专业选修课3实践案例教学

毕业设计答辩计时及评分系统的设计

|  |  |
| --- | --- |
| 姓名： | 姜程鑫 |
| 班级： | 13电子一 |
| 学号： | 13020713 |
| 同组成员： | 彩云 |
|  |  |
| 指导教师： | 姚文卿 |
| 报告成绩： |  |
| 答辩成绩： |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 2017 | 年 | 1 | 月 |

目录

[概述 1](#_Toc473204749)

[第一章 绪 论 2](#_Toc473204750)

[1.1设计的主要内容 2](#_Toc473204751)

[1.2设计的总体要求 2](#_Toc473204752)

[1.3成员分工 2](#_Toc473204753)

[第二章 设计方案 3](#_Toc473204754)

[2.1方案比较 3](#_Toc473204755)

[2.1.1 每个数码管均由单片机I/O同时控制，实现动态扫描 3](#_Toc473204756)

[2.1.2 利用锁存器实现数码管动态扫描 3](#_Toc473204757)

[2.2电路总体方案 3](#_Toc473204758)

[2.3设计流程图 4](#_Toc473204759)

[第三章 单片机概述 5](#_Toc473204760)

[3.1 STC89C52单片机简介 5](#_Toc473204761)

[3.2 STC89C52主要特性 5](#_Toc473204762)

[3.3 STC89C51外部结构及特性 6](#_Toc473204763)

[第四章 主要元件介绍 8](#_Toc473204764)

[4.1 SN74HC573锁存器 8](#_Toc473204765)

[4.1.1 SN74HC573介绍 8](#_Toc473204766)

[4.1.2 SN74HC573锁存器引脚接法和电气参数 8](#_Toc473204767)

[4.2 数码管动态扫描 10](#_Toc473204768)

[第五章 系统软件部分设计 11](#_Toc473204769)

[5.1 Keil软件程序设计 11](#_Toc473204770)

[5.1.1 程序框图 11](#_Toc473204771)

[5.1.2 延时子程序 12](#_Toc473204772)

[5.1.3 数码管动态扫描电路 12](#_Toc473204773)

[5.2 Proteus软件仿真 13](#_Toc473204774)

[第六章 系统硬件电路设计 15](#_Toc473204775)

[6.1单片机最小系统 15](#_Toc473204776)

[6.1.1 STC89C52芯片 15](#_Toc473204777)

[6.1.2 复位电路 15](#_Toc473204778)

[6.1.3 晶振电路 16](#_Toc473204779)

[6.2 数码管动态扫描电路 17](#_Toc473204780)

[6.3 键盘扫描电路 17](#_Toc473204781)

[行列式键盘工作原理 18](#_Toc473204782)

[设计心得 20](#_Toc473204783)

[参考文献 21](#_Toc473204784)

[附录 22](#_Toc473204785)

[附录一 仿真电路 22](#_Toc473204786)

[附录二 Protel电路原理图 23](#_Toc473204787)

[附录三 C语言设计代码 24](#_Toc473204788)

[附录四 实物图 32](#_Toc473204789)

**概述**

单片机设计实践是对所学知识的巩固和升华，我们这次做的“毕业设计答辩计时及评分系统的设计”，是对很多单片机的所学知识应用的集合，虽然在程序设计和实物制作上有一定难度，但有挑战才会有进步，才会对知识有更好的掌握。

课题《毕业设计答辩计时及评分系统的设计》由52单片机最小系统、复位电路、数码管动态显示模块、锁存器控制模块和行列式键盘电路等组成。通过全程进行数码管的动态扫描，对不同阶段的数据进行显示，与此同时，不同的阶段对应着不同的LED灯，在不同状态下，实现LED的点亮与熄灭，这一切还需要配合4x4键盘的不断扫描，读取键值来实现控制。

本次设计是直接在蓝桥杯单片机开发板CT-107D平台进行设计。

**第一章 绪 论**

**1.1设计的主要内容**

课题《毕业设计答辩计时及评分系统的设计》由52单片机最小系统、复位电路、数码管动态显示模块、锁存器控制模块和行列式键盘电路等组成。通过全程进行数码管的动态扫描，对不同阶段的数据进行显示，与此同时，不同的阶段对应着不同的LED灯，在不同状态下，实现LED的点亮与熄灭，这一切还需要配合4x4键盘的不断扫描，读取键值来实现控制。具体要求如下：

1、使用8位数码管显示同组中其中一位同学的学号，表示由该同学参加答辩，复位后常亮。

2、用2位数码管显示剩余分钟，复位后显示10表示设定10分钟，设置开始/取消按键，按下一次开始倒计时，再按一次则复位，时间到则蜂鸣音提示。

3、设置“打分”按键，按下一次后开始打分操作，此时停止倒计时并关闭蜂鸣器；给出十个按键分别表示数字0～9，此时由2位答辩老师分别给出答辩成绩，按下后所按数字显示在2位数码管上，每次给出成绩后按下打分按键表示确认。等第二位老师给出成绩并按打分键确认后，2位数码管显示出最终成绩（2位老师评分的平均值四舍五入）。

\*说明：以上数码管显示部分也可选用LCD显示。

**1.2设计的总体要求**

1、方案论证，确定总体电路原理图。

2、元器件选择，设计PCB图（或用万能电路实验板搭线）。

3、绘制程序流程图，编写C语言源程序。

4、安装调试，实现基本功能。

**1.3成员分工**

|  |  |
| --- | --- |
| 成员 | 任务 |
| 蔡云 | Tools.h文件的编写，负责数码管段选位选、led、蜂鸣器等控制开关函数的编写，完成代码编写的底部构架；Proteus仿真原理图的绘制。 |
| 姜程鑫 | Main.c文件的编写，负责整个设计的整合和main函数的编写，绘制原理图，以及实物作品的调试。 |

**第二章 设计方案**

**2.1方案比较**

**2.1.1 每个数码管均由单片机I/O同时控制，实现动态扫描**

提到动态扫描，无非就是控制数码管的段选和位选，利用人眼的视觉暂留效应，实现动态的显示。而单片机本身有4个I/O口，可以考虑每一个口控制一个数码管，这么做省去了其他元器件的辅助，也就保证了电路的稳定性，使得理论和实际的差距更小，制作过程中遇到的问题也会相对较少。

**2.1.2 利用锁存器实现数码管动态扫描**

除了直接通过单片机的I/O口来实现数码管的动态扫描，开可以利用锁存器。使用两个锁存器，一个控制位选，一个控制段选，这样一来，就只需要使用到单片机的一组I/O口，非常的节省系统资源。但是，势必会使得电路的复杂性上升，除了要保证理论上的正确，更重要的是要解决实际制作中的问题，因为电路的复杂肯定会使得产品的稳定性降低，各种小问题也会层出不穷！

**2.2电路总体方案**

本次设计中，我们要使用8个数码管来显示选手的学号、倒计时时间、老师打分以及最后得分的信息，8位段选，8位位选，占用2组I/O口；4x4的矩阵键盘也需要占用一组I/O口，led的控制也要占用一组I/O口，加上蜂鸣器的控制等等，可见I/O口的分配显得很紧凑，甚至出现端口分配不足的情况。所以决定选择方案二，利用锁存器，将数码管的段选信号、位选信号、LED控制信号、蜂鸣器控制信号统一从P0口输出，只要通过P2口控制相应的锁存器的状态即可，矩阵键盘接P3口。这样一来大大节省了端口空间，资源利用更加高效。

图2-1是电路总体框图，包括52单片机最小系统，数码管动态显示模块，键盘扫描电路，蜂鸣器电路等。

数码管动态显示

键盘扫描

LED控制

STC89C52

分数处理

蜂鸣器控制

5V USB直流稳压电

图2-1 电路基本框图

**2.3设计流程图**

整个设计的运行流程如下（图2-2）：



图2-2 设计流程图

**第三章 单片机概述**

**3.1 STC89C52单片机简介**

STC89C52[1]单片机是宏晶科技推出的新一代高速/低功耗/超强抗干扰的单片机，指令代码完全兼容传统8051单片机，12时钟/机器周期和6时钟/机器周期可以任意选择。

**3.2 STC89C52主要特性**

1.具有6时钟/机器周期和12时钟/机器周期可以任意选择，指令代码完全兼容传统8051.

2.工作电压：5.5V～3.3V（5V单片机）/3.8V～2.0V（3V单片机）

3.工作频率范围：0～40MHz，相当于普通8051的0～80MHz，实际工作频率可达48MHz

4.用户应用程序空间为8K字节

5.片上集成512字节RAM

6.通用I/O口（32个），复位后为：P1/P2/P3/P4是准双向口/弱上拉，P0口是漏极开路输出，作为总线扩展用时，不用加上拉电阻，作为I/O口用时，需加上拉电阻。

7.ISP（在系统可编程）/IAP（在应用可编程），无需专用编程器，无需专用仿真器，可通过串口（RxD/P3.0,TxD/P3.1）直接下载用户程序，数秒即可完成一片

8.具有EEPROM功能

9.具有看门狗功能

10.共3个16位定时器/计数器。即定时器T0、T1、T2

11.外部中断4路，下降沿中断或低电平触发电路，Power Down模式可由外部中断低电平触发中断方式唤醒

12.通用异步串行口（UART），还可用定时器软件实现多个UART

13工作温度范围：-40～+85℃（工业级）/0～75℃（商业级）

14.PDIP封装

**3.3 STC89C51外部结构及特性**

其外形封装有两种方式：双列直插式40脚封装（DIP）和方形44脚封装 （PLCC），直插式40 脚封装（DIP）如图3-1所示：

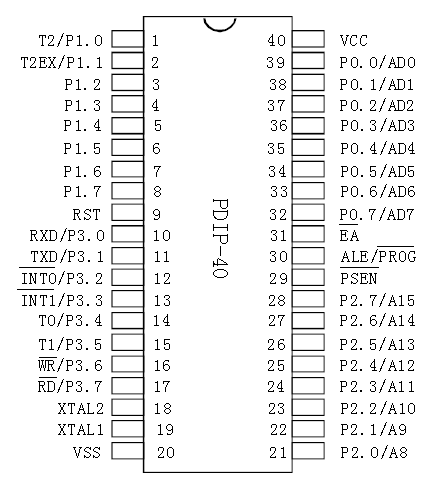


图3-1 STC89C51引脚排列

STC89C52[1]的 4 个 8 位I/O口的功能说明如下：

（1）P0口：P0 口是一个 8 位漏极开路的双向 I/O 口。作为输出口，每位能驱动8个TTL逻辑电平。对P0端口写“1”时，引脚用作高阻抗输入。 当访问外部程序和数据存储器时，P0口也被作为低 8 位地址/数据复用。在这种模式下，P0具有内部上拉电阻。在flash 编程时，P0口也用来接收指令字节；在程序校验时，输出指令字节。程序校验时，需要外部上拉电阻。

（2）P1口：P1口是一个具有内部上拉电阻的8位双向 I/O口，p1输出缓冲器能驱动4个TTL 逻辑电平。对P1端口写“1”时，内部上拉电阻把端口拉高，此时可以作为输入口使用。作为输入使用时，被外部拉低的引脚由于内部电阻的原因，将输出电流（IIL）。此外，P1.0 和 P1.2 分别作定时器/计数器 2 的外部计数输入（P1.0/T2）和定时器/计数器 2 的触发输入（P1.1/T2EX）。在 flash 编程和校验时，P1口接收低8位地址字节。

（3）P2口：P2口是一个具有内部上拉电阻的8位双向I/O口，P2输出缓冲器能驱动 4 个 TTL 逻辑电平。对 P2端口写“1”时，内部上拉电阻把端口拉高，此时可以作为输入口使用。作为输入使用时，被外部拉低的引脚由于内部电阻的原因，将输出电流（IIL）。在访问外部程序存储器或用 16 位地址读取外部数据存储器（例如执行MOVX @DPTR）时，P2口送出高八位地址。在这种应用中，P2口使用很强的内部上拉发送 1。在使用8 位地址（如 MOVX @RI）访问外部数据存储器时，P2 口输出 P2 锁存器的内容。在 flash 编程和校验时，P2 口也接收高 8 位地址字节和一些控制信号。

（4）P3口：P3口是一个具有内部上拉电阻的8位双向I/O口，p2 输出缓冲器能驱动4个TTL逻辑电平。对P3端口写“1”时，内部上拉电阻把端口拉高，此时可以作为输入口使用。作为输入使用时，被外部拉低的引脚由于内部电阻的原因，将输出电流（IIL）。在 flash 编程和校验时，P3 口也接收一些控制信号。P3 口亦作为AT89C52特殊功能（第二功能）使用，如下所示：

 P3.0 RXD(串行输入口)

 P3.1 TXD(串行输出口)

 P3.2 INTO(外部中断0 输入口)

 P3.3 INT1(外部中断 1 输入口)

 P3.4 TO(定时器 0 外部输入)

 P3.5 TI(定时器 1 外部输入)

 P3.6 WR(外部数据存储器写选通信号)

 P3.7(外部数据存储器读选通信号)

**第四章 主要元件介绍**

**4.1 SN74HC573锁存器**

**4.1.1 SN74HC573介绍**

SN74HC573AN[2]是包含八路D 型的透明锁存器，每个锁存器具有独立的D 型输入，以及适用于面向总线的应用的三态输出。其封装引脚图如图4-1。

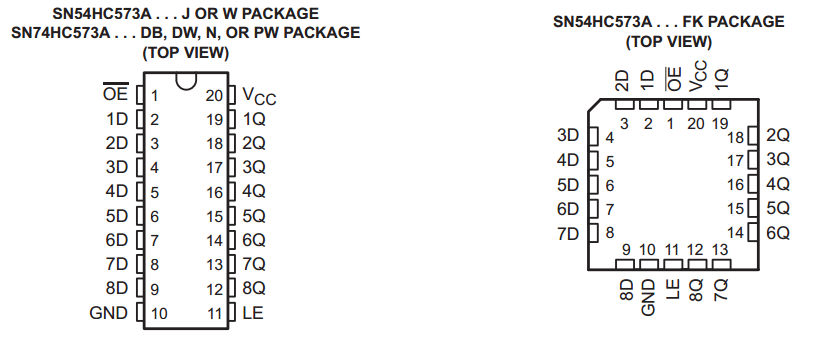


图4-1 SN74HC573AN封装引脚图

**4.1.2 SN74HC573锁存器引脚接法和电气参数**

**1.管脚简介**

SN74HC573AN[2]的外形及管脚排列如图3-1所示。

（1）VCC为5V电源；

（2）GND为地线；

（3）D控制信号输入；

（4）Q控制信号输出。

**2.SN74HC573AN的电气参数**

电气参数如表4-1所示：

表4-1 SN74HC573AN电气参数表

|  |  |
| --- | --- |
| 电气参数 | SN74HC573AN |
| 工作电压 | DC 2-6V |
| 输出高电平电流 | -7.8mA |
| 输出低电平电流 | 32Hz |
| 工作温区 | -40℃~85℃ |
| 最大传输延时 | 38ns |
| 驱动电流 | -6/6Ma |
| 逻辑电平 | CMOS |

SN74HC573AN[2]工作功能表如表4-2所示：

表4-2 SN74HC573AN工作功能表

|  |  |  |  |
| --- | --- | --- | --- |
| INPUT | | | OUTPUT |
| OE | LE | D |
| L | H | H | H |
| L | H | L | L |
| L | L | X | Q0 |
| H | X | X | Z |

只有当OE为低电平，LE为高电平时，输出等于输入，当LE变为低电平是，输入不再传输到输出，处于所存状态，故可以通过控制LE的电平来控制锁存器的工作状态。

**4.2 数码管动态扫描**

数码管要正常显示，就要用驱动电路来驱动数码管的各个段码，从而显示出我们要的数字，因此根据数码管的驱动方式的不同，可以分为静态式和动态式[3]两类。  
 ① 静态显示驱动：静态驱动也称直流驱动。静态驱动是指每个数码管的每一个段码都由一个单片机的I/O端口进行驱动，或者使用如BCD码二-十进制译码器译码进行驱动。静态驱动的优点是编程简单，显示亮度高，缺点是占用I/O端口多，如驱动5个数码管静态显示则需要5×8＝40根I/O端口来驱动，要知道一个89S51单片机可用的I/O端口才32个呢：），实际应用时必须增加译码驱动器进行驱动，增加了硬件电路的复杂性。  
 ② 动态显示驱动[4]：数码管动态显示接口是单片机中应用最为广泛的一种显示方式之一，动态驱动是将所有数码管的8个显示笔划"a,b,c,d,e,f,g,dp"的同名端连在一起，另外为每个数码管的公共极COM增加位选通控制电路，位选通由各自独立的I/O线控制，当单片机输出字形码时，所有数码管都接收到相同的字形码，但究竟是那个数码管会显示出字形，取决于单片机对位选通COM端电路的控制，所以我们只要将需要显示的数码管的选通控制打开，该位就显示出字形，没有选通的数码管就不会亮。通过分时轮流控制各个数码管的的COM端，就使各个数码管轮流受控显示，这就是动态驱动。在轮流显示过程中，每位数码管的点亮时间为1～2ms，由于人的视觉暂留现象及发光二极管的余辉效应，尽管实际上各位数码管并非同时点亮，但只要扫描的速度足够快，给人的印象就是一组稳定的显示数据，不会有闪烁感，动态显示的效果和静态显示是一样的，能够节省大量的I/O端口，而且功耗更低。

**第五章 系统软件部分设计**

本次设计在软件仿真的基础上进行实物制作，软件仿真可谓至关重要。

我们采用Keil软件和Proteus仿真软件进行软件部分的设计，Keil软件用于编程实现功能，Proteus软件用于仿真，检测理论设计的正确性。

**5.1 Keil软件程序设计**

系统程序主要包括主程序、延时子程序、数码管动态扫描子程序、键盘扫描子程序、蜂鸣器控制子程序和定时器中断服务子程序等（见附录3）。

**5.1.1 程序框图**

程序流程图如图5-1所示：



图5-1主流程图

主程序是实现数码管的动态显示，当输入学号按下“确认”按键后，是数码管后四位显示10分钟的倒计时，这一部分是利用定时器中断来实现倒计时，相比利用软件延时，采用定时器硬件中断，其精度更加高，且更稳定。

**5.1.2 延时子程序**

在这个设计中，为了程序的简洁并没有引入定时器中断进行计时，而是利用软件定时。延时子程序（见附件2）如下：

void delay(uint x) //延时函数，延时xms

{

uint i,j;

for (i=x; i>0; i--)

for(j=110;j>0;j--);

}

本端程序实现延时x毫秒，具体延时情况与单片机的晶振频率有关，我们在Keil软件的调试中，通过单步执行（或者通过计算t=1/f）得到执行一条程序的时间（一个机器周期），进而计算出1ms是需要多少个机器周期，从而写出该延时程序。

**5.1.3 数码管动态扫描电路**

由于在程序中需要多次用到数码管的动态扫描，所以我们考虑将其做成一个单独的子程序（见附件2），在需要时调用，一方面大面积的降低了程序篇幅，另一方面也使程序看上去更加简洁。

数码管动态扫描子程序如下：

//数码管全局显示函数

void display\_all(void)

{

uint i;

for(i=0;i<8;i++){ //对1-8位数码管交替扫描

display(duan[temp[i]],wei[i]); //送段选、位选数据

}

}

//数码管显示函数

void display(unsigned char duan, unsigned char wei)

{

P0=duan;

U7\_duan(); //开段选锁存器

P0=wei;

U8\_wei(); //开位选锁存器

delay(2); //决定刷新频率

}

数码管的动态显示，其原理就是依靠位选和段选锁存器，段选锁存器决定显示什么内容，位选锁存器决定哪几个数码管点亮；先将段选信号送入锁存器，这时，由于没有位选信号，所有数码管都是熄灭的；在送入位选信号后，相应的数码管才会点亮并且显示段选信号决定的数字。这时，我们只要让不同的数字，只需要不断地向不同的数码管输送段选信号，虽然数码管是同时显示相同的数字，但是利用人眼视觉暂留效应，让我们看起来不同的数码管显示了不同的数字。但是，需要注意的是每次送完段选信号，在送位选信号之前，先要关闭段选锁存器，防止在送入位选信号时，与之前的段选信号产生冲突，这种措施叫做“消影”。而程序中的U7\_duan();和U8\_wei();的作用正是将段选和位选锁存器打开一下，立刻自动关闭，防止接下来的数据对其进行干扰。

**5.2 Proteus软件仿真**

虽然程序在Keil软件设计完成，并且编译没有错误，但是很难保证事物就是完完全全的正确高效，所以还要利用软件仿真，确保将错误的可能性降到最小。仿真电路（见附件1）如图5-2所示：

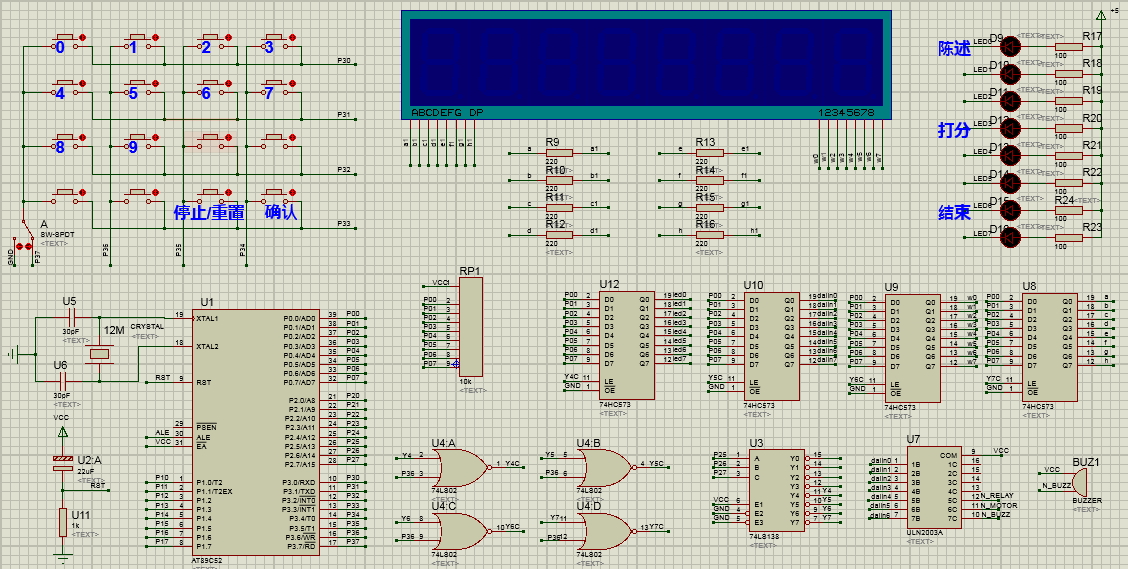


图5-2 Proteus仿真电路

在电路中，单片机的P0和P2口接了上拉电阻，P0内部没有集成上拉电阻，所以将其作为I/O口时，必须使用上拉电阻，而P2口虽然内置了上拉电阻，但是属于软上拉，对于电平要求较高的外部设备还是需要使用上拉电阻。这一点在电路连接时非常重要，因为一旦焊接完成，再改动就会影响电路的整体布局和美观。

将Keil程序生成的hex文件导入Proteus仿真软件，点击开始仿真，结果如图5-3所示：

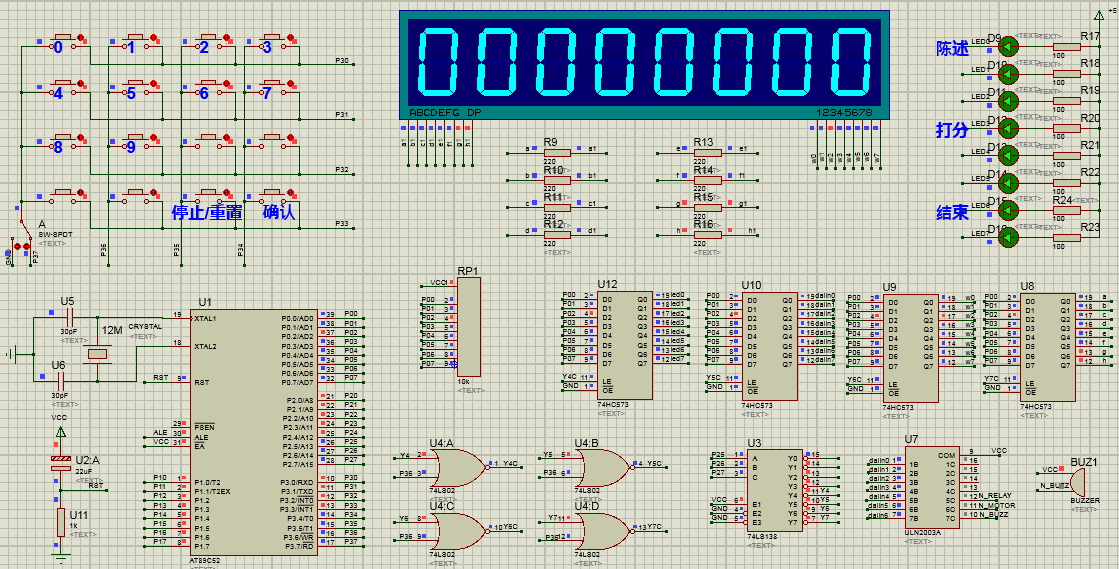


图5-3 Proteus仿真结果图

从仿真结果上看，运行非常完美，没有瑕疵！那么接下来就是实际电路的焊接过程了！

**第六章 系统硬件电路设计**

**6.1单片机最小系统**

**6.1.1 STC89C52芯片**

本次设计我们所采用的是STC89C52单片机，是一种带4k字节闪烁可编程可擦除只读存储器的低电压、高性能COMOS8的微处理器，该器件有40引脚，速度较快，价格便宜，烧录方便，通过串口即可下载，还可以实现在线编程，采用ATMEL高密度非易失存储器制造技术制造，与工业标准的MCS-51指令集和输出管脚相兼容。

**6.1.2 复位电路**

为确保微机系统中电路稳定可靠工作，复位电路是必不可少的一部分，复位电路的基本功能是：系统上电时提供复位信号，直至系统电源稳定后，撤销复位信号。为可靠起见，电源稳定后还要经一定的延迟才撤销复位，以防电源开关或电源插头分-合过程中引起的抖动而影响复位。当单片机的复位引脚出现2个机器周期以上的高电平时，单片机就执行复位操作。如果RST持续为高电平，单片机就处于循环复位状态。所以复位引脚的电容大一点没多大关系，顶多是复位时间长一点；但如果电容太小，高电平持续时间太短，则单片机无法正常复位，就不能工作，电容通常取10UF或22UF，铝电解电容即可。

单片机的复位电路在刚接通电时，刚开始电容是没有电的，电容内的电阻很低，通电后，5V的电源通过电阻给电解电容进行充电，电容两端的电会由0V慢慢的升到4V左右（此时间很短一般小于0.3秒），正因为这样，复位脚由低电位升到高电位，引起了内部电路的复位工作；当按下复位键时，电容两端放电，电容又回到0V了，于是又进行了一次复位工作。电路图如图4-1。



图6-1 复位电路

**6.1.3 晶振电路**

它是单片机系统正常工作的保证，如果振荡器不起振，系统将会不能工作。假如振荡器运行不规律，系统执行程序的时候就会出现时间上的误差，这在通信中会体现的很明显：电路将无法通信。它是由一个晶振和两个瓷片电容组成的，晶振和瓷片电容是没有正负的，两个瓷片电容相连的那端一定要接地，如图4-2所示。



图6-2 晶振电路

一般单片机的晶振工作于并联谐振状态，也可以理解为谐振电容的一部分。它是根据晶振厂家提供的晶振要求负载电容选值的，换句话说，晶振的频率就是在它提供的负载电容下测得的，能最大限度的保证频率值的误差，也能保证温漂等误差。

机器周期:通常从内存中读取一个指令字的最短时间来规定CPU周期,(也就是计算机通过内部或外部总线进行一次信息传输从而完成一个或几个微操作所需要的时间),它一般由12个时钟周期组成。而时钟周期=1秒/晶振频率，因此单片机的机器周期=12秒/晶振频率 ，补充其他几个周期：

指令周期（Instruction Cycle）：取出并执行一条指令的时间。

总线周期（BUS Cycle）：也就是一个访存储器或I/O端口操作所用的时间。

时钟周期（Clock Cycle）：又称节拍周期，是处理操作的最基本单位。(晶振频率的倒数，也称T状态)

指令周期、总线周期和时钟周期之间的关系：一个指令周期由若干个总线周期组成，而一个总线周期时间又包含有若干个时钟周期。

一般处理器的一个机器周期由12个时钟周期所组成。所以单片机用12M晶振，运行速度为1M。

两个电容的取值都是相同的，或者说相差不大，如果相差太大，容易造成谐振的不平衡，容易造成停振或者干脆不起振。其起到一个并联协振的作用，这样可以让它的脉冲更平稳与协调。

**6.2 数码管动态扫描电路**

数码管动态扫描电路采用LED数码管显示，通电后，4个数码管不断显示数字，一旦按下开始按键，4个数码管轮流显示随机数，显示时间会随着分数的增加而剪短，随后4个数码管全灭，等待用户输入记忆字符。

该电路由4个共阴数码管构成，除了公共端作为位选端口，其余端口按照a,b,c,d,e,f,g,dp分别连接，并连线至段选锁存器，4个公共端连线至位选锁存器（如图）。



图6-3 数码管动态扫描电路原理图

其中，两个锁存起的输入端均接到单片机P0口，由P3^0和P3^1两个引脚控制两个锁存器的工作状态。这样做大大节省了单片机的内部资源。SN74HC573AN锁存器的VCC和GND分别接电源和地，OE端接地，LE接相应的控制信号（段锁存器的LE接P3^0,位锁存器的LE接P3^1）

**6.3 键盘扫描电路**

在键盘中按键数量较多时，为了减少I/O口的占用，通常将按键排列成矩阵形式。在矩阵式键盘中，每条水平线和垂直线在交叉处不直接连通，而是通过一个按键加以连接。这样，一个端口（如P1口）就可以构成4\*4=16个按键，比之直接将端口线用于键盘多出了一倍，而且线数越多，区别越明显，比如再多加一条线就可以构成20键的键盘，而直接用端口线则只能多出一键（9键）。由此可见，在需要的键数比较多时，采用矩阵法来做键盘是合理的。

本次设计中采用行列式键盘，相对原来的简单排列键盘，更加节省资源，但是在控制上会有一定的变化，在软件的编程上尤为显著。行列式键盘连接电路如图4-4。



图6-4 键盘扫描电路连接图

**行列式键盘工作原理**

确定矩阵式键盘上何键被按下介绍一种“行扫描法”。

行扫描法 行扫描法又称为逐行（或列）扫描查询法，是一种最常用的按键识别方法，如上图所示键盘，介绍过程如下。

1、判断键盘中有无键按下 将全部行线Y0-Y3置低电平，然后检测列线的状态。只要有一列的电平为低，则表示键盘中有键被按下，而且闭合的键位于低电平线与4根行线相交叉的4个按键之中。若所有列线均为高电平，则键盘中无键按下。

2、判断闭合键所在的位置 在确认有键按下后，即可进入确定具体闭合键的过程。其方法是：依次将行线置为低电平，即在置某根行线为低电平时，其它线为高电平。在确定某根行线位置为低电平后，再逐行检测各列线的电平状态。若某列为低，则该列线与置为低电平的行线交叉处的按键就是闭合的按键。

单片机键盘有独立键盘和矩阵式键盘两种：独立键盘每一个I/O口上只接一个按键，按键的另一端接电源或接地（一般接地），这种接法程序比较简单且系统更加稳定；而矩阵式键盘式接法程序比较复杂，但是占用的I/O少。根据本设计的需要这里选用了独立式键盘接法。独立式键盘的实现方法是利用单片机I/O口读取口的电平高低来判断是否有键按下。将常开按键的一端接地，另一端接一个I/O口，程序开始时将此I/O口置于高电平，平时无键按下时I/O口保护高电平。当有键按下时，此I/O口与地短路迫使I/O口为低电平。按键释放后，单片机内部的上拉电阻使I/O口仍然保持高电平。我们所要做的就是在程序中查寻此I/O口的电平状态就可以了解我们是否有按键动作了。在用单片机对键盘处理的时候涉及到了一个重要的过程，那就是键盘的去抖动。这里说的抖动是机械的抖动，是当键盘在未按到按下的临界区产生的电平不稳定正常现象，并不是我们在按键时通过注意可以避免的。这种抖动一般10~200毫秒之间，这种不稳定电平的抖动时间对于人来说太快了，而对于时钟是微秒的单片机而言则是慢长的。硬件去抖动就是用部分电路对抖动部分加之处理，软件去抖动不是去掉抖动，而是避抖动部分的时间，等键盘稳定了再对其处理。所以这里选择了软件去抖动，实现法是先查寻按键当有低电平出现时立即延时10~200毫秒以避开抖动（经典值为20毫秒），延时结束后再读一次I/O口的值，这一次的值如果为1表示低电平的时间不到10~200毫秒，视为干扰信号。当读出的值是0时则表示有按键按下，调用相应的处理程序。

**设计心得**

本次课程设计，学到了好很多书本上没有的东西，感受深刻。

在软件设计上，最初是没有设计程序流程图的，全凭大脑思考，先建立一个基础程序，在此基础上进行修改升级，发现耗时耗力，最后实在无奈，将程序流程图设计好后，看着流程图进行设计，速度比之前快很多，思路也很明确。从中感受到：不管一个人是否很厉害，方法是最重要的。所谓工欲善其事，必先利其器，而这里的器就是画流程图。

在数码管的动态扫描中，最初设计完成时，在仿真软件中运行，数码管动态显示并不是很理想，会不停地闪烁，导致显示的字符不完整。查阅资料后发现在动态扫描函数中少了关键一步：“消影”，数码管的动态扫描是由段选和位选组合工作实现，送入段选信号后，接下来开位选，而上一步送入的段选信号还在P0口，一但开位选，该信号将首先送入位选锁存器，影响数码管的动态显示，故需要在送完段选信号后，通过程序将段选信号清除，即P0=0xff，那么下一步送入的才是真正的位选信号，这一步骤叫做消影。深刻感受到自己的渺小，在现在这个先进的时代，要充分利用身边的资源，不懂就问，吸取间接经验，省时高效。

**参考文献**

[1]STC89C52官方说明书

<http://www.alldatasheet.com/datasheet-pdf/pdf/139744/ETC1/STC89C52RC.html>

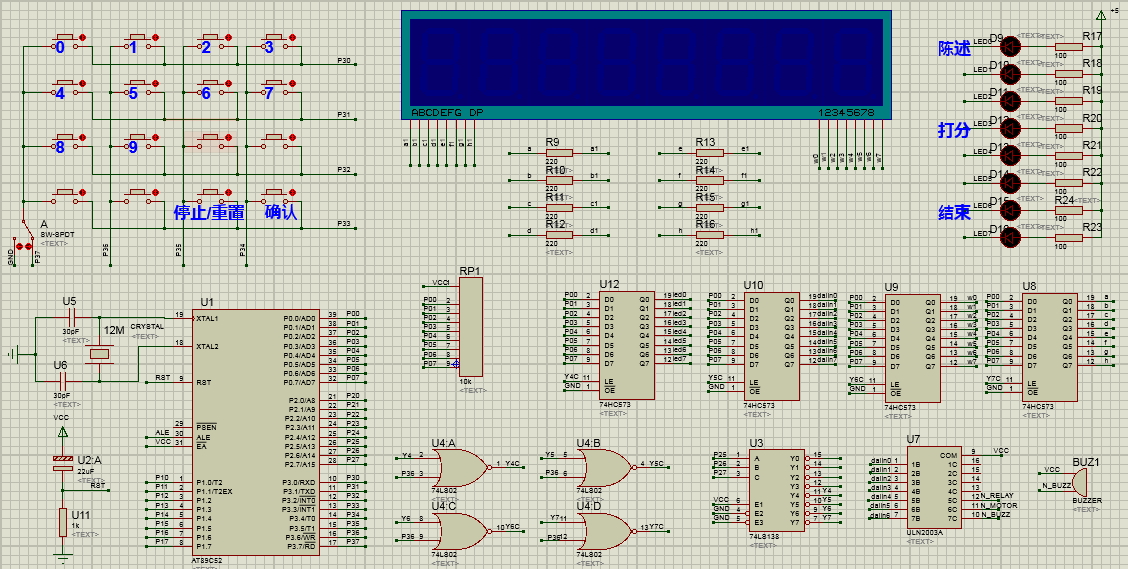
[2]SN74HC573AN芯片说明书

<http://www.alldatasheet.com/datasheet-pdf/pdf/27935/TI/SN74HC573AN.html>

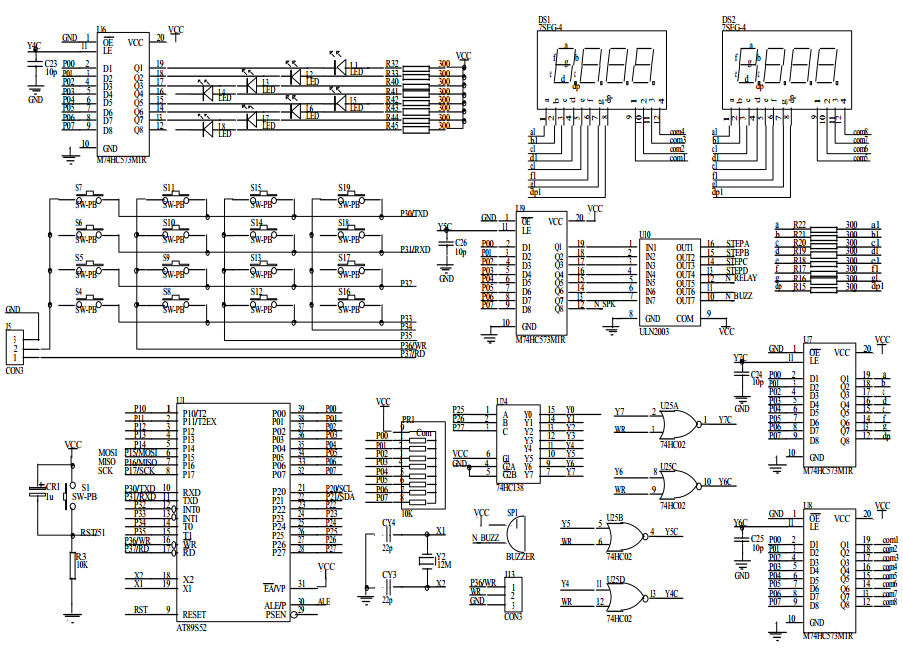
[3]郭天祥.新概念51单片机C语言教程—入门、提高、开发、拓展全攻略[M]

**附录**

**附录一 仿真电路**



**附录二 Protel电路原理图**



**附录三 C语言设计代码**

#include <reg52.h>

#include "tools.h"

#define uint unsigned int //宏定义

#define uchar unsigned char

//数码管字形表以及位选信号

// 0 1 2 3 4 5 6 7 8 9 10 11 12-c 13 14 15 - mie s

code uchar duan[19] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e,0xbf,0xff,0x92};

code uchar wei[9] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0xff}; //最后全亮

//数码管显示数据存储数组，显示的内容统一存放在该数组

uchar temp[8]={0,0,0,0,0,0,0,0}; //显示数据寄存在temp数组中

//全局变量定义

uchar secs = 0,mins = 1; //倒计时10分

uint value = 0; //按键值存储

uint flag = 0; //状态标示

uint average=0; //平均分

void beef(uint); //蜂鸣器函数声明

//定时器0中断初始化

void init()

{

TMOD=0x01; //定时器0和定时器1都是方式1,16位定时器

ET0=1; //使能定时器0

TR0=0; //定时器0关，等待输入学号按确认后开启

TH0=(65535-50000)/255; //定时器0高位装初值

TL0=(65535-50000)%255;; //定时器0低位装初值

EA=1; //开总中断

}

//定时器0终端服务函数

void time0()interrupt 1 //定时器0 服务

{

uchar count;

TH0=(65535-50000)/255; //定时器0高位装初值

TL0=(65535-50000)%255;; //定时器0低位装初值

count++;

if(count==20){ // 计时1s

count=0;

secs--;

}

if(secs>60)

{

secs = 59;

mins--;

}

if(secs==0&&mins==0){ // 时间到，蜂鸣器报警

display(0xc0,0xff); //数码管全部显示0

beef(3); //蜂鸣器响3声

flag=4; //状态标志为4，标志计时结束

secs=0,mins=1; //计时复位

TR0=0; //关闭定时器0计时

}

}

//蜂鸣器程序，直接调用，蜂鸣器响 a次

void beef(uint a)

{

while(a){

P0=P0|0x40; //P0口送蜂鸣器“发声”控制信号

U9\_beefcontrol(); //蜂鸣器控制开关，将控制信号送入锁存器，使蜂鸣器发声

delay(500); //蜂鸣器发声500ms

P0=P0&0xbf; //关蜂鸣器控制信号

U9\_beefcontrol(); //开关，执行蜂鸣器关闭

delay(500);

a--;

}

}

//数码管全局显示函数

void display\_all(void)

{

uint i;

for(i=0;i<8;i++){ //对1-8位数码管交替扫描

display(duan[temp[i]],wei[i]); //送段选、位选数据

}

}

//主函数

void main(void)

{

init(); //计时器中断初始化

P0=P0&0xbf; //关蜂鸣器

U9\_beefcontrol(); //蜂鸣器开关，送入控制信号

while(1){

value = read\_key16(); //读取按键值，没有按键按下，value=0xff

display\_all(); //数码管全局显示

//flag=0，第一步，监测按键是否按下

if(flag==0&&value!=0xff){ //有按键按下，即学号第一位数值

temp[0] = value; //读取学号第一位

flag++; //flag=1

value=0xff; //键值还原

}

//flag=1，第二步，监测学号第二位是否输入

if(flag==1&&value!=0xff){ //有按键按下，学号第二位输入

temp[1] = value; //读取学号第2位

flag++; //flag=2

value=0xff;

}

//flag=2，学号输入完毕，等待确认

if(flag==2&&value==15){ //学号输入完毕，按下确认按键

temp[2]=17; //三四两位数码管熄灭

temp[3]=17;

flag++; //flag=3

TR0=1; //开启定时器0中断，计时开始

value=0xff;

P0=0xfe; //led亮

U4\_LED(); //led开关

}

//计时过程中，数码管显示计时数据

if(TR0==1){

temp[4]=mins/10; //分钟计时十位

temp[5]=mins%10; //分钟计时个位

temp[6]=secs/10;

temp[7]=secs%10;

}

//监测计时是否完毕，或者选手提前陈述完毕（按下“结束--14”按键）

if((flag==3&&value==14)||(flag==4)){ //计时结束，或者按下“结束”

TR0=0; //停止计时

secs=0;mins=1; //计时还原

temp[4]=16; //显示 -

temp[5]=16;

temp[7]=16;

temp[2]=16;

temp[3]=0; //数码管显示0，等待打分

temp[6]=0; //数码管显示0，等待打分

flag=5; //flag=5，状态标志，标示进入打分环节

value=0xff;

P0=0xf7; //led亮

U4\_LED();

}

//第一个教师打分

if(flag==5&&value!=0xff){

temp[6]=value; //读取第一个分数，在第七个数码管显示

flag++; //flag=6，标示第一位老师打分完毕

value=0xff;

}

//第二个教师打分

if(flag==6&&value!=0xff){

temp[3]=value; //读取第二个分数，在第四个数码管显示

flag++; // flag=7，标示第二个老师打分完毕

value=0xff;

}

//教师打分完毕，等待确认

if(flag==7&&value==15){ //打分完毕，按下确认

average=(temp[3]+temp[6])/2+0.5; //两个成绩四舍五入

temp[2]=17; //熄灭

temp[3]=18; //s

temp[4]=12; //c

temp[5]=0; //o

temp[6]=16; //- 最终显示 “SCO-分数”的字形

temp[7]=average; //最后一位数码管显示最终成绩

value=0xff; //键值还原

flag++; //flag=8，标示选手答辩完毕

P0=0xbf; //led亮

U4\_LED();

}

//选手打分完毕，是否进入下一位答辩？（是否按下“重置”）

if(flag==8&&value==14){ //按下按键，进入下一轮答辩

temp[0]=0;temp[1]=0; //数码管显示数组内容清零

temp[2]=0;temp[3]=0;

temp[4]=0;temp[5]=0;

temp[6]=0;temp[7]=0;

flag=0; //标志位清零

P0=0xff; //led灭

U4\_LED();

}

}

}

#ifndef \_\_tools\_h\_\_

#define \_\_tools\_h\_\_

#include "tools.h"

//#include <reg52.h>

#include <intrins.h>

//延时函数

void delay(unsigned int x)

{

unsigned int i,j;

for(i=x;i>0;i--){

for(j=120;j>0;j--);

}

}

//读取矩阵键盘键值 返回的就是按键的对应值 0-15

bit key\_re;

unsigned char key\_press;

unsigned char key\_value;

unsigned char read\_key16(void)

{

unsigned char key\_temp;

static unsigned char col;

P3 = 0xf0;

key\_temp = (P3&0xf0);

if(key\_temp != 0xf0) //有按键按下

{key\_press++;

//delay(10);

}

else

key\_press = 0;//抖动

if(key\_press == 3) //连续三次都是有按键按下 则确定为有按键按下

{

key\_press = 0;

key\_re = 1;

switch(key\_temp)

{

case 0x70:

col = 1; //第一列按键按下

break;

case 0xb0:

col = 2; //第二列按键按下

break;

case 0xd0:

col = 3; //第三列按键按下

break;

case 0xe0:

col = 4; //第四列按键按下

break;

}

P3 = 0x0f;

key\_temp = (P3&0x0f);

switch(key\_temp)

{

case 0x0e:

key\_value = (col-1); //第一行按键按下

break;

case 0x0d:

key\_value = (3+col); //第二行按键按下

break;

case 0x0b:

key\_value = (7+col); //第三行按键按下

break;

case 0x07:

key\_value = (11+col); //第四行按键按下

break;

}

}

//连续三次检测到按键被按下，并且该按键已经释放

P3 = 0x0f;

key\_temp = (P3&0x0f);

if((key\_re == 1) && (key\_temp == 0x0f))

{

key\_re = 0;

return key\_value;

}

return 0xff; //无按键按下或被按下的按键未被释放

}

//U9，蜂鸣器等驱动控制Y5\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void U9\_beefcontrol(void)

{

P2=P2&0x1f;

P2=P2|0xa0;

\_nop\_();

\_nop\_();

P2=P2&0x1f;

}

//段锁存器和位锁存器LE控制\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//sbit HC138A = P2^5;

//sbit HC138B = P2^6;

//sbit HC138C = P2^7;

//数码管段选开关

void U7\_duan(void)

{

/\*HC138C=1;

HC138B=1;

HC138A=1;\*/

P2=P2&0x1f;

P2=P2|0xe0; //只影响高3位用于38译码

\_nop\_();

\_nop\_();

/\*HC138C=0;

HC138B=0;

HC138A=0;\*/

P2=P2&0x1f;

}

//数码管位选开关

void U8\_wei(void)

{

/\*HC138C=1;

HC138B=1;

HC138A=0;\*/

P2=P2&0x1f;

P2=P2|0xc0;

\_nop\_();

\_nop\_();

/\*HC138C=0;

HC138B=0;

HC138A=0;\*/

P2=P2&0x1f;

}

//数码管显示函数

void display(unsigned char duan, unsigned char wei)

{

P0=duan;

U7\_duan();

P0=wei;

U8\_wei();

delay(2); //决定刷新频率

}

//LED的U4锁存器LE控制\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8888

void U4\_LED(void)

{

P2=P2|0x80;

\_nop\_();

P2=P2&0x1f;

}

#endif

**附录四 实物图**

