

**《电子工艺实习与AI芯片设计》**

**实验报告**

**基于STC89C52的电子倒计时器设计**

姓 名： 杨学通

学 号： 08213129

任课教师： 王冠军

中国矿业大学

2023年04月

目录

[一 硬件焊接部分工作与体会 1](#_Toc131077623)

[1硬件焊接工作成果 1](#_Toc131077624)

[2工作体会 1](#_Toc131077625)

[3错误检测与改正 1](#_Toc131077626)

[板子不能烧写程序，提示“请给MCU上电” 1](#_Toc131077627)

[蜂鸣器不响 1](#_Toc131077628)

[时钟测试程序，数码管只显示不变化 1](#_Toc131077629)

[二 设计选题及设计内容 1](#_Toc131077630)

[1设计选题 1](#_Toc131077631)

[2设计内容 1](#_Toc131077632)

[数码管显示模块： 1](#_Toc131077633)

[4×4按键阵列： 1](#_Toc131077634)

[中断（interrupt）部分： 1](#_Toc131077635)

[三 设计开发环境与关键代码 1](#_Toc131077636)

[1 开发环境 1](#_Toc131077637)

[2关键代码 1](#_Toc131077638)

[变量声明部分 1](#_Toc131077639)

[延迟函数 1](#_Toc131077640)

[数码管显示函数 1](#_Toc131077641)

[按键扫描函数 1](#_Toc131077642)

[设置时间计算函数 1](#_Toc131077643)

[初始化函数 1](#_Toc131077644)

[定时器计数函数 1](#_Toc131077645)

[中断0：进入调整时间模式 1](#_Toc131077646)

[中断2：暂停倒计时 1](#_Toc131077647)

[主函数 1](#_Toc131077648)

[四 设计流程与步骤 1](#_Toc131077649)

[1 分频计时模块 1](#_Toc131077650)

[2 数码管显示模块 1](#_Toc131077651)

[3 中断按键阵列 1](#_Toc131077652)

[4 4×4按键阵列 1](#_Toc131077653)

[五 设计结果与分析 1](#_Toc131077654)

[1成果展示 1](#_Toc131077655)

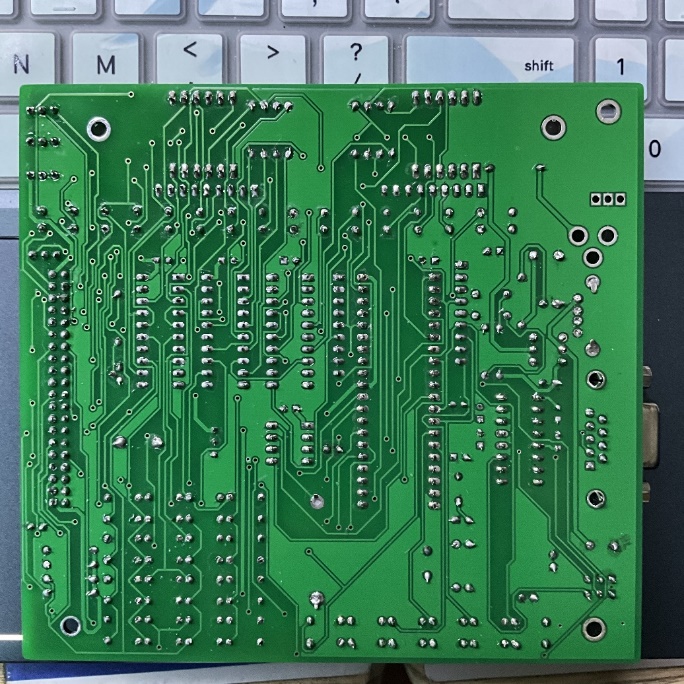
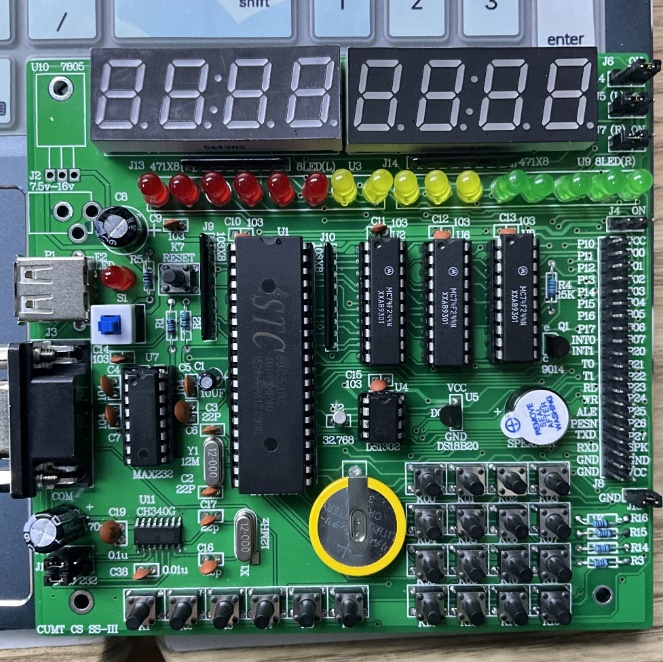
[2 结果分析 1](#_Toc131077656)

[数码管显示原理 1](#_Toc131077657)

[4×4按键原理 1](#_Toc131077658)

1. **硬件焊接部分工作与体会**

## 1硬件焊接工作成果



## 2工作体会

焊接工作的好坏直接影响到软件部分的运行。通过这次焊接工作，我掌握了几项焊接技能：

1. CH340G芯片的引脚密集细小且呈排状分布，可以将大量焊锡充分加热熔融，利用焊锡的重力，让焊锡在引脚间流动，挂粘到芯片的引脚上。而且这种方法的好处是，焊锡分子相互吸引，避免了焊锡在引脚间的滞留，造成短路；
2. 要使每个焊脚饱满光亮，首先要让焊锡充分加热熔融，并使足够多的焊锡流入接口的孔洞中。

焊接部分的实践，我掌握了焊锡焊接电路板的方法。第一次尝试焊接电路板难免会遇到许多问题，比如焊锡堵住焊接口的问题，在老师的帮助下，我很好的解决了这些问题。

## 3错误检测与改正

本次实验，我的板子焊接很牢固，在更换了一系列不配套的芯片后，能很好的运行测试软件。

在学有余力的同时，我帮助了很多同学，对遇到的问题总结如下：

### 板子不能烧写程序，提示“请给MCU上电”

可能原因：

STC89C52芯片不配套，更换后大部分板子可以上电；

CH340G芯片焊接粗糙，出现短路，或者断路，在重新焊接后，板子可正常上电；

J16、J17跳线接错位置（仅遇到两例）。

### 蜂鸣器不响

可能原因：

DS18B20三极管与S9014三极管焊反位置，三极管发烫现象严重。

### 时钟测试程序，数码管只显示不变化

可能原因：

DS1302芯片不适用，更换后时钟跳转，并且更换不同的1302芯片时钟跳转速度不同。

# 二 设计选题及设计内容

## 1设计选题

可以设置时间、可以随时暂停与继续的电子倒计时器。

## 2设计内容

### 数码管显示模块：

显示格式：分-秒-毫秒，每级显示两位，中间由“-”间隔开，如“08-50-22”表示倒计时8分55秒220毫秒；

### 4×4按键模块：

由3×3阵列控制输入1-9，此外，还有两个按键控制输入0，控制删除功能。

### 中断（interrupt）模块：

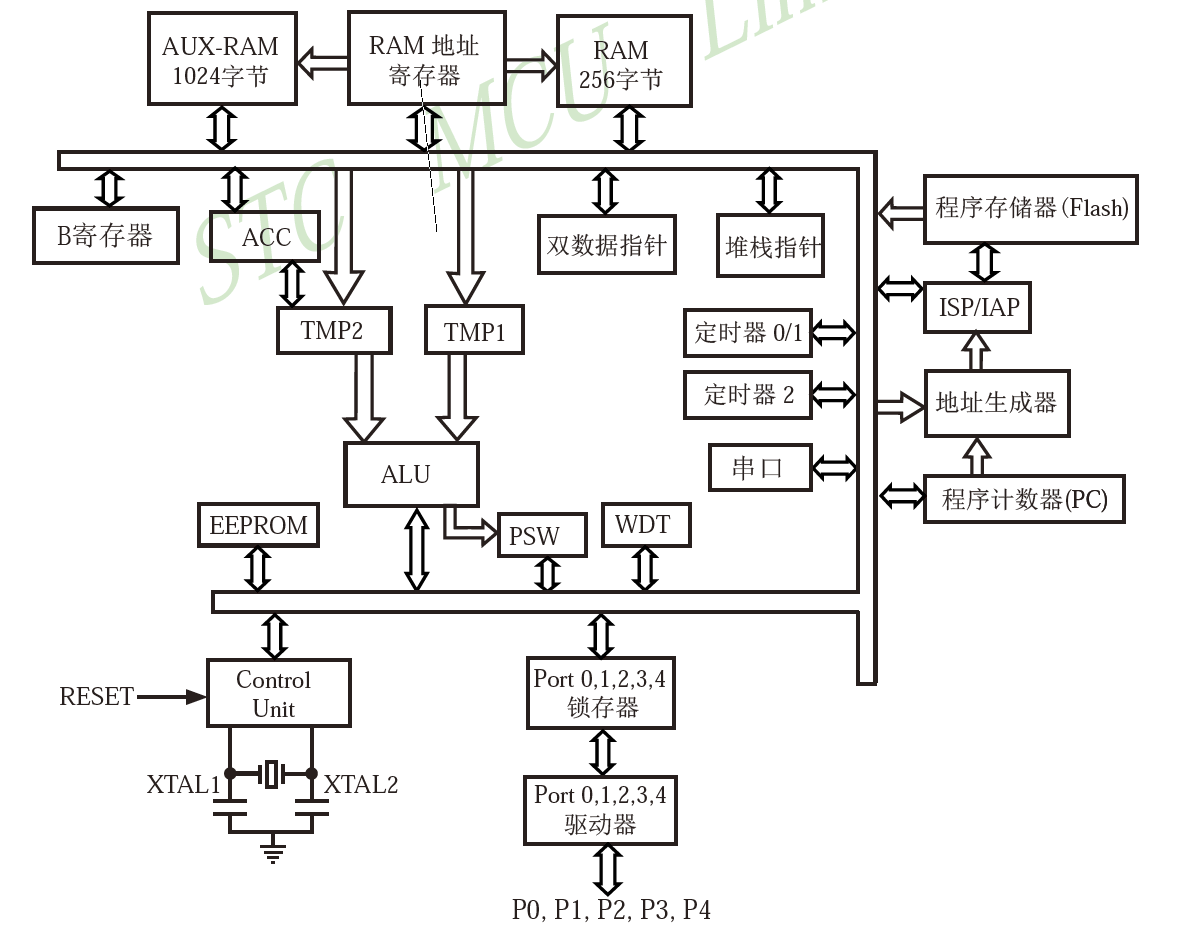
1. interrupt 0：控制当前为显示倒计时模式还是设置时间模式。利用interrupt 0还可以实现暂停倒计时功能；
2. interrupt 2：继续倒计时。

# 三 设计开发环境与关键代码

## 1 开发环境

### 硬件开发环境：

STC89C51RC系列单片机是宏晶科技推出的新一代超强抗干扰/高速/低功耗的单片机，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/机器周期可任意选择。STC89C52单片机中包含中央处理器(CPU)、程序存储器(Flash)、数据存储器(SRAM)、定时/计数器、UART串口、I/O接口、EE-PROM、看门狗等模块。STC89C52系列单片机几乎包含了数据采集和控制中所需的所有单元模块，可称得上一个片上系统。STC89C52系列单片机的内部结构框图如下图所示。

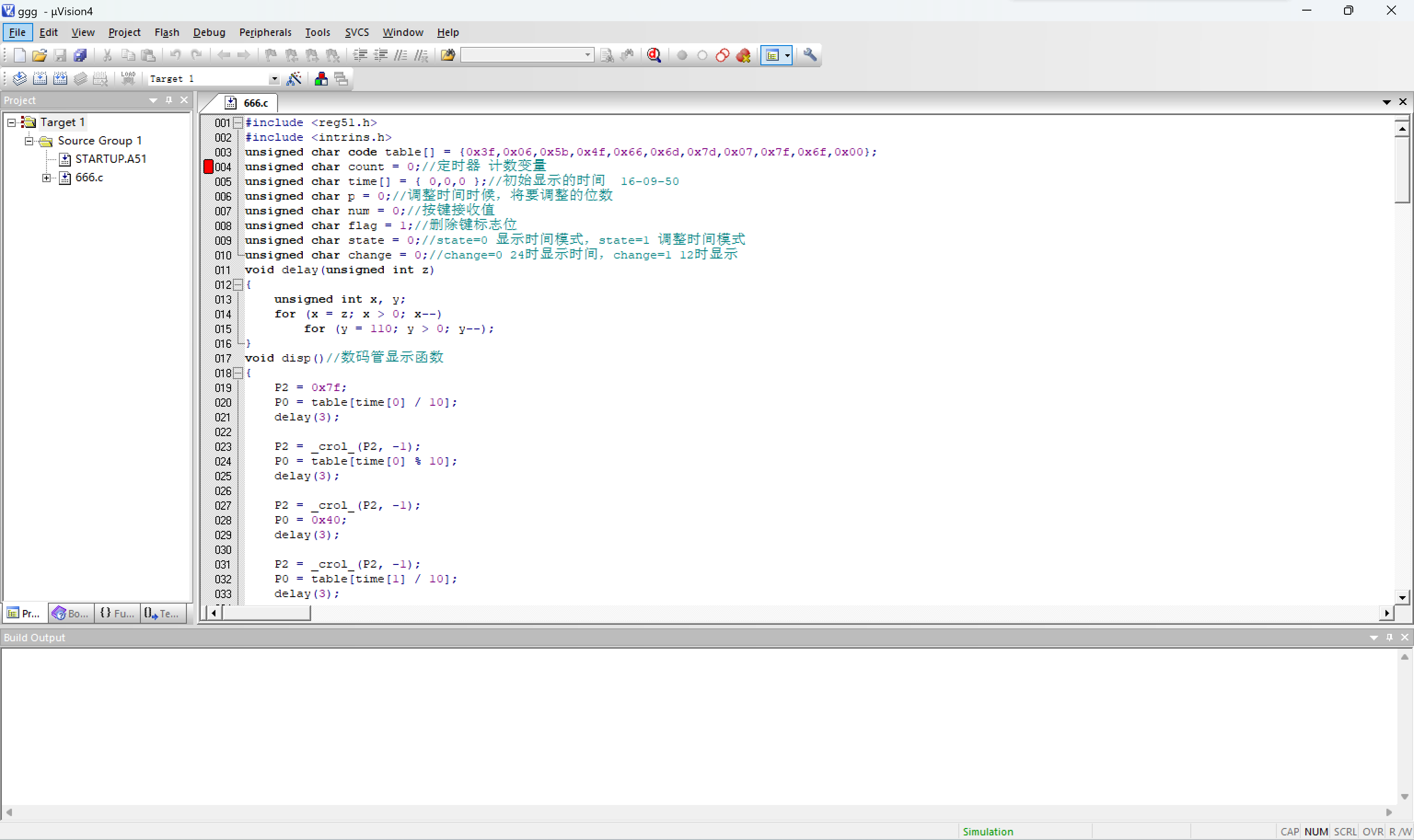


### 软件开发环境：

Keil uVision4

Keil C51是美国Keil Software公司出品的51系列兼容单片机C语言软件开发系统，与汇编相比，C语言在功能上、结构性、可读性、可维护性上有明显的优势，因而易学易用。Keil提供了包括C编译器、宏汇编、链接器、库管理和一个功能强大的仿真调试器等在内的完整开发方案，通过一个集成开发环境（μVision）将这些部分组合在一起。

Keil uVision4界面如下：



## 2关键代码

### 变量声明部分

unsigned char code table[] = { 0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x00 };

unsigned char count = 0;//定时器 计数变量

unsigned char time[] = { 0,0,0 };//初始显示的时间00-00-00

unsigned char p = 0;//调整时间变量，记录将要调整的位置

unsigned char num = 0;//按键接收值

unsigned char flag = 1;//删除键标志位

unsigned char state = 0;//state=0倒计时，state=1 调整时间

unsigned char look = 0;// look=0 倒计时，look=1 暂停倒计时

### 延迟函数

void delay(unsigned int z)

{

unsigned int x, y;

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

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

}

### 数码管显示函数

void disp()

{

P2 = 0x7f;

P0 = table[time[0] / 10];

delay(3);

P2 = \_crol\_(P2, -1);

P0 = table[time[0] % 10];

delay(3);

P2 = \_crol\_(P2, -1);

P0 = 0x40;

delay(3);

P2 = \_crol\_(P2, -1);

P0 = table[time[1] / 10];

delay(3);

P2 = \_crol\_(P2, -1);

P0 = table[time[1] % 10];

delay(3);

P2 = \_crol\_(P2, -1);

P0 = 0x40;

delay(3);

P2 = \_crol\_(P2, -1);

P0 = table[time[2] / 10];

delay(3);

P2 = \_crol\_(P2, -1);

P0 = table[time[2] % 10];

delay(3);

P0 = 0xff;

}

### 按键扫描函数

void keyscan()

{

P2 = 0xfe;

while ((P2 & 0xf0) != 0xf0){

delay(3);

while ((P2 & 0xf0) != 0xf0){

switch (P2){

case 0xee: num = 1; break;

case 0xde: num = 2; break;

case 0xbe: num = 3; break;

}

if (state && flag)p++;

else flag=1;

delay(300);

while ((P2 & 0xf0) != 0xf0) disp();

}

}

P2 = 0xfd;

while ((P2 & 0xf0) != 0xf0){

delay(3);

while ((P2 & 0xf0) != 0xf0){

switch (P2){

case 0xed: num = 4; break;

case 0xdd: num = 5; break;

case 0xbd: num = 6; break;

}

if (state && flag)p++;

else flag=1;

delay(300);

while ((P2 & 0xf0) != 0xf0) disp();

}

}

P2 = 0xfb;

while ((P2 & 0xf0) != 0xf0){

delay(3);

while ((P2 & 0xf0) != 0xf0){

switch (P2){

case 0xeb: num = 7; break;

case 0xdb: num = 8; break;

case 0xbb: num = 9; break;

}

if (state && flag) p++;

else flag=1;

delay(300);

while ((P2 & 0xf0) != 0xf0) disp();

}

}

P2 = 0xf7;

while ((P2 & 0xf0) != 0xf0){

delay(3);

while ((P2 & 0xf0) != 0xf0){

switch (P2){

case 0xe7: break;

case 0xd7: num = 0; break;

case 0xb7: num = 0; flag=0; break;

}

if (state && flag) p++;

else flag=1;

delay(300);

while ((P2 & 0xf0) != 0xf0) disp();

}

}

}

### 设置时间计算函数

void change\_time(){

if (state&&flag){

switch (p){

case 1:time[0] = (num \* 10 + time[0] % 10); break;

case 2:time[0] = (num + (time[0] / 10 \* 10)); break;

case 3:time[1] = (num \* 10 + time[1] % 10); break;

case 4:time[1] = (num + (time[1] / 10 \* 10)); break;

case 5:time[2] = (num \* 10 + time[2] % 10); break;

case 6:time[2] = (num + (time[2] / 10 \* 10)); break;

}

if (p >= 6) {

//全部调整完时间，进入正常时间显示模式，并打开定时器

state = 0;

TR1 = 1;

p = 0;

}

}

}

### 初始化函数

void init\_timer1()

{

TMOD = 0x15;

IE = 0x85;

IP = 0x04;

TL1 = (65535 - 10000) % 256;//10ms

TH1 = (65535 - 10000) / 256;

TR1 = 1;

ET1 = 1;

EA = 1;

}

### 定时器计数函数

void TIME\_1() interrupt 3

{

TL1 = (65535 - 10000) %256; //10ms

TH1 = (65535 - 10000) /256;

count++;

if (count == 1){ //每10ms毫秒级自减1

count = 0;

if (time[2]<=0){

if (time[1]>0) {

time[1]--;

time[2]=99;

}

else if(time[1]==0&&time[0]>0&&time[2]==0){

time[1]=59;

time[2]=99;

time[0]--;

}

else if(time[1]==0&&time[0]==0&&time[2]==0){

time[1]=0;

time[2]=0;

time[0]=0;

}

}

else time[2]--;

}

}

### 中断0：进入调整时间模式

void int0() interrupt 0 {

state = 1;

TR1 = 0;

}

### 中断2：暂停倒计时

void int1() interrupt 2{

state = 0;

TR1=1;

}

### 主函数

void main(){

init\_timer1();//定时器中断初始化

while (1){

change\_time();

disp();

keyscan();

}

}

# 四 设计流程与步骤

## 1 分频计时模块

12MHz的晶振经过12分频可以获得1MHz的系统时钟，也即时钟周期为1μs。对于计时计毫秒位，则需要每10ms自减1。故需要计数器计数（10×1000μs）/1（μs）=10000次，毫秒位置自减1。对于16位的计数器来说，其最大计数为216-1=65535，也就是说如果计数器从0开始计数，需要计数到65535才会产生溢出，而我们只需要计数器计数10000次就产生溢出，因此开始计数的位置不能从0开始，而是从65535-10000开始。

10ms计时器代码如下：

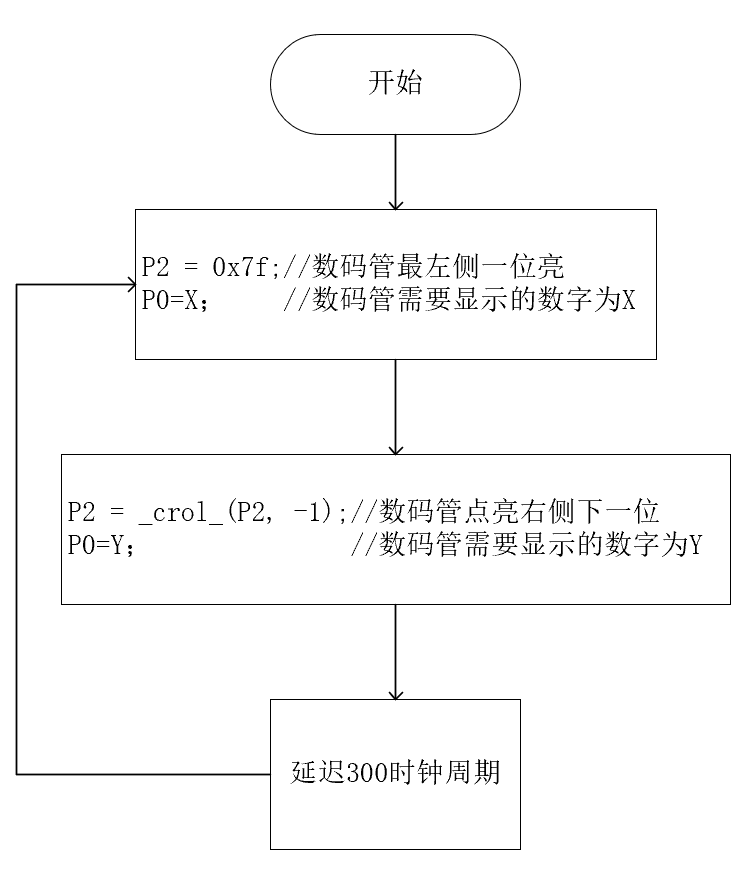
TL1 = (65535 - 10000) %256; //10ms

TH1 = (65535 - 10000) /256;

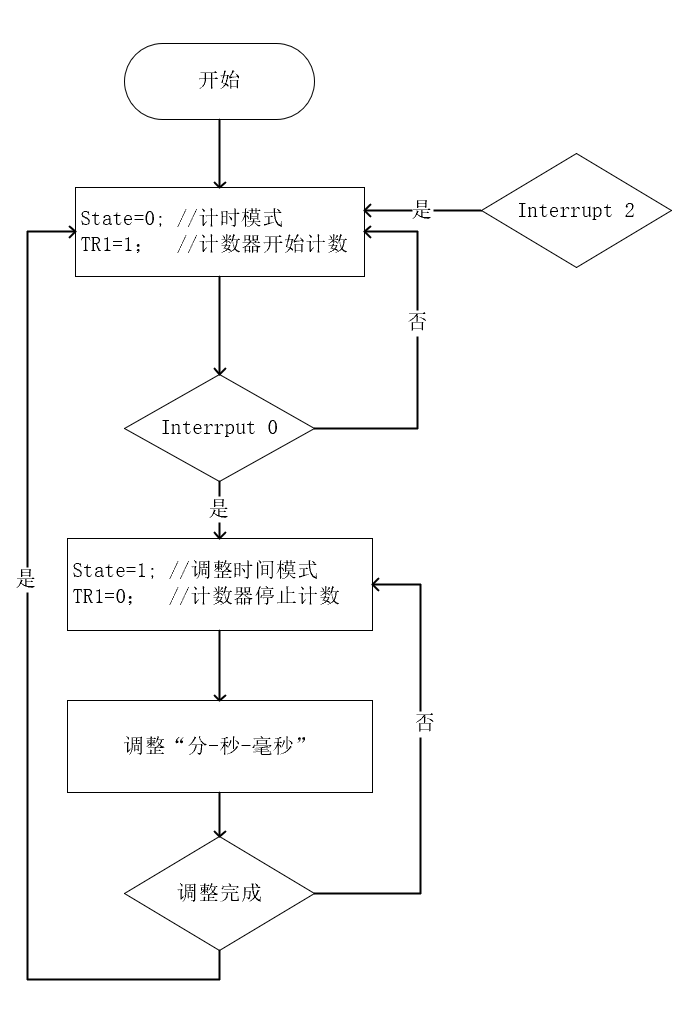
count++；

理论上来说（65535-X）其中X为多少，就是计时多少。

## 2 数码管显示模块



## 3 中断按键阵列



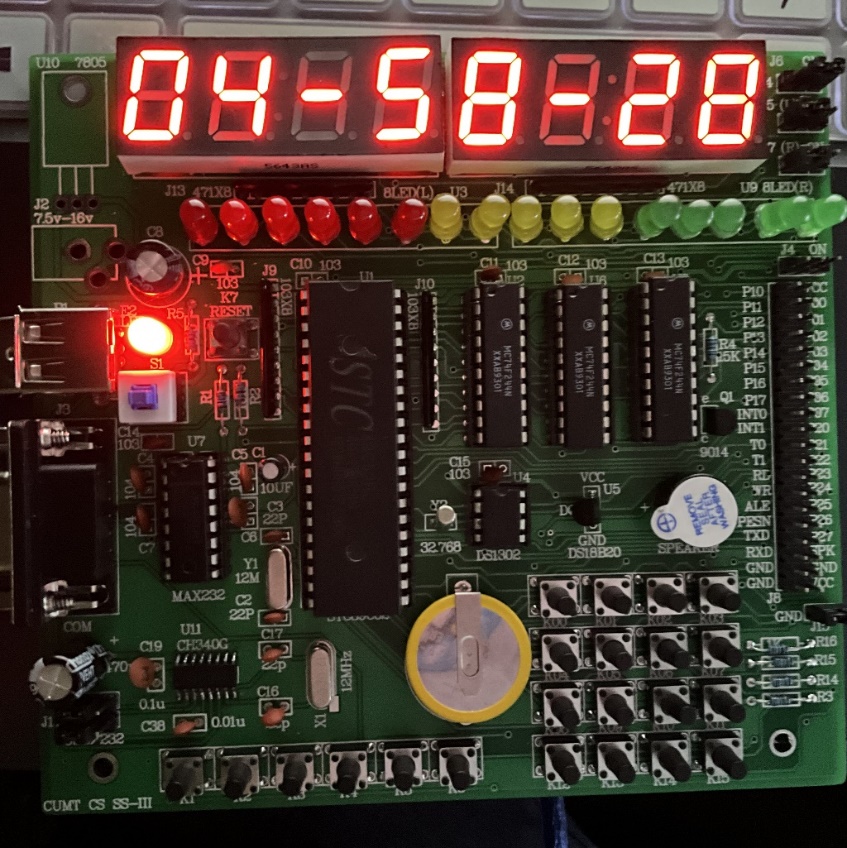
## 4 4×4按键阵列

按键对应关系如下：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| K00 | K01 | K02 | K03 |  | 1 | 2 | 3 | Φ |
| K10 | K11 | K12 | K13 | 4 | 5 | 6 | 0 |
| K20 | K21 | K22 | K23 | 7 | 8 | 9 | Φ |
| K30 | K31 | K32 | K33 | Φ | Del | Φ | Φ |

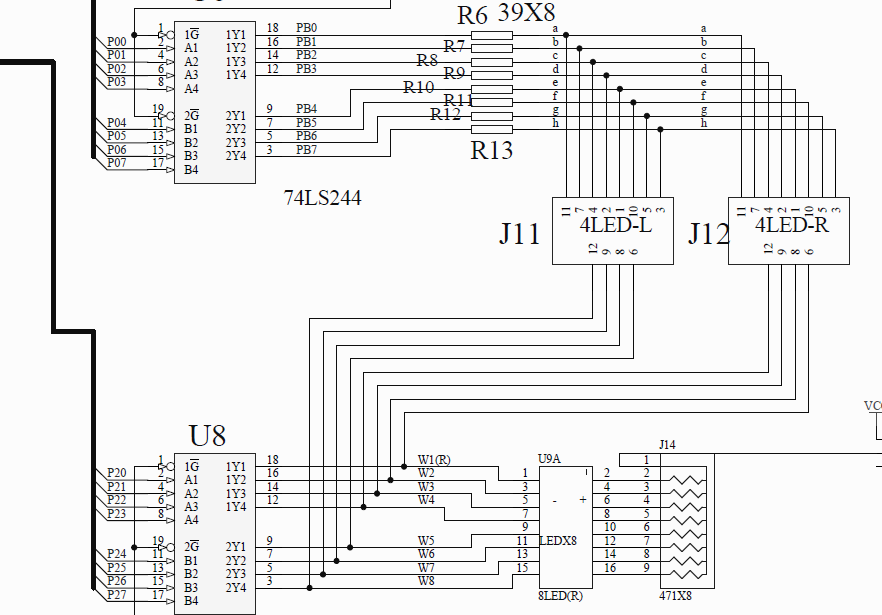
# 五 设计结果与分析

## 1成果展示



## 2 结果分析

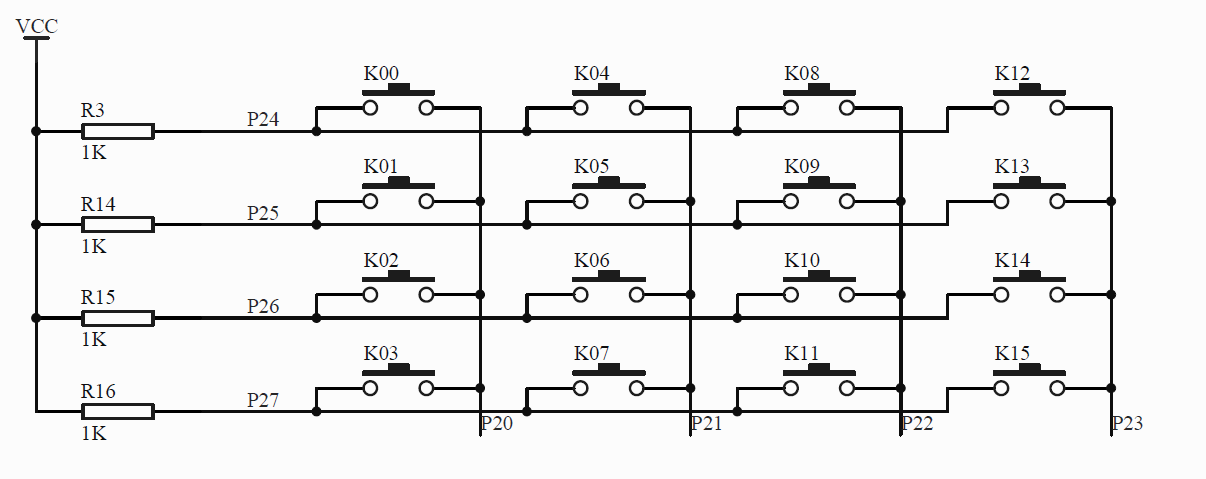
### 数码管显示原理



根据板子原理图以及数码显示部分的代码可知

P0全部为高电位，从第一个数码开始，P2给予低电位，此时点亮第一位数码管。其后，每延迟330个时钟周期，P2左移一位，给予下一位低电位，点亮下一位数码管，并以此循环下去。由于视觉残留效应，快速闪动的数码管，可以完整的显示全部数位。

### 4×4按键原理



根据板子原理图，P24，P25，P26，P27，P20，P21，P22，P23八根线接VCC此时为高电位，即0xff，某一时刻，将P20置为低电位，P21，P22，P23为高电位，即0xfe。假设此时K00按键被按下，则P24线相当于接地，为低电位，即0xee。如此将P20，P21，P22，P23依次改为低电位，检测P24，P25，P26，P27四根线，即可判断哪一个按键被按下。

中国矿业大学教师评阅书

|  |
| --- |
| 成 绩：  评阅教师签字： |