1

发光二极管 LED 控制

终极目标

能完成单片机最小系统和输出电路设计，能应用 C 语言程序完成单片机输入输出控制，实现对 LED 控制的设计、运行及调试。

促成目标

1. 了解 AT89S52 单片机结构；
2. 掌握 AT89S52 单片机的引脚功能；
3. 掌握 AT89S52 单片机最小系统电路设计；
4. 掌握 C 语言基本构成和基本语句；
5. 会利用单片机 I/O 口实现点亮一个 LED 和控制 LED 闪烁。

1.1 工作模块 1　点亮一个 LED

使用 AT89S52 单片机，P1.0 引脚接发光二极管（LED）的阴极，通过 C 语言程序控制，

从 P1.0 引脚输出低电平，使发光二极管点亮。

1.1.1 用 Proteus 设计第一个 LED 控制电路

Proteus 是英国 Labcenter Electronics 公司开发的多功能 EDA 软件。PROTEUS 不仅是模拟电路、数字电路、模/数混合电路的设计与仿真平台，也是目前较先进的单片机和嵌入式系统的设计与仿真平台。它实现了在计算机上完成从原理图与电路设计、电路分析与仿真、单片机代码级调试与仿真、系统测试与功能验证到形成 PCB 的完整的电子设计、研发过程。

按照工作任务要求，点亮一个 LED 电路是由 AT89S52 单片机和 1 个 LED 电路构成。AT89S52 单片机是美国 ATMEL公司生产的低电压、高性能 8 位单片机，具有丰富的内部资源，使用 AT89S52 单片机无须外部存储器。

LED 加正向电压发光，反之不发光。一般接法是阳极接高电平，阴极接单片机的某一输出口线，当该输出口线为低时，LED 亮，该输出口线为高时，LED 不亮。这样我们只要编程控制单片机的该输出口，就可控制指示灯亮或灭。

在本工作模块中，LED 的阳极通过 220Ω限流电阻后联接到 5V 电源上，限流电阻在这里起到了限流的作用，使通过 LED 的电流被限制在十几个毫安左右。P1.0 引脚接 LED 的阴极，P1.0 引脚输出低电平时对应的 LED 点亮，输出高电平时对应的 LED 熄灭。LED 点亮电路设计如图 1-1 所示。

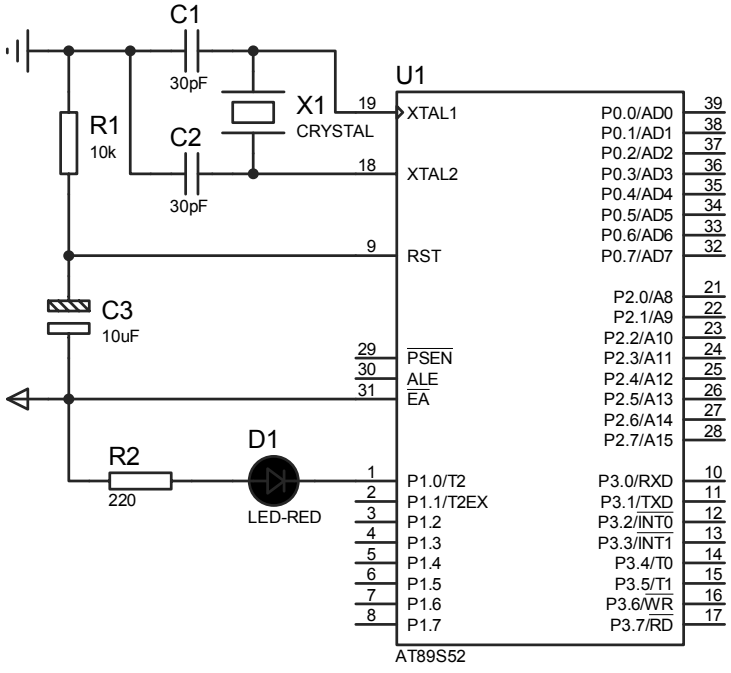


图 1-1　LED 点亮电路

本书使用 Proteus 7.5SP3 Professional 中文版。双击桌面上的 ISIS 7 Professional 图标或者单击屏幕左下方的“开始”→“程序”→Proteus 7 Professional→ISIS 7 Professional，进入 Proteus ISIS 集成环境，如图 1-2 所示。

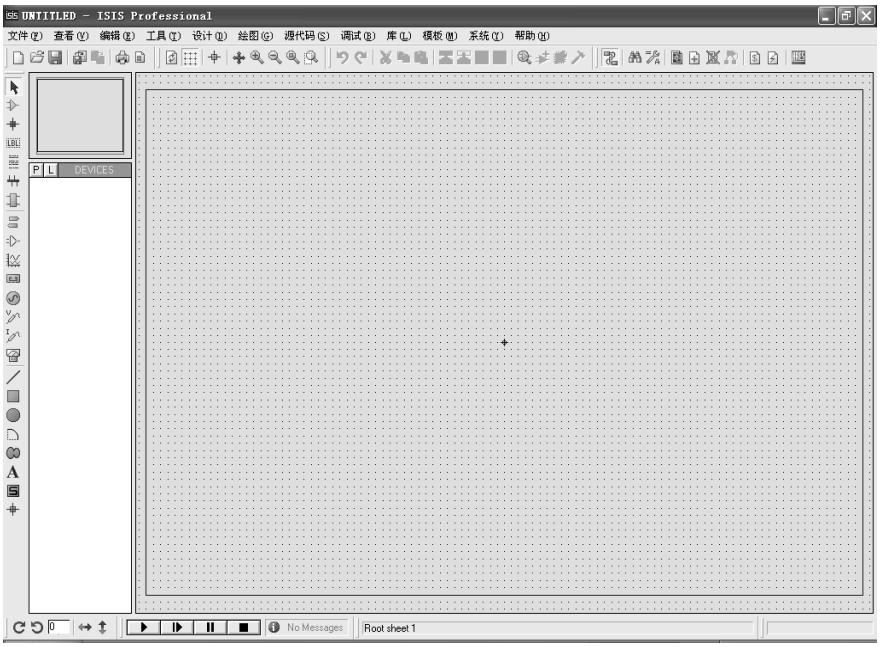


图 1-2　ISIS 集成环境

（1）新建设计文件。单击“文件”→“新建设计”命令，在弹出的“新建设计”对话框中选择 DEFAULT 模板后单击“确定”按钮，如图 1-3 所示。

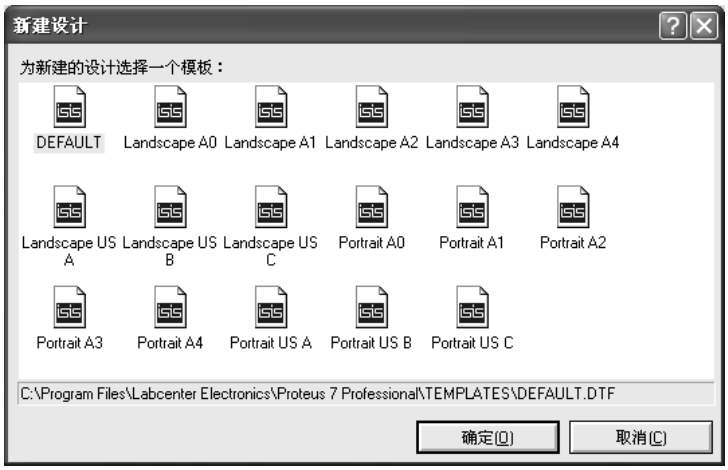


图 1-3　“新建设计”对话框

（2）设置图纸尺寸。单击“系统”→“设置图纸大小”命令，在弹出的 Sheet Size Configuration 对话框中选择 A4 图纸尺寸或自定义尺寸后单击“确定”按钮。

（3）设置网格。单击“查看”→“网格”命令，显示网格（再次单击，不显示网格）。单击“查看”→Snap xxth（或 Snap x.xin），可改变网格单位，默认为“Snap 0.1in”。

（4）保存设计文件。单击“文件”→“保存设计”，在弹出的“保存 ISIS 设计文件”对话框中指定文件夹、输入文件名“点亮一个 LED”并选择保存类型为“设计文件（*.DSN）”后单击“保存”。

（5）选取元器件。从 Proteus 元器件库中选取元器件 AT89C52（单片机），AT89S52 可用 AT89C52 代替。单击模式选择工具栏“元件”按钮，单击“器件选择”按钮，在弹出的“Pick Devices”（选取元器件）对话框的“关键字”栏中输入元器件名称 AT89C52（也可以是分类、小类、属性值），与关键字匹配的元器件“AT89C52”显示在元器件列表（结果）中。双击选中的元器件 AT89C52，便将所选元器件 AT89C52 加入到对象选择器窗口，单击“确定”完成元器件选取，如图 1-4 所示。

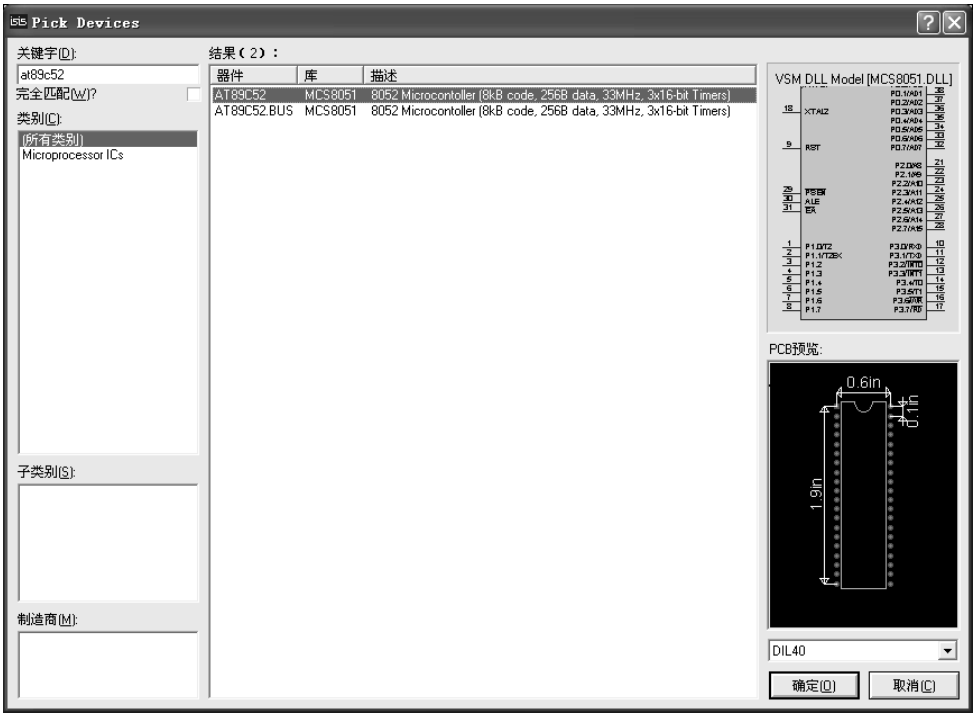


图 1-4 “Pick Devices”对话框

（6）用同样方法选取其他元器件。从 Proteus 元器件库中选取元器件 CRYSTAL（晶振）、CAP（电容）、CAP-ELEC（电解电容）、RES（电阻）、LED-RED（红色发光二极管）。

（7）放置元器件。单击对象选择器窗口的元器件 AT89C52，元器件名 AT89C52 变为蓝

底白字，预览窗口显示 AT89C52 元器件；单击方向工具栏按钮可实现元器件的左旋、右旋、水平和垂直翻转，以调整元器件的摆放方向；将鼠标指针移到编辑区某一位置，单击一次就可放置元器件 AT89C52。用同样方法放置其他元器件。按图 1-1 所示放置元器件。

（8）编辑元器件。单击模式选择工具栏“编辑”按钮，进入编辑状态。右击（或单击）元器件，该元器件变为红色表明被选中，鼠标指针放到被选中的元器件上，按住左键拖动，将鼠标移到编辑区某一位置松开，即完成元器件的移动。鼠标指针放到被选中的元器件上右击，单击弹出的快捷菜单中的方向工具栏按钮可实现元器件的旋转和翻转。右击被选中的元器件，可删除该元器件。被选中的元器件外单击，可撤销选中。按图 1-1 所示编辑元器件。

（9）放置终端。单击模式选择工具栏“终端”按钮，单击对象选择器窗口的电源终端“POWER”，该终端名背景变为蓝色，预览窗口显示该终端；单击方向工具栏“左旋转”按钮，电源终端逆时针旋转 90°；将鼠标指针移到编辑区某一位置，单击一次就可放置一个终端。用同样方法放置接地终端“GROUND”。

（10）连线。单击命令工具栏“实时 Snap（捕捉）”按钮，实时捕捉有效（再次单击，实时捕捉无效），当鼠标指针接近引脚末端时，该处会自动出现一个小方框“□”，表明可以自动连接到该点。按图 1-1 所示单击要连线的元器件起点和终点，完成连线。

（11）属性设置。先右击后单击元器件电容 C1，弹出的“编辑元件”对话框，如图 1-5 所示。将电容量改为 30pF，单击“确定”按钮完成元器件电容 C1 的属性编辑。用同样方法编辑其他元器件属性。

图 1-5　“编辑元件”对话框

（12）电气规则检测。单击“工具”→“电气规则检查”命令，弹出检查结果窗口，完成电气检测。若检测出错，根据提示修改电路图并保存，直至检测成功。电气规则检查窗口如图 1-6 所示。

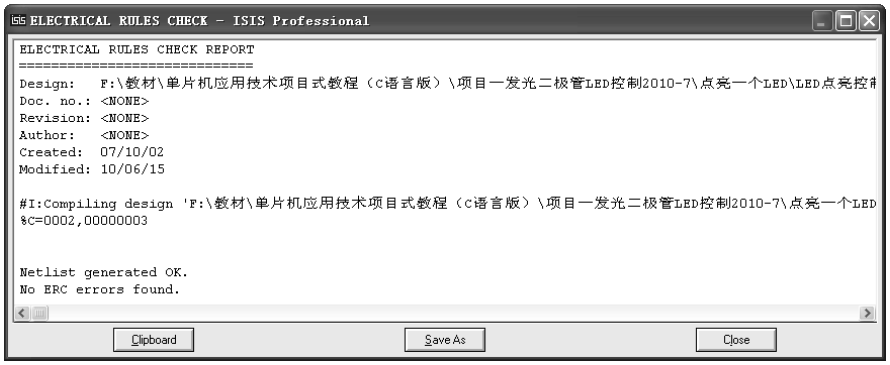


图 1-6　电气规则检查窗口

1.1.2　用 Keil C51 设计第一个 C 语言 LED 控制程序

Keil C51 是德国 Keil 软件公司开发的基于 8051 内核的微控制器软件开发平台，是 51 系列单片机 C 语言软件开发系统，是目前开发 8051 内核单片机的主流工具。Keil C51 软件提供丰富的库函数和功能强大的集成开发调试工具，全 Windows 界面。μVision2 集成开发环境可以完成从工程建立和管理、编译、连接、目标代码的生成、软件仿真和硬件仿真等完整的开发流程。

1. 编写点亮一个 LED 程序

由于 P1.0 引脚接 LED 的阴极，LED 的阳极通过 220Ω限流电阻后联接到 5V 电源上，所以从 P1.0 引脚输出低电平就可以点亮 LED。“点亮一个 LED”C 语言程序如下：

```
#include <AT89X52.H>          //包含 AT89X52.H 头文件
sbit LED=P1^0;                //定义 LED 为 P1.0 引脚
void main (void)
{
    LED=0;                    //P1.0 引脚输出低电平点亮 LED
    while(1);
}
```

程序编程说明：

（1）“#include <AT89X52.H>”语句是一个“文件包含”处理，是将 AT89X52.H 头文件的内容全部包含进来。这里程序中包含 AT89X52.H 头文件的目的是为了要使用 P1^0 这个符号，即通知 C 编译器，程序中所写的 P1^0 是指 AT89S52 单片机的 P1.0 引脚。

（2）P1.0 不能直接使用，这里用“sbit LED=P1^0;”就是定义用符号 LED 来表示 P1.0 引脚，你也可以起 P1_0 或 P10 一类的名字。

（3）“LED=0;”语句是使 P1.0 引脚输出低电平，点亮发光二极管 LED。

（4）“while(1);”语句的表达式是 1，也就是说 while 语句的表达式始终为真，进入死循环，LED 始终点亮。

（5）Keil C 支持 C++风格的注释，可以用“//”进行注释，也可以用/*……*/进行注释。

2. 建立第一个 C 程序项目

双击桌面上的 Keil μVision2 图标或者单击屏幕左下方的“开始”→“程序”→Keil μVision2，进入 Keil μVision2 集成开发环境，如图 1-7 所示。

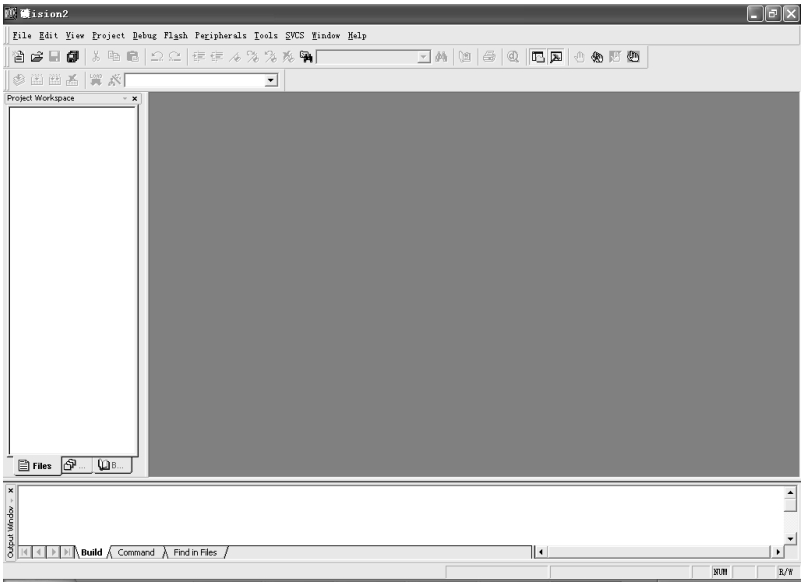


图 1-7　Keil μVision2 集成开发环境

（1）建立工程文件，选择单片机。单击“工程”→“新建工程”，在弹出的“新建工程”对话框中指定文件夹、输入文件名“点亮一个 LED”、单击“保存”，在弹出的“为目标 Target 1 选择设备”对话框中选择单片机型号（Atmel 的 89S52），单击“确定”按钮。选择单片机对话框如图 1-8 所示。

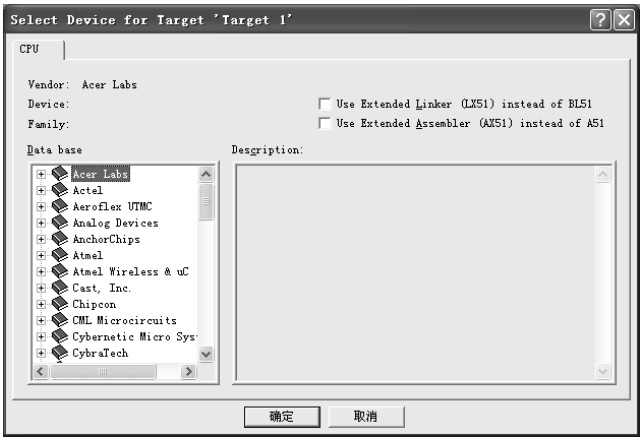


图 1-8　选择单片机对话框

（2）建立源文件，加载源文件。单击 “文件”→“新建”命令，在文件编辑窗口输入“点亮一个 LED”C 语言源程序，单击“文件”→“保存”命令，在弹出的“另存”对话框中指定文件夹（一般与工程文件放在同一文件夹中）、输入文件名“点亮一个 LED.c”（c 为 C 语言源程序的后缀）、单击“保存”按钮，完成源文件的建立。

在工程窗口中右击 Target 1 文件夹下的 Source Group 1 文件夹后，单击弹出菜单的 Add Files to Group‘Source Group 1’，在弹出的 Add Files to Group‘Source Group 1’对话框中将文件类型设为 C Source file (*.c)，单击刚才保存的源文件名“点亮一个 LED.c”，单击 Add 按钮再单击“关闭”按钮，完成源文件加载。源文件加载对话框如图 1-9 所示。

图 1-9　源文件加载窗口

（3）设置工程的配置参数。在工程窗口中右击 Target 1 文件夹，单击弹出快捷菜单中的 Options for Target‘Target 1’，参数设置对话框如图 1-10 所示。

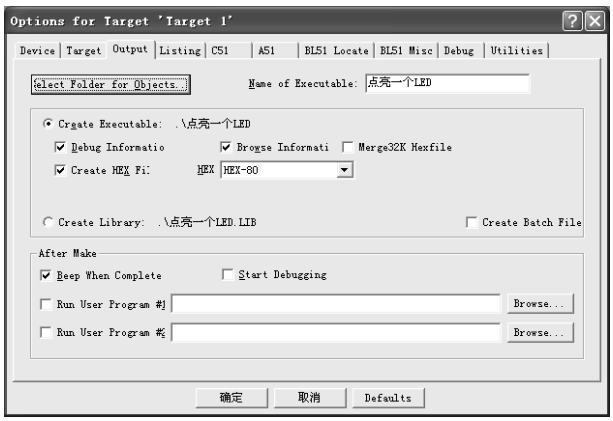


图 1-10　参数设置对话框

在弹出的 Options for Target‘Target 1’对话框中做以下设置：Target 标签页的晶振频率栏设为 12MHz，Output 标签页选中 Create Hex Files 复选框，其余采用默认设置。单击“确定”按钮，完成配置参数设置。

（4）进行编译和连接。单击“工程”→“构造目标”命令，完成编译，生成“点亮一个 LED.hex”的 HEX 文件。通过输出窗口查看编译信息，若提示出错，双击输出窗口出错信息行，文件编辑窗口出错指令所在行左侧会有箭头提示，逐个排除错误后重新编译。输出窗口如图 1-11 所示。

（5）进入调试模式，打开 P1 口对话框。在调试模式中，单击“外围设备”→I/O-Ports →Port 1，打开 P1 口对话框。

（6）全速运行程序。单击“调试”→“运行到”或调试工具栏的运行按钮，通过 P1 口对话框观察 P1.0 引脚是否输出低电平。调试窗口如图 1-12 所示，窗口中 P1 口的电平状态，打勾为高电平，不打勾则为低电平，Pins 为引脚的状态。

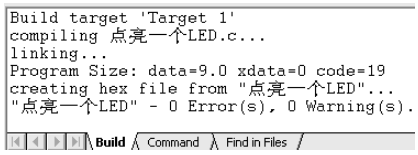


图 1-11　输出窗口

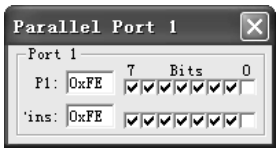


图 1-12　程序调试窗口

1.1.3　用 Proteus 仿真运行调试

1. 加载“点亮一个 LED.hex”目标代码文件

首先打开 Proteus“点亮一个 LED”电路。然后双击单片机 AT89S52，在弹出的“编辑元件”对话框中单击 Program File 栏的“打开”按钮，在弹出的“选择文件名”对话框找到前面编译生成的“点亮一个 LED.hex”文件，单击“打开”按钮，完成“点亮一个 LED.hex”文件加载。同时将 Clock Frequency 栏中的频率设为 12MHz，单击“确定”按钮，即可完成加载目标代码文件，如图 1-13 所示。

图 1-13　加载目标代码文件

2. 仿真运行调试

单击仿真工具栏“单步运行”按钮，进入单步运行状态。单击“调试”→8051 CPU Registers，单击“调试”→8051 CPU SFR Memory，分别打开工作寄存器窗口和特殊功能寄存器窗口。单击源代码调试窗口“单步执行”按钮一次，执行一条指令，通过各调试窗口观察每条指令执行后数据处理的结果，以加深对硬件结构和指令的理解。

“点亮一个 LED” Proteus 仿真运行如图 1-14 所示，在编辑区“点亮一个 LED”电路中，可以看到接在 P1.0 引脚上的 LED 被点亮，同时在打开的工作寄存器窗口和特殊功能寄存器窗口中也能看到 P1 口为 0xFE，即 P1.0 引脚为低电平，其他引脚都为高电平。

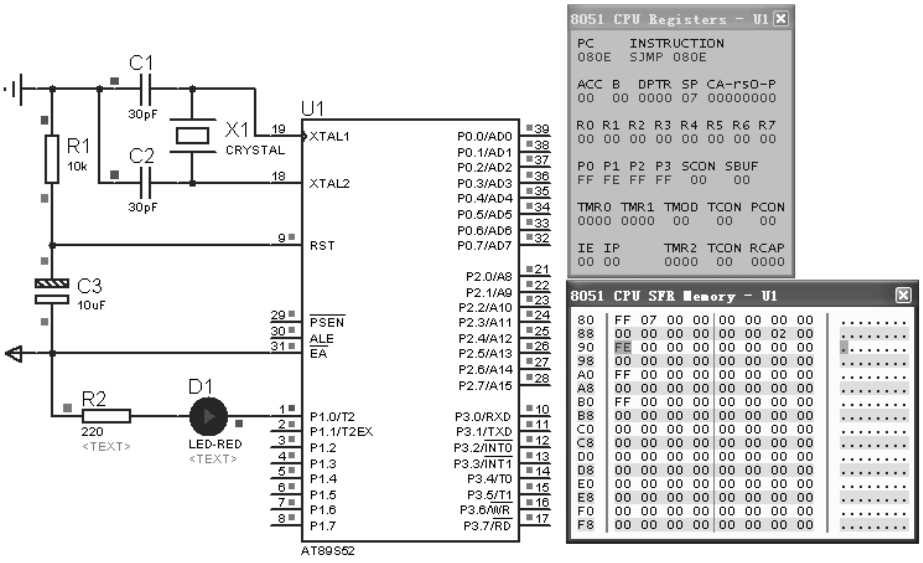


图 1-14 “点亮一个 LED”的 Proteus 仿真运行

单击仿真工具栏“运行”按钮，单片机全速运行程序。

1.2 认识单片机

随着微电子技术的不断发展，计算机技术也得到迅速发展，并且由于芯片的集成度的提高而使计算机微型化，出现了单片微型计算机（Single Chip Computer），简称单片机，它是微型计算机发展中的一个重要分支。

1.2.1 单片机概述

单片机又称为微控制器 MCU，它不是完成某一个逻辑功能的芯片，而是把一个计算机系统集成到一个芯片上，即一块芯片就构成了一台计算机。单片机集成了中央处理器 CPU、数

据存储器 RAM、程序存储器 ROM、定时器/计数器以及输入/输出接口电路等主要计算机部件。

1. 单片机的发展

单片机自问世以来，性能不断提高和完善，能满足很多应用场合的需要。特别是当前用 CMOS 工艺制成的各种单片机，由于功耗低，使用的温度范围大、抗干扰能力强、能满足一些特殊要求的应用场合，更加扩大了单片机的应用范围，也进一步促进了单片机技术的发展。单片机的发展主要经历了 4 个阶段。

第一阶段（1974～1976 年）为单片机初级阶段。由于受工艺及集成度的限制，单片机采用双片形式，且功能比较简单。如美国 Fairchild 公司 1974 年推出的单片机 F8，它包含 8 位 CPU、64B。F8 还需要外接一片 3851（内含 1KBROM、1 个定时器/计数器和 2 个 I/O 口）电路才能构成一个完整的微型计算机。

第二阶段（1976～1978 年）为低性能单片机阶段。单片机采用单芯片形式，是“小而全”。如美国 Intel 公司 1976 年推出的 MCS-48 系列单片机，8 位 CPU，并行 I/O 口，8 位定时器/计数器，无串行口，中断处理比较简单，RAM、ROM 容量较小，寻址范围不超过 4KB。它把单片机推向市场，促进了单片机的变革，各种 8 位单片机纷纷应运而生。

第三阶段（1978～1982 年）为高性能单片机阶段，也是单片机普及阶段。此时的单片机品种多，功能强，8 位 CPU，片内 RAM、ROM 容量加大，片外寻址范围可达 64KB，增加了串行口，多级中断处理系统，16 位定时器/计数器。如美国 Intel 公司在 MCS-48 基础上推出的高性能 MCS-51 系列单片机。

第四阶段（1982 年以后）为 16 位单片机阶段。是 16 位 CPU，片内 RAM、ROM 容量进一步增大，增加了 AD/DA 转换器，8 级中断处理功能，实时处理能力更强，它允许用户采用面向工业控制的专用语言，如 C 语言等。如美国 Intel 公司的 MCS-96 系列单片机。

总之，单片机发展可归结为以下几个方面：

（1）增加字长，提高数据精度和处理的速度；

（2）改进制作工艺，提高单片机的整体性能；

（3）由复杂指令集 CISC 转向简单指令集 RISC 技术；

（4）多功能模块集成技术，使一块“嵌入式”芯片具有多种功能；

（5）微处理器与 DSP 技术相结合；

（6）融入高级语言的编译程序；

（7）低电压、宽电压、低功耗。

目前，国际市场上 8 位、16 位单片机系列已有很多，32 位的单片机也已经进入了实用阶段。随着单片机技术的不断发展，新型单片机还将不断涌现，单片机技术正以惊人的速度向前发展着。

2. 单片机的特点

单片机作为微型计算机的一个分支，与一般的微型计算机没有本质上的区别，同样具有快速、精确、记忆功能和逻辑判断能力等特点。但单片机是集成在一块芯片上的微型计算机，

它与一般的微型计算机相比，在硬件结构和指令设置上均有独到之处，主要特点有：

（1）体积小、重量轻；价格低、功能强；电源单一、功耗低；可靠性高、抗干扰能力强，这是单片机得到迅速普及和发展的主要原因。同时由于它的功耗低，使后期投入成本也大大降低。

（2）使用方便灵活、通用性强。由于单片机本身就构成一个最小系统，只要根据不同的控制对象作相应的改变即可，因而它具有很强的通用性。

（3）目前大多数单片机采用哈佛（Harvard）结构体系。单片机的数据存储器空间和程序存储器空间相互独立。单片机主要面向测控对象，通常有大量的控制程序和较少的随机数据，将程序和数据分开，使用较大容量的程序存储器来固化程序代码，使用少量的数据存储器来存取随机数据。程序在只读存储器 ROM 中运行，不易受外界侵害，可靠性高。

（4）突出控制功能的指令系统。单片机的指令系统中有大量的单字节指令，以提高指令运行速度和操作效率；有丰富的位操作指令，满足了对开关量控制的要求；有丰富的转移指令，包括有无条件转移指令和条件转移指令。

（5）较低的处理速度和较小的存储容量。因为单片机是一种小而全的微型机系统，它是牺牲运算速度和存储容量来换取其体积小、功耗低等特色。

3. 单片机的应用

由于单片机是在一块芯片上集成了一台微型计算机所需的 CPU、存储器、输入/输出部件和时钟电路等。因此它具有体积小、使用灵活、成本低、易于产品化、抗干扰能力强、可在各种恶劣环境下可靠地工作等特点。特别是它应用面广，控制能力强，使它在工业控制、智能仪表、外设控制、家用电器、机器人、军事装置等方面得到了广泛的应用。单片机主要应用在以下几个方面：

（1）家用电器。单片机广泛应用在家用电器的自动控制中。如：洗衣机、空调机、电冰箱、电视机、音响设备等。单片机的使用提高了家用电器的性能和质量，降低家用电器的生产成本和销售价格。

（2）智能卡。尽管目前使用的各种卡主要是磁卡和 IC 卡，但是，带有 CPU 和存储器的智能卡，已经并将日益广泛用于金融卡、通信、信息、医疗保健、社会保险、教育、旅游、娱乐和交通等各个领域。

（3）智能仪器仪表。单片机体积小、耗电少 ，被广泛用于各类仪器仪表。如：智能电度表、智能流量计、气体分析仪、智能电压电流测试仪和智能医疗仪器等。单片机使仪器仪表走向了智能化和微型化，使仪器仪表的功能和可靠性大大提高。

（4）网络与通信。许多型号的单片机都有通信接口可方便地进行机间通信，也可方便地组成网络系统。如：单片机控制的无线遥控系统，列车无线通信系统和串行自动呼叫应答系统等。

（5）工业控制。单片机可以构成各种工业测控系统、数据采集系统。如：数控机床、汽车安全技术检测系统，报警系统和生产过程自动控制等。

4. 51 系列单片机的分类

单片机可分为通用型单片机和专用型单片机两大类。通用型单片机是把可开发资源全部

提供给使用者的微控制器。专用型单片机则是为过程控制、参数检测、信号处理等方面的特殊需要而设计的单片机。我们通常所说的单片机即指通用型单片机。

51 系列单片机源于 Intel 公司的 MCS-51 系列，在 Intel 公司将 MCS-51 系列单片机实行技术开放政策之后，许多公司都以 MCS-51 中的基础结构 8051 为基核推出了许多各具特色、具有优异性能的单片机，如 Philips、Dallas、Siemens、Atmel、华邦、LG 等。这样，把这些厂家以 8051 为基核推出的各种型号的兼容型单片机统称为 51 系列单片机。Intel 公司 MCS-51 系列单片机中的 8051 是其中最基础的单片机型号。

尽管各类单片机很多，但目前在我国使用最为广泛的单片机系列是 Intel 公司生产的 MCS-51 系列单片机，同时该系列还在不断地完善和发展。随着各种新型号系列产品的推出，它越来越被广大用户所接受。

（1）按片内不同程序存储器的配置来分。

1）片内带 MaskROM（掩膜 ROM）型：8051、80C51、8052、80C52。此类芯片是由半导体厂家在芯片生产过程中，将用户的应用程序代码通过掩膜工艺制作到 ROM 中。其应用程序只能委托半导体厂家“写入”，一旦写入后不能修改。此类单片机，适合大批量使用。

2）片内带 EPROM 型：8751、87C51、8752。此类芯片带有透明窗口，可通过紫外线擦除存储器中的程序代码，应用程序可通过专门的编程器写入到单片机中，需要更改时可擦除重新写入。此类单片机，价格较贵，不宜于大批量使用。

3）片内无 ROM（ROMLess）型：8031、80C31、8032。此类芯片的片内没有程序存储器，使用时必须在外部并行扩展程序存储器存储芯片。此类单片机由于必须在外部并行扩展程序存储器存储芯片，造成系统电路复杂，目前使用较少。

（2）按片内不同容量的存储器配置来分。

1）51 子系列型：芯片型号的最后位数字以 1 作为标志，51 子系列是基本型产品。片内带有 4KBROM/EPROM（8031、80C31 除外）、128BRAM、2 个 16 位定时器/计数器、5 个中断源等。

2）52 子系列型：芯片型号的最后位数字以 2 作为标志，52 子系列则是增强型产品。片内带有 8KBROM/EPROM（8032、80C32 除外）、256BRAM、3 个 16 位定时器/计数器、6 个中断源等。

（3）按芯片的半导体制造工艺的不同来分。

1）HMOS 工艺型：8051、8751、8052、8032。HMOS 工艺，即高密度短沟道 MOS 工艺。

2）CHMOS 工艺型：80C51、83C51、87C51、80C31、80C32，80C52。此类芯片型号中都字母“C”来标识。

这两类器件在功能上是完全兼容的，但采用 CHMOS 工艺的芯片具有低功耗的特点，它所消耗的电流要比 HMOS 器件小得多。CHMOS 器件比 HMOS 器件多了两种节电的工作方式（掉电方式和待机方式），常用于构成低功耗的应用系统。

此外，关于单片机的温度特性，与其他芯片一样按所能适应的环境温度范围，可划分为

三个等级：0℃～70℃民用级、－40℃～+85℃工业级和－65℃～+125℃军用级。因此在使用时应注意根据现场温度选择芯片。

5. AT89 系列单片机

在 MCS-51 系列单片机 8051 的基础上，Atmel 公司开发的 AT89 系列单片机问世以来，以其较低廉的价格和独特的程序存储器——快闪存储器（Flash Memory）为用户所青睐。表 1-1 列出了 AT89 系列单片机的几种主要型号。

表 1-1　AT89 系列单片机一览表

型号	快闪 ROM	片内 RAM	寻址范围	并行口线	串行口	中断源	定时器
AT89C51	4K	128	2×64K	32	1	5	2×16
AT89C52	8K	256	2×64K	32	1	6	3×16
AT89LV51	4K	128	2×64K	32	1	5	2×16
AT89LV52	8K	256	2×64K	32	1	6	3×16
AT89C2051	2K	128	2×4K	15	1	5	2×16
AT89C4051	4K	128	2×4K	15	1	5	2×16
AT89S51	4K	128	2×64K	32	1	5	2×16
AT89S52	8K	256	2×64K	32	1	6	3×16
AT89S53	12K	256	2×64K	32	1	7	3×16

采用快闪存储器（Flash Memory）的 AT89 系列单片机，不但具有 MCS-51 系列单片机的基本特性（如指令系统兼容，芯片引脚分布相同等），而且还具有一些独特的优点：

（1）片内程序存储器为电擦写型 ROM（可重复编程的快闪存储器）。整体擦除时间仅为 10ms 左右，可写入/擦除 1000 次以上，数据保存 10 年以上。

（2）两种可选编程模式，即可以用 12V 电压编程，也可以用 VCC 电压编程。

（3）宽工作电压范围，VCC=2.7～6V。

（4）全静态工作，工作频率范围：0Hz～24MHz，频率范围宽，便于系统功耗控制。

（5）三层可编程的程序存储器上锁加密，使程序和系统更加难以仿制。

总之，AT89 系列单片机与 MCS-51 系列单片机相比，前者和后者有兼容性，但前者的性能价格比等指标更为优越。本教程主要围绕 AT89S52 单片机进行介绍。

1.2.2　AT89S52 单片机结构

AT89S52 单片机是一个低功耗、高性能 CMOS 8 位单片机。AT89S52 采用 ATMEL 公司的高密度、非易失性存储技术制造，兼容标准 MCS-51 指令系统及 80C51 引脚结构。AT89S52 是一个功能强大的微控制器，具有较高的性价比，可为许多嵌入式控制应用系统提供高性价比的解决方案。

此外，AT89S52 支持两种软件可选的省电模式：空闲模式和掉电模式。AT89S52 设计和配置了振荡频率可为 0Hz 并可通过软件设置省电模式。空闲模式下，CPU 暂停工作，而 RAM 定时计数器、串行口、外中断系统可继续工作，掉电模式冻结振荡器而保存 RAM 的数据，停止芯片其他功能直至外中断激活或硬件复位。AT89S52 单片机包含以下部件：

（1）一个 8 位 CPU；

（2）一个片内振荡器及时钟电路；

（3）8K 字节可重复擦写的 Flash 闪速存储器；

（4）三级加密程序存储器；

（5）256×8 字节内部 RAM；

（6）3 个 16 位定时器/计数器；

（7）32 条可编程的 I/O 线（四个 8 位并行 I/O 端口）；

（8）一个可编程全双工串行口；

（9）具有 6 个中断源、两个优先级嵌套中断结构。

AT89S52 基本结构如图 1-15 所示。

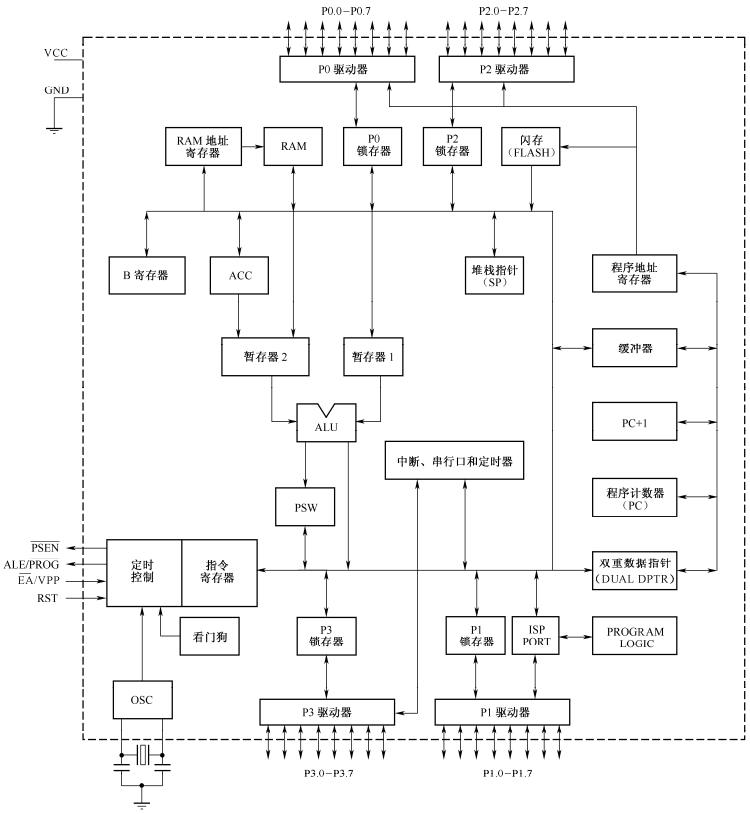
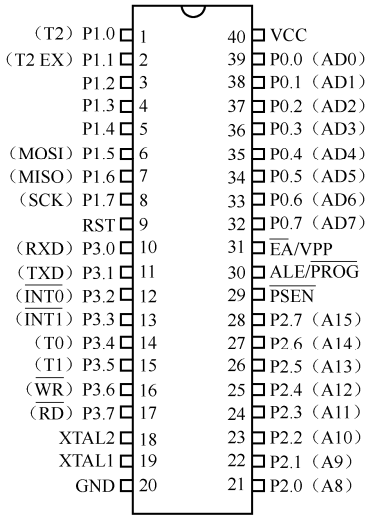


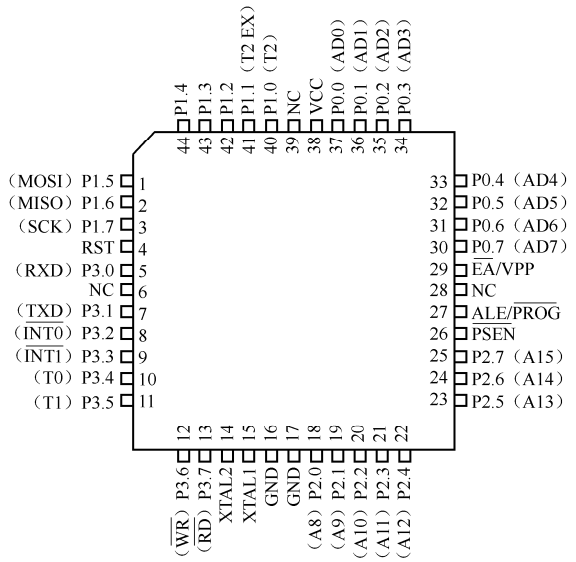
图 1-15　AT89S52 基本结构

1.2.3 AT89S52 单片机引脚功能

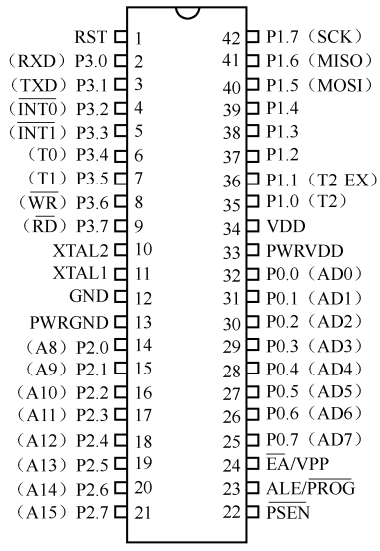
AT89S52 的引脚和封装共有四种，如图 1-16 所示。



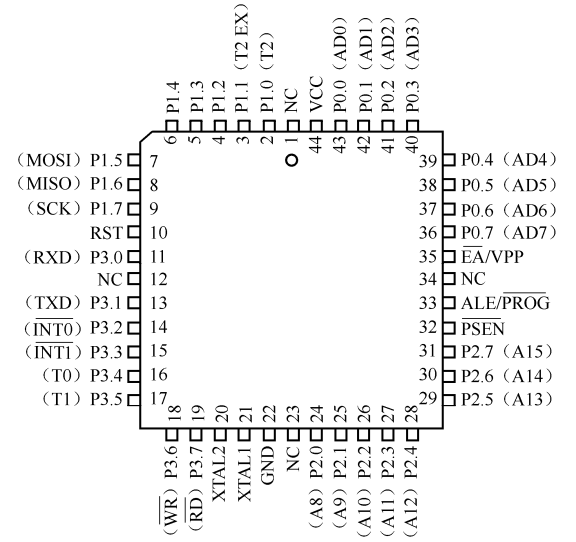
（a）40 引脚塑料双列直插式封装



（b）44 引脚薄型四方扁平封装



（c）42 引脚塑料双列直插式封装



（d）44 引脚塑料无引线芯片载体封装

图 1-16　AT89S52 的引脚和封装

下面以 40 引脚塑料双列直插式封装（PDIP）芯片（见图 1-16（a））为例，介绍各个引脚功能。

1. 电源引脚

（1）GND（20）：接地端。

（2）VCC（40）：正常操作时为+5V 电源。

通常在 VCC 和 GND 引脚之间接 0.1μ高频滤波电容。

2. 外接晶振引脚

（1）XTAL1（19）：内部振荡电路反相放大器的输入端，是外接晶体的一个引脚。当采用外部振荡器时，此引脚接地。

（2）XTAL2（18）：内部振荡电路反相放大器的输出端。是外接晶体的另一端。当采用外部振荡器时，此引脚接外部振荡源。

3. 控制或与其他电源复用引脚

（1）ALE/$\overline{\text{PROG}}$（30）：地址锁存允许/编程脉冲输入。在访问外部程序存储器和外部数据存储器时，该引脚输出一个地址锁存脉冲 ALE，其下降沿可降低 8 位地址锁存于片外地址锁存器中。在编程时，向该引脚输入一个编程负脉冲$\overline{\text{PROG}}$。正常操作时为 ALE 功能（允许地址锁存）提供把地址的低字节锁存到外部锁存器，ALE 引脚以不变的频率（振荡器频率的 1/6）周期性地发出正脉冲信号。

（2）$\overline{\text{PSEN}}$（29）：外部程序存储器读选通信号输出端，低电平有效。在从外部程序存储取指令（或数据）期间，$\overline{\text{PSEN}}$ 在每个机器周期内两次有效。在访问外部数据存储器时，$\overline{\text{PSEN}}$ 无效。

（3）$\overline{\text{EA}}$/VPP（31）：内部程序存储器和外部程序存储器选择端。当$\overline{\text{EA}}$/VPP 为高电平时，访问内部程序存储器，当超过内部程序存储器地址范围后，自动转向外部程序存储器；当$\overline{\text{EA}}$/VPP 为低电平时，则访问外部程序存储器。在 Flash 编程时，该引脚可连接 21V 的编程电源 VPP。

4. 输入/输出引脚

（1）P0.0～P0.7（32～39）：P0 口是一个 8 位漏极开路型双向 I/O 口。当用作通用 I/O 口时，每个引脚可驱动 8 个 TTL 负载；当用作输入时，每个端口首先置 1。在访问外部存储器时，它是分时传送的低字节地址和数据总线，此时，P0 口内含上拉电阻。

（2）P1.0～P1.7（1～8）：P1 口是一个带有内部提升电阻的 8 位准双向 I/O 口。当用作通用 I/O 口时，每个引脚可驱动 8 个 TTL 负载。当用作输入时，每个端口首先置 1。P1.1 和 P1.2 引脚也可用作定时器 2 的外部计数输入（P1.0/T2）和触发器输入（P1.1/T2EX）。

（3）P2.0～P2.7（21～28）：P2 口是一个带有内部提升电阻的 8 位准双向 I/O 口，在访问外部存储器时，它输出高 8 位地址。P2 口可以驱动 4 个 TTL 负载。当用作输入时，每个端口首先置 1。

（4）P3.0～P3.7（10～17）：P3 口是一个带有内部提升电阻的 8 位准双向 I/O 口，能驱动 4 个 TTL 负载。当用作输入时，每个端口首先置 1。P3 口还用于第二功能，请看项目 2 表 2-1。

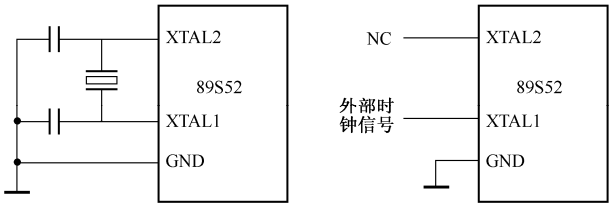
1.2.4 AT89S52 单片机最小系统

所谓最小系统就是指由单片机和一些基本的外围电路所组成的一个可以工作的单片机系统。一般来说，它包括单片机、晶振电路和复位电路。

1. 晶振电路

AT89S52 片内有一个由高增益反相放大器构成的振荡电路。XTAL1 和 XTAL2 分别为振荡电路的输入输出端。其振荡电路有两种组成方式：片内振荡器和片外振荡器。片内振荡器如图 1-17（a）所示。在 XTAL1 和 XTAL2 引脚两端跨接石英晶体振荡器和两个微调电容构成振荡电路，通常 C1 和 C2 一般取 30pF，晶振的频率取值在 1.2MHz～12MHz 之间。

片外振荡器如图 1-17（b）所示。XTAL1 是外部时钟信号的输入端，XTAL2 可悬空。由于外部时钟信号经过片内一个 2 分频的触发器进入时钟电路，因此对外部时钟信号的占空比没有严格要求，但高、低电平的时间宽度应不小于 20 ns。



（a）片内振荡器　　（b）片外振荡器

图 1-17　AT89S52 振荡器电路

2. 时序的概念

单片机内的各种操作都是在一系列脉冲控制下进行的，而各脉冲在时间上是有先后顺序的，这种顺序就称为时序。单片机内部已集成了振荡器电路，只需要外接一个石英晶体和两个频率微调电容就可工作。

（1）振荡周期是指晶体振荡器直接产生的振荡信号的周期，是振荡频率 fosc 的倒数，用 P 表示。

振荡周期 Tosc=1/fosc，如：为 6MHz 时 Tosc=1/6μs；为 12MHz 时 Tosc=1/12μs。

（2）状态周期，又称时钟周期，用 S 表示。每个状态周期是振荡周期的二倍，即每个状态周期分为 P1 和 P2 两个节拍，P1 拍节完成算术逻辑操作，P2 节拍完成内部寄存器间数据的传递。

（3）机器周期是机器的基本操作周期。一个机器周期含 6 个状态周期，分别用 S1～S6 表示，或用 S1P1、S1P2、…、S6P2 表示。

（4）指令周期，执行一条指令所占用的全部时间。一个指令周期通常由 1～4 个机器周期组成。AT89S52 系统中，有单周期指令、双周期指令和四周期指令。

如：fosc=12 MHz，1 个机器周期=6 个状态周期=12 个振荡周期。

则：振荡周期=1/12μs，振荡周期=1/6μs，机器周期=1μs，指令周期=1～4μs。各种周期的相互关系，如图 1-18 所示。

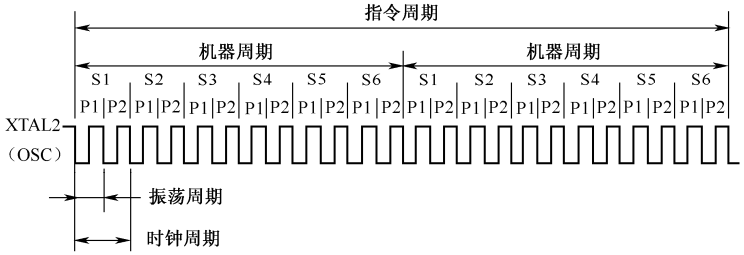


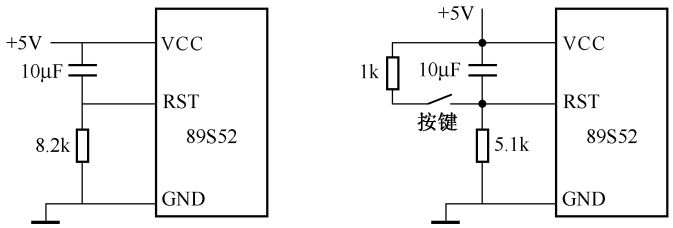
图 1-18　各种周期的相互关系

3. 复位电路

AT89S52 单片机的复位电路如图 1-19 所示。在 RST 输入端出现高电平时实现复位和初始化。

在振荡运行的情况下，要实现复位操作，必须使 RST 引脚至少保持两个机器周期（24 个振荡周期）的高电平。CPU 在第二个机器周期内执行内部复位操作，以后每一个机器周期重复一次，直至 RST 端电平变低。复位期间不产生 ALE 及 $\overline{\text{PSEN}}$ 信号。

图 1-19（a）为上电自动复位电路。加电瞬间，RST 端的电位与 VCC 相同，随着 RC 电路充电电流的减小，RST 的电位下降，只要 RST 端保持 10ms 以上的高电平就能使 AT89S52 单片机有效地复位，复位电路中的 RC 参数通常由实验调整。当振荡频率选用 6MHz 时，电容 C 选 22μF，电阻 R 选 1kΩ，便能可靠地实现加电自动复位。图 1-19（b）所示为为手动复位电路。



（a）上电自动复位电路　　（b）手动复位电路

图 1-19　复位电路

【技能训练 1-1】单片机最小系统应用——开关控制 LED 点亮

模块一是通过程序使 P1.0 引脚输出低电平来点亮 LED 的。在这里如果我们通过开关控制 LED 点亮，那么应如何实现？

1. 单片机最小系统应用

单片机最小系统只是单片机能满足工作的最低要求，它不能对外完成控制任务，实现人

机对话。要进行人机对话还要一些输入、输出部件，作控制时还要有执行部件。常见的输入部件有开关、按钮、键盘、鼠标等，输出部件有指示灯 LED、数码管、显示器等，执行部件有继电器、电磁阀等。

2. 电路设计

开关控制 LED 点亮是 AT89S52 单片机的一种最简单电路，它包含 3 个部分：单片机最小系统、输入电路和输出电路。单片机最小系统由 AT89S52 单片机、晶振电路和上电复位电路构成。输入部件是开关 SW，输出部件是 LED。如图 1-20 所示。由于只使用内程序存储器，AT89S52 的 EA 端接电源正端。

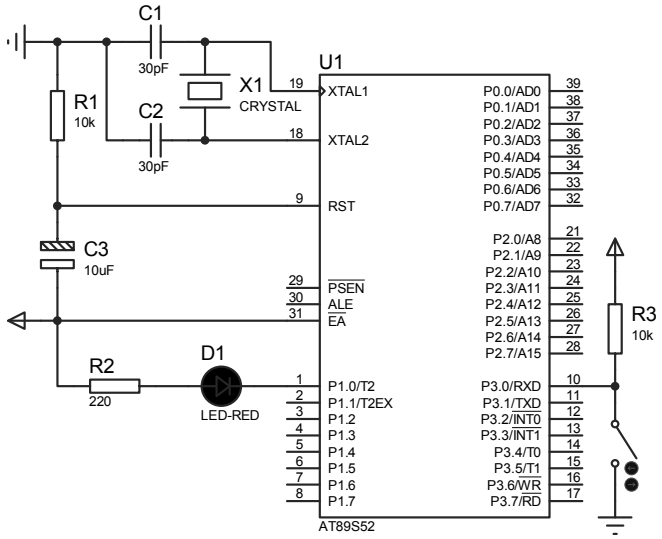


图 1-20　开关控制 LED 点亮电路

3. 程序设计

（1）开关控制点亮 LED 分析。

开关闭合：P1.0=0，LED 点亮；开关断开：P1.0=1，LED 熄灭。流程图如图 1-21 所示。

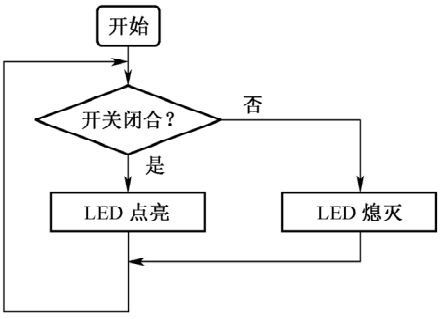


图 1-21　开关控制点亮 LED 流程图

（2）编写开关控制 LED 点亮程序。

```
#include <AT89X52.H>              //包含 AT89X52.H 头文件
sbit SW=P3^0;                     //定义 SW 是 P3.0 位对应的引用符号
sbit LED=P1^0;                    //定义 LED 是 P1.0 位对应的引用符号
void main (void)
{
        while(1)
        {
          if(SW==0)
            LED=0;                //开关闭合 SW=0：P1.0=0，LED 点亮
          else
            LED=1;                //开关断开 SW=1：P1.0=1，LED 熄灭
        }
}
```

1.3　工作模块 2　LED 闪烁控制

P1.0 引脚接发光二极管（LED）的阴极，通过程序控制，使 P1.0 引脚交替输出高电平和低电平，使发光二极管闪烁。

1.3.1　LED 闪烁控制设计与实现

LED 闪烁控制电路同模块一“点亮一个 LED”电路一样，是由 AT89S52 单片机最小系统和 LED 电路构成。

1. LED 闪烁功能实现分析

LED 的阳极通过 220Ω限流电阻后联接到 5V 电源上， P1.0 引脚接 LED 的阴极，P1.0 引脚输出低电平时，LED 点亮；输出高电平时，LED 熄灭。

LED 闪烁功能实现过程如下：

（1）P1.0 引脚输出低电平，LED 点亮；

（2）延时；

（3）P1.0 引脚输出高电平，LED 熄灭；

（4）延时；

（5）重复第一步（循环），这样就可以实现 LED 闪烁。

2. LED 闪烁控制程序设计

从以上分析可以看出，LED 闪烁控制 C 语言程序如下：

```
#include <AT89X52.H>              //包含 AT89X52.H 头文件
sbit LED=P1^0;                    //定义 LED 是 P1.0 位对应的引用符号
void   Delay()                    //延时函数
```

```
{
  unsigned char i, j;
       for (i=0;i<255;i++)
           for (j=0;j<255;j++);
}
void main()
{
   while(1)
    {
       LED = 0;                    // P1.0=0，LED 点亮
       Delay();                    //延时
       LED = 1;                    // P1.0=1，LED 熄灭
       Delay();
    }
 }
```


程序编程说明：

1）由于单片机执行指令的速度很快，如果不进行延时，点亮之后会马上就熄灭，熄灭了之后马上就点亮，速度太快，由于人眼的视觉暂留效应，人眼根本无法分辨，所以我们在控制 LED 闪烁的时候需要延时一段时间，否则我们就看不到“LED 闪烁”效果了。

2）延时函数是定义在前，使用在后。在这里使用了 2 条 for 语句构成双重循环（外循环和内循环），循环体是空的，实现延时的目的。如果想改变延时时间，可以通过改变循环次数调整来实现。

如果延时函数是使用在前，定义在后，程序应如何编写？

3）“unsigned char i, j;”语句是定义 i 和 j 两个变量为无符号字符型，取值范围为 0～255。

3. LED 闪烁控制调试及生成 HEX 文件

LED 闪烁程序设计好以后，我们还需要进行调试，看看是否与设计相符，首先要生成“LED 闪烁.hex”文件。在以后工作模块中不再详细叙述具体过程。

（1）建立工程文件，选择单片机。工程文件名为“LED 闪烁”，选择单片机型号为 Atmel 的 AT89S52。

（2）建立源文件，加载源文件。源文件名为“LED 闪烁.c”。

（3）设置工程的配置参数。“目标”标签页的晶振频率栏设为 12MHz，“输出”标签页的生成 HEX 文件选择框选中。

（4）进行编译和连接。

（5）进入调试模式，打开 P1 口对话框。在调试模式中，单击“外围设备”→I/O-Ports →Port 1，打开 P1 口对话框。

（6）全速运行程序。单击“调试”→“运行到”命令或单击调试工具栏的“运行”按钮，通过 P1 口对话框观察 P1.0 引脚的电平变化状态，以间接分析 LED 闪烁规律，是否与设计相符。调试窗口如图 1-22 所示。

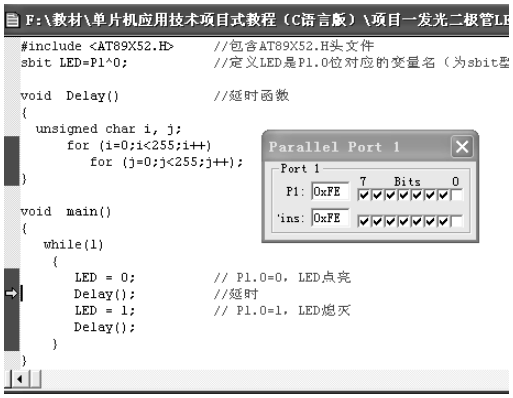


图 1-22　调试窗口

4. LED 闪烁控制 Proteus 仿真运行调试

LED 闪烁控制用 Proteus 仿真运行调试与工作模块 1 基本相同。在以后工作模块中不再详细叙述具体仿真运行调试过程。

（1）运行 Proteus 软件，打开 LED 闪烁 Proteus 仿真电路；

（2）加载 Keil 生成的“LED 闪烁.hex”HEX 文件；

（3）单击仿真工具栏“运行”按钮，单片机全速运行程序。

通过编辑区 LED 闪烁电路图观察 LED 闪烁规律，是否与设计要求相符。同时还可以通过 P1.0 引脚的电平变化状态，间接分析 LED 闪烁规律。LED 闪烁 Proteus 仿真运行如图 1-23 所示。

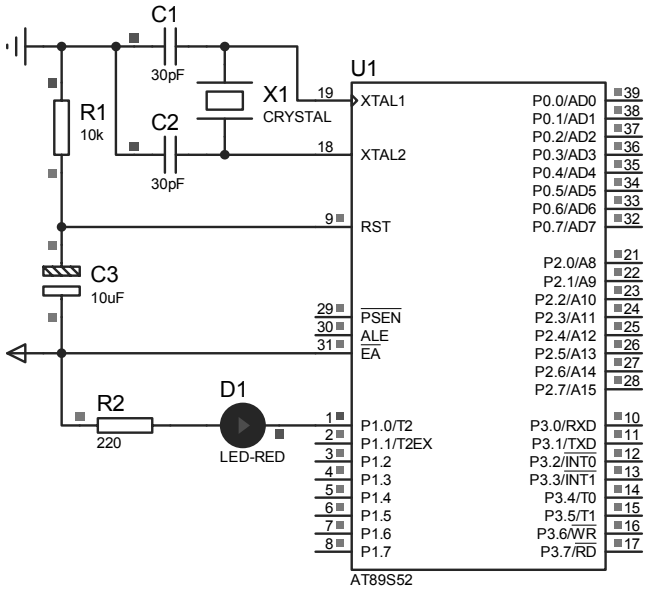


图 1-23　LED 闪烁 Proteus 仿真运行

1.3.2 LED 闪烁控制电路焊接制作

根据图 1-23 所示电路图，在万能板上完成单片机最小系统和 LED 电路焊接制作，元器件清单如表 1-2 所示。

表 1-2 单片机最小系统和 LED 电路元件清单

元件名称	参数	数量	元件名称	参数	数量
单片机	AT89S52	1 个	轻微按键		1 个
晶振	11.0592M	1 个	电阻	10kΩ	1 个
瓷片电容	104	2 个	电阻	220Ω	1 个
电解电容	10μF	1 个	LED		1 个
IC 插座	DIP40	1 个			

1. 电路板焊接

焊接电子元器件原则是由小到大、由矮到高；但首先要焊接 IC 插座、后面依次按模块电路（如：晶振电路、复位电路模块等）来进行焊接。焊接好的电路板如图 1-24 所示。

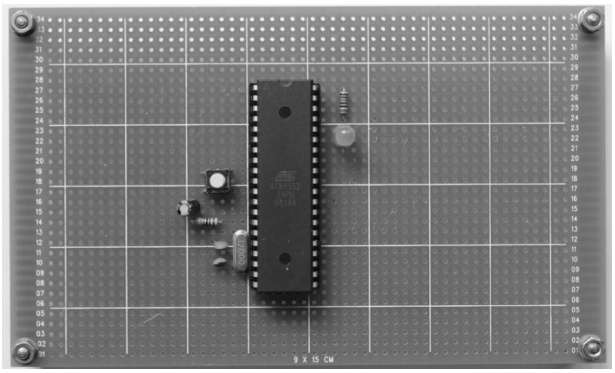


图 1-24 单片机最小系统和 LED 焊接电路板

元器件焊接时注意事项如下：

（1）电解电容（瓷片电容不分正负极）、发光二极管都具有一长一短两个引脚，长脚为正极、短脚为负极。

（2）焊接底座时要先焊接两个对角引脚，将其固定在线路板上，防止底座焊接不平，然后焊接其他引脚。底座焊接完后，将单片机芯片两排引脚分别向内侧压一下，以便插入芯片底座。

（3）晶振电路尽量靠近单片机芯片进行焊接，以减少寄生电容，更好地保证振荡器稳定和可靠地工作。选用复位电路所使用的开关时，最好选用点触开关，便于操作。

（4）焊接后，元器件外观要整齐、焊点要饱满（防止虚焊）、引脚不宜过高。在放置元器件时，还要考虑为方便以后开发，在线路板上预留一定空间。

2. 硬件检测与调试

（1）上电前，检测单片机 V_{CC} 和 GND 是否短路。上电后，检测单片机 40 脚和 20 脚之间是否有 5V 电压。

（2）检测晶振两端（18 脚与 19 脚）电压是否为 0.5～1.6V，如果有则说明晶振电路工作正常。

（3）按下复位按键，检测第 9 脚的电压是否会变化。若按键按下前电压为 0V，按键按下时电压立刻变为 5V，按键释放之后降为 0V，则表示复位电路正常。

3. 软件下载与调试

通过 ISP 下载器把“LED 闪烁.hex”文件烧入单片机芯片中，如果 LED 运行结果与设计功能相符，说明上面焊接过程和程序均正常，否则需进行调试，直到功能实现。软件下载、调试步骤如下：

（1）首先在计算机上运行 Easy 51Pro.exe，打开主窗口画面，如图 1-25 所示。

（2）单击右下角的“设置”按扭，弹出程序烧录界面，如图 1-26 所示，在“编程器类型”中设置选择“使用 Easy ISP 下载线”。

图 1-25　ISP 下载器主窗口

图 1-26　ISP 下载线设置界面

（3）把下载器的 8P 排线和单片机的 ISP 下载接口相接，另一端连接到 PC 机。在烧录界面中选择实际要烧录的芯片型号（AT89S52），再单击“检测器件”按钮，看看是否可以检测到所烧录的目标芯片。

（4）单击“自动打开文件”按钮，选择需要下载的程序 HEX 文件。然后一步一步的手动完成。也可以单击“自动完成”按钮，就会一项一项的自动往下进行，烧录完成就可以运行 LED 闪烁控制程序了。

上电后，就可以观察到 LED 不断闪烁，直到电源关闭才能熄灭。

1.3.3 C 语言程序的基本构成

随着单片机开发技术的不断发展， 目前已有越来越多的人从普遍使用汇编语言到逐渐使用高级语言开发，其中主要是以 C 语言为主，市场上几种常见的单片机均有其 C 语言开发环境。这里以最为流行的 AT89S52 单片机为例，来学习单片机的 C 语言编程技术。

1. C 语言程序的构成

（1）C 语言的程序是由一个或多个函数构成的，最简单的程序只有一个 main 函数，如工作模块 1 的“点亮一个 LED”C 语言程序。在一个 C 语言程序中必须有且仅有一个 main 函数，除了 main 函数，还可以有其他的函数，由于这些其他的函数是由用户根据需要自行设计的，因此将这些函数称为自定义函数，如工作模块 2 的“LED 闪烁”C 语言程序中的 Delay()函数。另外，在 C 语言程序中，还可以有由 C 语言本身提供的函数，即库函数。

那么库函数和用户自定义函数有什么区别呢？简单地说，任何使用 Keil C 语言的人，都可以直接调用 C 语言的库函数而不需要为这个函数写任何代码，只需要包含具有该函数说明的相应的头文件即可；而自定义函数则是完全个性化的，是用户根据自己需要而编写的。Keil C 提供了 100 多个库函数供我们直接使用。

一个 C 语言程序，总是从 main 函数开始执行的，而不管物理位置上这个 main()放在什么地方。在“LED 闪烁”C 语言程序中，main()就是放在了最后，事实上这往往是最常用的一种方式。

（2）一个函数由两部分组成。

1）函数的首部、即函数的第一行。包括函数名、函数类型、函数属性、函数参数（形参）名、参数类型。如：

```
void Delay()
```

一个函数名后面必须跟一对圆括号，即便没有任何参数也是如此。

2）函数体，即函数首部下面的大括号“{}”内的部份。如果一个函数内有多个大括号，则最外层的一对“{}”为函数体的范围。函数体一般包括：

声明部分：定义所用到的变量，如 void Delay()中的 unsigned char i, j;。

执行部分：由若干个语句组成。

在某些情况下也可以没有声明部分，甚至既没有声明部分，也没有执行部分，如：

```
void Delay()
{}
```

这是一个空函数，什么也不干，但它是合法的。

在编写程序时，可以利用空函数，比如主程序需要调用一个延时函数，可具体延时多少，怎样延时，暂时还不清楚，我们可以先把主程序的框架结构弄好，编译通过了再说。至于里面的细节，可以在以后慢慢地填，这样在主程序中就可以调用它了。

2. 标识符

变量名、常数名、数组名、函数名、文件名与类型名等统称为标识符。C 语言规定标识符

只能由字母、数字和下划线三种字符组成，且第一个字符必须为字母或下划线，如“1A”是错误的，编译时便会有错误提示。要注意的是 C 语言中大写字母与小写字母被认为是两个不同的字符，即 Sum 与 sum 是两个不同的标识符。

可以将标识符分为预定义标识符和用户标识符。标准库函数的名字，如 printf、sqrt、pow 与 sin 等，还有预编译处理命令，如 define 与 include 等，都属于预定义标识符。而用户标识符则是由用户根据需要定义的标识符。如用户定义的变量名 a、b、sum 与 x1 等、用户定义的函数名 f1、rep、facto 与 sort 等。

标识符在命名时应当简单，含义清晰，这样有助于阅读理解程序。标准的 C 语言并没有规定标识符的长度，但是各个 C 编译系统有自己的规定，在 Keil C 编译器中，只支持标识符的前 32 位为有效标识。

3. 关键字

关键字则是编程语言保留的特殊标识符，它们具有固定名称和含义，在程序编写中不允许标识符与关键字相同。在 Keil C 中的关键字除了有 ANSI C 标准的 32 个关键字外，还根据 51 单片机的特点扩展了相关的关键字。

在 Keil C 的文本编辑器中编写 C 程序，系统把保留字以不同颜色显示，缺省颜色为天蓝色。

1.3.4 C 语言基本语句

C 语言的程序是由一个或多个函数组成的，而函数又是由若干个语句组成的。语句是由一些基本字符和定义符按照 C 语言的语法规定组成的，每个语句以分号结束，分号是 C 语句的必要组成部分。C 语言的语句可分为以下 5 种类型：表达式语句、函数调用语句、控制语句、复合语句和空语句。

1. 表达式语句

表达式语句是由一个表达式加一个分号构成一个语句，其作用是计算表达式的值或改变变量的值。它的一般形式是：

表达式;

即在表达式末尾加上分号，就变成了表达式语句。最典型的表达式语句是：在赋值表达式后加一个分号构成赋值语句。例如

a=3

是一个赋值表达式，而

a=3;

是一个赋值语句。

2. 函数调用语句

由一个函数调用加一个分号构成函数调用语句，其作用是完成特定的功能。它的一般形式是：

函数名(参数列表);

例如

```
mDelay(100);  //调用延时函数，参数是 100
```

3. 控制语句

控制语句用于完成一定的控制功能，以实现程序的各种结构方式。C 语言有 9 种控制语句，可分为以下三类。

（1）条件判断语句：if 语句、switch 语句。

（2）循环语句：for 语句、while 语句、do-while 语句。

（3）转向语句：break 语句、continue 语句、goto 语句、return 语句。

4. 复合语句

复合语句是用一对大括号将若干条语句括起来的，也称为分程序，在语法上相当于一条语句。例如

```
main()
{……
    {t=x;
     x=y;
     y=t;}      //复合语句
}
```

5. 空语句

只有一个分号的语句称为空语句。它的一般形式是：

;

空语句是什么操作也不执行，常用于作为循环语句中的循环体，表示循环体什么也不做。

由于 C 语言程序的书写格式是自由的，所以，一个语句可写在一行上，也可分写在多行内。一行内可以写一个语句，也可写多个语句。书写的缩进没有要求，但是建议读者自己按一定的规范来写，可以给自己带来方便。

注释内容可以单独写在一行上，也可以写在一个语句之后。可以用/*…*/的形式为 C 程序的任何一部分作注释，在“/*”开始后，一直到“*/”为止的中间的任何内容都被认为是注释，所以在书写特别是修改源程序时特别要注意，有时无意之中删掉一个“*/”，结果，从这里开始一直要遇到下一个“*/”中的全部内容都被认为是注释了。原本好好的一个程序，编译已过通过了，稍作修改，一下出现了几十甚至上百个错误，这时就要检查一下，是不是有这样的情况，如果有的话，赶紧把这个“*/”补上。Keil C 也支持 C++风格的注释，就是用“//”引导的后面的语句是注释，例：

```
P1_0=!P1_0;  //取反 P1.0
```

这种风格的注释，只对本行有效，所以不会出现上面的问题，而且书写比较方便，所以在只需要一行注释的时候，我们往往采用这种格式。

1.4　技能拓展　音频控制应用

1.4.1　音频控制电路

音频控制电路由单片机最小系统、放大滤波电路和扬声器构成。放大滤波电路由 NPN 三极管 Q1、电阻 R3、电容 C4 构成，Q1 的基极经电阻 R2 接到 P0.0 引脚。如图 1-27 所示。

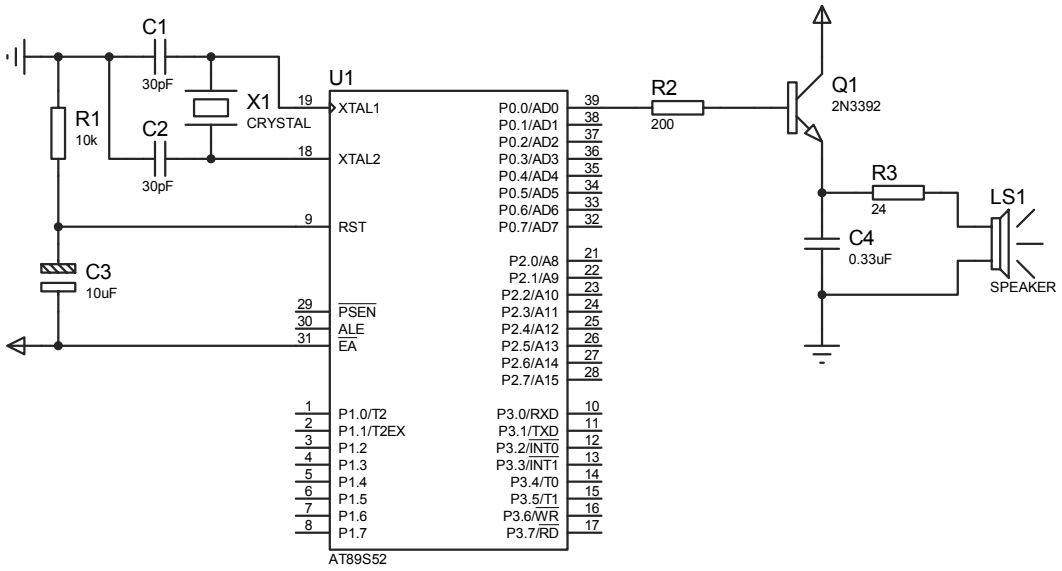


图 1-27　音频控制电路

1.4.2　音频控制程序

利用 AT89S52 端口输出脉冲方波，方波经放大滤波后，驱动扬声器发声，声音的频率高低由延时长短控制。

音频控制程序和工作模块 2“LED 闪烁”程序基本一样，下面写出程序不同之处。

```
#include <AT89X52.H>                    //包含 AT89X52.H 头文件
sbit SPK=P0^0;                          //定义 SPK 是 P0.0 位对应的引用符号
……
        SPK = 0;                        // P0.0=0，输出低电平
        Delay();                        //  延时
        SPK = 1;                        // P1.0=1，输出高电平
        Delay();
……
```

【技能训练 1–2】报警产生器

在图 1-27 基础上，P1.7 引脚接一个开关（参考图 1-20），用 P0.0 输出 1kHz 和 500Hz 的音频信号驱动扬声器，作报警信号，要求 1kHz 信号响 100ms，500Hz 信号响 200ms，交替进行。当开关合上报警信号响，当开关断开报警信号停止，参考程序如下。

```
#include <AT89X52.H>
#include <INTRINS.H>
unsigned char count;
void dely500(void)                                    //延时 500us，即 0.5ms
{
  unsigned char i;
  for(i=250;i>0;i--)
      {
        _nop_();
      }
}
void main(void)
{
  while(1)
      {
        if(P1_7==0)
          {
            for(count=200;count>0;count--)            //1kHz 信号响 100ms
              {
                P1_0=~P1_0;
                dely500();
              }
            for(count=200;count>0;count--)            //500Hz 信号响 200ms
              {
                P1_0=~P1_0;
                dely500();
                dely500();
              }
          }
      }
}
```


关键知识点小结

1．Proteus 能在计算机上完成从原理图与电路设计、电路分析与仿真、单片机代码级调试与仿真、系统测试与功能验证到形成 PCB 的完整的电子设计、研发过程。

2．Keil C51 是基于 8051 内核的微控制器软件开发平台，是 51 系列单片机 C 语言软件开发系统。可以完成从工程建立和管理、编译、连接、目标代码的生成、软件仿真和硬件仿真等完整的开发流程。

3．单片机的发展主要经历：第一阶段（1974～1976 年）为单片机初级阶段；第二阶段（1976～1978 年）为低性能单片机阶段；第三阶段（1978～1982 年）为高性能单片机阶段，也是单片机普及阶段，第四阶段（1982 年以后）为 16 位单片机阶段。

4．单片机主要应用在家用电器、智能卡、智能仪器仪表、网络与通信以及工业控制等方面。

5．AT89S52 单片机一个低功耗，高性能 CMOS 8 位单片机，在一块芯片中集成了 CPU，8K 字节的可反复擦写 1000 次的 Flash 只读程序存储器，256 字节内存，32 个输入/输出线，看门狗定时器，两个数据指针，三个 16 位定时器/计数器，6 矢量两个级别的中断结构，一个全双工串行口，片内振荡器和时钟电路。

6．AT89S52 单片机最小系统就是指由单片机和一些基本的外围电路所组成的一个可以工作的单片机系统。一般来说，它包括单片机、电源、晶振电路和复位电路。

7．单片机内的各种操作都是在一系列脉冲控制下进行的，而各脉冲在时间上是有先后顺序的，这种顺序就称为时序。定时单位有：振荡周期 fosc、状态周期、机器周期、指令周期。

1 个机器周期=6 个状态周期=12 个振荡周期 fosc；

1 个指令周期通常由 1～4 个机器周期组成。

8．C 语言的程序是由一个或多个函数构成的，在一个 C 语言程序中必须有且仅有一个 main 函数，除了 main 函数，还可以有自定义函数和库函数。一个函数由两部组成：函数的首部，包括函数名、函数类型、函数属性、函数参数（形参）名、参数类型；函数体，即函数首部下面的大括号“{}”内的部分。

9．C 语言规定标识符只能由字母、数字和下划线三种字符组成，且第一个字符必须为字母或下划线。标识符分为预定义标识符和用户标识符。标准库函数的名字和预编译处理命令都属于预定义标识符。而用户标识符则是由用户根据需要定义的标识符。在 Keil C 编译器中，只支持标识符的前 32 位为有效标识。

10．关键字则是编程语言保留的特殊标识符，它们具有固定名称和含义，在程序编写中不允许标识符与关键字相同。在 Keil C 中的关键字除了有 ANSI C 标准的 32 个关键字外，还根据 51 单片机的特点扩展了相关的关键字。

11．C 语言的语句是由一些基本字符和定义符按照 C 语言的语法规定组成的，每个语句以分号结束，分号是 C 语句的必要组成部分。C 语言的语句可分为：表达式语句、函数调用语句、控制语句、复合语句和空语句。

12．“#include <AT89X52.H>”语句是一个“文件包含”处理，是将 AT89X52.H 头文件的内容全部包含进来。“sbit LED=P1^0;”语句是定义用符号 LED 来表示 P1.0 引脚。Keil C 支持 C++风格的注释，可以用“//”进行注释，也可以用/*……*/进行注释。

问题与讨论

1-1　填空题。

（1）在 51 系列单片机中，无内部 ROM 的单片机型号是_____，有 4KB EPROM 的单片机型号是_____。

（2）典型单片机的结构可分为 CPU、_____、_____、_____四部分。

（3）单片机常用两种复位方式，即_____和_____。

（4）一个机器周期包含_____个晶振周期，若晶振周期的频率为 12MHz，则机器周期为_____，指令周期为_____~_____机器周期。

（5）当 P1 口做输入口输入数据时，必须先向该端口的锁存器写入_____，否则输入数据可能出错。

（6）MCS-51 系列单片机有_____个并行 I/O 口，_____个全双工串口，_____个 16 位定时器/计数器，_____中断源。

1-2　选择题。

（1）使用单片机开发系统调试程序时，对源程序进行汇编的目的是（　　）。

A．将源程序转换成目标程序　　B．将目标程序转换成源程序

C．将低级语言转换成高级语言　　D．连续执行键

（2）单片机的简称是（　　）。

A．MCP　　B．PLC　　C．MCU　　D．DSP

（3）以下叙述不正确的是（　　）。

A．一个 C 程序可以由一个或多个函数组成

B．一个 C 程序必须包含一个 main 函数

C．C 程序的基本组成单位是函数

D．在 C 程序中，注释说明只能位于一条语句的后面

（4）提高单片机的晶振频率 fosc，则机器周期 Tcy（　　）。

A．不变　　B．变长　　C．变短　　D．不定

（5）一个 C 语言程序的执行是从（　　）。

A．本程序的 main 函数开始，到 main 函数结束

B．本程序文件的第一个函数开始，到本程序文件的最后一个函数结束

C．本程序的 main 函数开始，到本程序文件的最后一个函数结束

D．本程序文件的第一个函数开始，到本程序文件的 main 函数结束

1-3　简叙 Keil C51 和 Proteus 软件的主要功能。

1-4　简叙单片机的发展及发展趋势。

1-5　简叙单片机的主要应用领域。

1-6　简叙单片机的主要特点。

1-7　如果只使用片外 ROM，$\overline{EA}$ 引脚应该如何连接？为什么？

1-8　在工作模块 2 中，如果把 LED 闪烁程序的延时函数 Delay()写在 main()后面，程序应该如何修改？

1-9　单片机最小系统由哪几部分组成？现要求 LED 的阳极接在 P1.0 引脚上，请完成 LED 点亮电路和 C 语言程序设计。

1-10　请完成用开关控制 LED 闪烁快和慢两种效果的电路和 C 语言程序设计。