计算机科学与技术学院

《单片机控制实验》  
期末题目设计报告

班级 28班、29班

团队成员姓名及学号

21201206 江琳艺

21200903 张乐佳

21200202 刘怡凝

21200905 孟珺逸

2022年 12 月 30 日

1. **设计题目：**

1、使用实验设备中的七段数码管，每隔1秒依次显示100以内的所有素数。

2、联合使用直流电机和LED点阵显示屏，调节速度并显示速度曲线。

1. **题目的设计内容和要求**：

1、使用实验设备中的七段数码管，每隔1秒依次显示100以内的所有素数。

要求：

（1）素数使用程序计算完成，不得在程序内存储列表；

（2）使用后两个数码管，将素数以16进制形式显示出来，第一个数码管保持全暗；

（3）全部显示完成后从头开始循环；

（4）必须使用汇编语言完成。

2、联合使用实验6的直流电机和实验4的LED点阵显示屏，完成以下内容：

（1）直流电机以脉宽调速方式工作，按下S1键增加速度，按下S2键减慢速度；

（2）将LED点阵显示屏看作一个直角坐标系，将测量出来的速度以曲线的方式显示出来。在每秒，将测量出来的转速换算成坐标系Y轴的相应坐标y，在(0,y)的位置点亮一个点，曲线上已有的其他值向右平移一位。

（3）Y轴坐标0对应速度50，速度每增加8，坐标加1，超过上界按最大值显示。

1. **分工及成员贡献度**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 学号 | 姓名 | 任务分工 | 贡献权重 |
| 1 | 21201206 | 江琳艺 | 第二题的整体设计、程序实现与调试 | 100% |
| 2 | 21200903 | 张乐佳 | 第一题的整体设计、程序实现与调试 | 60% |
| 3 | 21200202 | 刘怡凝 | 第一题的函数设计、程序实现与调试 | 60% |
| 4 | 21200905 | 孟珺逸 | 报告撰写 | 50% |

1. **题目设计开发和调试：**
2. **题目总体设计：**描述系统主要功能作用，根据问题要求设计方案，说明涉及到的原理，并分析设计的局限性。

**第一题：**

1. 系统主要功能

在数码管上，每隔1秒依次显示100以内的所有素数。

1. 设计方案
2. 数码管显示

首先初始化端口，用单片机P1.5作为模拟串口数据，使用P1.4模拟串口时钟，初始时显示0X02，然后计算出需要显示的素数，输出时，需要从高位到低位依次向移位寄存器输出8个比特。移位寄存器的数据线和时钟线分别接到单片机的P1.5和P1.4管脚，时钟(CLK)每次由低变高时使用位操作指令进行输出，按位调用显示函数显示数据，显示完毕后进行延时，判断是否到100，到100后重新从头开始循环。

1. 计算素数

首先判断数值是否超过100，超过100则重新从头开始循环，然后判断当前数值能否被除了1与它本身的其他数整除（除数从2开始，每次加1，直到与被除数相等，判断余数是否为0），若能被整除，则不是素数，否则为素数。

1. 涉及到的原理

本实验中，三个数码管以串行方式连接，要想输出一个字形码，就需要从高位到低位依次向移位寄存器输出8个比特。移位寄存器的数据线和时钟线分别接到单片机的P1.5和P1.4管脚，可以使用MCS-51里的位操作指令进行输出。连续输出3个字形，24个bit之后，欲显示的字形将稳定地显示在数码管上，程序可以转而执行其他工作。

七段字形的编码方式需要通过实验获得。这些编码作为程序中的常数，使用DB命令存放。在程序中，需要将数值转换为相应的字形编码，可以使用MOVC指令来完成。采用3个74HC164级联控制三个数码管的显示。其中使用单片机P1.5作为模拟串口数据，使用P1.4模拟串口时钟，CLR端接高电平。使用上一个74HC164的Q7作为下一个74HC164的输入端。

74LS164 是高速 CMOS 器件。74LS164 是8位边沿触发式移位寄存器，串行输入数据，然后并行输出。数据通过两个输入端（A或B）之一串行输入，任一输入端可以用作高电平使能端，控制另一输入端的数据输入。两个输入端或者连接在一起，或者把不用的输入端接高电平，一定不能悬空。时钟（CLK）每次由低变高时，数据右移一位，输入数据移位到 Q0，Q0 是两个数据输入端（A和 B）的逻辑与，它将上升时钟沿之前保持一个建立时间的长度。主复位(CLR)输入端上的一个低电平将使其它所有输入端都无效,同时非同步地清除寄存器，强制所有的输出为低电平。

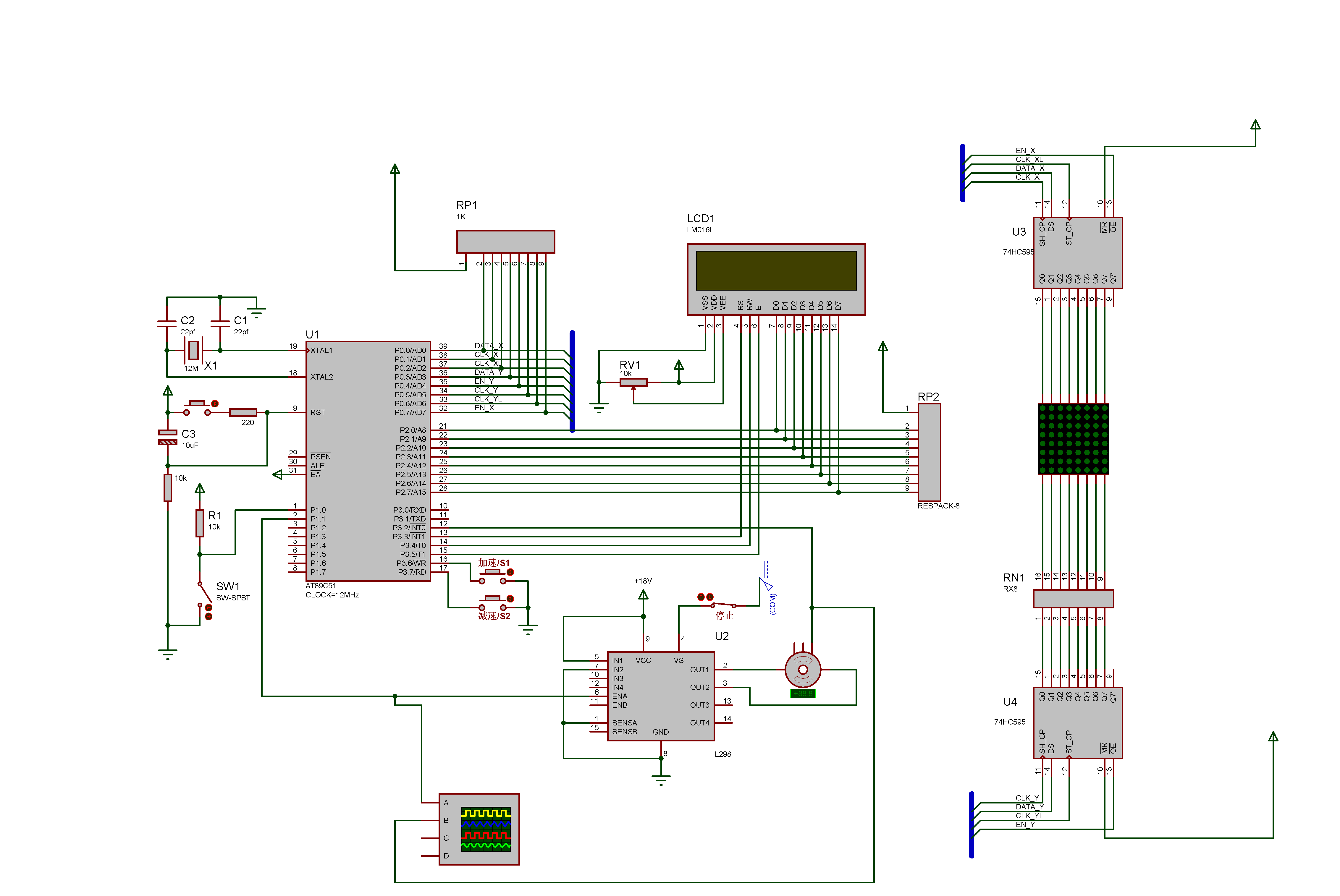
**第二题：**

1. 系统主要功能

以脉宽调速的方式调节直流电机的速度并在LED点阵显示屏上显示速度曲线。

1. 设计方案

接口电路设计：



1. 脉宽调速

通过记录每秒产生的脉冲次数计算当前转速，按下开关后，使用累加进位法通过改变占比调整转速。

1. LED点阵显示

将LED点阵显示屏看作一个直角坐标系，在每秒，将测量出的转速换算成坐标系Y轴的相应坐标y（Y轴坐标0对应速度50，速度每增加8，坐标加1，超过上界按最大值显示），在（0，y)的位置点亮一个点，曲线上已有的其他值依次向右平移一位，形成新的速度曲线。

1. 涉及到的原理：
2. LED点阵显示

高亮度 LED 发光管构成点阵，通过编程控制可以显示中英文字符、图形及视频动态图形，也可以根据坐标直接计算点阵中每个像素的值然后输出，绘制各种图形。为了能够显示出一个点阵字型，需要进行循环扫描，也就是每一次只点亮一行，然后在列上输出该列对应的 8 个点阵值。 输出一行后暂停一段时间，输出下一行。行和列可以互换。

1. 直流电机脉宽调制

对于直流电机来说，其转速由输入电压决定，脉宽调制是一种能够通过开关量输出达到模拟量输出效果的方法。基本原理是通过输出一个很高频率的0/1信号，其中1的比例为δ（也叫做占空比），在外围积分元件的作用下，使得总的效果相当于输出δ×A （A为高电平电压）的电压。

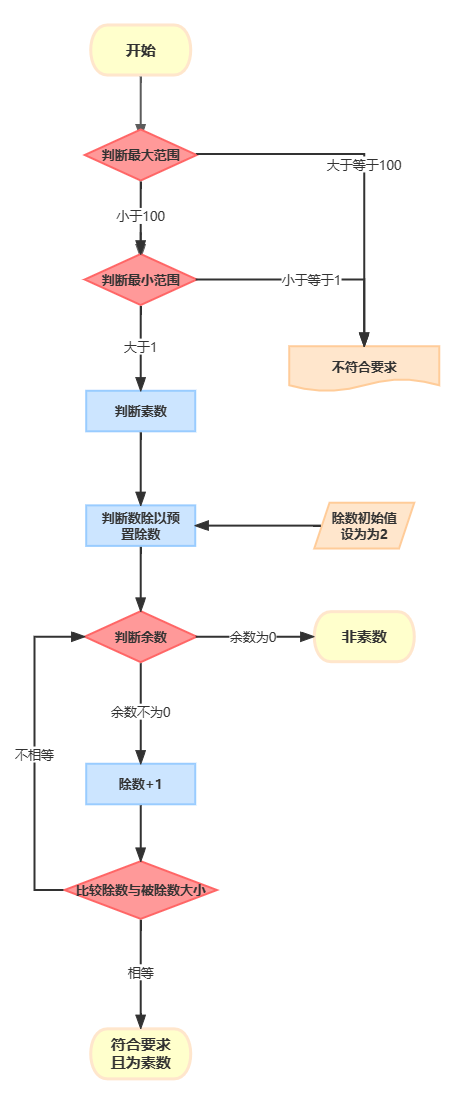
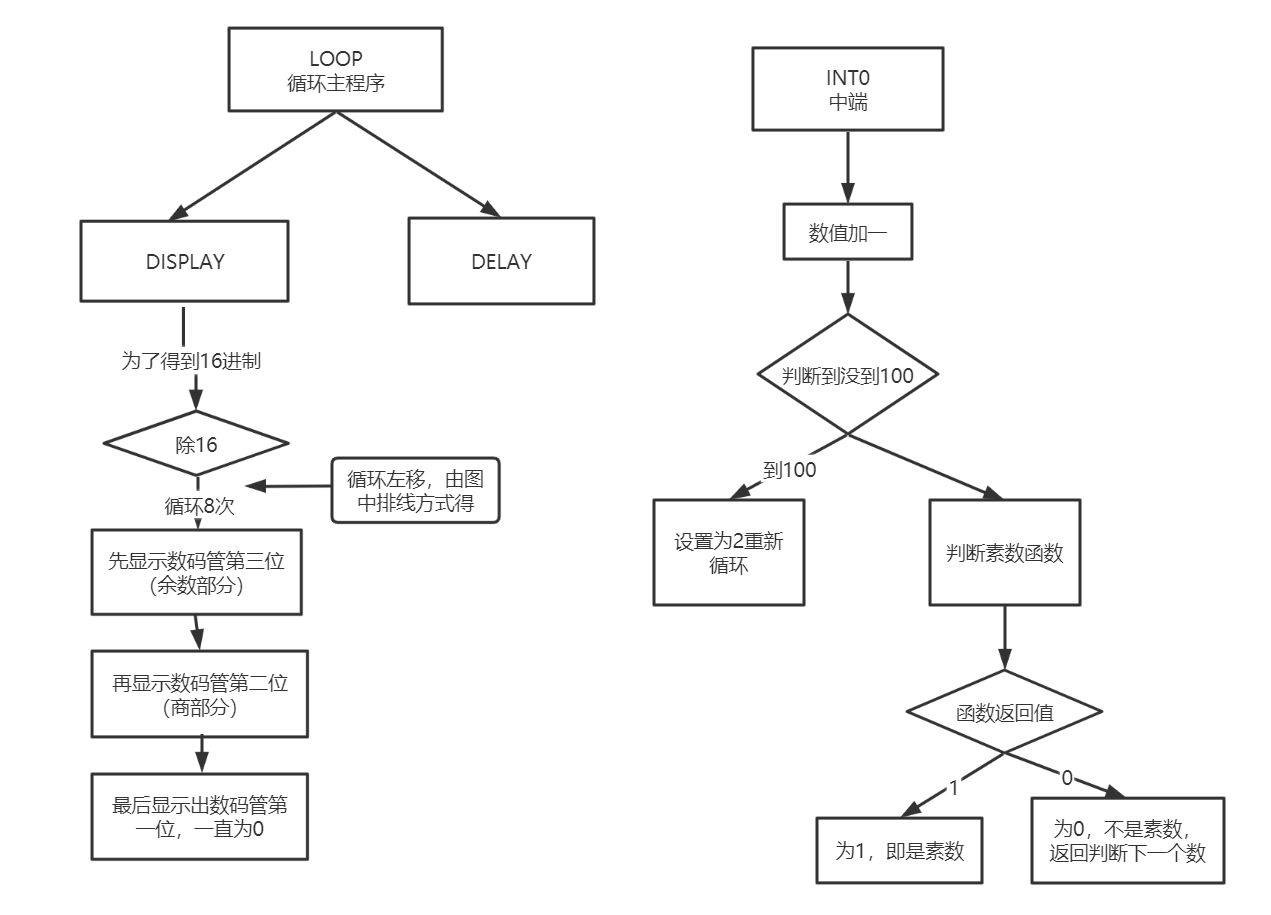
通过改变占空比就可以调整输出电压，从而达到模拟输出并控制电机转速的效果。使用单片机实现PWM，就是根据预定的占空比δ来输出0和1，这里δ就是控制变量。要达到更稳定的效果，可以采用累加进位法将总的周期内的0和1均匀分散开。设置一个累加变量x，每次加 N，若结果大于M，则输出1，并减去 M；否则输出0。这样整体的占空比也是N/M。

本实验的转速控制可以使用简单的比例控制算法，也就是当转速S大于预定值时，将输出0的个数减少；当转速小于预定值时，将输出0的个数增加。

1. 设计的局限性
2. LED闪烁的问题:通过按键调整速度的时候LED不可避免地发生闪烁;
3. 直流电机转速控制的精度:无法快速稳定在目标转速周围。
4. **程序设计、调试：**画出程序框图，对程序代码给出详细的分析和说明，并写出程序调试的详细过程。

**第一题：**

1. 程序框图



1. 代码分析与说明

程序清单

循环主程序：LOOP

中断程序:EINT0

中断结束程序：EINT0\_RE

素数判断程序：JUDGE\_PRIME

判断数值上界：JUDGE\_MAX

判断数值下界：JUDGE\_MIN

素数判断主程序：JUDGE\_MAIN

素数结果判断程序：为素数：JUDGE\_TRUE

非素数：JUDGE\_FALSE

数值增加级循环控制程序：NEXT\_PRIME

素数判断执行程序：JUDGE\_EXE

显示控制程序：DISPLAY\_A

延时程序：DELAY

**CLK BIT P1.4**

**DAT BIT P1.5**

**ORG 0000H**

**LJMP START**

**ORG 000BH**

**LJMP EINT0**

**ORG 0040H**

**START:**

**MOV TMOD,#01H ;SET T0 AT MODE 1**

**MOV TL0,#00H**

**MOV TH0,#00H**

**SETB ET0 ;ALLOW T0**

**SETB TR0 ;START T0**

**SETB EA ;START TIMER**

**mov R2, #8**

**mov R1, #2 ; 要显示的值，初值设为2**

**LOOP: ；循环的主程序**

**MOV A, R1**

**ACALL DISPLAY\_A ；显示R1的值**

**ACALL DELAY**

**JMP LOOP**

**;INT0------------------------------------------------------------中端**

**EINT0: ；每一秒修改一次，修改成下一个素数**

**MOV TL0, #0xB0**

**MOV TH0, #0x3C ；每次进入中断时先恢复初值**

**DJNZ R2, EINT0\_RE**

**MOV R2, #8**

**NEXT\_PRIME:**

**INC R1 ；数加一**

**INC R1**

**CJNE R1, #100, JUDGE\_EXE ；判断到没到100**

**MOV R1, #2 ；到100了就置为2重新开始循环**

**JUDGE\_EXE:**

**MOV A, R1 ；没到100就判断当前数是否为素数**

**ACALL JUDGE\_PRIME ；判断素数的函数，若是素数则A值为1**

**JZ NEXT\_PRIME ；若不是素数A为0，则跳回NEXT\_PRIME**

**EINT0\_RE:**

**RETI**

**;JUDGE\_R1\_IS\_PRIME--------------------------------------------------------判断是否是素数的函数**

**JUDGE\_PRIME:**

**MOV R4,#2 ；除数**

**JUDGE\_MAX:**

**CJNE R1, #100, JUDGE\_MIN ；判断到没到100，没到100跳转到J\_MIN**

**JMP JUDGE\_FALSE ；到100了跳转到J\_FALSE**

**JUDGE\_MIN:**

**CJNE R1, #2, JUDGE\_MAIN**

**JMP JUDGE\_TRUE**

**JUDGE\_MAIN: ；主函数**

**MOV A, R1**

**MOV B, R4**

**DIV AB ；用A除B**

**MOV A, B ；B赋值给A，整除 没有余数，A是0**

**JZ JUDGE\_FALSE ；上一句成立，则不是素数，跳转到J\_FALSE**

**INC R4**

**MOV A, R1**

**MOV B, R4**

**CJNE A, B, JUDGE\_MAIN ；R4加一后判断AB是否相等，不相等就继续除循环，相等就是素数了**

**JUDGE\_TRUE: ；最终结果**

**MOV A, #1 ；A值为1时是素数**

**RETI**

**JUDGE\_FALSE:**

**MOV A, #0 ；A值为0时不是素数**

**RETI**

**;DISPLAY\_A----------------------------------------------------显示部分**

**DISPLAY\_A:**

**MOV B,#16 ；为得到16进制**

**DIV AB ；AB相除，余数在B，B为最后一个数码管要显示的数**

**MOV R6, A ；防止A值丢失，先转存**

**MOV A, B ；循环8次，先展示B中的值**

**MOV R5, #08H**

**MOV DPTR,#TABLE ；查找TABLE中的码，输出**

**MOVC A, @A+DPTR**

**LOOP1:**

**RLC A ；进行的是循环左移，因为图中连线关系是反着的**

**CLR P1.4**

**MOV P1.5,C**

**SETB P1.4**

**DJNZ R5,LOOP1**

**MOV A, R6 ；再输出A中的值**

**MOV R5,#08H**

**MOV DPTR,#TABLE**

**MOVC A,@A+DPTR**

**LOOP2:**

**RLC A**

**CLR P1.4**

**MOV P1.5,C**

**SETB P1.4**

**DJNZ R5,LOOP2**

**MOV R5, #08H ；最后输出三个数码管中最前面那个0**

**MOV A, #00H**

**LOOP0:**

**RLC A**

**CLR P1.4**

**MOV P1.5,C**

**SETB P1.4**

**DJNZ R5,LOOP0**

**RET**

**;DELAY-------------------------------------------------------------延时**

**DELAY:**

**MOV R5,#2**

**DEL1:**

**MOV R6,#255**

**DEL2:**

**MOV R7, #225**

**DEL3:**

**DJNZ R7,DEL3**

**DJNZ R6,DEL2**

**DJNZ R5,DEL1**

**RET**

**TABLE: DB 03fH,006H,05bH,04fH,066H,06dH,07dH,007H,07fH,06fH,077H,07cH,039H,05eH,079H,071H**

**END**

1. 程序调试的详细过程

1）素数上下界设置失败：

数值范围超出要求，通过将数值设置相关子程序进行位置调整以达成条件。

2）素数执行程序与数值增加的不联动：

通过将数值增加子程序从素数判断程序中移出，置于主体程序中，完成数值的有效增加。

3）延迟时间不足：

单层循环延迟不够，需要两层循环延迟。

4）程序汇编后，仿真运行后数码管1显示，而目标数码管2，3不显示：

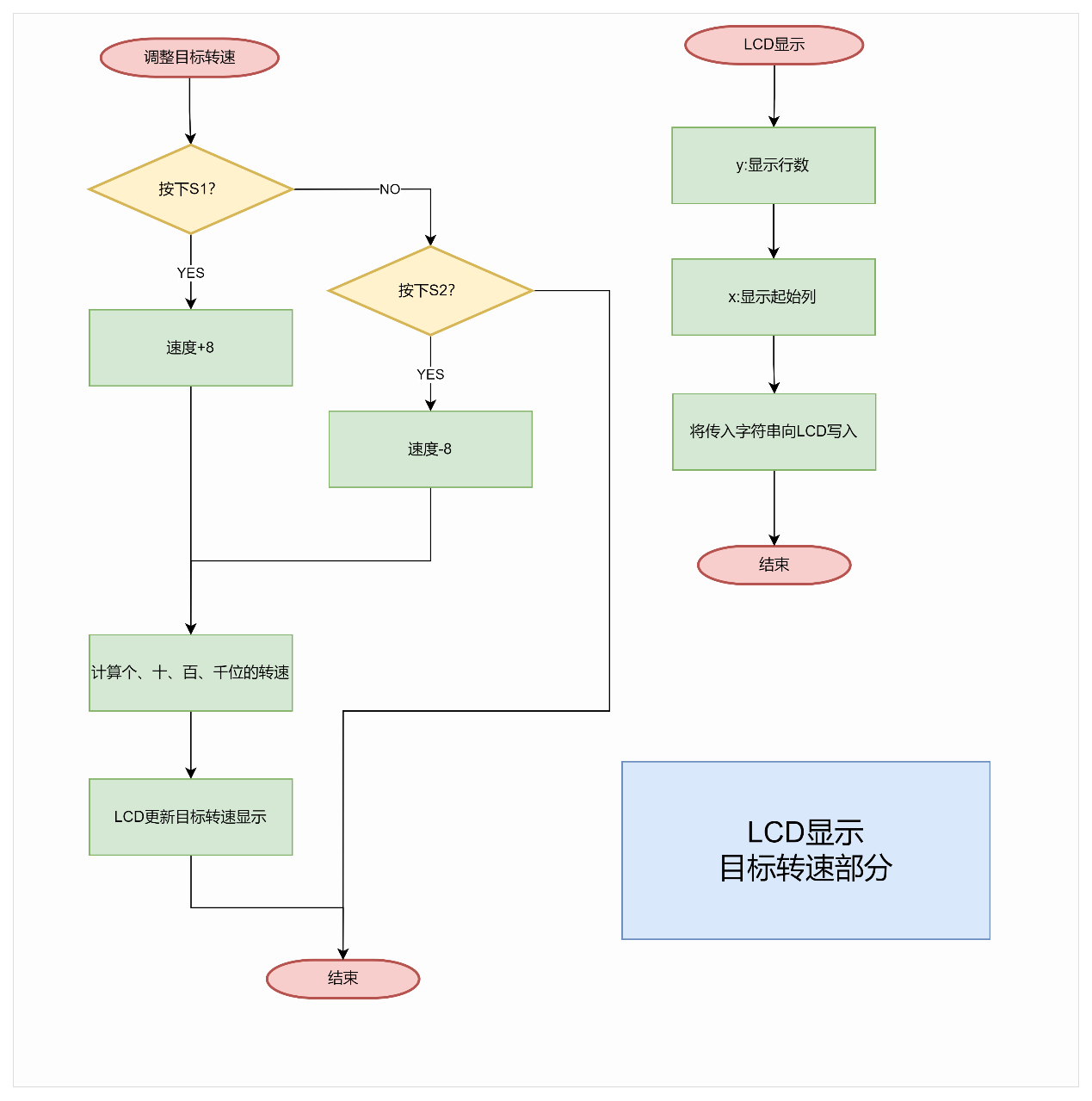
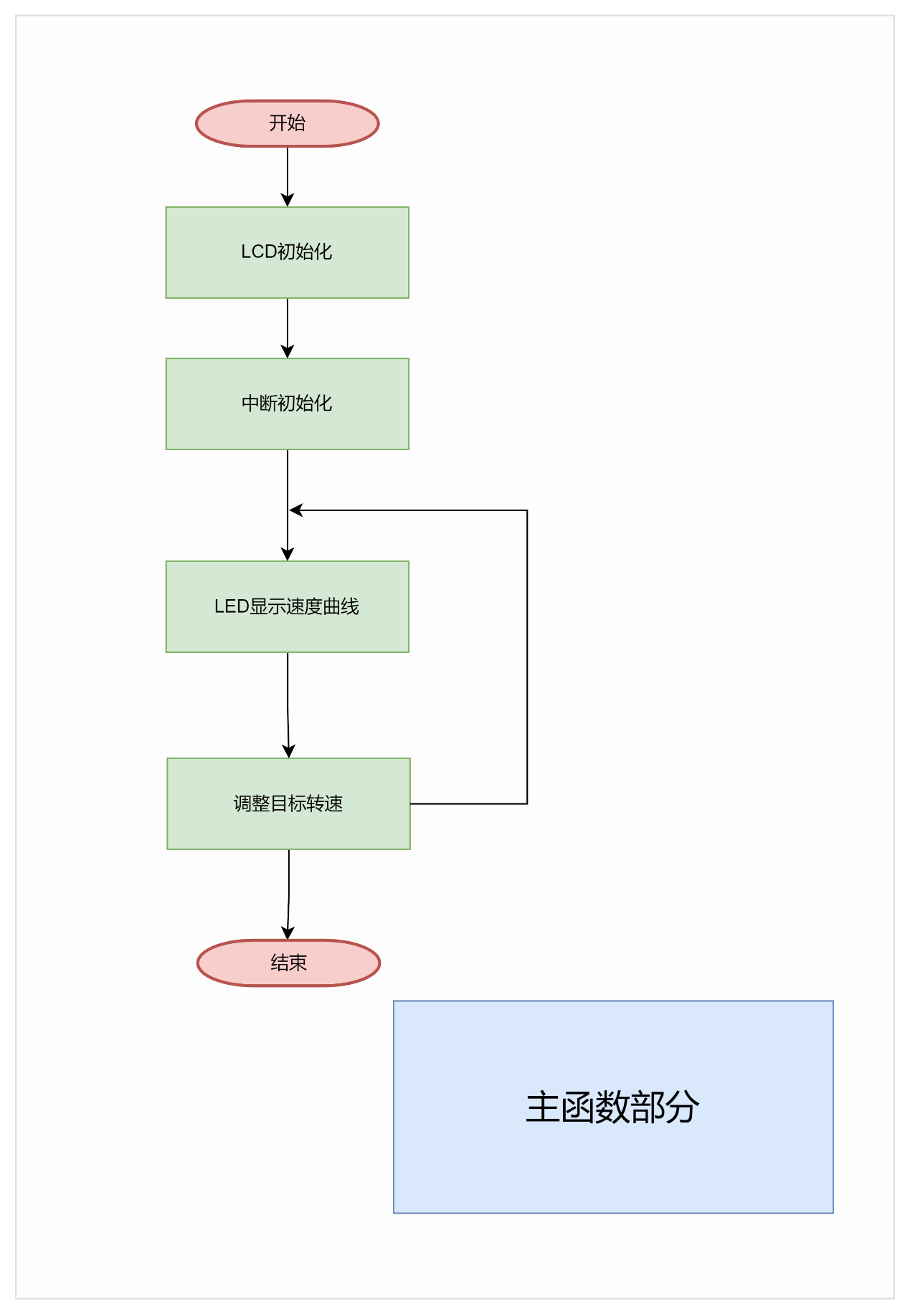
程序其实未编译成功，未显示正确信息，重新打开文件即可成功编译(会显示详细信息)。

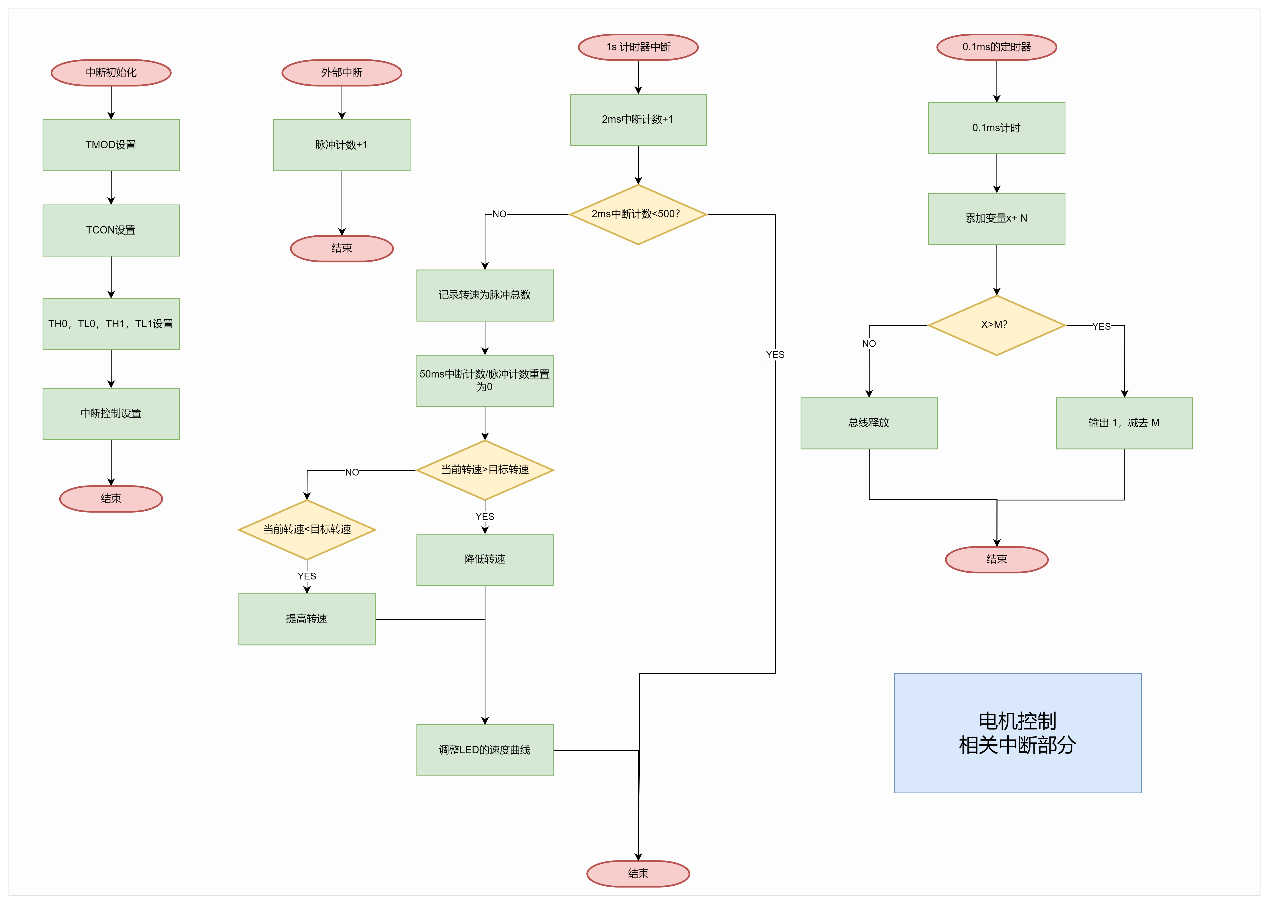
5）数码管1也有显示：

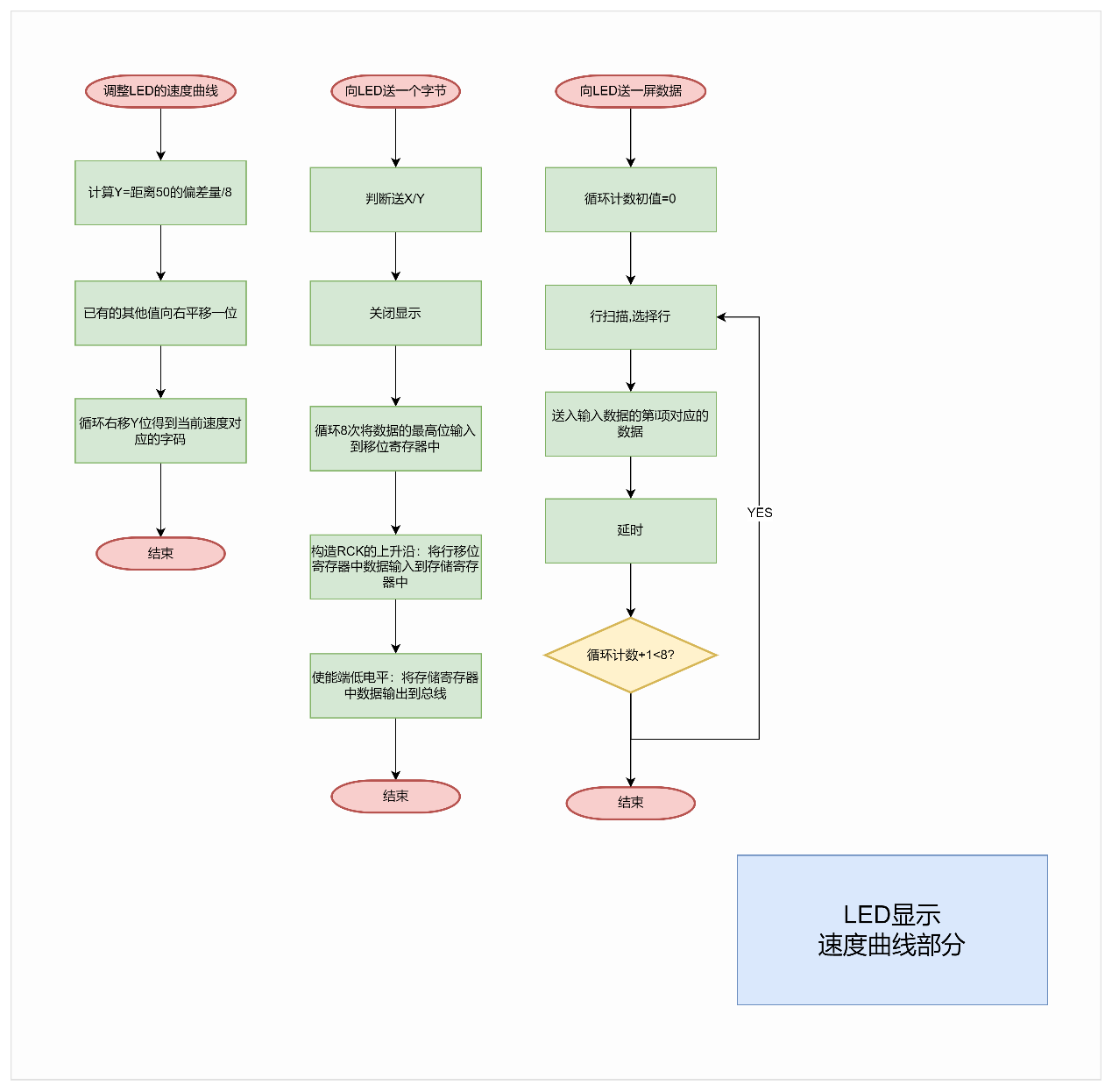
可能是受前一示例程序影响，在程序开头加上Mov P0,0 即可。

**第二题：**

1. 程序框图

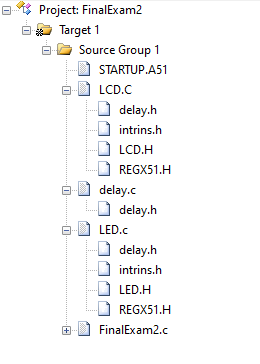






1. 代码分析与说明

程序代码文件结构：



|  |  |
| --- | --- |
| 程序文件 | 主要功能 |
| FinalExam2.c | 主函数部分，主要包含电机控制逻辑 |
| LED.H | LED部分头文件，包含相关引脚定义、LED函数声明 |
| LED.C | LED部分源文件，包含LED函数定义 |
| LCD.H | LCD部分头文件，包含相关引脚定义、LCD函数声明 |
| LCD.C | LCD部分源文件，包含LCD函数定义 |
| delay.h | 延时部分头文件，包含延时函数声明 |
| delay.c | 延时部分源文件，包含ms级软件延时函数定义 |

源程序代码及详细注释：

* **FinalExam2.c**

//FinalExam2.c   
#include <REGX51.H>   
#include<stdio.h>   
   
#define uchar unsigned char   
#define uint unsigned int   
   
#define THC0 0xf8   
#define TLC0 0x50 //2ms   
#define THC1 0xFF   
#define TLC1 0x9C //0.1ms   
//循环左移   
#define rol(value, bits) ((value << bits) | (value >> (sizeof(value)\*8 - bits)))   
   
#include "LCD.H"   
#include "LED.H"   
#include "delay.h"   
   
//设置速度（显示在LCD）的字码   
uchar SSChaCode[]={'S','e','t','u','p',':',' ',' ',' ',' ',' ','r','/','m','i','n'};   
//当前速度（显示在LCD）的字码：由于需要经常更新，导致LED闪烁故删除   
//uchar CSChaCode[]={'C','u','r',':',' ',' ',' ',' ',' ',' ',' ','r','/','m','i','n',' ',' ',' '};   
//LED速度大小的即Y轴的速度的TABLE//初始值全为1==空白屏   
uchar TAB\_Y[] = {0XFF, 0XFF, 0XFF , 0XFF, 0XFF, 0XFF, 0XFF, 0XFF};   
 //按键   
sbit S1=P3^6;   
sbit S2=P3^7;   
   
//直流电机输出口   
sbit MOTOR=P1^1;   
   
uint SSpeed=74;//Setup Speed(目标转速)   
uint CurSpeed=0; //Current Speed   
uint PCnt=0;//Pulse Count   
//占空比   
int N=180;//N段输出1   
int M=256; //M‐N 段输出 0，总体的占空比就是 N/M   
int X=0;//使用累加进位法的累加变量 x   
   
//函数声明   
void SetSpeed();//按下S1/S2调整目标转速   
void LEDVt(uint CurSpeed);//每秒调整LED的速度曲线   
void delay(uchar x);//延时函数   
   
void main()   
{   
 //InitialLCD显示   
 LCD\_init(); //LCD初始化   
 SSChaCode[7]=SSpeed/1000+'0';   
 SSChaCode[8]=SSpeed/100%10+'0';   
 SSChaCode[9]=SSpeed/10%10+'0';   
 SSChaCode[10]=SSpeed%10+'0'; //计算个、十、百、千位的初始目标转速   
 LCD\_Write\_String(0,0,SSChaCode); //LCD显示目标转速初值   
   
 //InitialInterrupt   
 //方式寄存器 TMOD   
 TMOD=0X11; //t1 t2定时   
 //控制寄存器TCON   
 IT0=1; //TCON中-外部中断0的触发方式选择位：1跳变沿触发方式，0电平触发方式   
 TR0=1; //定时器0停止运行（不计数）   
 TR1=1; //定时器1停止运行   
 //TH0,TL0,TH1,TL1   
 TH0=THC0;   
 TL0=TLC0; //2ms需要的定时器初值（1s=2ms\*500）   
 TH1=THC1;   
 TL1=TLC1; //0.1ms需要的定时器初值:65536-100=65436   
 //中断控制IntControl   
 EA=1; //中断允许   
 ET1=1; //timer1中断允许   
 ET0=1; //timer0中断允许   
 EX0=1; //INT0中断允许   
   
 while(1)   
 {   
 //LED显示速度曲线   
 SendData(TAB\_Y);   
 //设置目标转速   
 SetSpeed();   
   
 }   
}   
//按下S1/S2调整目标转速   
void SetSpeed()   
{   
 if(S1==0)//S1按下   
 {   
 delay(50); //延时防抖   
 if(S1==0)   
 {   
 SSpeed+=8; //每次速度+8   
 if(SSpeed>130) SSpeed=130; //速度上限130r/min   
 SSChaCode[7]=SSpeed/1000+'0';   
 SSChaCode[8]=SSpeed/100%10+'0';   
 SSChaCode[9]=SSpeed/10%10+'0';   
 SSChaCode[10]=SSpeed%10+'0'; //计算个、十、百、千位的转速   
 LCD\_Write\_String(0,0,SSChaCode); //LCD更新目标转速   
 }   
 }   
 if(S2==0)//S1按下   
 {   
 delay(50);   
 if(S2==0)   
 {   
 SSpeed-=8; //每次速度-8   
 if(SSpeed<50) SSpeed=50; //速度下限50r/min   
 SSChaCode[7]=SSpeed/1000+'0';   
 SSChaCode[8]=SSpeed/100%10+'0';   
 SSChaCode[9]=SSpeed/10%10+'0';   
 SSChaCode[10]=SSpeed%10+'0'; //计算个、十、百、千位的转速   
 LCD\_Write\_String(0,0,SSChaCode); //LCD更新目标转速   
 }   
 }   
}   
//外部中断：电机转速就是一秒钟之内 INT0 的中断个数   
void int0() interrupt 0   
{   
 PCnt++;   
}   
//1s的定时器中断1   
void t0() interrupt 1   
{   
 static unsigned int time=0;//1S定时器所需的计数器（500）   
   
 TH0=THC0;   
 TL0=TLC0;//2ms   
   
 time++;   
   
 if(time>500) //到达1s   
 {   
 CurSpeed=PCnt/6; //记录转速：仿真中砝码盘数n=360，即每转产生360个脉冲计数，故计算时需/6得到单位为r/min   
 time=0; //重置为0   
 PCnt=0; //重置为0   
 if(CurSpeed>SSpeed) N--; //降低转速   
 else if(CurSpeed<SSpeed) N++; //提高转速   
 LEDVt(CurSpeed); //调整LED的速度曲线   
 }   
}   
//0.1ms的定时器中断0，用来调整控制变量（占空比）   
void timer\_1() interrupt 3   
{   
 TH1=THC1;   
 TL1=TLC1; //0.1ms:65536-100=65436   
 //采用累加进位法将总的周期内的 0 和 1 均匀分散开   
 X+=N;//累加变量x，每次加 N   
 if(X>M) {MOTOR=1;X-=M;}//若结果大于 M，则输出 1，并减去 M   
 else MOTOR=0;//否则输出 0   
}   
//每秒调整LED的速度曲线   
void LEDVt(uint CurSpeed)   
{   
 uint i; //循环变量   
 uint x; //记录距离50的偏差量/8   
 uchar cc=0XFE; //新的当前速度对应的字码（初始值就是50对应的仅Y=0亮）   
   
 if(CurSpeed<50) CurSpeed=50; //若当前速度<50，超过下界按最小值显示   
 x=(CurSpeed-50)/8; //记录距离50的偏差量/8   
 if(x>7) x=7; //超过上界按最大值显示（Y=7）   
   
 for(i=7;i>0;i--)   
 {   
 TAB\_Y[i]=TAB\_Y[i-1]; //已有的其他值向右平移一位   
 }   
 TAB\_Y[0]=rol(cc,x); //循环右移偏差量，得到新的当前速度对应的字码   
}   
//延时   
void delay(uchar x)   
{   
 uint i,j;   
 for(i=x;i>0;i--)   
 for(j=50;j>0;j--);   
}

* **LED.H**

//LED.H   
#ifndef \_\_LED\_H\_\_   
#define \_\_LED\_H\_\_   
   
#include <REGX51.H>   
#include<intrins.h>   
#include"delay.h"   
  
sbit DATA\_X = P0^0; //数据输入   
sbit CLK\_X = P0^1; //移位寄存器时钟输入   
sbit WR\_X = P0^2; //存储器时钟输入   
sbit EN\_X = P0^7; //使能端   
   
sbit DATA\_Y = P0^3; //数据输入   
sbit CLK\_Y = P0^5; //移位寄存器时钟输入   
sbit WR\_Y = P0^6; //存储器时钟输入   
sbit EN\_Y = P0^4; //使能端   
  
void SendByte(unsigned char x,unsigned char dat); //向LED送一个字节   
void SendData(unsigned char \*s); //向LED送一屏数据   
#endif

* **LED.C**

//LED.C   
#include"LED.H"   
//X轴的TABLE,用于行扫描   
unsigned char code TAB\_X[] = {0x80, 0X40, 0X20 , 0X10, 0X08, 0X04, 0X02, 0X01};   
//向LED送一个字节   
void SendByte(unsigned char x, unsigned char dat)   
{   
 unsigned char a;   
 //x=0,control x; x=1,control y   
 if(x==0){ //控制X   
 EN\_X=1; //传送数据时关闭显示，减少闪烁   
 CLK\_X=0; //移位寄存器时钟输入清0，方便后构造上升沿   
 WR\_X=0; //存储器时钟输入清0，方便后构造上升沿   
 //串转并：循环8次   
 for(a=0;a<8;a++)   
 {   
 DATA\_X=dat>>7; //每次传送数据的最高位   
 dat<<=1;   
 CLK\_X=1; //将行的前8位在SRCK的上升沿输入到移位寄存器中   
 CLK\_X=0; //SRCK重新清零   
 }   
 WR\_X=1; //构造RCK的上升沿：将行移位寄存器中数据输入到存储寄存器中   
 WR\_X=0; //RCK重新清零   
 EN\_X=0; //使能端低电平：将存储寄存器中数据输出到总线   
 }else{ //控制Y   
 EN\_Y=1; //传送数据时关闭显示，减少闪烁   
 CLK\_Y=0; //移位寄存器时钟输入清0，方便后构造上升沿   
 WR\_Y=0; //存储器时钟输入清0，方便后构造上升沿   
 //串转并：循环8次   
 for(a=0;a<8;a++)   
 {   
 DATA\_Y=dat>>7; //每次传送数据的最高位   
 dat<<=1;   
 CLK\_Y=1; //将行的前8位在SRCK的上升沿输入到移位寄存器中   
 CLK\_Y=0; //SRCK重新清零   
 }   
 WR\_Y=1; //构造RCK的上升沿：将行移位寄存器中数据输入到存储寄存器中   
 WR\_Y=0; //RCK重新清零   
 EN\_Y=0; //使能端低电平：将存储寄存器中数据输出到总线   
 }   
}   
void SendData(unsigned char \*s) //向LED送一屏数据   
{   
 unsigned char i=0; //循环变量   
 for(i=0;i<8;i++) //由于是8\*8的矩阵，故循环8次   
 {   
 SendByte(0,TAB\_X[i]); //行扫描   
 SendByte(1,s[i]); //送入输入数据的第i项对应的数据   
 DelayMs(2); //延时2ms   
 }   
}

* **LCD.h**

//LCD.h   
#ifndef \_\_LCD\_H\_\_   
#define \_\_LCD\_H\_\_   
#include <REGX51.H>   
#include<intrins.h>   
#include"delay.h"   
//相关引脚定义   
sbit RS = P3^3; //数据/指令选择(H:数据 L: 指令)   
sbit RW = P3^4; //读/写选择（H:读 L:写）   
sbit EN = P3^5; //使能信号   
//宏定义   
#define RS\_CLR RS=0   
#define RS\_SET RS=1   
#define RW\_CLR RW=0   
#define RW\_SET RW=1   
#define EN\_CLR EN=0   
#define EN\_SET EN=1   
//LCD相关函数声明   
void LCD\_WriteString(unsigned char x,unsigned char y,unsigned char \*s);//LCD写入字符串   
void LCD\_init();//LCD初始化   
void LCD\_WriteCom(unsigned char com);//LCD写命令   
void LCD\_WriteData(unsigned char date);//LCD写字符   
#endif

* **LCD.c**

//LCD.c   
#include"LCD.H"   
void LCD\_WriteCom(unsigned char com) //LCD写命令   
{   
 RS\_CLR; //RS低电平：选择命令   
 RW\_CLR; //RW低电平：选择写入   
 P2=com; //将命令送入P2口   
 DelayMs(3); //延时3ms，等待数据稳定   
 EN\_SET; //写入时序   
 DelayMs(3); //延时3ms：保持时间   
 EN\_CLR;   
}   
void LCD\_WriteData(unsigned char date) //LCD写字符   
{   
 RS\_SET; //RS高电平：选择数据   
 RW\_CLR; //RW低电平：选择写入   
 P2=date; //将数据送入P2口   
 DelayMs(3); //延时3ms，等待数据稳定   
 EN\_SET; //写入时序   
 DelayMs(3); //延时3ms：保持时间   
 EN\_CLR;   
}   
void LCD\_init() //LCD初始化   
{   
 write\_com(0x38); //开显示   
 write\_com(0x0c); //开显示不显示光标   
 write\_com(0x06); //写一个指针加1   
 write\_com(0x01); //清屏   
}   
void LCD\_WriteString(unsigned char x,unsigned char y,unsigned char \*s) //LCD写入字符串   
{   
 if (y == 0) //选择0行   
 {   
 write\_com(0x80 + x); //数据指针起点：0行+x列   
 }   
 else //选择1行   
 {   
 write\_com(0xC0 + x); //数据指针起点：1行+x列   
 }   
 while (\*s) //将字符串作数据送入LCD   
 {   
 write\_data( \*s); //循环向LCD送入字符串中字符   
 s ++;   
 }   
}

* **delay.h**

//delay.h   
#ifndef \_\_DELAY\_H\_\_   
#define \_\_DELAY\_H\_\_   
   
void DelayUs2x(unsigned char t);//μS延时函数（12MHZ晶振）   
void DelayMs(unsigned char t);//mS延时函数（12MHZ晶振）   
   
#endif

* **delay.c**

//delay.c   
#include "delay.h"   
void DelayUs2x(unsigned char t) //uS延时函数（12MHZ晶振）  
{   
 while(--t);   
}   
void DelayMs(unsigned char t) //mS延时函数（12MHZ晶振）   
{   
 while(t--) //差不多1mS   
 {   
 DelayUs2x(245);   
 }   
}

1. 程序调试的详细过程

本程序首先完成电机控制部分（包含LCD显示目标转速&当前转速），后再此基础上增加LED显示速度曲线部分。程序调试过程中，出现的编译错误较少，程序可以build后又发现逻辑错误的情况更多。

* 编译错误:

主要是由于不熟悉keil编程中语法差异造成的，比如定义变量时出现：先提示信息error C141: syntax error near 'int'，后又提示信息未定义变量，这是由于不支持中途定义，需把变量定义放到函数最前面。

* 逻辑错误：

在电机控制部分：

1. 刚开始运行程序时，发现转速总是直接达到max，后来给电机输出控制引脚加上示波器才发现占空比不对，发现是由于输出的0、1颠倒，将输出反过来转速即可控制；
2. 初始时，转速的计算也有错误，由于本电路中电机的砝码盘数设置不同，设置为360，故每秒计算转速时需要除6，才为以r/min为单位的真实转速；
3. 开始发现转速控制的精度不高，即在目标转速的周围波动范围大，需要很久才能稳定，后来对电机的参数进行了调整，增大Pulses per Revolution（砝码盘数），调整L298N驱动电压后有了很大改善。

在LED显示部分：

1. 开始主要是调试LED的静态显示，显示总是错位，后来发现是行扫描顺序出错，应该先选择行，再送入对应的列数据，调整后即可正确显示；
2. 将LED部分加入写好的电机控制部分后，出现严重的闪烁，后来发现是由于LCD显示目标速度也放入主函数的循环中，相当于每次去处理显示LCD都大大拉长了LED的显示延时，导致严重闪烁，所以我把LCD显示目标速度部分修改成仅当目标速度有变化时才更新显示LCD ，这样就不会有严重闪烁了。
3. **过程分析和总结：**总结每个人负责的任务，对整个设计工作过程进行归纳和综合，对设计中所存在的问题和不足进行分析和总结，提出解决的方法、措施、建议和对这次设计的认识、收获和提高。
4. 每个人负责的任务

第一题的整体设计、程序实现与调试由张乐佳同学与刘怡凝同学共同负责，第二题的整体设计、程序实现与调试由江琳艺同学负责，报告的撰写由孟珺逸同学负责。

1. 整个设计的归纳和综合

在设计过程中，首先需要有一个清晰的思路，画出相应的流程图，先确定整体框架，再实现具体功能和细节。设计过程不可能一蹴而就，需要不断调试，发现问题、分析问题然后解决问题。设计过程中要善于利用已有资料以及各种渠道获取所需知识，在设计的同时中不断学习进步。

1. 问题和不足的分析与总结及解决方法

**第一题：**参考实验二和实验图，将设计过程分为五个主体部分：循环主程序、显示、延时、中断和函数。

设计中出现的问题：

最开始设置的间隔时间不正确：

50ms需要的定时器初值:65536-50000=15536（1S=50ms\*20）。

**第二题：**首先完成电机控制部分（包含LCD显示目标转速&当前转速），后再此基础上增加LED显示速度曲线部分。一开始对电路图设计毫无头绪，花了很多时间，后来在网络参照了很多设计图，进行了精简改进。代码部分思路还比较清晰，因为实验四和实验六中都做过相关实验，不过也由于电路设计的不同出现了部分代码逻辑错误。

设计中仍存在不足的部分：

1. LED闪烁问题

通过按键调整速度的时候LED不可避免地发生闪烁。

问题原因分析：

1. 按键需要延时消抖，延时耗时；
2. LCD需要更新显示，LCD显示耗时。

解决方法：

1. 延时消抖改为外部中断脚+定时器消抖；
2. 减少LCD内延时时间，减少处理时间，改善闪烁的情况。
3. 直流电机转速控制的精度

即使花了很多时间在减少误差上，比如调整电机参数，调整测速时间，占空比中M的值，每次占空比中N的加减大小，但是仍然无法快速稳定在目标转速周围。

解决方法：

a)改用其他控制算法，如PID算法

b)可使用多个计数器检测脉冲数,后取其平均值，可以减少误差，提高控制精度。

1. 认知、收获和提高

这次课程设计，既是对课程理论知识的一次复习和巩固，还让我们丰富了更多与该专业相关的其他知识，比如软件应用等，在摸索中学习，在摸索中成长，在学习的过程中带着问题去学我们发现效率很高，这是我们做这次课程设计的又一收获。在真正设计之前我们做了相当丰富的准备，首先巩固了课程理论，再一遍熟悉课程知识的构架，然后结合加以理论分析、总结，有了一个清晰的思路和一个完整的的软件流程图之后才着手设计。在设计程序时，我们不能妄想一次就将整个程序设计好，反复修改、不断改进是程序设计的必经之路，养成注释程序的好习惯是非常必要的，一个程序的完美与否不仅仅是实现功能，而应该让人一看就能明白你的思路，这样也能为资料的保存和交流提供了方便。

通过这次设计工作，我们体会到了团队合作的重要性，通过分析问题与解决问题的过程，让我们处理问题更加有条理，思路更加清晰明了，分析解决问题和实践能力都有所提高。此次设计过程也是对我们所学知识的一次综合运用，让我们认识到学习单片机时学好基础知识的重要性，如数字逻辑电路知识、操作系统的知识、微机系统的知识等都在单片机的实验过程中有所应用。

本次设计收获了很多知识，最基本的就是能熟练使用peotues、keil软件，也学习了二者联调的方法，慢慢摸索明白了该从何入手设计电路图。