第5章 单片机原理、接口技术及应用

5.1 单片机概述

5.1.1 单片机的定义与分类

单片机是一种采用超大规模集成电路技术把具有数据处理能力的中央处理器CPU、随机存储器RAM、只读存储器ROM、多种I/O口和中断系统、定时器/计时器等功能(可能还包括显示驱动电路、脉宽调制电路、模拟多路转换器、A/D转换器等电路)集成到一块硅片上构成的一个小而完善的计算机系统,在工业控制领域被广泛应用。从20世纪80年代,由当时的4位、8位单片机,发展到现在的32位的高速单片机。

常用单片机的分类如下。

(1) ATMEL单片机(51单片机):ATMEL公司的8位单片机有AT89、AT90两个系列,AT89系列是8位Flash单片机,与8051系列单片机相兼容,静态时钟模式;AT90系列单片机是增强RISC结构、全静态工作方式、内载在线可编程Flash的单片机,也叫AVR单片机。

(2) STC单片机:STC公司的单片机主要是基于8051内核,是新一代增强型单片机,指令代码完全兼容传统8051,速度快8～12倍,带ADC,4路PWM,双串口,有全球唯一ID号,加密性好,抗干扰强。

(3) PIC单片机:是MICROCHIP公司的产品,其突出的特点是体积小,功耗低,精简指令集,抗干扰性好,可靠性高,有较强的模拟接口,代码保密性好,大部分芯片有其兼容的Flash程序存储器的芯片。

(4) EMC单片机:是台湾义隆公司的产品,有很大一部分与PIC 8位单片机兼容,且相兼容产品的资源相对比PIC的多,价格便宜,有很多系列可选,但抗干扰较差。

(5) PHILIPS 51LPC系列单片机(51单片机):PHILIPS公司的单片机是基于80C51内核的单片机,嵌入了掉电检测、模拟以及片内RC振荡器等功能,这使51LPC在高集成度、低成本、低功耗的应用设计中可以满足多方面的性能要求。

(6) HOLTEK单片机:台湾盛扬半导体的单片机,价格便宜,种类较多,但抗干扰较差,适用于消费类产品。

(7) TI公司单片机(51单片机):德州仪器提供了TMS370和MSP430两大系列通用单片机。TMS370系列单片机是8位CMOS单片机,具有多种存储模式、多种外围接口模式,适用于复杂的实时控制场合;MSP430系列单片机是一种超低功耗、功能集成度较高的

16 位低功耗单片机，特别适用于要求功耗低的场合。

(8) 松翰单片机(SONIX)：是台湾松翰公司的单片机，大多为 8 位机，有一部分与 PIC 8 位单片机兼容，价格便宜，系统时钟分频可选项较多，有 PMW ADC 内振内部杂讯滤波。缺点 RAM 空间过小，但抗干扰较好。

5.1.2　单片机的历史及发展趋势

1971 年 Intel 公司研制出世界上第一个 4 位的微处理器；Intel 公司的霍夫研制成功世界上第一块 4 位微处理器芯片 Intel 4004，标志着第一代微处理器问世，微处理器和微机时代从此开始。因发明微处理器，霍夫被英国《经济学家》杂志列为“二战以来最有影响力的 7 位科学家”之一。

1971 年 11 月，Intel 推出 MCS-4 微型计算机系统(包括 4001 ROM 芯片、4002 RAM 芯片、4003 移位寄存器芯片和 4004 微处理器)。其中 4004 包含 2300 个晶体管，尺寸规格为 3mm×4mm，计算性能远远超过当年的 ENIAC，最初售价为 200 美元。

1972 年 4 月，霍夫等人开发出第一个 8 位微处理器 Intel 8008。由于 8008 采用的是 P 沟道 MOS 微处理器，因此仍属第一代微处理器。

1973 年 Intel 公司研制出 8 位的微处理器 8080；1973 年 8 月，霍夫等人研制出 8 位微处理器 Intel 8080，以 N 沟道 MOS 电路取代了 P 沟道，第二代微处理器就此诞生。

频率为 2MHz 的 8080 芯片运算速度比 8008 快 10 倍，可存取 64KB 存储器，使用了基于 6μm 技术的 6000 个晶体管，处理速度为 0.64MIPS(Million Instructions Per Second)。

1975 年 4 月，MITS 发布第一个通用型 Altair 8800，售价 375 美元，带有 1KB 存储器。这是世界上第一台微型计算机。

1976 年 Intel 公司研制出 MCS-48 系列 8 位的单片机，这也是单片机的问世。

Zilog 公司于 1976 年开发的 Z80 微处理器，广泛用于微型计算机和工业自动控制设备。当时，Zilog、Motorola 和 Intel 在微处理器领域三足鼎立。

20 世纪 80 年代初，Intel 公司在 MCS-48 系列单片机的基础上，推出了 MCS-51 系列 8 位高档单片机。MCS-51 系列单片机无论是片内 RAM 容量，I/O 口功能，系统扩展方面都有了很大的提高。

从单片机的发展历程看，未来单片机技术将向多功能、高性能、高速度、低电压、低功耗、外围电路内装化及片内储存器容量增加的方向发展。

5.1.3　MCS-51 系列单片机

作为主流的单片机品种，MCS-51 系列单片机市场份额占有量巨大，PHILIPS 公司、ATMEL 公司等纷纷开发了以 8051 为内核的单片机产品，这些产品都归属于 MCS-51 单片机系列。

MCS-51 系列单片机的主要产品如图 5.1 所示。

其中，AT89S51 单片机是一种新型的在线可编程的单片机，内部有：4KB、Flash 存储器，它使得单片机产品的软件可在线升级，也使得单片机的学习开发、程序的下载较过去方便许多。

型　　号	制 造 技 术	片内程序存储器	片内数据存储器
8051AH	HMOS	ROM(4KB)	128 字节
8031AH	AHMOS	无	128 字节
8751AH	HMOS	EPROM(4KB)	128 字节
AT89C51/AT89S51	CHMOS	FlashROM(4KB)	128 字节
80C31	CHMOS	无	128 字节
8051	HMOS	ROM(8KB)	256 字节
8031	HMOS	无	256 字节

图 5.1　MCS-51 系列单片机的主要产品

5.1.4　单片机的应用

目前单片机渗透到我们生活的各个领域，几乎很难找到哪个领域没有单片机的踪迹。导弹的导航装置，飞机上各种仪表的控制，计算机的网络通信与数据传输，工业自动化过程的实时控制和数据处理，广泛使用的各种智能 IC 卡，民用豪华轿车的安全保障系统，录像机、摄像机、全自动洗衣机的控制，以及程控玩具、电子宠物等，这些都离不开单片机。更不用说自动控制领域的机器人、智能仪表、医疗器械以及各种智能机械了。因此，单片机的学习、开发与应用将造就一批计算机应用与智能化控制的科学家、工程师。

单片机广泛应用于仪器仪表、家用电器、医用设备、航空航天、专用设备的智能化管理及过程控制等领域，大致可分如下几个范畴。

1. 在智能仪器仪表上的应用

单片机具有体积小、功耗低、控制功能强、扩展灵活、微型化和使用方便等优点，广泛应用于仪器仪表中，结合不同类型的传感器，可实现诸如电压、功率、频率、湿度、温度、流量、速度、厚度、角度、长度、硬度、元素、压力等物理量的测量。采用单片机控制使得仪器仪表数字化、智能化、微型化，且功能比起采用电子或数字电路更加强大。例如精密的测量设备(功率计，示波器，各种分析仪)。

2. 在工业控制中的应用

单片机具有体积小、控制功能强、功耗低、环境适应能力强、扩展灵活和使用方便等优点，用单片机可以构成形式多样的控制系统、数据采集系统、通信系统、信号检测系统、无线感知系统、测控系统、机器人等应用控制系统。例如工厂流水线的智能化管理，电梯智能化控制、各种报警系统，与计算机联网构成二级控制系统等。

3. 在家用电器中的应用

可以这样说，现在的家用电器基本上都采用了单片机控制，从电饭煲、洗衣机、电冰箱、空调机、彩电、其他音响视频器材、再到电子称量设备，五花八门，无所不在。

4. 在计算机网络和通信领域中的应用

现代的单片机普遍具备通信接口，可以很方便地与计算机进行数据通信，为在计算机网络和通信设备间的应用提供了极好的物质条件，现在的通信设备基本上都实现了单片机智能控制，从手机，电话机、小型程控交换机、楼宇自动通信呼叫系统、列车无线通信，再到日常工作中随处可见的移动电话，集群移动通信，无线电对讲机等。

5. 单片机在医用设备领域中的应用

单片机在医用设备中的用途亦相当广泛，例如医用呼吸机、各种分析仪、监护仪、超声诊断设备及病床呼叫系统等。

6. 在各种大型电器中的模块化应用

某些专用单片机设计用于实现特定功能，从而在各种电路中进行模块化应用，而不要求使用人员了解其内部结构。如音乐集成单片机，看似简单的功能，微缩在纯电子芯片中(有别于磁带机的原理)，就需要复杂的类似于计算机的原理。例如：音乐信号以数字的形式存在存储器中(类似于 ROM)，由微控制器读出，转化为模拟音乐电信号(类似于声卡)。

在大型电路中，这种模块化应用极大地缩小了体积，简化了电路，降低了损坏、错误率，也便于更换。

7. 单片机在汽车设备领域中的应用

单片机在汽车电子中的应用非常广泛，例如汽车中的发动机控制器，基于 CAN 总线的汽车发动机智能电子控制器，GPS 导航系统，ABS 防抱死系统，制动系统等。

此外，单片机在工商、金融、科研、教育、国防、航空航天等领域都有着十分广泛的用途。

5.2　MCS-51 单片机

5.2.1　MCS-51 单片机结构

MCS-51 单片机的内部组成如图 5.2 所示。通常采用 DIP 或 PLLD 封装，其内核是 8051 CPU，CPU 的内部集成有运算器和控制器，运算器完成运算操作(包括数据运算、逻辑运算等)，控制器完成取指令、对指令译码以及执行指令。

- 中央处理器；
- 数据存储器(RAM)；
- 程序存储器(ROM)；
- 定时/计数器(ROM)；
- 并行输入输出(I/O)口；
- 全双工串行口；
- 中断系统；
- 时钟电路。

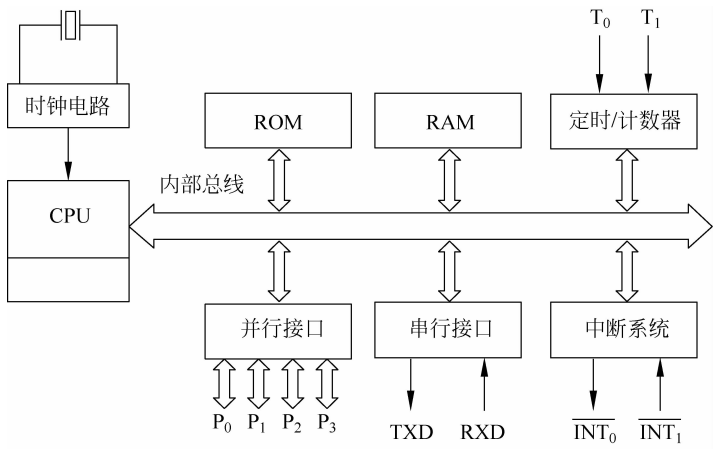


图 5.2 MCS-51 单片机的内部组成

8051 是 MCS-51 系列单片机的典型产品，本章以这一代表性的机型进行系统的讲解。8051 单片机包含中央处理器、程序存储器(ROM)、数据存储器(RAM)、定时/计数器、并行接口、串行接口和中断系统等几大单元及数据总线、地址总线和控制总线等三大总线，现在分别加以说明。

中央处理器：中央处理器(CPU)是整个单片机的核心部件，是 8 位数据宽度的处理器，能处理 8 位二进制数据或代码，CPU 负责控制、指挥和调度整个单元系统协调的工作，完成运算和控制输入输出功能等操作。

数据存储器：8051 内部有 128 个 8 位用户数据存储单元和 128 个专用寄存器单元，它们是统一编址的，专用寄存器只能用于存放控制指令数据，用户只能访问，而不能用于存放用户数据。所以，用户能使用的 RAM 只有 128 个，可存放读写的数据，运算的中间结果或用户定义的字型表。

程序存储器(ROM)：8051 共有 4096 个 8 位掩膜 ROM，用于存放用户程序、原始数据或表格。

定时/计数器(ROM)：8051 有两个 16 位的可编程定时/计数器，以实现定时或计数产生中断用于控制程序转向。

并行输入输出(I/O)口：8051 共有 4 组 32 位双向 I/O 口(P_0、P_1、P_2 或 P_3)，用于对外部数据的传输。

全双工串行口：8051 内置一个全双工串行通信口，用于与其他设备间的串行数据传送，该串行口既可以用作异步通信收发器，也可以当同步移位器使用。

中断系统：8051 具备较完善的中断功能，有两个外中断、两个定时/计数器中断和一个串行中断，可满足不同的控制要求，并具有两级的优先级别选择。

时钟电路：8051 内置最高频率达 12MHz 的时钟电路，用于产生整个单片机运行的脉冲时序，但 8051 单片机需外置振荡电容。

外部引脚可分为输入输出引脚、复位、电源、地、时钟源、外部程序访问和编程等几类，如图 5.3 所示。

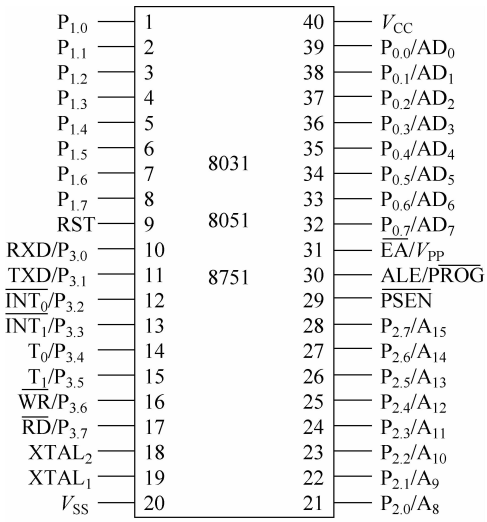


图 5.3　MCS-51 系列单片机引脚

1. 输入输出引脚

- P_0：$P_{0.1}$～$P_{0.7}$，漏极开路双向 I/O，一般为数据总线口。
- P_1：$P_{1.0}$～$P_{1.7}$，准双向 I/O 通道。
- P_2：$P_{2.0}$～$P_{2.7}$，准双向 I/O 通道，一般为地址总线口。
- P_3：$P_{3.0}$～$P_{3.7}$，一般作 I/O 口，具有第二功能，P_3 口的第二功能如表 5.1 所示。

表 5.1　P_3 口的第二功能

I/O 口	第二功能	注　　释
$P_{3.0}$	RXD	串行口数据接收端
$P_{3.1}$	TXD	串行口数据发送端
$P_{3.2}$	$\overline{INT_0}$	外部中断请求 0
$P_{3.3}$	$\overline{INT_1}$	外部中断请求 1
$P_{3.4}$	T_0	定时/计数器 0
$P_{3.5}$	T_1	定时/计数器 1
$P_{3.6}$	$\overline{WR}$	外部 RAM 写信号
$P_{3.7}$	$\overline{RD}$	外部 RAM 读信号

2. 复位、电源、地

复位：意即从头开始，当 8051 通电，时钟电路开始工作，在 RESET 引脚上出现 24 个时钟周期以上的高电平，系统即初始复位。RESET 由高电平下降为低电平后，系统即从 0000H 地址开始执行程序。8051 的复位方式可以是上电复位，也可以是手动复位。RESET/Vpd 还是一个复用脚，V_{CC} 掉电期间，此脚可接上备用电源，以保证单片机内部 RAM 的数据不丢失。

电源：+5V；地：接地脚，如图 5.4 所示。

3. 时钟源

时钟源分为外部时钟和内部时钟。图 5.5 所示为单片机使用内部时钟时，只需外接一个晶体和两个 15～30pF 的电容即可；如使用外部时钟，则 $XTAL_1$ 接地，外部时钟由 $XTAL_2$ 输入。

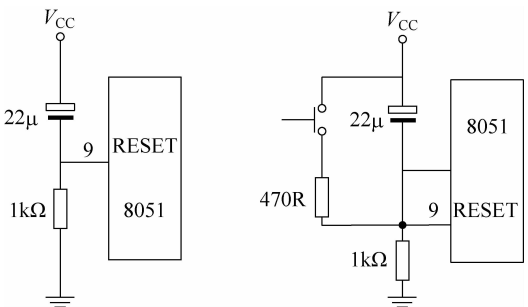


图 5.4 上电及复复位电路

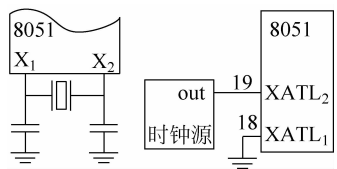


图 5.5 单片机时钟电路

5.2.2 MCS-51 单片机的典型应用

1. A/D 转换器技术

A/D 转换器是一种能把输入模拟电压或电流变成与成正比的数字量，即能把被控对象的各种模拟信息变成计算机可以识别的数字信息。A/D 转换器种类很多，但从原理上通常可以分为以下 4 种：计数器式 A/D 转换器、双积分式 A/D 转换器、逐次逼近式 A/D 转换器和并行 A/D 转换器。

计数器式 A/D 转换器结构很简单，但转换速度也很慢，所以很少采用。双积分式 A/D 转换器抗干扰能力强，转换精度也很高，但速度不够理想，常用于数字式测量仪表中。计算机中广泛采用逐次逼近式 A/D 转换器作为接口电路，它的结构不太复杂，转换速度也高。并行 A/D 转换器的转换速度最快，但因结构复杂而造价较高，故只用于那些转换速度极高的场合。

A/D 的主要参数如下。

1）转换精度

转换精度通常用分辨率和量化误差来描述。

分辨率：分辨率 $UREF/2^N$，它表示输出数字量变化一个相邻数码所需输入模拟电压的变化量，其中 N 为 A/D 转换的位数，N 越大，分辨率越高，习惯上分辨率常以 A/D 转换位数表示。

量化误差：量化误差是指零点和满度校准后，在整个转换范围内的最大误差。通常以相对误差形式出现，并以 LSB(Least Significant Bit，数字量最小有效位所表示的模拟量)为单位。

2）转换时间

指 A/D 转换器完成一次 A/D 转换所需时间。转换时间越短，适应输入信号快速变化能力越强。当 A/D 转换的模拟量变化较快时就需选择转换时间短的 A/D 转换器，否则会引起较大误差。

2. D/A 转换器技术

D/A 转换器的主要技术指标包括：转换精度、转换速度和温度系数等。

1) 转换精度

D/A 转换器的转换精度通常用分辨率和转换误差来描述。

分辨率用于表征 D/A 转换器对输入微小量变化的敏感程度。其定义为 D/A 转换器模拟量输出电压可能被分离的等级数。输入数字量位数愈多，输出电压可分离的等级愈多，即分辨率愈高。所以在实际应用中，往往用输入数字量的位数表示 D/A 转换器的分辨率。此外，D/A 转换器也可以用能分辨最小输出电压与最大输出电压之比给出。N 位 D/A 转换器的分辨率可表示为 $1/(2^N-1)$。它表示 D/A 转换器在理论上可以达到的精度。D/A 转换器的转换精度通常用分辨率和转换误差来描述。

由于 D/A 转换器中各元件参数存在误差，基准电压不够稳定和运算放大器的零漂等各种因素的影响，使得 D/A 转换器实际精度还与一些转换误差有关，如比例系数误差、失调误差和非线性误差等。

比例系数误差是指实际转换特性曲线的斜率与理想特性曲线斜率的偏差。

失调误差由运算放大器的零点漂移引起，其大小与输入数字量无关，该误差使输出电压的偏移特性曲线发生平移。

非线性误差是一种没有一定变化规律的误差，一般用在满刻度范围内，偏离理想的转移特性的最大值来表示。引起非线性误差的原因较多，如电路中的各模拟开关不仅存在不同的导通电压和导通电阻，而且每个开关处于不同位置（接地或接 V_{REF}）时，其开关压降和电阻也不一定相等。又如，在电阻网络中，每个支路上电阻误差不相同，不同位置上的电阻的误差对输出电压的影响也不相同等，这些都会导致非线性误差。

综上所述，为获得高精度的 D/A 转换精度，不仅应选择位数较多的高分辨率的 D/A 转换器，而且还需要选用高稳定的 V_{REF} 和低零漂的运算放大器才能达到要求。

2) 转换速度

当 D/A 转换器输入的数字量发生变化时，输出的模拟量并不能立即达到所对应的量值，它需要一段时间。通常用建立时间和转换速率两个参数来描述 D/A 转换器的转换速度。

建立时间（t_{set}）指输入数字量变化时，输出电压变化到相应稳定电压值所需要时间。一般用 D/A 转换器输入的数字量 N_B 从全 0 变为全 1 时，输出电压达到规定的误差范围（LSB/2）时所需时间表示。D/A 转换器的建立时间较快，单片集成 D/A 转换器建立时间最短可达 0.1μs 以内。

转换速率（SR）用大信号工作状态下（输入信号由全 1 到全 0 或由全 0 到全 1），模拟电压的变化率表示。一般集成 D/A 转换器在不包含外接参考电压源和运算放大器时，转换速率比较高。实际应用中，要实现快速 D/A 转换不仅要求 D/A 转换器有较高的转换速率，而且还应选用转换速率较高的集成运算放大器。

3) 温度系数

温度系数是指在输入不变的情况下，输出模拟电压随温度变化产生的变化量。一般用满刻度输出条件下温度每升高 1℃，输出电压变化的百分数作为温度系数。

3. A/D 转换技术的典型应用

基于 8051 单片机的模拟信号采集系统设计，单片机采用中断方式，A/D 转换器采用

ADC0809。对 ADC0809 的简单介绍如下。

ADC0809 采用逐次逼近工作原理，如图 5.6 所示。逐次逼近式 A/D 转换器与计数式 A/D 转换类似，只是数字量由“逐次逼近寄存器 SAR”产生。SAR 使用“对分搜索法”产生数字量，以 8 位数字量为例，SAR 首先产生 8 位数字量的一半，即 10000000B，试探模拟量 V_i 的大小，若 $V_o > V_i$，清除最高位，若 $V_o < V_i$，保留最高位。在最高位确定后，SAR 又以“对分搜索法”确定次高位，即以低 7 位的一半 y1000000B（y 为已确定位）试探模拟量 V_i 的大小。在 bit6 确定后，SAR 以对分搜索法确定 bit5 位，即以低 6 位的一半 yy100000B（y 为已确定位）试探模拟量的大小。重复这一过程，直到最低位 bit0 被确定，转换结束。

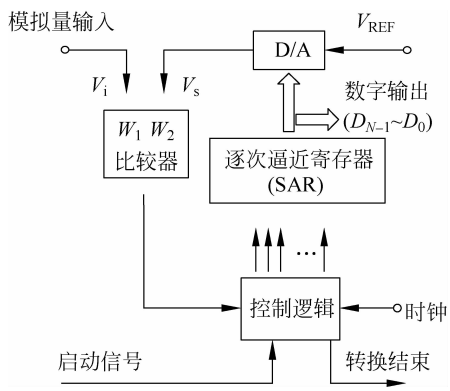


图 5.6 逐次逼近型 A/D 转换原理

转换过程如下。

(1) 首先发出“启动信号”信号 S。当 S 由高变低时，“逐次逼近寄存器 SAR”清零，DAC 输出 $V_o = 0$，“比较器”输出 1。当 S 变为高电平时，“控制电路”使 SAR 开始工作。

(2) SAR 首先产生 8 位数字量的一半，即 10000000B，试探模拟量的 V_i 大小，若 $V_o > V_i$，“控制电路”清除最高位，若 $V_o < V_i$，保留最高位。

(3) 在最高位确定后，SAR 又以对分搜索法确定次高位，即以低 7 位的一半 y1000000B（y 为已确定位）试探模拟量 V_i 的大小。在 bit6 确定后，SAR 以对分搜索法确定 bit5 位，即以低 6 位的一半 yy100000B（y 为已确定位）试探模拟量 V_i 的大小。重复这一过程，直到最低位 bit0 被确定。

(4) 在最低位 bit0 确定后，转换结束，“控制电路”发出“转换结束”信号 EOC。该信号的下降沿把 SAR 的输出锁存在“缓冲寄存器”里，从而得到数字量输出。从转换过程可以看出：启动信号为负脉冲有效。转换结束信号为低电平。

ADC0809 芯片简介如下。

(1) 逐次逼近型 8 位 A/D 转换器；

(2) 片内有 8 路模拟开关，可对 8 路模拟电压量实现分时转换；

(3) 典型工作频率 500～640kHz，转换速度 100～128μs；

(4) 片内带有三态输出缓冲器，可直接与单片机的数据总线相连接。

引脚功能简介如下。

(1) CLK：时钟信号。

典型值为 500～640kHz。

（2）V_{REF+}、V_{REF-}：基准电压输入。

通常 V_{REF+} 接＋5V、V_{REF-} 接地。

（3）ALE：地址锁存允许，其上升沿锁存 ADDC～ADDA 的地址信号。

（4）START：A/D 转换启动信号，上升沿启动 A/D 转换。

（5）EOC：转换完成信号，转换完成后输出高电平。该信号可用作向单片机提出中断申请或者作为查询信号。

（6）OE：数字量输出允许信号，高电平为输出允许，转换后的数字量从 D_0～D_7 脚输出。

（7）IN_0～IN_7：模拟电压输入，8 个引脚可分别接 8 路模拟信号。

（8）D_0～D_7：数字量输出端。

（9）ADDA、ADDB、ADDC：通道选择信号。

ADDC	ADDB	ADDA	模拟通道
0	0	0	IN_0
0	0	1	IN_1
		…	…
1	1	1	IN_7

转换结束信号 EOC 根据不同的方式和单片机的连接形式不同。

（1）若采用延时方式：EOC 悬空，在启动转换后延时 100μs，再读转换结果。

（2）若采用查询方式时，可将 EOC 接并行口（P_1 或 P_3）的某线，检测 EOC 变高后，再读入转换结果。

（3）若采用中断方式可将 EOC 经非门反相接到单片机的中断请求端，一旦转换完成，EOC 变为高电平，向 8XX51 提出中断请求，进入中断服务后读入转换结果。

A/D 转换器的应用实例如图 5.7 所示。

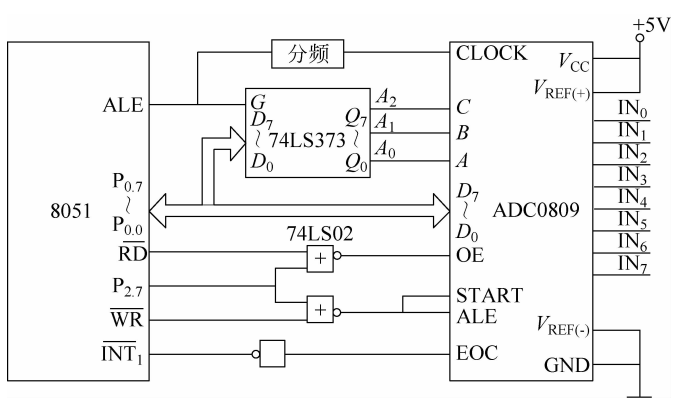


图 5.7　数据采集电路图

电路工作原理说明如下。

（1）模拟信号输入 IN_0～IN_7，对应的通道地址：7FF8H～7FFFH；

（2）锁存通道地址和启动转换的指令：

```
MOV  DPTR,#7FF8H
```

```
MOVX  @DPTR,A
```

(3) 读取转换结果的指令：

```
MOV  DPTR,#7FF8H
MOVX  A,@DPTR
```

(4) 主程序：

```
    ORG    0000H
        LJMP   MAIN
        ORG   0013H            ; INT1 中断入口地址
        LJMP   INT1
        ORG   0030H
    MAIM: MOV  R0,#60H         ; 置数据存储区首址
    MOV   R2,#08H              ; 置 8 路数据采集初值
    SETB   IT1                 ; 设置边延触发中断
    SETB   EA
    SETB  EX1                  ; 开放外部中断 1
    MOV   DPTR,#7FF8H          ; 指向 0809 通道 0
RD: MOVX  @DPTR,A              ; 启动 A/D 转换
HE: MOV   A,R2                 ; 8 路巡回检测数送 A
    JNZ   HE                   ; 等待中断,8 路未完继续
    ⋮
.....
INT1: MOVX  A,@DPTR            ; 读取 A/D 转换结果
    MOV  @R0,A                 ; 向指定单元存数
    INC   DPTR                 ; 输入通道数加 1
    INC   R0                   ; 存储单元地址加 1
    MOVX   @DPTR,A             ; 启动新通道 A/D 转换
    DEC   R2                   ; 待检通道数减 1
    RETI                       ; 中断返回
```

4. 电表箱防窃电控制器的设计实例

随着供电网络的不断延伸，电力客户数量不断增加，传统的电能计量箱管理方式已不能适应企业发展的需要，严重影响了企业经济效益的提高和服务工作的开展，主要表现在以下几个方面。

(1) 电能计量箱的资料难以及时更新，不能反映电能计量箱现场的实际情况，给电力客户服务、日常维护等工作带来了很大的障碍。

(2) 电能计量箱的锁具目前采用的机械或磁性钥匙防复制能力差，容易破解。当电能计量装置被非法操作时，容易与电力客户发生法律纠纷。供电企业不得不频繁更换锁具，浪费了大量的人力和物力。

(3) 供电企业对电能计量箱的开启、关闭情况没有一种有效的监控手段，无法掌握和控制电能计量箱开、关的人员、时间、次数等信息。

(4) 由于电能计量箱数量极大，导致钥匙数量、规格相应较多，发放钥匙的过程占用了较多的时间，在钥匙管理上经常发生差错。

(5) 现有计量箱的挂锁实行分台区分锁号使用，须向厂家定制，成本较高。同时如果钥匙有丢失等情况时，相同编号的几十套锁具需要全部更换。

(6) 传统计量箱的挂锁由于无防雨装置,数月后很快锈蚀,致使供电企业维护人员不得不采取主动破坏行为。

按照国家电网公司智能电网建设“十二五”规划,电能计量在装备现代化上将不断上升,自动化水平将越来越高,自动采集范围将从电能采集扩大到防盗,防窃电,用电监测,节能调控。

目前国内对电表箱防窃电控制器的研究多集中在对箱门的开合检测,用 TTU 或负控装置将检测结果通过 GPRS 或载波方式传递到主站。这种方式解决了箱门开闭的检测,但无法了解是何人对箱门进行开闭,同时在停电状态下,也无法检测到箱门开闭状态。新型防窃电控制器可提供智能化检测和控制、非法开箱智能报警、485 抄表、电能管理、Internet 远程控制等多种功能。新型防窃电控制器是充满智慧的控制系统,可提供实时的监测数据,提高用电的安全性,节约各种能源费用。

方案总体设计框图如图 5.8 所示,系统中主要包括控制器、GPRS 网络及主站管理系统。

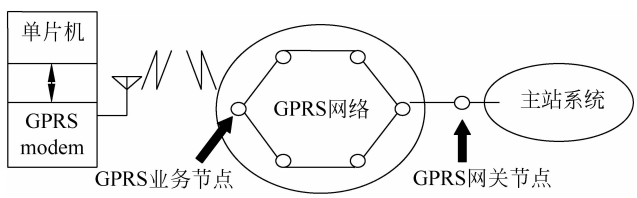


图 5.8　总体设计图

电表箱防窃电控制器具备以下功能。

(1) 具有自动检测和控制功能。

(2) 计量箱门被非法打开后,智能控制装置会在 60 秒内通过控制开关自动切断电源,同时发出蜂鸣声警报。

(3) 计量箱断电后,用户不能自行恢复用电,必须由管理人员持有电子密钥才可以恢复用电。

(4) 智能控制装置应在进线电源断电时仍能发挥作用,对于断电后非法打开表箱具有记忆功能,并在进线电源来电时,智能装置同样具有切断主电源的功能。

(5) 在控制计量箱箱门没有按要求完全关闭时,应在一分钟内采取断电提醒,送电后待表箱门完全关闭方能正常运行。

(6) 智能控制装置需自动记录最近 40 次表箱开启时间记录及 40 次电密钥匙操作记录,包括钥匙编号及操作时间等信息,可通过读数钥匙将信息采集至系统管理中心进行读取查询。

控制器电路图如图 5.9 所示,电路中主要有控制器 AT89C51、上拉电阻排、复位、时钟及报警电路。

防窃电开关量检测电路如图 5.10 所示,设计了两路开关量输入,如果发生窃电行为,系统就会检测到非法开箱信息,立即做出报警、断电及通知主管部门,让管理人员及时得到窃电行为的信息,及时采取必要的措施。

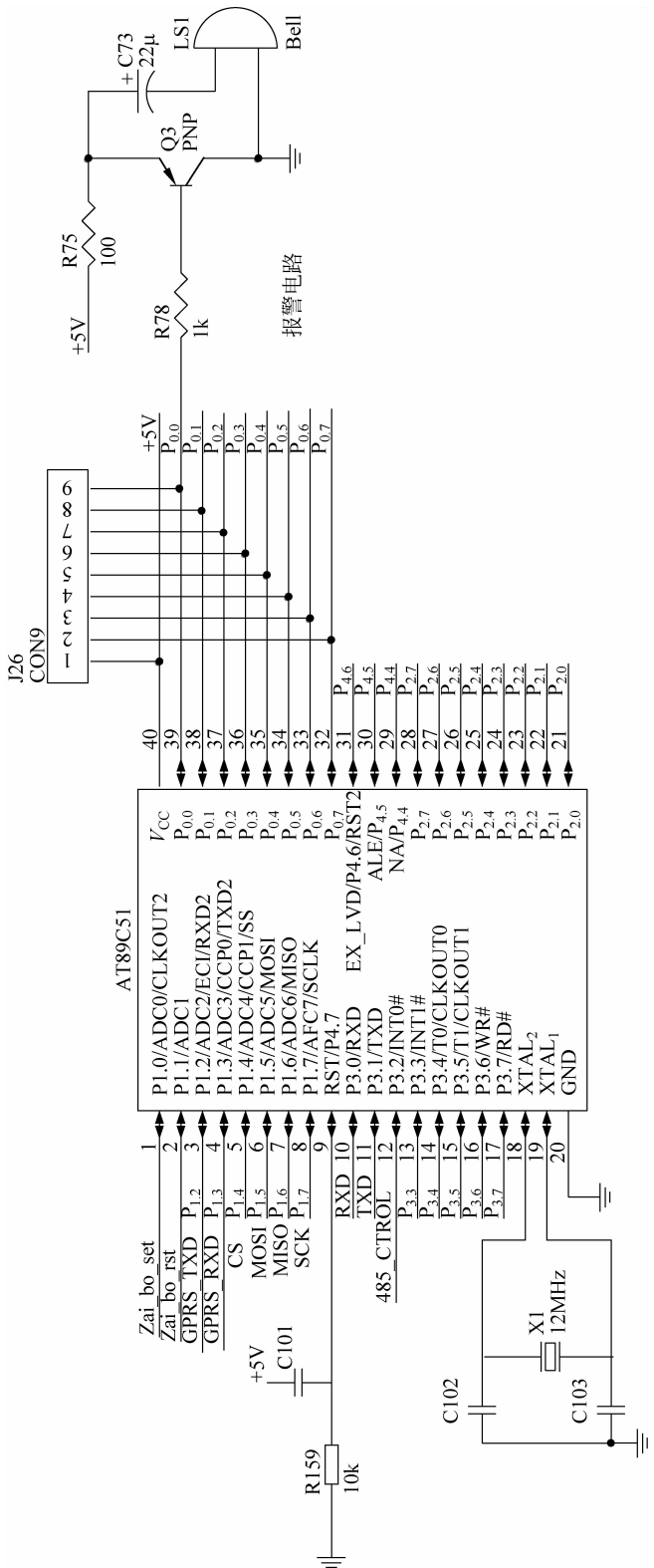

J26
CON9
AT89C51
报警电路
Bell
LS1
Q3
PNP
R75
100
R78
1k
+5V
+C73
22μ
X1
12MHz
C101
C102
C103
R159
10k
Zai_bo_set
Zai_bo_rst
GPRS_TXD
GPRS_RXD
CS
MOSI
MISO
SCK
485_CTROL
RXD
TXD
EX_LVD/P4.6/RST2
ALE/P4.5
NA/P4.4
GND

图 5.9 控制器电路

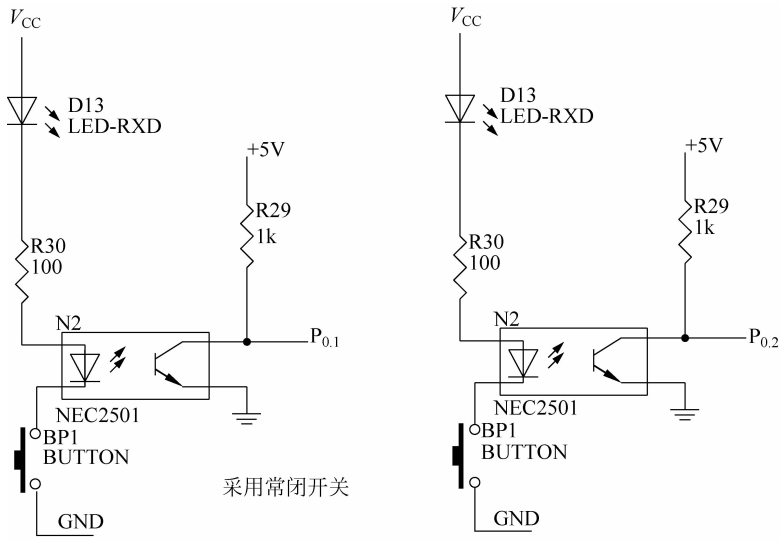


图 5.10　防窃电开关量检测电路

GPRS 电路及存储电路如图 5.11 所示，该电路可以通过移动或联通网络及时上传各种信息；存储电路可以存储 40 组记录数据，以供查询。

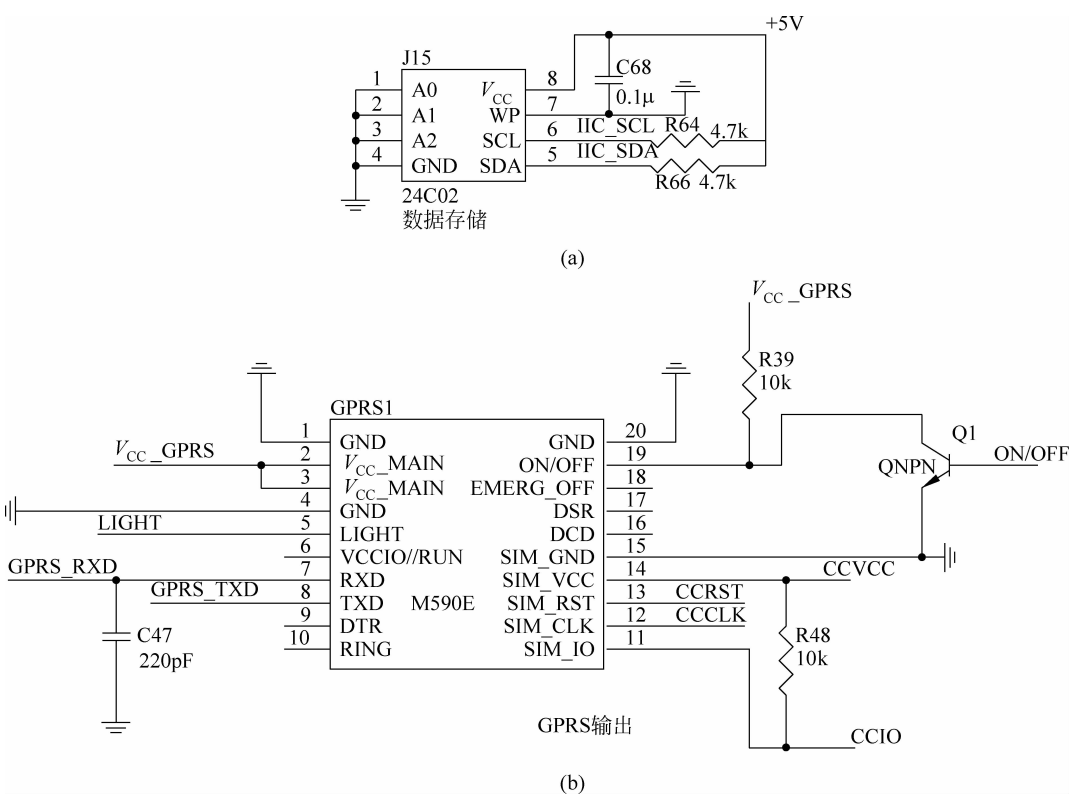


图 5.11　GPRS 及存储电路

系统调试及通信电路如图 5.12 所示，该电路能完成系统程序的更新下载及各种调试功能。

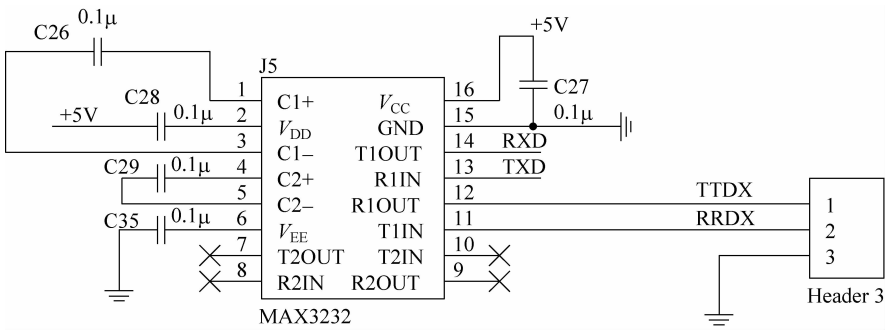


图 5.12 通信及调试电路

系统软件设计采用模块化设计，简化了设计流程，增强了程序的易读性。

系统流程如下：当计量箱门被打开时，则计量箱上的行程开关得到输入信号，控制继电器给防窃电控制器供电，控制器开始进入运行状态。此时，LED 指示灯点亮，同时锁定继电器，等待电子钥匙。若有电子钥匙插入则判断密码，若密码正确，则控制器无动作，LED 指示灯保持慢速闪烁。若密码不正确，LED 指示灯快速闪烁，蜂鸣器鸣叫警告，延时一定时间后控制器控制断路器动作断电。当关闭计量箱门时，判断控制器是否处于合法开门状态，若是，则控制继电器断电，关闭控制器，流程结束。程序流程如图 5.13 所示。

下面给出部分密钥的源程序，对 EEPROM 的读写程序：

```
/*
* 文 件 名: IAP.c
* 芯     片: STC12C5A60S2
* 晶     振: 11.0592MHz
* 功能描述: STC 内部 1KB(0x0000～0x3FF)EEPROM 读写
*           第一扇区 0x0000～0x01FF
*           第二扇区 0x0200～0x03FF
*/
#include "STC12C5A.H"
#include "IAP.h"
U8 xdata EEPROM_sector[512];
/* ------------------------------------------------------------------------
* 函 数 名: Open_IAP
* 功能描述: 允许 ISP/IAP
* 输入参数: 无
* 返 回 值: 无
------------------------------------------------------------------------
*/
void Open_IAP(void)
{
    //IAPEN   SWBS    SWRST   CMD_FAIL -       WT2     WT1     WT0
    /* IAPEN: ISP/IAP 功能允许位.0: 禁止 ISP/IAP 编程改变 Flash,1:允许编程改变 Flash
SWBS: 软件选择从用户主程序区启动(0),还是从 ISP 程序区启动(1)
SWRST: 0: 不操作; 1: 产生软件系统复位,硬件自动清零
```

```
CMD_FAIL: 如果送了 ISP/IAP 命令,并对 ISP_TRIG 送 5Ah/A5h 触发失败,则为 1,需由软件清零 * /
     //IAP_CONTR = 0X80;  //允许 ISP/IAP,系统时钟< 30MHz 时
     //IAP_CONTR = 0X81;  //允许 ISP/IAP,系统时钟< 24MHz 时
     //IAP_CONTR = 0x82;  //允许 ISP/IAP,系统时钟< 20MHz 时,设置等待时间 WT2,WT1,WT0(010)
     //IAP_CONTR = 0X83;  //允许 ISP/IAP,系统时钟< 12MHz 时
     //IAP_CONTR = 0X84;  //允许 ISP/IAP,系统时钟< 6MHz 时
     //IAP_CONTR = 0X85;  //允许 ISP/IAP,系统时钟< 3MHz 时
     //IAP_CONTR = 0X86;  //允许 ISP/IAP,系统时钟< 2MHz 时
     //IAP_CONTR = 0X87;  //允许 ISP/IAP,系统时钟< 1MHz 时
}
/ * ----------------------------------------------------------------------
 * 函 数 名: Close_IAP
 * 功能描述: 禁止 ISP/IAP
 * 输入参数: 无
 * 返 回 值: 无
----------------------------------------------------------------------
 * /
void Close_IAP(void)
{
    IAP_CONTR = 0x00;     //禁止 ISP/IAP
    IAP_CONTR = 0;        //关闭 IAP 功能
    IAP_CMD   = 0;        //清命令寄存器,使命令寄存器无命令,此句可不用
    IAP_TRIG  = 0;        //清命令触发寄存器,使命令触发寄存器无触发,此句可不用
    IAP_ADDRH = 0x80;
    IAP_ADDRL = 0;
    / * IAP_ADDRH: ISP/IAP 操作时的地址寄存器高 8 位
    IAP_ADDRL: ISP/IAP 操作时的地址寄存器低 8 位 * /
}
/ * ----------------------------------------------------------------------
 * 函 数 名: Read_IAP_Byte
 * 功能描述: 从 EEPROM 指定的单元读取一个字节数据
 * 输入参数: addr:16b 地址
 * 返 回 值: IAP_DATA:从指定的地址读取到的数据
----------------------------------------------------------------------
 * /
void REEPROMU8(U8 * pRData,U16 nAddr)
{    IAP_ADDRH = (nAddr & 0xFF00)>> 8;
     IAP_ADDRL = nAddr & 0x00FF;
     IAP_CMD = 0x01;
     IAP_TRIG = 0x5A;
     IAP_TRIG = 0xA5;     //对 IAP_TRIG 先写 0x5A 再写 0xA5,ISP/IAP 命令才会生效
     * pRData = IAP_DATA;
}
/ * ----------------------------------------------------------------------
 * 函 数 名: Write_IAP_Byte
 * 功能描述: 把一个字节数据写入 EEPROM 指定的单元,写入数据前应先擦除扇区
 * 输入参数: addr:16b 地址; writeVal:要写入的数据
 * 返 回 值: 无
----------------------------------------------------------------------
 * /
void WEEPROMU8(U8 nWData,U16 nAddr)
```

```
{
    IAP_CMD = 0x02;
    IAP_ADDRH = (nAddr & 0xFF00)>> 8;
    IAP_ADDRL = nAddr & 0x00FF;
    IAP_DATA = nWData;
    IAP_TRIG = 0x5A;
    IAP_TRIG = 0xA5;      //对 IAP_TRIG 先写 0x5A 再写 0xA5,ISP/IAP 命令才会生效
}
/* ------------------------------------------------------------------
* 函 数 名: Erase_IAP_Sector
* 功能描述: 擦除扇区,没有字节擦除
* 输入参数: addr: 扇区地址,扇区中任意一个字节地址都是该扇区地址
* 返 回 值: 无
------------------------------------------------------------------
*/
void Erase_IAP_Sector(U16 addr)
{
    IAP_CMD = 0x03;
    IAP_ADDRH = (addr & 0xFF00)>> 8;
    IAP_ADDRL = addr & 0x00FF;
    IAP_TRIG = 0x5A;
    IAP_TRIG = 0xA5;      //对 IAP_TRIG 先写 0x5A 再写 0xA5,ISP/IAP 命令才会生效
}
/* ------------------------------------------------------------------
* 函 数 名: Erase_IAP_Byte
* 功能描述: 擦除指定单元内容
* 输入参数: addr: 单元地址
* 返 回 值: 无
* ------------------------------------------------------------------
*/
void Erase_IAP_Byte(U16 addr)
{
    //STC 内部扩展 1KB(0x0000～0x03FF)ROM
    //临时存放 EEPROM 一个扇区 512B 数据
    U16 sectorAddr;       //扇区首地址
    U16 unit;             //某个扇区的某个单元 0～511
    U16 i;
    sectorAddr = ((int)(addr / 512)) * 512;
    unit = addr % 512;
    for (i = 0; i < 512; i++)
    {
    REPROMU8(&EEPROM_sector[i],(sectorAddr + i));              //读取扇区
    }
    Erase_IAP_Sector(sectorAddr);                              //擦除扇区
    EEPROM_sector[unit] = 0xFF;                                //要擦除的单元置为 0xFF
    for (i = 0; i < 512; i++)
    {
    WEEPROMU8(EEPROM_sector[i],sectorAddr + i );               //重新写入扇区
    }
}
/* ------------------------------------------------------------------
```

```
* 函 数 名: Erase_IAP_nByte
* 功能描述: 擦除多个指定单元内容
* 输入参数: addr: 单元地址
* 返 回 值: 无
* ----------------------------------------------------------------------
*/
void Erase_IAP_nByte(U16 n,U16 addr)
{
     //STC 内部扩展 1KB(0x0000～0x03FF)ROM
     //临时存放 EEPROM 一个扇区 512B 数据
     U16 sectorAddr;                                          //扇区首地址
     U16 unit;                                                //某个扇区的某个单元 0～511
     U16 i;
     sectorAddr = ((int)(addr / 512)) * 512;
     unit = addr % 512;
     for (i = 0; i < 512; i++)
     {
     REEPROMU8(&EEPROM_sector[i],(sectorAddr + i));           //读取扇区
     }
     Erase_IAP_Sector(sectorAddr);                            //擦除扇区
     for(i = unit;i < n + unit;i++)
     {
    EEPROM_sector[i] = 0xFF;                                  //要擦除的单元置为 0xFF
     }
    for (i = 0; i < 512; i++)
     {
     WEEPROMU8(EEPROM_sector[i],sectorAddr + i );             //重新写入扇区
     }
}
void EEPROMToMem(char * pData,U16 nRLen,U16 nStartAddr)       //从 EEPROM 读数据到内存
{  // pData: 数组指针; nRLen: 数据长度; nStartAddr: 开始地址
    U16 i;
    for(i = 0;i < nRLen;i++)
        {
             REEPROMU8((U8 * )pData,nStartAddr);
             pData++;nStartAddr++;
        }
    }
void MemToEEPROM(const char * pData,U16 nRLen,  U16  nStartAddr)     //从内存到 EEPROM
    { // pData: 数组指针; nRLen: 数据长度; nStartAddr: 开始地址
    U16 i;
    Erase_IAP_nByte(nRLen,nStartAddr);
    for(i = 0;i < nRLen;i++)
        {
             WEEPROMU8( * pData,nStartAddr);
             pData++;nStartAddr++;
        }
    }
```

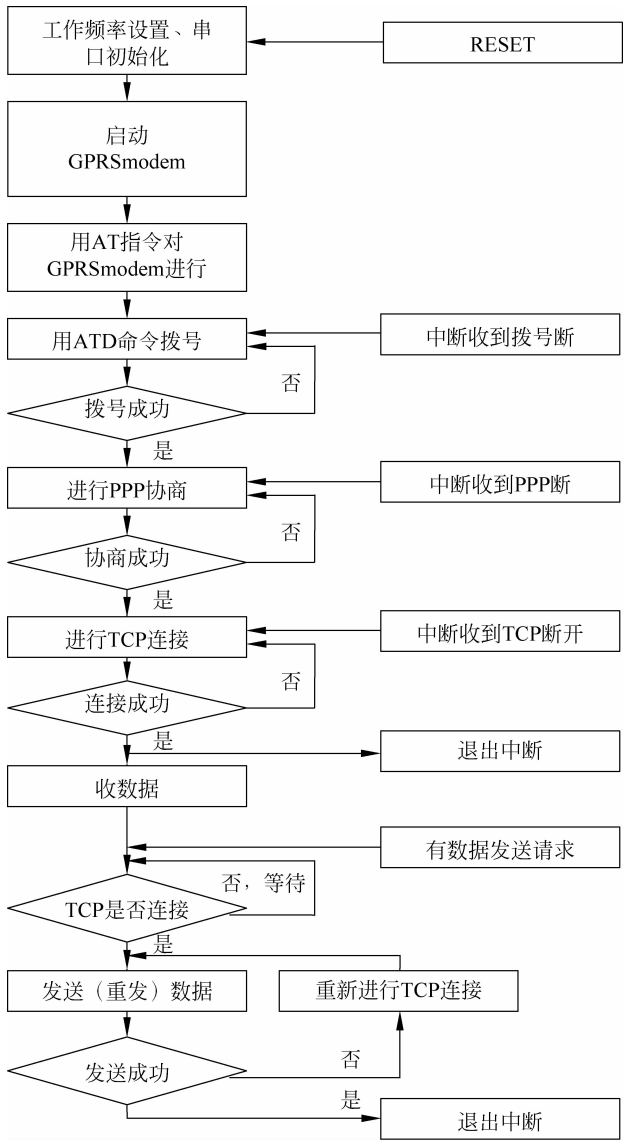


图 5.13 程序流程图

下面是串行通信的子程序，完成数据通信功能。

```
/*
* 模 块 名: UART.c
* 芯      片: STC12C5A60S2
* 晶      振: 11.0592MHz
* 功能描述: 串口驱动
*/
//---------------- Include files ---------------------------//
#include"stc12c5a.h"
#include"uart.h"
#include "config.h"
```

```
//#include "DELAY.H"
/*
* ---- FUNCTION ----------------------------------------------------------
* Name     : Uart1_Init
* Description : 串口 1 初始化程序
-------------------------------------------------------------------------
*/
#define MYARRAYSIZE(_Array) (sizeof(_Array)/sizeof((_Array)[0]))
#define FOSC 11059200
#define BAUD 9600
#define _DEBUG_PROTEUS
void Uart1_Init(void)                          //9600bps@11.0592MHz
{
#ifdef _DEBUG_PROTEUS
    TMOD = 0x20;                               //设定定时器 1 为 8 位自动重装方式
    TH1 = TL1 = -(FOSC/12/32/BAUD);            //设定定时初值
    SCON = 0x50;                               //8 位数据,可变波特率
#endif
    EA = 1;
    TR1 = 1;                                   //启动定时器 1
    ES = 1;                                    //允许串口 1 中断
}
void timer0_Init()
{
    TMOD |= 0x01;                              //设定定时器 0 为 16 位定时方式
    TL0 = T0_10MS;
    TH0 = T0_10MS/256;                         //设定定时初值
    //T0x12 = 0;                               //设定定时器 0 为 T12 分频定时方式
    ET0 = 1;                                   //定时器 0 中断
    TR0 = 0;                                   //关闭定时器 0
    timer0_restart_10ms();
}
void timer0_restart_10ms(){TL0 = T0_10MS; TH0 = T0_10MS/256;TR0 = 1;}
static U8 xdata m_nU485RxBuff[32];
static U8 * m_pU485RxBuff = m_nU485RxBuff;
/* ----------------------------------
UART interrupt service routine
---------------------------------- */
//判断串口接收区没有溢出
#define IsU485RxFinish() (m_pU485RxBuff == &m_nU485RxBuff[sizeof(m_nU485RxBuff)])
void Uart1_Isr()interrupt 4 using 1
{
    U8 rxtmp;
    RI = 0;
    if(IsU485RxFinish())
    {
        rxtmp = SBUF;
    }
    else
    {
         *m_pU485RxBuff = SBUF;
```

```
        timer0_restart_10ms();
        m_pU485RxBuff++;
    }
  P2 | = 0x40;;Is_Power_Down = 1;PCON = 0X02;};
    //timer0_restart_10ms();
}
void timer0_interrupt(void) interrupt 1      //超时
{
    TF0 = 0;
    TR0 = 0;                                 //停止工作
    LED1 = !LED1;
}
//输入数组长度必须为 32 字节
U8 U485RxFrame(char * pOutBuff)
{
    char i, * pRxBuff;
    //如果定时器 0 在工作说明串口正在接收数据
    if(TR0) return 0;
    //如果接收缓冲区空
    if(m_pU485RxBuff == m_nU485RxBuff) return 0;
    //判断是否接收完成
    if(IsU485RxFinish())
    {
        ES = 0;
        pRxBuff = m_nU485RxBuff;
        for(i = sizeof(m_nU485RxBuff);i;i -- )
        {
            * pOutBuff ++ = * pRxBuff ++;
        }
        m_pU485RxBuff = m_nU485RxBuff;
        ES = 1;
        return sizeof(m_nU485RxBuff);
    }
    m_pU485RxBuff = m_nU485RxBuff;
    return 0;
}
//输入数组长度必须为 32 字节
void U485TxFrame(const U8 * pFrame)
{
    signed char i;
    ES = 0;
    for(i = 32;i;i -- )
    {
        SBUF = * pFrame ++;
        while(!TI);                          // 等待数据传送
        TI = 0;
    }
    ES = 1;
}
void Uart1TxFrame(U8 * pTxBuff,I8 nBuffX)
{
```

```
    signed char i;
    ES = 0;
    for(i = 0;i < nBuffX;i++)
    {
        SBUF = * pTxBuff ++;
        while(!TI);                          // 等待数据传送
        TI = 0;
    }
    ES = 1;
}
/*
* FUNCTION
-------------------------------------------------------------------------------
* Name          : UART1_Send_Byte
* Description   : 串口 1 发送单个字节数据
-------------------------------------------------------------------------------
*/
void UART1_Send_Byte(uchar ddata)
{
    ES = 0;
    SBUF = ddata;
    while(!TI);                              // 等待数据传送
    TI = 0;                                  // 清除数据传送标志
    ES = 1;
}
/*
* FUNCTION
-------------------------------------------------------------------------------
* Name          : UART1_Send_String
* Description   : 串口 1 发送以'\\0'结尾的字符串
-------------------------------------------------------------------------------
*/
void UART1_Send_String(char * pInstr)
{
    while( * pInstr)
    {
        SBUF = * pInstr ++;
        while(!TI);                          // 等待数据传送
        TI = 0;
    }
}
```

5.3 嵌入式系统简介

嵌入式系统本身是一个外延极广的名词，凡是与产品结合在一起的具有嵌入式特点的控制系统都可以叫嵌入式系统，而且有时很难给它下一个准确的定义。现在人们讲嵌入式系统时，某种程度上指近些年比较热的具有操作系统的嵌入式系统，本文在进行分析和展望时，也沿用这一观点。

一般而言，嵌入式系统的构架可以分成 4 个部分：处理器、存储器、输入输出(I/O)和软件(由于多数嵌入式设备的应用软件和操作系统都是紧密结合的，在这里我们对其不加区

分，这也是嵌入式系统和 Windows 系统的最大区别）。

嵌入式系统一般由以下几部分组成：（1）嵌入式微处理器；（2）外围硬件设备；（3）嵌入式操作系统；（4）特定的应用程序。

可从几方面来理解嵌入式系统的含义：

（1）嵌入式系统是面向用户、面向产品、面向应用的，它必须与具体应用相结合才会具有生命力、才更具有优势。因此可以这样理解上述三个面向的含义，即嵌入式系统是与应用紧密结合的，它具有很强的专用性，必须结合实际系统需求进行合理的裁剪应用。

（2）嵌入式系统是将先进的计算机技术、半导体技术、电子技术和各个行业的具体应用相结合后的产物，这一点就决定了它必然是一个技术密集、资金密集、高度分散、不断创新的知识集成系统。所以，介入嵌入式系统行业，必须有一个正确的定位。例如 Palm 之所以在 PDA 领域占有 70％以上的市场，就是因为其立足于个人电子消费品，着重发展图形界面和多任务管理；而风河的 Vxworks 之所以在火星车上得以应用，则是因为其高实时性和高可靠性。

（3）嵌入式系统必须根据应用需求对软硬件进行裁剪，满足应用系统的功能、可靠性、成本、体积等要求。所以，如果能建立相对通用的软硬件基础，然后在其上开发出适应各种需要的系统，是一个比较好的发展模式。目前的嵌入式系统的核心往往是一个只有几 KB 到几十 KB 微内核，需要根据实际的使用进行功能扩展或者裁剪，但是由于微内核的存在，使得这种扩展能够非常顺利地进行。

5.3.1 嵌入式系统的发展

随着信息化、智能化、网络化的发展，嵌入式系统技术也将获得广阔的发展空间。美国著名未来学家尼葛洛庞帝 1999 年 1 月访华时预言，4～5 年后嵌入式智能（计算机）工具将是 PC 和因特网之后最伟大的发明。

进入 20 世纪 90 年代，嵌入式技术全面展开，目前已成为通信和消费类产品的共同发展方向。在通信领域，数字技术正在全面取代模拟技术。在广播电视领域，数字音频广播（DAB）也已进入商品化试播阶段。而软件、集成电路和新型元器件在产业发展中的作用日益重要。所有上述产品中，都离不开嵌入式系统技术。前途不可估量的维纳斯计划生产机顶盒，核心技术就是采用 32 位以上芯片级的嵌入式技术。在个人领域中，嵌入式产品将主要是个人商用，作为个人移动的数据处理和通信软件。由于嵌入式设备具有自然的人机交互界面，GUI 屏幕为中心的多媒体界面给人很大的亲和力。手写文字输入、语音拨号上网、收发电子邮件以及彩色图形、图像已取得初步成效。

目前一些先进的 PDA 在显示屏幕上已实现汉字写入、短消息语音发布，应用范围也将日益广阔。对于企业专用解决方案，如物流管理、条码扫描、移动信息采集等，这种小型手持嵌入式系统将发挥巨大的作用。在自动控制领域，不仅可以用于 ATM 机，自动售货机，工业控制等专用设备，和移动通信设备结合、GPS、娱乐相结合，嵌入式系统同样可以发挥巨大的作用。近期长虹推出的 ADSL 产品，结合网络、控制、信息，这种智能化、网络化将是家电发展的新趋势。

在硬件方面，不仅有各大公司的微处理器芯片，还有用于学习和研发的各种配套开发包。目前低层系统和硬件平台经过若干年的研究，已经相对比较成熟，实现各种功能的芯片

应有尽有。而且巨大的市场需求给我们提供了学习研发的资金和技术力量。

从软件方面讲，也有相当部分的成熟软件系统。国外商品化的嵌入式实时操作系统，已进入我国市场的有 WindRiver、Microsoft、QNX 和 Nuclear 等产品。我国自主开发的嵌入式系统软件产品如科银(CoreTek)公司的嵌入式软件开发平台 DeltaSystem，中科院推出的 Hopen 嵌入式操作系统。同时由于是研究热点，所以我们可以在网上找到各种各样的免费资源，从各大厂商的开发文档，到各种驱动，程序源代码，甚至很多厂商还提供微处理器的样片。这对于我们从事这方面的研发，无疑是个资源宝库。对于软件设计来说，不管是刚入门还是进一步开发，都相对来说比较容易。这就使得很多新手能够比较快的进入研发状态，利于发挥大家的积极性和创造性。

5.3.2　嵌入式系统的分类与应用

根据其现状，嵌入式处理器可以分成下面几类。

1) 嵌入式微处理器(Micro Processor Unit，MPU)

嵌入式微处理器是由通用计算机中的 CPU 演变而来的。它的特征是具有 32 位以上的处理器，具有较高的性能，当然其价格也相应较高。但与计算机处理器不同的是，在实际嵌入式应用中，只保留和嵌入式应用紧密相关的功能硬件，去除其他的冗余功能部分，这样就以最低的功耗和资源实现嵌入式应用的特殊要求。和工业控制计算机相比，嵌入式微处理器具有体积小、重量轻、成本低、可靠性高的优点。目前主要的嵌入式处理器类型有 Am186/88、386EX、SC-400、Power PC、68000、MIPS、ARM/ StrongARM 系列等。

其中，ARM/StrongARM 是专为手持设备开发的嵌入式微处理器，属于中档的价位。

2) 嵌入式微控制器(Microcontroller Unit，MCU)

嵌入式微控制器的典型代表是单片机，从 20 世纪 70 年代末单片机出现到今天，虽然已经经过了 30 多年的历史，但这种 8 位的电子器件目前在嵌入式设备中仍然有着极其广泛的应用。单片机芯片内部集成 ROM/EPROM、RAM、总线、总线逻辑、定时/计数器、看门狗、I/O、串行口、脉宽调制输出、A/D、D/A、Flash RAM、EEPROM 等各种必要功能和外设。和嵌入式微处理器相比，微控制器的最大特点是单片化，体积大大减小，从而使功耗和成本下降、可靠性提高。微控制器是目前嵌入式系统工业的主流。微控制器的片上外设资源一般比较丰富，适合于控制，因此称微控制器。

由于 MCU 低廉的价格，优良的功能，所以拥有的品种和数量最多，比较有代表性的包括 8051、MCS-251、MCS-96/196/296、P51XA、C166/167、68K 系列以及 MCU 8XC930/931、C540、C541，并且有支持 I2C、CAN-Bus、LCD 及众多专用 MCU 和兼容系列。目前 MCU 占嵌入式系统约 70%的市场份额。近来 Atmel 出产的 AVR 单片机由于其集成了 FPGA 等器件，所以具有很高的性价比，势必将推动单片机获得更高的发展。

3) 嵌入式 DSP 处理器(Embedded Digital Signal Processor，EDSP)

DSP 处理器是专门用于信号处理方面的处理器，其在系统结构和指令算法方面进行了特殊设计，具有很高的编译效率和指令的执行速度。在数字滤波、FFT、谱分析等各种仪器上 DSP 获得了大规模的应用。

DSP 的理论算法在 20 世纪 70 年代就已经出现，但是由于专门的 DSP 处理器还未出现，所以这种理论算法只能通过 MPU 等由分立元件实现。MPU 较低的处理速度无法满足

DSP 的算法要求，其应用领域仅仅局限于一些尖端的高科技领域。随着大规模集成电路技术发展，1982 年世界上诞生了首枚 DSP 芯片。其运算速度比 MPU 快了几十倍，在语音合成和编码解码器中得到了广泛应用。至 20 世纪 80 年代中期，随着 CMOS 技术的进步与发展，第二代基于 CMOS 工艺的 DSP 芯片应运而生，其存储容量和运算速度都得到成倍提高，成为语音处理、图像硬件处理技术的基础。到 20 世纪 80 年代后期，DSP 的运算速度进一步提高，应用领域也从上述范围扩大到了通信和计算机方面。20 世纪 90 年代后，DSP 发展到了第五代产品，集成度更高，使用范围也更加广阔。

目前最为广泛应用的是 TI 的 TMS320C2000/C5000 系列，另外如 Intel 的 MCS-296 和 Siemens 的 TriCore 也有各自的应用范围。

4) 嵌入式片上系统(System on Chip)

SoC 追求产品系统最大包容的集成器件，是目前嵌入式应用领域的热门话题之一。SoC 最大的特点是成功实现了软硬件无缝结合，直接在处理器片内嵌入操作系统的代码模块。而且 SoC 具有极高的综合性，在一个硅片内部运用 VHDL 等硬件描述语言，实现一个复杂的系统。用户不需要再像传统的系统设计一样，绘制庞大复杂的电路板，一点点的连接焊制，只需要使用精确的语言，综合时序设计直接在器件库中调用各种通用处理器的标准，然后通过仿真之后就可以直接交付芯片厂商进行生产。由于绝大部分系统构件都是在系统内部，整个系统特别简洁，不仅减小了系统的体积和功耗，而且提高了系统的可靠性和设计生产效率。

由于 SoC 往往是专用的，所以大部分都不为用户所知，比较典型的 SoC 产品是 Philips 的 Smart XA。少数通用系列如 Siemens 的 TriCore，Motorola 的 M-Core，某些 ARM 系列器件，Echelon 和 Motorola 联合研制的 Neuron 芯片等。

预计不久的将来，一些大的芯片公司将通过推出成熟的、能占领多数市场的 SoC 芯片，一举击退竞争者。SoC 芯片也将在声音、图像、影视、网络及系统逻辑等应用领域中发挥重要作用。

从软件方面划分，主要可以依据操作系统的类型。目前嵌入式系统的软件主要有两大类：实时系统和分时系统。其中实时系统又分为两类：硬实时系统和软实时系统。

实时嵌入系统是为执行特定功能而设计的，可以严格地按时序执行功能。其最大的特征就是程序的执行具有确定性。在实时系统中，如果系统在指定的时间内未能实现某个确定的任务，会导致系统的全面失败，则系统被称为硬实时系统。而在软实时系统中，虽然响应时间同样重要，但是超时却不会导致致命错误。一个硬实时系统往往在硬件上需要添加专门用于时间和优先级管理的控制芯片，而软实时系统则主要在软件方面通过编程实现时限的管理。比如 Windows CE 就是一个多任务分时系统，而 μcos-Ⅱ则是典型的实时操作系统。

嵌入式系统具有非常广阔的应用前景，其应用领域可以包括如下。

1) 工业控制

基于嵌入式芯片的工业自动化设备将获得长足的发展，目前已经有大量的 8、16、32 位嵌入式微控制器在应用中。网络化是提高生产效率和产品质量、减少人力资源主要途径，如工业过程控制、数字机床、电力系统、电网安全、电网设备监测、石油化工系统。就传统的工业控制产品而言，低端产品采用的往往是 8 位单片机。但是随着技术的发展，32 位、64 位的

处理器逐渐成为工业控制设备的核心，在未来几年内必将获得长足的发展。

2）交通管理

在车辆导航、流量控制、信息监测与汽车服务方面，嵌入式系统已经获得了广泛的应用，内嵌 GPS 模块，GSM 模块的移动定位终端已经在各种运输行业获得了成功的使用。目前 GPS 设备已经从尖端产品进入了普通百姓的家庭，只需要几千元，就可以随时随地找到你的位置。

3）信息家电

信息家电将称为嵌入式系统最大的应用领域，冰箱、空调等的网络化、智能化将引领人们的生活步入一个崭新的空间。即使你不在家里，也可以通过电话线、网络进行远程控制。在这些设备中，嵌入式系统将大有用武之地。

4）家庭智能管理系统

水、电、煤气表的远程自动抄表，安全防火、防盗系统，其中嵌有的专用控制芯片将代替传统的人工检查，并实现更高、更准确和更安全的性能。目前在服务领域，如远程点菜器等已经体现了嵌入式系统的优势。

5）POS 网络及电子商务

公共交通非接触智能卡（Contactless Smartcard，CSC）发行系统，公共电话卡发行系统，自动售货机，各种智能 ATM 终端将全面走入人们的生活，到时手持一卡就可以行遍天下。

6）环境工程与自然

水文资料实时监测，防洪体系及水土质量监测、堤坝安全，地震监测网，实时气象信息网，水源和空气污染监测。在很多环境恶劣，地况复杂的地区，嵌入式系统将实现无人监测。

7）机器人

嵌入式芯片的发展将使机器人在微型化，高智能方面优势更加明显，同时会大幅度降低机器人的价格，使其在工业领域和服务领域获得更广泛的应用。

这些应用中，可以着重于在控制方面的应用。就远程家电控制而言，除了开发出支持 TCP/IP 的嵌入式系统之外，家电产品控制协议也需要制订和统一，这需要家电生产厂家来做。同样的道理，所有基于网络的远程控制器件都需要与嵌入式系统之间实现接口，然后再由嵌入式系统来控制并通过网络实现控制。所以，开发和探讨嵌入式系统有着十分重要的意义。

5.3.3　嵌入式处理器

目前世界上具有嵌入式功能特点的处理器已经超过 1000 种，流行体系结构包括 MCU，MPU 等 30 多个系列。鉴于嵌入式系统广阔的发展前景，很多半导体制造商都大规模生产嵌入式处理器，并且公司自主设计处理器也已经成为了未来嵌入式领域的一大趋势，其中从单片机、DSP 到 FPGA 有着各式各样的品种，速度越来越快，性能越来越强，价格也越来越低。目前嵌入式处理器的寻址空间可以从 64KB 到 16MB，处理速度最快可以达到 2000MIPS，封装从 8 个引脚到三百多个引脚不等。

嵌入式微处理器与普通台式计算机的微处理器设计在基本原理上是相似的，但是工作稳定性更高，功耗较小，对环境（如温度、湿度、电磁场、振动等）的适应能力强，体积更小，且集成的功能较多。在桌面计算机领域，对处理器进行比较时的主要指标就是计算速度，从

33MHz 主频的 386 计算机到现在 3GHz 主频的 Pentium 4 处理器，速度的提升是用户最主要关心的变化，但在嵌入式领域，情况则完全不同。嵌入式处理器的选择必须根据设计的需求，在性能、功耗、功能、尺寸和封装形式、SoC 程度、成本、商业考虑等诸多因素之中进行折中，择优选择。

5.3.4 嵌入式系统的组成

一个嵌入式系统装置一般都由嵌入式计算机系统和执行装置组成，嵌入式计算机系统是整个嵌入式系统的核心，由硬件层、中间层、系统软件层和应用软件层组成。执行装置也称为被控对象，它可以接收嵌入式计算机系统发出的控制命令，执行所规定的操作或任务。执行装置可以很简单，如手机上的一个微小型的电机，当手机处于震动接收状态时打开；也可以很复杂，如 SONY 智能机器狗，上面集成了多个微小型控制电机和多种传感器，从而可以执行各种复杂的动作和感受各种状态信息。

下面对嵌入式计算机系统的组成进行介绍。

硬件层：硬件层中包含嵌入式微处理器、存储器（SDRAM、ROM、Flash 等）、通用设备接口和 I/O 接口（A/D、D/A、I/O 等）。在一片嵌入式处理器基础上添加电源电路、时钟电路和存储器电路，就构成了一个嵌入式核心控制模块。其中操作系统和应用程序都可以固化在 ROM 中。

1. 嵌入式微处理器

嵌入式系统硬件层的核心是嵌入式微处理器，嵌入式微处理器与通用 CPU 最大的不同在于嵌入式微处理器大多工作在为特定用户群所专用设计的系统中，它将通用 CPU 许多由板卡完成的任务集成在芯片内部，从而有利于嵌入式系统在设计时趋于小型化，同时还具有很高的效率和可靠性。

嵌入式微处理器的体系结构可以采用冯·诺依曼体系或哈佛体系结构；指令系统可以选用精简指令系统（Reduced Instruction Set Computer，RISC）和复杂指令系统（Complex Instruction Set Computer，CISC）。RISC 计算机在通道中只包含最有用的指令，确保数据通道快速执行每一条指令，从而提高了执行效率并使 CPU 硬件结构设计变得更为简单。

嵌入式微处理器有各种不同的体系，即使在同一体系中也可能具有不同的时钟频率和数据总线宽度，或集成了不同的外设和接口。据不完全统计，目前全世界嵌入式微处理器已经超过 1000 多种，体系结构有 30 多个系列，其中主流的体系有 ARM、MIPS、PowerPC、X86 和 SH 等。但与全球 PC 市场不同的是，没有一种嵌入式微处理器可以主导市场，仅以 32 位的产品而言，就有 100 种以上的嵌入式微处理器。嵌入式微处理器的选择是根据具体的应用而决定的。

2. 存储器

嵌入式系统需要存储器来存放和执行代码。嵌入式系统的存储器包含 Cache、主存和辅助存储器。

1) Cache

Cache 是一种容量小、速度快的存储器阵列，它位于主存和嵌入式微处理器内核之间，

存放的是最近一段时间微处理器使用最多的程序代码和数据。在需要进行数据读取操作时，微处理器尽可能地从 Cache 中读取数据，而不是从主存中读取，这样就大大改善了系统的性能，提高了微处理器和主存之间的数据传输速率。Cache 的主要目标就是：减小存储器（如主存和辅助存储器）给微处理器内核造成的存储器访问瓶颈，使处理速度更快，实时性更强。

在嵌入式系统中 Cache 全部集成在嵌入式微处理器内，可分为数据 Cache、指令 Cache 或混合 Cache，Cache 的大小依不同处理器而定。一般中高档的嵌入式微处理器才会把 Cache 集成进去。

2）主存

主存是嵌入式微处理器能直接访问的寄存器，用来存放系统和用户的程序及数据。它可以位于微处理器的内部或外部，其容量为 256KB～1GB，根据具体的应用而定，一般片内存储器容量小，速度快，片外存储器容量大。

常用作主存的存储器有：

ROM 类 NOR Flash、EPROM 和 PROM 等。

RAM 类 SRAM、DRAM 和 SDRAM 等。

其中 NOR Flash 凭借其可擦写次数多、存储速度快、存储容量大、价格便宜等优点，在嵌入式领域内得到了广泛应用。

3）辅助存储器

辅助存储器用来存放大数据量的程序代码或信息，它的容量大、但读取速度与主存相比就慢很多，用来长期保存用户的信息。

嵌入式系统中常用的外存有：硬盘、NAND Flash、CF 卡、MMC 和 SD 卡等。

3. 通用设备接口和 I/O 接口

嵌入式系统和外界交互需要一定形式的通用设备接口，如 A/D、D/A、I/O 等，外设通过和片外其他设备的或传感器的连接来实现微处理器的输入输出功能。每个外设通常都只有单一的功能，它可以在芯片外也可以内置芯片中。外设的种类很多，可从一个简单的串行通信设备到非常复杂的 802.11 无线设备。

目前嵌入式系统中常用的通用设备接口有 A/D（模/数转换接口）、D/A（数/模转换接口），I/O 接口有 RS-232 接口（串行通信接口）、Ethernet（以太网接口）、USB（通用串行总线接口）、音频接口、VGA 视频输出接口、I2C（现场总线）、SPI（串行外围设备接口）和 IrDA（红外线接口）等。

中间层：硬件层与软件层之间为中间层，也称为硬件抽象层（Hardware Abstract Layer，HAL）或板级支持包（Board Support Package，BSP），它将系统上层软件与底层硬件分离开来，使系统的底层驱动程序与硬件无关，上层软件开发人员无需关心底层硬件的具体情况，根据 BSP 层提供的接口即可进行开发。该层一般包含相关底层硬件的初始化、数据的输入输出操作和硬件设备的配置功能。BSP 具有以下两个特点。

硬件相关性：因为嵌入式实时系统的硬件环境具有应用相关性，而作为上层软件与硬件平台之间的接口，BSP 需要为操作系统提供操作和控制具体硬件的方法。

操作系统相关性：不同的操作系统具有各自的软件层次结构，因此，不同的操作系统具

有特定的硬件接口形式。

实际上，BSP 是一个介于操作系统和底层硬件之间的软件层次，包括了系统中大部分与硬件联系紧密的软件模块。设计一个完整的 BSP 需要完成两部分工作：嵌入式系统的硬件初始化以及 BSP 功能，设计硬件相关的设备驱动。

1）嵌入式系统硬件初始化

系统初始化过程可以分为 3 个主要环节，按照自底向上、从硬件到软件的次序依次为片级初始化、板级初始化和系统级初始化。

片级初始化：完成嵌入式微处理器的初始化，包括设置嵌入式微处理器的核心寄存器和控制寄存器、嵌入式微处理器核心工作模式和嵌入式微处理器的局部总线模式等。片级初始化把嵌入式微处理器从上电时的默认状态逐步设置成系统所要求的工作状态。这是一个纯硬件的初始化过程。

板级初始化：完成嵌入式微处理器以外的其他硬件设备的初始化。另外，还需设置某些软件的数据结构和参数，为随后的系统级初始化和应用程序的运行建立硬件和软件环境。这是一个同时包含软硬件两部分在内的初始化过程。

系统级初始化：该初始化过程以软件初始化为主，主要进行操作系统的初始化。BSP 将对嵌入式微处理器的控制权转交给嵌入式操作系统，由操作系统完成余下的初始化操作，包含加载和初始化与硬件无关的设备驱动程序，建立系统内存区，加载并初始化其他系统软件模块，如网络系统、文件系统等。最后，操作系统创建应用程序环境，并将控制权交给应用程序的入口。

2）硬件相关的设备驱动程序

BSP 的另一个主要功能是硬件相关的设备驱动。硬件相关的设备驱动程序的初始化通常是一个从高到低的过程。尽管 BSP 中包含硬件相关的设备驱动程序，但是这些设备驱动程序通常不直接由 BSP 使用，而是在系统初始化过程中由 BSP 将它们与操作系统中通用的设备驱动程序关联起来，并在随后的应用中由通用的设备驱动程序调用，实现对硬件设备的操作。与硬件相关的驱动程序是 BSP 设计与开发中另一个非常关键的环节。

系统软件层：系统软件层由实时多任务操作系统（Real-Time Operation System，RTOS）、文件系统、图形用户接口（Graphic User Interface，GUI）、网络系统及通用组件模块组成。RTOS 是嵌入式应用软件的基础和开发平台。

5.4 嵌入式系统开发技术

5.4.1 嵌入式系统的结构设计

1. 嵌入式系统的硬件架构

嵌入式系统硬件结构主要由微处理器 MPU、外围电路，以及外设组成，微处理器为 ARM 嵌入式处理芯片，如 ARM7TMDI 系列及 ARM9 系列微处理器，MPU 为整个嵌入式系统硬件的核心，决定了整个系统功能和应用领域。外围电路根据微处理器不同而略有不同，主要由电源管理模型、时钟模块、闪存 Flash、随机存储器 RAM，以及只读存储器 ROM

组成。这些设备是一个微处理器正常工作所必需的设备。外部设备将根据需要而各不相同，如通用通信接口 USB、RS-232、RJ-45 等，输入输出设备，如键盘、LCD 等。外部设备将根据需要定制。嵌入式处理系统主要包括嵌入式微处理器、存储设备、模拟电路及电源电路、通信接口以及外设电路。

2. 嵌入式系统的软件结构

嵌入式系统与传统的单片机在软件方面最大的不同点就是可以移植操作系统，从而使软件设计层次化，传统的单片机在软件设计时将应用程序与系统、驱动等全部混在一起编译，系统的可扩展性，可维护性不高，上升到操作系统后，这一切变得简单可行。

嵌入式操作系统在软件上呈现明显的层次化，从与硬件相关的 BSP 到实时操作系统内核 RTOS，到上层文件系统、GUI 界面，以及用户层的应用软件。各部分可以清晰地划分开来。当然，在某些时候划分也不完全符合应用要求，需要程序设计人员根据特定的需要来设计自己的软件。

5.4.2　嵌入式系统的设计方法

1. 嵌入式系统的设计流程

如图 5.14 所示，嵌入式系统设计一般由 5 个阶段构成：系统需求分析，体系结构设计，硬件、软件设计，系统集成和系统测试。各个阶段之间往往要求不断的反复和修改，直至完成最终设计目标。

1）系统需求分析

确定设计任务和设计目标，并提炼出设计规格说明书，作为正式设计指导和验收的标准。系统的需求一般分功能性需求和非功能性需求两方面。功能性需求是系统的基本功能，如输入输出信号、操作方式等；非功能需求包括系统性能、成本、功耗、体积、重量等因素。

2）体系结构设计

描述系统如何实现所述的功能和非功能需求，包括对硬件、软件和执行装置的功能划分以及系统的软件、硬件选型等。一个好的体系结构是设计成功与否的关键。

3）硬件、软件设计

基于体系结构，对系统的软件、硬件进行详细设计。为了缩短产品开发周期，设计往往是并行的。应该说，嵌入式系统设计的工作大部分都集中在软件设计上，采用面向对象技术、软件组件技术、模块化设计是现代软件工程经常采用的方法。

4）系统集成

把系统的软件、硬件和执行装置集成在一起，进行调试，发现并改进单元设计过程中的错误。

5）系统测试

对设计好的系统进行测试，看其是否满足规格说明书中给定的功能要求。针对系统的不同的复杂程度，目前有一些常用的系统设计方法，如瀑布设计方法、自顶向下的设计方法、自下向上的设计方法、螺旋设计方法、逐步细化设计方法和并行设计方法等。根据设计对象复杂程度的不同，可以灵活地选择不同的系统设计方法。

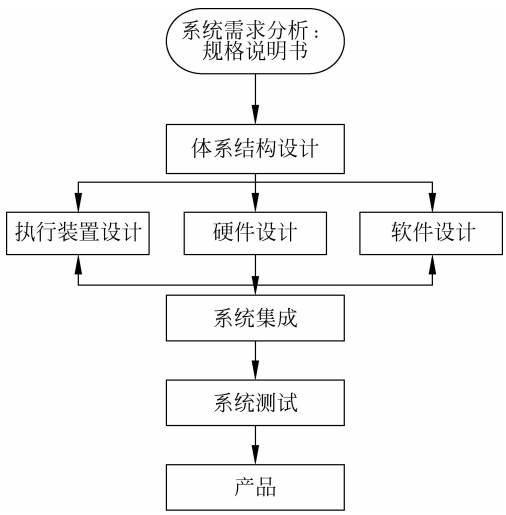


图 5.14　嵌入式系统的设计阶段

2. 嵌入式系统的一般设计方法

通常在嵌入式系统的开发和应用中，是按照如图 5.15 所示的流程进行的。整个系统的开发过程将改变为如图 5.16 所示的过程。

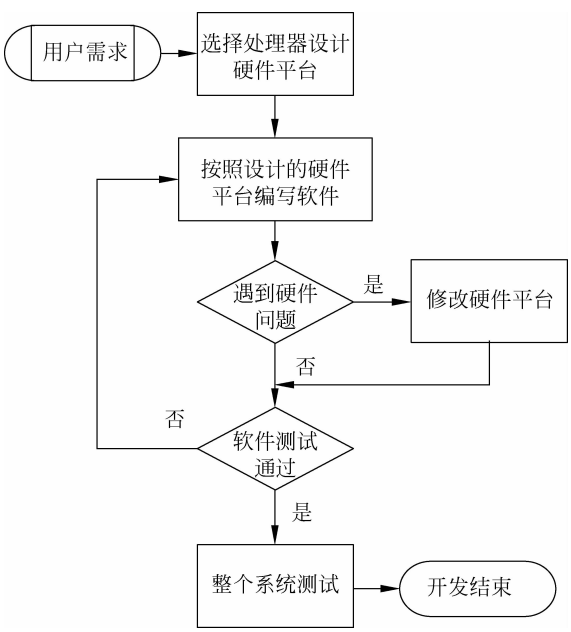


图 5.15　嵌入式系统的开发流程

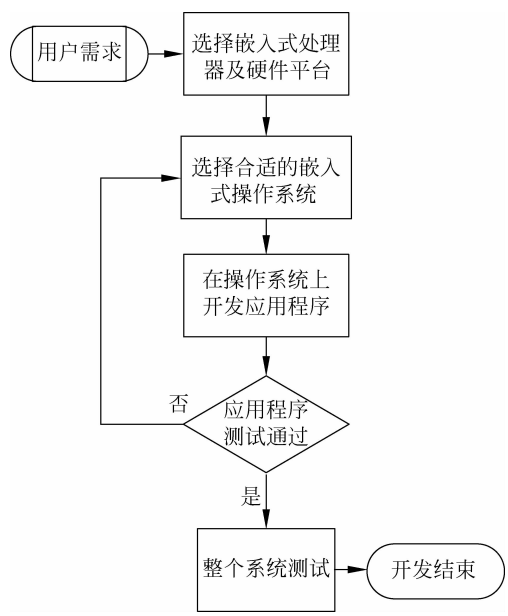


图 5.16　嵌入式系统的开发过程

可见，在应用嵌入式系统开发的过程中，因为对应于每一个处理器的硬件平台都是通用的、固定的、成熟的，所以在开发过程中减少了硬件系统错误的引入机会；同时，因为嵌入式操作系统屏蔽了底层硬件的很多复杂信息，使得开发者通过操作系统提供的 API 函数可以

完成大部分工作，大大地简化了开发过程，提高了系统的稳定性。

综上所述，嵌入式系统的开发可以说是把开发者从反复进行硬件平台的设计过程中解放出来，从而可以把主要的精力放在编写特定的应用程序上。这个过程更类似于在系统机（如 PC）上的某个操作系统下开发应用程序。

3. 嵌入式系统的硬件/软件协同设计技术

传统的嵌入式系统设计方法如图 5.17 所示，硬件和软件分为两个独立的部分，由硬件工程师和软件工程师按照拟定的设计流程分别完成。这种设计方法只能改善硬件/软件各自的性能，而有限的设计空间不可能对系统做出较好的性能综合优化。20 世纪 90 年代初，国外有些学者提出“这种传统的设计方法，只是早期计算机技术落后的产物，它不能求出适合于某个专用系统的最佳计算机应用系统的解”。因为，从理论上来说，每一个应用系统，都存在一个适合于该系统的硬件、软件功能的最佳组合，如何从应用系统需求出发，依据一定的指导原则和分配算法对硬件/软件功能进行分析及合理的划分，从而使系统的整体性能、运行时间、能量耗损、存储能量达到最佳状态，已成为硬件/软件协同设计的重要研究内容之一。

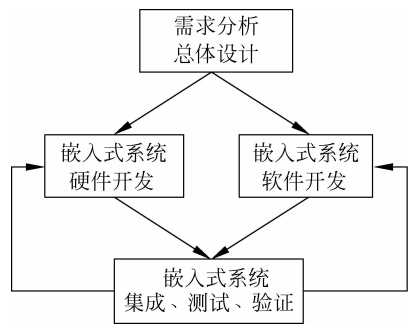


图 5.17　传统的嵌入式系统的设计方法

应用系统的多样性和复杂性，使硬件/软件的功能划分、资源调度与分配、系统优化、系统综合、模拟仿真存在许多需要研究解决的问题，因而使国际上这个领域的研究日益活跃。系统协同设计与传统设计相比有两个显著的特点：

（1）描述硬件和软件使用统一的表示形式；

（2）硬件、软件划分可以选择多种方案，直到满足要求。

显然，这种设计方法对于具体的应用系统而言，容易获得满足综合性能指标的最佳解决方案。传统方法虽然也可改进硬件、软件性能，但由于这种改进是各自独立进行的，不一定能使系统综合性能达到最佳。传统的嵌入式系统开发采用的是软件开发与硬件开发分离的方式，其过程可描述如下。

（1）需求分析；

（2）软硬件分别设计、开发、调试、测试；

（3）系统集成：软硬件集成；

（4）集成测试；

（5）若系统正确，则结束，否则继续进行；

（6）若出现错误，需要对软、硬件分别验证和修改；

（7）返回(3)，继续进行集成测试。

虽然在系统设计的初始阶段考虑了软硬件的接口问题，但由于软硬件分别开发，各自部分的修改和缺陷很容易导致系统集成出现错误。由于设计方法的限制，这些错误不但难以定位，而且更重要的是，对它们的修改往往会涉及整个软件结构或硬件配置的改动。显然，这是灾难性的。

为避免上述问题，一种新的开发方法应运而生——软硬件协同设计方法。一个典型的硬件/软件协同设计过程如图 5.18 所示。首先，应用独立于任何硬件和软件的功能性规格方法对系统进行描述，采用的方法包括有限态自动机(FSM)、统一化的规格语言(CSP、VHDL)或其他基于图形的表示工具，其作用是对硬件/软件统一表示，便于功能的划分和综合；然后，在此基础上对硬件/软件进行划分，即对硬件、软件的功能模块进行分配。但是，这种功能分配不是随意的，而是从系统功能要求和限制条件出发，依据算法进行的。完成硬件/软件功能划分之后，需要对划分结果做出评估。方法之一是性能评估，另一种方法是对硬件、软件综合之后的系统依据指令级评价参数做出评估。如果评估结果不满足要求，说明划分方案选择不合理，需要重新划分硬件/软件模块，以上过程重复直到系统获得一个满意的硬件/软件实现为止。

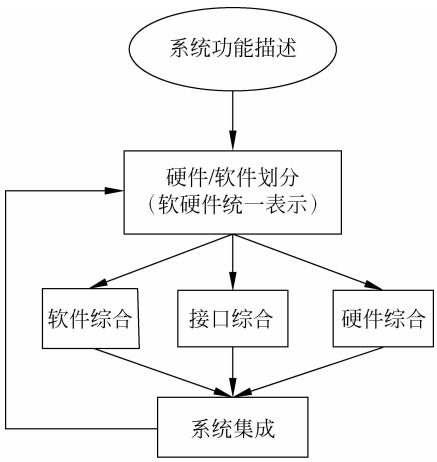


图 5.18 嵌入式系统的硬件/软件协同设计方法

软硬件协同设计过程可归纳为：

(1) 需求分析；

(2) 软硬件协同设计；

(3) 软硬件实现；

(4) 软硬件协同测试和验证。

这种方法的特点是在协同设计(Co-design)、协同测试(Co-test)和协同验证(Co-verification)上，充分考虑了软硬件的关系，并在设计的每个层次上给以测试验证，使得尽早发现和解决问题，避免灾难性错误的出现。

5.4.3 嵌入式系统的开发技术

当进行嵌入式系统开发时，选择合适的开发工具可以加快开发进度、节省开发成本。因此一套含有编辑软件、编译软件、汇编软件、连接软件、调试软件、工程管理及函数库的集成开发环境(IDE)是必不可少的。

ARM SDT 是 ARM Software Development Toolkit 的简写，是 ARM 公司为方便用户在 ARM 芯片上进行应用软件开发而推出的一整套集成开发工具。ARM SDT 由一套完备的应用程序构成，并附带支持文档和例子，可以用来编写和调试 ARM 系列的 RISC 处理器的应用程序，可以开发 C++或 ARM Assembly 程序。它的 Windows 开发工具有两个：APM 和 ADW。

下面以北京博创兴业科技有限公司的 UP-NetARM300 嵌入式开发板为例，介绍 ARM SDT 2.5 仿真开发环境。ARM SDT 2.5 仿真开发环境的配置方法如下：

(1) 运行 ARM SDT 2.5 集成开发环境(ARM Project Manager)。选择 File|New 菜单，新建一个工程文件(workl.apj)。

(2) 在新建的工程中，使用菜单对整个工程的汇编进行设置。

(3) 选中工程树的“根部”,通过菜单对整个工程的连接方式进行设置。

(4) 在弹出的对话框中,选中 Entry and Base 标签,设置连接的 Read-Only(只读)和 Read-Write(读写)地址。地址 0x0c080000 是开发板上 SDRAM 的真实地址,是由系统的硬件决定的;0x0c200000 指的是系统可读写的内存的地址。也就是说,在 0x0e080000～0xc1ffffff 之间是只读区域,存放程序的代码段,从 0x0c200000 开始是程序的数据段。

(5) 选择 Lillker Configuration 的 ImageLayout 标签,设置程序的入口模块。指定在生成的代码中,程序是从 44binit.s 开始运行的。

(6) 选择 Project | Edit Project Templete 菜单,使用 New 按钮,为编译器新建一个 Romlmage 文件。

(7) 设置 Romlmage 的内容。使编译器编译的时候可以生成 system.bin 文件,这就是系统的启动文件。

(8) 将 build_target 变量设为 system.bin。使编译器编译的时候可以生成 system.bin 文件,这就是系统的启动文件。

(9) 在 Project Template Editor 对话框中,单击 Edit Detail 按钮,在弹出的对话框中可以重新命名模板。

至此,工程文件设置完毕。因为设置过程比较繁琐,可以保存此工程,下次新建项目的时候复制即可。然后,在新建的工程中加入相应 *.c 文件及 *.h 文件即可进行编辑、编译、修改和调试。

5.4.4　嵌入式系统开发技术的发展趋势及其挑战

1. 嵌入式系统设计的新发展

随着微电子技术的飞速发展,微处理器已经变成低成本器件。在可能的情况下,各种机电设备已经或者正在嵌入 CPU 构成的嵌入式系统。现在系统研究的重点已从通用系统转向专用系统,以及从一般性能转向可靠性、可用性、安全性、自主性、可扩展性、功能性、灵活性、成本、体积、功耗及可管理性上。嵌入式应用从以前的简单控制发展到今天,已经有很多非常复杂,非常高端的应用。例如苹果公司最近推出的 iPhone 手机,里面有 ARM11,有 ARM9,也有 ARM7 MCU。

(1) 32 位嵌入式处理器比例快速升高,InStat/MDR 曾预测在 2001—2006 年期间,32 位向控制器(MCU)的复合年增长率可达 22.6%。而全球 32 位 MCU 市场在 2003 年的增长幅度实际已走过 30%,在 2004 年达到 38%。

(2) 可供选择的可编程计算部件方案增多。

① 除过去常用的通用处理器 GPP(Gereral Purpose Processor)、嵌入式处理器 EP(Embedded Processor)、微控制器 MCU(Micro-Control Unit)、数字信号处理器 DSP(Digital Signal Processor)外,目前发展很快,可以给我们提供新的选择的还有各种专用处理器 ASP(Application Specific Processor)或专用标准产品 ASSP(Application Specific Standard Product)。它们都是针对一些特定应用而设计的,如用于 HDTV、ADSL、Cable Modem 等的专用处理器。与 MCU 相比,ASP/ASSP 集成的资源可能比一般 MCU 更多、更专业化,所以 ASP 的价格要高于 MCU。

② 目前的一个发展趋势是以 FPGA 为代表的现场可编程技术在迅速崛起。这是由于市场对通用可配置处理器的呼声越来越高，传统的 MCU 在市场需求中显得越来越力不从心。可配置、可扩展处理器逐渐浮出水面。利用半定制器件可以构成基本 FPGA 的硬核处理器或基于 FPGA 的软核处理器，并由此可编程片上系统应运而生。这是一种面向消费电子、工业、办公自动化、电信和汽车应用中的嵌入式控制功能而开发的高性能、现场可编程、混合信号阵列。它集 MCU 和 FPGA/CPLD 的优点于一身，实现可配置 SoC，既适应了设计人员对系统部件集成的需要，又能实现可配置需求的灵活性，从而为许多现实应用提供一种平衡解决方案。

(3) 微控制器的发展特点。

尽管由于市场对多功能产品需求的增加和 IT 技术的推动，使 32 位 MCU 产品日益成为市场的热点；但目前 8 位 MCU 仍然是技术市场的主流，并且还有相当广阔的应用空间和旺盛的生命力，16 位 MCU 也占有一定的市场份额。各种 MCU 根据自己在市场上的定位，也都有了很大发展，MCU 总的发展具有以下一些特点：

① 微控制器 SoC 化；

② 多核结构处理器；

③ 更低功耗；

④ 更宽工作电压范围；

⑤ 更先进的工艺和更小的封装；

⑥ 低噪声布线技术。

2. 嵌入式系统设计的新挑战

要求更高的应用需求推动嵌入式设计正在由 8/16 位转向功能更强大的 32 位 MCU。这种升级给工程师带来性能空间和处理速度提升的同时也带来了严峻的挑战，提出了一系列前所未有的全新问题。首先，要对开发工具和软件进行新的投资，并对设计流程进行重新定义；另外，要对原软件能多大程度地用于新架构、要把已有软件移植到 32 位的新架构上还必须做多少工作，以及器件和开发工具的成本、存储器的种类、规模、性能和容量、可选器件的种类等进行评估。是否向 32 位升级主要取决于经过综合考虑后的总的系统成本等。

1) 发生了哪些变化

国内原来熟悉 8 位 MCU 开发的工程师大部分出身于电子工程和其他机电专业，而非计算机专业。随着嵌入式系统设计技术的发展，已经在很多方面发生了很大的变化。这与传统的 8 位 MCU 的开发有着许多明显的不同：首先是复杂度大为提高，其次开发形式、手段和工具也有了很大不同；另外系统越来越多地是建立在 RTOS 平台上，使用的开发程序设计语言不再是开始效率很低的汇编语言，而越来越多地使用开发效率很高的高级语言。C 语言已成为主流通用开发语言。

(1) 开发的复杂度越来越大；

(2) 开发形式、手段和工具越来越多；

(3) 开发平台使用 RTOS：RTOS 的引入解决了嵌入式软件开发标准化的难题，促进嵌入式开发软件的模块化和可移植化，为软件工程化打下基础。随着嵌入式系统中软件比重不断上升、应用程序越来越大，这对开发人员的知识结构、应用程序接口和程序档案的组织

管理等都提出了新的要求。引入 RTOS 相当于引入了一种新的管理模式，对于开发单片机和开发人员都是一个飞跃。

2）设计者面对的新挑战

（1）转变观念，需要熟悉新的开发模式。

嵌入式系统应用不再是过去单一的单片机应用模式，而是越来越多样化，这可为用户提供更多的不同层次的选择方案。嵌入式系统实现的最高形式是片上系统 SoC，而 SoC 的核心技术是重用和组合 IP 核构件。从单片机应用设计到片上系统设计及其中间的一系列的变化，从底层大包大揽的设计到利用 FPGA 和 IP 模块进行功能组合 PSoC/SoPC 设计，这是一个观念的转变。学习和熟悉新的开发模式将会事半功倍地构建功能强大和性能卓越的嵌入式系统，但同时也给系统的设计验证工作提出了许多新的挑战。

（2）进入的技术门槛提高，需要学习全新的 RTOS 技术。

现代高端嵌入式系统都是建立在 RTOS 基础上的。这对于未受过计算机专业训练的各专业领域的工程技术人员来说，需要学习全新的 RTOS 技术，深入了解 RTOS 的工作机制和系统的资源配置，掌握底层软件、系统软件和应用软件的设计和调试方式。进入的技术门槛要比所熟悉的开发方法高得多。这对于开发者来说，也是一个新的挑战。

（3）选择适合的开发工具，熟悉新的开发环境。

目前从 8 位升级到 32 位的一个最大障碍就是开发工具的投入。32 位开发工具要比 8 位开发工具复杂得多，使用的技术门槛要高得多，同时其投资也要高得多。进入 32 位系统开发的工程师不得不面对与 8 位系统很不相同的开发环境。如何正确选择处理器架构、评估嵌入式操作系统，以及使用陌生的开发工具，都是一个新的挑战。

（4）熟悉硬件/软件协同设计和验证技术、设计管理技术。

软/硬件并行设计是嵌入式系统设计的一项关键任务。在设计过程中的主要问题，是软硬件设计的同步与集成。这要求控制一致性与正确性，但随着技术细节不断增加，需要消耗大量时间。目前，业界已经开发 Polis、Cosyma 及 Chinook 等多种方法和工具来支持集成式软硬件协同设计。目标是提供一种统一的软硬件开发方法，它支持设计空间探索，并使系统功能可以跨越硬件和软件平台复用。

团队开发的最大问题就是设计管理问题。现在有越来越多的公司开始重视技术管理，利用各种技术管理软件（例如软件版本管理软件）对全过程进行监督管理。这对每一个参与开发的人来说，似乎增加了不少麻烦，但是对整个公司的产品上市、升级、维护以及战略利益都具有长远的效益。

（5）SoC 设计所面临的巨大的挑战。

SoC 已经开始成为新一代应用电子技术的核心，这已成为电子技术的革命标志。过去应用工程师面对的是各种 ASIC 电路，而现在越来越多所面对的是巨大的 IP 模块库，所有设计工作都是以 IP 模块为基础。SoC 设计技术使嵌入式系统设计工程师变成了一个面向应用的电子器件设计工程师。随着 SoC 应用的日益普及，在测试程序生成、工程开发、硅片查错、量产等领域对 SoC 测试技术提出了越来越高的要求。掌握新的测试理念及新的测试流程、方法和技术，是对单片机应用工程师提出的新挑战。

5.5 Keil 与 Proteus 用法简介

1. Keil 软件使用方法简介

Keil C51 软件是众多单片机应用开发的优秀软件之一，它集编辑、编译、仿真于一体，支持汇编，汇编语言和 C 语言的程序设计，界面友好，易学易用。

下面介绍 Keil C51 软件的使用方法。

学习程序设计语言、某种程序软件，最好的方法是直接操作实践。下面通过简单的编程、调试，引导大家学习 Keil C51 软件的基本使用方法和基本的调试技巧。

1）新建工程

单击 Project 菜单，在弹出的下拉菜单中选中 New Project 选项。

然后选择要保存的路径，输入工程文件的名字，比如保存到 D 盘的 CMJ51 文件夹里，工程文件的名字为 CMJ51；这时会弹出一个对话框，要求选择单片机的型号，读者可以根据使用的单片机来选择，Keil C51 几乎支持所有的 51 核的单片机，这里还是以大家用得比较多的 Atmel 的 89C51 来说明，选择 89C51 之后，然后单击“确定”按钮。

2）新建文件

单击 File 菜单，再在下拉菜单中单击 New 选项。

单击 File 菜单下的 save，出现一个对话框，输入文件名，后缀名为 *.asm，进行保存到 D 盘 CMJ51 文件夹下。

3）添加文件

回到编辑界面后，单击 Target 1 前面的＋号，然后在 Source Group 1 上单击右键，然后单击 Add File to Group “Source Group 1”。选中 cmj1.asm，然后单击 Add。

4）汇编连接

单击图标(rebuilt all target files)对文件 cmj1.asm 进行编译。

5）运行看结果

单击图标(start/stop debug session)，进入运行状态。单击一次图标(step into)，程序就能运行一条，一直到程序的结束(ret)。

特殊功能寄存器中的内容可以在界面中直接看到，存储单元的内容在 memory ♯1 中输入如某一单元的地址，即可看到该单元的内容。

2. Proteus 软件使用方法简介

Proteus 软件是 Labcenter Electronics 公司的一款电路设计与仿真软件，它包括 ISIS、ARES 等软件模块，ARES 模块主要用来完成 PCB 的设计，而 ISIS 模块用来完成电路原理图的布图与仿真。Proteus 的软件仿真基于 VSM 技术，它与其他软件最大的不同也是最大的优势就在于它能仿真大量的单片机芯片，比如 MCS-51 系列、PIC 系列等，以及单片机外围电路，比如键盘、LED、LCD 等。通过 Proteus 软件的使用读者能够轻易地获得一个功能齐全、实用方便的单片机模拟实验室。

本文中由于主要使用 Proteus 软件在单片机方面的仿真功能，所以本章重点研究 ISIS

模块的用法。在下面的内容中，如不特别说明，本章所说的 Proteus 软件特指其 ISIS 模块。

下面来熟悉一下 Proteus 的界面。Proteus 是一个标准的 Windows 窗口程序，和大多数程序一样，没有太大区别，其启动界面如图 5.19 所示。

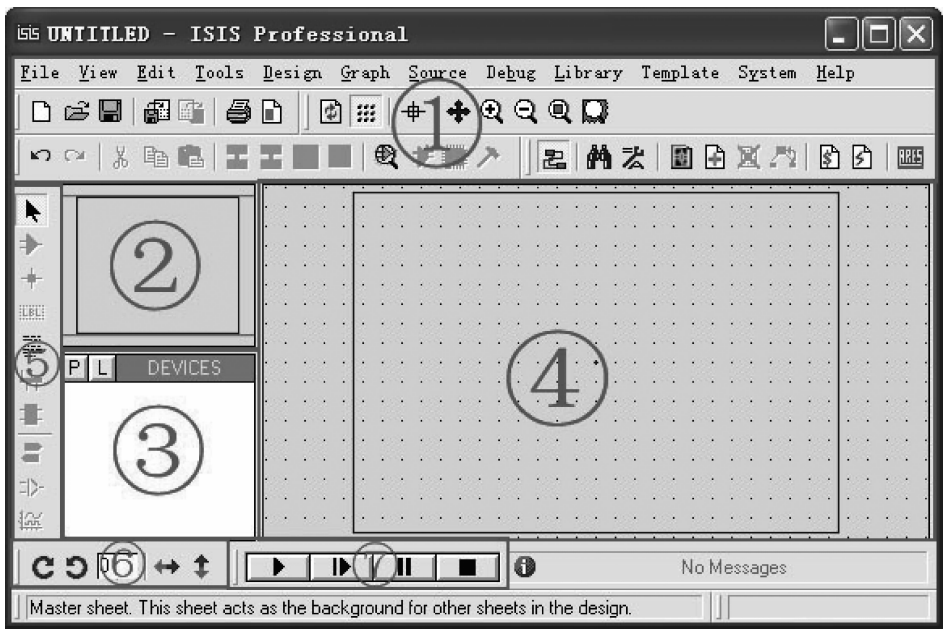


图 5.19　Proteus 标准窗口

如图 5.19 中所示，区域①为菜单及工具栏，区域②为预览区，区域③为元器件浏览区，区域④为编辑窗口，区域⑤为对象拾取区，区域⑥为元器件调整工具栏，区域⑦为运行工具条。

下面就以建立一个和前文在 Keil 简介中所讲的工程项目相配套的 Proteus 工程为例来详细讲述 Proteus 的操作方法以及注意事项。

首先单击启动界面区域③中的 P 按钮(Pick Devices，拾取元器件)来打开 Pick Devices(拾取元器件)对话框从元件库中拾取所需的元器件。对话框如图 5.20 所示。

在对话框中的 Keywords 里面输入要检索的元器件的关键词，比如要选择项目中使用的 AT89C51，就可以直接输入。输入以后能够在中间的 Results 结果栏里面看到搜索的元器件的结果。在对话框的右侧，还能够看到选择的元器件的仿真模型、引脚以及 PCB 参数。

这里有一点需要注意，可能有时候选择的元器件并没有仿真模型，对话框将在仿真模型和引脚一栏中显示 No Simulator Model(无仿真模型)。那么我们就不能够用该元器件进行仿真了，或者我们只能做它的 PCB 板，或者我们选择其他的与其功能类似而且具有仿真模型的元器件。

搜索到所需的元器件以后，可以双击元器件名来将相应的元器件加入到文档中，那么接着还可以用相同的方法来搜索并加入其他的元器件。当我们已经将所需的元器件全部加入到文档中时，我们可以单击 OK 按钮来完成元器件的添加。

添加好元器件以后，下面我们所需要做的就是将元器件按照我们的需要连接成电路。

首先在元器件浏览区中单击我们需要添加到文档中的元器件，这时我们就可以在浏览区看到所选择的元器件的形状与方向。如果其方向不符合读者的要求，读者可以通过单击元器件调整工具栏中的工具来任意进行调整，调整完成之后在文档中单击并选定好需要放置的位置即可。接着按相同的操作即可完成所有元器件的布置，接下来是连线。事实上 Proteus 的自动布线功能是如此的完美以至于在做布线时从来都不会觉得这是一项任务，而通常像是在享受布线的乐趣。布线时只需要单击选择起点，然后在需要转弯的地方单击一下，按照读者所需走线的方向移动鼠标到线的终点单击即可。本例布线的结果如图 5.21 所示(仿真在上面的 Keil 操作介绍中的简单例子)。

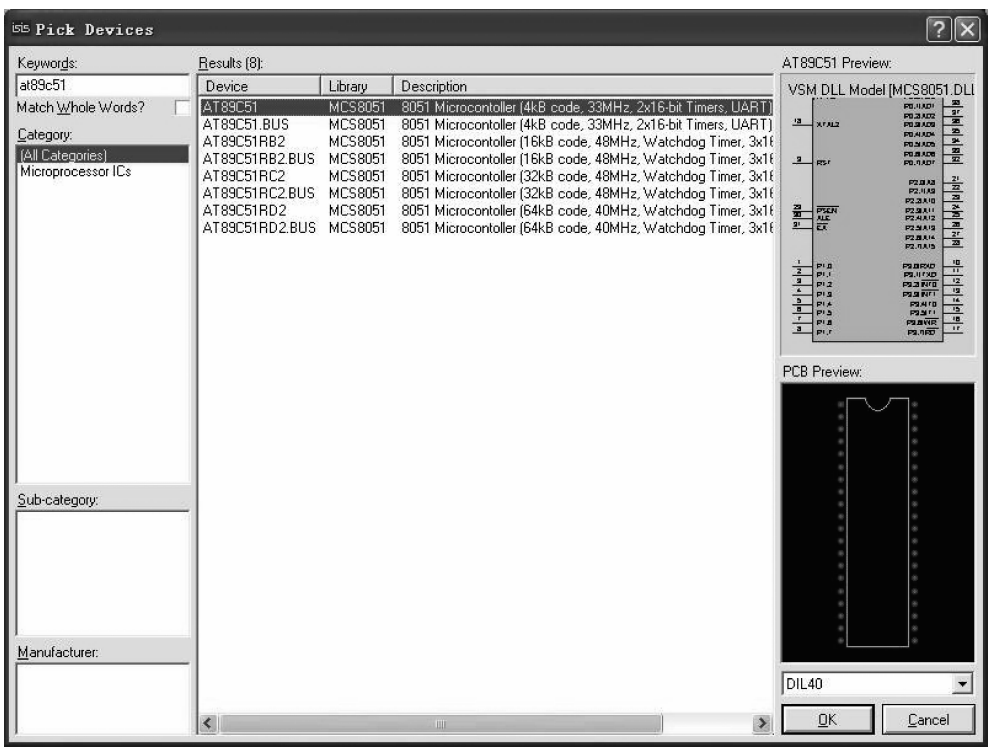


图 5.20 Proteus 放置元器件

因为该工程十分简单，没有必要加上复位电路，所以这点在图中予以忽略，请读者注意。除此以外，读者可能还发现，单片机系统没有晶振，这一点需注意。事实上在 Proteus 中单片机的晶振可以省略，系统默认为 12MHz，而且很多时候，当然也为了方便，只需要取默认值就可以了。

下面来添加电源。先说明一点，Proteus 中单片机芯片默认已经添加电源与地，所以可以省略。然后在元器件浏览区中单击 POWER(电源)来选中电源，通过区域⑥中的元器件调整工具进行适当的调整，然后就可以在文档区中单击放置电源了。放置并连接好线路的电路图一部分如图 5.22 所示。

连接好电路图以后还需要做一些修改。由图 5.22 可以看出，图中的 R_1 电阻值为 10kΩ，这个电阻作为限流电阻显然太大，将使发光二极管 D_1 亮度很低或者根本就不亮，影响仿真结果，所以要进行修改。修改方法如下：首先双击电阻图标，这时软件将弹出 Edit

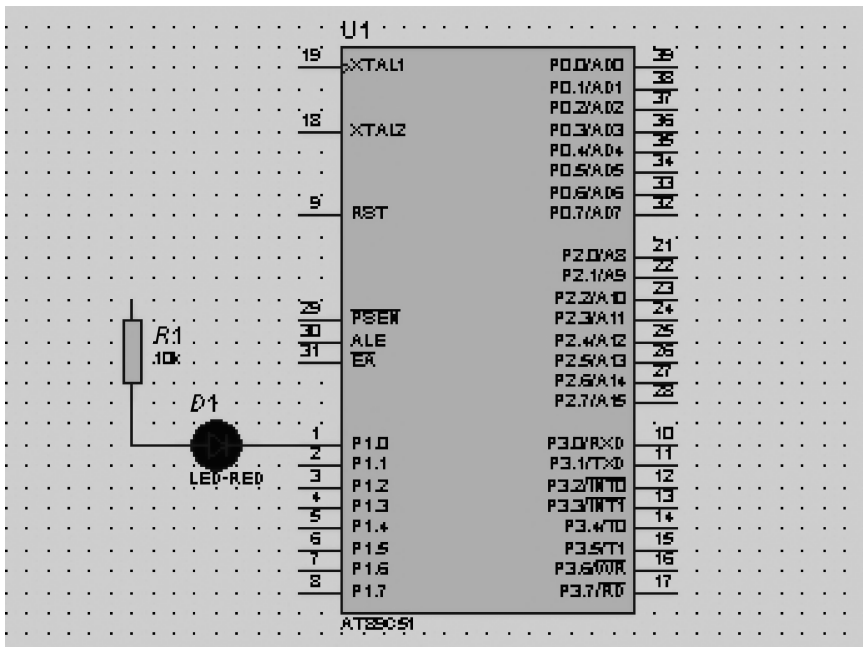


图 5.21　仿真电路图

Component 对话框，对话框中的 Component Referer 是组件标签之意，可以随便填写，也可以取默认，但要注意在同一文档中不能有两个组件标签相同；Resistance 就是电阻值了，可以在其后的框中根据需要填入相应的电阻值。填写时需注意其格式，如果直接填写数字，则单位默认为 Ω；如果在数字后面加上 K 或者 k，则表示 kΩ 之意。这里填入 270，表示 270Ω。

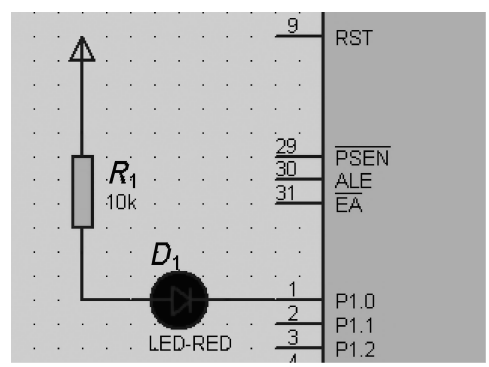


图 5.22　发光二极管电路图

修改好各组件属性以后就要将程序（HEX 文件）载入单片机了。首先双击单片机图标，系统同样会弹出 Edit Component 对话框，在这个对话框中单击 Program files 框右侧的 ，来打开选择程序代码窗口，选中相应的 HEX 文件后返回，这时，按钮左侧的框中就填入了相应的 HEX 文件，单击对话框的 OK 按钮，回到文档，程序文件就添加完毕了。

装载好程序，就可以进行仿真了。首先来熟悉一下上面第一个图中区域⑦的运行工具条。因为比较简单，只作以下介绍：

工具条从左到右依次是 Play、Step、Pause、Stop 按钮，即运行、步进、暂停、停止。下面单击 Play 按钮来仿真运行，可以看到系统按照程序在运行着，而且还能看到其高低电平的实时变化。如果已经观察到了结果就可以单击 Stop 来停止运行。

3. KeilC 与 Proteus 连接调试

下面以一个简单的实例来完整的展示一个 KeilC 与 Proteus 相结合的仿真过程。

单片机电路设计：

如图 5.23 所示，电路的核心是单片机 AT89C51。单片机的 P_1 口 8 个引脚接 LED 显示器的段选码(a、b、c、d、e、f、g、dp)的引脚上，单片机的 P_2 口 6 个引脚接 LED 显示器的位选码(1、2、3、4、5、6)的引脚上，电阻起限流作用，总线使电路图变得简洁。

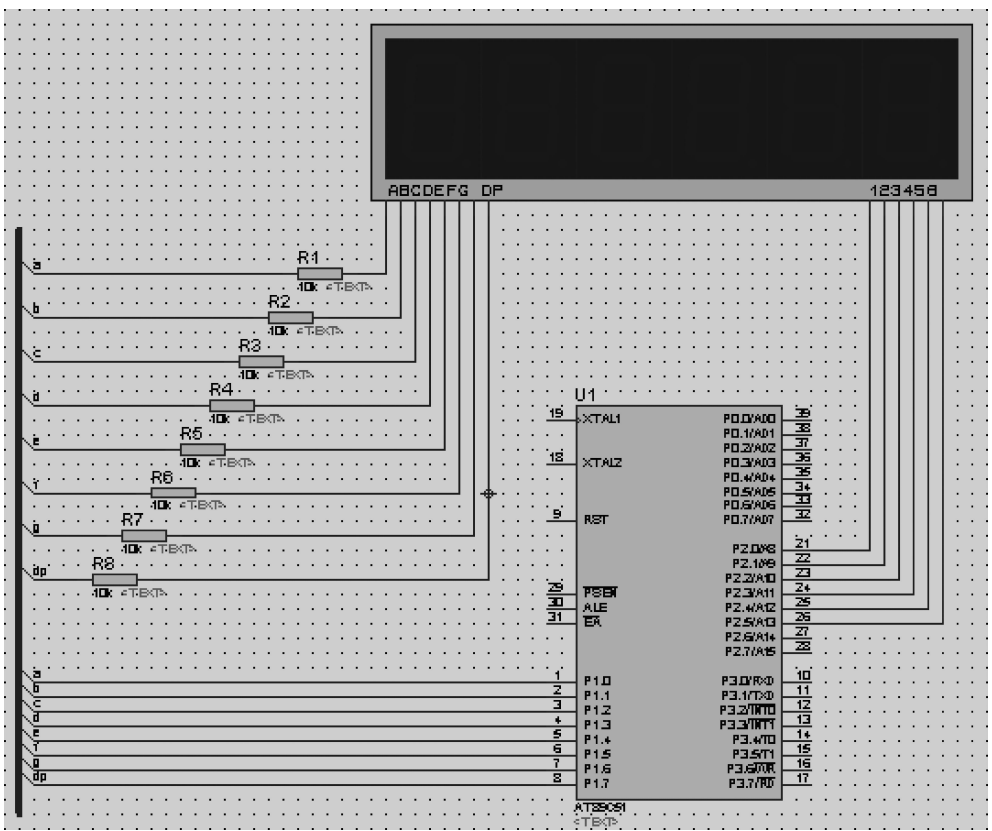


图 5.23 单片机电路图

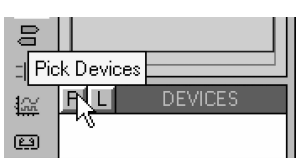


图 5.24 放置元器件

程序设计：

实现 LED 显示器的选通并显示字符。

电路图的绘制：

(1) 将所需元器件加入到对象选择器窗口。Picking Components into the Schematic 单击对象选择器按钮 P，如图 5.24 所示。

弹出 Pick Devices 页面，在 Keywords 输入 AT89C51，系统在对象库中进行搜索查找，并将搜索结果显示在 Results 中，如图 5.25 所示。

在 Results 栏中的列表项中，双击 AT89C51，则可将 AT89C51 添加至对象选择器窗口。

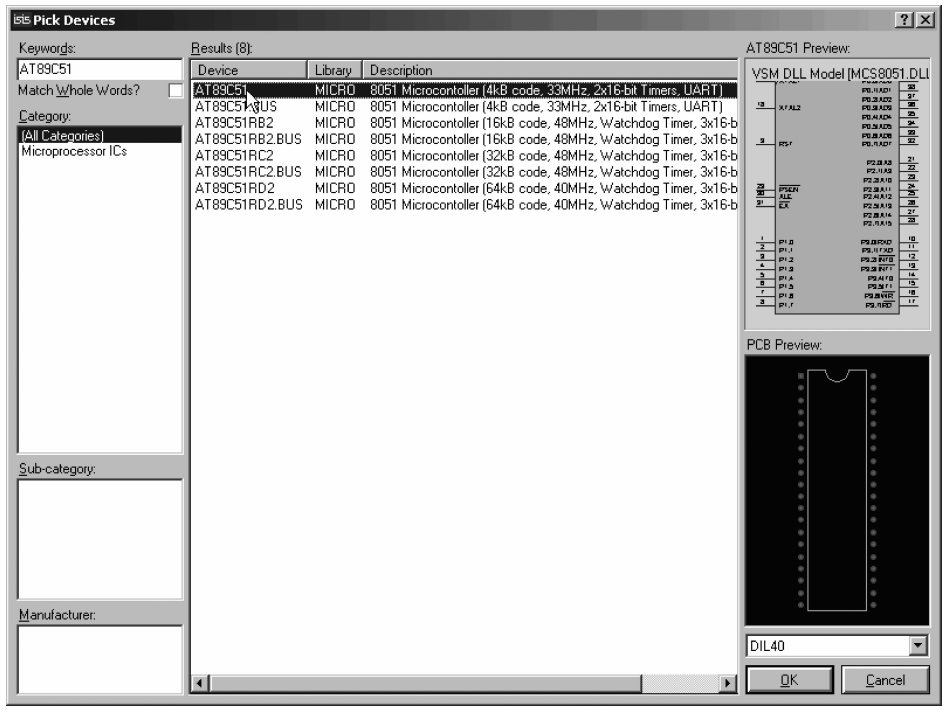


图 5.25　搜索元器件

接着在 Keywords 栏中重新输入 7SEG，如图 5.26 所示。双击 7SEG-MPX6-CA-BLUE，则可将 7SEG-MPX6-CA-BLUE(6 位共阳 7 段 LED 显示器)添加至对象选择器窗口。

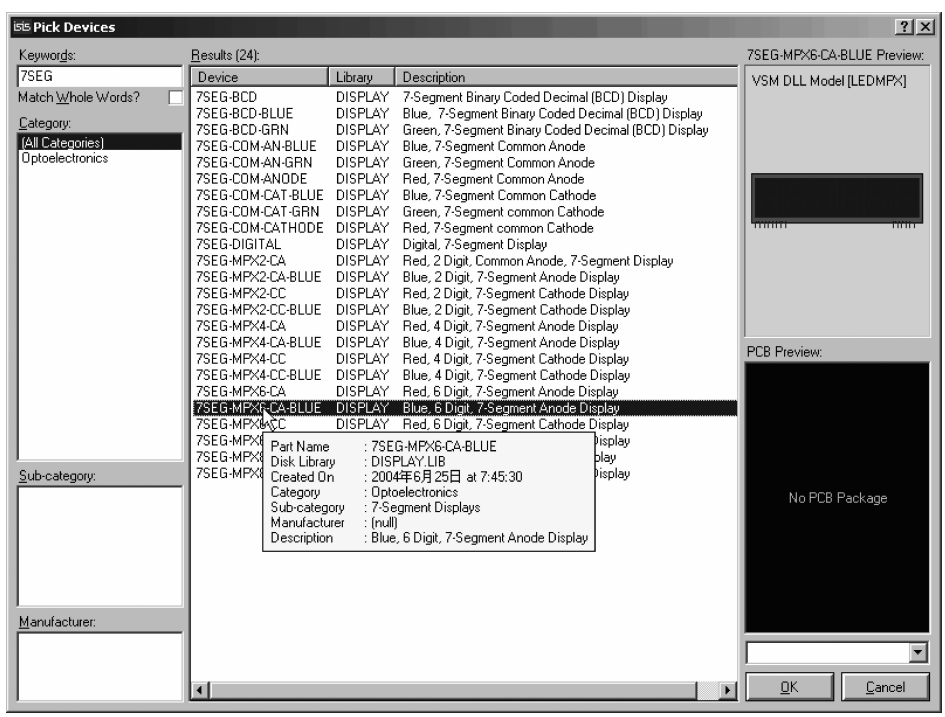


图 5.26　查找 7 段数码管器件

最后，在 Keywords 栏中重新输入 RES，选中 Match Whole Words，如图 5.27 所示。在 Results 栏中获得与 RES 完全匹配的搜索结果。双击 RES，则可将 RES(电阻)添加至对象选择器窗口。单击 OK 按钮，结束对象选择。

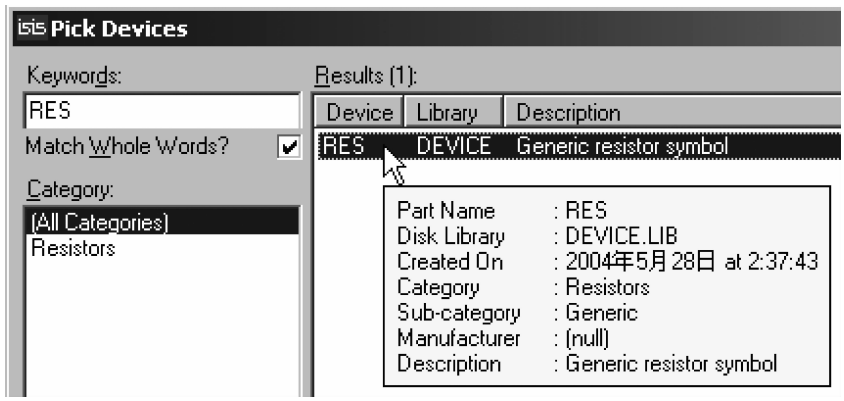
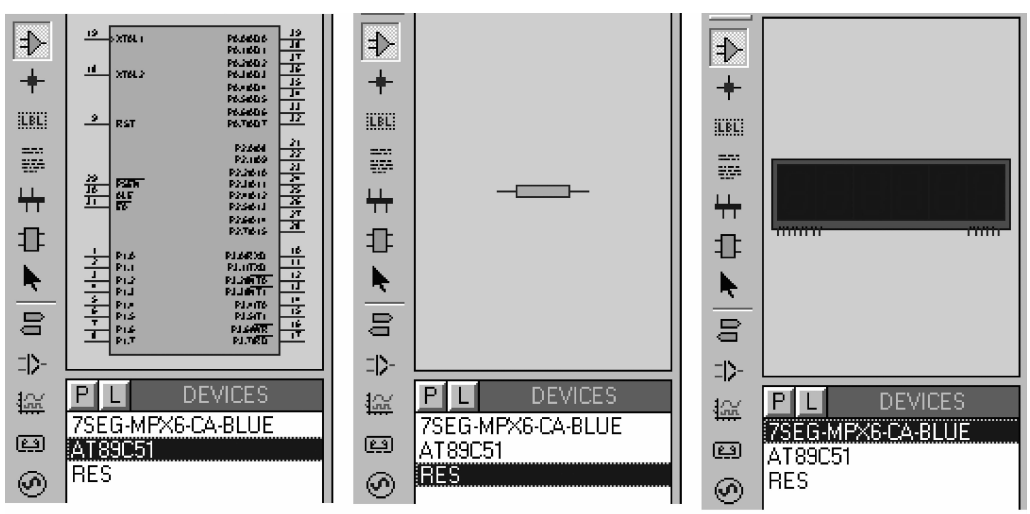


图 5.27 搜索电阻器件

经过以上操作，在对象选择器窗口中，已有了 7SEG-MPX6-CA-BLUE、AT89C51、RES 三个元器件对象，若单击 AT89C51，在预览窗口中，见到 AT89C51 的实物图，如图 5.28(a)所示。若单击 RES 或 7SEG-MPX6-CA-BLUE，在预览窗口中，见到 RES 和 7SEG-MPX6-CA-BLUE 的实物图，如图 5.28(b)、(c)所示。此时，我们已注意到在绘图工具栏中的元器件按钮 处于选中状态。



(a) (b) (c)

图 5.28 放置元器件

(2) 放置元器件至图形编辑窗口 Placing Components onto the Schematic。

在对象选择器窗口中，选中 7SEG-MPX6-CA-BLUE，将鼠标置于图形编辑窗口该对象的欲放位置，单击鼠标左键，该对象被完成放置。同理，将 AT89C51 和 RES 放置到图形编

辑窗口中，如图 5.29 所示。

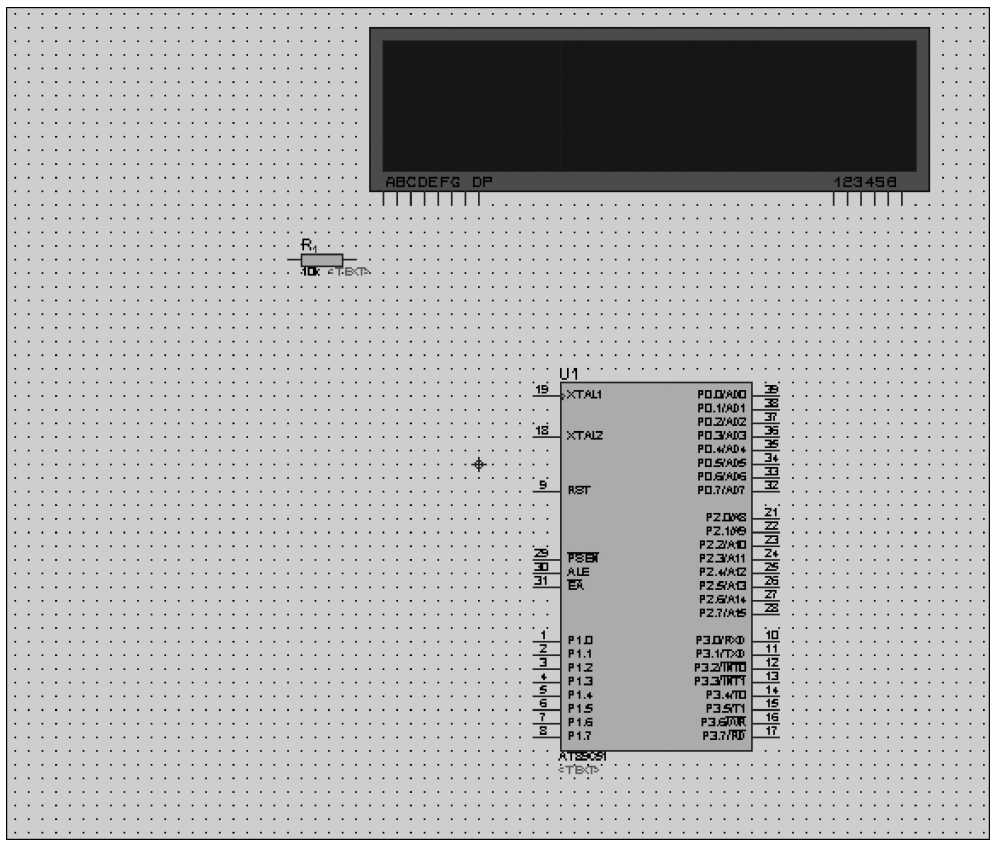


图 5.29　放好的元器件

若对象位置需要移动，将鼠标移到该对象上，单击鼠标右键。此时已经注意到，该对象的颜色已变至红色，表明该对象已被选中。按下鼠标左键，拖动鼠标，将对象移至新位置后，松开鼠标，完成移动操作。

由于电阻 R_1～R_8 的型号和电阻值均相同，因此可利用复制功能作图。将鼠标移到 R_1，单击鼠标右键，选中 R_1，在标准工具栏中，单击复制按钮，拖动鼠标，按下鼠标左键，将对象复制到新位置，如此反复，直到按下鼠标右键，结束复制。此时已经注意到，电阻名的标识，系统自动加以区分，如图 5.30 所示。

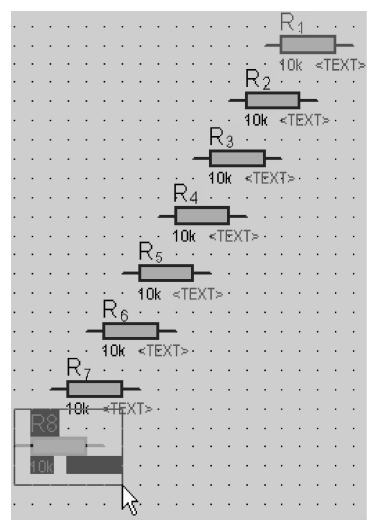


图 5.30　复制电阻元器件

(3) 放置总线至图形编辑窗口。

单击绘图工具栏中的总线按钮，使之处于选中状态。将鼠标置于图形编辑窗口，单击鼠标左键，确定总线的起始位置；移动鼠标，屏幕出现粉红色细直线，找到总线的终了位置，单击鼠标左键，再单击鼠标右键，以表示确认并结束画总线操作。此后，粉红色细直线被蓝色的

粗直线所替代，如图 5.31 放好的元器件所示。

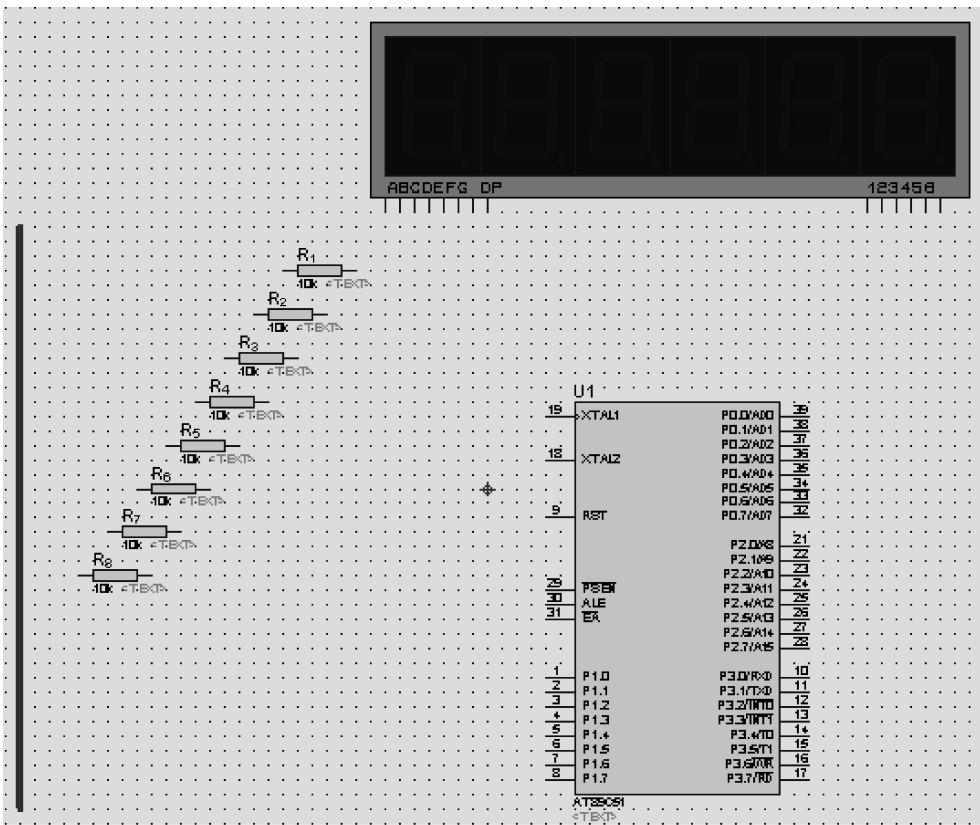


图 5.31 放好的元器件图

(4) 元器件之间的连线 Wiring Up Components on the Schematic。

Proteus 的智能化可以在想要画线的时候进行自动检测。下面，我们来操作将电阻 R_1 的右端连接到 LED 显示器的 A 端。当鼠标的指针靠近 R_1 右端的连接点时，跟着鼠标的指针就会出现一个“×”号，表明找到了 R_1 的连接点，单击鼠标左键，移动鼠标(不用拖动鼠标)，将鼠标的指针靠近 LED 显示器的 A 端的连接点时，跟着鼠标的指针就会出现一个“×”号，表明找到了 LED 显示器的连接点，同时屏幕上出现了粉红色的连接，单击鼠标左键，粉红色的连接线变成了深绿色，同时，线形由直线自动变成了 90°的折线，这是因为我们选中了线路自动路径功能。

Proteus 具有线路自动路径功能(WAR)，当选中两个连接点后，WAR 将选择一个合适的路径连线。WAR 可通过使用标准工具栏里的 WAR 命令按钮 来关闭或打开，也可以在菜单栏的 Tools 下找到这个图标。

同理，我们可以完成其他连线，画好的电路图如图 5.32 所示。在此过程的任何时刻，都可以按 Esc 键或者单击鼠标的右键来放弃画线。

(5) 元器件与总线的连线。

画总线的时候为了和一般的导线区分，我们一般喜欢画斜线来表示分支线。此时我们需要自己决定走线路径，只需在想要拐点处单击鼠标左键即可，如图 5.33 所示。

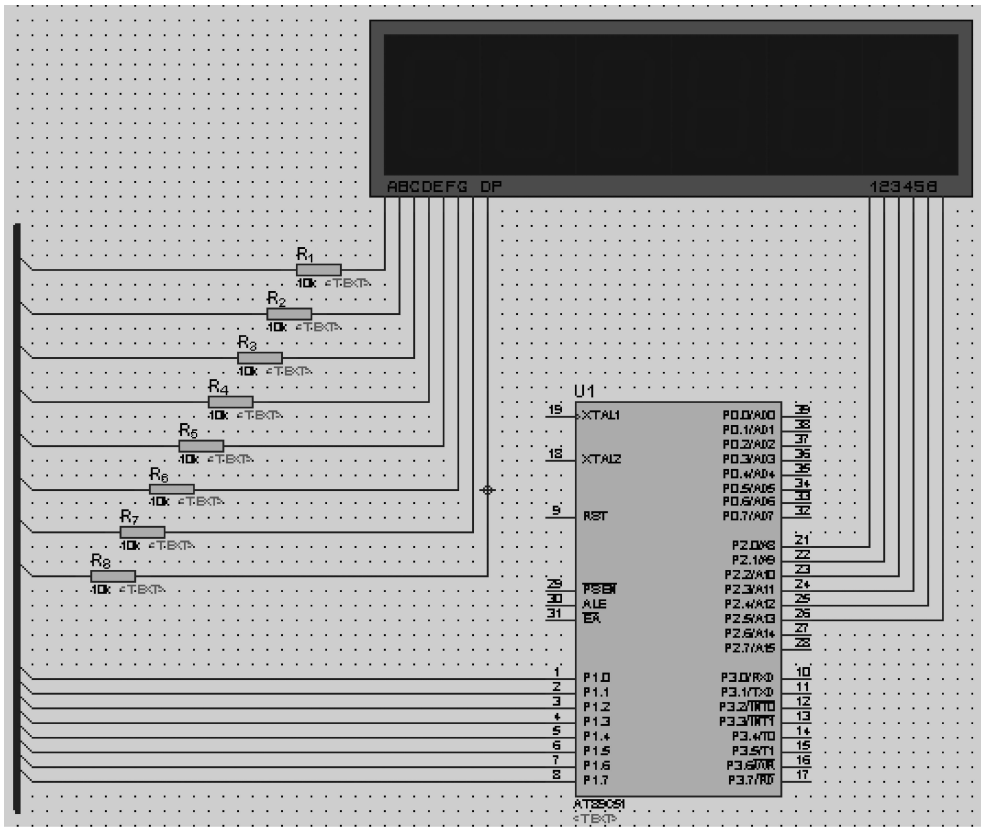


图 5.32　画好的电路图

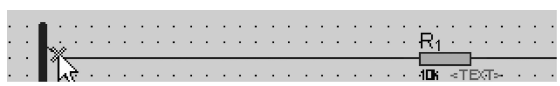


图 5.33　导线的连接

(6) 给与总线连接的导线贴标签 PART LABELS。

单击绘图工具栏中的导线标签按钮，使之处于选中状态。将鼠标置于图形编辑窗口的欲标标签的导线上，跟着鼠标的指针就会出现一个“×”号，表明找到了可以标注的导线，单击鼠标左键，弹出编辑导线标签窗口，如图 5.34 所示。

在 String 栏中，输入标签名称(如 a)，单击 OK 按钮，结束对该导线的标签标定。同理，可以标注其他导线的标签。注意，在标定导线标签的过程中，相互接通的导线必须标注相同的标签名。

至此，我们便完成了整个电路图的绘制。

KeilC 与 Proteus 连接调试如下。

(1) 假若 KeilC 与 Proteus 均已正确安装在 C:\Program Files 的目录里，把 C:\Program Files\Labcenter Electronics\Proteus 6 Professional\MODELS\VDM51.dll 复制到 C:\Program Files\keilC\C51\BIN 目录中。

(2) 用记事本打开 C:\Program Files\keilC\C51\TOOLS.INI 文件，在[C51]栏目下加入：

```
TDRV5 = BIN\\VDM51.DLL ("Proteus VSM Monitor - 51 Driver")
```

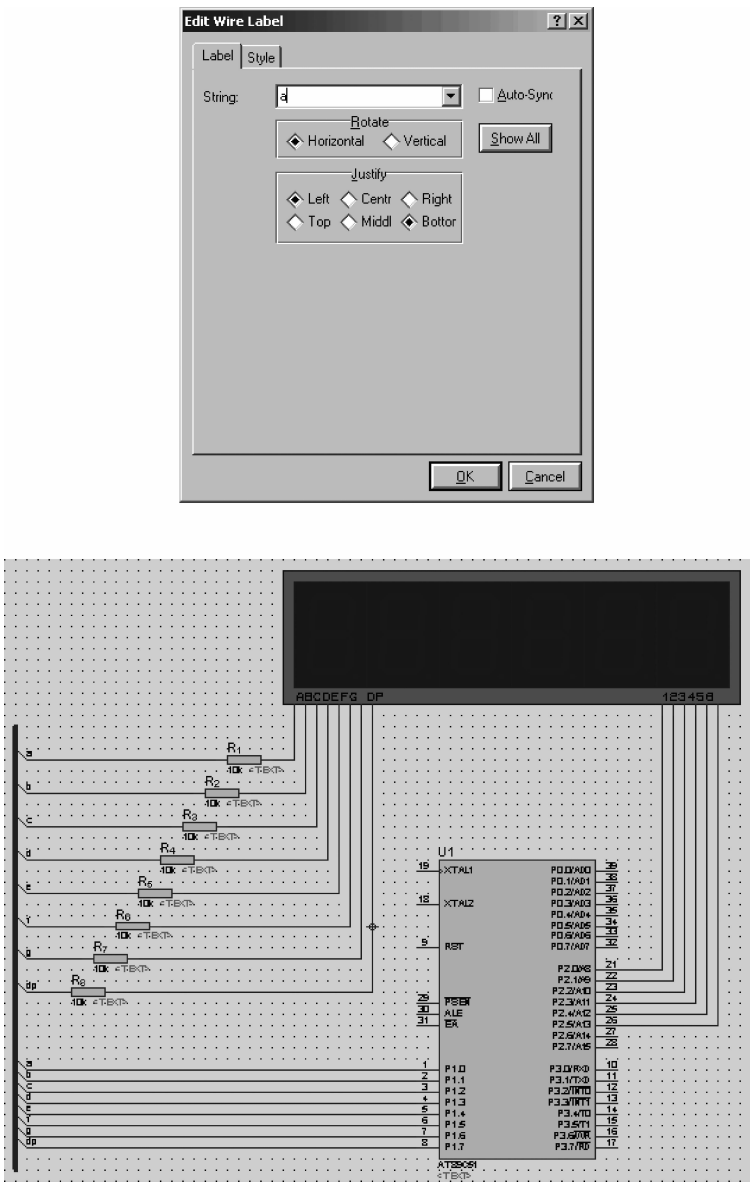


图 5.34 放置导线标签及电路图

其中“TDRV5”中的“5”要根据实际情况写，不要和原来的重复（步骤(1)和(2)只需在初次使用设置）。

(3) 进入 KeilC μVision2 开发集成环境，创建一个新项目(Project)，并为该项目选定合适的单片机 CPU 器件(如：Atmel 公司的 AT89C51)。并为该项目加入 KeilC 源程序。

源程序如下：

```
#define LEDS 6
#include "reg51.h"
//led 灯选通信号
unsigned char code Select[ ] = {0x01,0x02,0x04,0x08,0x10,0x20};
unsigned char code LED_CODES[ ] =
```

```
    {   0xc0,0xF9,0xA4,0xB0,0x99,          //0～4
        0x92,0x82,0xF8,0x80,0x90,          //5～9
        0x88,0x83,0xC6,0xA1,0x86,          //A,b,C,d,E
        0x8E,0xFF,0x0C,0x89,0x7F,0xBF      //F,空格,P,H,.,-   };
void main()
{
char i=0;
long int j;
while(1)
{
P2=0;
P1=LED_CODES[i];
P2=Select[i];
for(j=3000;j>0;j--);                  //该 LED 模型靠脉冲点亮,第 i 位靠脉冲点亮后,会自动熄灭
            //修改循环次数,改变点亮下一位之前的延时,可得到不同的显示效果
  i++;
  if(i>5) i=0;
 }
}
```

(4) 单击 Project|Options for Target 选项或者单击工具栏的 option for target 按钮，弹出窗口，单击 Debug 按钮，出现如图 5.35 所示页面。

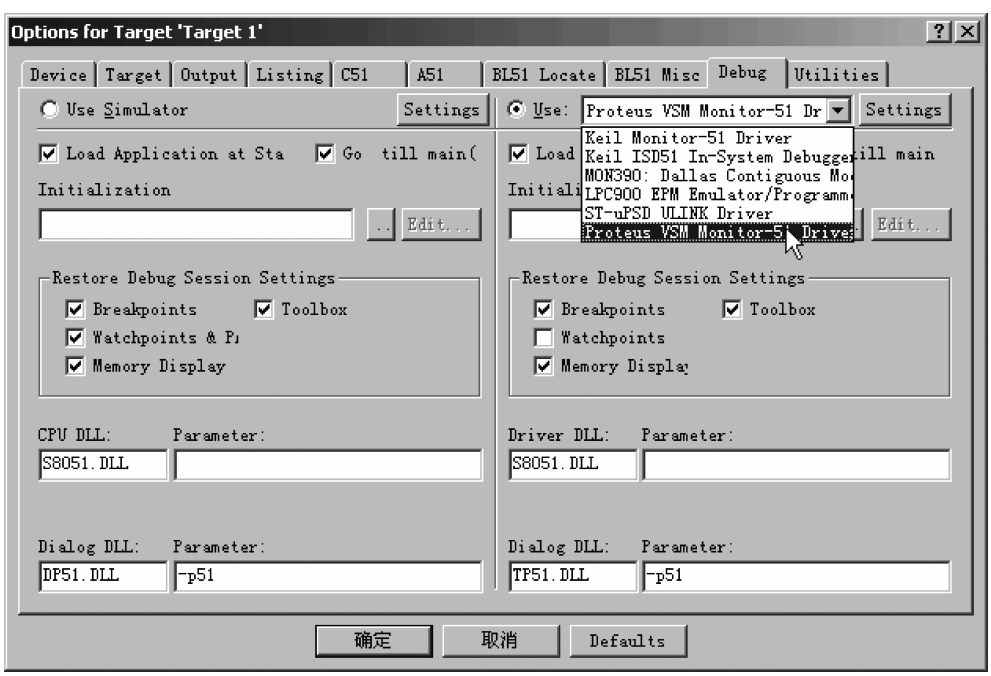


图 5.35　选择仿真连接软件

在出现的对话框里在右栏上部的下拉菜单里选中 Proteus VSM Monitor-51 Driver。并且还要单击一下 Use 前面表明选中的小圆点。

再单击 Setting 按钮，设置通信接口，在 Host 后面添上“127.0.0.1”，如果使用的不是同一台计算机，则需要在这里添上另一台计算机的 IP 地址（另一台计算机也应安装 Proteus）。在 Port 后面添加“8000”。设置好的情形如图 5.36 所示，单击 OK 按钮即可。最

后将工程编译，进入调试状态，并运行。

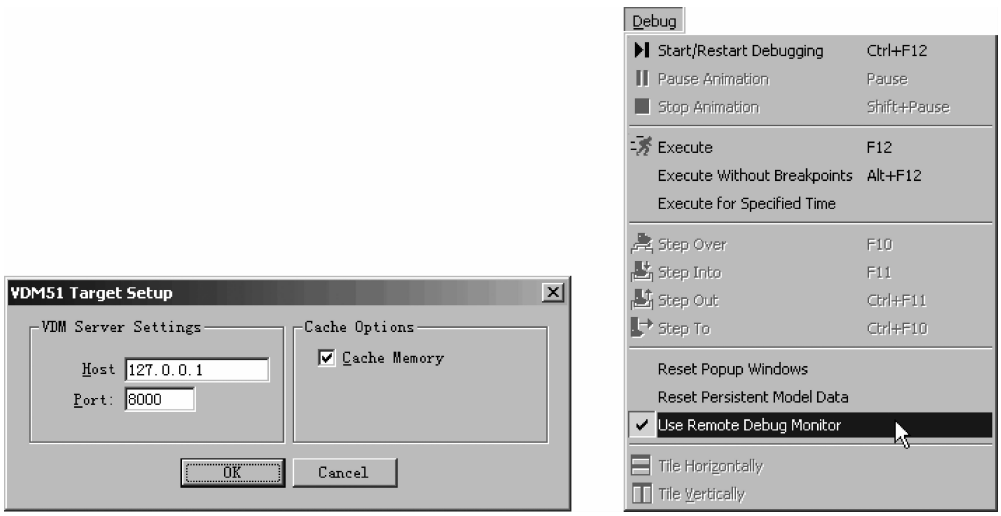


图 5.36 仿真连接设置图

(5) Proteus 的设置。

进入 Proteus 的 ISIS，鼠标左键单击菜单 Debug，选中 Use Remote Debug Monitor，如图 5.36 所示。此后，便可实现 KeilC 与 Proteus 连接调试。

(6) KeilC 与 Proteus 连接仿真调试。

单击仿真运行开始按钮 ▶，我们能清楚地观察到每一个引脚的电频变化，红色代表高电频，蓝色代表低电频。在 LED 显示器上，循环显示 0、1、2、3、4、5，如图 5.37 所示。

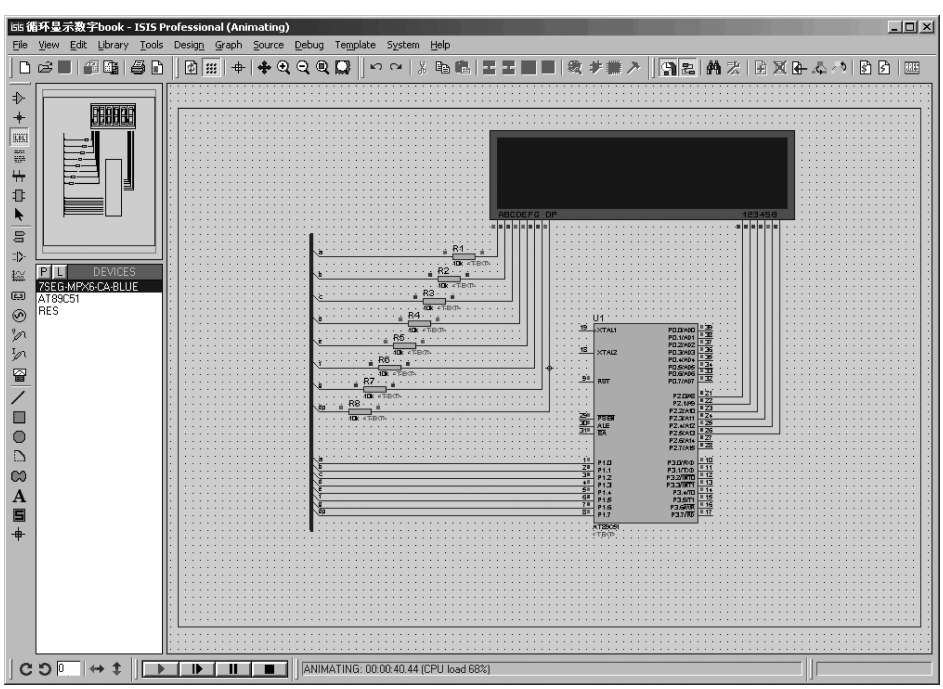


图 5.37 仿真电路图运行效果

本章小结

本章主要讨论了单片机的含义、分类、发展及其典型应用，并以 MCS-51 单片机为实例，详细说明了单片机系统的设计过程和开发关键；同时重点对嵌入式系统的概念、分类、组成、开发设计等部分做了详细的讨论。然后，又详细地介绍了当前常用的开发工具，对 Keil 与 Proteus 的用法和项目设计流程做了详细的说明，使读者能清晰地了解单片机及嵌入式系统的基本概念、基本架构、相互关系及应用开发，使读者对嵌入式系统的设计流程有一个清晰的认识，为进一步学习嵌入式系统打下坚实的基础。