**《51单片机PCB设计与编程仿真》**

**专业班级： 电信16101班**

**设 计 者： Sophie**

**设计时间： 2018-7-1**

目录

[电子技术课程综合设计报告 1](#_Toc518407748)

[一、 设计题目 3](#_Toc518407749)

[二、 主要指标和要求 3](#_Toc518407750)

[三、 方案选择及工作原理 3](#_Toc518407751)

[1.51单片机最小系统 3](#_Toc518407752)

[2.流水灯（P1.0~P1.7） 5](#_Toc518407753)

[3.矩阵键盘(P3.0~P3.7) 6](#_Toc518407754)

[4．数码管60秒计时(P0.0~P0.7) 7](#_Toc518407755)

[四、 各模块及顶层文件的设计等 8](#_Toc518407756)

[1．流水灯的实现 8](#_Toc518407757)

[2. 数码管计时实现 9](#_Toc518407758)

[3. 矩阵键盘的实现 12](#_Toc518407759)

[五、 仿真、调试中遇到的问题，解决的方法以及实验效果等 18](#_Toc518407760)

[六、收获、体会和改进设计的建议 18](#_Toc518407761)

## 设计题目

本次课题主要研究51单片机及其应用，主要内容包括：单片机最小系统、流水灯的实现、矩阵键盘、数码管60秒倒计时。

## 主要指标和要求

课题的实验主要借助protues 进行PCB电路图设计与仿真，使用

keil uVison4进行代码编写与编译。

## 方案选择及工作原理

### 1.51单片机最小系统

最小系统作为单片机的心脏，在开发伊始我们就需要将它设计好。单片机最小系统是指用最少的元件组成的单片机可以工作的系统。对51系列单片机来说，最小系统一般应该包括:单片机、复位电路、晶振电路 。

#### 1.0单片机

在原件库中选取这款AT89C52单片机，它包含有40个引脚，如右图所示，注意在此图中隐去了20引脚（GND接地端）和40引脚（VCC电源端）。引脚9（RST）为复位引脚，在后面设计复位电路时会用到，引脚18,19用来接外部晶振，后面也会用到，其他的P端口为单片机的I/O口，部分端口还可复用为其他功能



#### 1.1复位电路

在元件库中选取一个10uF的电容和10K的电阻以及一个按钮。

复位电路的用途：当单片机系统在运行中，受到环境干扰出现程序跑飞的时候，按下复位按钮内部的程序自动从头开始执行。

51单片机复位原理：51单片机要复位只需要在第9引脚(RST)接高电平持续2US就可以实现 ,因此我们可以通过电容的充放电来实现这一过程。



#### 1.2.晶振电路

在元件库中选取2个30pF的电容和一个11.0592MHz的晶振。单片机晶振的作用是为系统提供基本的时钟信号 。



### 2.流水灯（P1.0~P1.7）



使用单片机P1组引脚(P1.0~P1.7)来控制流水灯模块,在该模块中我们需要以下元器件，一个8线排阻，8个LED灯，一个74HC573锁存器。在该模块中我们也可以将LED直接与单片机引脚相连，但考虑到将来该组引脚还会用于其他功能，所以我们用一个74HC573锁存器作为中间连接层，在该组引脚需要进行其他功能时，我们通过该锁存器关闭Q端信号输出即可使流水灯不乱蹿。

### 3.矩阵键盘(P3.0~P3.7)



使用单片机P3组引脚(P3.0~P3.7)来控制矩阵键盘模块,在该模块中我们需要以下元器件：16个按钮即可。

矩阵键盘又称行列键盘，它是用四条I/O线作为行线，四条I/O线作为列线组成的键盘。在行线和列线的每个交叉点上设置一个按键。这样键盘上按键的个数就为4\*4个。这种行列式键盘结构能有效地提高单片机系统中I/O口的利用率。

矩阵键盘检测原理：检测时，先送入一列低电平，其余几列全为高电平(此时确定了列数)，然后立即轮流检测一次各行是否有低电平，若检测到某一行为低电平（此时又确定了行数），则我们便可确认当前按下的是哪行哪列的键了，用同样的方法轮流送各列一次低电平，轮流检测一次各行是否变为低电平，这样即可检测完所有的按键，当按键被按下时便可知道是哪一个按键被了。

### 4．数码管60秒计时(P0.0~P0.7)



使用单片机P0组引脚(P0.0~P0.7)以及P2.6和P2.7引脚来控制数码管模块,在该模块中我们需要以下元器件：1个2位7段共阳数码管，两个74HC573锁存器。在该实验中锁存器U3用于控制段选，U4用于控制位选，注意此时P0口同时接了10K的上拉电阻

## 各模块及顶层文件的设计等

### 1．流水灯的实现

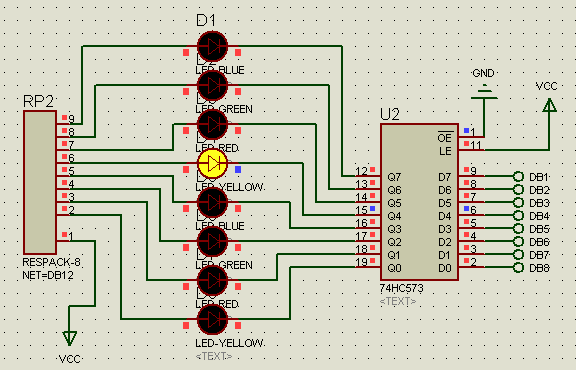
在前已经实现了电气连接部分，现在我们开始设计并实现流水灯的功能，主要工作为在Keil uVision4中编写、编译功能代码。

#### 代码实现：

实现流水灯的方式有很多，在这里我使用了intrins.h库中提供的左、右移函数很方便的实现了位移操作。

|  |
| --- |
| #include<reg52.h>  #include<intrins.h>  //延时函数  void delay(unsigned char i)  {  unsigned char m,n;  for(m=i;m>0;m--)  for(n=125;n>0;n--);  }  void main()  { unsigned char k;  while(1)  {  P1=0xfe;  for(k=0;k<8;k++)  {  delay(2000);  P1=\_crol\_(P1,1);  }  }  } |

#### 仿真结果：



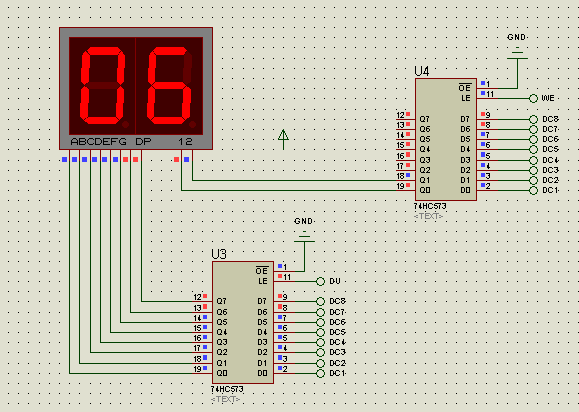
### 2. 数码管计时实现

在这里我使用了单片机的一个定时器来实现计时功能

#### 代码实现：

|  |
| --- |
| #include<reg51.h>  #define uchar unsigned char  sbit dula=P2^6; //申明U1锁存器的锁存端  sbit wela=P2^7; //申明U2锁存器的锁存端  //j、k用于声明延时变量  //c1、c0分别用于存储显示个位和十位的值  unsigned char j,k,c1,c0,m,n=255;  unsigned char pp;  unsigned char code table[]={0xc0,0xf9,0xa4,0xb0,//0~3  0x99,0x92,0x82,0xf8,//4~7  0x80,0x90,0x88,0x83,//8~b  0xc6,0xa1,0x86,0x8e //c~f  };  void delay(unsigned char i)  {  for(j=i;j>0;j--)  for(k=125;k>0;k--);  }    void display(uchar a,uchar b)  {  dula=1;  P0=table[a];  delay(5);  dula=0;  wela=1;  P0=0xfe;  wela=0;    dula=1;  P0=table[b];  delay(5);  dula=0;  //开启位选  wela=1;  P0=0xfd;  wela=0;  }  void main()  {  TMOD=0x01; //模式设置，00000001，可见采用的是定时器0，工作与模式1（M1=0，M0=1）。  TR0=1; //打开定时器  TH0=(65536-46080)/256;// 由于晶振为11.0592,故所记次数应为46080，计时器每隔50000微秒发起一次中断。  TL0=(65536-46080)%256;//46080的来历，为50000\*11.0592/12  ET0=1; //开定时器0中断  EA=1; //开总中断  while(1)  {  //1秒钟过去  if(pp==20)  {  //将记录1秒是否到的变量归零  pp=0;  //秒数加1  m++;  n--;  P1=n;//闪烁灯  if(m==60)  {  m=0; //若到了60s，则归零  }  }  c0=m%10; //取出当前描述的个位与十位  c1=m/10;  display(c1,c0); //显示  }  }  void time0() interrupt 1  {TH0=(65536-46080)/256;  TL0=(65536-46080)%256;  pp++;  } |

#### 仿真结果



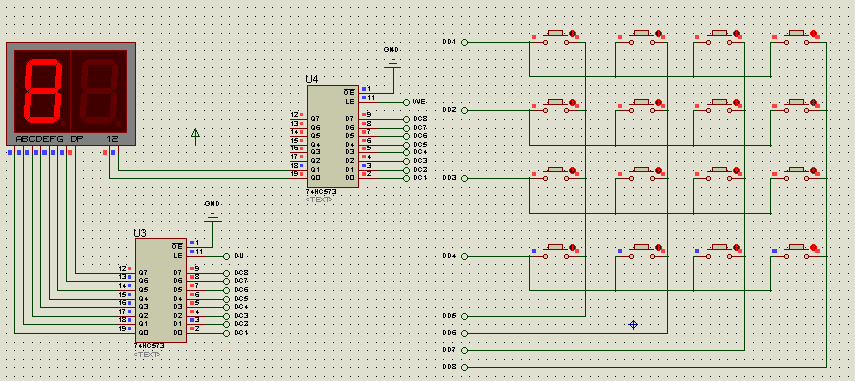
### 3. 矩阵键盘的实现

在前面。矩阵键盘的电气连接已经完成，键盘扫描检测的原理也分析了一下。现在我们来具体实现一下，

#### 代码实现：

|  |
| --- |
| #include<reg51.h>  #define uchar unsigned char  sbit dula=P2^6; //申明U1锁存器的锁存端  sbit wela=P2^7; //申明U2锁存器的锁存端  //j、k用于声明延时变量  //c1、c0分别用于存储显示个位和十位的值  unsigned char j,k,c1,c0,m,n=255;  unsigned char pp;  unsigned char code table[]={0xc0,0xf9,0xa4,0xb0,//0~3  0x99,0x92,0x82,0xf8,//4~7  0x80,0x90,0x88,0x83,//8~b  0xc6,0xa1,0x86,0x8e //c~f  };  void delay(unsigned char i)  {  for(j=i;j>0;j--)  for(k=125;k>0;k--);  }  void display(uchar a,uchar b)  {    dula=1;  P0=table[a];  delay(5);  dula=0;  wela=1;  P0=0xfe;  wela=0;  dula=1;  P0=table[b];  delay(5);  dula=0;  //开启位选  wela=1;  P0=0xfd;  wela=0;  //delay(10);  }  //矩阵键盘扫描  void matrixkeyscan()  {  uchar temp,key;  P3=0xfe; //扫描第一行按键  temp=P3;  temp=temp&0xf0; //判断temp高4位是否有0，如果temp的高4位有0那么 temp的高4位数据实际上就是矩阵键盘的4个列线  if(temp==0xf0)  {  delay(10);  temp=P3;  temp=temp&0xf0;  if(temp!=0xf0)  {  temp=P3;  switch(temp)  {  case 0xee:  key=0;  break;  case 0xde:  key=1;  break;  case 0xbe:  key=2;  break;  case 0x7e:  key=3;  break;  }  while(temp!=0xf0)  {  temp=P3;  temp=temp&0xf0;  }  c0=key%10; //取出当前描述的个位与十位  c1=key/10;  display(c1,c0); //显示  }    }  P3=0xfd; //扫描第二行按键  temp=P3;  temp=temp&0xf0; //判断temp高4位是否有0，如果temp的高4位有0那么 temp的高4位数据实际上就是矩阵键盘的4个列线  if(temp==0xf0)  {  delay(10);  temp=P3;  temp=temp&0xf0;  if(temp!=0xf0)  {  temp=P3;  switch(temp)  {  case 0xed:  key=4;  break;  case 0xdd:  key=5;  break;  case 0xbd:  key=6;  break;  case 0x7d:  key=7;  break;  }  while(temp!=0xf0)  {  temp=P3;  temp=temp&0xf0;  }  c0=key%10; //取出当前描述的个位与十位  c1=key/10;  display(c1,c0); //显示  }    }  P3=0xfb; //扫描第三行按键  temp=P3;  temp=temp&0xf0; //判断temp高4位是否有0，如果temp的高4位有0那么 temp的高4位数据实际上就是矩阵键盘的4个列线  if(temp==0xf0)  {  delay(10);  temp=P3;  temp=temp&0xf0;  if(temp!=0xf0)  {  temp=P3;  switch(temp)  {  case 0xeb:  key=8;  break;  case 0xdb:  key=19;  break;  case 0xbb:  key=10;  break;  case 0x7b:  key=11;  break;  }  while(temp!=0xf0)  {  temp=P3;  temp=temp&0xf0;  }  c0=key%10; //取出当前描述的个位与十位  c1=key/10;  display(c1,c0); //显示  }    }  P3=0xf7; //扫描第四行按键  temp=P3;  temp=temp&0xf0; //判断temp高4位是否有0，如果temp的高4位有0那么 temp的高4位数据实际上就是矩阵键盘的4个列线  if(temp==0xf0)  {  delay(10);  temp=P3;  temp=temp&0xf0;  if(temp!=0xf0)  {  temp=P3;  switch(temp)  {  case 0xe7:  key=12;  break;  case 0xd7:  key=13;  break;  case 0xb7:  key=14;  break;  case 0x77:  key=15;  break;  }  while(temp!=0xf0)  {  temp=P3;  temp=temp&0xf0;  }  c0=key%10; //取出当前描述的个位与十位  c1=key/10;  display(c1,c0); //显示  }    }  }  //实现矩阵键盘扫描  void main()  {  while(1)  {  matrixkeyscan();  }    } |

#### 仿真结果：



## 仿真、调试中遇到的问题，解决的方法以及实验效果等

在实验过程中，遇到的主要问题有两方面的：电路连接、代码编写。在电路连接方面主要体现在标号错位导致仿真结果与预期结果相差甚远。代码编写上的问题主要是在实现数码管60s计时上，由于同一个I/O需要控制两种情况——段选和位选，也就是进位的显示控制。这两种情况有需要不断的判断并切换状态，因此需要把控好显示个位与显示十位的时机，在这两者切换时可给适当延时，不然个位和十位有可能就串线了造成显示混乱。

## 收获、体会和改进设计的建议

通过本次课题实验，熟悉了单片机基本开发流程，加深了对单片机以及定时器中断的理解，同时也在编程上也又复习了一遍C语言。该次实验基本达到了实验前制定的指标和要求，但也存在需要改进的地方，比如电路图的绘制还可以更规范简洁点，代码也可以写的更精简点。