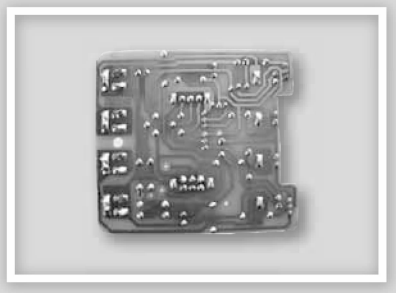
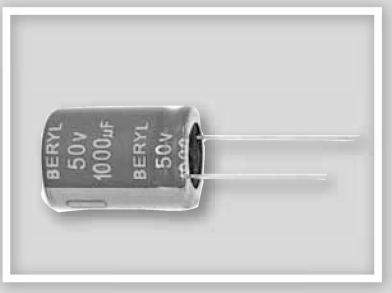
项目二

按键计数器





项目介绍

在很多由单片机控制的应用系统中，为了更好地控制系统的工作状态，实现人机对话，数码管和键盘是必不可少的组成部分。图 2－0－1 就是我们生活中常见的智能电表，该表的人机界面就是由四位 LED 数码管和独立按键构成。

本项目是制作一个按键计数器电路，并能够根据控制要求编写相关单片机程序。具体的功能要求为：系统刚上电时，数码管显示“00”，每次按动加法键，数码管显示数据增加 1，每次按动减法键，显示数据减 1，通过按键设置，让数码管显示范围在“00～20”之间变化。

图 2－0－1　智能电表

通过本项目的学习和实践，我们掌握按键检测、消抖的程序设计方法；数码管结构类型、段码、静态与动态显示程序设计方法。

学习目标

- 了解键盘检测的电路结构和原理、键盘作用、如何实现键盘检测、消抖、键盘编码等内容。
- 掌握独立按键控制、消抖动等基本原理与程序设计方法。
- 了解 4×4 按键矩阵构成及按键扫描程序。
- 认识各种数码管，包括数码管的种类、显示原理、段码。
- 掌握单只数码管静态显示数字或字母的程序设计。
- 学会单只数码管动态显示数字或字母的程序设计。
- 掌握多位数码管静态显示数字或字母的程序设计。
- 了解多位数码管动态扫描显示原理，能读懂动态扫描显示程序。

任务一　按键输入检测

知识准备

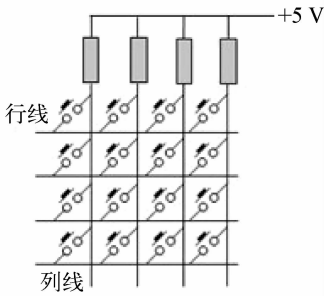
在很多项目中都会用到按键，如图 2－1－1 所示的电子秤。本次任务将按键接在单片机的管脚上，通过让单片机识别按键是否按下，完成对 LED 发光二极管的控制。具体功能要求为：LED 发光二极管亮的时候，按一下按键，发光二极管灭；发光二极管灭的时候，按一下按键，发光二极管亮。

图 2－1－1　电子秤

一、键盘的分类

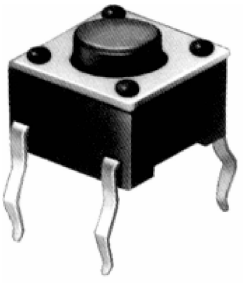
键盘分编码键盘和非编码键盘两种。键盘上闭合键的识别由专用的硬件编码器实现，并产生键编码号或键值的称为编码键盘，如计算机键盘。而靠软件编程来识别的称为非编码键盘，在单片机组成的各种系统中，用得最多的是非编码键盘。非编码键盘分为行列式（又称为矩阵式）键盘（图 2－1－2）和独立按键（图 2－1－3）。本次任务，我们选择的是独立按键。

(a) 4×4矩阵键盘外形图

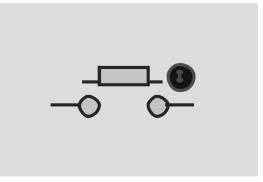


(b) 内部连接图

图 2－1－2　矩阵键盘



(a) 独立键盘外形图



(b) 图形符号

图 2－1－3　独立按键

二、独立按键连接图

独立按键连接电路有两种，一种是按键的公共端接地，当按键按下时，单片机的控制引脚输入低电平“0”，如图 2－1－4 所示。另一种是按键的公共端接电源，这样当按键按下时，单片机的控制引脚接高电平“1”，如图 2－1－5 所示。

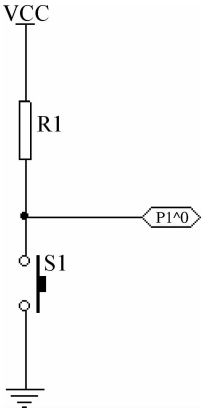


图 2－1－4　控制引脚输入低电平“0”

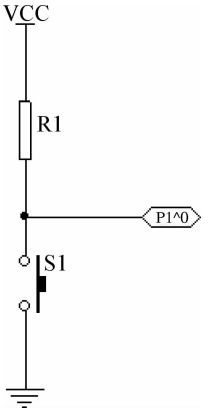


图 2－1－5　控制引脚接高电水“1”

应用时一般选择图 2－1－4 所示的这种电路，电源 VCC 选择＋5 V 的直流电源，上拉电阻选择10 kΩ 的。

三、按键抖动

通常的按键所用开关多为机械弹性开关，当机械触点断开、闭合时，电压信号波形如图 2－1－6 所示。由于机械触点的弹性作用，一个按键开关在闭合时不会马上稳定地接通，在断开时也不会一下子断开。因而在闭合及断开的瞬间均伴随一连串的抖动。抖动时间的长短由按键的机械特性决定，一般为 5～10 ms，这是一个很重要的时间参数，在很多场合都要用到。

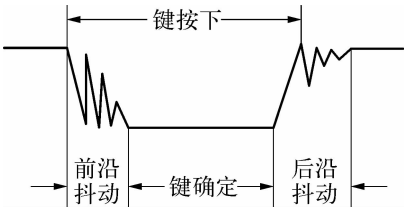


图 2－1－6　电压信号波形

四、按键程序编写

由于按键按下后，会出现抖动，单片机应该避开这段时间，待键盘稳定后，再对键盘的信息进行识别。目前消抖动的方法有两种：一种是采用硬件电路来实现，如增加 RS 触发器电路、滤波电路等；另一种是软件来实现，即当检测出按键闭合后，执行一个延时程序，产生 10～20 ms 的延时，等延时过后，再查询是否有按键按下，如果没有键按下，说明上次查询结果为干扰；如果仍有键按下，则说明按键闭合已经稳定，单片机可以对键盘进行识别了。

任务实施

一、绘制原理图

在 PROTEUS 软件中绘制的原理图，如图 2－1－7 所示。

二、编写源程序

根据原理图，编写源程序如下：

```
/**************************************************************
    按键控制灯程序
灯接在 P0~7 端口上
开关接在 P2~0 端口上
灯亮的时候，按一下开关，灯灭
灯灭的时候，按一下开关，灯亮
**************************************************************/
#include"reg51.h"
```

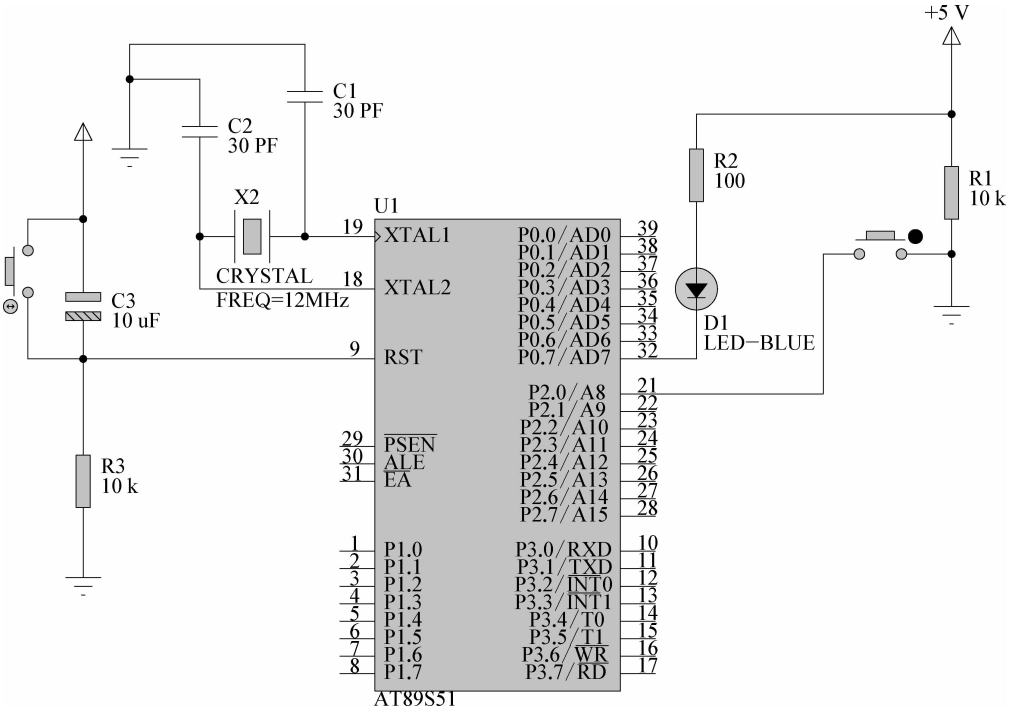


图 2-1-7　按键控制灯原理图

```
sbit key=P2^0;
sbit led=P0^7;
bit   led_flag; //状态标记
/*************************
        延时函数
延时时间=time*1 毫秒

*************************/
void delay_ms(unsigned int time)
{
unsigned int i,j;
for(i=0;i<time;i++)
for(j=0;j<121;j++);
}
void main(void)
{
    led=1;//开始的时候,先熄灭灯
while(1)
```

```
{
        if(key==0)//当发现有键按下,注意此时还不能确定按下,也许是抖动
        {
        delay_ms(10);//如果发现有键按下,先延时一段时间,去抖动
          if(key==0)//延时过后,发现按键还按着,可以确定按键的确是按下了
           {
           //下面开始执行按键相关的动作
             led_flag=~led_flag;
             if(led_flag==1)
             led=0;
             else
             led=1;
        //完成相关任务后,等待按键松开
             while(key==0);//等待按键松开
           }
        }
    }

}
```

三、程序分析

① 按键识别的步骤：先判断按键是否按下，如果没按，直接退出。如果按下，此时还不能确定是否为有效按下，有可能是抖动，此时应去抖动，调用延时函数。等去抖动完成后，再次判断按键是否还依然按着，如果没有，说明刚才检测按下是一个错误状态，如果依然按着，说明按键是的确按下了，程序可以进行相关参数的设定和修改，最后就是等待按键的释放。

② 本程序将灯亮还是灭分成两个状态。在亮的时候，如果按键按下，灯灭。在灭的时候，如果按键按下，灯亮。这里的 led_flag 就是作灯状态的一个标记。

四、编译与仿真

将上述源程序在 KEIL C 中编译并生成 HEX 文件，在 PROTEUS 中作原理图仿真。正确的编译结果如图 2-1-8 所示。

需要注意的是，PROTEUS 对单片机的仿真，在画原理图时可以省略最小化系统部分，本图中我们还是保留了复位电路和晶振电路部分。还需要注意的是 PROTEUS 隐藏了单片机的电源和地管脚，实际制作电路板的时候需要接上。

最后的仿真运行结果如图 2-1-9、2-1-10 所示。

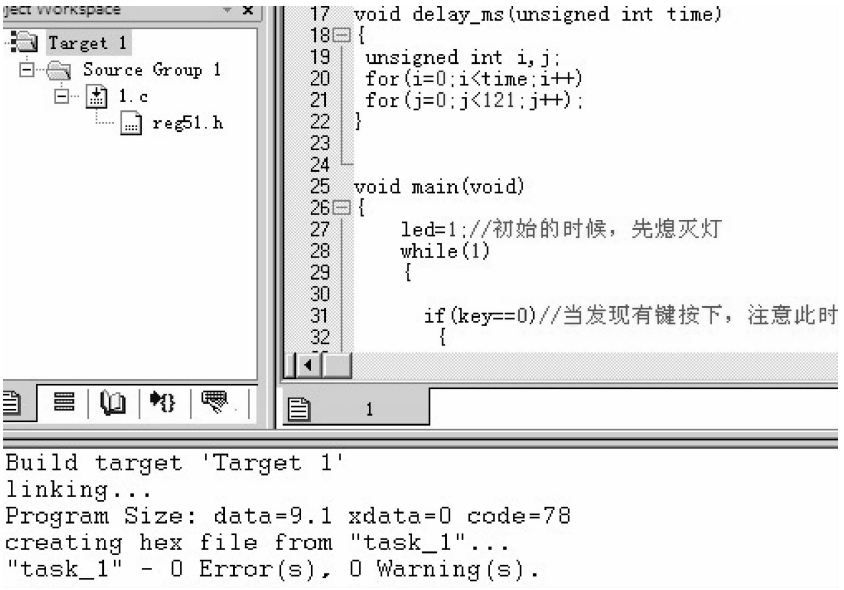


图 2－1－8　正确的编译结果

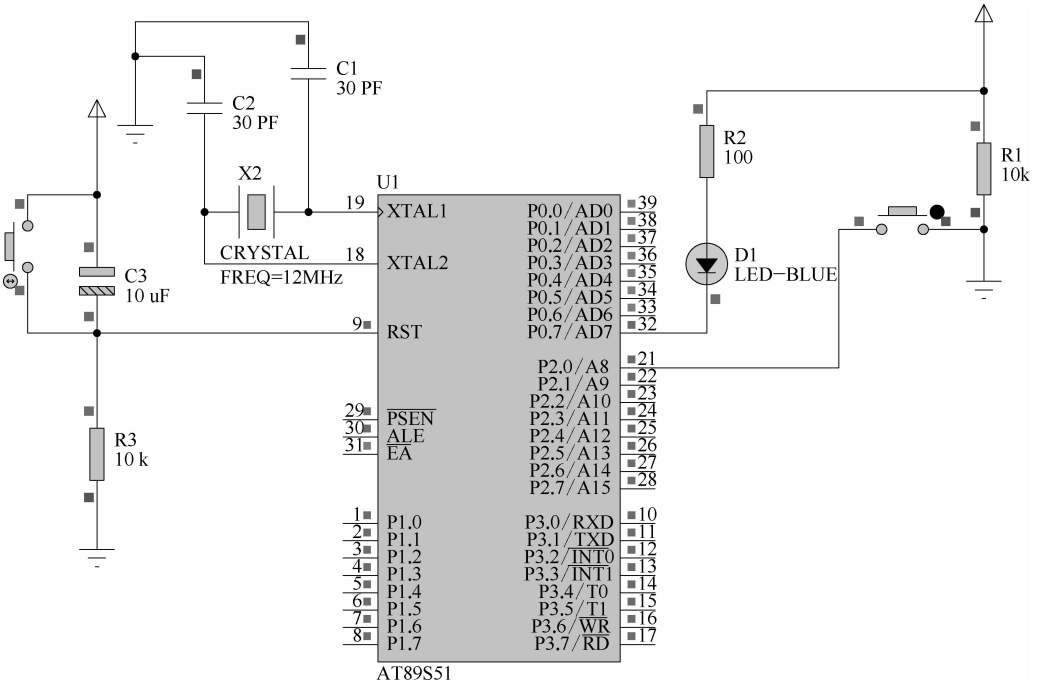


图 2－1－9　灯灭

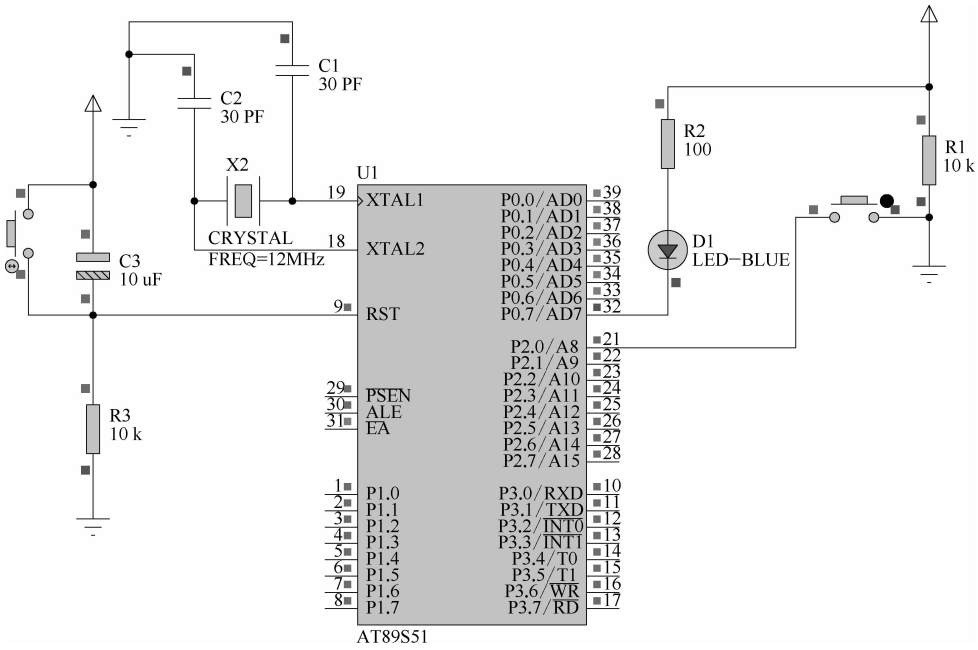


图 2－1－10　灯亮

知识拓展

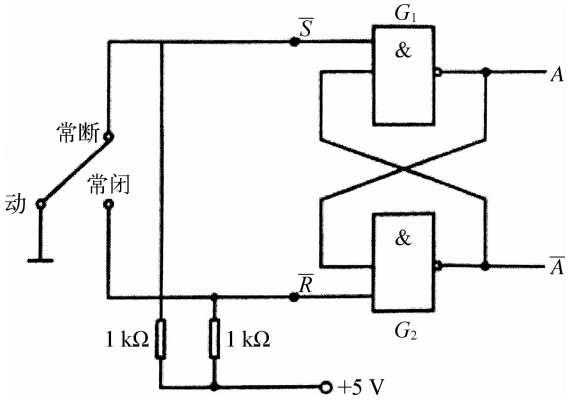
试编写程序：按键按下灯亮，松开灯灭。

目标检测

1. 简述键盘扫描的主要思路。

2. 用示波器观察按键抖动现象。

3. 除了软件去抖动外，还可以利用硬件，右图为 RS 触发器抵抗开关的抖动，试分析其原理。



任务二　单只数码管显示

知识准备

数码管作为一种应用十分普遍的显示器件可以在各种各样的设备上见到，图 2－2－1 就是电子钟显示的效果图。数码管很适用在对价格、亮度等条件比较敏感，同时基本上只要求显示数字量的时候，所以在数据显示、定时控制等场合用得很多。本次任务主要是单片机控制单只数码管上实现 0 到 9 数字的显示。

图 2－2－1　电子钟

一、数码管

数码管也叫 LED 数码显示器，其实是由多个 LED 排列封装而成，图 2－2－2 给出了一些常见数码管的实物图，其引脚如图2－2－3 所示。

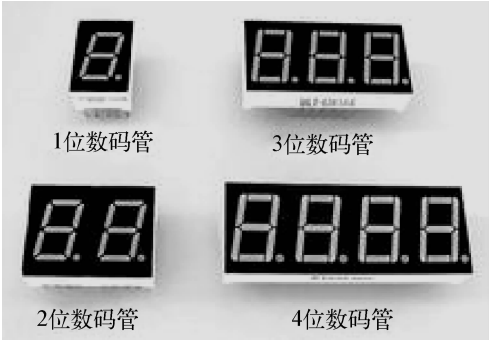


图 2－2－2　数码管实物图

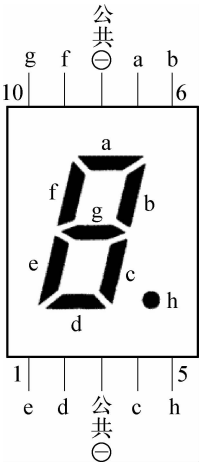


图 2－2－3　单只数码管引脚图

二、LED 七段数码管结构原理

LED 七段数码管通常由 8 个 LED 发光二极管组成，其中 7 个发光 LED 二极管构成 7 笔字形，一个构成小数点，通称七段 LED 数码管。LED 七段数码管分为共阳极接法（8 个发光二极管阳极连在一起，如图 2－2－4 所示）和共阴极接法（8 个发光二极管阴极连在一

起，如图 2－2－4 所示）。

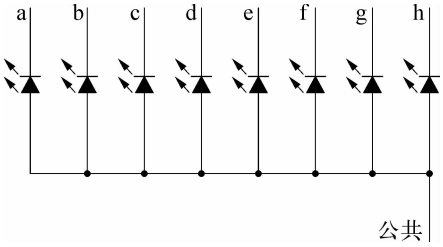


图 2－2－4　共阳极接法

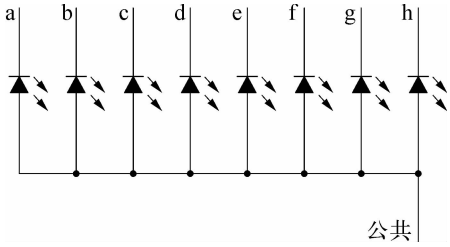


图 2－2－5　共阴极接法

通过控制 LED 七段数码管中的公共端，可控制该位数码管的亮或灭。对于共阳数码管，公共端接低电平，数码管熄灭。对于共阴数码管，公共端接高电平，数码管熄灭。如果想点亮具体一个 LED 发光二极管，对于共阳数码管，必须先将公共端接高电平，如果想让 a 段亮，只需给 a 段接口送一个低电平。对于共阴数码管，必须将公共端接低电平，如果想让 a 段亮，只需给 a 段接口送一个高电平。

三、LED 七段数码管显示字符和字母的方法

如果我们要显示“6”，并不是给数码管写个“6”就行了，而是点亮这个数码管的相应的一些 LED 发光二极管，即点亮 a，c，d，e，f，g 段，其余段灭。如果 a，b，c，d，e，f，g，h 是接在单片机的 P0 端口上，所以要想显示“6”，对应的 I/O 端口输出如表 2－2－1 所示。对于共阳数码管将 0X82 这个段码输给 P0 端口，对于共阴数码管将 0X7D 这个段码输给 P0 端口就可以了。通过该表可以看出，其实共阳和共阴数码管的段码互为反码。按此方法，我们可以推出 0～9 和一些常见英文字符的段码，如表 2－2－2 所示。

表 2－2－1　数字“6”对应的段码

段名称	H 小数点	G	F	E	D	C	B	A	对应段码
亮 灭	灭	亮	亮	亮	亮	亮	灭	亮	
对应引脚	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	
共阳	1	0	0	0	0	0	1	0	0X82
共阴	0	1	1	1	1	1	0	1	0X7D

表 2－2－2　LED 七段数码管的段码表

显示字符	共阴数码管段码	共阳数码管段码
0	3FH	C0H
1	06H	F9H

续 表

显示字符	共阴数码管段码	共阳数码管段码
2	5BH	A4H
3	4FH	B0H
4	66H	99H
5	6DH	92H
6	7DH	82H
7	07H	F8H
8	7FH	80H
9	6FH	90H
A	77H	88H
B	7CH	83H
C	39H	C6H
D	5EH	A1H
E	79H	86H
F	71H	8EH
“灭”	00H	FFH

任务实施

一、绘制原理图

在 PROTEUS 中画好的原理图，如图 2－2－6 所示。本图中我们省去单片机的复位电路和晶振电路。但是在制作电路板的时候，还是要加进去，不能省去。

二、编写源程序

```
/*************************************************************
    共阳数码管显示 0－9
数码管的输入端接在 P0 端口上
*************************************************************/
#include"reg51.h"
#define port_0 P0
```

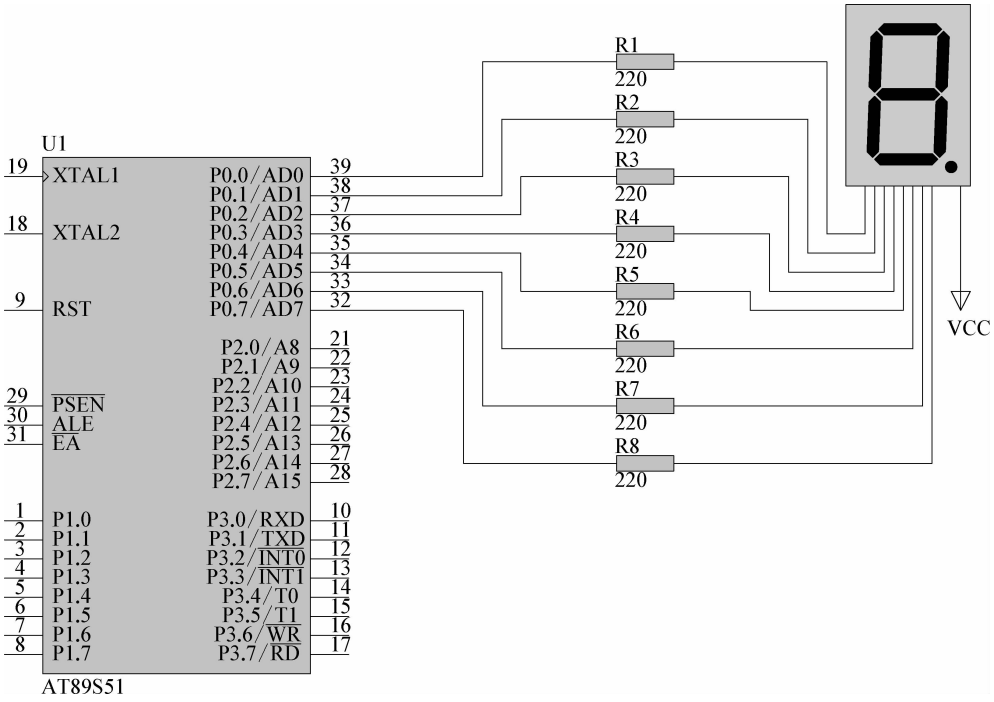


图 2-2-6　单只数码管仿真图

```
unsigned char code table[]={
        0xC0,/*0*/
        0xF9,/*1*/
        0xA4,/*2*/
        0xB0,/*3*/
        0x99,/*4*/
        0x92,/*5*/
        0x82,/*6*/
        0xF8,/*7*/
        0x80,/*8*/
        0x90/*9*/
};//定义0到9数字对应的段码,把段码放在code里,节省RAM。
void delay_time(unsigned int time);//函数说明
unsigned char count;
void main(void)
{

    while(1)
```

```
    {
        port_0=table[count];//输出 0-9 对应的段码
        delay_time(1000);//每个段码显示的时间
        count++;//对 count 进行处理,使其数值在 0-9 之间变化
        if(count>9)
        count=0;

    }
}

/**************************************************************
        延时函数
延时时间=time*1 毫秒
**************************************************************/
void delay_time(unsigned int time)
{
unsigned int i,j;
for(i=0;i<time;i++)
for(j=0;j<121;j++);
}
```

三、程序分析

① port_0=table[count]：当 count=0 时，port_0=table[0]，相当于把“0”的段码通过单片机的 P0 端口送到数码管上，数码管显示“0”。如果想显示其他数字，通过修改里面的 count 就可以了。

② delay_time()是个延时函数：控制前后两次显示的切换的快慢。延时时间越长，切换就越慢，每个数字显示的时间就长；反之时间越短，切换就越快，每个数字显示的时间就短。

③ 在任务一和任务二中，我们都调用到了一个 delay_time()的延时函数，程序中，delay_time()函数的定义如果出现在 main()函数前，就不需要对 delay_time()做说明而直接使用。如果 delay_time()函数出现在 main()函数后，需要在 main()外先说明一下这个函数的类型，如本次任务。

四、编译与仿真

将上述源程序在 KEIL C 中编译并生成 HEX 文件，在 PROTEUS 中作原理图仿真。正确的编译结果如 2-1-7 所示。

PROTEUS 对单片机的仿真结果见图 2-2-8～图 2-2-17。

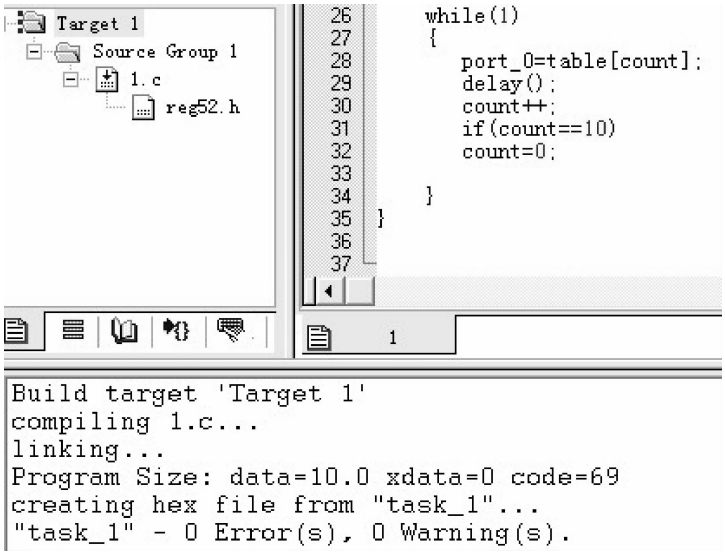


图 2－2－7　编译仿真图

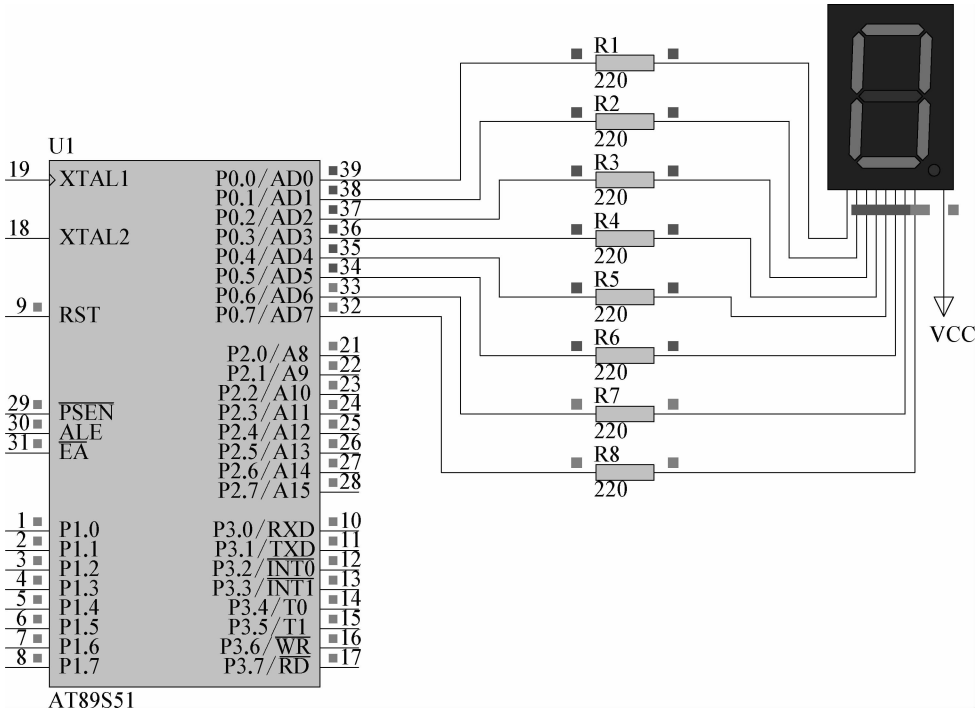


图 2－2－8　显示数字“0”

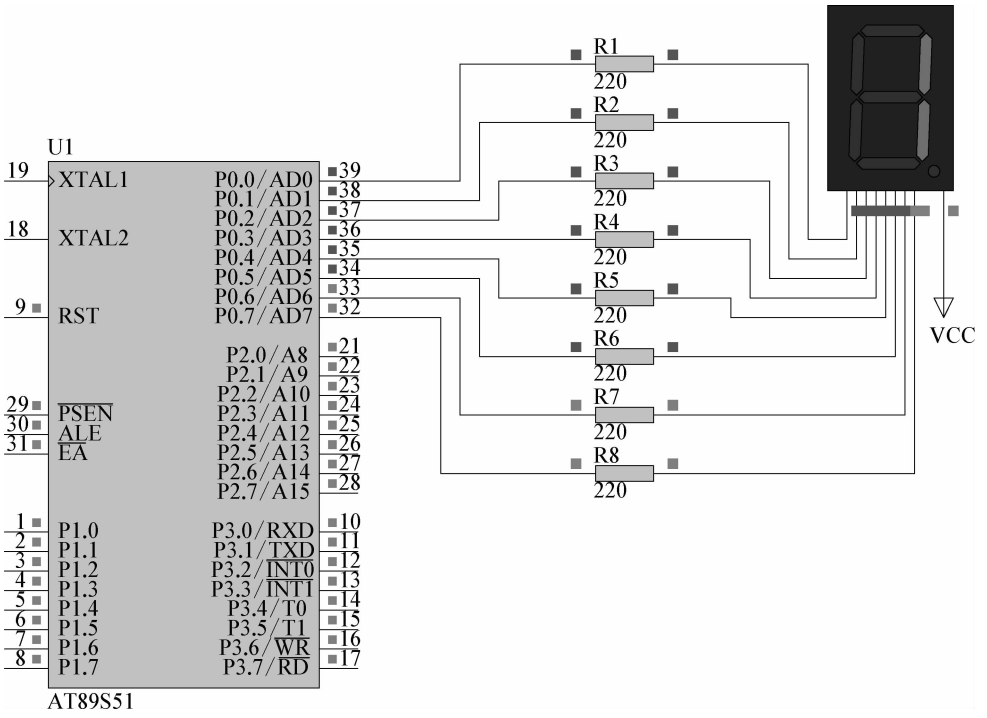


图 2－2－9　显示数字“1”

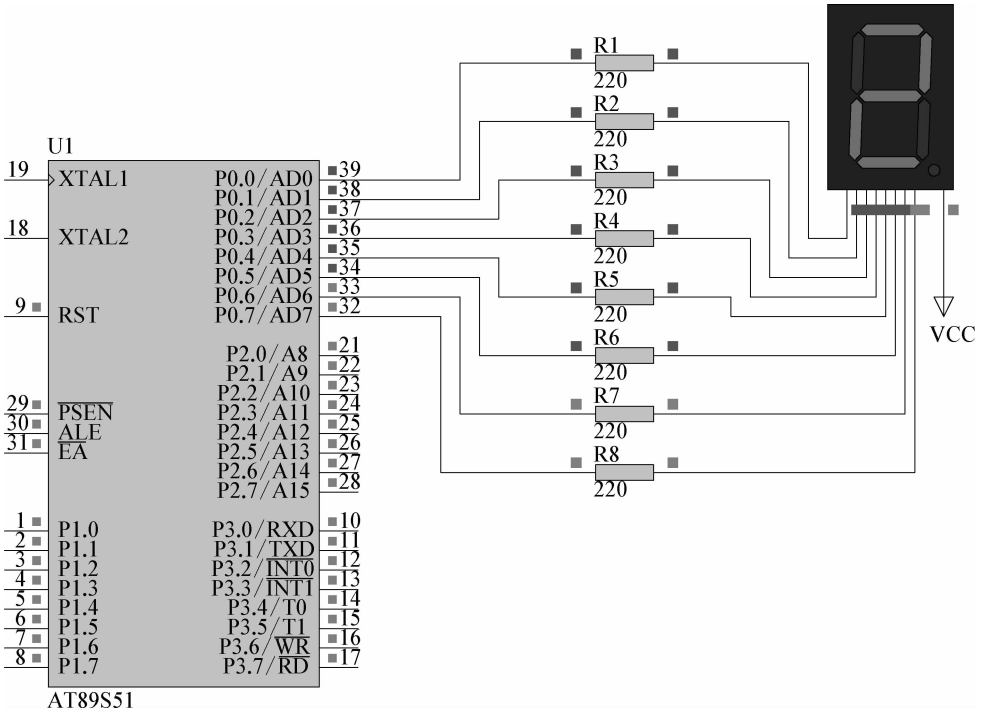


图 2－2－10　显示数字“2”

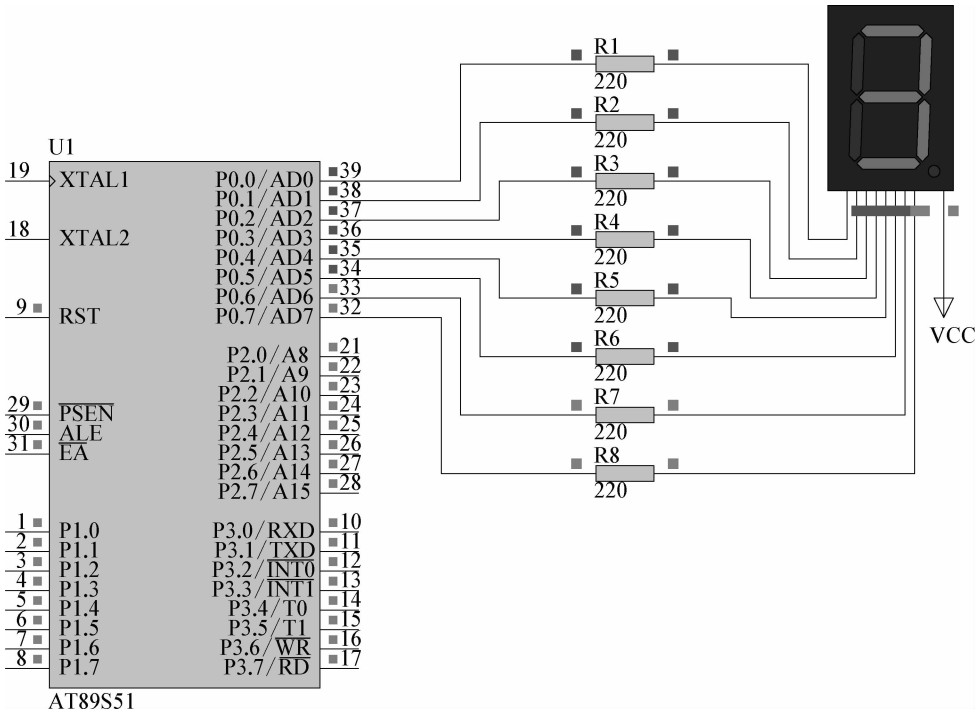


图 2－2－11　显示数字“3”

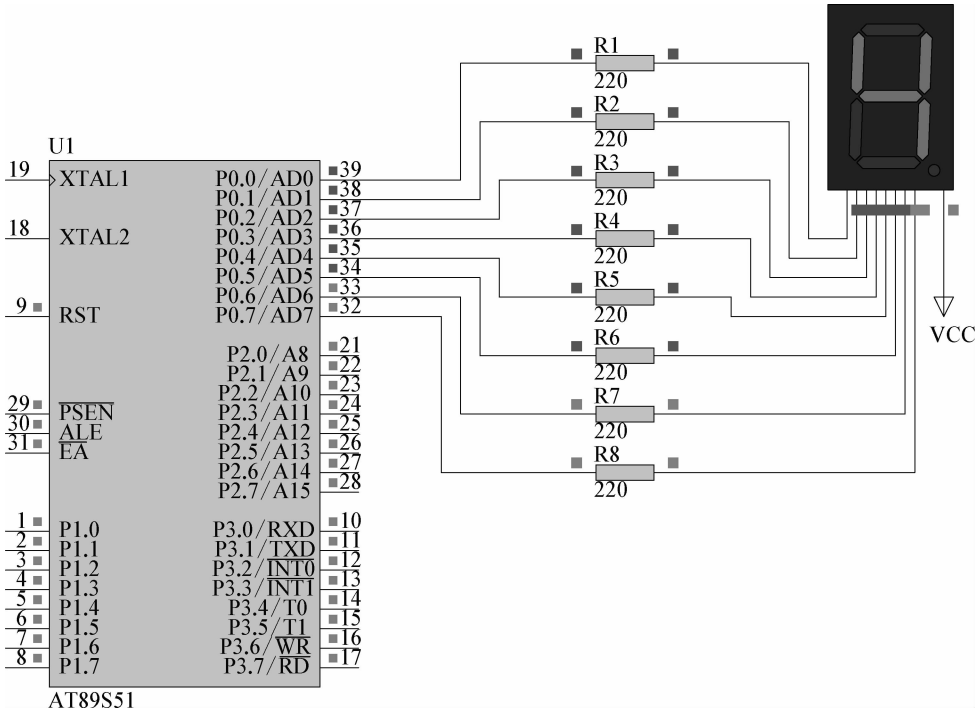


图 2－2－12　显示数字“4”

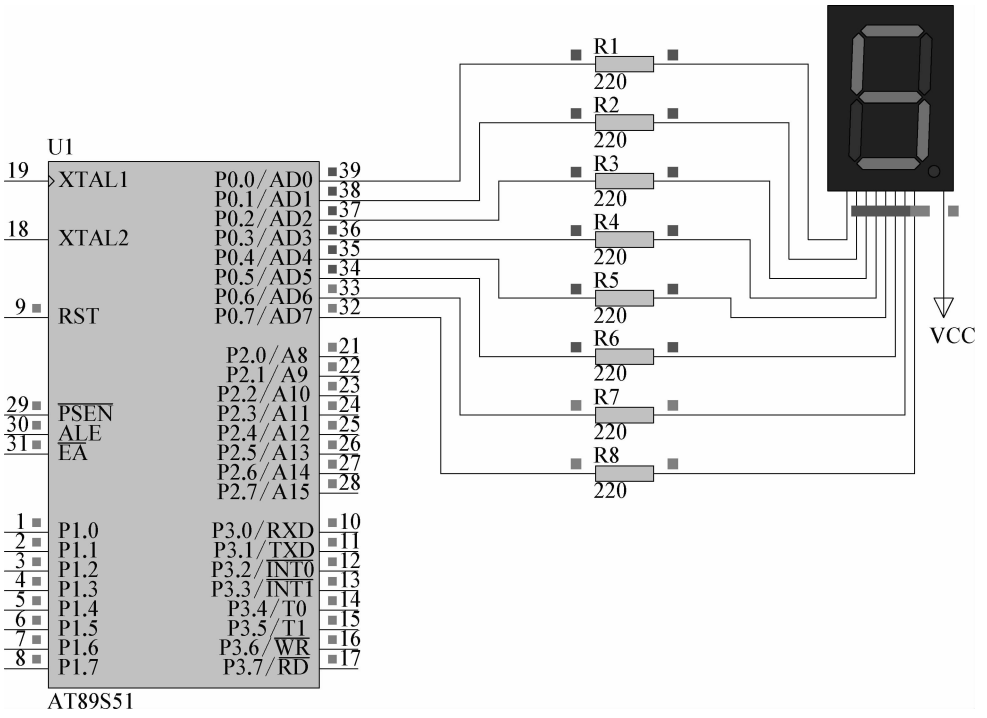


图 2－2－13　显示数字“5”

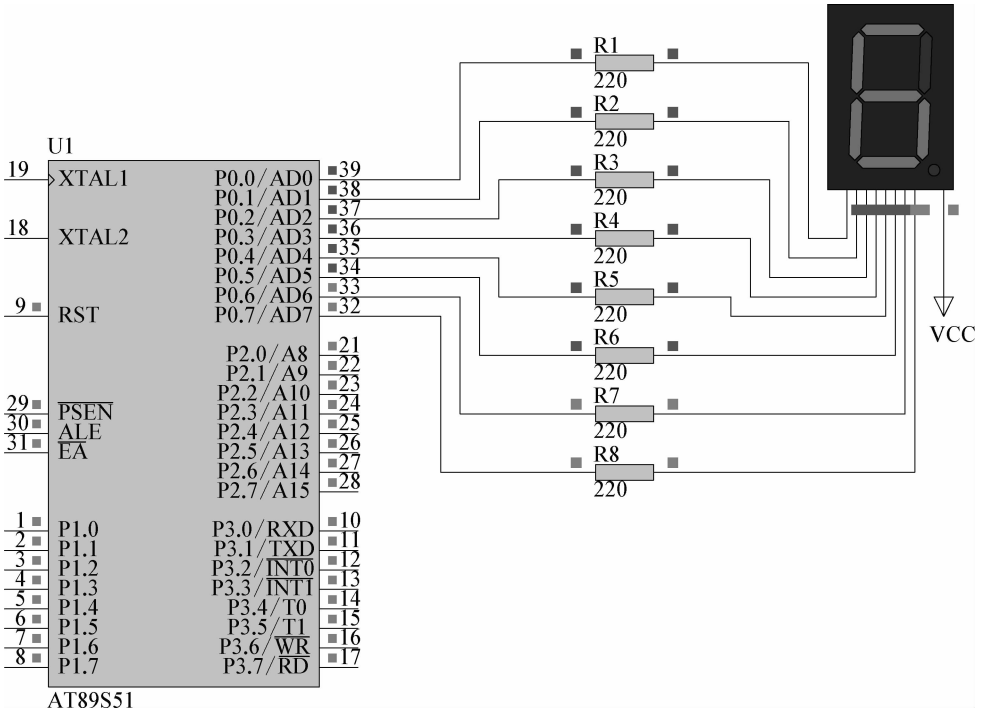


图 2－2－14　显示数字“6”

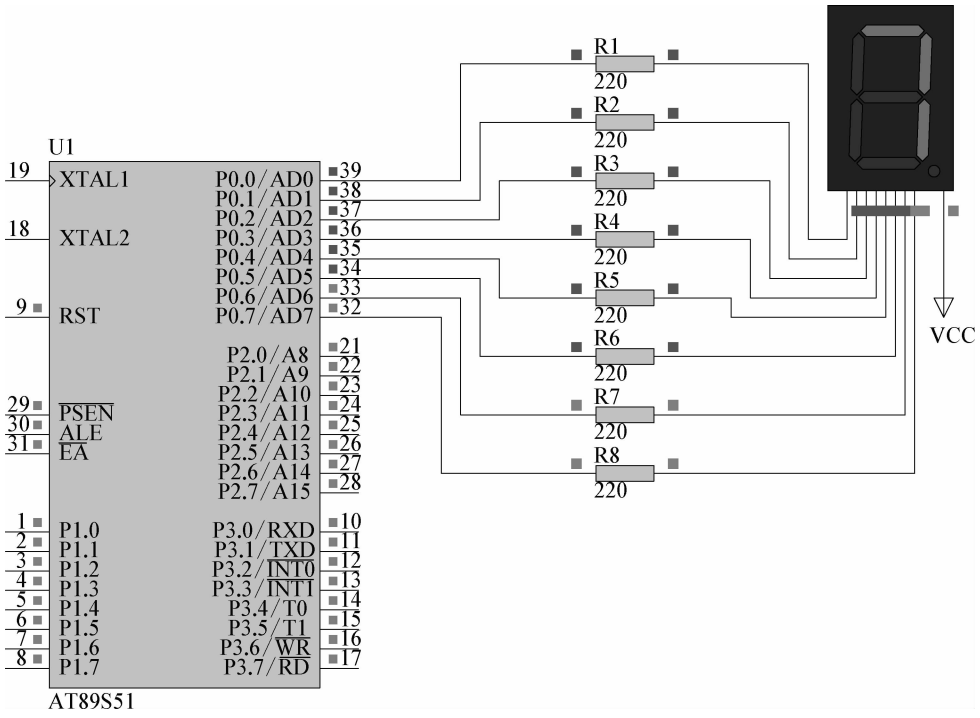


图 2－2－15　显示数字“7”

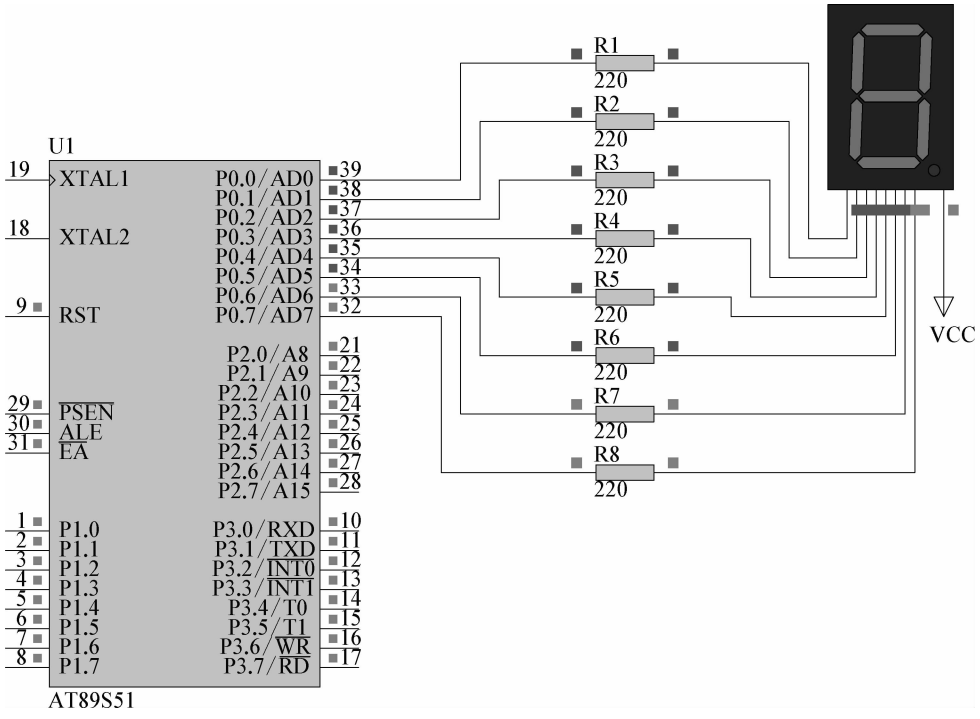


图 2－2－16　显示数字“8”

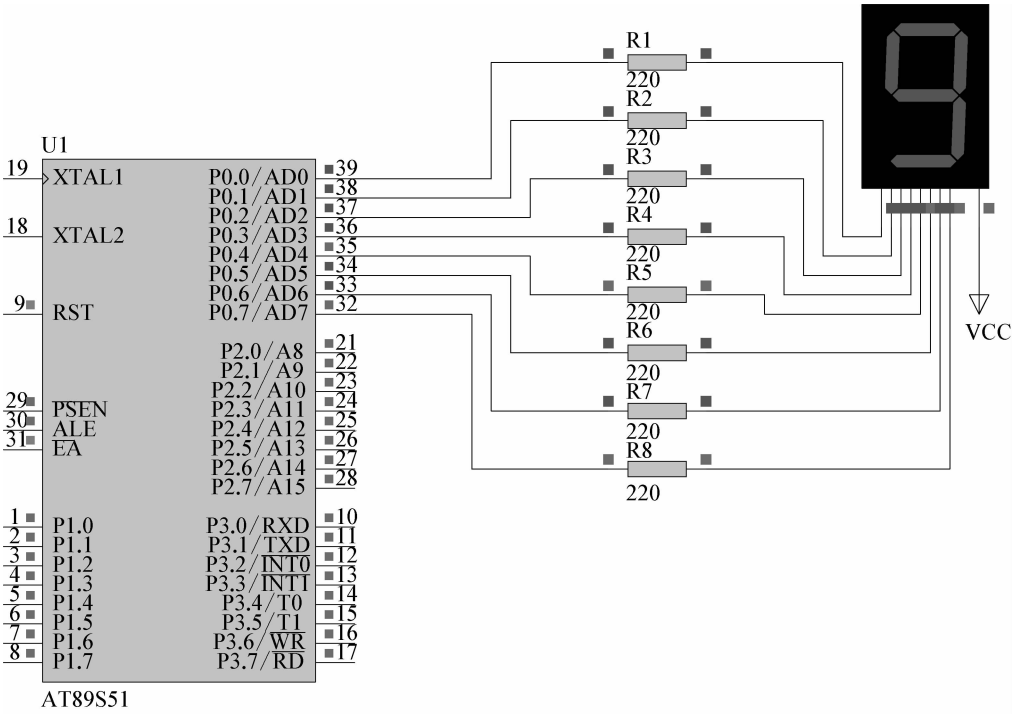


图 2－2－17　显示数字“9”

知识拓展

① 推导共阴和共阳数码管显示“H”的时候的段码。

② 通过修改延时函数，观察数码管显示。

③ 本次任务选择的是共阳数码管，如选择共阴数码管，如何编写程序？

目标检测

1. 描述单位 LED 数码管的物理结构。

2. 简述单位 LED 数码管的显示原理。

3. 在 PROTUES 中绘制数码管显示电路图，将本任务“知识拓展”的例子中 LED 数码管变更到其他口，实现显示 9 到 0 的倒计时。修改程序输入仿真，观察现象。

任务三　多位数码管显示

知识准备

由于很多项目中都会用到多位数码管，多位数码管的显示一般有两种方法：静态显示和动态显示。静态显示因为比较浪费单片机的接口，所以一般用在数码管上只有一个或两个的场合。如果数码管的数量比较多，一般使用动态扫描显示的方法。本任务主要是单片机控制两位共阳数码管上完成静态 00～59 s 的循环计时并掌握如何在两位共阳数码管动态显示。

一、静态显示

所谓静态显示，是指当显示器显示某个字符时，相应位的发光二极管处于恒定的导通或截止状态，直到需要显示另一个字符为止。静态显示方式，LED 的亮度高，软件编程比较容易，但占用比较多的 I/O 端口资源，因此常用于显示位数不多的情况。例如，LED 数码管显示“0”时，段 a、b、c、d、e、f 导通，g、dp 截止；单片机只需将所要显示的数据送出去，直到下次显示数据需要更新时再传送一次数据。

二、动态显示原理

LED 数码管显示的基本原理是利用人眼的“视觉暂留”效应和发光二极管的余晖现象来工作的。接口电路把所有数码管的 8 个笔画段 a～dp 的各同名端相互连接在一起，并把它们接到输出口上，每个数码管的公共端 COM 各自独立地受控制。这样一组数码管显示需要由两组信号来控制：一组是 I/O 端口输出的字形代码，连接到各数码管笔画段，用于控制显示字形，称为段码。另一组是 I/O 口输出的各个控制信号，连接到每个数码管的公共端，用来选择第几位数码管显示，称为位码。当单片机输出段码时，所有的数码管都能接收到，但是要点亮哪一个数码管，就取决于此时相应的位码。

所谓动态，就是利用循环扫描的方式，分时轮流选通多个数码管的 COM 端，使数码管轮流导通。当扫描速度达到一定程度时，人眼就分辨不出来了，认为是各个数码管同时发光的，一般每个数码管点亮的时间为短暂的(1～5 ms)。

三、数码管静态显示驱动电路

将单片机的管脚直接和数码管相连，当然中间需要增加限流电阻，如图 2－3－1 所示。对于共阳数码管，公共端需要接上高电平，要想显示什么字符，只需单片机输出该字符的显示段码就可以了。

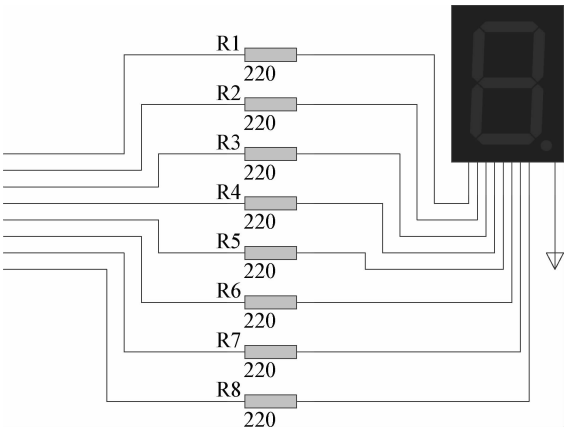


图 2－3－1　静态数码管驱动电路

四、数码管动态显示驱动电路

单片机的管脚和动态数码管相连，中间加上限流电阻，这个和静态显示电路一样。不同的是两个数码管的公共端都接在了单片机的管脚上，这样单片机就可以控制数码管的亮灭。考虑到单片机 I/O 口带负载能力，中间增加了三极管用来驱动，如图 2－3－2。由于我们选择的是共阳数码管，当单片机的 21 脚输出高电平时，三极管 Q1 截止，无论送什

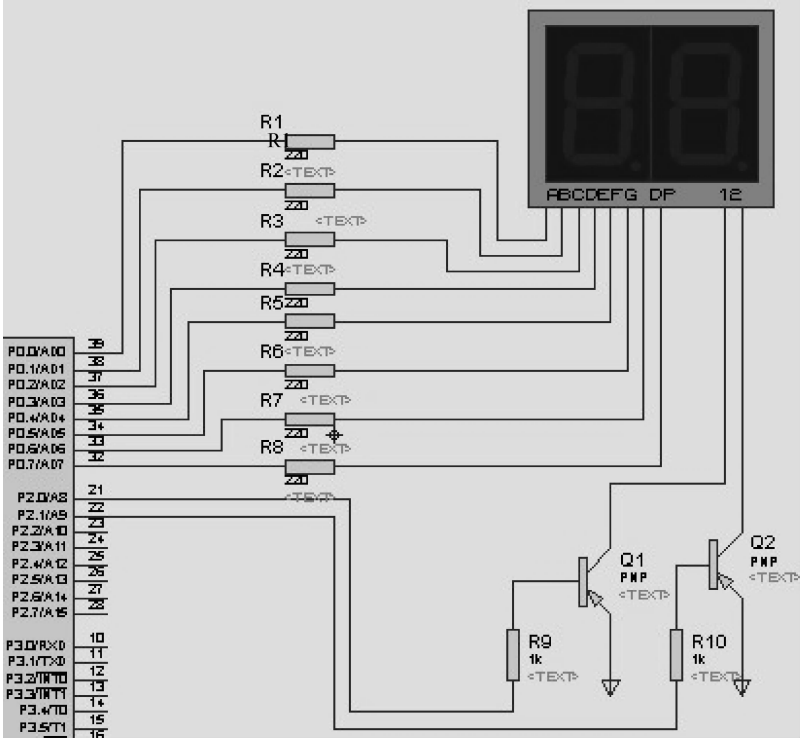


图 2－3－2　动态数码管驱动电路

么段码，数码管 1 都不亮。当单片机的 21 脚输出低电平时，三极管 Q1 导通，数码管 1 公共端为高电平，此时如果单片机送段码，数码管 1 上就可以显示相关字符了。

任务实施

一、静态显示 00～59 s 循环计时

1. 绘制原理图

在 PROTEUS 软件中绘制的原理图，如图 2－3－3 所示。

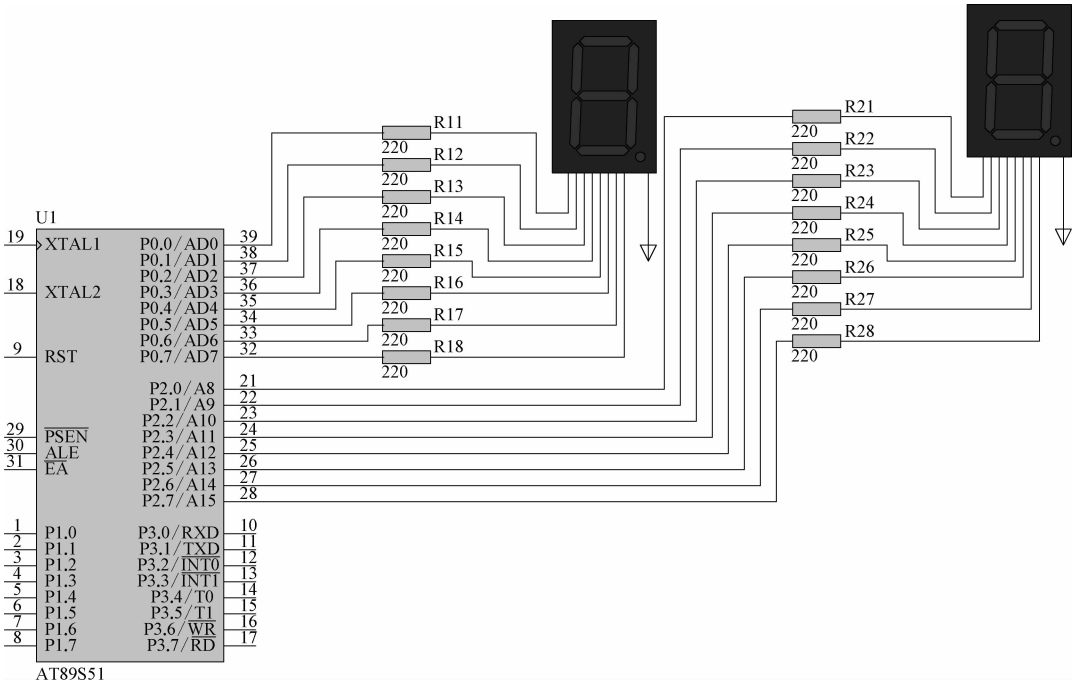


图 2－3－3　两只数码管静态显示仿真电路图

2. 编写源程序

根据原理图，编写源程序如下：

```
/**************************************************************
  数码管静态显示 00－59 计数
  个位数码管 2 接在 P2 端口上
  十位数码管 1 接在 P0 端口上

**************************************************************/
```

```
#include"reg51.h"
#define port_0 P0 //宏定义,给 P0 端口重新起一个名字
#define port_2 P2 //宏定义,给 P2 端口重新起一个名字
unsigned char code table[]={
        0xC0,/*0*/
        0xF9,/*1*/
        0xA4,/*2*/
        0xB0,/*3*/
        0x99,/*4*/
        0x92,/*5*/
        0x82,/*6*/
        0xF8,/*7*/
        0x80,/*8*/
        0x90/*9*/
};
unsigned char second;//秒计数

/**************************************************************
        延时函数
延时时间=time*1 毫秒

**************************************************************/
void delay_time(unsigned int time)
{
    unsigned int i,j;
    for(i=0;i<time;i++)
    for(j=0;j<121;j++);
}
void main(void)
{
    second=0; //初始的时候,计时从 00 开始的
    port_2=table[second%10];//送个位段码
    port_0=table[second/10];//送十位段码
while(1)
{
    delay_time(1000); //延时 1 秒
```

```
    second++; //每 1 秒到,秒计数加 1
     if(second==60) //秒计时范围在 00 到 59 之间
     {
      second=0;
      }
    port_2=table[second%10]; //刷新数码管
    port_0=table[second/10];  //刷新数码管
}
}
```

3. 程序分析

(1) 程序运行的时候,数码管要求显示 00,接着 1 s 到后,数码管显示 01。所以初始化需要加上这几句 second=0; port_2=table[second%10]; port_0=table[second/10]。

(2) 当 second=29 的时候,second%10=9,那么 table[second%10]=table[9],而 table[9]=0X90,正好对应了 9 的段码,通过执行 port_2=table[second%10]语句,数码管个位显示 9。同样 second/10=2,那么 table[second/10]=table[2],而 table[2] =0X90,正好对应了 2 的段码,通过执行 port_2=table[second/10]语句,数码管个位显示 2。这里需要注意的是,排段码表的时候要顺序对应。

4. 编译与仿真

将上述源程序在 KEIL C 中编译并生成 HEX 文件,在 PROTEUS 中作原理图仿真。正确的编译结果如图 2-3-4 所示。

图 2-3-4　正确的编译结果

PROTEUS 对单片机的仿真结果如图 2－3－5 所示。

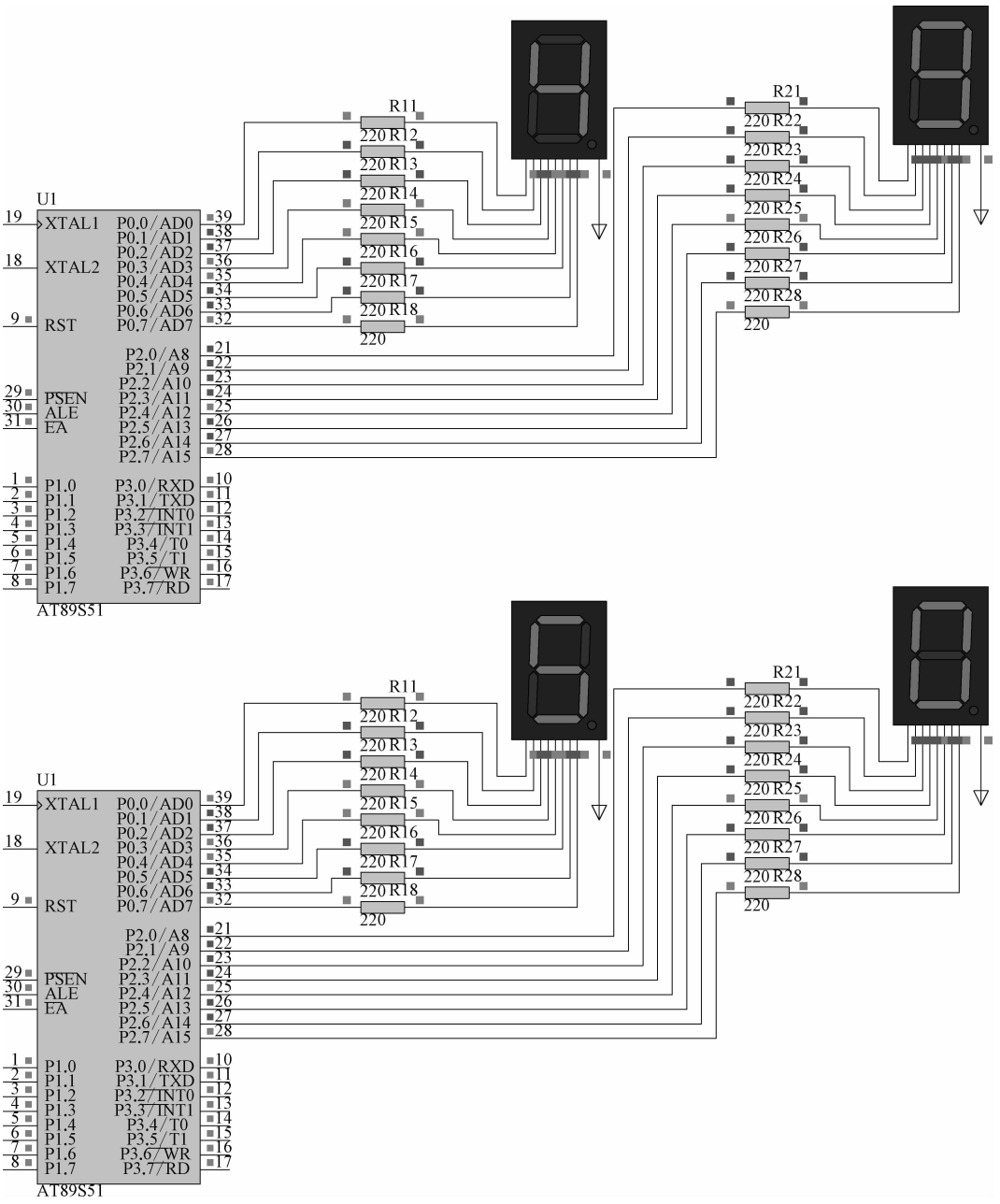


图 2－3－5　PROTEUS 仿真图

二、动态显示电路图

1. 绘制原理图

在 PROTEUS 软件中绘制的原理图，如图 2－3－6 所示。

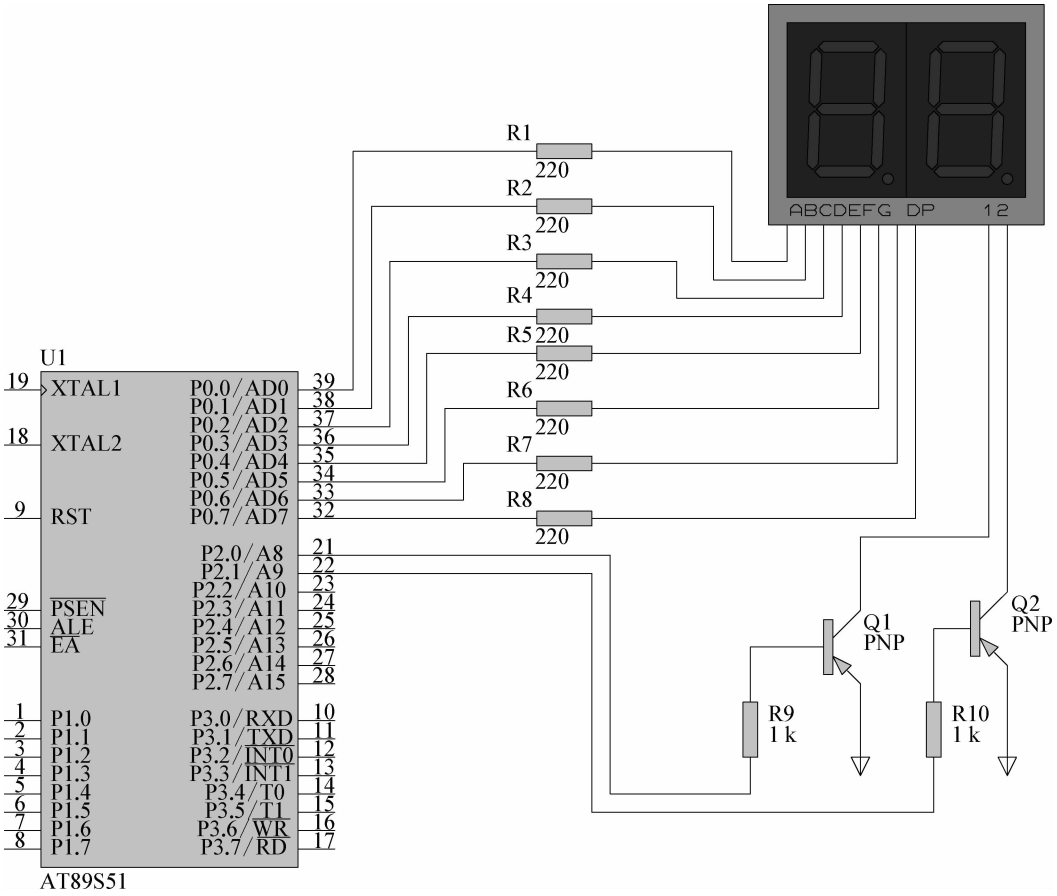


图 2-3-6　动态数码管仿真电路图

2. 编写源程序

根据原理图，编写源程序如下：

```
/*************************************************************
        数码管动态显示
    数码管的数据段接 P0 端口
    数码管 1 的控制端接 P2^0 端口
    数码管 2 的控制端接 P2^1 端口
*************************************************************/
#include <reg51.h>
#define port_0 P0
sbit cs_1=P2^0;
sbit cs_2=P2^1;
unsigned char code table[]={
```

```
        0xC0,/*0*/
        0xF9,/*1*/
        0xA4,/*2*/
        0xB0,/*3*/
        0x99,/*4*/
        0x92,/*5*/
        0x82,/*6*/
        0xF8,/*7*/
        0x80,/*8*/
        0x90,/*9*/
        0x88,/*A*/
        0x83,/*b*/
        0xC6,/*C*/
        0xA1,/*d*/
        0x86,/*E*/
        0x8E,/*F*/
        0xFF/*17,无显示*/
}; //0到9,A到F,不亮的段码表
unsigned char   dispbuf[]={0,0};//显示缓冲区
void delay(unsigned int time)
{
   while(time--);
}
//数码管动态显示函数
void disp(void)
{
        port_0=0xff;
        cs_1=1;
        cs_2=0;//选择驱动低位数码管的三极管导通
      port_0=table[dispbuf[0]];
        delay(5);

         port_0=0xff;
      cs_2=1;
      cs_1=0; //选择驱动高位数码管的三极管导通
      port_0=table[dispbuf[1]];
```

```
        delay(5);

    }
void main(void)
{

    dispbuf[0]=6;//将需要显示的数送进缓冲区
    dispbuf[1]=10;//将需要显示的数送进缓冲区
    while(1)
    {
        disp();
    }

}
```

3. 编译与仿真

图 2-3-7　正确的编译结果

仿真结果如图 2-3-8 所示。

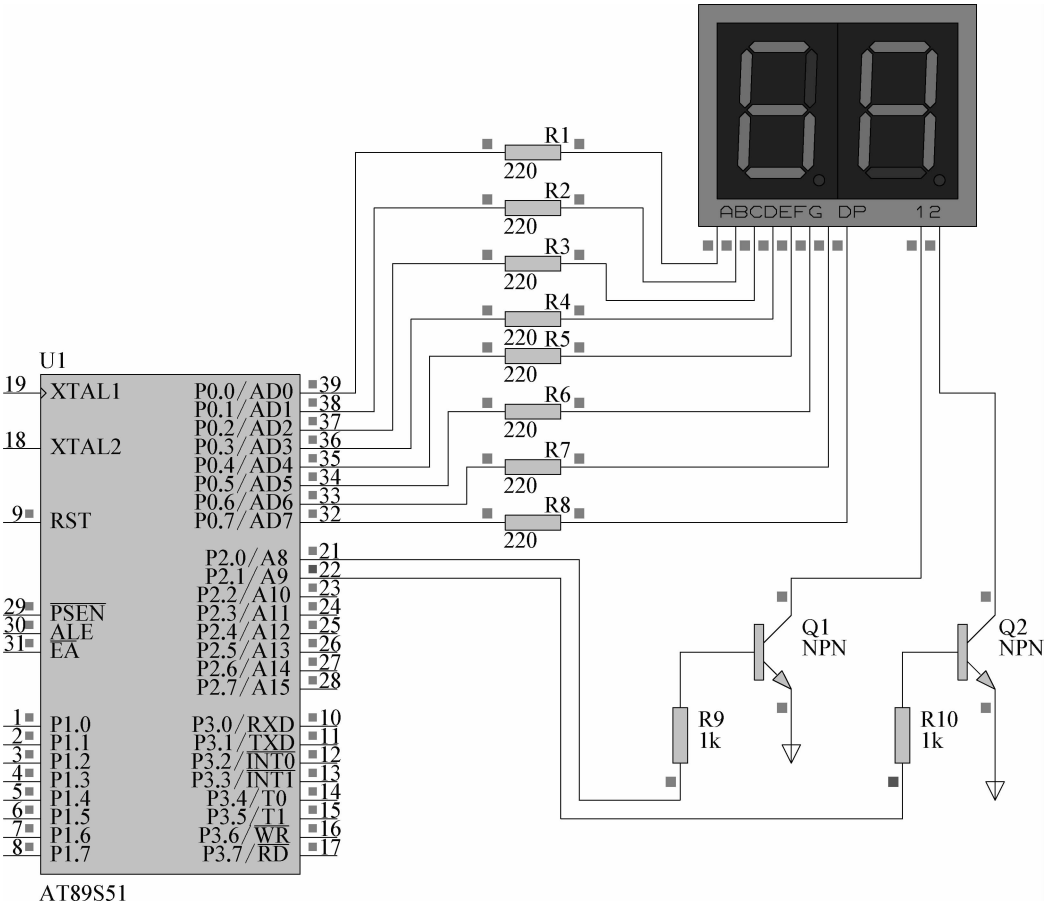


图 2-3-8　仿真结果

知识拓展

试修改程序并仿真，看能否动态显示 59 到 00 倒计数。

目标检测

1. 叙述多位 LED 数码管扫描显示工作过程。
2. 结合项目一中的 delay_ms()延时 1 秒钟函数，完成数码管 60 秒钟计时。
3. 编写程序，实现数码管闪烁。

任务四　按键计数器制作与调试

知识准备

本次任务是在前面几个任务的基础上，制作一个按键计数器电路，并能够根据控制要求编写单片机程序。具体的功能要求为：系统刚上电时，数码管显示“00”，每次按动加法键，数码管显示数据增加 1，每次按动减法按钮，显示数据减 1，通过按键设置，让数码管显示范围在“00～20”之间变化。

一、绘制原理图

在 Proteus 软件中绘制的原理图，如图 2－4－1 所示。

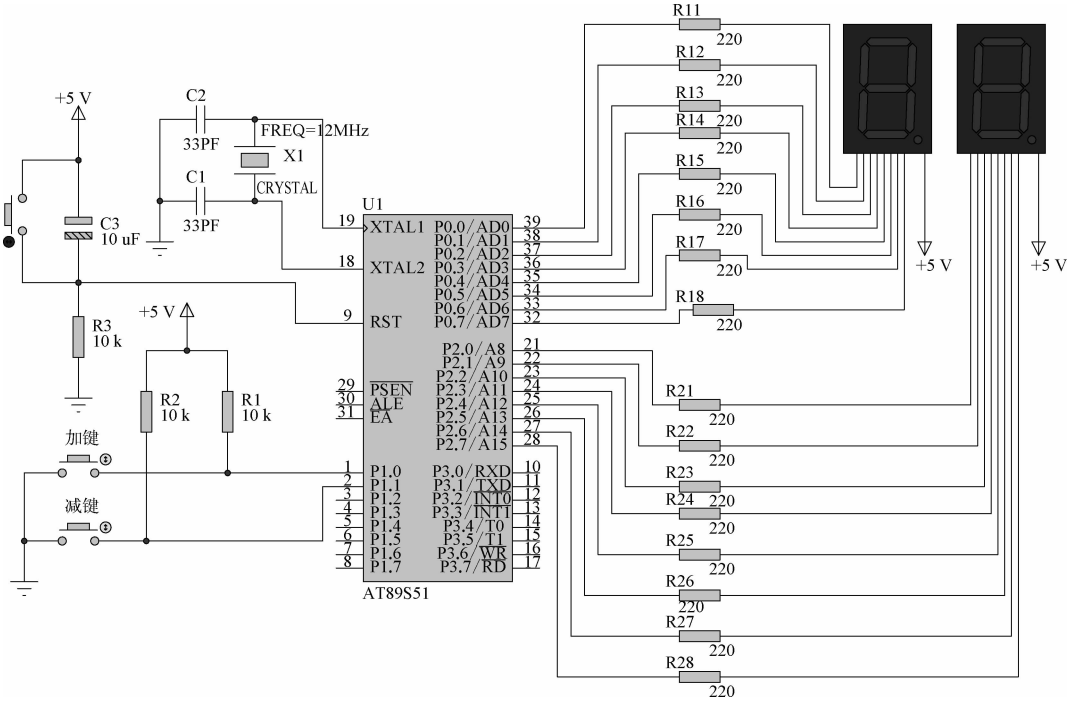


图 2－4－1　仿真原理图

二、编写源程序

根据原理图，编写源程序如下：

```
/*************************************************************
        按键计数器
  高位数码管的输入端接在 P0
  低位数码管的输入端接在 P2
  加法键接在 P1_0 上
  减法键接在 P1_1 上
*************************************************************/
#include"reg52.h"
#define port_0 P0
#define port_2 P2
sbit key_jia=P1^0;
sbit key_jian=P1^1;
unsigned char code table[]={
        0xC0,/*0*/
        0xF9,/*1*/
        0xA4,/*2*/
        0xB0,/*3*/
        0x99,/*4*/
        0x92,/*5*/
        0x82,/*6*/
        0xF8,/*7*/
        0x80,/*8*/
        0x90/*9*/
};
unsigned char count;
bit jia_flag,jian_flag;//jia_flag 加法按键标记,jian_flag 减法按键标记
/*************************************************************
        延时函数
延时时间=time*1 毫秒

*************************************************************/
void delay_time(unsigned int time)
{
unsigned int i,j;
for(i=0;i<time;i++)
```

```
for(j=0;j<121;j++);
}

void key_process(void)
{
    if(jia_flag==1)//加法键按下标记=1时,执行加法任务
    {
       jia_flag=0; // 清除加法键标记
       count++;//按一下计数加1
       if(count==21) //控制在显示20后,再按下时回到0
       count=0;
    }
   if(jian_flag==1)//减法键按下标记=1时,执行加法任务
    {
       jian_flag=0;//按一下计数减1
       count--;
       if(count==0xff) //要求显示0后,再按一下回到20.
       count=20;
}
}
/**************************************************************
   按键函数
**************************************************************/
void key_scan(void)
{
    unsigned char key_in;
    key_in=port_1&0x03;//读取按键的低两位,key=0000 00XX

if(key_in!=0x03)//初次判断有没有按键按下,如果key_in=0000 0011说明没有键按下
      {
            delay_time(10);//如果按下,延时去抖动
            key_in=port_1&0x03; //去抖动结束后,再次读取按键的低两位
            if(key_in!=0x03)//去抖动后再次判断有没有按下,如果按下,下面开始查询是哪个键按下
              {
```

```
                switch(key_in)//
                {
            case 0x0 2:jia_flag=1;break;//如果 key_in=0000 0010 说明是加法
                键按下
            case 0x0 1:jian_flag=1;break;//如果 key_in=0000 0001 说明是减
                法键按下
              default:break;
                }
              key_process();//执行按键按下,相关功能
              port_2=table[count%10];//显示按下按键的相关信息
              port_0=table[count/10];
              while((port_1&0x03)! =0x03);//等待按键松开
            }
        }
}

void task_key(void)
{
    key_scan();

}
void main(void)
{
port_2=table[0];//开机显示 00
port_0=table[0];//开机显示 00
while(1)
{


  task_key();//按键查询
}
}
```

三、编译与仿真

将上述源程序在 KEIL C 中编译并生成 HEX 文件,在 PROTEUS 中作原理图仿真。正确的编译结果如图 2-4-2 所示。

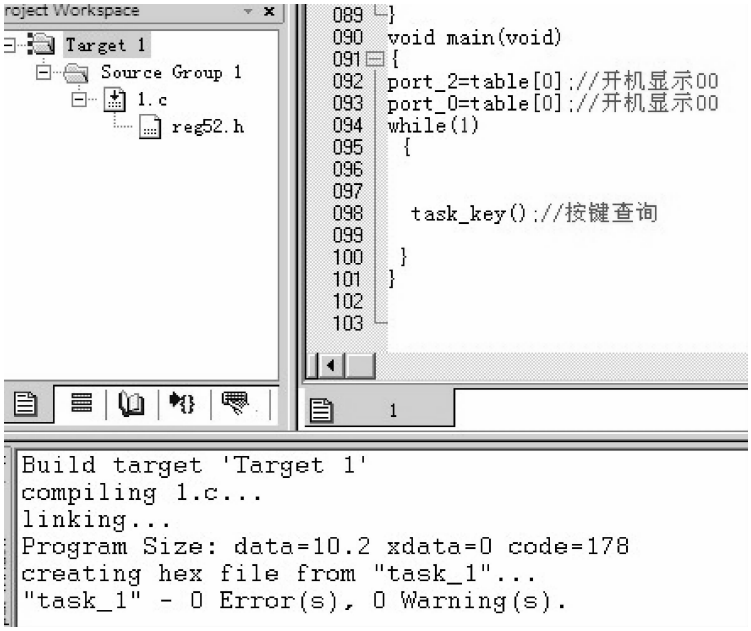


图 2－4－2　编译结果

PROTEUS 对单片机的仿真结果如图 2－4－3 所示。

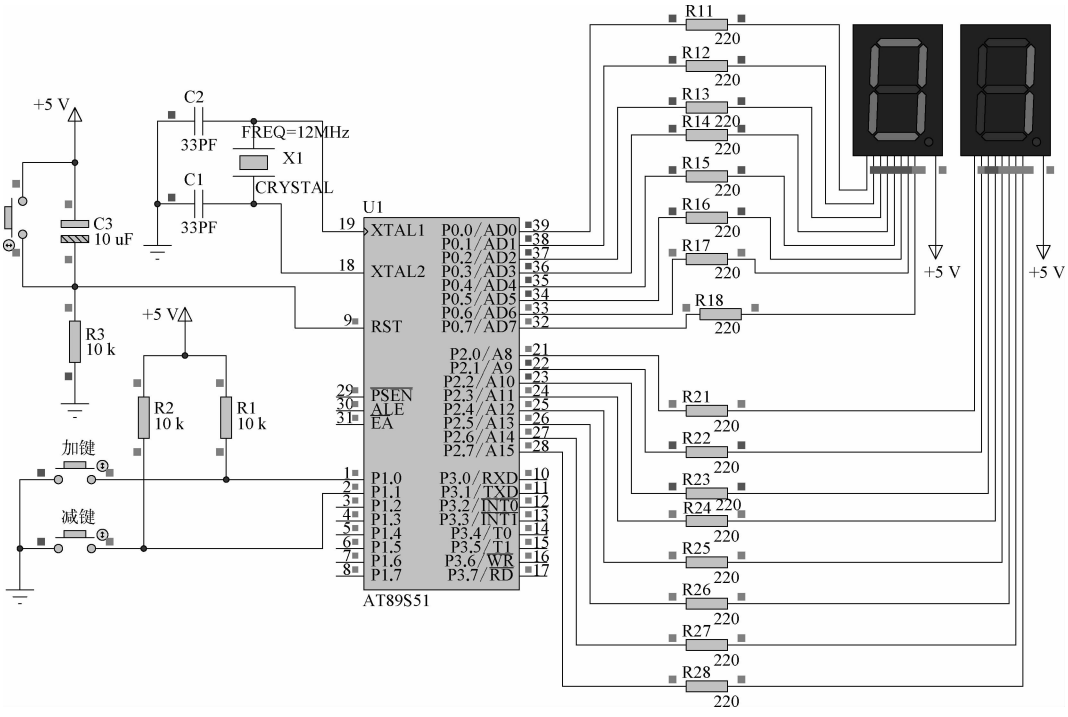


图 2－4－3　仿真结果

四、实物展示

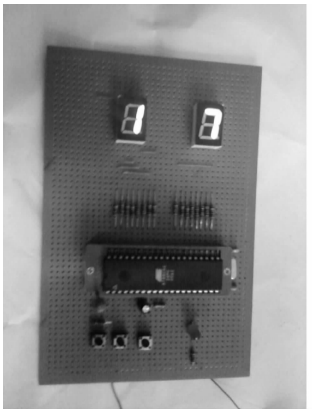


图 2－4－4　显示 17

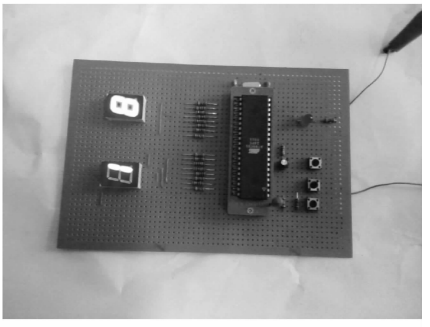


图 2－4－5　显示 18

知识拓展

尝试修改程序，等按键松开后，让显示的数值发生改变。

目标检测

一、填空题

1. LED 数码显示按显示过程分为_____显示和_____显示两种。前者，对每位显示来说是连续显示的，显示亮度较高；后者，对每位显示来说是断续显示，显示亮度较低，功耗较低。

2. 键盘需要通过接口电路与单片机相连，编程时通常采用_____或_____方式检查有无按键按下并识别有效键。

3. 按键较少时，通常采用每一个按键占用一个 I/O 口线，按键较多时，通常采用_____式键盘。

4. LED 数码管按其公共端的连接，有共_____极和共_____极两种。

5. LED 数码管，共_____极型的公共端应接地，共_____极型的公共端接高电平。

6. LED 动态显示时，扫描频率较高时，眼睛感觉不出抖动的原因是人的眼睛有_____效应。

二、选择题

1. 根据视觉暂留效应，动态扫描驱动 8 个 LED，每一位的时间间隔不大于_____即能不闪烁又不占用过多的时间　　（　　）

A. 2.5 μs　　B. 20 μs　　C. 2.5 ms　　D. 20 ms

2. LED显示内容与显示代码存在一定的转换关系　　(　　)

A. 显示内容与显示代码没关系

B. 显示内容与显示代码通过简单转换得到

C. 显示内容与显示代码可转可不转换

D. 显示内容与显示代码通过复杂计算转换得到

3. 动态接口技术应是　　(　　)

A. 硬件简单,软件复杂　　B. 软件复杂

C. 硬件复杂　　D. 硬件简单,软件只要几条语句即可

4. 常用键盘/显示接口芯片有　　(　　)

A. LS373,8255,8279　　B. 8155,8279

C. 8279,8255　　D. 8155,8255

5. 为防止按键按住不放而产生多次键处理,其防范的方法是　　(　　)

A. 当判断键按下时,系统处理一次后退出

B. 当判断键按下后,系统等待键释放后再处理

C. 当判断键按下后不做处理,等待键释放再处理

D. 当判断键按下后,系统判断键释放确认一次按键成功,若一直没释放,则判断是否超时并退出

6. 已知1只共阴极LED显示器,其中a笔段为字形代码的最低位,若需显示数字1,它的字形代码应为　　(　　)

A. 06H　　B. F9H　　C. 30H　　D. CFH

三、编程题

1. 用共阴数码管完成00～59秒循环计时(动态和静态两种显示)。

2. 编写程序:按个位按键,实现对个位数码管数字的设定;按十位按键,实现对十位数码管数字的设定。请设定显示“68”,按键与数码管与单片机接口自定义。

3. 编写程序:按“设置”键,进入数码管设定状态,数码管闪烁,再按“数字”键,调节显示数字“00”到“99”,再按一下“设置”键,退出设定状态,数码管显示设定的数值。请设定数码管显示“15”,按键及数码管与单片机接口自定义。