAUSE                 ;
        LJMP    XDOWNLOOPEND            ;
;
XDOWNPACK:  MOV     A,#BLKSIZE          ;包处理
        ADD     A,BLKCHK                ;
        ADD     A,#03H                  ;
        MOV     R4,A                    ;计算包长度
XDOWNPACKLOOP: MOV  B,#03H              ;
        LCALL   GETCHAR                 ;
        MOV     A,B                     ;
        JZ      XDOWNLOOPEND            ;
        DJNZ    R4,XDOWNPACKLOOP        ;读入一包数据

        MOV     R0,#XBUFFSTART          ;
        INC     R0                      ;
        MOV     A,@R0                   ;读入包号
        MOV     B,A                     ;
        INC     R0                      ;
        MOV     A,@R0                   ;读入包号反值
        ADD     A,B                     ;
        XRL     A,#0FFH                 ;校验包号
        JNZ     XDOWNLOOPEND            ;
        MOV     A,B                     ;是否当前包号
        XRL     A,R5                    ;
        JZ      XDOWNCHECK              ;是转 CRC 计算
        MOV     A,R5                    ;
        DEC     A                       ;是否上次包号
        XRL     A,B                     ;
        JNZ     XDOWNLOOPEND            ;
        MOV     R6,#ACK                 ;是发握手信号
        LJMP    XDOWNLOOP               ;
;
XDOWNCHECK:  MOV    A,BLKCHK            ;
```

```
                JZ      XDOWNCHECKSUM       ;CheckSum 处理
                LCALL   XCRCCHECK           ;计算 CRC
                JNZ     XDOWNLOOPEND        ;
                LJMP    XDOWNNEXTPACK       ;是读下一包
;
XDOWNCHECKSUM: LCALL XCHECKSUM              ;计算 CheckSum
                JNZ     XDOWNLOOPEND        ;
XDOWNNEXTPACK:MOV   R6,#ACK                 ;是发握手信号
                LCALL   PROGRAMCHIP         ;
                JZ      XDOWNCAN1           ;
                INC     R5                  ;初始化
                CLR     A                   ;
                MOV     R7,#XMODEMMXAERROR  ;最大错误次数
                MOV     RXBUFFTAIL,#XBUFFSTART ;
                MOV     RXBUFFHEAD,#XBUFFSTART;
                CLR     A                   ;
                MOV     RXBUFFLEN,A         ;
                CLR     RXCOMMOKFLAG        ;
                LJMP    XDOWNLOOP           ;
;
XDOWNCAN1:  LJMP    XDOWNCAN                ;
;
;
;*************************************************************
;/////////////////////////////////////////////////////////////
;/////////////////////////////////////////////////////////////
;*************************************************************
;CRC16 子程序 16 位 CRC          CCITT — 12
;CheckSum 子程序
;
CRC16TBL:   DW      00000H,01021H,02042H,03063H,04084H,050A5H,060C6H,070E7H
            DW      08108H,09129H,0A14AH,0B16BH,0C18CH,0D1ADH,0E1CEH,0F1EFH
            DW      01231H,00210H,03273H,02252H,052B5H,04294H,072F7H,062D6H
            DW      09339H,08318H,0B37BH,0A35AH,0D3BDH,0C39CH,0F3FFH,0E3DEH
            DW      02462H,03443H,00420H,01401H,064E6H,074C7H,044A4H,05485H
            DW      0A56AH,0B54BH,08528H,09509H,0E5EEH,0F5CFH,0C5ACH,0D58DH
            DW      03653H,02672H,01611H,00630H,076D7H,066F6H,05695H,046B4H
            DW      0B75BH,0A77AH,09719H,08738H,0F7DFH,0E7FEH,0D79DH,0C7BCH
            DW      048C4H,058E5H,06886H,078A7H,00840H,01861H,02802H,03823H
            DW      0C9CCH,0D9EDH,0E98EH,0F9AFH,08948H,09969H,0A90AH,0B92BH
            DW      05AF5H,04AD4H,07AB7H,06A96H,01A71H,00A50H,03A33H,02A12H
            DW      0DBFDH,0CBDCH,0FBBFH,0EB9EH,09B79H,08B58H,0BB3BH,0AB1AH
            DW      06CA6H,07C87H,04CE4H,05CC5H,02C22H,03C03H,00C60H,01C41H
```

```
                DW      0EDAEH,0FD8FH,0CDECH,0DDCDH,0AD2AH,0BD0BH,08D68H,09D49H
                DW      07E97H,06E86H,05ED5H,04EF4H,03E13H,02E32H,01E51H,00E70H
                DW      0FF9FH,0EFBEH,0DFDDH,0CFFCH,0BF1BH,0AF3AH,09F59H,08F78H
                DW      09188H,081A9H,0B1CAH,0A1EBH,0D10CH,0C12DH,0F14EH,0E16FH
                DW      01080H,000A1H,030C2H,020E3H,05004H,04025H,07046H,06067H
                DW      083B9H,09398H,0A3FBH,0B3DAH,0C33DH,0D31CH,0E37FH,0F35EH
                DW      002B1H,01290H,022F3H,032D2H,04235H,05214H,06277H,07256H
                DW      0B5EAH,0A5CBH,095A8H,08589H,0F56EH,0E54FH,0D52CH,0C50DH
                DW      034E2H,024C3H,014A0H,00481H,07466H,06447H,05424H,04405H
                DW      0A7DBH,0B7FAH,08799H,097B8H,0E75FH,0F77EH,0C71DH,0D73CH
                DW      026D3H,036F2H,00691H,016B0H,06657H,07676H,04615H,05634H
                DW      0D94CH,0C96DH,0F90EH,0E92FH,099C8H,089E9H,0B98AH,0A9ABH
                DW      05844H,04865H,07806H,06827H,018C0H,008E1H,03882H,028A3H
                DW      0CB7DH,0DB5CH,0EB3FH,0FB1EH,08BF9H,09BD8H,0ABBBH,0BB9AH
                DW      04A75H,05A54H,06A37H,07A16H,00AF1H,01AD0H,02AB3H,03A92H
                DW      0FD2EH,0ED0FH,0DD6CH,0CD4DH,0BDAAH,0AD8BH,09DE8H,08DC9H
                DW      07C26H,06C07H,05C64H,04C45H,03CA2H,02C83H,01CE0H,00CC1H
                DW      0EF1FH,0FF3EH,0CF5DH,0DF7CH,0AF9BH,0BFBAH,08FD9H,09FF8H
                DW      06E17H,07E36H,04E55H,05E74H,02E93H,03EB2H,00ED1H,01EF0H
                RET
;
;
;*************************************************
;*                                               *
;*                  校验包的 CRC                 *
;*                                               *
;*************************************************
;CRCHI AND CRCLO 内容为 0：OK          ELSE：FAIL
XCRCCHECK:      MOV     R0,#XBUFFSTART          ;
                INC     R0                      ;
                INC     R0                      ;
                INC     R0                      ;
                CLR     A                       ;
                MOV     CRCLO,A                 ;
                MOV     CRCHI,A                 ;
                MOV     A,#BLKSIZE              ;
                INC     A                       ;
                INC     A                       ;
                MOV     R4,A                    ;计算包长度
XCRCLOOP:       MOV     A,@R0                   ;
                INC     R0                      ;
                XRL     A,CRCHI                 ;
                MOV     DPTR,#CRC16TBL          ;
```

```
        LCALL   DPTRADDC                ;调整 DW 地址
        LCALL   DPTRADDC                ;
        CLR     A                       ;
        MOVC    A,@A+DPTR               ;查表
        XRL     A,CRCLO                 ;
        MOV     CRCH1,A                 ;获得高 CRC
        CLR     A                       ;
        INC     A                       ;
        MOVC    A,@A+DPTR               ;
        MOV     CRCLO,A                 ;获得低 CRC
        DJNZ    R4,XCRCLOOP             ;
        MOV     A,CRCLO                 ;
        JNZ     XCRCERROR               ;
        MOV     A,CRCHI                 ;
        JNZ     XCRCERROR               ;
XCRCERROR:  RET                         ;
;
;
;*****************************************************
;*                                                   *
;*                  校验包的 CheckSum                *
;*                                                   *
;*****************************************************
XCHECKSUM:  MOV     R0,#XBUFFSTART      ;
            INC     R0                  ;
            INC     R0                  ;
            INC     R0                  ;
            CLR     A                   ;
            MOV     CRCHI,A             ;
            MOV     A,#BLKSIZE          ;
            MOV     R4,A                ;计算包长度
XSUMLOOP:   MOV     A,@R0               ;
            ADD     A,CRCHI             ;
            MOV     CRCHI,A             ;
            INC     R0                  ;
            DJNZ    R4,XSUMLOOP         ;
            MOV     A,@R0               ;
            XRL     A,CRCHI             ;
            RET
;
;
;=====================================================
;/////////////////////////////////////////////////////
```

```
;////////////////////////////////////////////////////////////////////////
;=====================================================
;IN-SYSTEM-PROGRAMMING 子程序
;
;*************************************************
;*                                               *
;*                    ERASE CHIP                 *
;*                                               *
;*************************************************
ERASECHIP:  MOV     TIMEH1,#P15MS          ;约 15 ms
            MOV     TIMELO,#0FFH           ;
            MOV     TL0,TIMELO             ;
            MOV     TH0,TIMEHI             ;
            MOV     SFRCN,#22H             ;擦除 64 KB APROM 模式
            SETB    TR0                    ;
            ORL     PCON,#01H              ;
            MOV     DPTR,#ERASETBL         ;显示 OK
            LCALL   DISPINFO               ;
            ANL     PCON,#0FEH             ;
            RET
;
;
;*************************************************
;*                                               *
;*                   BLANK CHIP                  *
;*                                               *
;*************************************************
;A = 0: ERROR          A = FF: OK
BLANKCHIP:  CLR     A                      ;
            MOV     SFRCN,A                ;读 64 KB APROM 模式
            MOV     SFRAH,A                ;
            MOV     SFRAL,A                ;
            CLR     TR0                    ;
            DEC     A                      ;
            MOV     TIMEHI,A               ;约 1.5 μs
            MOV     TIMELO,#P1US           ;
            MOV     TL0,TIMELO             ;
            MOV     TH0,TIMEHI             ;
BLANKLOOP:  SETB    TR0                    ;
            ORL     PCON,#01H              ;调用空闲模式
            MOV     A,SFRFD                ;读一字节
            CJNE    A,#0FFH,BLANKERROR     ;比较是否为空
            INC     SFRAL                  ;下一地址
```

```
        MOV     A,SFRAL                 ;
        JNZ     BLANKLOOP               ;
        LCALL   WATDOG                  ;
        INC     SFRAH                   ;
        MOV     A,SFRAH                 ;
        CJNE    A,#00H,BLANKLOOP        ;结束地址 FFFF
        MOV     DPTR,#BLANKOKTBL        ;成功
        LCALL   DISPINFO                ;
        MOV     A,#0FFH                 ;
        ANL     PCON,#0FEH              ;
        RET
;
BLANKERROR: MOV     DPTR,#BLANKERRORTBL ;失败
        LCALL   DISPINFO                ;
        ANL     PCON,#0FEH              ;
        CLR     A                       ;
        RET     ;
;
;*********************************************
;*                                           *
;*              PROGRAM CHIP                 *
;*                                           *
;*********************************************
;
PROGRAMCHIP:MOV     R3,#03H             ;
REPROGRAM:  MOV     SFRCN,#21H          ;编程 64 KB APROM 模式
        MOV     TIMELO,#P50US           ;约 50 μs
        MOV     TIMEHI,#0FFH            ;
        MOV     TL0,TIMELO              ;
        MOV     TH0,TIMEHI              ;
        MOV     R1,#XBUFFSTART          ;
        INC     R1                      ;
        INC     R1                      ;
        INC     R1                      ;
        MOV     R4,#BLKSIZE             ;
PROGRAMLOOP:MOV     SFRAL,DPLO          ;低字节地址
        MOV     A,@R1                   ;
        MOV     SFRFD,A                 ;读一字节
        SETB    TR0                     ;
        ORL     PCON,#01H               ;
        INC     DPLO                    ;下一地址
        INC     R1                      ;
```

```
                DJNZ    R4,PROGRAMLOOP          ;写一包数据
                LCALL   WATDOG                  ;
                MOV     TIMELO,#P1US            ;初始化校验
                MOV     TIMEHI,#0FFH            ;约 1.5 μs
                MOV     TL0,TIMELO              ;
                MOV     TH0,TIMEHI              ;
                MOV     R1,#XBUFFSTART          ;
                INC     R1                      ;
                INC     R1                      ;
                INC     R1                      ;
                MOV     R4,#BLKSIZE             ;
                MOV     SFRCN,#00H              ;读 64 KB APROM 模式
                MOV     A,DPLO                  ;
                DEC     A                       ;
                SUBB    A,#7FH                  ;
                MOV     R2,A                    ;计算校验起始地址
VERIFYLOOP:     MOV     SFRAL,R2                ;
                SETB    TR0                     ;
                ORL     PCON,#01H               ;
                MOV     A,@R1                   ;
                INC     R1                      ;
                INC     R2                      ;
                CJNE    A,SFRFD,PROGRAMERROR    ;比较
                DJNZ    R4,VERIFYLOOP           ;
                MOV     A,DPLO                  ;
                JNZ     PROGRAMOUT              ;
                INC     DPHI                    ;高字节地址
                MOV     SFRAH,DPHI              ;
PROGRAMOUT:     CLR     A                       ;成功
                DEC     A                       ;
                LCALL   WATDOG                  ;
                ANL     PCON,#0FEH              ;
                RET     ;
;
PROGRAMERROR:   LCALL   WATDOG                  ;
                DJNZ    R3,REPROGRAM            ;出错重试三次
                CLR     A                       ;
                ANL     PCON,#0FFH              ;
                RET     ;
;
;
;*************************************************************
;/////////////////////////////////////////////////////////////
```

```
;////////////////////////////////////////////////////////////////////////
;*********************************************************************
;系统显示信息
;
;*********************************************
;*                                           *
;*                系统显示信息               *
;*                                           *
;*********************************************
COMMANDTBL: DB      "?",    00DH,00AH,0FFH,0FFH,0FFH,0FFH,0FFH
            DB      00DH,   00AH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
            DB      "RAM",  00DH,00AH,0FFH,0FFH,0FFH
            DB      "SEC",  00DH,00AH,0FFH,0FFH,0FFH
            DB      "SETB", 00DH,00AH,0FFH,0FFH
            DB      "INFO", 00DH,00AH,0FFH,0FFH
            DB      "ERASE",00DH,00AH,0FFH
            DB      "BLANK",00DH,00AH,0FFH
            DB      "RESET",00DH,00AH,0FFH
            DB      "UPDAT",00DH,00AH,0FFH
            DB      0FFH,0FFH,0FFH,0FFH
            RET
;
LOGOTBL:    DB      "      MCS-51 Updata Code Ver 1.00",00DH,00AH
            DB      "           Input ? to Help",0FFH
            RET
;
ROMMARKTBL: DB      00DH,00AH,"ROM>",0FFH
            RET
;
HELPTBL:    DB      00DH,00AH
 DB "*************************************************",00DH,00AH
 DB "*     ?     Online help command                   *",00DH,00AH
 DB "*   RAM     View mcu interram     * SEC    Security mcu code    *",00DH,00AH
 DB "*   SETB    Set mcu baud rate     * INFO   View mcu info        *",00DH,00AH
 DB "*   ERASE   Erase mcu rom         * BLANK  View mcu rom blank   *",00DH,00AH
 DB "*   RESET   Reset mcu             * UPDAT  Updata mcu romcode   *",00DH,00AH
 DB "*************************************************",0FFH,0FFH
            RET
;
COMMERRORTBL:DB     00DH,00AH,"Bad command",0FFH
            RET
;
WARNINGTBL: DB      00DH,00AH,"Are you sure (Y/N)",0FFH
```

```
                RET
SECTBL:         DB      00DH,00AH,"Are you security (Y/N)",0FFH
                RET
INPUTTBL:       DB      00DH,00AH,"Input:  ",0FFH
                RET
;
SYSINFOTBL:     DB      00DH,00AH,"Winbond W78E516B 512B/64 KB"
                DB      00DH,00AH,"          Ver 1.00",0FFH
                RET
;
BAUDINFOTBL:    DB      00DH,00AH,"    Baud rate 38 400",00DH,00AH,0FFH
                DB      00DH,00AH,"    Baud rate 19 200",00DH,00AH,0FFH
                DB      00DH,00AH,"    Baud rate 9 600 ",00DH,00AH,0FFH
                DB      00DH,00AH,"    Baud rate 4 800 ",00DH,00AH,0FFH
                RET
SETBFUNTBL:     DB      00DH,00AH,"0) 38 400 1) 19 200 2)9 600 3)4 800 4)Exit Input:  ",
0FFH
                RET
SETERRORTBL:    DB      00DH,00AH,"Error !!!    Input:  ",0FFH
                RET
SFRNAMETBL:     DB    00DH,00AH,"  PSW  = ",0FFH
                DB              "  IE   = ",0FFH,0FFH,0FFH
                DB              "  TCON  = ",0FFH,0FFH
                DB              "  TMOD  = ",0FFH,0FFH
                DB    00DH,00AH,"  SCON = ",0FFH
                DB              "  SBUF = ",0FFH,0FFH,0FFH
                DB              "  T2CON = ",0FFH,0FFH
                DB              "  T2MOD = ",0FFH,0FFH
                DB    00DH,00AH,"  PCON = ",  0FFH
                DB              "  SP   = ",  0FFH,0FFH,0FFH
                DB              "  DPL   = ",0FFH,0FFH
                DB              "  DPH   = ",0FFH,0FFH
                DB    00DH,00AH,"  P0   = ",  0FFH
                DB              "  P1   = ",  0FFH,0FFH,0FFH
                DB              "  P2    = ",0FFH,0FFH
                DB              "  P3    = ",0FFH,0FFH
                RET
UPDATAOKTBL:    DB    00DH,00AH,"   Updata Ok !!! ",0FFH
                RET
UPDATAERRORTBL: DB    00DH,00AH,"   Updata Error !!! ",0FFH
                RET
ERASETBL:       DB    00DH,00AH,"   Erase Ok !!! ",0FFH
                RET
```

```
BLANKOKTBL:     DB      00DH,00AH,"    Blank Ok !!! ",0FFH
                RET
BLANKERRORTBL:  DB      00DH,00AH,"    Blank Error !!! ",0FFH
                RET
VERIFYOKTBL:    DB      00DH,00AH,"    Verify Ok !!! ",0FFH
                RET
VERIFYERRORTBL: DB      00DH,00AH,"    Verify Error !!! ",0FFH
                RET
SECOKTBL:       DB      00DH,00AH,"    Security Ok !!! ",0FFH
                RET
SECERRORTBL:    DB      00DH,00AH,"    Security Error !!! ",0FFH
                RET
;*************************************************
;*                    结  束                     *
;*************************************************
                END
```

应用程序(含在线编程控制程序)

```
;                  ****************************
;                  *     可在系统修改程序     *
;                  *     电子屏字符显示器     *
;                  *       "电子设计"         *
;                  *     2001.10.23   LRM     *
;                  ****************************
;          4 个显示字符数据表以在 50H~6FH 单元内,字符用 8×8 点阵,R4(30H)用于
;          控制显示静止字的时间,R5(31H)静止字显示跳转地址步距,B 内放显示首址
;
T2CON      EQU     0C8H    ;T2 控制寄存器
T2MOD      EQU     0C9H    ;
TL2        EQU     0CCH    ;T2 计数寄存器低字节
TH2        EQU     0CDH    ;T2 计数寄存器高字节
TR2        EQU     0CAH    ;T2 启动位
RCAP2L     EQU     0CAH    ;T2 计数重载寄存器低字节
RCAP2H     EQU     0CBH    ;T2 计数重载寄存器高字节
CHPCON     EQU     0BFH    ;在系统编程控制寄存器
CHPENR     EQU     0F6H    ;编程状态下 MTP ROM 的控制字节寄存器
SFRAL      EQU     0C4H    ;编程状态下的目标低地址
SFRAH      EQU     0C5H    ;编程状态下的目标高地址
SFRFD      EQU     0C6H    ;编程状态下 MTP ROM 的编程数据
SFRCN      EQU     0C7H    ;
TXOKFLAG   EQU     003H
```

```
;
            ORG     0000H
            LJMP    START
;
;************;
; 中断入口程序 ;
;************;
;
            ORG     0003H
            RETI
            ORG     000BH
            RETI
            ORG     0013H
            RETI
            ORG     001BH
            RETI
            ORG     0023H
            LJMP    INTS
            ORG     002BH
            RETI
;
;初始化
CLEARMEN:   MOV     PSW,#00H            ;设第一组寄存器
            MOV     SP,#0F1H            ;设置堆栈指针
            MOV     SCON,#01010000B     ;串口工作方式 1(8 BIT UART)允许接收
            MOV     T2CON,#00110000B    ;T2CON
            MOV     A,#0ECH
            MOV     TL2,A               ;设置波特率 (19 200)
            MOV     RCAP2L,A            ;
            MOV     A,#0FFH
            MOV     TH2,A               ;
            MOV     RCAP2H,A            ;
            SETB    ES                  ;允许串口中断
            MOV     IP,#00H             ;低优先级
            SETB    TR2                 ;启动定时计数器 2
            SETB    REN                 ;启动串口接收中断
            CLR     TI                  ;清串口发送中断标志位
            CLR     RI                  ;清串口接收中断标志位
            SETB    EA                  ;开放所有中断
            RET
CLEARMEN1:  MOV     A,#0FFH
            MOV     P1,A
            MOV     P2,A
```

```
        MOV     P3,A
        MOV     P0,A
        MOV     DPTR,#TAB2
        CLR     A
        MOV     20H,A
        MOV     21H,A
        MOV     22H,A
        MOV     23H,A
        MOV     R3,A
        MOV     R1,#50H
        MOV     R2,#20H
CLLOOP: MOVC    A,@A+DPTR
        MOV     @R1,A
        MOV     A,R3
        INC     A
        MOV     R3,A
        INC     R1
        DJNZ    R2,CLLOOP
        MOV     30H,#0A0H
        MOV     31H,#08H
        RET
;
START:  LCALL   CLEARMEN            ;初始化
        LCALL   CLEARMEN1
START1: LCALL   DISP1
        AJMP    START1
PUTPASS: MOV    DPTR,#PASSTBL       ;
        LCALL   DISPINFO            ;显示"Pass Error"
        AJMP    LOOP3               ;
APROM:  LJMP    APROMOUT            ;
;
;
DISP1:  MOV     B,#50H
        MOV     R4,30H
        MOV     R5,31H
LOOP:   JBC     04H,APROM           ;进入 4 KB—LDROM 模式处理程序
        JBC     05H,PUTPASS         ;口令提示处理程序
LOOP3:  LCALL   DISPLAY
        DJNZ    R4,LOOP
        MOV     R4,30H
        MOV     A,B
        CJNE    A,#68H,CONT
        RET
```

```
CONT:       ADD     A,R5
            MOV     B,A
            AJMP    LOOP
;
;
DISPLAY:    MOV     A,#0FFH
            MOV     P0,A
            MOV     P2,A
            MOV     R6,#0FEH
            MOV     R0,B
            MOV     R7,#08H
DISLOOP:    MOV     A,@R0
            MOV     P0,A
            MOV     P2,R6
            LCALL   DL1MS
            INC     R0
            MOV     A,R6
            RL      A
            MOV     R6,A
            DJNZ    R7,DISLOOP
            RET
;
;
DL1MS:      MOV     R3,#0FFH                        ;256 * 4
LOOPK:      NOP
            NOP
            DJNZ    R3,LOOPK
            RET
;
;
TAB1:       DB      0EFH,83H,0ABH,83H,0ABH,83H,0EEH,0E0H        ;电
            DB      0FFH,0C7H,0EFH,83H,0EFH,0EFH,0CFH,0EFH      ;子
            DB      0B1H,0B5H,04H,0BFH,0B1H,0B5H,9BH,0A4H       ;设
            DB      0BBH,0BBH,1BH,0A0H,0BBH,0BBH,9BH,0BBH       ;计
            DB      00H,00H,00H,00H
;
TAB2:       DB      0F7H,0EFH,0C1H,0D5H,000H,0D5H,0D9H,0BDH     ;舟
            DB      0FFH,0F7H,0F7H,0F7H,0D5H,0D5H,0C1H,0FFH     ;山
            DB      0F7H,0EFH,0C1H,0D5H,000H,0D5H,0D9H,0BDH     ;舟
            DB      0FFH,0F7H,0F7H,0F7H,0D5H,0D5H,0C1H,0FFH     ;山
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;          进入 4 KB  LDROM 的程序          ;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INTS:         PUSH PSW                      ;堆栈保护
              PUSH ACC                      ;
              PUSH B                        ;
              MOV    PSW,#08H               ;启用第1组寄存器
              JBC    RI,RXINTS              ;RI为1,接收中断,RI清0
              JNB    TI,INTSOUT             ;TI为0,中断退出
              CLR    TI                     ;TI为1,先清0
              SETB TXOKFLAG                 ;设发送OK标志
              LJMP INTSOUT                  ;中断退出
;
RXINTS:       MOV    A,SBUF                 ;接收中断处理。
              XRL    A,#0DH                 ;是否回车符
              JNZ    PASSFLAG               ;不是回车,口令出错处理
              SETB   04H                    ;进入4 KB-LDROM标志置1
INTSOUT:      POP    B                      ;出栈
              POP    ACC                    ;
              POP    PSW                    ;
              RETI                          ;中断退出
PASSFLAG:     SETB   05H                    ;口令出错标志置1
              LJMP   INTSOUT                ;中断退出
;
;口令出错提示表
PASSTBL:      DB     "  pass ERROR !   ",0DH,0AH,0FFH,0FFH,0FFH,0FFH,0FFH
              RET
;DPTR内容是表的首址
;显示表格内容
DISPINFO:     CLR    A                      ;
              MOV    R2,A                   ;
READDISPCODE: MOV    A,R2                   ;
              MOVC   A,@A+DPTR              ;查表
              MOV    R1,A                   ;
              INC    R2                     ;计数器加1
              MOV    A,R2                   ;
              JNZ    READOVER               ;计数器溢出处理
              INC    DPH                    ;
READOVER:     MOV    A,R1                   ;
              INC    A                      ;
              JZ     DISPINFOOUT            ;读到结束符结束
              CLR    TXOKFLAG               ;
              DEC    A                      ;
              LCALL  PUTCHAR                ;发送字符
              LJMP   READDISPCODE           ;
```

```
DISPINFOOUT:RET
;A 内容待发字符
PUTCHAR:    CLR     TXOKFLAG            ;
            MOV     SBUF,A              ;
            JNB     TXOKFLAG,$          ;
            RET

APROMOUT:   MOV     CHPENR,#87H         ;进入 LDROM 方式
            MOV     CHPENR,#59H
            MOV     CHPCON,#03H
            MOV     TCON,#00H           ;关 T0、T1
            MOV     T2CON,#00H          ;关 T2
            MOV     IP,#00H             ;同优先级
            MOV     IE,#82H             ;允许 T0 中断,总中断开放
            MOV     TL0,#0FBH           ;装初值,定时为 5 μs
            MOV     TH0,#0FFH
            MOV     TMOD,#01H           ;T0 为 16 位定时器
            MOV     TCON,#10H           ;开启 T0
            MOV     PCON,#01H           ;进入在线编程模式,等待唤醒后进入 4 KB--
                                         LDROM 程序
            END     ;程序结束
```

第 18 章　实例 13　电子定时器的设计

本电子定时器能定时给电器供电或断电，最大时间可以长达 30 h，操作使用方便，采用 AT89C2051 单片机控制，4 位共阳数码管显示时间，继电器作电器电源输出控制，其电路如图 18.1 所示。

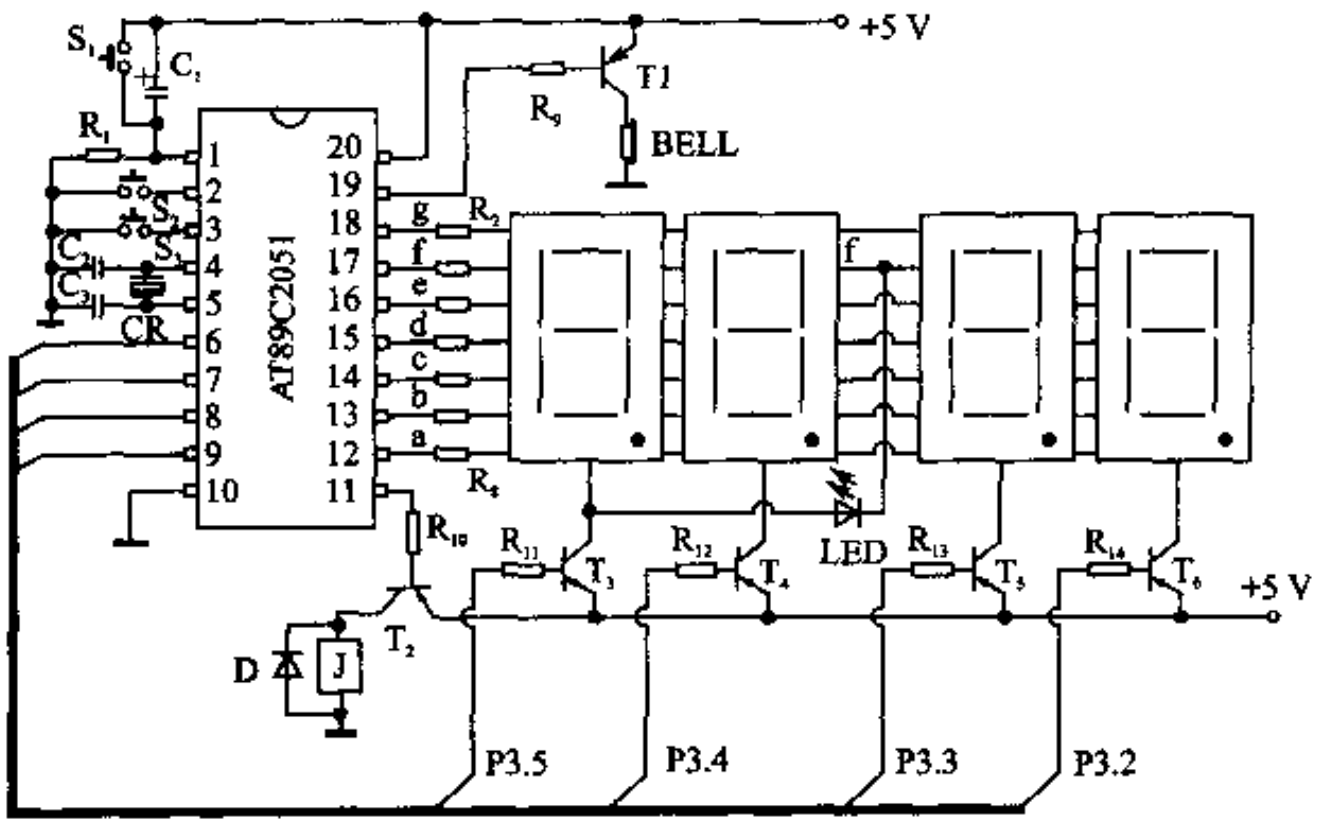


图 18.1　电子定时器电气原理图

1. 系统硬件电路的设计

(1) 芯片的选择

硬件电路要实现对交流大电流电源的控制、定时时间的设定显示和到点提醒等功能。若采用 40 脚单片机有利于设计，但会增大电路板的体积。本设计采用 ATMEL 公司的 AT89C2051 单片机，芯片为 20 脚，体积小，工作电压范围宽(2.7 V～6 V)。

(2) 交流控制接口电路

本设计采用继电器控制。也可采用可控硅控制等。

(3) 显示电路

显示电路采用 4 个 LED 数码管。为了在定时精度达到分(钟)的时候能显示出时钟在计时，两个数码管之间增加了一个发光二极管，以其闪烁来代表秒走动；为了使硬件显示电路简单，采用单片机直接驱动 LED 数码管(AT89C2051 输出口能吸收 20 mA 电流)，用动态扫描法实现 LED 显示。

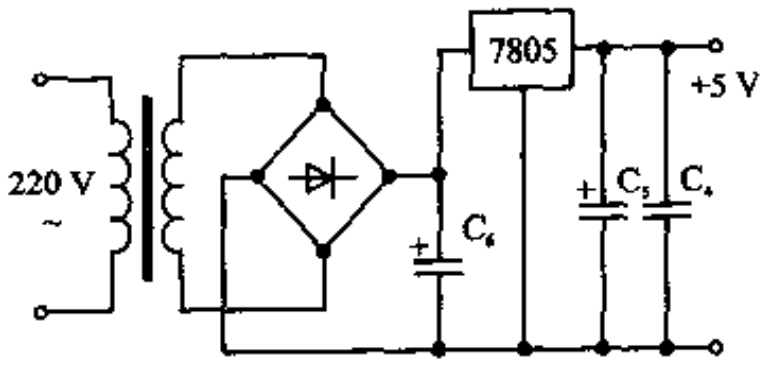


图 18.2　电源电路

(4) 电源电路

电源电路采用普通三端集成稳压电路，如图 18.2 所示。

(5) 报警电路

报警电路采用普通 5 V 成品小蜂鸣器。

2. 系统主要程序的设计

程序采用模块化、结构化设计，并采用了软件抗干扰技术，其软件的可靠性较好，可维护性强。其主要程序模块有：

(1) 主程序

主程序有 3 个状态：待命状态、计时工作状态和到点工作状态。

(2) 菜单(设置)程序

菜单程序完成定时方式和定时时间的设定。

(3) 到点工作程序

到点工作程序根据所选定的不同定时方式，作不同的处理。

(4) 抗干扰(出错)程序

程序跑飞时能被软件陷阱捕获，被抗干扰程序处理，返回复位状态，重新启动系统。

3. 主程序流程图

主程序流程图如图 18.3 所示。

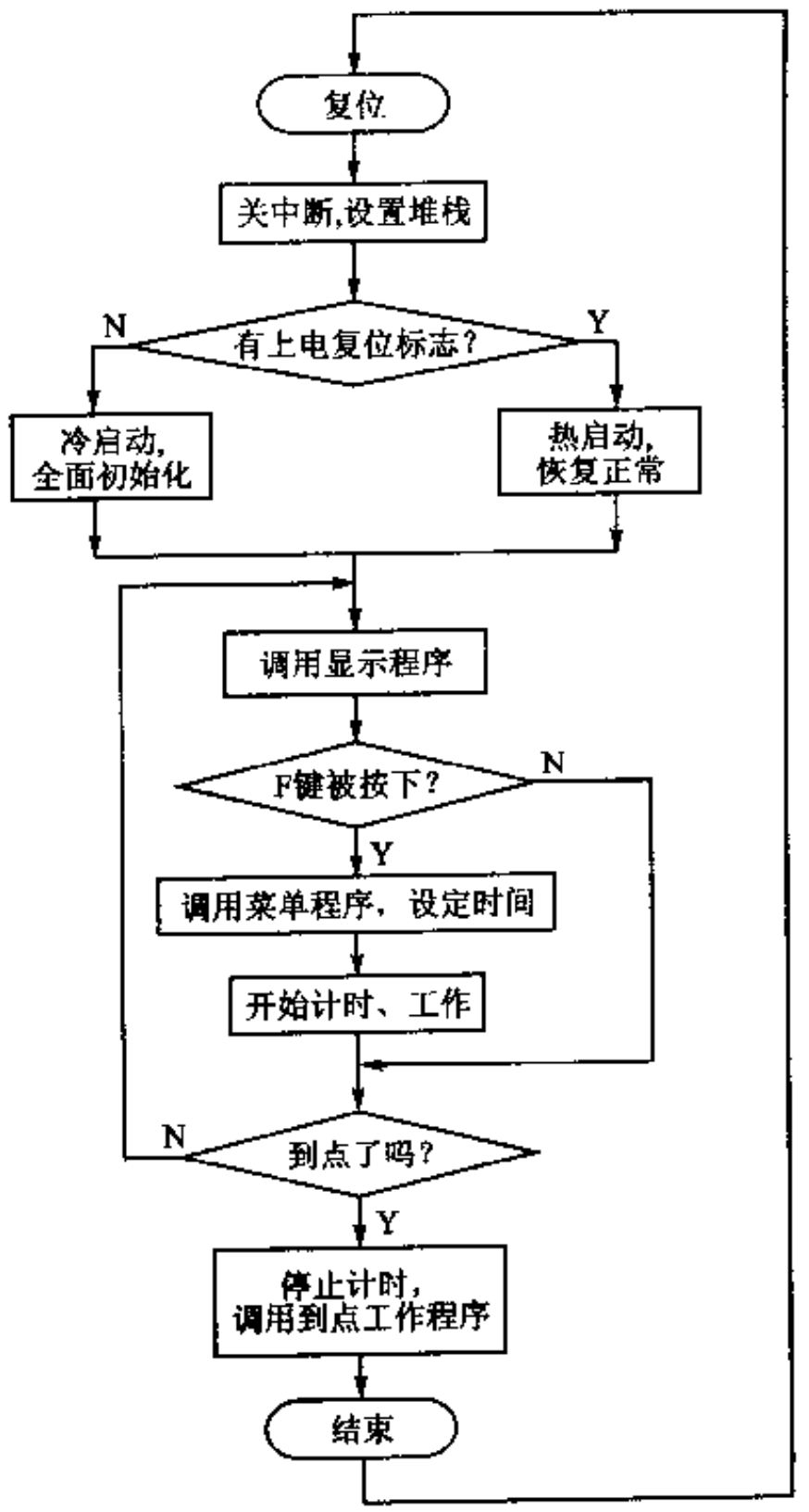


图 18.3　主程序流程图

4. 操作说明

(1) 通电以后，显示窗显示"----"，表示机器处于待命状态。

(2) 按功能键(F 键)，显示窗显示"F-01"，进入功能设置，此时按修改键(M 键)选择不同定时方式：

方式 1：定时关电源，定时范围为 1 s～30 min 59 s；

方式 2：定时关电源，定时范围为 1 min～30 h 59 min；

方式 3：定时开电源，定时范围为 1 s～30 min 59 s；

方式 4：定时开电源，定时范围为 1 min～30 h 59 min。

(3) 选定定时方式之后，再按功能键，进入时间设定。先设定秒(方式 1、方式 3)或分(方式 2、方式 4)，再设定分(方式 1、方式 3)或小时(方式 2、方式 4)。按修改键增加数字，按功能键确定，按住修改键不放可快进。

(4) 设定时间完毕，按下功能键，定时器长鸣一声，开始工作。

(5) 定时时间到了，在定时方式 1 和方式 2，定时器会鸣叫，此时按任意键可停止鸣叫。

(6) 在任何情况下，按下复位键(R 键)，定时器重新回到待命状态。

以下是应用系统控制源程序：

```
;********************
;       定时器程序      ;
;      采用 89C2051     ;
;        2001.10        ;
;********************
;
;*********************
;          伪定义          ;
;*********************
          SL        EQU   30H          ;SL 存放秒的个位数
          SH        EQU   31H          ;SH 存放秒的十位数
          ML        EQU   32H          ;ML 存放分的个位数
          MH        EQU   33H          ;MH 存放分的十位数
          HL        EQU   34H          ;HL 存放小时的个位数
          HH        EQU   35H          ;HH 存放小时的十位数
          ;
          L0        EQU   36H          ;L0～L3：显示数据存储器
          L1        EQU   37H
          L2        EQU   38H
          L3        EQU   39H
          DSPLYP    EQU   3AH          ;显示数据指针(DISPLAY－POINT)
          PLYTS     EQU   3BH          ;显示次数计数器(DISPLAY－TIMES)
          ;
          LPLMOD    BIT   39H          ;低两位显示方式(LOW－PLAY－MOD)
          HPLMOD    BIT   3AH          ;高两位显示方式(HIGH－PLAY－MOD)
          BRIGHT    BIT   3BH          ;DISPLAY 子程序参数：亮/灭指示位
```

```
;
TCOUNT  EQU   3CH              ;时间计数器(TIME－COUNT)
;
ADDRES  EQU   3DH              ;加 1 子程序参数
MAX     EQU   3EH              ;加 1 子程序参数
IFDEC   BIT   20H              ;BCD 加法子程序参数
;
R_MOD   EQU   3FH              ;响铃方式参数
;
LED4    BIT   30H              ;发光管状态位
BELL    BIT   P1.7             ;蜂鸣器
SWITCH  BIT   P3.7             ;继电器
FKEY    BIT   P3.0             ;功能键(S1)
MKEY    BIT   P3.1             ;修改键(S2)
;
WORKIN  BIT   38H              ;工作状态指示位
;

;********************
;       中断入口       ;
;********************
        ORG     0000H
START:  LJMP    MAIN           ;0000H 引向主程序
        LJMP    ERR            ;0003H
        NOP
        NOP
        LJMP    ERR            ;引向出错处理程序
        LJMP    PGT0           ;000BH 引向中断处理程序 PGT0
        NOP
        NOP
        LJMP    ERR            ;引向出错处理程序
        LJMP    ERR            ;0013H INT1
        NOP
        NOP
        LJMP    ERR
        LJMP    ERR            ;001BH T1
        NOP
        NOP
        LJMP    ERR
        LJMP    ERR            ;0023H
        NOP
        NOP
        LJMP    ERR
```

```
        LJMP    ERR                     ;002BH
        NOP
        NOP
        ;
;************
;   主程序   ;
;************
MAIN:   MOV     IE, #00H                ;关中断
        MOV     SP, #57H                ;设置堆栈指针
        MOV     PSW, #00H               ;选用寄存器组 0
        MOV     TMOD, #11H              ;设定中断工作方式为 T0 和 T1
;
        MOV     A,56H
        CJNE    A, #0AAH,CSTART         ;判断上电复位标志,无标志转冷启动
        MOV     A,57H
        CJNE    A, #55H,CSTART          ;无标志转冷启动
        AJMP    HSTART                  ;有上电复位标志转热启动
        NOP
        NOP
        LJMP    ERR                     ;软件陷阱,引向出错处理程序
CSTART: MOV     P1, #0FFH               ;冷启动,全面初始化
        MOV     P3, #0FFH
        MOV     TCON, #00H              ;计时停止
        MOV     TL0, #0B0H              ;赋中断 T0 初值
        MOV     TH0, #3CH
        MOV     TCOUNT, #0AH            ;赋定时器初值
        MOV     R5, #00H                ;R5 为一空单元(备用)
        MOV     R4, #00H                ;R4 为工作模式选择寄存器
        MOV     SL, #00H                ;定时单元清零
        MOV     SH, #00H                ;秒
        MOV     ML, #00H                ;
        MOV     MH, #00H                ;分
        MOV     HL, #00H                ;
        MOV     HH, #00H                ;时
        MOV     PLYTS, #64H             ;赋显示次数初值为 100 次
        MOV     DSPLYP, #L0             ;显示指针指向显存单元
        MOV     L0, #0AH                ;送显示数据"------"
        MOV     L1, #0AH
        MOV     L2, #0AH
        MOV     L3, #04H
        SETB    LED4                    ;LED4 为数码管之间的发光二极管
        SETB    LPLMOD                  ;设定显示方式为闪烁
        SETB    HPLMOD                  ;
```

```
        SETB    BRIGHT          ;允许显示
        CLR     WORKIN          ;清工作标志,待命
        AJMP    SETUP           ;转开始工作
        NOP
        NOP
        LJMP    ERR             ;软件陷阱
HSTART: MOV     SCON,#00H       ;有上电标志,热启动,清串行口控制寄存器
        MOV     IP,#00H         ;清中断优先控制寄存器
        SETB    FKEY            ;重设按键
        SETB    MKEY
        SETB    EA              ;开中断
        AJMP    BEGIN           ;转向继续工作
        NOP
        NOP
        LJMP    ERR             ;软件陷阱
SETUP:  SETB    EA              ;开中断
;
MAIN1:  ACALL   DISPLY          ;调用显示
        JB      FKEY,JUDGE      ;按键扫描
        ACALL   KEYDLY          ;延时消抖动
        JB      FKEY,JUDGE      ;无键按下转向判断是否到点
        CLR     ET0             ;功能键被按下
        CLR     TR0             ;暂停计时
        ACALL   MENU            ;调用菜单设置程序
BEGIN:  SETB    WORKIN          ;置工作标志位,开始工作
        SETB    ET0             ;开中断
        SETB    TR0             ;开始计时
        MOV     A,R4            ;移入工作模式选择
        RL      A               ;指针放大
        MOV     DPTR,#M_TAB
        JMP     @A+DPTR         ;根据工作模式跳转到相应程序段
M_TAB:  AJMP    WORK1
        AJMP    WORK2
        AJMP    WORK3
        AJMP    WORK4
        NOP
        NOP
        LJMP    ERR             ;软件陷阱
WORK1:  ;
WORK2:  CLR     SWITCH          ;工作方式 1 和 2:开继电器
        AJMP    MAIN2
        NOP
        NOP
```

```
        LJMP    ERR             ;软件陷阱
WORK3:  ;
WORK4:  SETB    SWITCH          ;工作方式 3 和 4:不开继电器
MAIN2:  CLR     BELL            ;蜂鸣器短鸣一声,以示开始工作
        ACALL   DL05S
        SETB    BELL
JUDGE:  JNB     WORKIN,MAIN1    ;判断是否在定时之中
        MOV     A,SL            ;判断秒是否为零
        JNZ     MAIN1
        MOV     A,SH            ;判断秒是否为零
        JNZ     MAIN1
        MOV     A,ML            ;判断分是否为零
        JNZ     MAIN1
        MOV     A,MH            ;判断分是否为零
        JNZ     MAIN1
        MOV     A,HL            ;判断时是否为零
        JNZ     MAIN1
        MOV     A,HH            ;判断时是否为零
        JNZ     MAIN1           ;若时、分、秒全为零
        CLR     ET0             ;停止计时
        CLR     TR0             ;
        ACALL   ACTION          ;调用到点工作子程序
        AJMP    MAIN            ;返回
        NOP
        NOP
        LJMP    ERR             ;软件陷阱
;
;*******************
;       倒计时程序      ;
;*******************
PGT0:   CLR     EA              ;关中断
        PUSH    ACC             ;保护现场
        PUSH    PSW
        PUSH    DPL
        PUSH    DPH
        MOV     PSW,#08H        ;选用寄存器组 1
        CLR     TR0             ;暂停计时
        MOV     A,#0B7H         ;中断同步修正
        ADD     A,TL0
        MOV     TL0,A
        MOV     A,#3CH
        ADD     A,TH0
        MOV     TH0,A
```

```
        SETB    TR0             ;恢复计时
        DEC     TCOUNT          ;定时器 T0 每 50 000 ms 溢出一次
        MOV     A,TCOUNT        ;溢出 10 次为 0.5 s
        JNZ     OUTT0           ;判断是否到 0.5 s
        MOV     TCOUNT,#0AH     ;
        CPL     LED4            ;若到 0.5 sLED 取反
        JNB     LED4,OUTT0      ;LED 每闪烁一次是 1 s
        MOV     R0,#SH          ;移入秒位的地址
        SETB    IFDEC           ;BCD 子程序参数,使其做减法
        ACALL   ADDBCD          ;调用 BCD 子程序,秒减 1
        CJNE    R3,#99H,OUTT0   ;判断秒是否要借位
        MOV     SH,#05H         ;要借位则送数据 59(否则显示 99)
        MOV     SL,#09H
        MOV     R0,#MH          ;移入分位的地址
        ACALL   ADDBCD          ;分减 1
        CJNE    R3,#99H,OUTT0   ;判断分是否要借位
        MOV     MH,#05H
        MOV     ML,#09H
        MOV     R0,#HH
        ACALL   ADDBCD
OUTT0:  POP     DPH             ;恢复现场
        POP     DPL
        POP     PSW
        POP     ACC
        SETB    EA
        RETI                    ;中断返回
        NOP
        NOP
        LJMP    ERR             ;软件陷阱
;
;
;*****************************
;    BCD 子程序(加 1 或减 1)    ;
;*****************************
ADDBCD: MOV     A,@R0           ;移入被操作数的高位
        DEC     R0              ;指针减 1
        SWAP    A
        ORL     A,@R0           ;移入被操作数的低位
        MOV     B,#01H          ;B 寄存器送立即数#01H
        MOV     C,IFDEC         ;减法标志位为 1
        MOV     B.3,C           ;
        MOV     B.4,C           ;B 寄存器的值被改为#99H
        MOV     B.7,C           ;
```

```
        ADD     A,B             ;对一个压缩的 BCD 码加 #99H 等于对其减 1
        DA      A               ;BCD 码调整
        MOV     R3,A            ;暂存结果
        ANL     A,#0FH          ;取低位码
        MOV     @R0,A           ;存数
        MOV     A,R3            ;取回结果
        INC     R0              ;指针加 1
        SWAP    A               ;交换
        ANL     A,#0FH          ;取结果数的高位
        MOV     @R0,A           ;存数
        RET
        NOP
        NOP
        LJMP    ERR             ;软件陷阱
;
;***************
;     加 1 程序    ;
;***************
ADDONE: MOV     R0,ADDRES       ;移入被加数单元的地址
        CLR     IFDEC           ;设定 BCD 子程序做加法
        ACALL   ADDBCD          ;调用 BCD 子程序
        CLR     C               ;判断被加数是否大于
        MOV     A,R3            ;最大值"MAX"
        CJNE    A,MAX,JGOVER
JGOVER: JC      ENDADO
        CLR     A               ;若大于"MAX",则清零
        MOV     @R0,A
        DEC     R0
        MOV     @R0,A
ENDADO: RET
        NOP
        NOP
        LJMP    ERR             ;软件陷阱
;
;********************
;     调时快进程序    ;
;********************
QUICK:  CLR     LPLMOD          ;设定显示方式不闪烁
        CLR     HPLMOD
        ACALL   ADDONE          ;调用加 1 子程序
        MOV     L0,R4           ;将工作模式选择数移入显存
        INC     L0              ;加 1 转化成显示值
        ACALL   KEYDLY          ;延时
```

```
        ACALL   DL100
        JNB     MKEY,QUICK          ;判断键是否松开
        SETB    LPLMOD              ;若松开则恢复闪烁显示方式
        CJNE    R6,#02H,ENDQUK
        SETB    HPLMOD
        CLR     LPLMOD
ENDQUK: RET
        NOP
        NOP
        LJMP    ERR                 ;软件陷阱
;
;*******************
;    功能菜单程序    ;
;*******************
MENU:   MOV     R6,#00H             ;初始化;R6 计功能键按键次数
        MOV     ADDRES,#05H         ;将 R5 的地址送入,以便于改变 R4 的值
        MOV     MAX,#04H            ;定义工作模式选择寄存器 R4 的最大值
        MOV     DSPLYP,#L0          ;定义显示指针指向显存
        SETB    LPLMOD              ;设定低两位数闪烁
        CLR     HPLMOD
        MOV     L0,R4               ;送显示数据"F-0X"
        INC     L0
        MOV     L1,#00H
        MOV     L2,#0BH
        MOV     L3,#00H
        SETB    LED4
;
WAITFK: ACALL   DISPLY              ;等待 F 键释放
        JNB     FKEY,WAITFK
MENU1:  ACALL   DISPLY
        JB      MKEY,JGFKEY         ;M 键扫描
        ACALL   KEYDLY              ;延时消抖动
        JB      MKEY,JGFKEY         ;未按下则转 F 键扫描
        CJNE    R6,#00H,NEXT1       ;
        MOV     SL,#00H             ;若中途改变定时方式,则清除原计时数据
        MOV     SH,#00H
        MOV     ML,#00H
        MOV     MH,#00H
        MOV     HL,#00H
        MOV     HH,#00H
NEXT1:  MOV     R2,#00H             ;R2 用于判断按键时间是否超过 0.5 s
NEXT2:  ACALL   ADDONE              ;调用加 1 程序
```

```
        MOV     L0,R4                   ;移入工作方式选择数
        INC     L0
        ;
WAITMK: ACALL   DISPLY                  ;等待 M 键释放
        INC     R2                      ;R2 自增 1
        CLR     C
        CJNE    R2,#0C8H,JGQUIC         ;若 R2 大于等于 200 则调用快进子程序
JGQUIC: JC      WATMK1
        ACALL   QUICK
WATMK1: JNB     MKEY,WAITMK             ; R2 小于 200 则等待 M 键释放
        ;
JGFKEY: JB      FKEY,MENU1              ;功能(F)键扫描
        ACALL   KEYDLY                  ;延时消抖动
        JB      FKEY,MENU1              ;键未按下则转修改(M)键扫描
        INC     R6                      ;F 键按键次数加 1
        MOV     A,R6                    ;移入按键次数
        RL      A                       ;指针放大
        MOV     DPTR,#FUNTAB
        JMP     @A+DPTR                 ;根据按键次数跳转到相应的程序段
FUNTAB: AJMP    WAITFK
        AJMP    SETLOW
        AJMP    SETHI
        AJMP    ENDMEN
        NOP
        NOP
        AJMP    ENDMEN
;STWKMD:                                ;工作模式设定,不需另外改变菜单
;
SETLOW: MOV     MAX,#60H                ;设置低位(秒位或分位)
        MOV     A,R4                    ;移入工作模式选择数
        RL      A                       ;指针放大
        MOV     DPTR,#FTAB1
        JMP     @A+DPTR                 ;根据工作模式选择数跳转到相应的程序段
FTAB1:  AJMP    SETSS
        AJMP    SETM60
        AJMP    SETSS
        AJMP    SETM60
        NOP
        NOP
        AJMP    WAITFK
SETSS:  MOV     DSPLYP,#SL              ;设定显示区域为 MM:SS
        MOV     ADDRES,#SH
        AJMP    WAITFK
```

```
SETM60: MOV     DSPLYP,#ML          ;设定显示区域为 HH:MM
        MOV     ADDRES,#MH
        AJMP    WAITFK
;
SETHI:  CLR     LPLMOD              ;设置高位(分位或时位)
        SETB    HPLMOD              ;高两位数码管闪烁
        MOV     MAX,#31H            ;最大数为 30
        MOV     A,R4                ;移入工作模式选择数
        RL      A                   ;指针放大
        MOV     DPTR,#FTAB2         ;移入表首地址
        JMP     @A+DPTR             ;根据工作模式选择数跳转到相应的程序段
FTAB2:  AJMP    SETM30
        AJMP    SETHH
        AJMP    SETM30
        AJMP    SETHH
        NOP
        NOP
        AJMP    WAITFK              ;返回等待键释放
SETM30: MOV     ADDRES,#MH          ;移入分位的地址
        AJMP    WAITFK              ;转向等待键释放
SETHH:  MOV     ADDRES,#HH          ;移入时位的地址
        AJMP    WAITFK              ;转向等待键释放
        ;
ENDMEN: CLR     HPLMOD              ;恢复不闪烁显示方式
        RET
        NOP
        NOP
        LJMP    ERR                 ;软件陷阱
;
;*****************
;   到点工作程序   ;
;*****************
ACTION: MOV     L0,R4               ;移入工作模式选择数
        INC     L0                  ;送显示数"F-0X"
        MOV     L1,#00H
        MOV     L2,#0BH
        MOV     L3,#00H
        SETB    LED4
        MOV     DSPLYP,#L0          ;指针指向显存
        SETB    LPLMOD              ;设定显示方式不闪烁
        SETB    HPLMOD
        MOV     A,R4                ;移入工作模式选择数
        RL      A                   ;
```

```
        MOV     DPTR,#A_TAB
        JMP     @A+DPTR                 ;根据工作模式选择数跳转
A_TAB:  AJMP    ACTF1                   ;工作模式一
        AJMP    ACTF1                   ;工作模式二
        AJMP    ACTF3                   ;工作模式三
        AJMP    ACTF3                   ;工作模式四
        NOP
        NOP
        LJMP    ERR                     ;软件陷阱
ACTF1:  SETB    SWITCH                  ;工作模式一(或二):关继电器
        MOV     R_MOD,#82H              ;响铃模式参数#82H
        MOV     R2,#96H                 ;响铃次数参数#96H
ACTF11: ACALL   RING                    ;调用响铃子程序
        JNB     MKEY,ENDACT             ;等待键按下
        JNB     FKEY,ENDACT             ;有键按下则结束响铃
        DJNZ    R2,ACTF11               ;次数未满继续响铃
        MOV     R_MOD,#0FFH             ;参数#0FF 使响铃无效
        AJMP    ACTF11                  ;无键按下返回
        NOP
        NOP
        LJMP    ERR                     ;软件陷阱
ACTF3:  CLR     SWITCH                  ;工作模式三(或四):开继电器
        CLR     BELL                    ;蜂鸣器短鸣一声
        ACALL   DL1S
        SETB    BELL
        MOV     R_MOD,#0FFH             ;响铃模式参数#0FFH
        MOV     R2,#96H                 ;响铃时间参数#96H
ACTF31: ACALL   RING                    ;调用响铃子程序
        JNB     MKEY,ENDACT             ;等待键按下
        JNB     FKEY,ENDACT             ;有键按下则结束
        DJNZ    R2,ACTF31               ;次数未满继续
        MOV     R2,#96H                 ;重新赋值
        CLR     BELL                    ;短鸣一声(说明:响铃模式参数#0FFH 使
                                        ;响铃程序无效,仅起延时作用,
                                        ;每延时一段时间短鸣一声,以
                                        ;提醒使用者继电器仍在工作)
        ACALL   DL100
        SETB    BELL
        AJMP    ACTF31
        NOP
        NOP
        LJMP    ERR                     ;软件陷阱
ENDACT: SETB    SWITCH                  ;关继电器
```

```
AWAITF: ACALL   DISPLY          ;调用显示
        JNB     FKEY,AWAITF     ;等待键释放
AWAITM: ACALL   DISPLY          ;调用显示
        JNB     MKEY,AWAITM     ;等待键释放
        RET
        NOP
        NOP
        LJMP    ERR             ;软件陷阱
;
;***************
;    响铃程序    ;
;***************
RING:   MOV     R5,#18H         ;R5 为循环控制变量
RING1:  JNB     MKEY,R_EXIT     ;键扫描
        JNB     FKEY,R_EXIT     ;有键按下则退出
        MOV     A,R_MOD         ;移入响铃模式参数
        MOV     C,ACC.7         ;根据响铃模式参数改变响铃
        MOV     BELL,C
        RL      A
        MOV     R_MOD,A
        ACALL   DL100           ;延时
        DJNZ    R5,RING1        ;循环次数控制
R_EXIT: SETB    BELL            ;关闭响铃
        RET
        NOP
        NOP
        LJMP    ERR             ;软件陷阱
;
;***************
;    显示程序    ;
;***************
DISPLY: PUSH    ACC             ;数据压栈保护
        PUSH    PSW
        MOV     PSW,#10H        ;选用寄存器组 2
        MOV     R0,DSPLYP       ;移入显示指针
        MOV     R2,#0FDH        ;R2 寄存的是数码管选通数
        MOV     A,PLYTS         ;移入显示循环控制量
        JNZ     PLAY            ;不为 0 则转 PLAY
        MOV     PLYTS,#64H      ;否则从新赋值
        CPL     BRIGHT          ;亮灭指示位取反
PLAY:   DEC     PLYTS           ;显示循环控制量减 1
        JNB     LPLMOD,PLAYL    ;低两位数码管不闪则“PLAYL”
        JB      BRIGHT,PLAYL    ;亮灭指示为 1 也“PLAYL”
```

```
        ACALL   NOPLAY          ;否则灭灯延时
        AJMP    PLAY1           ;转显示高位数码管
        NOP
        NOP
        LJMP    ERR             ;软件陷阱
;用来显示低位
PLAYL:  ORL     P1,#7FH         ;清原显示数据
        ORL     P3,#3CH         ;清原选通数据
        MOV     A,R2            ;移入数码管位选数
        RL      A               ;换一位
        ANL     P3,A            ;选通低位的个位数码管
        MOV     R2,A            ;暂存位选数
        MOV     A,@R0           ;移入显示数值
        MOV     DPTR,#TABLE1    ;移入表首地址
        MOVC    A,@A+DPTR       ;查表
        ANL     P1,A            ;送显示数据
        ACALL   DL1MS           ;延时
        INC     R0              ;指向低位的十位数
        JB      P3.3,PLAYL      ;显示低位的十位数
;
PLAY1:  JNB     HPLMOD,PLAYH    ;高两位数码管不闪则"PLAYH"
        JB      BRIGHT,PLAYH    ;亮灭指示为1也"PLAYH"
        ACALL   NOPLAY          ;否则灭灯延时
        AJMP    OUTPLY          ;转结束
        NOP
        NOP
        LJMP    ERR             ;软件陷阱
;
;用来显示高位
PLAYH:  ORL     P1,#7FH         ;清原显示数据
        ORL     P3,#3CH         ;清原选通数据
        ANL     P3,#0EFH        ;选通高位的个位数数码管
        MOV     A,@R0           ;移入显示数值
        MOV     DPTR,#TABLE1    ;移入表首地址
        MOVC    A,@A+DPTR       ;查表
        ANL     P1,A            ;送显示数据
        ACALL   DL1MS           ;延时
        INC     R0              ;指向高位的十位数
        ;                       ;显示高位的十位数
        ORL     P1,#7FH         ;清原显示数据
        ORL     P3,#3CH         ;清原选通数据
        ANL     P3,#0DFH        ;选通高位的十位数
        MOV     A,@R0           ;移入显示数值
```

```
        MOV     C,LED4              ;指针放大十小灯状态
        RLC     A                   ;
        MOV     DPTR,#TABLE2        ;移入表首地址
        MOVC    A,@A+DPTR           ;查表
        ANL     P1,A                ;送显示数据
        ACALL   DL1MS               ;延时
        ;
OUTPLY: POP     PSW                 ;恢复数据
        POP     ACC
        RET
        NOP
        NOP
        LJMP    ERR                 ;软件陷阱
        ;
TABLE1: DB  0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H,0BFH,8EH,0FFH
        ;       "0","1",    ...     ...                         "9",   "-","F"," "
        ;
TABLE2: DB  0FFH,0DFH,0F9H,0D9H,0A4H,  84H,0B0H,  90H,0BFH,09FH
        ;   " ", " :",  "1", "1:", "2",   "2:", "3", "3:", "-", "-:"
        ;
        ;
        ;
;**********************
;     无显示(灭灯)程序     ;
;**********************
NOPLAY: ORL     P1,#7FH             ;清显示数据
        ORL     P3,#3CH             ;清选通数据
        INC     R0                  ;指针自增 2
        INC     R0
        ACALL   DL1MS               ;延时
        RET
        NOP
        NOP
        LJMP    ERR                 ;软件陷阱
;
;********************
;       延时程序      ;
;********************
DL1MS:  MOV     R3,#0F9H            ;延时 1 250 μs 只为 DISPLY 所调用
DL1MS1: NOP
        NOP
        NOP
        DJNZ    R3,DL1MS1
```

```
        RET
        NOP
        NOP
        LJMP    ERR                 ;软件陷阱
;
KEYDLY: CLR     BELL                ;按键消抖动专用延时程序,
        ACALL   DISPLY              ;在消除抖动的同时发出按键提示音
        SETB    BELL
        RET
        NOP
        NOP
        LJMP    ERR                 ;软件陷阱
;
DL50MS: MOV     R7,#0AH             ;50 ms 延时程序
DL50M1: ACALL   DISPLY              ;每调用一次显示程序 5 ms
        DJNZ    R7,DL50M1           ;调用 10 次
        RET
        NOP
        NOP
        LJMP    ERR                 ;软件陷阱
;
DL100:  ACALL   DL50MS              ;延时 100 ms
        ACALL   DL50MS
        RET
        NOP
        NOP
        LJMP    ERR                 ;软件陷阱
;
DL05S:  ACALL DL100                 ;延时 0.5 s
        ACALL DL100
        ACALL DL100
        ACALL DL100
        ACALL DL100
        RET
        NOP
        NOP
        LJMP    ERR                 ;软件陷阱
;
DL1S:   ACALL DL05S                 ;延时 1 s
        ACALL DL05S
        RET
        NOP
        NOP
```

```
        LJMP    ERR                     ;软件陷阱
;
;*****************************
;       ERR(出错处理)程序       ;
;*****************************
ERR:    CLR     EA                      ;关中断
        MOV     DPTR,#ERR1              ;准备返回地址
        PUSH    DPL                     ;压栈
        PUSH    DPH                     ;
        RETI                            ;中断返回
ERR1:   MOV     56H,#0AAH               ;建立上电标志(出错标志)
        MOV     57H,#55H
        MOV     A,#00H                  ;准备返回地址
        PUSH    A                       ;压栈
        PUSH    A
        RETI                            ;中断返回
        NOP
        NOP
        LJMP    ERR                     ;软件陷阱
;
        ORG     07FAH
        NOP
        NOP
        NOP
        LJMP    ERR                     ;软件陷阱
;
        END                             ;程序结束
```

参 考 文 献

1　何立民. 单片机高级教程　应用与设计. 北京：北京航空航天大学出版社，2000

2　何立民. 单片机中级教程　原理与应用. 北京：北京航空航天大学出版社，2000

3　何立民. MCS－51 单片机应用系统设计. 北京：北京航空航天大学出版社，1990