职业教育课程改革创新规划教材

基于 C 语言与 Proteus 联合仿真的单片机技术

主　编　丘利丽　何　波

参　编　李旭伟　陆志强

主　审　陈琨韶

電子工業出版社

Publishing House of Electronics Industry

北京・BEIJING

内 容 简 介

本书适用于一体化教学。全书包括五个学习情境，分别是海珠桥灯饰工程的设计与调试、数字钟的设计与调试、轻工 LED 电子显示屏的设计与调试、家居报警系统的设计与调试和超声波汽车倒车雷达的设计与调试。这五个学习情境的整体结构采用由易到难、循序渐进的方式，内容包含了单片机最小系统、传感器、按键输入、定时中断、流水灯、数码管、点阵、LCD 液晶显示器、继电器、蜂鸣器、步进电机和超声波知识点。每个学习情境分为几个学习任务，学习任务之间互有关联，都是为了实现学习情境中的最终产品而服务。每个学习任务中的程序层层递进，后面的程序在前面程序的基础上，稍作改动，即可实现新任务，让读者轻轻松松学习单片机。

本书可作为职业院校的单片机教材，也可以作为广大单片机爱好者及相关电子工程技术人员的参考书。

为方便教学，本书配有电子教学资料包，有需要的教师请登录华信教育资源网（www.hxedu.com.cn），免费注册后下载。

图书在版编目（CIP）数据

基于 C 语言与 Proteus 联合仿真的单片机技术 / 丘利丽，何波主编. —北京：电子工业出版社，2019.4

ISBN 978-7-121-35409-0

Ⅰ. ①基… Ⅱ. ①丘… ②何… Ⅲ. ①单片微型计算机—职业教育—教材 Ⅳ. ①TP368.1

中国版本图书馆 CIP 数据核字（2018）第 247164 号

策划编辑：白　楠
责任编辑：白　楠　　特约编辑：王　纲
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×1 092　1/16　印张：12.75　字数：326.4 千字
版　　次：2019 年 4 月第 1 版
印　　次：2019 年 4 月第 1 次印刷
定　　价：30.50 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

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

本书咨询联系方式：（010）88254592，bain@phei.com.cn。

前　言

随着嵌入式技术的飞速发展，嵌入式系统产品正不断渗透到各行各业，如智能家居、车载电子设备等。因此，单片机技术作为嵌入式计算机控制系统的重要技术，已经越来越受到各个应用领域的重视，尤其对于直接面向企业的职业院校，掌握单片机技术已经成为机电技术应用、电气控制、数控技术、电子信息、计算机应用等专业学生的基本技能。世界技能大赛之电子技术大赛中，单片机技术竞赛内容占有半壁江山。因此，全国的高职院校越来越重视单片机技术的教学。

单片机技术是一门理论与实践结合较强的技术。目前有关单片机的教材大多偏重理论，应用性方面的介绍比较少，并且章节之间没有太多的联系，不适应于现在高职教育提倡的“工学结合”的一体化教学模式。基于上述背景，编者结合自己十余年的单片机教学和指导学生参加世界技能大赛之电子技术大赛的经验，花费了两年多的时间编写本书，该书基于企业的“P（任务分析）-D（硬件及软件设计）-C 任务检查）-A（任务评估）”模式，体现了高职“学中教、教中学”的一体化教学特色。本书的特点包括以下几个方面。

1．一体化教学，按企业的电子技术产品开发过程实施教学。

本书遵循企业的电子技术产品开发过程原则，让学生经历“任务分析→硬件设计→软件设计→硬件安装→整机调试→整机产品评估”工作过程，让学生从任务中来，到任务中去，提早让学生体验就业岗位，提高学生的职业认同感。

2．选取典型、完整、难度适中的产品贯彻学习情境，结合理论和实践教学。

本书创设了五个学习情境，分别是海珠桥灯饰工程的设计与调试、数字钟的设计与调试、轻工 LED 电子显示屏的设计与调试、家居报警系统的设计与调试和超声波汽车倒车雷达的设计与调试。这五个学习情境的整体结构采用由易到难、循序渐进的方式，内容包含了单片机最小系统、传感器、按键输入、定时中断、流水灯、数码管、点阵、LCD 液晶显示器、继电器、蜂鸣器、步进电机和超声波知识点。每个学习情境分为几个学习任务，学习任务之间互相关联，都是为了实现学习情境中的最终产品而服务。每个学习任务中的程序层层递进，后面的程序在前面程序的基础上，稍作改动，即可实现任务，让读者轻轻松松学单片机。每个实例演练完后，进一步提出“思考题”，让读者能即学即用，所学知识更加扎实。

3．本书配套了“单片机实训开发板”和 Proteus 仿真图，方便“虚实相结合”的教学。

编者根据多年的教学经验，自行研发了与教材对应的一套“单片机实训开发板”。该套开发板全部是 PCB 板，分为多个模块，如单片机最小系统模块、流水灯模块、点阵模块、LCD 模块等。该开发板使用简单，模块与模块之间采用跳线的形式连接，并且只要 1 条 USB 线把开发板与计算机连接，就可以实现程序下载。

编者建议：为提高学生的学习兴趣，有条件的学校可以为每位上课的学生配一套“单片机实训开发板”。这样学生可以利用开发板在实验室、图书馆、宿舍等地方随时随地学习

单片机技术。若读者需要本书配套的开发板，可以与编者联系，邮箱为“38729128@qq.com”。

4．本书基于 C 语言与 Proteus 联合仿真，采用多文件、多任务的编程思路与方法。

C 语言具有易阅读、易移植的特点，现已成为嵌入式产品开发的主流语言。本书结合学习情境，采用 C 语言与 Proteus 联合仿真，在任务中理解和掌握 C 语言的理论知识，并且应用到实际任务中，达到举一反三的目的。在实际工作中，项目是一个大的工程，需要按功能进行分解。一般一个功能对应一个任务，一个任务对应一个程序文件，所以编者在本书中引入多文件、多任务的编程思路和方法。例如，学习情境五就是一个较大的工程，包含多个任务。通过该学习情境的学习，读者可以掌握多任务、多实时调度、多文件程序结构的综合系统调试方法。

广州市轻工高级技工学校的丘利丽、何波对本书的编写思路与大纲进行了总体策划，指导了全书的编写。丘利丽编写了学习情境一和学习情境二，何波编写了学习情境四和学习情境五，广州市轻工高级技工学校的陆志强参与编写了学习情境二，广州市轻工高级技工学校的李旭伟编写了学习情境三。广州市轻工高级技工学校的陈琨韶担任了本书主审，并提供了宝贵的编写建议。在此，一一表示衷心的感谢！

由于时间仓促和编者水平有限，书中难免有错误和不妥之处，恳请读者对本书提出批评与建议。

编　者

目　　录

情境 1　海珠桥灯饰工程的设计与调试

情境介绍

随着人们生活环境的不断改善和美化，在许多场合可以看到彩色霓虹灯不断变化。闪烁的 LED，以造价低廉、控制简单等特点得到了广泛的应用，用彩灯来装饰街道和城市建筑物已经成为一种时尚。

海珠桥是广州历史悠久的广府文化传承代表。因此，本情境以单片机控制技术为基础，由学生自主完成海珠桥的硬件和软件设计与调试，设计丰富多彩的海珠桥灯饰效果，使其摇身变为珠水之上的巨型灯屏。

学习任务一：8 位流水灯的设计与调试

任务描述

在 Proteus 仿真软件和单片机开发板上实现 8 位流水灯顺序点亮效果，并能控制它们的点亮速度。

任务目标

（1）能正确分析单片机最小系统的电路结构及各部分的功能。

（2）学会根据任务要求，自主设计 8 位流水灯的硬件电路。

（3）正确理解 Keil C 语言的基本结构、数据类型、常数、变量、运算符、循环指令及选择指令的知识点。

（4）熟练使用 Proteus 和 Keil μVision3 软件，完成程序的设计与调试。

（5）能正确使用 STC-ISP-V488 程序下载软件，完成程序的下载，并观察开发板上 8 位流水灯的工作过程。

建议课时：18 课时

任务分析

单片机 P2 口连接 8 个发光二极管，利用各引脚输出电位的变化，控制发光二极管的亮灭：输出电位为高电平，发光二极管灭；输出电位为低电平，发光二极管亮。为了清楚地分辨发光二极管的点亮和熄灭，编写延时程序，在 P2 口输出信号由一种状态向另一种状态变化时，实现一定时间的间隔。

任务实施

一、硬件电路设计

1．硬件设计思路

设计思路：利用 STC 单片机芯片，外加振荡电路、复位电路、控制电路、电源，组成一个单片机最小系统。在最小系统的基础上，利用 P2 口的 8 个引脚控制 8 个发光二极管。由于发光二极管具有普通二极管的共性——单向导电性，因此只要在其两极间加上合适的正向电压，发光二极管即可点亮；将电压撤除或加反向电压，发光二极管即熄灭。根据发光二极管的特性，结合单片机 P2 口的输出信号，即可实现流水灯的控制效果。

2．电路硬件设计

选用 51 单片机芯片，该芯片共有 40 个引脚，如图 1-1-1 所示。

（1）主电源电路

VCC（40 脚）：接+5V 电源，又称电源引脚。

GND（20 脚）：接地。

（2）时钟电路

单片机时钟信号的提供有两种方式：内部方式和外部方式。

内部方式是指使用内部振荡器，在 XTAL1（19 脚）和 XTAL2（18 脚）之间外接石英晶体和陶瓷电容 C_1 和 C_2［图 1-1-2（a）］，它们和单片机的内部电路构成一个完整的振荡器，振荡频率和石英晶体的振荡频率相同。电容器 C_1 和 C_2 容量为 30pF，石英晶体的

振荡频率为 12MHz。

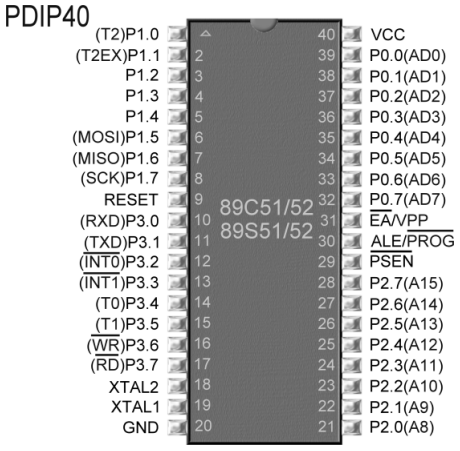
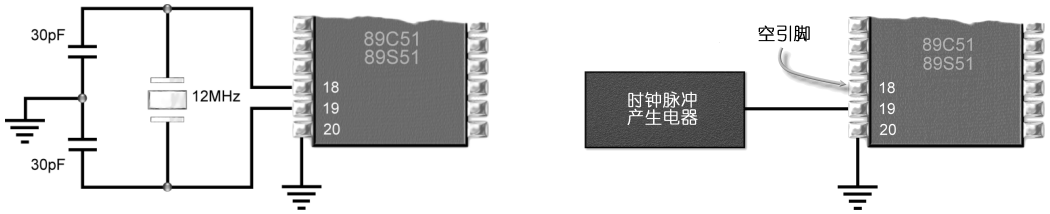


图 1-1-1　单片机芯片引脚

当使用外部信号源为单片机提供时钟信号时，XTAL1 为空引脚，XTAL2 外接时钟信号，如图 1-1-2（b）所示。



（a）内部振荡电路产生脉冲　　（b）使用外部信号源产生脉冲

图 1-1-2　单片机时钟电路

本任务使用内部振荡器，因此在 XTAL1 和 XTAL2 之间外接 12MHz 的石英晶体和 30pF 的陶瓷电容 C_1 和 C_2 即可。

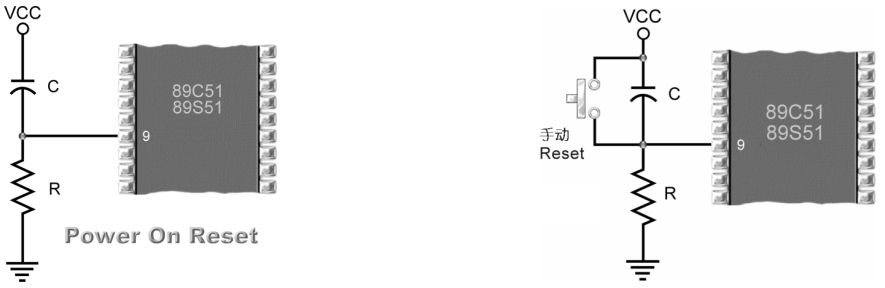
（3）复位电路

复位是单片机的初始化操作，使 CPU 及其他功能部件都处于一个确定的初始状态，并从这个状态开始工作。除系统正常上电（开机）外，在单片机工作过程中，如果程序运行出错或操作错误使系统处于死机状态，也必须进行复位，使系统重新启动。

复位引脚是第 9 脚，此引脚连接高电平超过两个机器周期，即可产生复位的动作。以 12MHz 的时钟脉冲为例，每个时钟脉冲为 1/12μs，两个机器周期为 2μs，因此，在该引脚上产生一个 2μs 以上的高电平脉冲，即可产生复位的动作。

复位有上电复位和按键复位两种，如图 1-1-3 所示。上电复位［图 1-1-3（a)］利用复位电路电容充放电来实现；而按键复位［图 1-1-3（b)］通过使 RST 端经电阻器 R 与+5V 电源接通而实现，它兼有自动复位的功能。

电路中 R 和 C 组成一个典型的充放电电路，充放电时间 $T=1/RC$。根据理论计算结果可知，选择时钟频率为 12MHz，一个机器周期是 1μs。只要 $T>2$μs，就可以复位。本任务开发板选用 $R=10$kΩ，$C=10$μF。



（a）上电复位　　（b）按键复位

图 1-1-3　单片机复位电路

（4）存储器设置电路

31 脚 EA 为复用引脚。当 EA 为低电平时，系统使用外部存储器。当 EA 为高电平时，系统使用内部存储器。对于初学者而言，所写的程序比较简单，大多使用内部存储器，所以就把 31 脚直接接到 VCC。

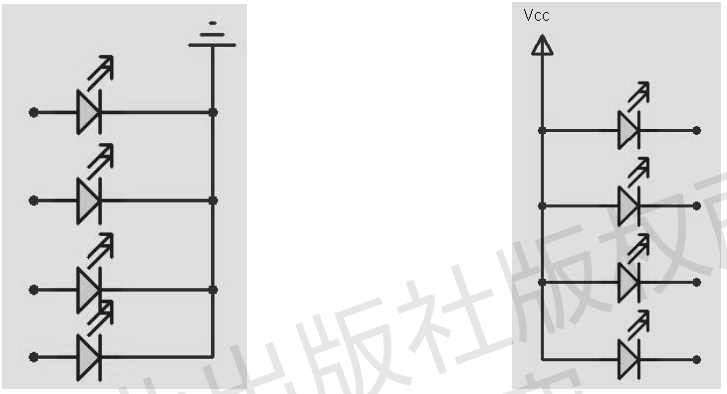
30 脚 ALE 是地址锁存信号，其功能是在存取外部存储器时，将原本在 P0 的地址信号锁存到外部存储器 IC，让 P0 口空出来，以传输数据。简单讲，当外接存储器电路时，让 ALE=1，P0 被用作地址总线；让 ALE=0，P0 被用作数据总线。

29 脚 PSEN 是程序存储器使能端，其功能是读取外部存储器。通常此引脚连接到外部存储器的 OE 引脚。

相对于前面的引脚，29、30 脚比较难以理解。但是只要不用外部存储器，就可以当它们不存在，悬空处理即可。

（5）流水灯控制电路

发光二极管的连接方法：若将它们的阴极连接在一起，阳极信号受控制，即构成共阴极接法，如图 1-1-4（a）所示；若将它们的阳极连接在一起，阴极信号受控制，则构成共阳极接法，如图 1-1-4（b）所示。由于 P2 口引脚输出高电位时电压大约是 5V，为保证发光二极管可靠工作，必须在发光二极管和单片机输出引脚间连接一只限流电阻。



（a）共阴极发光二极管的接法　　（b）共阳极发光二极管的接法

图 1-1-4　发光二极管的接法

本任务选用硅型普通发光二极管，限流电阻取 220Ω。

3．硬件电路原理图

单片机的 P0、P1 和 P2 端口都是双向的 I/O 端口，P3 端口既可作为普通的 I/O 端口，又可用于第二功能操作中。在该任务中，选择 P2 端口作为流水灯的控制端口，实现数据的输入输出。综上分析，得到图 1-1-5 所示的 8 位流水灯的电路原理图。

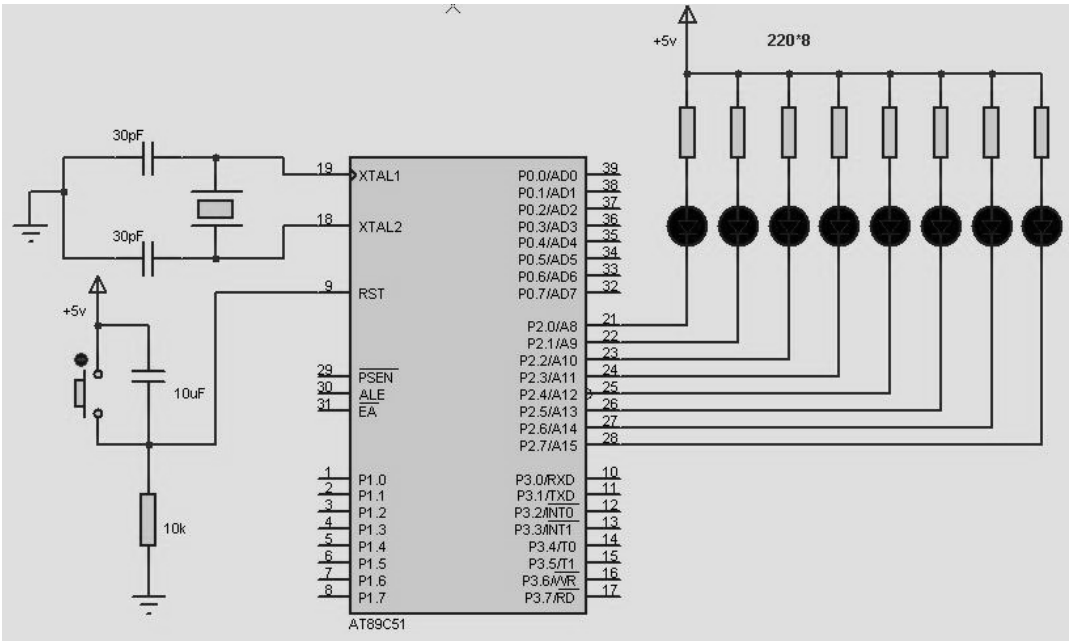


图 1-1-5　8 位流水灯的电路原理图

根据电路原理图，确定本任务所需要的元器件清单，见表 1-1-1。

表 1-1-1　海珠桥灯饰元器件清单

序　　号	名　　称	型　　号	数量（个）
1	单片机	AT89C51	1
2	发光二极管：LED	-	8
3	电阻：RES	220Ω	8
		10kΩ	1
4	电容：CAP	10μF	1
		30pF	2
5	晶体振荡器：CRYSTAL	12MHz	1
6	按钮：BUTTON	不带自锁	1

4．在 Proteus 仿真软件上绘制流水灯电路原理图

（1）打开软件

选择“程序”→“Proteus 7 Professional”→“ISIS 7 Professional”命令，启动 Proteus 仿真软件，出现 ISIS Professional 图像编辑窗口，如图 1-1-6 所示。

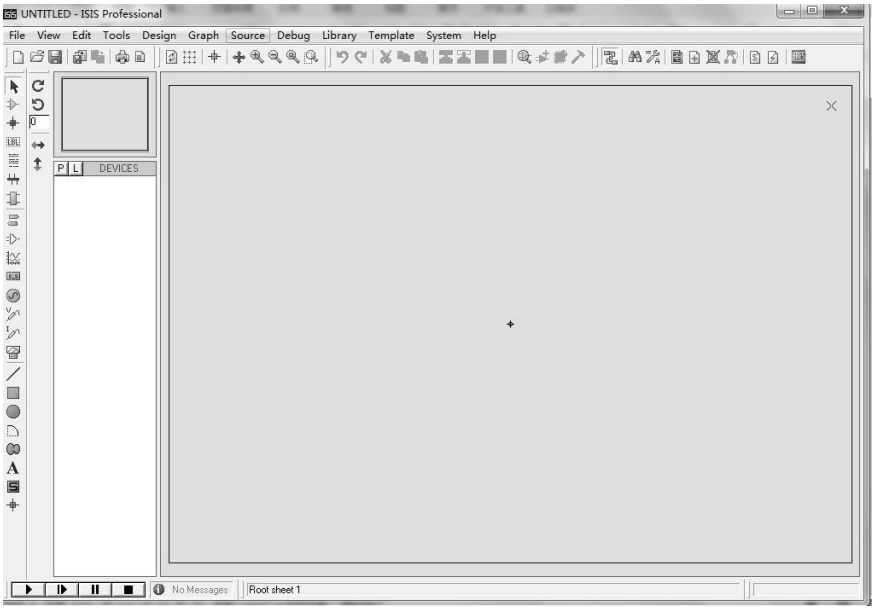


图 1-1-6　ISIS Professional 图像编辑窗口

（2）从 Proteus 库中选取元器件

以电阻 RES 为例，讲述元器件的选择方法。

在元件选择器工具栏（Mode Selector Toolbar）中单击选择元器件按钮，单击元器件列表上方的“ P ”按钮，打开元器件选择窗口，如图 1-1-7 所示。

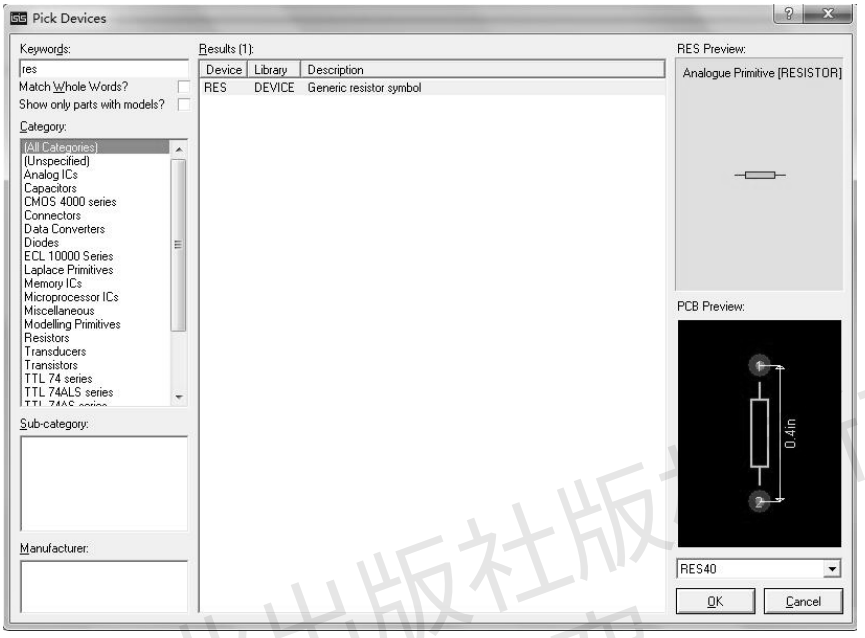


图 1-1-7　元器件选择窗口

在元器件选择窗口左上角的关键字栏中输入关键字，例如需要电阻就输入“ res ”，

从元件库中选取元器件。以此类推，可以选取单片机、电容、发光二极管、按钮、晶振等元器件。

（3）放置元器件

在对象选择器中单击要放置的元器件（蓝色高亮条表示目前选取的元器件），然后在编辑窗口中合适的位置单击就放置了一个元器件。依次把各元器件放入编辑区中的适当位置。

若要改变元器件的放置方向，可以右击选中元器件后再单击按钮或。若要镜像，可以先右击选中元器件，再单击按钮或。若要多个元器件一起转向，可先按住左键拖出方框选中多个元器件，再单击相应的操作按钮。

（4）放置电源和地

单击元件选择器工具栏中的端子按钮，在对象选择器中选取电源（POWER）、地（GROUND），分别放置于编辑窗口的合适位置上。

（5）连线

分别单击要连线（元器件引脚、终端、线）的起点和终点，在这两点间会自动生成一条线。若终点在空白处，双击即可结束画线。

（6）元器件属性设置

先左键双击各元器件，在弹出的属性编辑对话框（Edit Component）中，按电路原理图中各元器件的值设置相应的属性。

（7）绘制原理图并保存设计

原理图绘制完毕后，单击“File”→“Save as”保存到指定的文件夹，如图 1-1-8 所示。

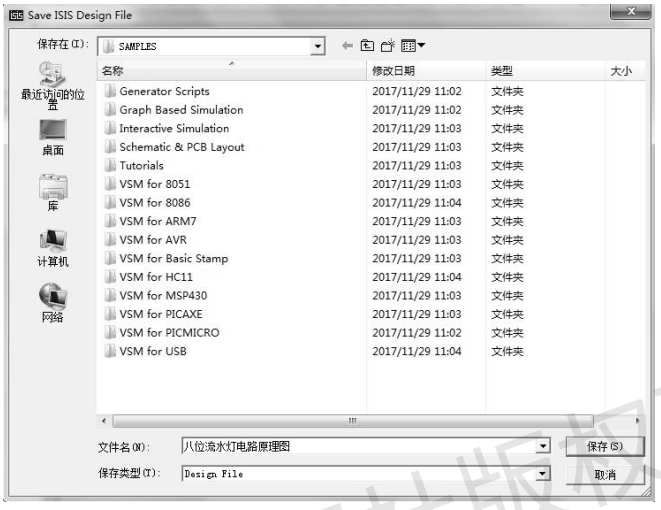


图 1-1-8　保存“八位流水灯电路原理图”

二、软件程序设计与调试

1．软件设计思路

利用单片机的 P2 口来控制 8 个 LED，让这 8 个 LED 依次点亮，其设计步骤如下。

当 P2 口的引脚输出低电平（0）时，其所连接的 LED 呈现正向偏压而发亮；当引脚输出高电平（1）时，其所连接的 LED 呈现反向截止而熄灭。因此，我们的程序设计就要让 P2.0 口接的灯亮，输出为 11111110，以十六进制表示为“ FE ”；延时一段时间后，P2.1 口接的灯亮，输出为 11111101，以十六进制表示为“ FD ”。以此类推，周而复始。

2．绘制程序流程图

有了设计思路后，可以将思路转换成流程图，如图 1-1-9 所示。

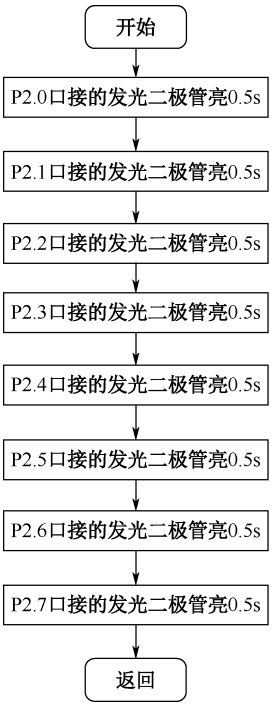


图 1-1-9　8 位流水灯循环点亮程序流程图

3．编写程序

8 位流水灯循环点亮的程序如下：

```
-------------------------------------------------------------------------------
      //====================================声明区=========================
      #include <reg51.h>       //定义8051寄存器的头文件
      #define led P2
      void delay(int);         //声明延迟函数
      //====================================主程序=========================
      main( )                  //主函数开始
      { led=0xff;              //流水灯初始状态，全熄灭
      while(1)                 //无限循环
      {led=0xfe;               //P2.0口的灯亮
      delay(500);              //延时0.5s
      led=0xfd;                //P2.1口的灯亮
      delay(500);              //延时0.5s
      led=0xfb;                //P2.2口的灯亮
```

```
        delay(500);                 //延时0.5s
        led=0xf7;                   //P2.3口的灯亮
        delay(500);                 //延时0.5s
        led=0xef;                   //P2.4口的灯亮
        delay(500);                 //延时0.5s
        led=0xdf;                   //P2.5口的灯亮
        delay(500);                 //延时0.5s
        led=0xbf;                   //P2.6口的灯亮
        delay(500);                 //延时0.5s
        led=0x7f;                   //P2.7口的灯亮
        delay(500);                 //延时0.5s
        }                           //while循环结束
        }                           //主程序结束
        //==================================延迟函数=========================
        void delay(int x)           //延迟函数开始，x=延迟次数
        {int i, j;                  //声明整数变量i，j
        for (i=0;i<x;i++)           //计数x次，延迟x×1ms
        for (j=0;j<120;j++);        //计数120次，延迟1ms
        }                           //延迟函数结束
-------------------------------------------------------------------------------
```

4．在 Keil μVision3 集成开发环境中新建工程和文件，编写流水灯程序

（1）在“开始”菜单中，选择“程序”→“Keil μVision3”选项，即可进入集成开发环境，如图 1-1-10 所示。

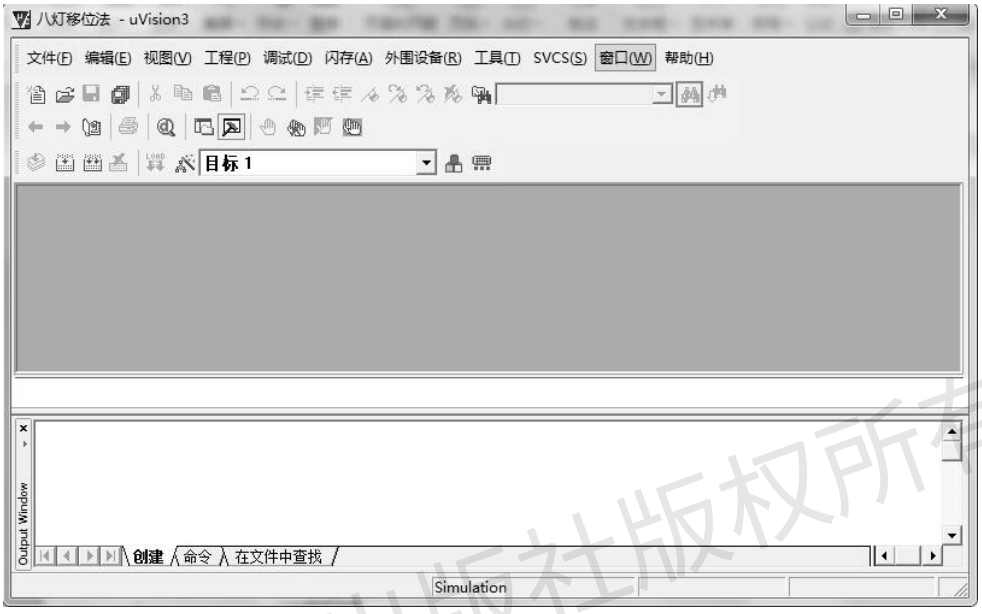


图 1-1-10　Keil μVision3 集成开发环境

（2）打开一个项目，启动“工程”菜单下的新建工程命令，出现如图 1-1-11 所示的对话框。

图 1-1-11　保存项目

（3）在“ 文件名 ”栏中填写要新增的项目名称，再单击“ 保存 ”按钮，出现如图 1-1-12 所示的对话框。

图 1-1-12　选择器件

（4）选择所要使用的 CPU 芯片，如 Atmel 公司的 AT89C51，再单击 “ 确定 ” 按钮，关闭对话框，出现如图 1-1-13 所示的对话框。

图 1-1-13　添加启动代码

（5）这时系统询问我们要不要将汇编语言的启动代码放入所编辑的项目文件，在此单击 “ 否 ” 按钮关闭此对话框，则在左边产生 “ 目标 1 ” 项目，如图 1-1-14 所示。

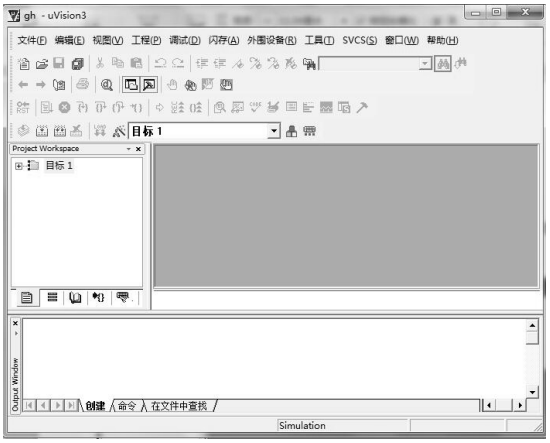


图 1-1-14　新建项目界面

（6）单击“ 工程 ”→“ 为目标 1 设置选项 ”，出现如图 1-1-15 所示的对话框。

图 1-1-15　选择晶振频率

（7）在此对话框中设置此芯片的工作频率为 12MHz，然后单击“ 输出 ”选项卡，如图 1-1-16 所示。

图 1-1-16　选择产生十六进制文件

（8）选择产生十六进制文件，单击“ 确定 ”按钮关闭对话框。

（9）单击“ 文件 ”→“ 新建文件 ”，编辑区将打开一个全新的编辑窗口，然后在对话框中输入要保存的文件名，后缀名为“ .c ”，再单击“ 保存 ”按钮，如图 1-1-17 所示。

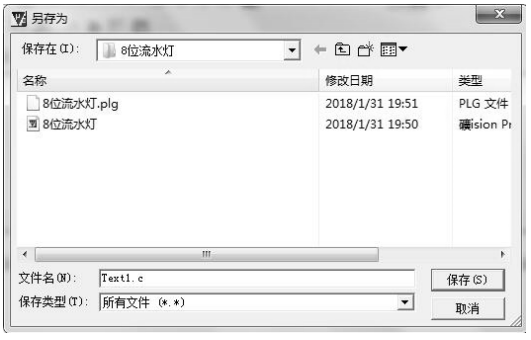


图 1-1-17　保存 C 文件

（10）选择“ 目标 1 ”下面的“ 源代码组 1 ”项，单击鼠标右键，弹出快捷菜单，选择“ 添加文件到组‘ 源代码组 1 ’”，如图 1-1-18 所示。

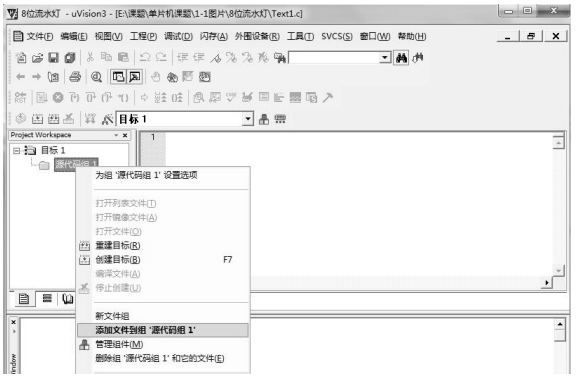


图 1-1-18　添加源代码步骤 1

（11）单击刚才编辑的“ Text1.c ”文件，再单击“ Add ”按钮，最后单击“ Close ”按钮，如图 1-1-19 所示，即可把 Text1.c 文件加入源代码组。

图 1-1-19　添加源代码步骤 2

（12）在编辑窗口输入源代码，接着单击菜单“ 工程 ”→“ 创建目标 ”，即可进行编译与连接，并将过程记录在下方的输出窗口，如图 1-1-20 所示。“ 0 个错误，0 个警告 ”表示没有错误。

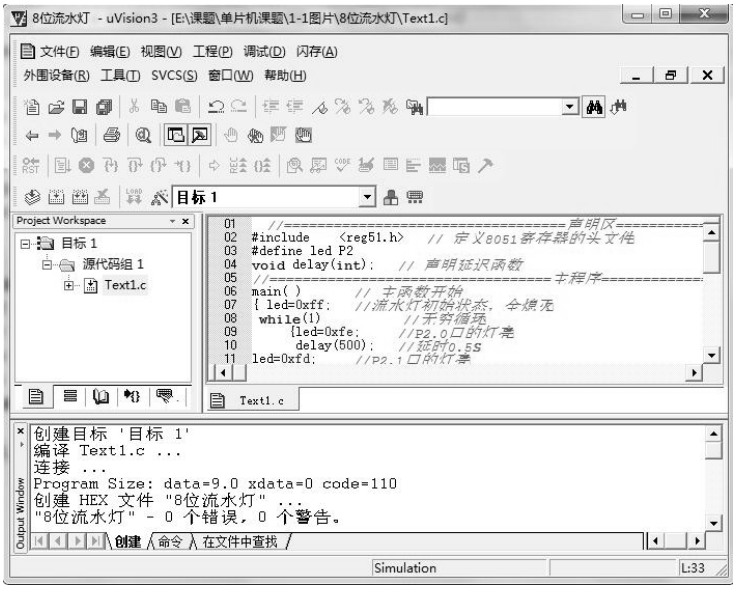


图 1-1-20　编译与连接

5．调试程序

（1）选择菜单栏中的“ 调试 ”→“ 启动/停止调试 ”命令，进入程序调试状态。选择菜单栏中的“ 外围设备 ”→“ I/O-Ports ”→“ Port 2 ”命令，弹出 Parallel Port 2 小窗口，当前 P2 口的值为 0xFF，如图 1-1-21 所示。

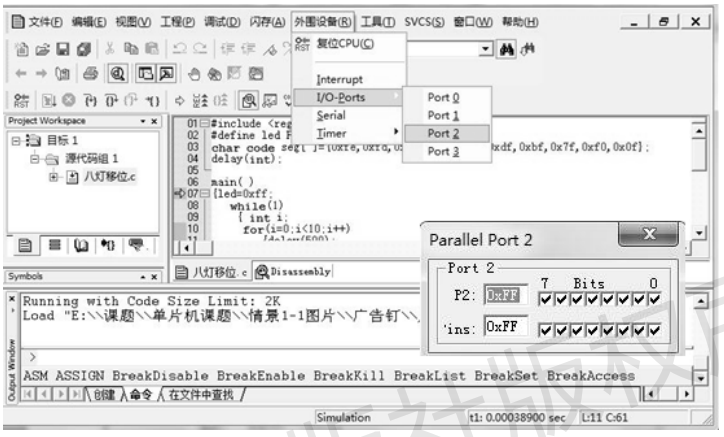


图 1-1-21　程序调试状态

（2）采用“ 单步调试 ”方式调试。

单击“ 调试 ”→“ 单步 ”，或者按快捷键 F10，光标停在程序第 7 行，P2 口的值为 0xFF，如图 1-1-22 所示。

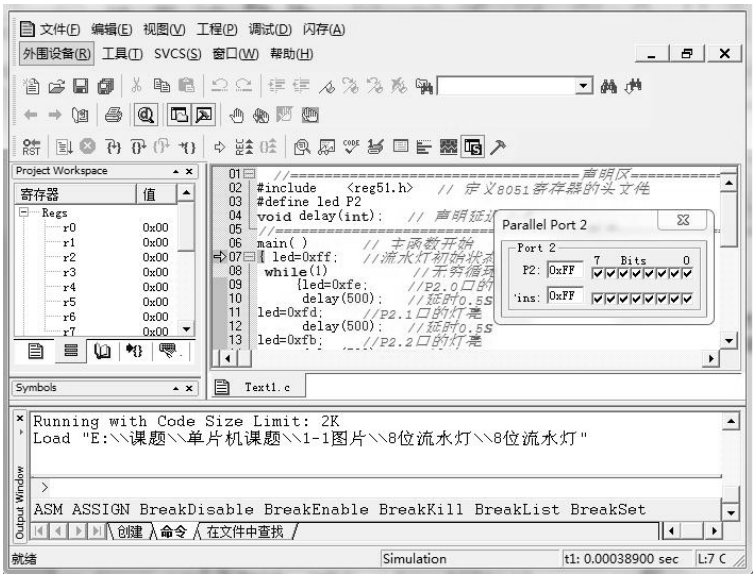


图 1-1-22　单步调试状态 1

然后再按 F10 键三次, 黄色箭头走到程序的第 10 行, P2 口的值为 0xFE, 如图 1-1-23 所示。依此方法可以完成其他代码的调试。

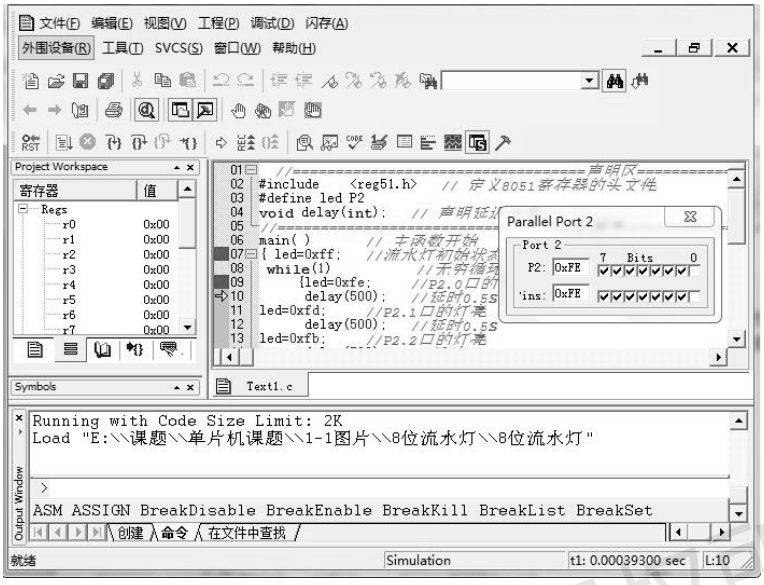


图 1-1-23　单步调试状态 2

（3）采用“ 断点 ”方式调试。

程序运行至第 7 行，但是想在第 15 行设置一个断点，可以用鼠标双击该行或者单击工具栏中的“ 调试 ”→“ 插入/删除断点 ”，再单击“ 调试 ”→“ 全速运行 ”按钮。还可以采用“ 运行到光标处 ”的方法，即把光标放在第 15 行，然后单击“ 调试 ”→“ 运行到光标处 ”按钮，如图 1-1-24 所示。运行后，黄色箭头走到程序的第 15 行，如图 1-1-25

所示。P2 口的值为 0xFB。断点调试方法可以越过某一段程序，提高调试效率。

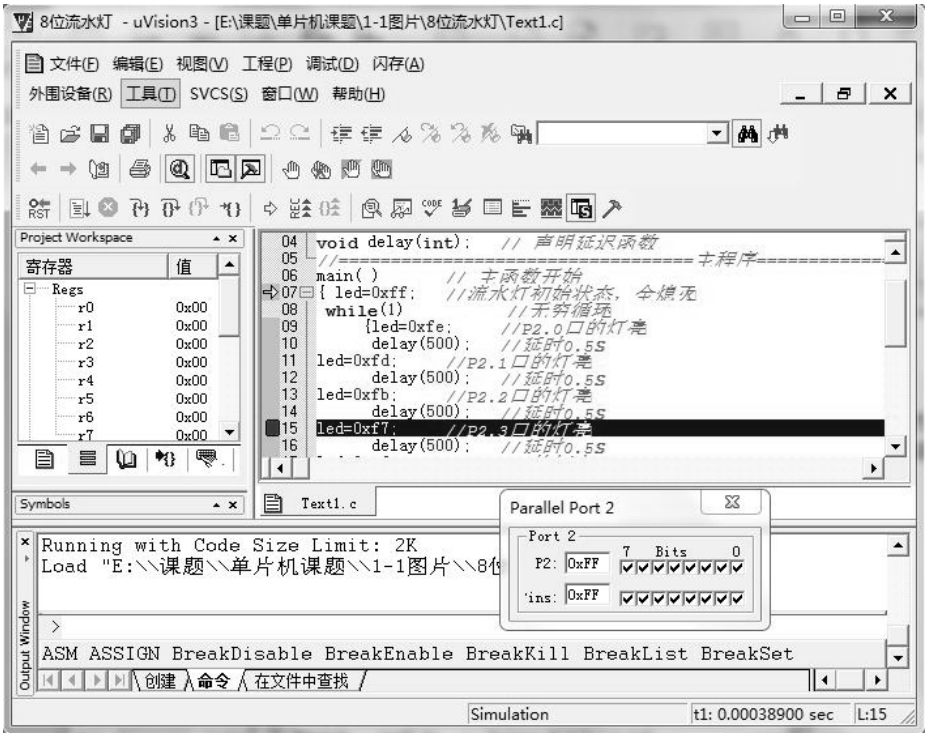


图 1-1-24　断点调试状态 1

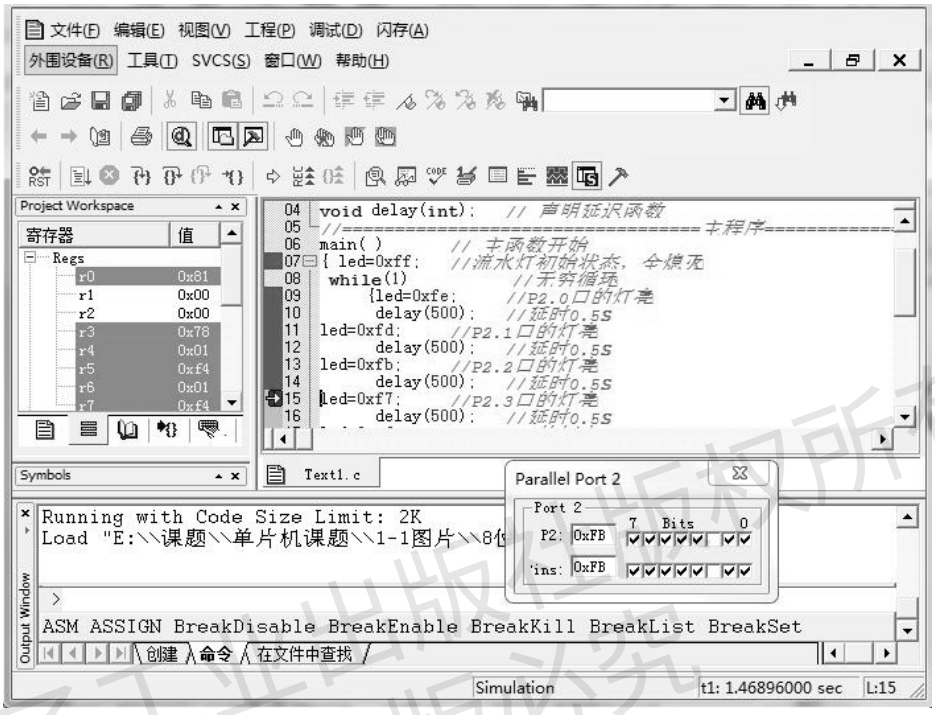


图 1-1-25　断点调试状态 2

6．在 Proteus 软件中仿真程序

（1）在 Proteus 软件中打开绘制好的“ 8 位流水灯电路原理图 ”，双击单片机，弹出如图 1-1-26 所示的对话框，单击，添加相应的文件，再单击“ OK ”按钮。

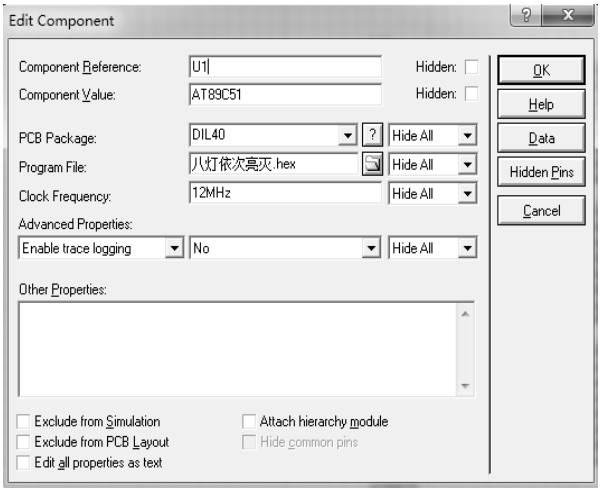


图 1-1-26　添加文件

（2）在 Proteus 软件界面上，单击仿真按钮，即可看到 8 位流水灯依次点亮的仿真效果，如图 1-1-27 所示。

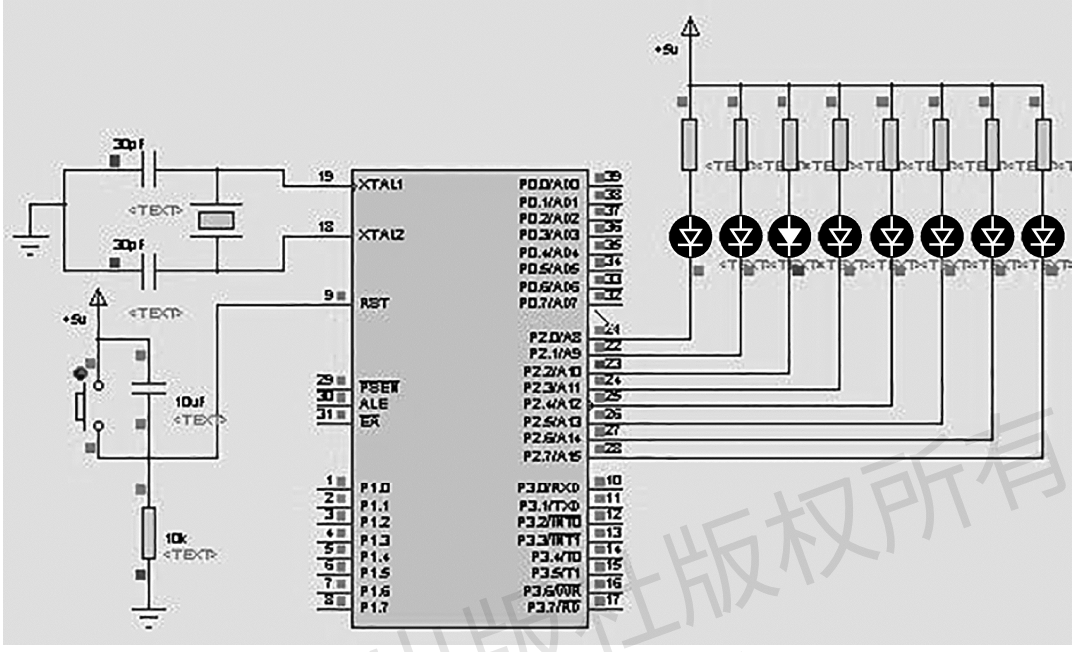


图 1-1-27　8 位流水灯依次点亮仿真效果图

7．在开发板上实现流水灯亮灭

该开发板采用 STC12C5A60S2 单片机，该单片机和 AT89C51 单片机的工作原理和

编程方法一致。但是 STC12C5A60S2 单片机的运行速度比 AT89C51 单片机快 6 倍。所以编程时，只要把延时时间加长，即可实现异曲同工的效果。下载的具体步骤如下。

（1）把单片机放入 40DIP 插座中，并卡住。然后用排线把单片机的 P2 脚与发光二极管相连。最后用 USB 线把开发板和 PC 连接在一起，按下电源开关，观察电源指示灯是否亮。若亮说明开发板与 PC 连接正常，可以工作。

（2）双击 PC 桌面上的“ STC-ISP-V488 ”图标，启动 STC 单片机程序下载软件，如图 1-1-28 所示。在“ MCU Type ”框中选择 STC12C5A60S2 单片机，单击“ 打开程序文件 ”，添加相应的 HEX 文件，选择对应的 COM 口，本书实例中的计算机分配 COM3 口。

图 1-1-28　STC 程序下载软件界面

（3）关闭开发板电源，单击“ Download/下载 ”，稍等片刻，按下电源按钮，等待下载完毕，在信息栏中可以看见程序的下载过程。

（4）下载完成后，即可看见开发板上的 8 个发光二极管依次亮灭，实现了该任务的要求，如图 1-1-29 所示。

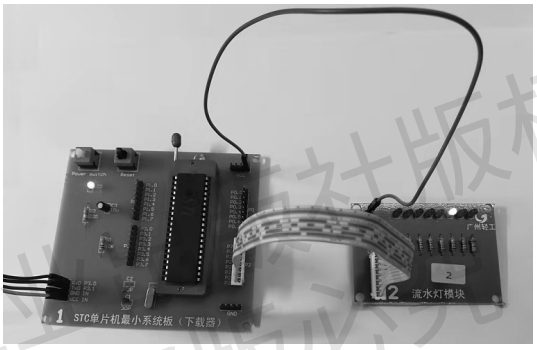


图 1-1-29　开发板上 8 位流水灯效果

知识点提升

一、移位法实现 8 位流水灯的灯饰效果

上述任务的程序设计采取的方法是传送法，原理是利用二极管的单向导电性，直接给 P2 口送一个数据，从而控制发光二极管的亮灭。如果有 8 个发光二极管，就编写 8 小段程序。如果有较多的发光二极管，则该编程方法不够科学。因此，编者设计了另一种方法：移位法。程序如下：

```
//==声明区=============================================================
#include <reg51.h>                    //定义8051寄存器的头文件，P2-17～19
#define  LED P2                       //定义LED接至P2
delay(int);                           //声明延迟函数
//============================主程序=================================
main()                                //主程序开始
{int  i;                              //声明整型数据i
LED=0xfe;                             //初值=11111110，只有最右1灯亮
while(1)                              //无限循环，程序一直运行
{for(i=0;i<7;i++)                     //左移7次
{ delay(500);                         //延迟0.5s
LED=(LED<<1)|0x01;                    //左移1位，并设定最低位元为1
}                                     //左移结束，只有最左1灯亮
for(i=0;i<7;i++)                      //右移7次
{delay(500);                          //延迟0.5s
LED=(LED>>1)|0x80;                    //右移1位，并设定最高位元为1
}                                     //结束右移，只有最右1灯亮
}                                     //while循环结束
}                                     //主程序结束
//============================延时1ms子程序=========================
delay(int x)                          //延时函数开始
{int i, j;                            //声明整型变量i，j
for(i=0;i<x;i++)                      //计数x次，延迟x×1ms
for(j=0;j<120;j++);                   //计数120次，延迟1ms
}
```

从程序设计上可以看出，本实例采取循环移位法，首先左移 7 次，再右移 7 次，如此不断循环。左移采用“ LED<<1 ”指令，右移采用“ LED>>1 ”指令。对于计数循环方式，采用 for 语句即可达到目的。

LED 的初始值为 11111110，左移时，右边将移入 0，变成 11111100，所以，必须将最右边的位变为 1。我们在左移后再利用 OR 运算，即“ LED=（LED<<1）|0x01 ”指令，可将 11111100 变成 11111101。在进行右移时，可应用“ LED=（LED>>1）|0x80 ”。

二、查表法实现 8 位流水灯的灯饰效果

除了传送法、移位法，编者还设计了查表法供读者参考，程序如下：

```
-------------------------------------------------------------------------
       #include <reg51.h>              //定义寄存器头文件
       #define led P2                  //定义led接P2口
       char code seg[ ]={0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f};//流
水灯循环亮灭的数据码
       delay(int);                     //声明延时函数
       //============================主程序================================
       main( )                         //主函数开始
       { while(1)                      //无限循环
       { int i;                        //声明整型变量i
       for(i=0;i<8;i++)                //循环8次
       {delay(500);                    //延时0.5s
       led=seg[i];                     //取表格中码送至led
       }
       }                               //无限循环结束
       }                               //主函数结束
       //============================延时1ms子程序=========================
       delay(int x)                    //延时函数开始
       {int i, j;                      //声明整型变量i, j
       for(i=0;i<x;i++)                //计数x次, 延迟x×1ms
       for(j=0;j<120;j++);             //计数120次, 延迟1ms
       }
-------------------------------------------------------------------------
```

从程序可看出，如果编写各种各样的代码，流水灯可以实现丰富多彩的流水效果，如从左到右依次亮、从右到左依次亮、从两边向中间亮、从中间向两边亮、闪烁两次等。这留给读者慢慢思考。

知识点链接

一、单片机的定义

微型计算机系统包括中央处理单元、存储器及输入/输出单元三大部分。中央处理器就像人体的大脑，控制整个系统的运行；存储器存放系统运行所需的数据及程序，包括数据存储器和程序存储器；输入/输出单元是计算机系统与外部沟通的管道。这三部分分别由不同的元件组成，然后把它们组装在电路板上，形成一个微型计算机系统。

单片机与微型计算机有相似之处，它是把中央处理器、数据存储器、程序存储器、定时/计数器以及输入/输出接口电路等主要功能部件集成在一块芯片上的微型计算机。

单片机具有结构简单、控制功能强、可靠性高、性价比高等特点，被广泛应用于工业控制、智能仪器仪表、家用电器、电子玩具等领域。

二、单片机的分类

单片机生产厂商多、型号种类多，但是都以传统的 8051 单片机作为内核。

1．Intel 公司的单片机

30 年前，Intel 公司推出了 MCS-51 系列单片机，它的基本芯片是 8031、8051 和 8751，后来又推出了低功耗单片机，如 80C31、80C51、80C52 等，虽然型号不同，但是都是 8051 单片机的派生产品。现在这些单片机虽然停产了，但是 Intel 公司的 8051 是单片机领域的奠基石，成为后续单片机发展良好的技术平台。

2．Atmel 公司的单片机

Atmel 公司推出的 MCS-51 单片机在市场上占有一定的比例，它提供了丰富的外围接口和内部资源，常用的型号有 AT89C51、AT89C52、AT89C51、AT89S52，同样也是 8051 的派生品。其中 AT89S 系列单片机具有系统编程 ISP 功能，无需专用的仿真器或编程器，只要通过 ISP 下载线和软件，就可以把程序下载到单片机中，在嵌入式控制领域得到了广泛的应用。

3．STC 公司的单片机

STC 公司是中国本土 MCU 领航者，生产了 STC10、STC11、STC12C5A 等系列单片机，它们是高速、低功耗、超强抗干扰的新一代 8051 单片机，其指令完全兼容传统的 8051 单片机，而且速度要快 8～12 倍。采用串口下载程序，其内部资源与 Atmel 公司的单片机差不多。

三、单片机的内部结构

单片机发展至今，虽然有许多厂商各自开发了不同的兼容芯片，但其基本结构并没有多大变化，图 1-1-30 为 8051 单片机的内部结构。

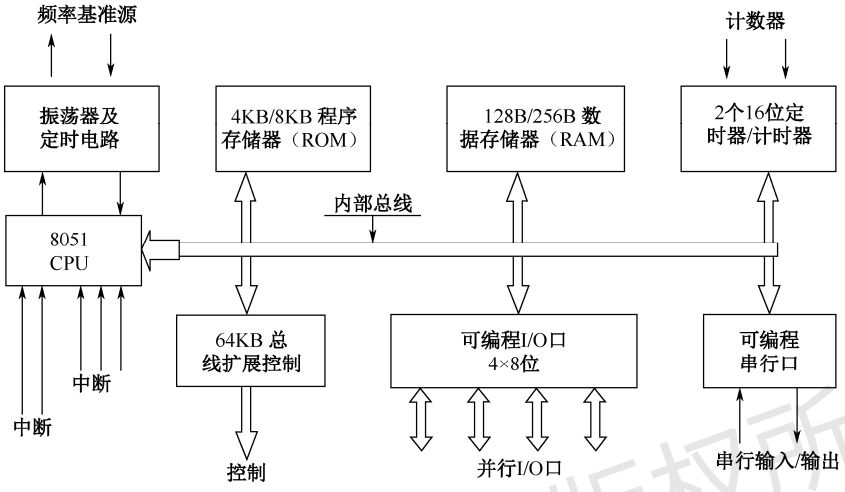


图 1-1-30　8051 单片机内部结构

1．中央处理器（CPU）

中央处理器由运算器和控制器组成，完成数据运算和控制功能。其中，运算器包括 8 位算术逻辑单元（ALU）、8 位累加器（ACC）、8 位暂存器、寄存器和程序状态寄存器，控制器包括指令寄存器（IR）、程序计数器（PC）、指令译码器（ID）等。

2．存储器

程序存储器（ROM）：内部 4KB，外部最多可扩展至 64KB。

数据存储器（RAM）：内部 128B，外部最多可扩展至 64KB。

3．I/O 端口

4 个 8 位双向 I/O 端口，即 P0、P1、P2 和 P3。

4．全双工串行通信接口

该接口可以实现单片机与单片机之间或者单片机与计算机之间的数据通信。

5．定时器/计数器与中断

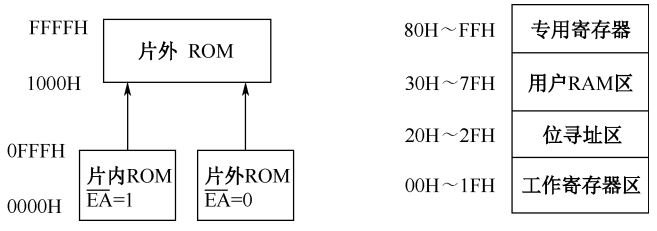
8051 单片机内部有两个 16 位定时器/计数器（T0 和 T1），可实现定时或计数的功能。还有 5 个中断源，即 INT0、INT1、T0、T1 和 RXD/TXD，可以进行中断源高、低优先级设置。

6．振荡电路

在单片机的 XTAL1 和 XTAL2 两端外接石英晶体即可产生一定频率的时钟信号。

四、单片机的存储器结构

存储器主要包括片内数据存储器、片外数据存储器、片内程序存储器和片外程序存储器。其中，程序存储器和数据存储器是独立编址的。不同单片机的片内存储器大小不尽相同，但它们的结构较为相似，单片机存储器结构如图 1-1-31 所示。



（a）片内 ROM　　（b）片内 RAM

图 1-1-31　单片机存储器结构

1．程序存储器

单片机程序存储器用于存放编译器编译出的二进制程序代码和程序执行过程中不会改变的原始数据，8051 内核单片机的片内 ROM 大小不相同，如 AT89C51 片内有 4KB 的 ROM，AT89S52 片内有 8KB 的 ROM，ST12C5A6052 片内有 60KB 的 ROM。由于当前单片机的内部 ROM 容量很大，一般不需要外扩 ROM。

AT89C51 片内外的 ROM 是统一编址的，如图 1-1-31（a）所示。若单片机 EA 引脚为高电平，则执行片内 ROM 中的程序（地址范围 0000H～0FFFH，即 4KB 地址）；如果片外加有 ROM（地址范围 1000H～FFFFH），则 CPU 执行完内部的 ROM 指令，就会自动执行片外的 ROM 指令。若 EA 引脚为低电平，则只能从片外 ROM 开始执行。

程序存储器中有 6 个特殊的地址，见表 1-1-2。相邻中断源入口地址的间隔为 8 个单元。在汇编语言中，当程序中断时，一般在这些入口地址处编写一条跳转指令，而相应

的中断服务程序编写在转移地址中；如果没有用到相应的中断功能，这些特殊地址单元也可作为一般程序存储器用于存放程序代码。在 C 语言中，Cx51 编译器会自动跳转到相应的中断入口地址，用户只要写好中断服务程序，其他事情由编译器完成。

表 1-1-2　程序存储器中的特殊地址

入口地址	用途说明
0000H	系统复位，PC=0000H，表示单片机从 0000H 单元开始执行程序
0003H	外部中断 0 中断时，PC=0003H，进入外部中断 0 中断服务程序
000BH	定时器/计数器 0 中断时，PC=000BH，进入定时器/计数器 0 中断服务程序
0013H	外部中断 1 中断时，PC=0013H，进入外部中断 1 中断服务程序
001BH	定时器/计数器 1 中断时，PC=001BH，进入定时器/计数器 1 中断服务程序
0023H	串口中断时，PC=0023H，进入串口中断服务程序

2．数据存储器

AT89C51 单片机片内数据存储器共有 256 字节（单元），分成两个部分：低 128 字节（地址 00H～7FH）和高 128 字节（地址 80H～FFH）。在这一区间的数据存储器又可分为 4 个部分，如图 1-1-31（b）所示。

（1）工作寄存器区

一般 8951 内核单片机有 4 个工作寄存器区，占据内部 RAM 的 00H～1FH，共计 32 个存储单元，可存放 32 字节的数据。具体配置见表 1-1-3。

表 1-1-3　寄存器组的选择

RS1	RS0	寄存器组	地　址
0	0	第 0 组	00H～07H
0	1	第 1 组	08H～0FH
1	0	第 2 组	10H～17H
1	1	第 3 组	18H～1FH

一般通过程序状态寄存器（PSW）中 RS1 和 RS0 位的组合状态来决定当前工作寄存器为 4 组中的哪一组。在 C 语言编程过程中，一般不会直接使用工作寄存器组，但是在汇编语言和 C 语言混合编程时，工作寄存器组是汇编子程序和 C 语言函数之间重要的数据传递工具。

（2）位寻址区

片内 RAM 的 20H～2FH 单元为位寻址区，共 16 个单元、128 位，位寻址区既可进行字节操作，又可以对单元中的每一位进行位操作。

（3）用户 RAM 区

片内 RAM 的 30H～7FH 单元为用户 RAM 区，共 80 个单元，编程者可以用它来存放数据，一般应用中常把堆栈开辟在此区中。

（4）专用寄存器区

片内 RAM 的高 128 字节地址 80H～FFH 为专用寄存器区，存放的是特殊功能寄存器的地址。以汇编语言编写程序时，必须熟练掌握这些寄存器。若以 C 语言编写程序，

它们就不是那么重要了。在程序的声明区放置 Keil C 所提供的“ reg51.h ”头文件，只要把它包含到程序里即可，而不必记忆这些位置。本书采用 C 语言编程，故不一一介绍特殊功能寄存器。

五、Keil C 语言的基本结构

一般来说，C 语言的程序可看成由一些函数所构成，其中的主程序是以“ main ”开始的函数，而每个函数可视为独立的个体，就像个模块一样，所以 C 语言是一种模块化的程序语言。C 程序的基本结构如图 1-1-32 所示，其中各项说明如下。

1．指定头文件

头文件是一种预先定义好的基本数据。C51 程序里，头文件是定义 51 单片机内部寄存器地址的数据。指定头文件的方式有两种。

第一种方式：#include〈头文件文件名〉。若采用这种方式，编译程序将从 Keil μVision3 的头文件夹中查找所指定的头文件。

第二种方式：#include“ 包含头文件文件名 ”。若采用这种方式，编译程序将从源程序所在文件夹中查找所指定的头文件。

2．声明区

在指定头文件之后，可声明程序之中所使用的常数、变量、函数等，其作用域将扩展至整个程序，包括主程序与所有函数。在此建议，若程序中用到函数，则可在此先声明所有用到的函数。这样，函数放置的先后顺序将不会受到影响。换言之，函数放置在引用该函数的程序之前或之后都可以。若没有在此声明函数，则在使用函数之前必须先定义该函数。

3．主程序

主程序（主函数）以 main()开头，整个内容放置在一对大括号，即{ }里，分为声明区与程序区。在声明区里所声明的常数、变量等仅适用于主程序，而不影响其他函数。若在主程序之中使用了某变量，但在之前的声明区中没有声明，也可在主程序的声明区中声明。另外，程序区就是以语句所构成的程序内容。

4．函数定义

函数是一种具有独立功能的程序，其结构与主程序类似。可将所要处理的数据传入函数中，称为形式参数；也可将函数处理完成后的结果返回给调用它的程序，称为返回值。不管是形式参数还是返回值，在定义函数的第一行都应该交代清楚。其格式如下：

```
返回值数据类型　函数名称（数据类型 形式参数）
```

例如，要将一个无符号字符（unsigned char）实参传递给函数，函数执行完成时要返回一个整型（int）数据，此函数的名称为 My_func，则函数定义为

```
int My_func (unsigned char x)
```

若不要传入函数，则可在小括号内指定为 void。同样，若不要返回值，则可在函数名称左边指定为 void 或不指定。另外，函数的起始符号、结束符号、声明区及程序区都与主程序一样。在一个 C 语言的程序里可使用多个函数，并且函数中也可以调用函数。

5．注释

注释就是说明，属于编译器不处理的部分。C 语言的注释以“ /* ”开始，以“ */ ”结束，放置注释的位置可接续于语句完成之后，也可独立于一行。其中的文字，可使用中文。另外，也可以输入“ // ”，其右边整行都是注释。

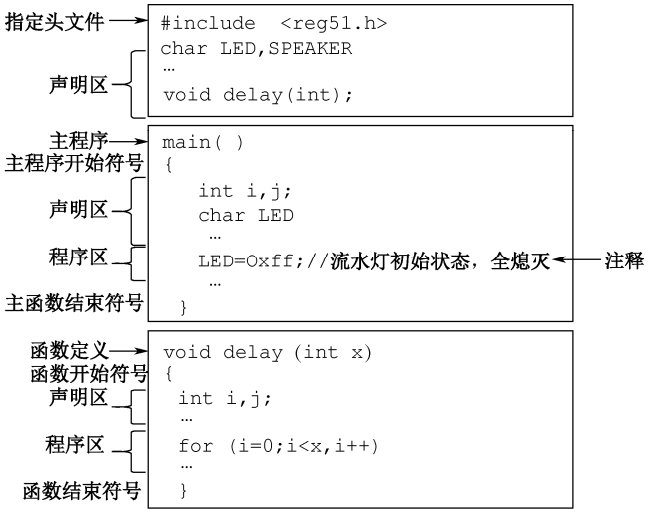


图 1-1-32　C 语言的基本结构

六、数据类型

数据是计算机处理的对象，任何程序设计都要进行数据处理。C 语言中包括字符（char）、整型（int）、浮点数（float）、空类型（void）、位类型（bit）、可寻址位（sbit）和特殊功能寄存器（sfr）几大类数据，具体见表 1-1-4。

表 1-1-4　C 语言的数据类型

名　　称	数 据 类 型	长　　度	范　　围
char	有符号字符	8	-128～+127
unsigned　char	无符号字符	8	0～255
int	有符号整型	16	-32768～32767
unsigned int	无符号整型	16	0～65535
short	短整型	16	-32768～32767
unsigned short	无符号短整型	16	0～65535
long	长整型	32	-2147483648～2147483647
unsigned long	无符号长整型	32	0～4294967295
float	单精度浮点数	32	-
double	双精度浮点数	64	-
void	空	0	无
bit	位类型	1	0 或 1
sbit	可寻址位	1	0 或 1
sfr	特殊功能寄存器	1	0～255

七、常量

在程序运行的过程中，其值不能改变的量，称为常量。其数据类型分为整型、浮点型、字符型、位类型和字符串型。常量的特点如下。

（1）整型常量可以表示为十进制数、十六进制数或八进制数等，如十进制数 10、-40 等。十六进制数以 0x 开头，如 0x13、0xAB 等；八进制数以字母 o 开头，如 o13、o27 等。

（2）浮点型常量可以分为十进制和指数两种表示形式，如 0.456、5895.568、234e4 等。

（3）字符型常量是用单引号括起来的单一字符，如'A'、'5'。

（4）字符串型常量是用双引号括起来的一串字符，如"51"、"hello"等。

（5）位类型的值是一个二进制数，即 0 或 1。

根据上述常量特点可知，常量可以是数值常量，也可以是符号常量。数值常量可以在程序中直接引用，如 a=15、a=2.65、a='c'等；但是符号常量不能直接使用，在使用之前必须用编译预处理命令“ #define ”先进行定义，例如：

```
#define PI  3.14
```

在此语句之后的程序中，PI 字符常量的值都为 3.14，这样便于程序修改。一般把程序中多处出现的同一常量用字符常量来代替，若要修改常量的值，可在预定义的时候修改。

八、变量

在程序运行中，其值可以改变的量称为变量。一个变量主要由两部分构成，一部分是变量名，另一部分是变量值。每个变量都在内存中占据一定的存储单元（地址），并在该内存单元中存放该变量的值。

1．变量的定义

变量必须先定义后使用，用标识符作为变量名，并指出所用的数据类型和存储模式，这样编译系统才能为变量分配相应的存储空间。变量的定义格式如下：

```
［存储种类］ 数据类型 “ 存储类型 ” 变量名表；
```

其中，数据类型和变量名表是必需的；存储种类和存储类型是可选项，一般忽略。例如：

```
-------------------------------------------------------------------------
    int A, b;              //定义a为整型变量
    float x, y, z          //定义x、y、z为单精度实型变量
    char seg               //定义seg为字符变量
    long int t             //定义t为长整型变量
-------------------------------------------------------------------------
```

进行变量定义时，应注意以下几点。

（1）允许在一个数据类型标识符后，声明多个相同类型的变量，各变量名之间用逗号隔开。

（2）数据类型标识符与变量名之间至少用一个空格隔开。

（3）最后一个变量名必须以“ ;”结尾。

（4）变量声明必须放在变量使用之前，一般放在函数体的开头部分。

（5）在同一个程序中变量不允许重复定义。

例如：

```
-------------------------------------------------------------------------
      int x,  y,  z；
      int a,  b, x；          //变量x被重复定义
-------------------------------------------------------------------------
```

2．变量的初始化

在定义变量的同时可以给变量赋初值，称为变量的初始化。变量初始化的一般格式为：

```
    数据类型标识符 变量名1=常量1，变量名2=常量2，…，变量名n=常量n；
```

例如：

```
-------------------------------------------------------------------------
      int m=3, n=5           //定义m和n为整型变量，同时分别赋初值3和5
      float x=0, y=0, z=0；  //定义x、y、z为单精度实型变量，同时都赋初值0
      char ch=‘a’；          //定义ch为字符型变量，同时赋初值字符‘a’
-------------------------------------------------------------------------
```

3．变量存储类型

单片机的存储器可以分为程序存储器（ROM）和数据存储器（RAM），它们又可以分为片内 ROM、片外 ROM、片内 RAM 和片外 RAM 四种物理存储空间。C51 编译器支持这四种物理存储空间，见表 1-1-5。

表 1-1-5　存储器形式

存 储 类 型	描　　述	适 用 范 围
code	程序存储器	0x0000～0xffff(64KB)
data	直接寻址的内部数据存储器	0x00～0x7f(128B)
idata	间接寻址的内部数据存储器	0x80～0xff(128B)
bdata	位寻址的内部数据存储器	0x20～0x2f(16B)
xdata	以 DATA 寻址的外部数据存储器	64KB 之内
pdata	以 R0、R1 寻址的外部数据存储器	256B 之内

C51 内部的 4KB 程序存储器可扩展至 64KB，程序存储器可以存放程序代码，也可以存放固定的数据，如七段数码显示器的显示码、LED 点阵的显示码、LCM 的显示字符串，以下就是七段数码显示器的显示码。

```
-------------------------------------------------------------------------
      char code seg[ ]={0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80,
0x98};
-------------------------------------------------------------------------
```

4．变量的作用范围

变量的作用范围或有效范围与该变量在哪里声明有关，大致可分为两种，说明如下。

（1）全局变量

在程序开头的声明区或没有大括号限制的声明区所声明的变量，其作用范围为整个程序，称为全局变量，如图 1-1-33 所示。其中的 LED、SPEAKER 就是全局变量。

（2）局部变量

在大括号内的声明区所声明的变量，其作用范围将受限于大括号，称为局部变量，图 1-1-34 中的 i、j 就是局部变量。若在主程序与各函数之中都声明了相同名称的变量，则脱离主程序或函数时，该变量将自动无效，又称自动变量。

如图 1-1-34 所示，在主程序与 delay 子程序中各自声明了 i、j 变量，但主程序中的 i、j 与 delay 子程序中的 i、j 是各自独立的。

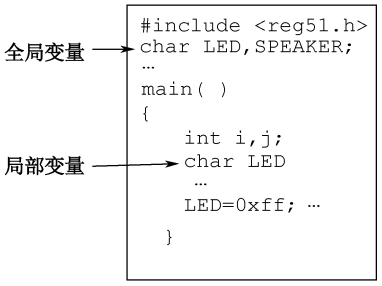


图 1-1-33　全局变量与局部变量

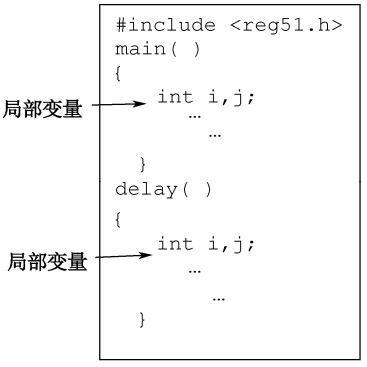


图 1-1-34　局部变量

九、Keil C 的运算符

运算符就是程序语句中的操作符号，Keil C 的运算符可分为以下几种。

1．算术运算符

算术运算符是执行算术运算功能的操作符号，有加、减、乘、除、取余数、自增和自减运算符（表 1-1-6）。

表 1-1-6　算术运算符

符　号	功　能	范　例	说　明
+	加	A=x+y	将 x 与 y 变量的值相加，其和放入 A 变量
-	减	B=x-y	将 x 变量的值减去 y 变量的值，其差放入 B 变量
*	乘	C=x*y	将 x 与 y 变量的值相乘，其积放入 C 变量
/	除	D=x/y	将 x 变量的值除以 y 变量的值，其商放入 D 变量
%	取余数	E=x%y	将 x 变量的值除以 y 变量的值，其余数放入 E 变量
++	加 1	x++	执行运算后将 x 变量的值加 1
--	减 1	y--	执行运算后将 y 变量的值减 1

程序范例：

```
-------------------------------------------------------------------------
      main ( )
      {  int A, B, C, D,  E, x, y;
      x=7；y=2；
      A=x+y； B=x-y； C=x*y； D=x/y； E=x%y；
```

```
    x++；y--；
    }
```

程序结果：A=0x09、B=0x05、C=0x0E、D=0x03、E=0x01、x=0x08、y=0x01。

2．关系运算符

关系运算符用于处理两变量间的大小关系，见表 1-1-7。

表 1-1-7　关系运算符

符　号	功　能	范　例	说　明
==	相等	x==y	将 x 与 y 变量的值进行比较，相等则结果为 1，否则为 0
! =	不相等	x!=y	将 x 与 y 变量的值进行比较，不相等则结果为 1，否则为 0
>	大于	x>y	若 x 变量的值大于 y 变量的值，其结果为 1，否则为 0
<	小于	x<y	若 x 变量的值小于 y 变量的值，其结果为 1，否则为 0
>=	大于等于	x>=y	若 x 变量的值大于或等于 y 变量的值，其结果为 1，否则为 0
<=	小于等于	x<=y	若 x 变量的值小于或等于 y 变量的值，其结果为 1，否则为 0

程序范例：

```
    main ( )
    {  char  A, B, C, D,  E, F, x, y；
    x=7；y=2；
    A= (x==y) ； B= (x!=y)； C=(x>y)； D=(x<y)；
    E=(x>=y)；  F=(x<=y)；
    }
```

程序结果：A=0x00、B=0x01、C=0x01、D=0x00、E=0x01、F=0x00。

3．逻辑运算符

逻辑运算符就是执行逻辑运算功能的操作符号，逻辑运算包括与、或和反相运算，其结果为 1 或 0，见表 1-1-8。

表 1-1-8　逻辑运算符

符　号	功　能	范　例	说　明
&&	与运算	(x>y) && (y>z)	若 x 变量的值大于 y 变量的值，且 y 变量的值也大于 z 变量的值，其结果为 1，否则为 0
‖	或运算	(x>y) ‖ (y>z)	若 x 变量的值大于 y 变量的值，或 y 变量的值大于 z 变量的值，其结果为 1，否则为 0
!	反相运算	! x>y	若 x 变量的值大于 y 变量的值，其结果为 1，否则为 0

程序范例：

```
    main ( )
    {  char A, B, C, D,  E, F, x, y；
    x=7；y=2；z=5；
```

```
        A= (x>y) && (y<z) ;  B= (x==y) || (y<=z) ;  C=!(x>z) ;
        }
--------------------------------------------------------------------------------
```

程序结果：A=0x01、B=0x01、C=0x00。

4．布尔运算符

布尔运算符与逻辑运算符非常相似，两者最大的差异在于布尔运算符针对变量中的每一位，逻辑运算符则对整个变量进行操作。布尔运算符见表 1-1-9。

表 1-1-9　布尔运算符

符　号	功　能	范　例	说　明
&	与运算	x&y	将 x 与 y 变量的每一位进行与运算
|	或运算	x|y	将 x 与 y 变量的每一位进行或运算
^	异或运算	x^y	将 x 与 y 变量的每一位进行异或运算
～	取反运算	～x	将 x 变量的每一位进行取反运算
《	左移	x《n	将 x 变量的值左移 n 位
》	右移	x》n	将 x 变量的值右移 n 位

程序范例：

```
--------------------------------------------------------------------------------
        main ( )
        {   char A, B, C, D,  E, F, x, y;
        A=x&y;  B=x|y;  C=x^y; D=～x; E=x《3; F=x》4;
        }
--------------------------------------------------------------------------------
```

程序结果：A=0x21、B=0x77、C=0x56、D=0xdA、E=0x28、F=0x02。

5．赋值运算符

赋值运算符是一种很有效率且特殊的操作符号，包括最常见的“ = ”，还有将算术运算、逻辑运算变形的操作符号，见表 1-1-10。

表 1-1-10　赋值运算符

符　号	功　能	范　例	说　明
=	赋值	A=x	将 x 变量的值放入 A 变量，与 A=x 相等
+=	相加	B+=x	将 B 变量的值与 x 变量的值相加，其和放入 B 变量，与 B=B+x 相等
-=	相减	C-=x	将 C 变量的值与 x 变量的值相减，其差放入 C 变量，与 C=C-x 相等
*=	相乘	D*=x	将 D 变量的值与 x 变量的值相乘，其积放入 D 变量，与 D=D*x 相等
/=	相除	E/=x	将 E 变量的值与 x 变量的值相除，其商放入 E 变量，与 E=E/x 相等
%=	取余数	F%=x	将 F 变量的值与 x 变量的值相除，其余数放入 F 变量，与 F=F%x 相等
&=	与运算	G&=x	将 G 变量的值与 x 变量的值相与，其结果放入 G 变量，与 G=G&x 相等
|=	或运算	H|=x	将 H 变量的值与 x 变量的值相或，其结果放入 H 变量，与 H=H|x 相等
^=	异或运算	I^=x	将 I 变量的值与 x 变量的值相异或，其结果放入 I 变量，与 I=I^x 相等
《=	左移	J≪=n	将 J 变量的值左移 n 位，与 J=J≪n 相等
》=	右移	K≫=n	将 K 变量的值右移 n 位，与 K=K≫n 相等

程序范例：

```
-------------------------------------------------------------------------
      main ( )
      {   char A, B, C, D,  E, F, G,  H, I, J,  K, x, y；
      A=x；B+=x；C-=x；D*=x；E/=x；F%=x；G&=x；H|=x；I^=x；J《=n；K》=n；
      }
-------------------------------------------------------------------------
```

程序结果：A=0x96、B=0xd0、C=0x6B、D=0x4e、E=0x01、F=0x11、G=0x90、H=0x9f、I=0xC3、J=0xa0、K=0x0e。

十、Keil C 的循环指令

循环指令就是将程序流程控制在指定的循环里，直到符合指定的条件才脱离循环继续往下执行。Keil C 所提供的循环指令有 for 语句、while 语句、do-while 语句。

1．for 语句

for 语句是一个很实用的计数循环，其格式如下：

```
   for（表达式1；表达式2；表达式3）
   {语句组；    //循环体
   }
```

其中有 3 个表达式，说明如下。

表达式 1 为初始值。例如，从 0 开始则写成“ i=0；”，其中的 i 必须事先声明。“ ；”是分隔符，不可缺少。

表达式 2 为判断条件，以此为执行循环的条件。例如“ i<20；”，则只要 i<20 就继续执行循环。若此表达式空白，只输入“ ；”，如“ for（i=0；；i++）”或“ for（ ；；）”，则会无条件执行循环，不会跳出循环。

表达式 3 为条件运算方式，最常见的是自增或自减，如“ i++ ”或“ i-- ”；当然也可以是其他运算方式，如每次增加 2，即“ i+=2 ”。

范例 1：for（i=0；i<8；i++），说明循环执行 8 次。

范例 2：for（x=100；x>0；x--），说明循环执行 100 次。

范例 3：for（ ；；），说明是无限循环。

范例 4：for（num=0；num<99；num+=5），说明循环执行 20 次。

在 for 语句下面，可利用一对大括号将所要执行的指令逐行写入。例如，用 for 语句求 1～100 的累加和，代码如下：

```
-------------------------------------------------------------------------
      main( )
      {    int i;
      int sum=0;
      for (i=1；i<=100;i++)
      {sum=sum+i;
      }
      }
-------------------------------------------------------------------------
```

上述 for 语句的执行过程：先给 i 赋值 1，判断 i 是否小于等于 100，若是，则执行循环体“ sum=sum+i ”语句一次，然后 i 自增 1，再重新判断，直到 i=101 时，条件 i<=100 不成立，循环结束。

若循环中只执行一条指令，可不使用大括号。例如，要从 0 到 9，将 table 数组中的数据顺序输出到 P2，代码如下：

```
-------------------------------------------------------------------------
      for(i=0;i<8;i++)
      P2=table[i];
-------------------------------------------------------------------------
```

2．while 语句

while 语句的一般格式如下：

```
      while（表达式）
      {语句组；  //循环体
      }
```

while 语句循环原理：“ 表达式 ”通常是逻辑表达式或关系表达式，为循环条件；“ 语句组 ”是循环体，即被重复执行的程序段。该语句的执行过程如下：首先判断“ 表达式 ”的值，当值为真时，执行“ 语句组 ”；否则，就不执行。例如，求整数 1～100 的累加和，代码如下：

```
-------------------------------------------------------------------------
      main( )
      {    int i, sum；
      i=1；sum=0；
      while(i<=100)
      {sum=sum+i；
      i++；
      }
      }
-------------------------------------------------------------------------
```

变量 i 的取值范围为 1～100，所以初值为 1，while 语句的条件“ i<=100 ”，终值为 100，循环次数为 100。

while 语句使用过程中的注意事项如下：

（1）使用 while 语句时要注意，当表达式的值为真时，执行循环体，循环体执行一次完成后，再次回到 while，进行循环条件判断，如果仍然为真，则重复执行循环体程序；为假则退出整个 while 循环语句。

（2）如果循环条件一开始就为假，那么 while 后面的循环体一次都不会执行。

（3）如果循环条件总为真，如 while（1），表达式为常量 1，循环条件永远成立，则为无限循环，即死循环。在单片机 C 语言程序设计中，无限循环是一个非常有用的语句，在上述程序示例中都使用了该语句。

3．do-while 语句

while 语句是在执行循环体之前进行循环条件判断，如条件不成立，则该循环语句组

不执行。但是有时候需要先执行一次循环体后，再进行循环条件的判断。do-while 语句可以满足这种要求。do-while 语句的一般格式如下：

```
do
{语句组； //循环体
}while（表达式）；
```

do-while 语句循环原理：先执行循环体“ 语句组 ”一次，再计算“ 表达式 ”的值，如果“ 表达式 ”的值为真，则继续执行循环体“ 语句组 ”，直到表达式的值为假为止。

do-while 语句使用过程中的注意事项如下：

（1）在使用 if 语句、while 语句时，表达式括号后面都不能加分号，但在 do-while 表达式括号后必须加分号。

（2）do-while 语句与 while 语句相比，更适合处理不论条件是否成立，都需要先执行一次循环体的情况。

4．在循环体中使用 break 和 continue 语句

（1）break 语句

当 break 语句用于 while、do-while 和 for 循环语句时，不论循环条件是否满足，都可以使程序立即终止整个循环而执行后面的语句。通常 break 语句总是与 if 语句一起使用，即满足 if 语句的条件时便跳出循环。例如：

```
-------------------------------------------------------------------------
main( )
{ int i=0, sum, sum1;
sum=0;
for(i=0; ;i++)
{ if(i>10)  break;
sum=sum+i;
}
sum1=sum;
}
-------------------------------------------------------------------------
```

在上述程序中，如果 if 语句的条件成立，则运行 break 语句，程序就跳出 for 循环体，执行 sum1=sum 语句。

（2）continue 语句

continue 语句的作用是结束本次循环，强行执行下一次循环。它与 break 语句的不同之处在于：break 语句是直接结束整个循环，而 continue 语句是结束当前循环体的执行，再次进入循环条件判断，准备继续开始下一次循环体的执行。例如，求出 1～100 范围内所有不能被 5 整除的整数之和，代码如下：

```
-------------------------------------------------------------------------
main( )
{  int i, sum;
sum=0;
for(i=1;i<=100;i++)
{  if(i%5==0)   continue;
```

```
        sum=sum+i;
        }
        }
-------------------------------------------------------------------------------
```

程序分析：设置了一个 for 循环语句并进行 if 语句判断，若 i 对 5 取余运算的结果为 0，即能被 5 整除，则执行 continue 语句，退出本次循环；若不成立，则跳过 continue 语句，执行 sum=sum+i。再重新进行 for 循环条件判断。

十一、Keil C 的选择语句

选择指令是按条件决定程序流程。Keil C 所提供的选择指令有 if-else 语句及 switch-case 语句。

1．if-else 语句

if-else 语句提供条件判断的语句，称为条件选择，其格式如下：

```
    if(表达式)
    { 循环体1；
    …
    }
    else
    { 循环体2；
    …
    }
```

在这个语句中，将先判断表达式是否成立。若成立，则执行循环体 1，否则执行循环体 2。其中 else 部分也可以省略，即

```
    if(表达式)  { 循环体1；}
    其他指令
```

if-else 语句也可利用 else if 指令串接为多重条件判断，其格式如下：

```
    if(表达式1)
    { 循环体1；}
    else if(表达式2)
    { 循环体2；}
    else if(表达式3)
    { 循环体 3；}
    else{ 循环体4；}
    …
```

在这种流程中，从表达式 1 开始判断，若表达式 1 成立，则表达式 2 和表达式 3 都没有作用。若表达式 1 不成立，而表达式 2 成立，则表达式 3 没有作用。

2．switch-case 语句

if 语句一般用于单一条件或分支数目较少的场合，如果使用 if 语句来编写超过 3 个分支的程序，就会降低程序的可读性。C 语言提供了一种用于多分支选择的 switch-case 语句，一般格式如下：

```
switch(表达式)
{ case(常数1)：
{ 循环体1；}
break;
case(常数2)：
{ 循环体2；}
break;
…
default
{ 循环体n；}
break；
}
```

在 switch-case 语句中，表达式的值决定流程，并没有优先级的问题。若没有一个路径的常数与表达式的值相同，程序将执行 default 路径下的循环体。注意，每个 case 语句块结束时必须有一个 break 指令，否则会继续执行下一个 case 循环体。

任务评估

任务评估见表 1-1-11。

表 1-1-11　任务评估表

评 价 项 目	评 价 标 准	得　分
硬件设计	能正确分析单片机最小系统的电路结构及各部分的功能　10 分	
	可以自主设计 8 位流水灯的硬件电路　10 分	
软件设计与调试	掌握发光二极管的工作原理　10 分	
	正确理解 Keil C 语言的基本结构、数据类型、常数、变量、运算符、循环指令与选择指令等知识点　20 分	
	能根据任务分析，正确绘制程序流程图　10 分	
	熟练使用 Proteus 和 Keil μVision3 软件，完成 8 位流水灯程序的设计与调试　20 分	
软硬件调试	能正确使用 STC-ISP-V488 程序下载软件，完成程序的下载，并根据接线图正确完成流水灯模块与最小系统模块的接线，观察 8 位流水灯的工作过程　10 分	
团队合作	各成员分工协作，积极参与　10 分	

学习任务二：海珠桥灯饰工程的设计与调试

任务描述

学生在 Proteus 仿真软件上自主完成海珠桥硬件设计，并且在 Keil μVision3 中编写丰富多彩的灯饰效果程序，最后在开发板上观看海珠桥灯饰效果。

任务目标

（1）能自主设计并用 Proteus 软件绘制海珠桥灯饰工程效果图。

（2）根据任务分析，正确绘制程序流程图。

（3）能用 Keil μVision3 软件编写及调试程序。

（4）能充分发挥想象力，设计丰富多彩的海珠桥循环灯饰代码。

（5）能按照原理图，正确完成单片机开发板各电子元器件的接线，观看海珠桥灯饰效果。

建议课时：6 课时

任务分析

单片机有 4 个 I/O 口，每个口都有 8 位，共有 32 位，可以接 32 个 LED。为了实现更好的灯饰效果，本任务采取每位并联两个 LED，共有 64 个 LED。同样利用各引脚输出电位的变化，控制 LED 的亮灭。输出电位为高电平时，LED 灭；输出电位为低电平时，LED 亮。编写不同的循环码，可以实现不同的循环灯饰效果，并且利用亮灭的时间间隔，控制 LED 的循环速度，让海珠桥成为珠江边一颗璀璨的明珠。

任务实施

一、硬件电路设计

1．硬件设计思路

为了设计更丰富的海珠桥灯饰效果，可以在单片机的每个输出口并联两个或者更多的 LED。海珠桥灯饰可分为桥拱、桥身两部分。读者可以根据兴趣爱好，设计不同的灯饰图案。

2．硬件电路原理图

根据任务分析及设计思路，画出海珠桥灯饰工程的电路原理图，如图 1-2-1 所示。因灯饰工程较大，故省略最小系统。

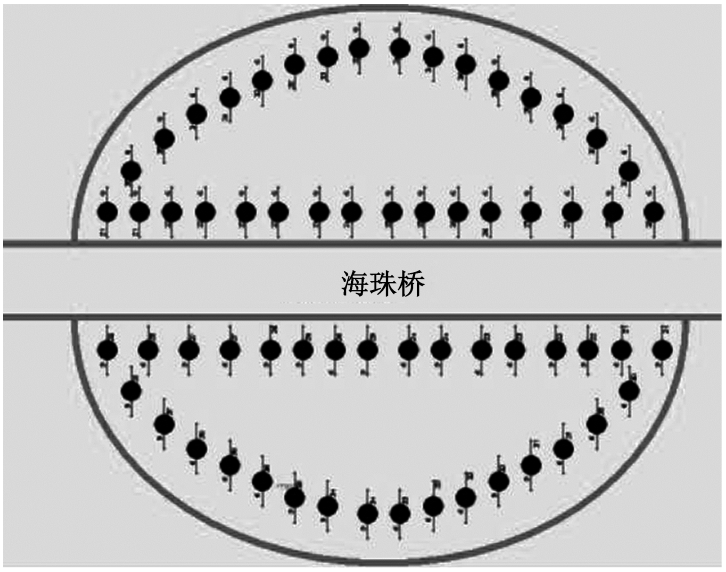


图 1-2-1　海珠桥灯饰工程电路原理图

根据电路原理图，确定本任务所需要的元器件清单，见表 1-2-1。

表 1-2-1　海珠桥灯饰元器件清单

序　　号	名　　称	型　　号	数量（个）
1	单片机	AT89C51	1
2	彩灯：LED	-	64
3	电阻：RES	100Ω	32
		10kΩ	1
4	电容：CAP	10μF	1
		22pF	2
5	晶体振荡器：CRYSTAL	12MHz	1
6	按钮：BUTTON	不带自锁	1

打开 Proteus 仿真软件，根据原理图及元器件清单，绘制海珠桥灯饰工程的电路原理图。绘制步骤在任务一中已详细说明，此任务不再介绍。

二、软件程序设计与调试

1．软件设计思路

单片机的 4 个 I/O 口控制了 64 个彩灯，可设计程序，让这 64 个彩灯依次点亮，形成 64 位的跑马灯。因此，我们的程序设计就要让 P0 口接的灯依次点亮，接着 P1 口接的灯依次点亮，接着 P2 口接的灯依次点亮，接着 P3 口接的灯依次点亮，以此类推，周而复始。

2．绘制程序流程图

有了设计思路后，可以将思路转换成流程图，如图 1-2-2 所示。

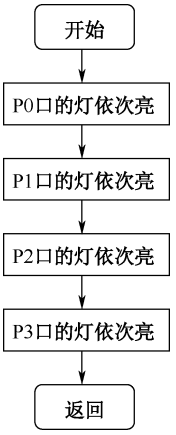


图 1-2-2　海珠桥灯饰工程流程图

3．编写程序

海珠桥灯饰工程的参考程序如下：

```
--------------------------------------------------------------------------------
      #include <reg51.h>
      #define led0 P0                  //定义led0接P0口
      #define led1 P1                  //定义led1接P1口
      #define led2 P2                  //定义led2接P2口
      #define led3 P3                  //定义led3接P3口
      char code seg[ ]={0xfe, 0xfd, 0xfb, 0xf7, 0xef,
      0xdf, 0xbf, 0x7f, 0xff};         //灯依次点亮的数据码
      delay(int);
      main( )
      { led0=0xff;                     //初始化，P0口的灯全灭
      led1=0xff;                       //初始化，P1口的灯全灭
      led2=0xff;                       //初始化，P2口的灯全灭
      led3=0xff;                       //初始化，P3口的灯全灭
      while(1)
      { int i;
      for(i=0;i<9;i++)
      {delay(200);
      led0=seg[i];                     //P0口的灯依次点亮
      }
      for(i=0;i<9;i++)
      {delay(200);
      led1=seg[i];                     //P1口的灯依次点亮
      }
      for(i=0;i<9;i++)
      {delay(200);
      led2=seg[i];                     //P2口的灯依次点亮
      }
      for(i=0;i<9;i++)
      {delay(200);
```

```
        led3=seg[i];     //P3口的灯依次点亮
        }
        }
        }
        delay(int x)              // 1ms延时程序
        { int i, j;
        for(i=0;i<x;i++)
        for(j=0;j<120;j++);
        }
```

4．编译程序

在编辑窗口输入源代码后，单击左上方的编译按钮即可进行编译与连接，输出窗口显示“ 0 个错误，0 个警告 ”，表明没有语法错误，可以调试。

5．调试仿真程序

在 Proteus 软件中进行仿真，把 HEX 文件加入单片机，单击仿真按钮，即可看到 64 个流水灯依次点亮，形成跑马灯的效果。

6．在开发板上实现的效果

把程序下载至单片机，单片机的 P0 口、P1 口、P2 口与 P3 口分别与流水灯相连，确认连线无误后通电，看见海珠桥的灯饰依次点亮，实现了任务的要求（图 1-2-3）。

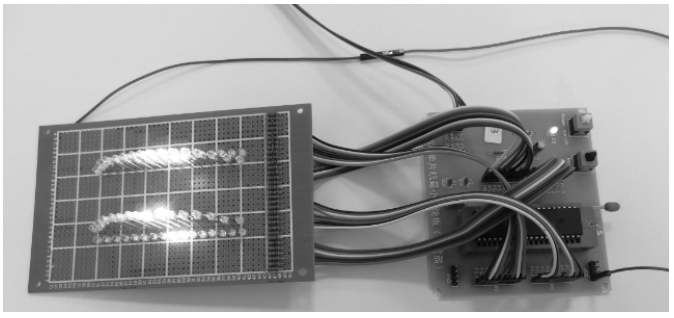


图 1-2-3　海珠桥灯饰循环效果图

知识点提升

上述任务的程序设计实现的灯饰效果只有一种，不够丰富多彩。因此，编者充分发挥想象力，设计了多种灯饰循环效果，供读者参考。程序如下：

```
        #include<reg51.h>
        #define led0 P0
        #define led1 P1
        #define led2 P2
        #define led3 P3
        delay(int);
```

```
        char code seg1[ ]=
        {0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f, 0x7f, 0xbf, 0xdf, 0xef,
0xf7, 0xfb, 0xfd, 0xfe,
        0xff, 0x00, 0xff, 0x00, 0xff, 0x00, 0xff, 0x00, 0xff, 0x00, 0xff, 0x00,
0xff, 0x00, 0xff, 0x00,
        0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
        0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
        0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f, 0x7f, 0xbf, 0xdf, 0xef,
0xf7, 0xfb, 0xfd, 0xfe,
        0xfe, 0xfc, 0xf8, 0xf0, 0xe0, 0xc0, 0x80, 0x00, 0x00, 0x80, 0xc0, 0xe0,
0xf0, 0xf8, 0xfc, 0xfe,
        0xfe, 0xfc, 0xf8, 0xf0, 0xe0, 0xc0, 0x80, 0x00, 0x00, 0x80, 0xc0, 0xe0,
0xf0, 0xf8, 0xfc, 0xfe,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
        0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55,
0xaa, 0x55, 0xaa, 0x55,
        0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55,
0xaa, 0x55, 0xaa, 0x55,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
        0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55,
0xaa, 0x55, 0xaa, 0x55, };
        char code seg2[]=
        {0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f, 0x7f, 0xbf, 0xdf, 0xef,
0xf7, 0xfb, 0xfd, 0xfe,
        0xff, 0x00, 0xff, 0x00, 0xff, 0x00, 0xff, 0x00, 0xff, 0x00, 0xff, 0x00,
0xff, 0x00, 0xff, 0x00,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x7f, 0xbf, 0xdf, 0xef,
0xf7, 0xfb, 0xfd, 0xfe,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xfd, 0xfb, 0xf7,
0xef, 0xdf, 0xbf, 0x7f,
        0x7f, 0xbf, 0xdf, 0xef, 0xf7, 0xfb, 0xfd, 0xfe, 0xfe, 0xfd, 0xfb, 0xf7,
0xef, 0xdf, 0xbf, 0x7f,
        0xfe, 0xfc, 0xf8, 0xf0, 0xe0, 0xc0, 0x80, 0x00, 0x00, 0x80, 0xc0, 0xe0,
```

```
0xf0, 0xf8, 0xfc, 0xfe,
        0xfe, 0xfc, 0xf8, 0xf0, 0xe0, 0xc0, 0x80, 0x00, 0x00, 0x80, 0xc0, 0xe0,
0xf0, 0xf8, 0xfc, 0xfe,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
        0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55,
0xaa, 0x55, 0xaa, 0x55,
        0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa,
0x55, 0xaa, 0x55, 0xaa,
        0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55,
0xaa, 0x55, 0xaa, 0x55,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, };
        Char code seg3[]=
        {0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f, 0x7f, 0xbf, 0xdf, 0xef,
0xf7, 0xfb, 0xfd, 0xfe,
        0xff, 0x00, 0xff, 0x00, 0xff, 0x00, 0xff, 0x00, 0xff, 0x00, 0xff, 0x00,
0xff, 0x00, 0xff, 0x00,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
        0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xfd, 0xfb, 0xf7,
0xef, 0xdf, 0xbf, 0x7f,
        0x7f, 0xbf, 0xdf, 0xef, 0xf7, 0xfb, 0xfd, 0xfe, 0xfe, 0xfd, 0xfb, 0xf7,
0xef, 0xdf, 0xbf, 0x7f,
        0xfe, 0xfc, 0xf8, 0xf0, 0xe0, 0xc0, 0x80, 0x00, 0x00, 0x80, 0xc0, 0xe0,
0xf0, 0xf8, 0xfc, 0xfe,
        0xfe, 0xfc, 0xf8, 0xf0, 0xe0, 0xc0, 0x80, 0x00, 0x00, 0x80, 0xc0, 0xe0,
0xf0, 0xf8, 0xfc, 0xfe,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
        0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55,
0xaa, 0x55, 0xaa, 0x55,
        0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa,
```

```
0x55, 0xaa, 0x55, 0xaa,
       0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55,
0xaa, 0x55, 0xaa, 0x55,
       0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, };
       Char code seg4[]=
       {0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f, 0x7f, 0xbf, 0xdf, 0xef,
0xf7, 0xfb, 0xfd, 0xfe,
       0xff, 0x00, 0xff, 0x00, 0xff, 0x00, 0xff, 0x00, 0xff, 0x00, 0xff, 0x00,
0xff, 0x00, 0xff, 0x00,
       0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
       0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x7f, 0xbf, 0xdf, 0xef,
0xf7, 0xfb, 0xfd, 0xfe,
       0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
       0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f, 0x7f, 0xbf, 0xdf, 0xef,
0xf7, 0xfb, 0xfd, 0xfe,
       0xfe, 0xfc, 0xf8, 0xf0, 0xe0, 0xc0, 0x80, 0x00, 0x00, 0x80, 0xc0, 0xe0,
0xf0, 0xf8, 0xfc, 0xfe,
       0xfe, 0xfc, 0xf8, 0xf0, 0xe0, 0xc0, 0x80, 0x00, 0x00, 0x80, 0xc0, 0xe0,
0xf0, 0xf8, 0xfc, 0xfe,
       0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
       0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
       0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
       0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
       0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55,
0xaa, 0x55, 0xaa, 0x55,
       0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55,
0xaa, 0x55, 0xaa, 0x55,
       0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
       0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55, 0xaa, 0x55,
0xaa, 0x55, 0xaa, 0x55,
       };
       //======================================主程序======================
       main()
       {int i;
       while(1)
       {
       for(i=0;i<256;i++)
       {
       led0=seg1[i];
```

```
        led1=seg2[i];
        led2=seg3[i];
        led3=seg4[i];
        delay(3000);
        }
        }
        }
        //==================================延时函数=========================
        delay(int x)
        {int i, j;
        for(i=0;i<x;i++)
        for(j=0;j<120;j++);
        }
-------------------------------------------------------------------------------
```

从程序可看出，灯饰循环效果如下：

P0、P1、P2、P3 的灯依次从左到右、从右到左亮。

P0、P1、P2、P3 的灯闪烁八次。

P0 的灯依次点亮→P1 的灯依次点亮→P2 的灯依次点亮→P3 的灯依次点亮，形成跑马灯。

P0、P3 的灯从右到左亮，P1、P2 的灯从左到右亮。

P0、P1、P2、P3 的灯依次亮，直至全亮。
P0、P1、P2、P3 的灯依次灭，直至全灭。 } 重复两次

重复三次：P0、P3 的灯全亮→P1、P2 的灯全亮。

P0、P1、P2、P3 的灯交替闪烁若干次。

思考题

能否自主设计其他灯饰，如金字塔，再编写丰富多彩的循环码，让金字塔披上一件五彩斑斓的华丽衣裳？

任务评估

任务评估见表 1-2-2。

表 1-2-2　任务评估表

评 价 项 目	评 价 标 准		得　分
硬件设计	能自主设计海珠桥灯饰工程效果图	10 分	
	能用 Proteus 软件，熟练绘制海珠桥灯饰工程原理图	10 分	
软件设计与调试	根据任务分析，正确绘制程序流程图	10 分	
	能用 Keil μVision3 软件，编写及调试程序	30 分	

续表

评 价 项 目	评 价 标 准	得　分
软件设计与调试	能充分发挥想象力，设计丰富多彩的海珠桥循环灯饰效果　20 分	
软硬件调试	能按照原理图，正确完成单片机开发板各电子元器件的接线，通电后观看海珠桥的灯饰效果　10 分	
团队合作	各成员分工协作，积极参与　10 分	

知识考核

一、选择题

1．8951 单片机内部程序存储器（ROM）容量为（　　）。

A．2KB　　B．4KB　　C．256B　　D．8KB

2．学校或培训机构常用的单片机封装形式是（　　）。

A．QFP　　B．DIP40　　C．PLCC　　D．PDIP42

3．8951 单片机第 9 引脚是（　　）引脚。

A．电源　　B．复位　　C．时钟　　D．接地

4．单片机复位以后，P2 口的状态是（　　）。

A．低电平　　B．高电平　　C．不定　　D．高阻

5．单片机常用编程（C 语言）调试软件是（　　）。

A．Altium Designer　　B．Proteus

C．Keil μVision3　　D．Protel 99se

6．用 Keil μVision3 软件编译过的（　　）后缀文件可以加到 Proteus 软件中的单片机上进行仿真。

A．hex　　B．doc　　C．asm　　D．c

7．8951 单片机内部数据存储器（RAM）容量为（　　）。

A．2KB　　B．4KB　　C．256B　　D．128B

8．8951 单片机电源引脚是第（　　）脚。

A．1　　B．9　　C．20　　D．40

9．8951 单片机 18 和 19 引脚是（　　）引脚。

A．时钟脉冲　　B．定时器/计数器　　C．存储器　　D．电源

10．单片机常用软硬件联合调试软件是（　　）。

A．Altium Designer　　B．Proteus

C．Keil μVision3　　D．Protel 99se

11．8951 单片机外部存储器最多可扩展（　　）KB。

A．2　　B．4　　C．256　　D．64

12．51 单片机有（　　）组 8 位输入/输出口。

A．1　　B．2　　C．4　　D．5

13．8951 单片机接地引脚是第（　　）脚。

A．1　　B．9　　C．20　　D．40

14．8951 单片机 31 引脚是（　　）引脚。

A．时钟脉冲　　B．定时器/计数器　C．存储器选择　　D．电源

15．（　　）是无符号字符数据类型。

A．char　　B．unsigned char　C．int　　D．unsigned int

16．（　　）自定义变量是不合法的。

A．kaishi　　B．3duan　　C．key　　D．hao

17．下面有关 for 语句的说法，错误的是（　　）。

A．for 语句是循环指令

B．for 语句后面紧跟小括号

C．for 语句结构的大括号不能省略

D．for 语句里面遇到 break 语句则跳出循环

18．函数起始和终止符号是（　　）。

A．大括号　　B．小括号　　C．书名号　　D．双引号

19．（　　）是整数类型。

A．char　　B．unsigned char　C．int　　D．unsigned int

20．（　　）自定义变量是合法的。

A．if　　B．while　　C．switch　　D．shuma

21．下面有关 while 语句的说法，错误的是（　　）。

A．while 语句是循环指令

B．while 语句后面的小括号里为空时表示死循环

C．while 语句结构的大括号不能省略

D．while 语句里面遇到 break 语句则跳出循环

22．共阴极接法的 LED 灯，单片机输出（　　）电平时灯才亮。

A．高　　B．低　　C．不定　　D．高阻

23．用 C 语言编写的源文件的后缀是（　　）。

A．txt　　B．doc　　C．asm　　D．c

24．一条语句结束的符号是（　　）。

A．句号　　B．分号　　C．逗号　　D．点

25．（　　）是无符号整数类型。

A．char　　B．unsigned char　C．int　　D．unsigned int

26．自定义变量取名不符合规则的是（　　）。

A．使用下画线　　B．容易理解和有意义的

C．数字开头　　D．用大小写字母

二、综合题

1．什么是单片机？单片机内部包括哪几部分？

2．单片机的存储器如何分类？

3．请用 C 语言编写一个延时 1ms 的子程序 delay，晶振为 12MHz。

4．请画出单片机最小系统电路图。

5．单片机 P1 口已接上 8 个发光二极管，采用共阳极接法，请编写程序，使 8 个 LED 灯按顺序循环点亮，要求有一定的延时。