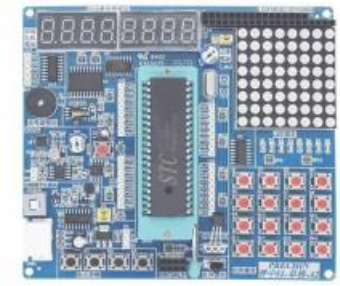
51单片机学习——1天学完普中基本实验例程，走马观花式学习，大家切勿效仿。

原创



> 本文代码均来自普中51实验例程

本人有一点STM32基础，但没有玩过51，最近一时兴起，想见见51真面目。

关于51单片机的理论和历史我在这就略过了，我的目标是把所有功能简单过一遍，再练几个综合项目，过程越快越好，毕竟我不能保证我这一时的兴致能维持很久。□

文章目录

开发平台：

实验1：点亮第一个LED

实验2：LED闪烁

实验3：LED流水灯

实验4：蜂鸣器

实验5：动态数码管显示

实验6：独立按键

实验7：矩阵按键

实验8：单片机IO扩展--74HC595

实验9：LED点阵（点亮一个点）

实验10：LED点阵(显示数字)

实验11：直流电机

实验12：外部中断0

实验13：外部中断1

实验14：定时器0中断

实验15：定时器1中断

实验16：EEPROM-IIC

实验17：DS18B20温度传感器

实验18：DS1302时钟

实验19：红外通信

实验20：AD模数转换

1. 电位器AD值
2. 光敏电阻AD值
3. 热敏电阻AD值
4. 外部输入AD值

实验21：DA数模转换

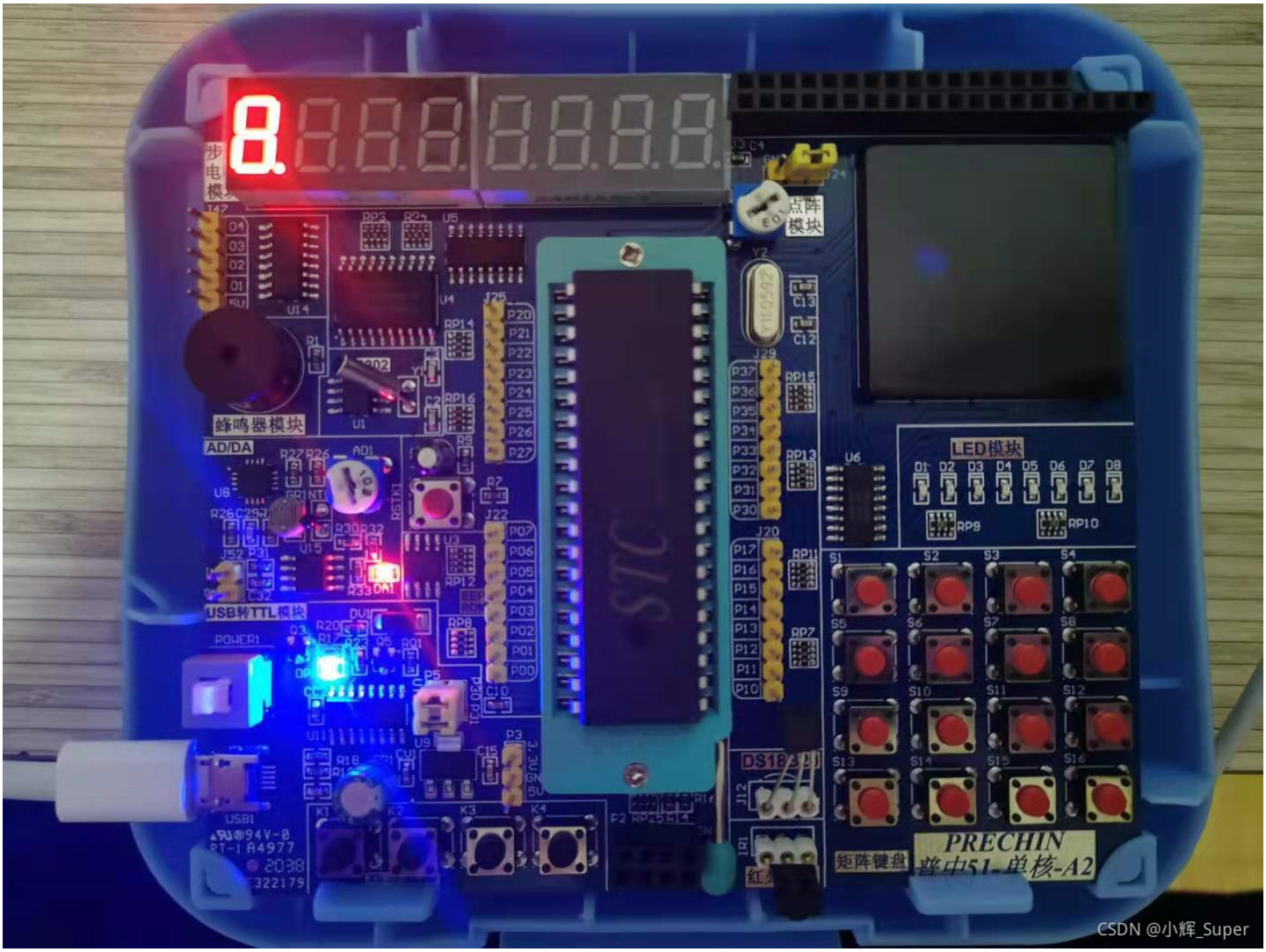
实验21：串口通信

实验22：LCD1602液晶

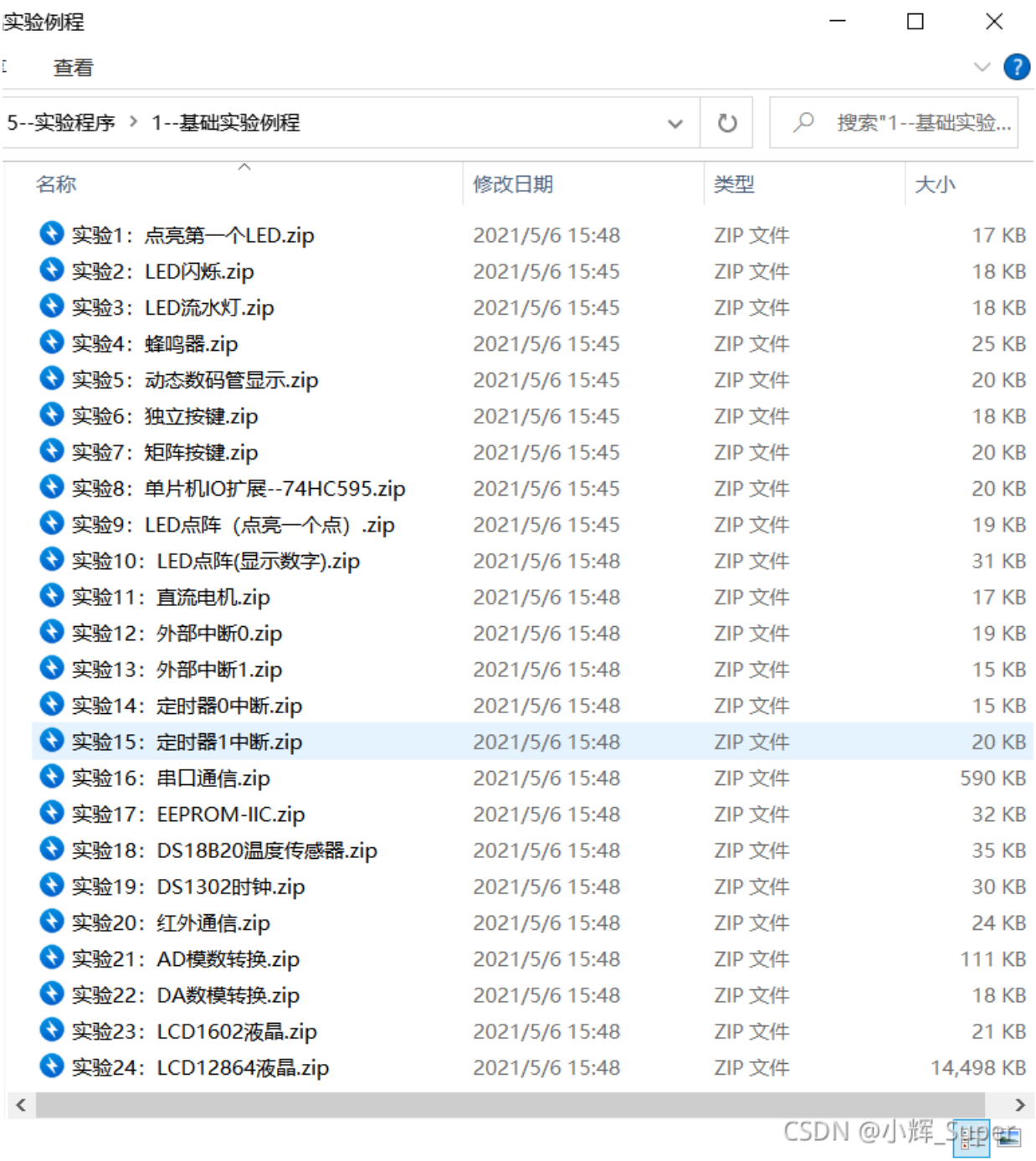
实验23：LCD12864液晶（空）

开发平台：

普中51-单核-A2


步电模块
蜂鸣器模块
AD/DA
USB转TTL模块
POWER1
点阵模块
LED模块
矩阵键盘
PRECHIN
普中51-单核-A2
CSDN @小辉_Super

今天是学习51第一天，趁着这兴奋劲，就多学点，定一个小目标，把基础实验例程都无脑过一遍：



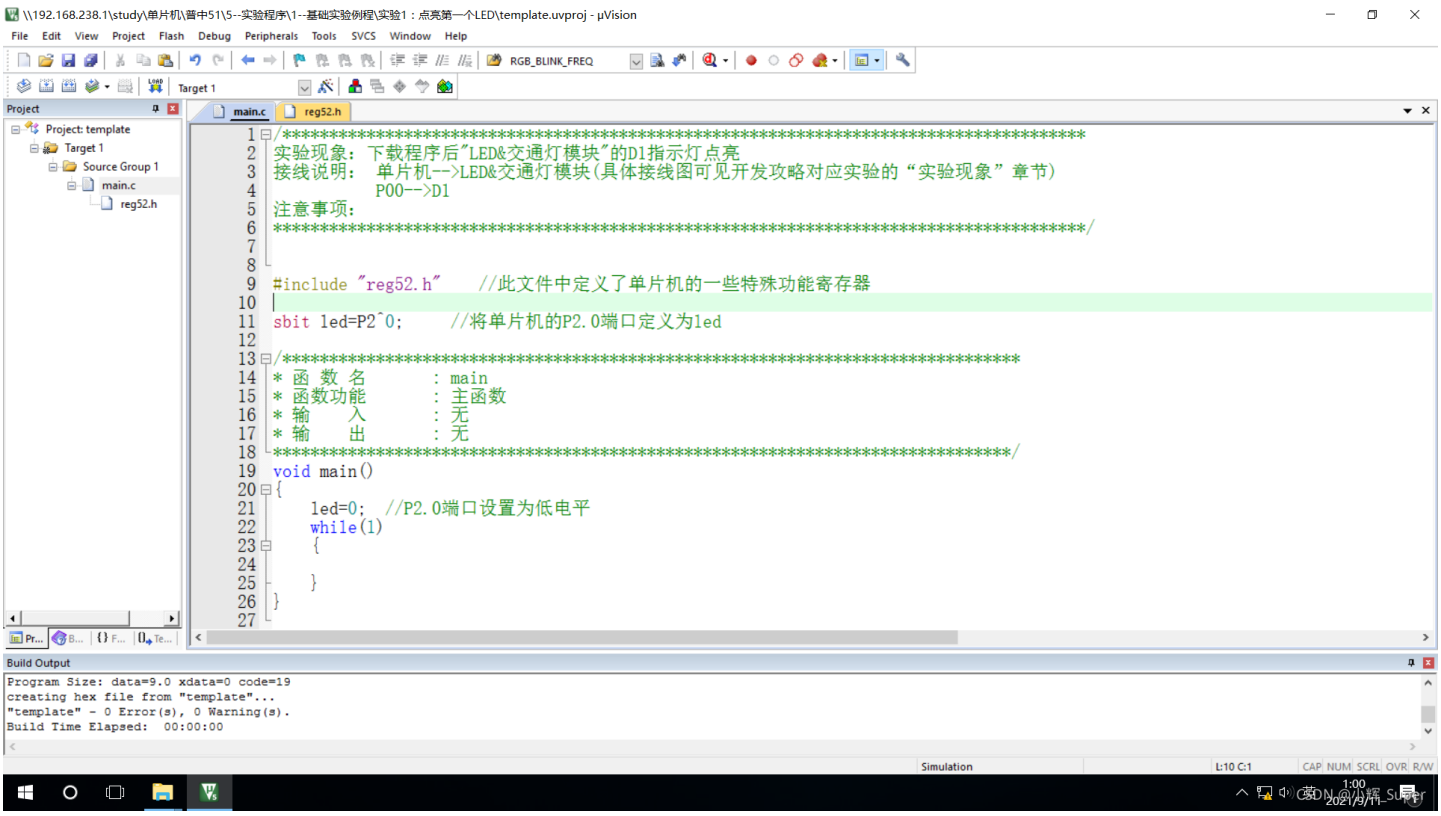
实验1：点亮第一个LED

从工程的文件看，51单片机的系统核心文件只有reg52.h，这也说明它的功能少，外设简单，毕竟这是8位单片机。

在main.c的第11行出现了sbit，这是C51特有的变量类型，用来操作1位的数据。

sbit led = P2^0 的作用是将P2.0端口定义为led，当给led赋值0时，相当于把单片机P0.0引脚拉低，由于开发板的LED灯的正极接了VCC，负极接P2.0，所以P2.0为低电平时，LED就亮了。

普中给的注释中存在一些问题，比如P00-->D1，D1是开发板丝印，但P00错了，应该是P20，以代码为准



我们再来看看reg52.h，它是51系列单片机的头文件，定义了一些寄存器和引脚。

下面的这个文件添加了注释（参考了CSDN博主南木Sir的文章），由于51是8位单片机，所以这些寄存器也都是8位的，细节我就不去探究了。

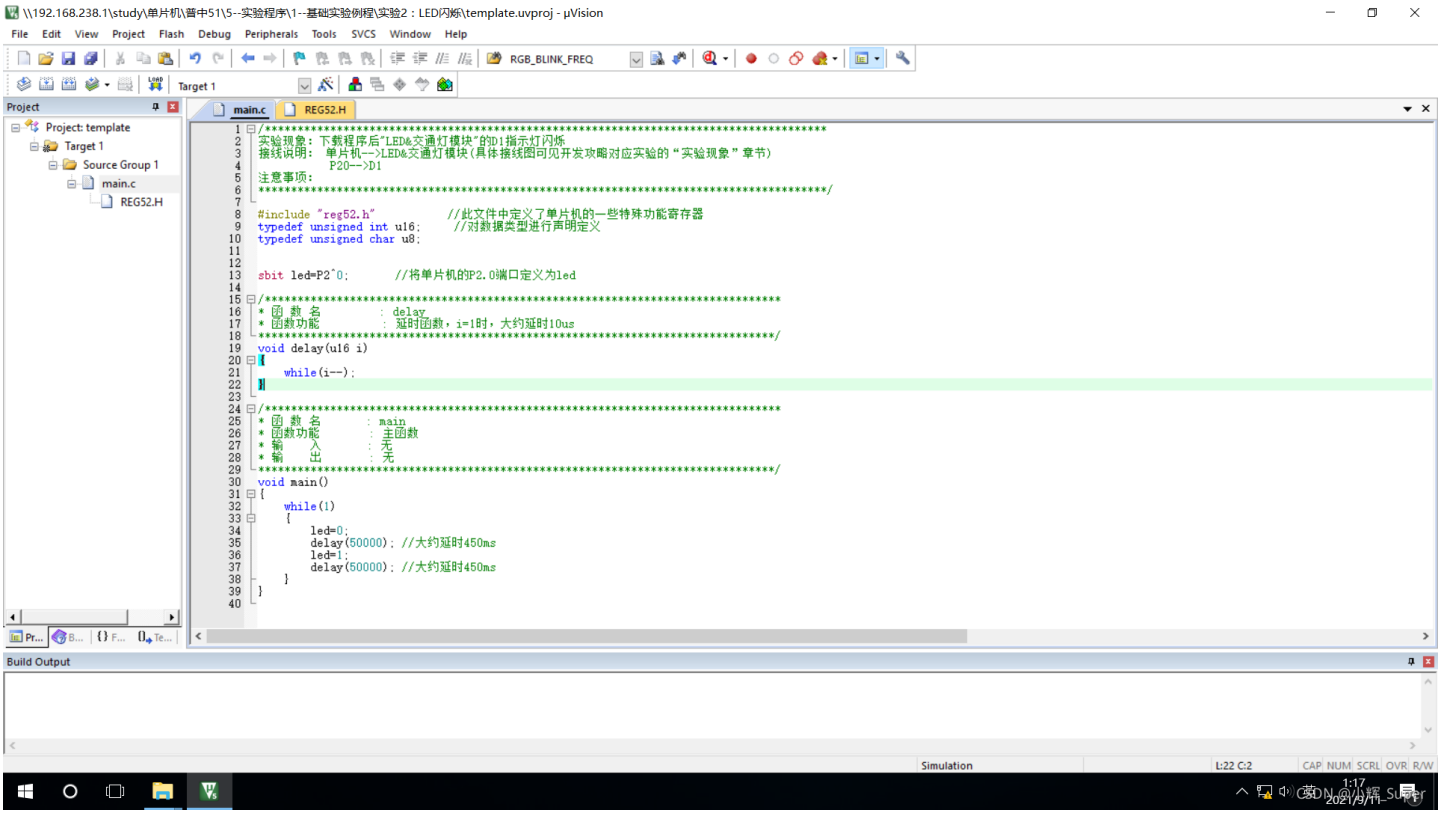
```
/*--------------------------------------------------------------------------
REG52.H

Header file for generic 80C52 and 80C32 microcontroller.
Copyright (c) 1988-2002 Keil Elektronik GmbH and Keil Software, Inc.
All rights reserved.
--------------------------------------------------------------------------*/


#ifndef __REG52_H__
#define __REG52_H__


/*  特殊字节（8位）寄存器 */
sfr P0 = 0x80; // P0口特殊寄存器寻址位
sfr P1 = 0x90; // P1口特殊寄存器寻址位
sfr P2 = 0xA0; // P2口特殊寄存器寻址位
sfr P3 = 0xB0; // P3口特殊寄存器寻址位
sfr PSW = 0xD0; //程序状态字寄存器
sfr ACC = 0xE0; //累加器
sfr B = 0xF0; //B 特殊寄存器
sfr SP = 0x81; //堆栈指针寄存器
sfr DPL = 0x82; //数字指针（低位）
sfr DPH = 0x83; //数字指针（高位）
sfr PCON = 0x87; //电源控制寄存器
sfr TCON = 0x88; //定时器/计数器 0 和 1 控制
```

```
sfr TMOD = 0x89; //定时器/计数器 0 和 1 模式
sfr TL0 = 0x8A; //定时器/计数器 0 低8位寄存器
sfr TL1 = 0x8B; //定时器/计数器 1低8位寄存器
sfr TH0 = 0x8C; //定时器/计数器 0高8位寄存器
sfr TH1 = 0x8D; //定时器/计数器 1高8位寄存器
sfr IE = 0xA8; //中断允许寄存器
sfr IP = 0xB8; //中断优先寄存器（低）
sfr SCON = 0x98; //串口控制寄存器
sfr SBUF = 0x99; //串口数据缓冲器

/*  8052扩展寄存器 */
sfr T2CON = 0xC8; //定时器/计数器2 控制
sfr RCAP2L = 0xCA; //定时器/计数器2 重载/捕捉低位
sfr RCAP2H = 0xCB; //定时器/计数器2 重载/捕捉高位
sfr TL2 = 0xCC; //定时器/计数器2 低位
sfr TH2 = 0xCD; //定时器/计数器2 高位



/*  位寄存器  */


/*  PSW （程序状态字寄存器） */
sbit CY = PSW^7; //进位、借位标志。进位、借位CY=1；否则CY=0
sbit AC = PSW^6; //辅助进位、借位标志。当D3向D4有借位或进位时，AC=1；否则AC=0
sbit F0 = PSW^5; //用户标志位
sbit RS1 = PSW^4; //寄存器组选择控制位1
sbit RS0 = PSW^3; //寄存器组选择控制位0
sbit OV = PSW^2; //溢出标志。有溢出OV=1，否则OV=0
sbit F1 = PSW^1; //保留位，无定义
sbit P = PSW^0; //8052 only 奇偶校验标志位，由硬件置位或清0；
    //存在ACC中的运算结果有奇数个1时P=1，否则P=0


/*  TCON （定时器/计数器 0 和 1 控制） */
sbit TF1 = TCON^7; //定时器1溢出标志位。当字时器1计满溢出时，
    //由硬件使TF1置“1”，并且申请中断。
    //进入中断服务程序后，由硬件自动清“0”，在查询方式下用软件清 “0”
sbit TR1 = TCON^6; //定时器1运行控制位。由软件清“0”关闭定时器1。
    //当GATE=1，且INT1为高电平时，TR1置“1”启动定时器1；
    //当GATE=0，TR1置“1” 启动定时器1
sbit TF0 = TCON^5; //定时器0溢出标志。其功能及操作情况同TF1。
sbit TR0 = TCON^4; //定时器0运行控制位。其功能及操作情况同TR1。
sbit IE1 = TCON^3; //外部中断1请求标志。
sbit IT1 = TCON^2; //外部中断1触发方式选择位。
sbit IE0 = TCON^1; //外部中断0请求标志。
sbit IT0 = TCON^0; //外部中断0触发方式选择位。


/* IE （中断允许寄存器） */
sbit EA = IE^7; //允许/禁止总中断
sbit ET2 = IE^5; //8052 only 允许/禁止定时器2(T2)中断
sbit ES = IE^4; //允许/禁止串口中断
sbit ET1 = IE^3; //允许/禁止T1溢出中断
sbit EX1 = IE^2; //允许/禁止外部中断1(INT1)
sbit ET0 = IE^1; //允许/禁止T0溢出中断
sbit EX0 = IE^0; //允许/禁止外部中断0(INT0)


/* IP （中断优先寄存器低） */
```

```
sbit PT2 = IP^5; //定时/计数器T2优先级设定位。
sbit PS = IP^4; //串行口优先级设定位;
sbit PT1 = IP^3; //定时/计数器T1优先级设定位;
sbit PX1 = IP^2; //外部中断0优先级设定位;
sbit PT0 = IP^1; //定时/计数器T0优先级设定位;
sbit PX0 = IP^0; //外部中断0优先级设定位;

/* P3 （第二功能） */
sbit RD = P3^7; //外部数据存储器读脉冲
sbit WR = P3^6; //外部数据存储器写脉冲
sbit T1 = P3^5; //定时器/计数器1外部输入
sbit T0 = P3^4; //定时器/计数器0外部输入
sbit INT1 = P3^3; //外部中断0。
sbit INT0 = P3^2; //外部中断1。
sbit TXD = P3^1; //串行数据输出口
sbit RXD = P3^0; //串行数据输入口


/* SCON（控制寄存器，它是一个可寻址的专用寄存器，用于串行数据的通信控制） */
sbit SM0 = SCON^7; //串行口工作方式控制位0。
sbit SM1 = SCON^6; //串行口工作方式控制位1。
sbit SM2 = SCON^5; //多机通信控制位。
sbit REN = SCON^4; //允许接收位。用于控制数据接收的允许和禁止，
    // ren=1时，允许接收，ren=0时，禁止接收。
sbit TB8 = SCON^3; //发送接收数据位8。
sbit RB8 = SCON^2; //接收数据位8。
sbit TI = SCON^1; //发送中断标志位。
sbit RI = SCON^0; //接收中断标志位。


/*  P1（第二功能） */
sbit T2EX = P1^1; // 8052 only 定时/计数器2捕捉/重装入触发
sbit T2 = P1^0; // 8052 only 定时/计数器2外部输入


/*  T2CON  （定时器/计数器2 控制） */
sbit TF2 = T2CON^7; //定时器2 溢出标记
sbit EXF2 = T2CON^6; //定时器2 外部标记
sbit RCLK = T2CON^5; //0=串口时钟应用定时器1 溢出,  1=定时器 2
sbit TCLK = T2CON^4; //0=串口时钟应用定时器1 溢出,  1=定时器 2
sbit EXEN2 = T2CON^3; //定时器 2 外部允许
sbit TR2 = T2CON^2; //0=停止定时器, 1=开始定时器
sbit C_T2 = T2CON^1; //0=定时器, 1=计数器
sbit CP_RL2 = T2CON^0; //0=重载, 1=捕捉选择。


#endif
```

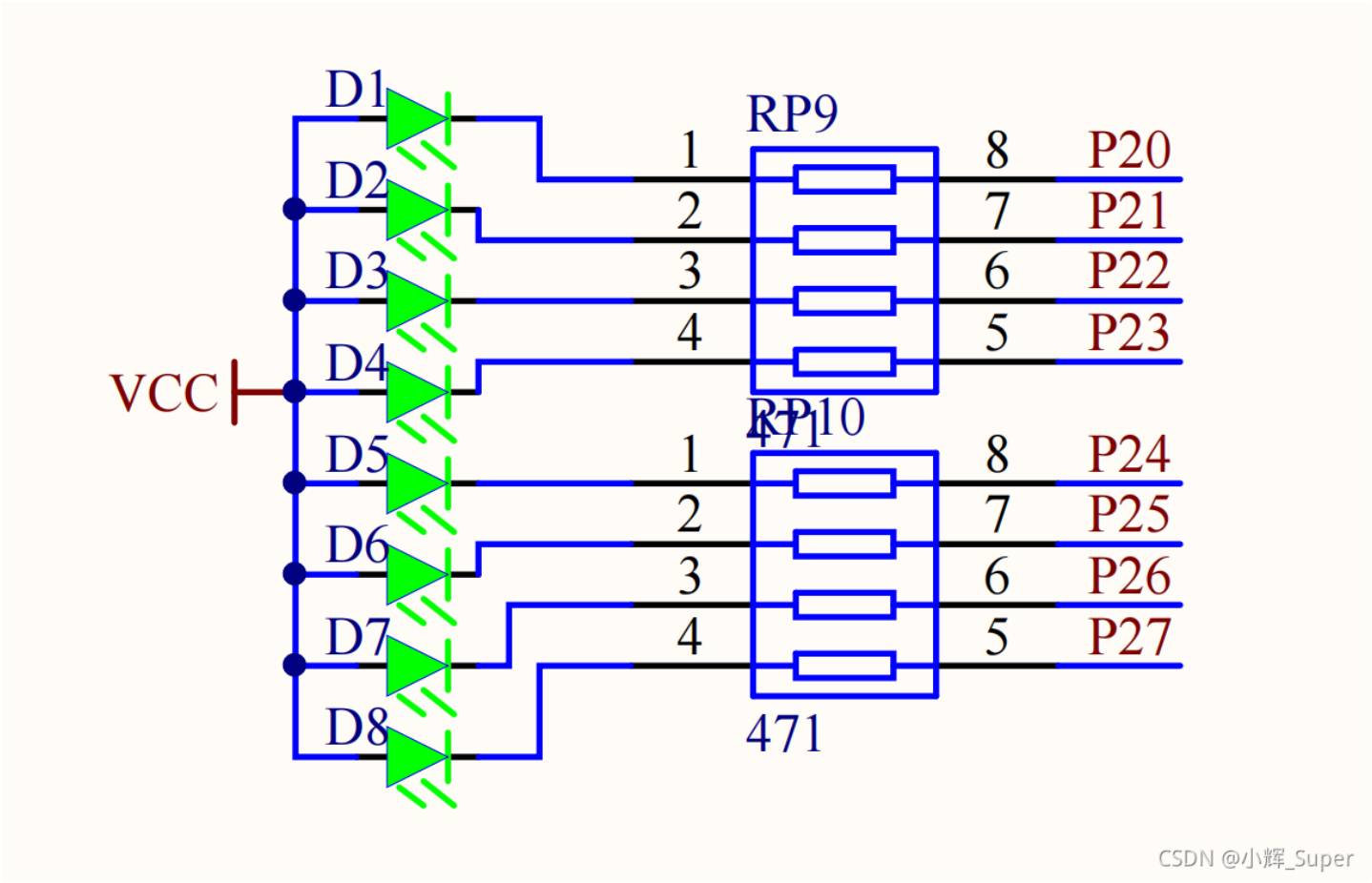

实验2：LED闪烁

第二个实验给在灯亮和灯灭前后加了延时，实现了LED闪烁的功能，延时函数直接通过while实现，这么说51运行一行代码要10us，开发板上贴了一个10.0592M的晶振，不应该这么慢吧。

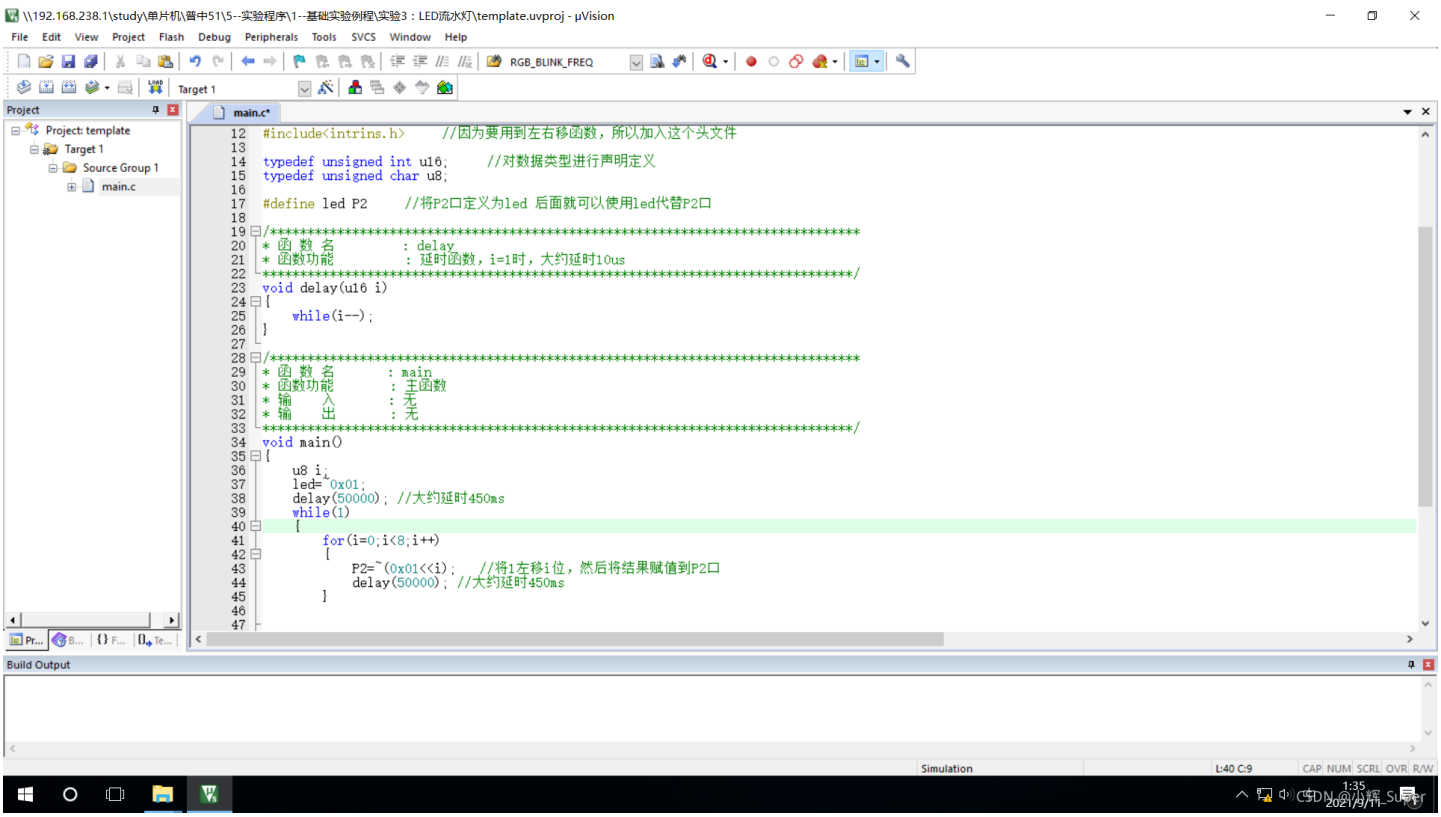


实验3：LED流水灯

下图为普中51-A2开发板LED部分的原理图，有8个LED，对应单片机IO引脚P2.0-P2.7：

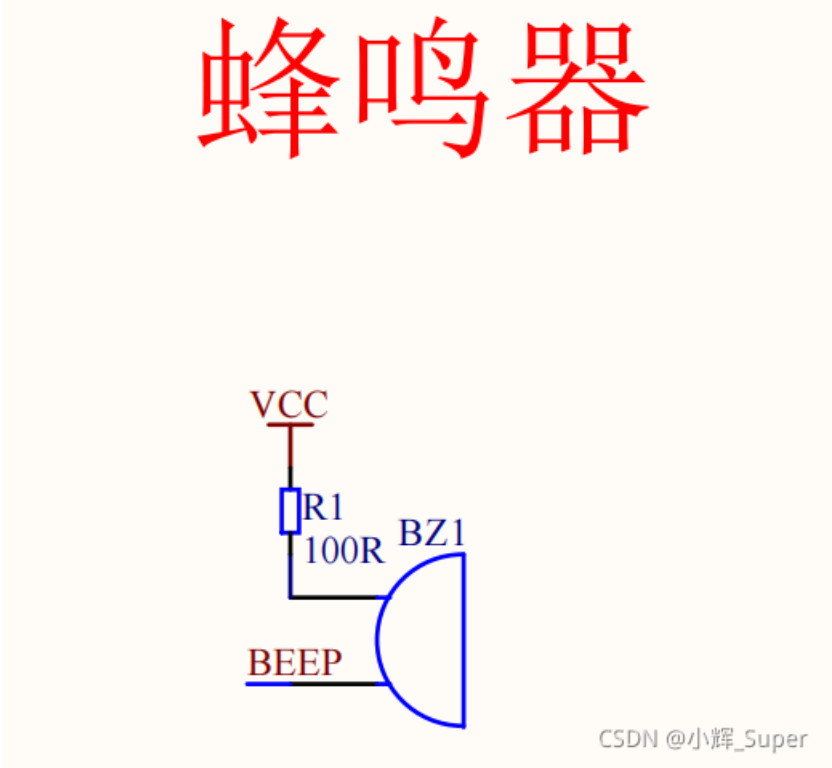


代码上，通过循环的移位运算，使P2的8个脚每次只有一个被拉低，形成流水灯效果。

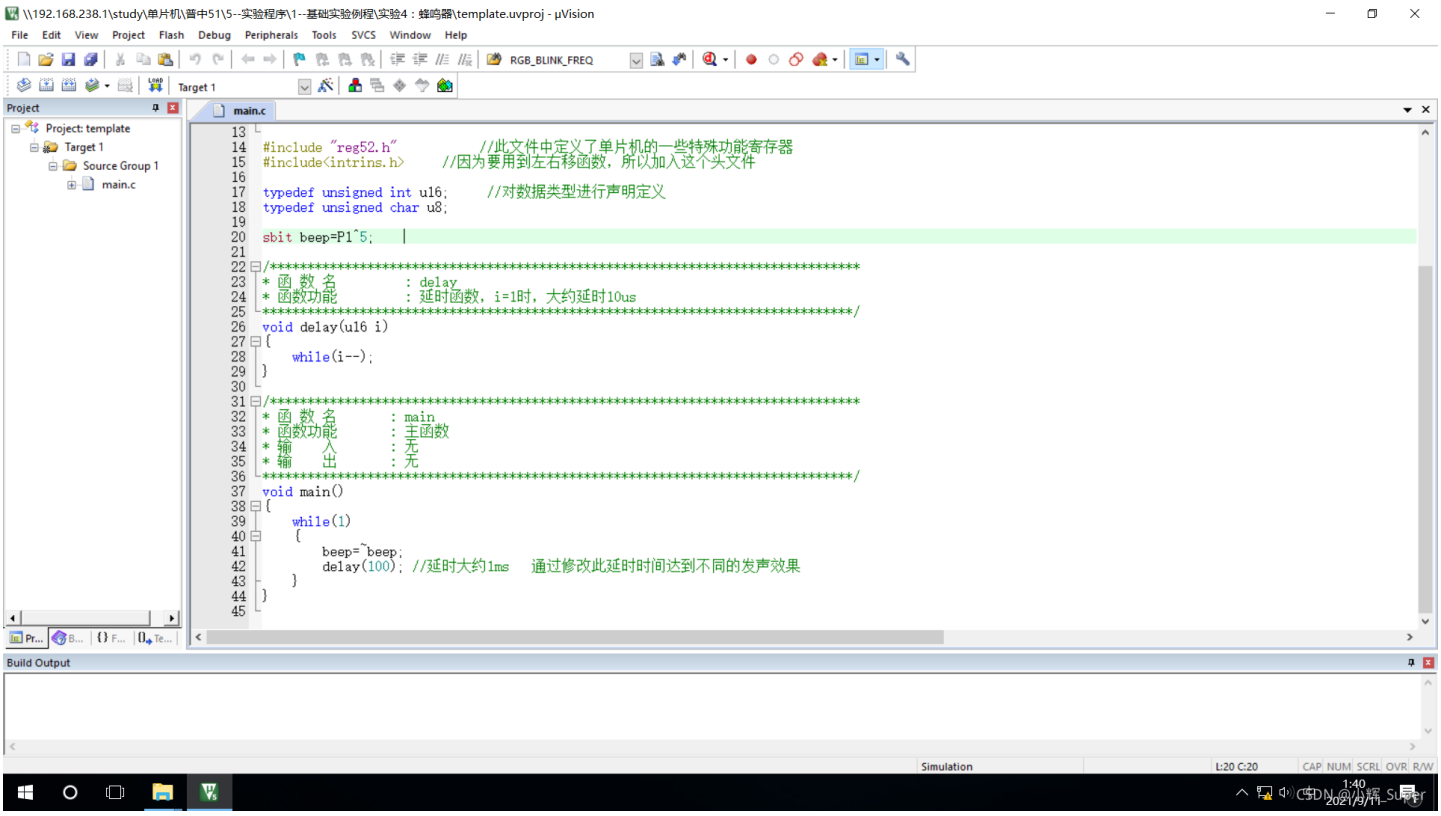


实验4：蜂鸣器

开发板上的蜂鸣器是集成的，放大电路封装在模块内部，这里可以把它当做一个LED，但是如果它是有源的，单片机给电平蜂鸣器就会工作，不幸的是，开发板上使用的是无源蜂鸣器，它需要一定频率的脉冲（高低电平）才会发声，好处是可以模拟曲调形成音乐效果。

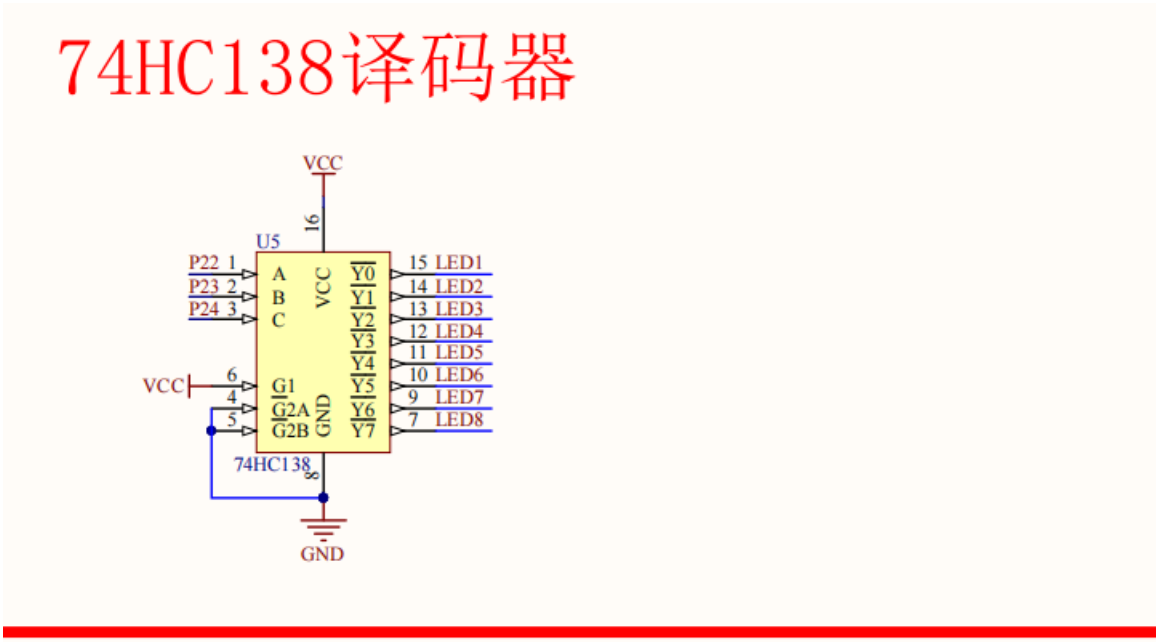


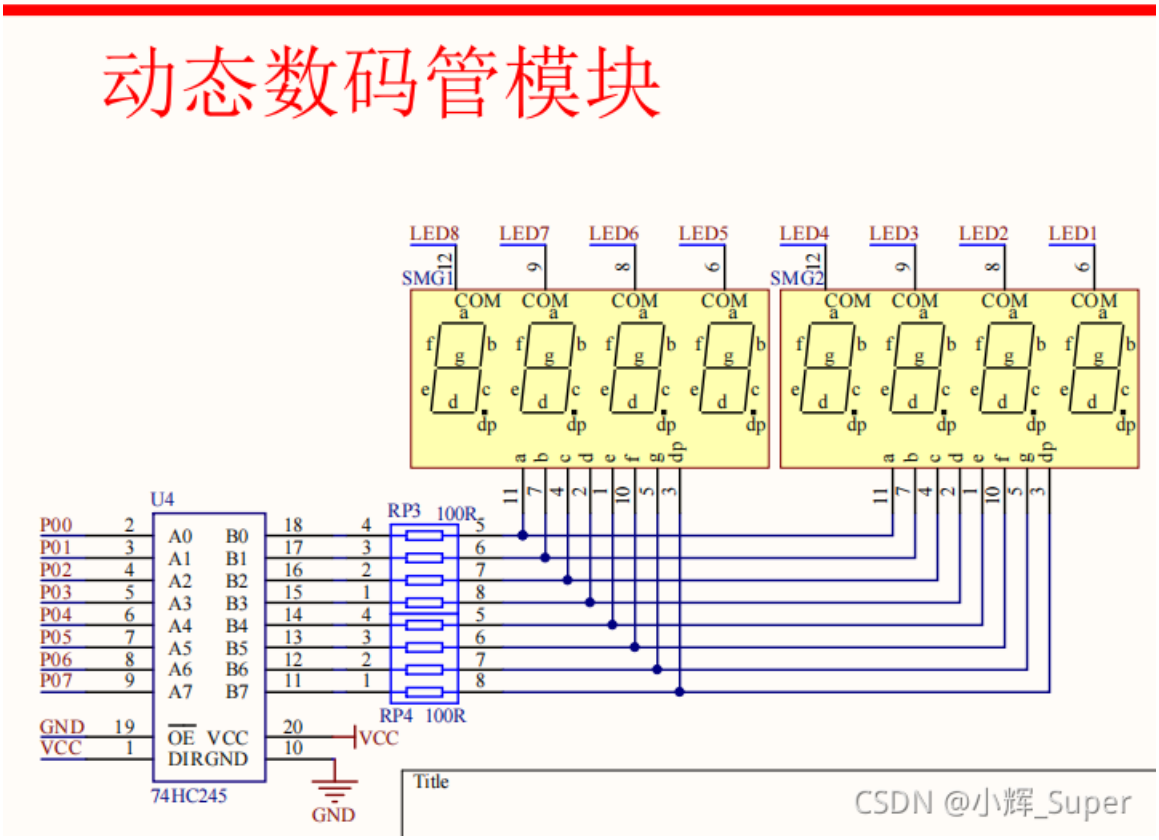
代码实现上，给了蜂鸣器1ms的周期，如果修改周期（即频率），蜂鸣器的声音也不同。



实验5：动态数码管显示

由于51单片机IO资源有限，不能通过IO直接控制所有数码管，需要用到一些芯片。
74HC138是3入8出译码器，它可以通过单片机的3个IO的状态（000-111）选择8个数码管中的1个进行操作（片选）。
74HC245是一种三态输出（A->B、B->A、高阻），八路信号收发器，这里用到了输入功能，将单片机P0.0-P0.7输入到片选选中的数码管中。





实验示例的代码中出现了code关键字，下面列举了C51中一些关键字及解释：
code ：程序存储区（64KB），
data ：可直接寻址的内部数据存储区（128B）
idata：不可直接寻址的内部数据存储区（256B）
bdata：可位寻址内部数据存储区（16B）
xdata：外部数据存储区（64KB）
pdata：分页的外部数据存储区

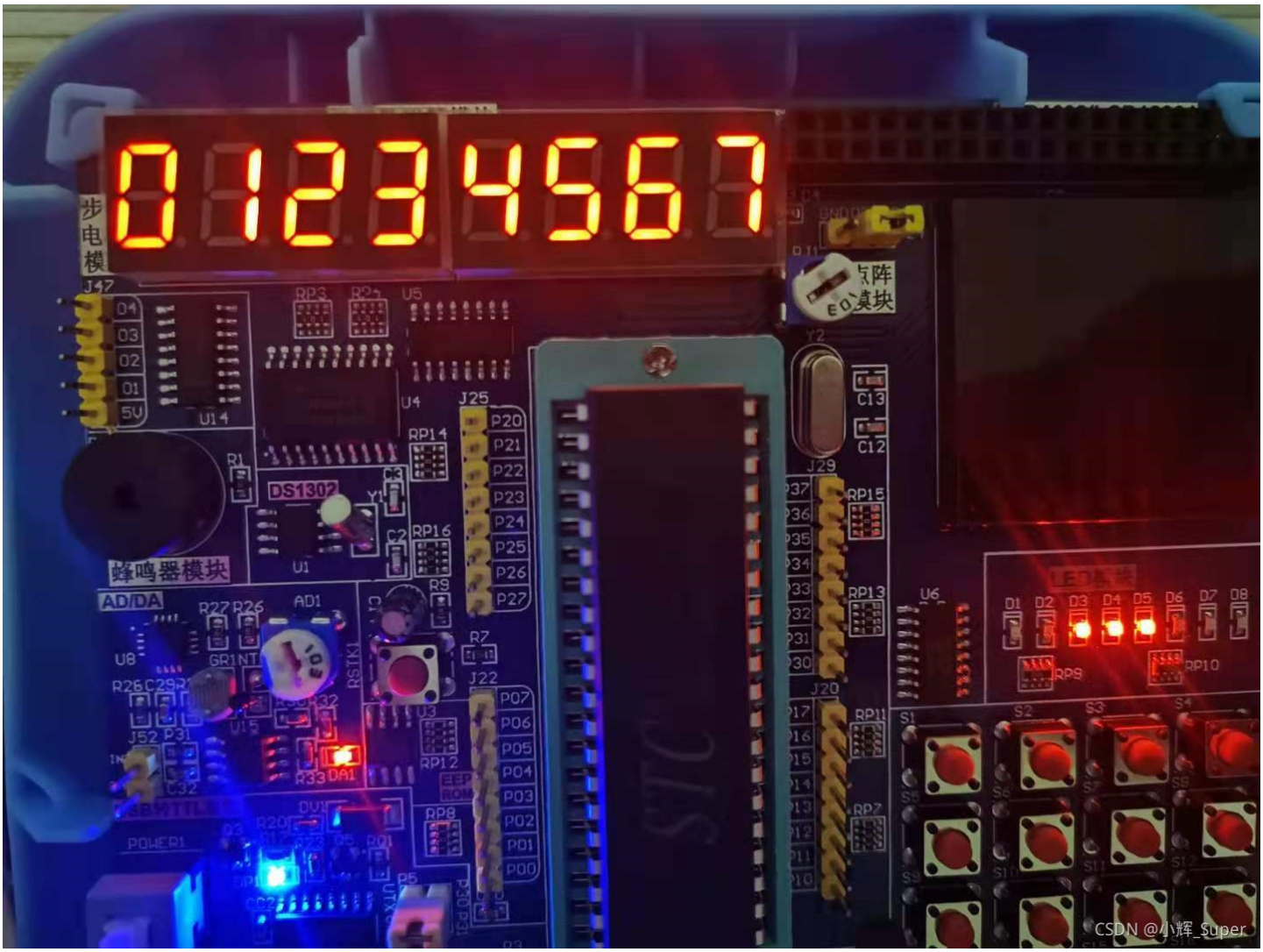
code区在运行的时候是不可以更改的，data区放全局变量和临时变量，是要不断的改变的，所以code类似于定义常量。

代码中smgduan数组存放了16个8进制数据，分别对应数码管的0-F这16种显示，for循环中i决定了片选和数码管显示值，即i=0时，选中第一个数码管，并显示0；i=7时，选中最后一个数码管，显示7。

```
/*********************************************************************************
实验现象：下载程序后"动态数码管模块"从左至右显示0-7
接线说明：  单片机-->动态数码管模块(具体接线图可见开发攻略对应实验的“实验现象”章节)

注意事项：
*********************************************************************************/

#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;

/* 3位二进制编码，用于选择0-7号数码管 */
sbit LSA=P2^2;
sbit LSB=P2^3;
sbit LSC=P2^4;

/* 数码管上的数值0-F */
u8 code smgduan[17]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,
     0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};//显示0~F的值

/*********************************************************************************
* 函 数 名         : delay
* 函数功能     : 延时函数，i=1时，大约延时10us
*********************************************************************************/
void delay(u16 i)
{
 while(i--);
}

/*********************************************************************************
* 函 数 名         : DigDisplay
* 函数功能     : 数码管动态扫描函数，循环扫描8个数码管显示
*********************************************************************************/
void DigDisplay()
{
 u8 i;
 for(i=0;i<8;i++)
 {
  switch(i)  //位选，选择点亮的数码管，
  {
     case(0):
    LSA=1;LSB=1;LSC=1; break;//显示第0位
   case(1):
    LSA=0;LSB=1;LSC=1; break;//显示第1位
   case(2):
    LSA=1;LSB=0;LSC=1; break;//显示第2位
   case(3):
    LSA=0;LSB=0;LSC=1; break;//显示第3位
   case(4):
    LSA=1;LSB=1;LSC=0; break;//显示第4位
   case(5):
    LSA=0;LSB=1;LSC=0; break;//显示第5位
   case(6):
    LSA=1;LSB=0;LSC=0; break;//显示第6位
   case(7):
```

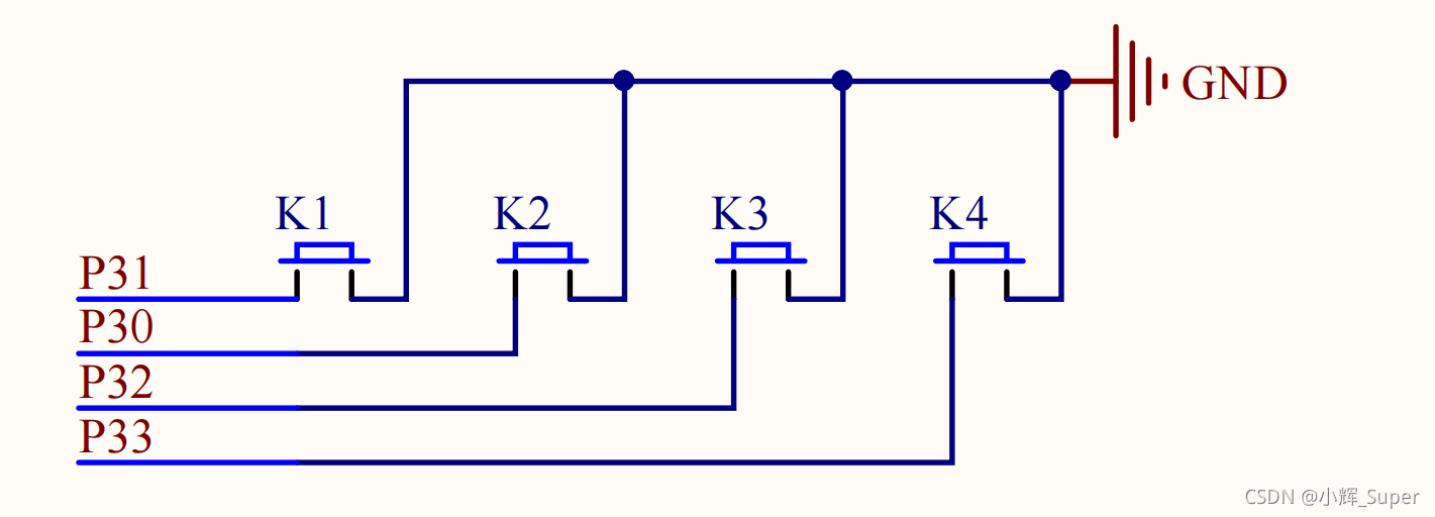
```
    LSA=0;LSB=0;LSC=0; break;//显示第7位
  }
  P0=smgduan[i];//发送段码
  delay(100); //间隔一段时间扫描
  P0=0x00;//消隐
 }
}

/*******************************************************************************
* 函 数 名       : main
* 函数功能   : 主函数
* 输    入       : 无
* 输    出       : 无
*******************************************************************************/
void main()
{
 while(1)
 {
  DigDisplay();  //数码管显示函数
 }
}
```

运行结果：



实验6：独立按键

独立按键即平时我们所说的轻触按键，区别于开发板上的矩阵按键。由原理图得知，该开发板检测按键的方法是判断单片机IO脚是否被拉低。



该代码的核心部分为keypros函数，作用是检测按键K1，如果检测到按键按下（P3.1为低电平）则延时10ms进行消抖，消抖后再次判断是否按下，如果是，则执行按键操作，即切换D1灯的状态。

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

*
实验现象：下载程序后按下K1按键可以对D1小灯状态取反
接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）

注意事项：


*********************************************************************************

*/

#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;

sbit k1=P3^1;  //定义P31口是k1
sbit led=P2^0;  //定义P20口是led

/*********************************************************************************
* 函 数 名         : delay
* 函数功能     : 延时函数，i=1时，大约延时10us
*********************************************************************************/
void delay(u16 i)
{
 while(i--);
}

/*********************************************************************************
* 函 数 名         : keypros
* 函数功能     : 按键处理函数，判断按键K1是否按下
*********************************************************************************/
void keypros()
{
```

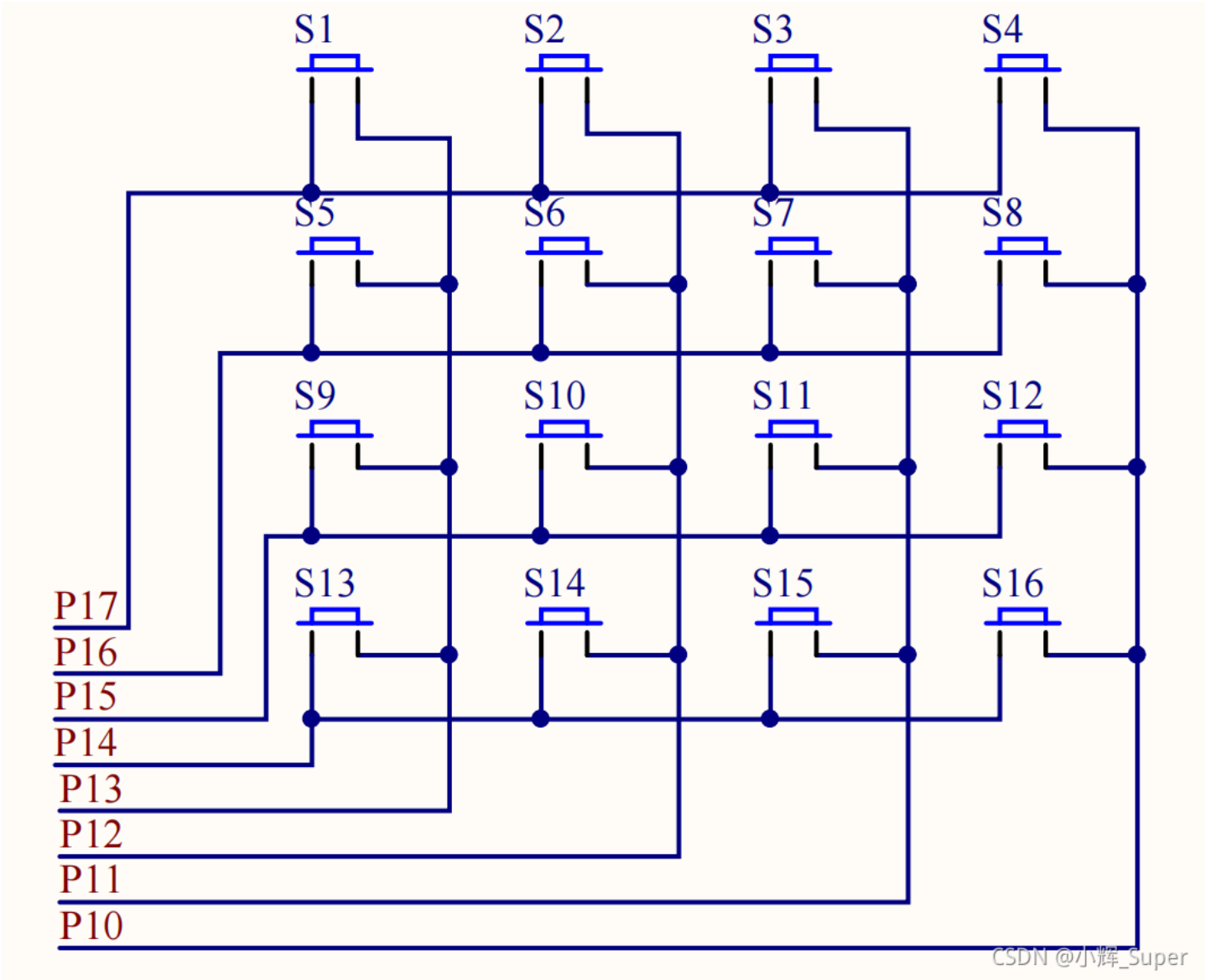
```
 if(k1==0)    //检测按键K1是否按下
 {
  delay(1000);   //消除抖动 一般大约10ms
  if(k1==0)  //再次判断按键是否按下
  {
   led=~led;   //led状态取反
  }
  while(!k1);  //检测按键是否松开
 }
}

/*******************************************************************************
* 函 数 名       : main
* 函数功能    : 主函数
* 输    入       : 无
* 输    出       : 无
*******************************************************************************/
void main()
{
 led=1;
 while(1)
 {
  keypros();  //按键处理函数
 }
}
```

实验7：矩阵按键

4*4矩阵按键可用8个IO控制16个按键，开发板中单片机的P1.7控制矩阵按键的第一行，P1.3控制矩阵按键的第一列。当按键按下时，其对应的行列IO就被导通。
矩阵按键检测方法有多种，最常用的是行列扫描法和线翻转法：

1. 行列扫描法，按键检测时，轮流设置单独一列为低电平，其他列为高电平，若检测到某一行也为低电平，则我们便可确认当前被按下的键是哪一行哪一列的（最多判断4*4=16次）。 当然我们也可以将行线置低电平，扫描列是否有低电平。
2. 线翻转法，就是将所有行线设置为低电平时，检测所有列线是否有低电平，如果有，就得出按键的列线值；然后再翻转，所有列线设为低电平，检测所有行线的值， 如果有按键按下，行线的电平就会拉低，就得出了行线的值。

本实验例程中使用的检测方法为线翻转法。



代码实现上，程序通过读取4*4矩阵按键控制晶体管（没有片选，默认第一个）的显示值，细心的人会发现这里的smgduan和实验5中的数码管显示数组不一样，其实只是进行了取反，从该数组的值来看，该数码管在电路中是共阴极的。

代码的核心功能全在KeyDown函数中，该函数先将所有行电平置低，列电平置高，如果有按键按下，列电平就会发生变化，然后进行10ms消抖；接下来通过P1的低4位（P1.0-P1.3）判断哪一列为低电平，得出按键的列数；再将所有行的电平置高，列电平置低，检测P1的高四位（P1.4-P1.7）判断哪一列为低电平，得出按键的行数。按键的编号为行数+列数x4（第一位为0）。

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

*
实验现象：下载程序后数码管显示0，按下矩阵按键上的按键显示对应的数字

注意事项：


*********************************************************************************

*/
#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;
```

```c
#define GPIO_DIG P0    //数码管P0.0-P0.7
#define GPIO_KEY P1    //矩阵按键8个IO管脚为P1.0-P1.7


u8 KeyValue; //用来存放读取到的键值

//数码管显示值，需要翻转
u8 code smgduan[17]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,
     0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e};//显示0~F的值

/*******************************************************************************
* 函 数 名         : delay
* 函数功能     : 延时函数，i=1时，大约延时10us
*******************************************************************************/
void delay(u16 i)
{
 while(i--);
}

/*******************************************************************************
* 函 数 名         : KeyDown
* 函数功能     : 检测有按键按下并读取键值
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/
void KeyDown(void)
{
 char a=0;
 GPIO_KEY=0x0f;
 if(GPIO_KEY!=0x0f)//读取按键是否按下
 {
  delay(1000);//延时10ms进行消抖
  if(GPIO_KEY!=0x0f)//再次检测键盘是否按下
  {
   //测试列
   GPIO_KEY=0X0F;
   switch(GPIO_KEY)
   {
    case(0X07): KeyValue=0;break;
    case(0X0b): KeyValue=1;break;
    case(0X0d): KeyValue=2;break;
    case(0X0e): KeyValue=3;break;
   }
   //测试行
   GPIO_KEY=0XF0;
   switch(GPIO_KEY)
   {
    case(0X70): KeyValue=KeyValue;break;
    case(0Xb0): KeyValue=KeyValue+4;break;
    case(0Xd0): KeyValue=KeyValue+8;break;
    case(0Xe0): KeyValue=KeyValue+12;break;
   }

  }
 }
 while((a<50)&&(GPIO_KEY!=0xf0))  //检测按键松手检测
 {
  delay(100);
  a++;
```

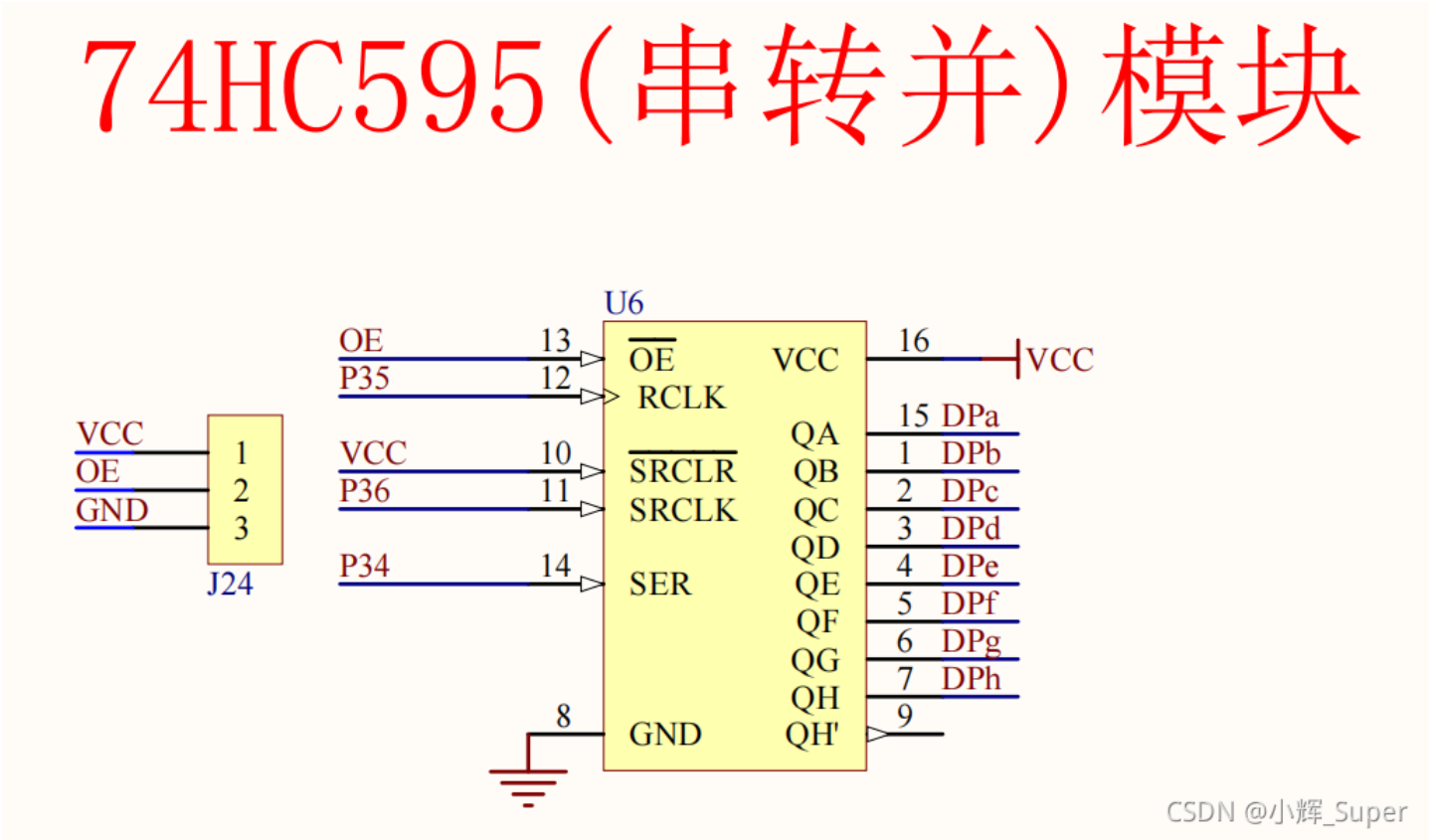
```
 }
}


/*******************************************************************************
* 函 数 名         : main
* 函数功能   : 主函数
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/
void main()
{

 while(1)
 {
  KeyDown();      //按键判断函数
  GPIO_DIG=~smgduan[KeyValue];   //
 }
}
```

实验8：单片机IO扩展–74HC595

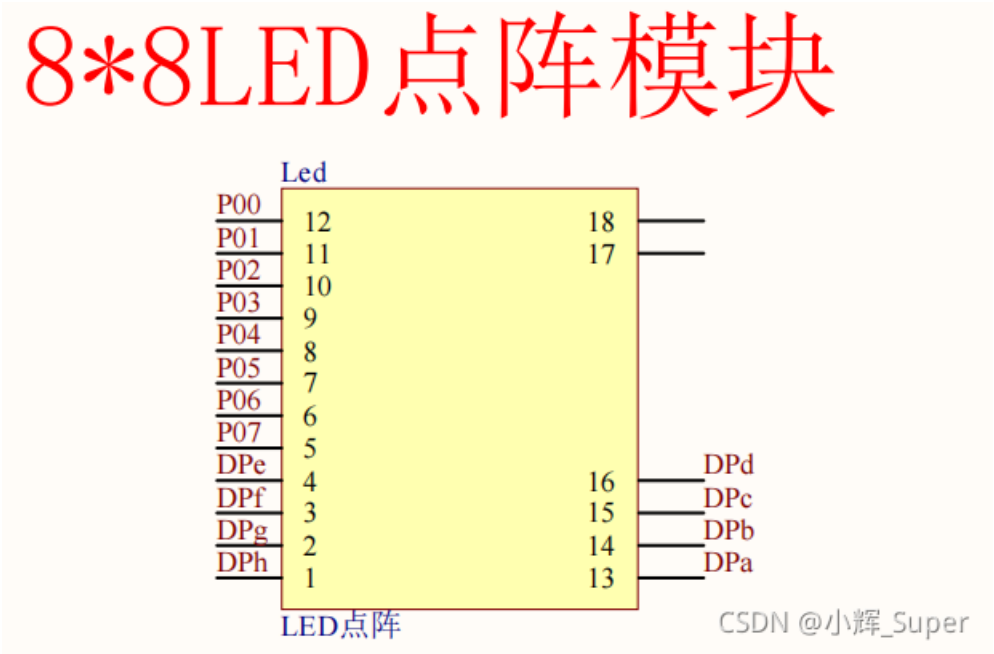
74HC595 是一个 8 位串行输入、并行输出的位移缓存器，其中并行输出为三态输出（即高电平、低电平和高阻抗）。

15和1-7 脚 DPa–DPh：并行数据输出
9 脚 DPh 非：串行数据输出
10 脚 SRCLR 非（ MR）： 低电平复位引脚
11 脚 SRCLK（ SHCP）： 移位寄存器时钟输入
12 脚 RLCK（ STCP）： 存储寄存器时钟输入
13 脚 OE 非（ OE）： 输出有效 （使能）
14 脚 SER（ DS）： 串行数据输入

一开始程序烧录没有反应，后来发现J24上的跳帽默认接到VCC上，即没有给74HC595使能。



该开发板的扩展IO接到了8*8LED点阵模块上（点阵模块是下个实验）：



例程代码的核心为Hc595SendByte函数，作用是向74HC595发送1个字节的数据，for循环中，串行数据输入脚（SER）随着移位寄存器时钟（SRCLK）读取数据的0-7位，SER是读取是高位在前的；每个时钟周期间加了两个_nop_()来延时，*nop*();指令需要耗费1个机械周期也就是12个时钟周期，期间不做任何事，是单片机最小的延时；在数据传输完成后，还需要用存储寄存器时钟（RCLK）来进行存储转换。

main函数中使用到了_crol_()，作用是循环左移，例程用它来对0xFE进行循环左移，显示在点阵上效果就是7个灯亮，1个灯灭，且状态会循环左移，形成类似流水灯效果。

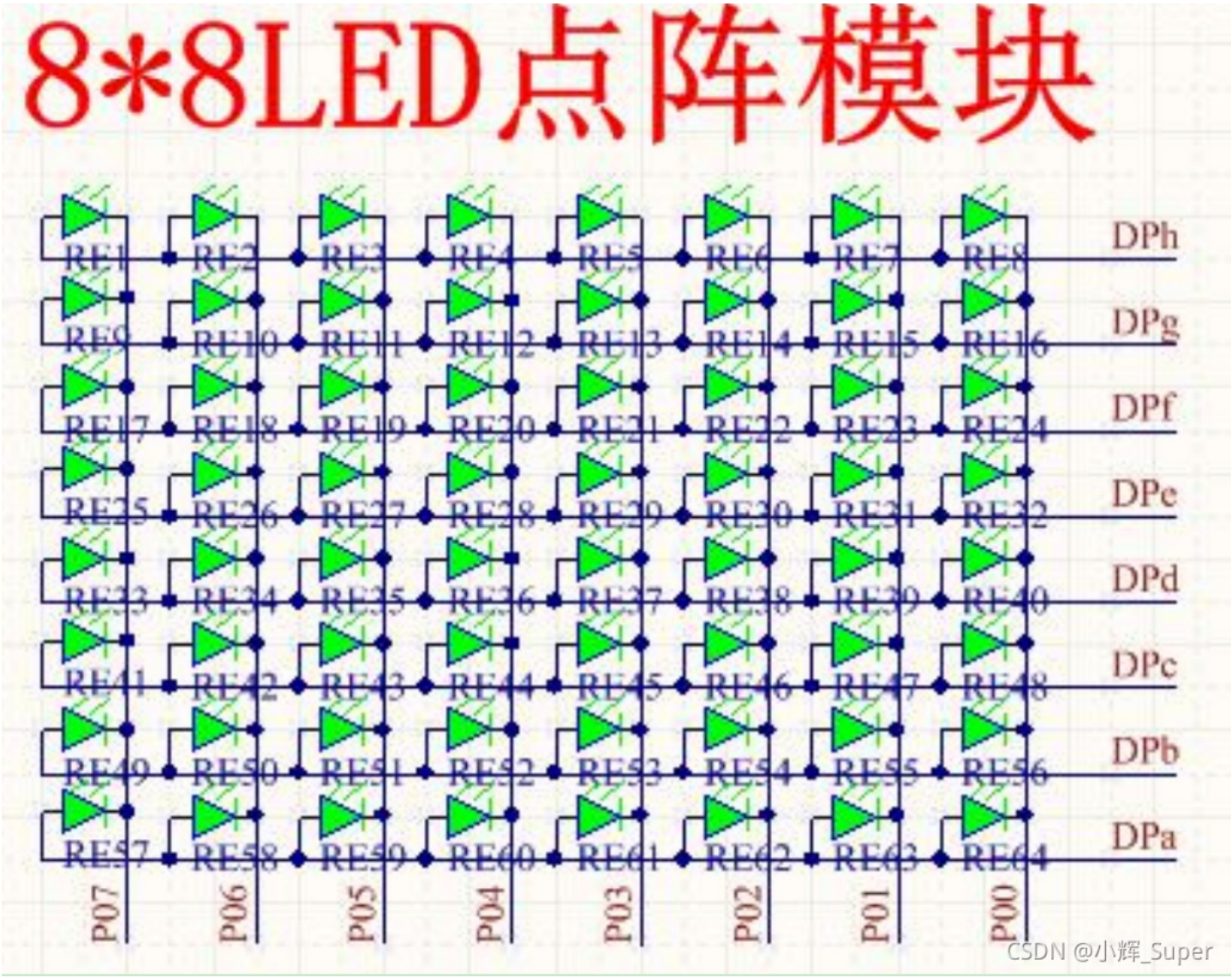
```
/*******************************************************************************
```

```
*
实验现象：下载程序后，LED灯从左往右点亮，类似流水灯效果

接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）

注意事项：


*******************************************************************************

*/

#include "reg51.h"    //此文件中定义了单片机的一些特殊功能寄存器
#include "intrins.h"

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;
u8 ledNum;

//--定义使用的IO口--//
sbit SRCLK=P3^6;
sbit RCLK=P3^5;
sbit SER=P3^4;
sbit LED=P0^7; //控制第一列点阵


/*******************************************************************************
* 函 数 名         : delay
* 函数功能     : 延时函数，i=1时，大约延时10us
*******************************************************************************/
void delay(u16 i)
{
 while(i--);
}

/*******************************************************************************
* 函 数 名         : Hc595SendByte(u8 dat)
* 函数功能     : 向74H595发送一个字节的数据
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/
void Hc595SendByte(u8 dat)
{
 u8 a;

 SRCLK = 1;
 RCLK = 1;

 for(a=0;a<8;a++)   //发送8位数
 {
  SER = dat >> 7;   //从最高位开始发送
  dat <<= 1;

  SRCLK = 0;    //发送时序
  _nop_();
  _nop_();
  SRCLK = 1;
 }
 RCLK = 0;
```

```
 RCLK = 0;
 _nop_();
 _nop_();
 RCLK = 1;
}


/*******************************************************************************
* 函 数 名       : main
* 函数功能   : 主函数
* 输    入       : 无
* 输    出       : 无
*******************************************************************************/
void main()
{
 LED=0;   //使能第一列点阵
 ledNum = ~0x01;

 while(1)
 {
  Hc595SendByte(ledNum);
  ledNum = _crol_(ledNum, 1);
  delay(50000);
 }
}
```

实验9：LED点阵（点亮一个点）

开发板上点阵由64个LED构成，P0控制着LED的负极，DP（扩展IO）控制着正极，所以要点亮LED，必须拉低其对应的P0，同时对应的DP也需要给高电平。



这次的代码和上一实验区别不大，只是Hc595SendByte的形参变多了，可以一次传输两个字节数据，在数据传输完成后，只需要用一个存储寄存器时钟（RCLK）周期来进行存储转换。
实验的结果应该是一个灯亮，但我的结果是亮6个灯，难道硬件有问题？

```
/*********************************************************************************
实验现象：下载程序后，LED点阵左上角第一个点的LED被点亮果

接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）

注意事项：


*********************************************************************************/

#include "reg51.h"     //此文件中定义了单片机的一些特殊功能寄存器
#include "intrins.h"

typedef unsigned int u16;    //对数据类型进行声明定义
typedef unsigned char u8;

//--定义使用的IO口--//
sbit SRCLK=P3^6;
```

```
sbit RCLK=P3^5;
sbit SER=P3^4;
sbit LED=P0^7;

/*********************************************************************************
* 函 数 名         : Hc595SendByte(u8 dat1,u8 dat2)
* 函数功能     : 通过595发送2个字节的数据
* 输    入         : dat1：第2个595输出数值
*                    * dat2: 第1个595输出数值
* 输    出         : 无
*********************************************************************************/
void Hc595SendByte(u8 dat1,u8 dat2)
{
 u8 a;

 SRCLK = 1;
 RCLK = 1;

 for(a=0;a<8;a++)   //发送8位数
 {
  SER = dat1 >> 7;   //从最高位开始发送
  dat1 <<= 1;

  SRCLK = 0;    //发送时序
  _nop_();
  _nop_();
  SRCLK = 1;
 }

 for(a=0;a<8;a++)   //发送8位数
 {
  SER = dat2 >> 7;   //从最高位开始发送
  dat2 <<= 1;

  SRCLK = 0;    //发送时序
  _nop_();
  _nop_();
  SRCLK = 1;
 }

 RCLK = 0;
 _nop_();
 _nop_();
 RCLK = 1;
}



/*********************************************************************************
* 函 数 名       : main
* 函数功能   : 主函数
* 输    入       : 无
* 输    出      : 无
*********************************************************************************/
void main()
{ LED=0;  //使第一列为低电平。

 while(1)
 {
    Hc595SendByte(0xfe,0x01);
 }
```

```
}
```

```
/*******************************************************************************
*                8*8LED点阵——显示数字实验                *
实现现象：下载程序后点阵上显示数字0

注意事项：一定要将J24短接片短接到GND端。
*******************************************************************************/

#include "reg51.h"    //此文件中定义了单片机的一些特殊功能寄存器
#include<intrins.h>

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;

sbit SRCLK=P3^6;
sbit RCLK=P3^5;
sbit SER=P3^4;

u8 ledduan[]={0x00,0x00,0x3e,0x41,0x41,0x41,0x3e,0x00};
u8 ledwei[]={0x7f,0xbf,0xdf,0xef,0xf7,0xfb,0xfd,0xfe};
/*******************************************************************************
* 函 数 名         : delay
* 函数功能     : 延时函数，i=1时，大约延时10us
*******************************************************************************/
void delay(u16 i)
{
 while(i--);
}

/*******************************************************************************
* 函数名         : Hc595SendByte(u8 dat)
* 函数功能     : 向74HC595发送一个字节的数据
* 输入           : 无
* 输出           : 无
*******************************************************************************/
void Hc595SendByte(u8 dat)
{
 u8 a;
 SRCLK=0;
 RCLK=0;
 for(a=0;a<8;a++)
 {
  SER=dat>>7;
  dat<<=1;

  SRCLK=1;
  _nop_();
  _nop_();
  SRCLK=0;
 }

 RCLK=1;
 _nop_();
 _nop_();
 RCLK=0;
}

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

```
* 函 数 名         : main
* 函数功能   : 主函数
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/
void main()
{
 u8 i;
 while(1)
 {
  P0=0x7f;
  for(i=0;i<8;i++)
  {
   P0=ledwei[i];    //位选
   Hc595SendByte(ledduan[i]); //发送段选数据
   delay(100);     //延时
   Hc595SendByte(0x00);  //消隐
  }
 }
}
```

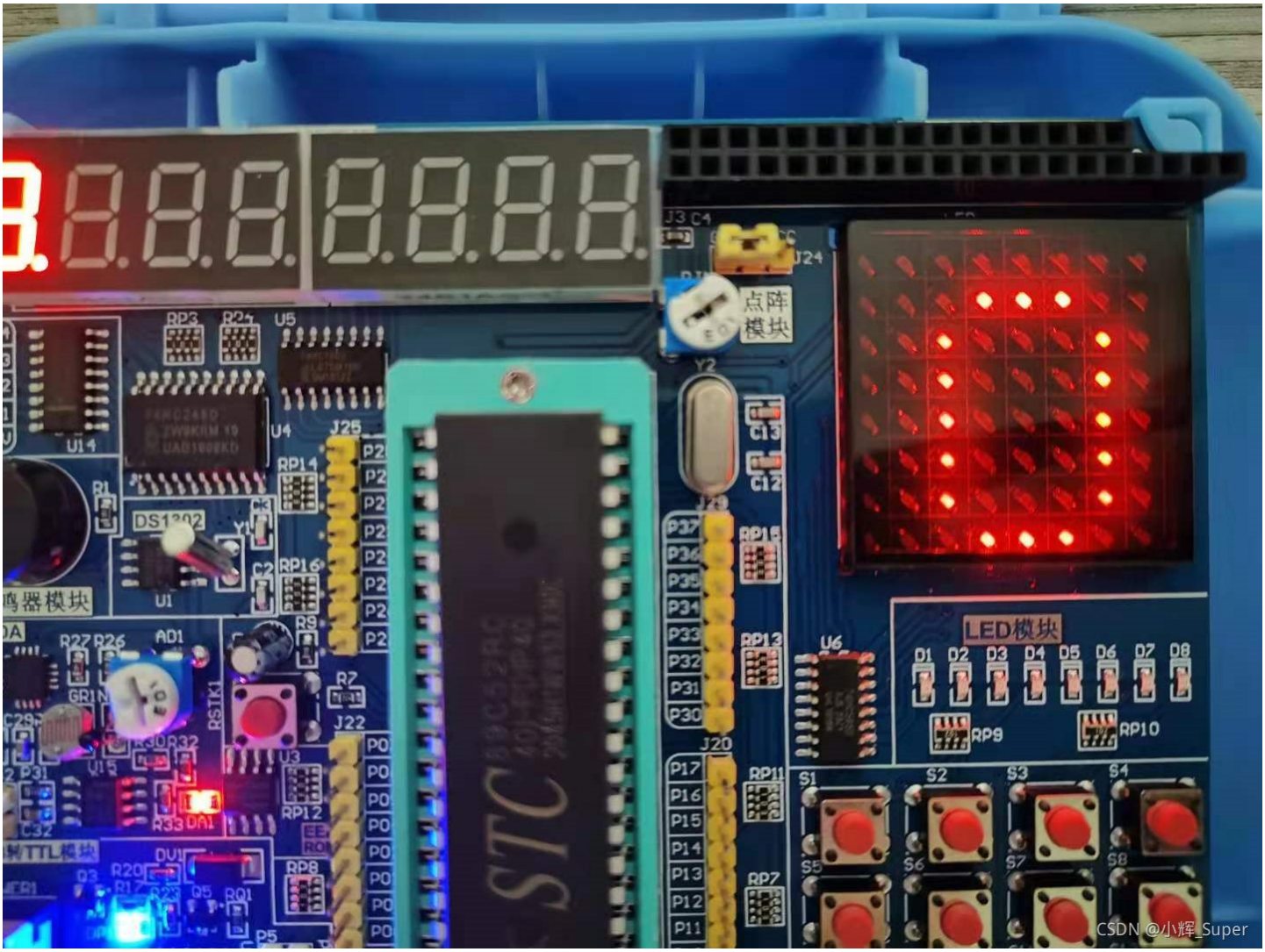
实验10：LED点阵(显示数字)

既然可以点亮特定的一个点，那就可以通过点亮多个不同的点在点阵上显示各种有趣的图案，甚至广告牌效果，当然，这里我只想先过一遍例程。

代码实现上，和上两个实验类似，只是这里多了两个数组ledduan和ledwei，位选和段选，说白了就是列和行，其实大多情况我们都是用一个二维数组存储这列点阵数据，代码中实现了数字0的点阵LED显示。

```
/*******************************************************************************
*                 8*8LED点阵——显示数字实验                    *
实现现象：下载程序后点阵上显示数字0

注意事项：一定要将J24短接片短接到GND端。
*******************************************************************************/

#include "reg51.h"    //此文件中定义了单片机的一些特殊功能寄存器
#include<intrins.h>

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;

sbit SRCLK=P3^6;
sbit RCLK=P3^5;
sbit SER=P3^4;

u8 ledduan[]={0x00,0x00,0x3e,0x41,0x41,0x41,0x3e,0x00};
u8 ledwei[]={0x7f,0xbf,0xdf,0xef,0xf7,0xfb,0xfd,0xfe};
/*******************************************************************************
* 函 数 名         : delay
* 函数功能     : 延时函数，i=1时，大约延时10us
*******************************************************************************/
void delay(u16 i)
{
 while(i--);
}

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

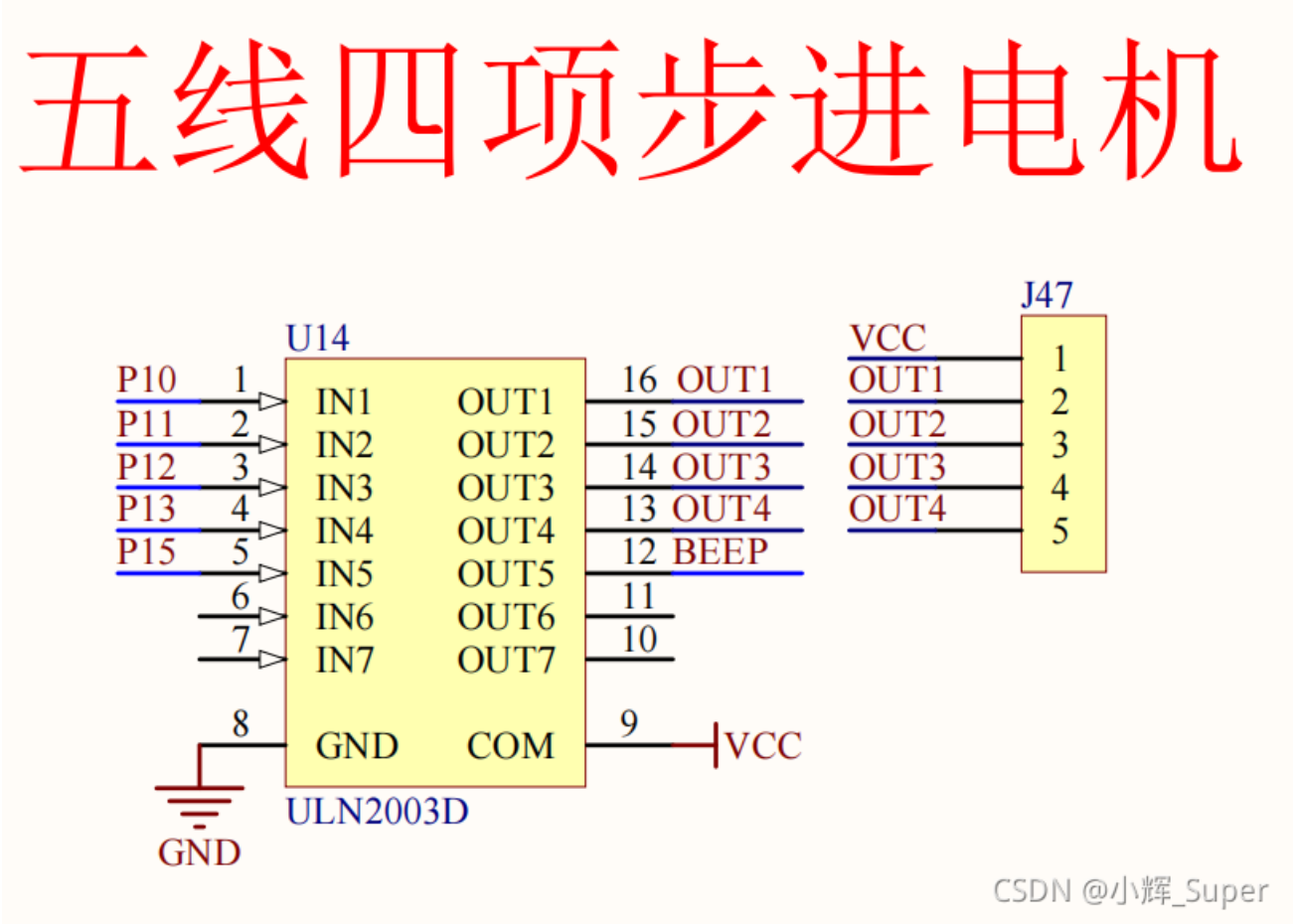
```
* 函数名         : Hc595SendByte(u8 dat)
* 函数功能      : 向74HC595发送一个字节的数据
* 输入           : 无
* 输出           : 无
*******************************************************************************/
void Hc595SendByte(u8 dat)
{
 u8 a;
 SRCLK=0;
 RCLK=0;
 for(a=0;a<8;a++)
 {
  SER=dat>>7;
  dat<<=1;

  SRCLK=1;
  _nop_();
  _nop_();
  SRCLK=0;
 }

 RCLK=1;
 _nop_();
 _nop_();
 RCLK=0;
}

/*******************************************************************************
* 函 数 名       : main
* 函数功能   : 主函数
* 输    入       : 无
* 输    出       : 无
*******************************************************************************/
void main()
{
 u8 i;
 while(1)
 {
  P0=0x7f;
  for(i=0;i<8;i++)
  {
   P0=ledwei[i];    //位选
   Hc595SendByte(ledduan[i]); //发送段选数据
   delay(100);     //延时
   Hc595SendByte(0x00);  //消隐
  }
 }
}
```



实验11：直流电机

51 单片机主要是用来控制而非驱动，如果直接使用芯片的 GPIO 管脚去驱动大功率器件，要么将芯片烧坏，要么就是驱动不起来。所以要驱动大功率器件，比如电机，就必须搭建外部驱动电路，这个开发板上的驱动芯片为ULN2003D，该芯片是一个 单片高电压、高电流的达林顿晶体管阵列集成电路。不仅可以用来驱动我们的直 流电机，还可用来驱动五线四相步进电机，比如 28BYJ-48 步进电机。开发板上的蜂鸣器也是由该芯片驱动。

原理图中，该芯片可以接7个输入，7个输出，实验中使用到了P1.0，所以电机的一极需要接到OUT1，另一端接VCC（直流电机正负极反接只影响转动方向）。



代码很简单，通过控制P1.0来控制驱动芯片，从而驱动电机转动，代码上和点亮LED没太大区别。

```
/*********************************************************************************
实验现象：下载程序后，直流电机旋转大约5S，然后停止

接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）
     1，单片机-->五线四相步进电机模块
       P10-->IN1
     2，五线四相步进电机模块输出-->直流电机
       5V-->直流电机两脚任意一个
    O1-->直流电机两脚任意一个

注意事项：


*********************************************************************************/

#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器
#include<intrins.h>  //因为要用到左右移函数，所以加入这个头文件

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;

sbit moto=P1^0;

/*********************************************************************************
* 函 数 名         : delay
* 函数功能     : 延时函数，i=1时，大约延时10us
*********************************************************************************/
void delay(u16 i)
{
 while(i--);
}

/*********************************************************************************
* 函 数 名       : main
* 函数功能   : 主函数
* 输    入       : 无
* 输    出       : 无
*********************************************************************************/
void main()
{
 u8 i;
 moto=0;   //关闭电机
 for(i=0;i<100;i++)   //循环100次，也就是大约5S
 {
  moto=1;    //开启电机
  delay(5000); //大约延时50ms
 }
 moto=0;   //关闭电机

 while(1)
 {

 }
}
```

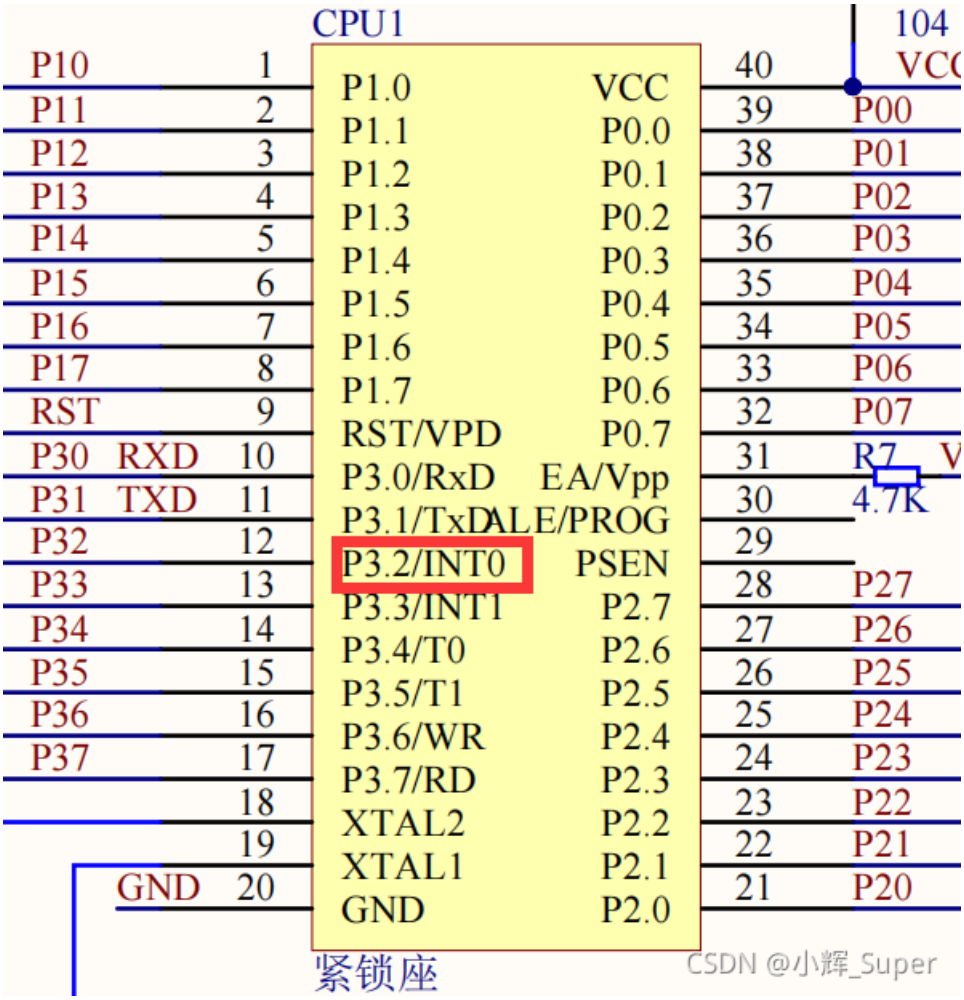
实验12：外部中断0

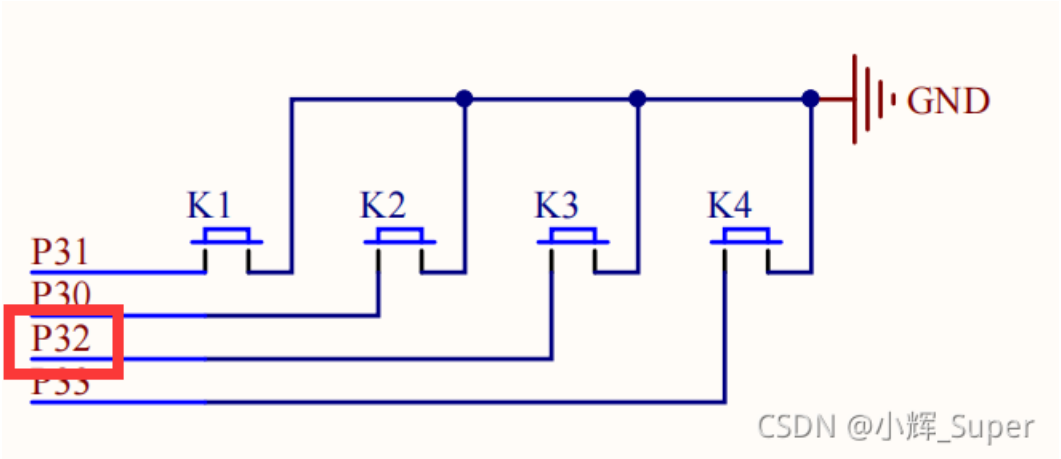
中断是单片机很重要的一个功能，它能使单片机活起来，让我们在操控单片机工作时随心所欲，无所不能。生活中，我们做事情不是按照一件做完再做下一件的死板模式进行的，而是依照事情的轻重缓急进行灵活调配，单片机也是如此，中断系统能让单片机处理临时突发事件，处理完后再回到离开的地方继续执行。

对于C51中断子函数（中断服务函数）的写法，有些不解，第一次看见关键字放在函数命名后的，这应该是Keil里C51的固定写法，interrupt 关键词后跟一个整数，表示中断号，取值范围0-31。中断号必须为常数，不允许使用操作符表达式。

STC89C5X 系列单片机提供了 4 个外部中断：外部中断 O(INTO)、外部中断 1(INT1)、外部中断 2(INT2)、外部中断 3(INT3)。

本实验使用的是外部中断0，与我平时用的ARM单片机不同的是，51的中断体现在硬件引脚上，不能自由分配，由51单片机原理图得知INT0对应的IO为P3.2，接到了开发板上的K3按键，所以只能用K3按键来触发中断。





Int0Init是中断初始化函数，该函数里IT0=1;表示设置下降沿触发，所以当按键K3按下时，对P3.2（INT0）产生下降沿信号，从而触发中断，此时程序会调用外部中断0的服务函数（用interrupt 0定义的函数），这个中断子函数名字可以自定义，在例程中，中断服务函数完成了消抖和切换LED状态的操作。

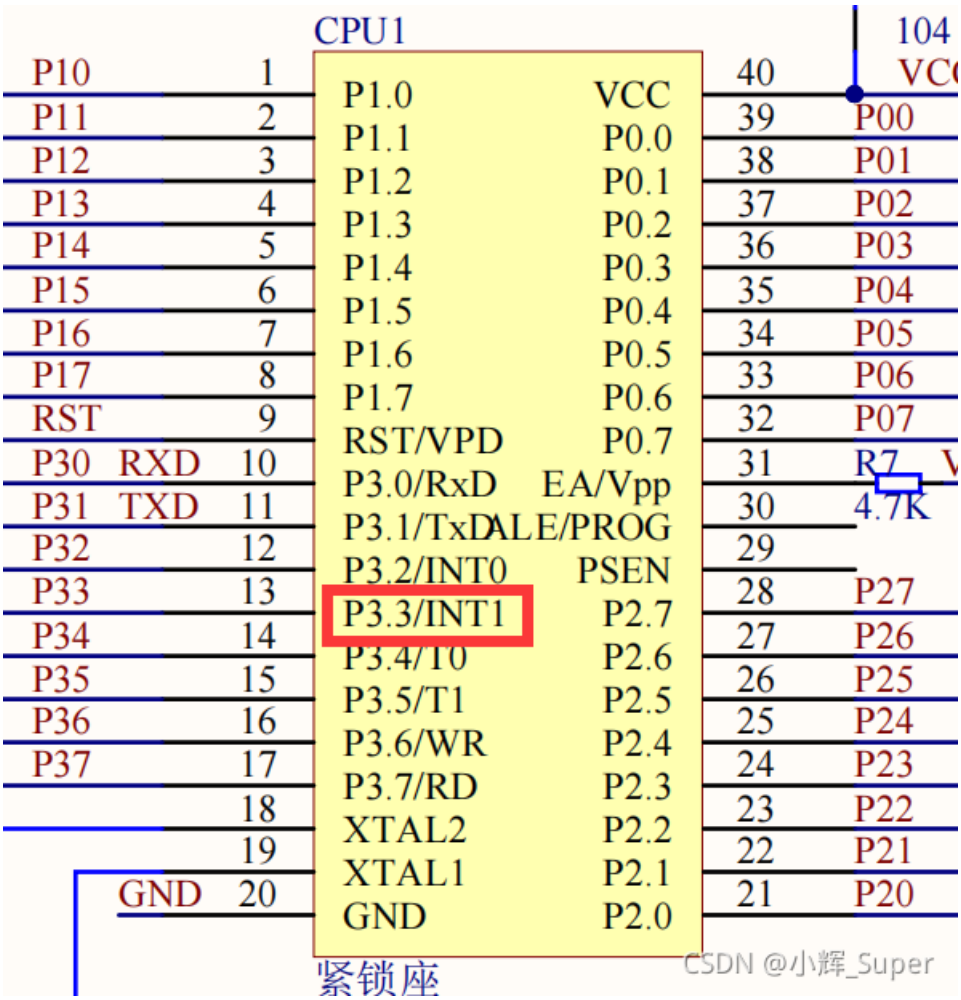
```
/*************************************************************
**********************
实验现象：下载程序后，操作 K3 按键使 D1 状态取反
接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）
1，单片机-->LED&交通灯模块
P20-->D1
2，单片机-->独立按键模块
```

```
P32-->K3
注意事项:
*******************************************************************
**********************/
#include "reg52.h" //此文件中定义了单片机的一些特殊功能寄存器
typedef unsigned int u16; //对数据类型进行声明定义
typedef unsigned char u8;
sbit k3=P3^2; //定义按键 K3
sbit led=P2^0; //定义 P20 口是 led
/*******************************************************************
***************
* 函 数 名 : delay
* 函数功能 : 延时函数，i=1 时，大约延时 10us
*******************************************************************
**************/
void delay(u16 i)
{
while(i--);
}
/*******************************************************************
***************
* 函 数 名 : Int1Init()
* 函数功能 : 设置外部中断 1
* 输 入 : 无
* 输 出 : 无
*******************************************************************
**************/
void Int0Init()
{
//设置 INT0
IT0=1;//跳变沿触发方式（下降沿）
EX0=1;//打开 INT0 的中断允许。
EA=1;//打开总中断
}
/*******************************************************************
***************
* 函 数 名 : main
* 函数功能 : 主函数
* 输 入 : 无
* 输 出 : 无
*******************************************************************
**************/
void main()
{
Int0Init(); // 设置外部中断 0
while(1);
}
/*******************************************************************
***************
* 函 数 名 : Int0() interrupt 0
* 函数功能 : 外部中断 0 的中断函数
* 输 入 : 无
* 输 出 : 无
*******************************************************************
**************/
void Int0() interrupt 0 //外部中断 0 的中断函数
{
delay(1000); //延时消抖
if(k3==0)
```

```
{
led=~led;
}
}
```

实验13：外部中断1

外部中断1和外部中断0的使用方法相同，开发板P3.3连接的是K4按键。



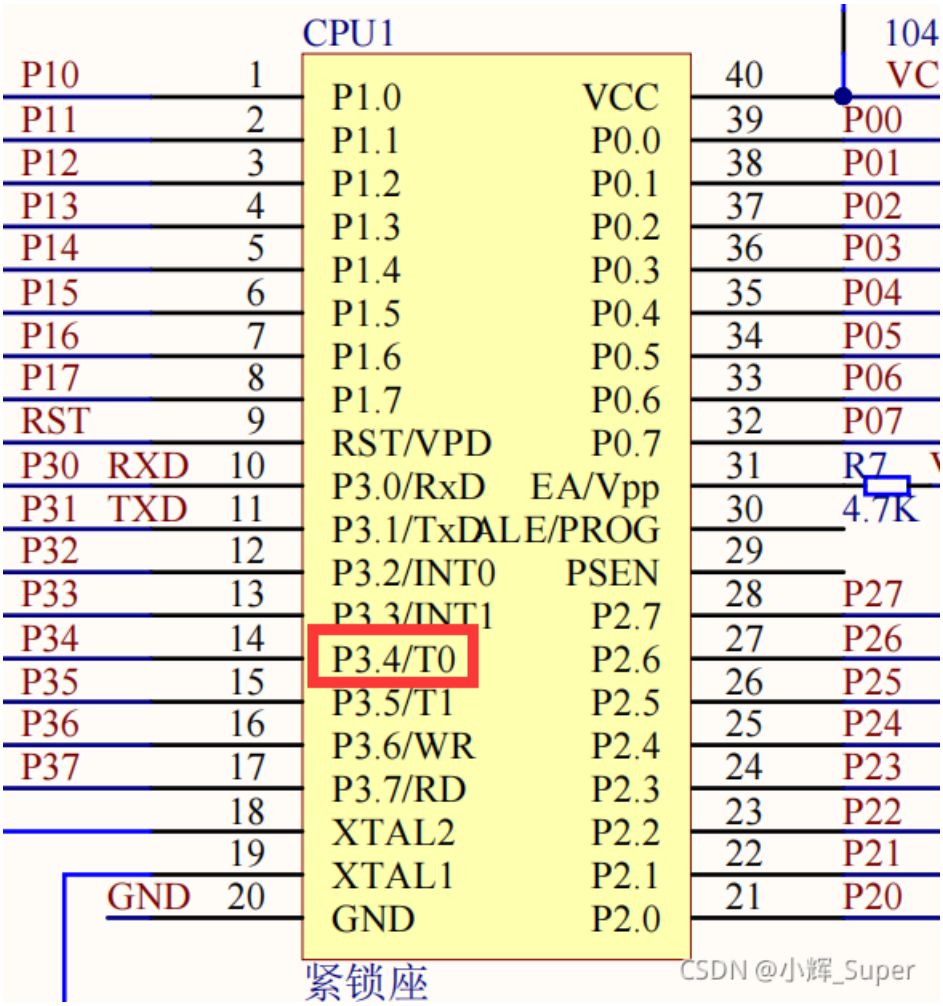
代码和实验12几乎没有区别，只是将按键IO换成P3^3，中断号换成interrupt 2。

```
/*********************************************************************************
实验现象：下载程序后，操作K4按键使LED1（D11）状态取反

接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）
     1，单片机-->LED&交通灯模块
       P20-->D1
     2，单片机-->独立按键模块
       P33-->K4

注意事项：

*********************************************************************************/
#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器
typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;
sbit k4=P3^3;  //定义按键K4
sbit led=P2^0;  //定义P20口是led
```

```
/*************************************************************************
* 函 数 名         : delay
* 函数功能		   : 延时函数，i=1时，大约延时10us
*************************************************************************/
void delay(u16 i)
{
 while(i--);
}
/*************************************************************************
* 函 数 名         : Int1Init()
* 函数功能		   : 设置外部中断1
* 输    入         : 无
* 输    出         : 无
*************************************************************************/
void Int1Init()
{
 //设置INT1
 IT1=1;//跳变沿出发方式（下降沿）
 EX1=1;//打开INT1的中断允许。
 EA=1;//打开总中断
}
/*************************************************************************
* 函 数 名       : main
* 函数功能		 : 主函数
* 输    入       : 无
* 输    出       : 无
*************************************************************************/
void main()
{
 Int1Init();  // 设置外部中断1
 while(1);
}
/*************************************************************************
* 函 数 名         : Int1() interrupt 2*
* 函数功能		   : 外部中断0的中断函数
* 输    入         : 无
* 输    出         : 无
*************************************************************************/

void Int1() interrupt 2  //外部中断1的中断函数
{
 delay(1000);  //延时消抖
 if(k4==0)
 {
  led=~led;
 }
}
```

实验14：定时器0中断

STC89C5X 单片机内有两个可编程的定时/计数器 T0、T1 和一个特殊功能定时器 T2。定时/计数器的实质是加 1 计数器（16 位），由高 8 位和低 8 位两 个寄存器 THx 和 TLx 组成。它随着计数器的输入脉冲进行自加 1，也就是每来一 个脉冲，计数器就自动加 1，当加到计数器为全 1 时，再输入一个脉冲就使计数 器回零，且计数器的溢出使相应的中断标志位置 1，向 CPU 发出中断请求（定时 /计数器中断允许时）。如果定时/计数器工作于定时模式，则表示定时时间已到； 如果工作于计数模式，则表示计数值已满。可见，由溢出时计数器的值减去计数 初值才是加 1 计数器的计数值。——《普中51单片机开发攻略–A2》



定时器中断的使用方法和外部中断类似，都是先配置在编写中断服务函数，区别于外部中断的是定时器需要配置模式TMOD和计数值TH0、TL0，这里的计数值怎么填呢？

51定时器计数值计算公式是：机器周期=1/单片机的时钟频率。51 单片机内部时钟频率是外部时钟的12分频，也就是说当外部晶振的频率输入到单片机里面的时候要进行12分频。比如说你用的是 12MHZ 晶振，那么单片机内部的时钟频率就是 12/12MHZ，当你使用 12MHZ 的外部晶振的时候，机器周期=1/1M=1us。如果我们想定时 1ms的初值是多少呢？1ms/1us=1000。也就是要计数 1000 个，初值=65535-1000+1 （因为实际上计数器计数到 65536（2 的 16 次方）才溢出，所以后面要加 1）=64536=FC18H，所以初值即为 THx=0XFC，TLx=0X18。——《普中51单片机开发攻略–A2》

每当定时器的计数值到达65535，计数值就会溢出，从而触发中断，程序会自动调用定时器中断函数（需要用interrupt 1指定）。本例程中，中断服务函数内部首先重新赋值了计数值TH0和TL0，即开启下一次计时，接着将静态变量i进行了自加操作，当 i 到达1000时，切换LED的电平，实现灯的闪烁，1ms计时进行了1000次，所以灯闪烁的频率是1秒。

定时器中使用静态变量或全局变量自加自减可以实现长时间计时的功能。

```
/*********************************************************************************
实验现象：下载程序后，D1小灯循环点亮1秒，熄灭1秒。使用单片机内部定时器可以实现准确延时

接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）
    1，单片机-->LED&交通灯模块
      P20-->D1
```

```
注意事项：


*******************************************************************************/

#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;

sbit led=P2^0;  //定义P20口是led


/*******************************************************************************
* 函 数 名         : Timer0Init
* 函数功能     : 定时器0初始化
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/
void Timer0Init()
{
 TMOD|=0X01;//选择为定时器0模式，工作方式1，仅用TR0打开启动。

 TH0=0XFC; //给定时器赋初值，定时1ms
 TL0=0X18;
 ET0=1;//打开定时器0中断允许
 EA=1;//打开总中断
 TR0=1;//打开定时器
}

/*******************************************************************************
* 函 数 名       : main
* 函数功能   : 主函数
* 输    入       : 无
* 输    出      : 无
*******************************************************************************/
void main()
{
 Timer0Init();  //定时器0初始化
 while(1);
}

/*******************************************************************************
* 函 数 名         : void Timer0() interrupt 1
* 函数功能     : 定时器0中断函数
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/
void Timer0() interrupt 1
{
 static u16 i;
 TH0=0XFC; //给定时器赋初值，定时1ms
 TL0=0X18;
 i++;
 if(i==1000)
 {
  i=0;
  led=~led;
 }
```

```
 }
}
```

实验15：定时器1中断

定时器1和定时器0的使用方法一样，只需把0改成1，把中断号改成3。
本实验定时器服务函数里实现了数码管从0-F的循环显示。

```
/*********************************************************************************
实验现象：下载程序后，静态数码管间隔一秒循环显示0-F

接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）
     1，单片机-->静态数码管模块
       J22-->J8

注意事项：


*********************************************************************************/

#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;


u8 code smgduan[17]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,
     0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};//显示0~F的值
u8 n=0;
/*********************************************************************************
* 函 数 名         : Timer1Init
* 函数功能     : 定时器1初始化
* 输    入         : 无
* 输    出         : 无
*********************************************************************************/
void Timer1Init()
{
 TMOD|=0X10;//选择为定时器1模式，工作方式1，仅用TR1打开启动。

 TH1=0XFC; //给定时器赋初值，定时1ms
 TL1=0X18;
 ET1=1;//打开定时器1中断允许
 EA=1;//打开总中断
 TR1=1;//打开定时器
}

/*********************************************************************************
* 函 数 名       : main
* 函数功能   : 主函数
* 输    入       : 无
* 输    出       : 无
*********************************************************************************/
void main()
{
 Timer1Init();  //定时器1初始化
 while(1);
}

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

```
/*******************************************************************************
* 函 数 名         : void Timer1() interrupt 3
* 函数功能		   : 定时器0中断函数
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/
void Timer1() interrupt 3
{
 static u16 i;
 TH1=0XFC; //给定时器赋初值，定时1ms
 TL1=0X18;
 i++;
 if(i==1000)
 {
  i=0;
  P0=smgduan[n++];
  if(n==16)n=0;
 }
}
```

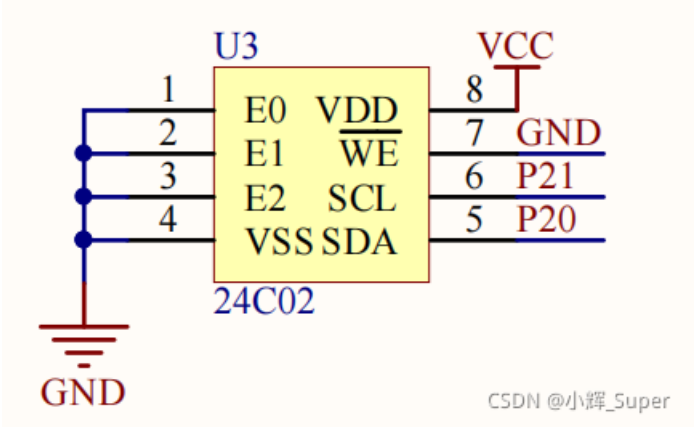
实验16：EEPROM-IIC

本实验涉及I2C协议，I2C（Inter－Integrated Circuit）总线是由 PHILIPS 公司开发的两线式 串行总线，用于连接微控制器及其外围设备。是微电子通信控制领域广泛采用的 一种总线标准。

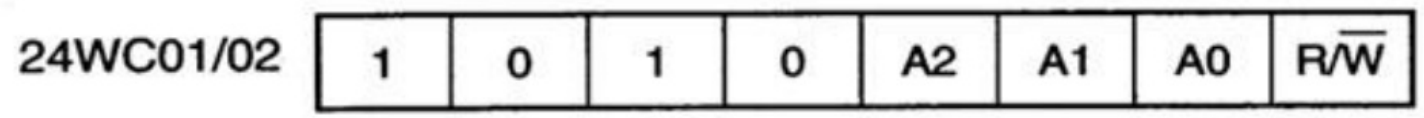
本人对I2C只是有一定了解，能够通过代码实现模拟I2C，但是要把I2C讲清楚，可不是一件简单的差事，加上我打算一天过完所有基本实验，所以关于I2C协议的细节我就暂时不细讲了。

本实验的另一个主角是EEPROM芯片，开发板上的芯片型号为AT24C02，AT24C02是一个2K位串行CMOS E2PROM， 内部含有256个8位字节，CATALYST公司的先进CMOS技术实质上减少了器件的功耗。AT24C02有一个16字节页写缓冲器。该器件通过IIC总线接口进行操作，有一个专门的写保护功能。——百度百科

此芯片内保存的数据在掉电情况下都不丢失， 所以通常用于存放一些比较重要的数据等，芯片参数等细节我就不去研究，就简单看看原理图吧：
AT24C02有8个引脚，5号脚和6号脚与单片机IO相连，分别是I2C_SDA数据线和I2C_SCL时钟线，该芯片与单片机的所有数据传输都由两个引脚完成，1-3号引脚为地址输入引脚，原理图中都接地，所以都是0。



作为I2C设备，都需要设有I2C设备地址，单片机每次对I2C外设进行读写时，都需要先发送设备的地址和读写位，这个地址有7位的，也有10位的，这里只讨论7位地址。下图为AT24C0X芯片的地址示意图，最低位为读写位，告诉I2C从机（这里指AT24C0X)接下来是读取数据还是写入数据；高7位为设备地址，设备地址的高4位为固定值，由芯片厂商写入，低3位可以通过硬件引脚设置，由于原理图中3个引脚都接地，所以开发板中AT24C02的设备地址为1010000，转成16进制为50。（I2C传输时还要加上读写位）



实验中I2C的模拟都在i2c.c文件中，文件中包括I2C起始信号发送函数I2cStart，终止信号发送函数I2cStop，单字节发送函数I2cSendByte，单字节读取函数I2cReadByte。这四个函数都是针对I2C协议编写的，目的是模拟底层I2C信号；除了这四个基本函数，文件还包括At24c02写数据函数At24c02Write，At24c02读数据函数At24c02Read，这两个函数是针对EEPROM的功能函数，里面是用I2C基本函数组合而成的，这两个函数里实现了I2C通信的完整过程。

i2c.c:

```
#include"i2c.h"

/*******************************************************************************
* 函数名         : Delay10us()
* 函数功能		 : 延时10us
* 输入           : 无
* 输出           : 无
*******************************************************************************/

void Delay10us()
{
 unsigned char a,b;
 for(b=1;b>0;b--)
  for(a=2;a>0;a--);

}
/*******************************************************************************
* 函数名         : I2cStart()
* 函数功能	     : 起始信号：在SCL时钟信号在高电平期间SDA信号产生一个下降沿
* 输入           : 无
* 输出           : 无
* 备注           : 起始之后SDA和SCL都为0
*******************************************************************************/

void I2cStart()
{
 SDA=1;
 Delay10us();
 SCL=1;
 Delay10us();//建立时间是SDA保持时间>4.7us
 SDA=0;
 Delay10us();//保持时间是>4us
 SCL=0;
 Delay10us();
}
/*******************************************************************************
* 函数名         : I2cStop()
* 函数功能	     : 终止信号：在SCL时钟信号高电平期间SDA信号产生一个上升沿
* 输入           : 无
* 输出           : 无
```

```
* 备注            : 结束之后保持SDA和SCL都为1；表示总线空闲
*******************************************************************************/

void I2cStop()
{
 SDA=0;
 Delay10us();
 SCL=1;
 Delay10us();//建立时间大于4.7us
 SDA=1;
 Delay10us();
}
/*******************************************************************************
* 函数名          : I2cSendByte(unsigned char dat)
* 函数功能    : 通过I2C发送一个字节。在SCL时钟信号高电平期间，保持发送信号SDA保持稳定
* 输入            : num
* 输出            : 0或1。发送成功返回1，发送失败返回0
* 备注            : 发送完一个字节SCL=0,SDA=1
*******************************************************************************/

unsigned char I2cSendByte(unsigned char dat)
{
 unsigned char a=0,b=0;//最大255，一个机器周期为1us，最大延时255us。
 for(a=0;a<8;a++)//要发送8位，从最高位开始
 {
  SDA=dat>>7;  //起始信号之后SCL=0，所以可以直接改变SDA信号
  dat=dat<<1;
  Delay10us();
  SCL=1;
  Delay10us();//建立时间>4.7us
  SCL=0;
  Delay10us();//时间大于4us
 }
 SDA=1;
 Delay10us();
 SCL=1;
 while(SDA)//等待应答，也就是等待从设备把SDA拉低
 {
  b++;
  if(b>200)  //如果超过2000us没有应答发送失败，或者为非应答，表示接收结束
  {
   SCL=0;
   Delay10us();
   return 0;
  }
 }
 SCL=0;
 Delay10us();
  return 1;
}
/*******************************************************************************
* 函数名          : I2cReadByte()
* 函数功能      : 使用I2c读取一个字节
* 输入            : 无
* 输出            : dat
* 备注            : 接收完一个字节SCL=0,SDA=1.
*******************************************************************************/

unsigned char I2cReadByte()
{
```

```
 unsigned char a=0,dat=0;
 SDA=1;   //起始和发送一个字节之后SCL都是0
 Delay10us();
 for(a=0;a<8;a++)//接收8个字节
 {
  SCL=1;
  Delay10us();
  dat<<=1;
  dat|=SDA;
  Delay10us();
  SCL=0;
  Delay10us();
 }
 return dat;
}


/*****************************************************************************
* 函数名         : void At24c02Write(unsigned char addr,unsigned char dat)
* 函数功能     : 往24c02的一个地址写入一个数据
* 输入           : 无
* 输出           : 无
*****************************************************************************/

void At24c02Write(unsigned char addr,unsigned char dat)
{
 I2cStart();
 I2cSendByte(0xa0);//发送写器件地址
 I2cSendByte(addr);//发送要写入内存地址
 I2cSendByte(dat); //发送数据
 I2cStop();
}
/*****************************************************************************
* 函数名         : unsigned char At24c02Read(unsigned char addr)
* 函数功能     : 读取24c02的一个地址的一个数据
* 输入           : 无
* 输出           : 无
*****************************************************************************/

unsigned char At24c02Read(unsigned char addr)
{
 unsigned char num;
 I2cStart();
 I2cSendByte(0xa0); //发送写器件地址
 I2cSendByte(addr); //发送要读取的地址
 I2cStart();
 I2cSendByte(0xa1); //发送读器件地址
 num=I2cReadByte(); //读取数据
 I2cStop();
 return num;
}
```

I2C的头文件定义了I2C信号的IO引脚：

i2c.h

```
#ifndef __I2C_H_
#define __I2C_H_

#include <reg52.h>

sbit SCL=P2^1;
sbit SDA=P2^0;

void I2cStart();
void I2cStop();
unsigned char I2cSendByte(unsigned char dat);
unsigned char I2cReadByte();
void At24c02Write(unsigned char addr,unsigned char dat);
unsigned char At24c02Read(unsigned char addr);

#endif
```

main.c实现了通过4个按键操控EEPROM和数码管，主要函数包括Keypros()，datapros()，DigDisplay()。
Keypros负责读取按键并处理，K1按下时，单片机将全局变量num写入AT24C02的1号地址空间；K2按下时，单片机从AT24C02的1号地址空间读取数据，并赋值给num；K3按下时，num加1；K4按下时，num赋值为0。
datapros函数负责将num的千位到个位数字传给控制数码管的数组。
DigDisplay函数则将数码管数组disp[i]的值通过数码管显示出来。

main.c

```
/*********************************************************************************
实验现象：下载程序后数码管后4位显示0，按K1保存显示的数据，按K2读取上次保存的数据，
    按K3显示数据加一，按K4显示数据清零。最大能写入的数据是255

接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）


注意事项：
*********************************************************************************/

#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器
#include "i2c.h"

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;

sbit LSA=P2^2;
sbit LSB=P2^3;
sbit LSC=P2^4;

sbit k1=P3^1;
sbit k2=P3^0;
sbit k3=P3^2;
sbit k4=P3^3;  //定义按键端口

char num=0;
u8 disp[4];
u8 code smgduan[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};

/*********************************************************************************
* 函 数 名         : delay
* 函数功能     : 延时函数，i=1时，大约延时10us
*********************************************************************************/
void delay(u16 i)
```

```
{
 while(i--);
}


/**********************************************************************************
* 函数名         :Keypros()
* 函数功能    :按键处理函数
* 输入           : 无
* 输出           : 无
**********************************************************************************/
void Keypros()
{
 if(k1==0)
 {
  delay(1000);  //消抖处理
  if(k1==0)
  {
   At24c02Write(1,num);   //在地址1内写入数据num
  }
  while(!k1);
 }
 if(k2==0)
 {
  delay(1000);  //消抖处理
  if(k2==0)
  {
   num=At24c02Read(1);   //读取EEPROM地址1内的数据保存在num中
  }
  while(!k2);
 }
 if(k3==0)
 {
  delay(100);  //消抖处理
  if(k3==0)
  {
   num++;    //数据加1
   if(num>255)num=0;
  }
  while(!k3);
 }
 if(k4==0)
 {
  delay(1000);  //消抖处理
  if(k4==0)
  {
   num=0;   //数据清零
  }
  while(!k4);
 }
}

/**********************************************************************************
* 函数名         :datapros()
* 函数功能    :数据处理函数
* 输入           : 无
* 输出           : 无
**********************************************************************************/
void datapros()
```

```
{
 disp[0]=smgduan[num/1000];//千位
 disp[1]=smgduan[num%1000/100];//百位
 disp[2]=smgduan[num%1000%100/10];//十位
 disp[3]=smgduan[num%1000%100%10];//个位
}


/*****************************************************************************
* 函数名         :DigDisplay()
* 函数功能    :数码管显示函数
* 输入            : 无
* 输出            : 无
*****************************************************************************/
void DigDisplay()
{
 u8 i;
 for(i=0;i<4;i++)
 {
  switch(i)  //位选，选择点亮的数码管，
  {
   case(0):
    LSA=1;LSB=1;LSC=0; break;//显示第0位
   case(1):
    LSA=0;LSB=1;LSC=0; break;//显示第1位
   case(2):
    LSA=1;LSB=0;LSC=0; break;//显示第2位
   case(3):
    LSA=0;LSB=0;LSC=0; break;//显示第3位
  }
  P0=disp[i];//发送数据
  delay(100); //间隔一段时间扫描
  P0=0x00;//消隐
 }
}

/*****************************************************************************
* 函 数 名       : main
* 函数功能    : 主函数
* 输    入       : 无
* 输    出       : 无
*****************************************************************************/
void main()
{
 while(1)
 {
  Keypros();  //按键处理函数
  datapros();  //数据处理函数
  DigDisplay();//数码管显示函数
 }
}
```

实验17：DS18B20温度传感器

DS18B20 是由 DALLAS 半导体公司推出的一种的“一线总线（单总线）”接 口的温度传感器。与传统的热敏电阻等测温元件相比，它是一种新型的体积小、 适用电压宽、与微处理器接口简单的数字化温度传感器。

和I2C一样，单总线也是一种通信协议，目的都是为了传输数据，所有的单总线器件都要求采用严格的信号时序，以保证数据的完整性。DS18B20 时序包括如下几种：初始化时序、写（0 和 1）时序、 读（0 和 1）时序。这些时序我暂时不想了解，下面参考《普中51单片机开发攻略–A2》里的描述：

初始化时序：
单总线上的所有通信都是以初始化序列开始。主机输出低电平，保持低电平 时间至少 480us（该时间的时间范围可以从 480 到 960 微秒），以产生复位脉 冲。接着主机释放总线，外部的上拉电阻将单总线拉高，延时 15～60 us，并进入接收模式。接着 DS18B20 拉低总线 60~240 us，以产生低电平应答脉冲，若为低电平，还要做延时，其延时的时间从外部上拉电阻将单总线拉高算起最少要 480 微秒。

写时序
写时序包括写 0 时序和写 1 时序。所有写时序至少需要 60us，且在 2 次 独立的写时序之间至少需要 1us 的恢复时间，两种写时序均起始于主机拉低总 线。写 1 时序：主机输出低电平，延时 2us，然后释放总线，延时 60us。写 0 时序：主机输出低电平，延时 60us，然后释放总线，延时 2us。

读时序
单总线器件仅在主机发出读时序时，才向主机传输数据，所以，在主机发出 读数据命令后，必须马上产生读时序，以便从机能够传输数据。所有读时序至少 需要 60us，且在 2 次独立的读时序之间至少需要 1us 的恢复时间。每个读时 序都由主机发起，至少拉低总线 1us。主机在读时序期间必须释放总线，并且在 时序起始后的 15us 之内采样总线状态。
典型的读时序过程为：主机输出低电平延时 2us，然后主机转入输入模式延时 12us，然后读取单总线当前的电平，然后延时 50us。

除了时序，DS18B20另一个重要的操作就是将读取到的数据转换成温度值，DS18B20 温度传感器的内部存储器包括一个高速的暂存器 RAM 和一个非易失性的可电擦除的 EEPROM,后者存放高温度和低温度触发器 TH、TL 和配置寄存器。配置寄存器是配置不同的位数来确定温度和数字的转化，配置寄存器结构如下

TM	R1	R0	1	1	1	1	1

低五位一直都是"1"，TM 是测试模式位，用于设置 DS18B20 在工作模式还 是在测试模式。在 DS18B20 出厂时该位被设置为 0，用户不需要去改动。R1 和 R0 用来设置 DS18B20 的精度（分辨率），可设置为 9，10，11 或 12 位，对 应的分辨率温度是 0.5℃，0.25℃，0.125℃和 0.0625℃。R0 和 R1 配置如下 图

R1	R0	精度	最大转换时间	
0	0	9-bit	93.75 ms	(t_{CONV}/8)
0	1	10-bit	187.5 ms	(t_{CONV}/4)
1	0	11-bit	375 ms	(t_{CONV}/2)
1	1	12-bit	750 ms	(t_{CONV})

在初始状态下默认的精度是 12 位，即 R0=1、 R1=1。高速暂存存储器由 9 个字节组成，其分配如下：

寄存器内容	字节地址
温度值低位（LS Byte）	0
温度值高位（MS Byte）	1
高温限值（TH）	2
低温限值（TL）	3
配置寄存器	4
保留	5
保留	6
保留	7
CRC校验值	8

当温度转换命令（44H）发布后，经转换所得的温度值以二字节补码形式存 放在高速暂存存储器的第 0 和第 1 个字节。存储的两个字节，高字节的前 5 位 是符号位 S，单片机可通过单线接口读到该数据，读取时低位在前，高位在后， 数据格式如下

温度寄存器格式 图 2

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LS Byte	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
MS Byte	S	S	S	S	S	2^6	2^5	2^4

如果测得的温度大于 0，这 5 位为‘ 0’，只要将测到的数值乘以 0.0625 （默认精度是 12 位）即可得到实际温度；如果温度小于 0，这 5 位为‘ 1’， 测到的数值需要取反加 1 再乘以 0.0625 即可得到实际温度。温度与数据对应 关系如下：

温度/数据关系 表 2

温度 ℃	数据输出（二进制）	数据输出（十六进制）
+125	0000 0111 1101 0000	07D0h
+85	0000 0101 0101 0000	0550h
+25.0625	0000 0001 1001 0001	0191h
+10.125	0000 0000 1010 0010	00A2h
+0.5	0000 0000 0000 1000	0008h
0	0000 0000 0000 0000	0000h
-0.5	1111 1111 1111 1000	FFF8h
-10.125	1111 1111 0101 1110	FF5Eh
-25.0625	1111 1110 0110 1111	FE6Eh
-55	1111 1100 1001 0000	FC90h

*上电复位时温度寄存器默认值为+85℃

比如我们要计算+85 度，数据输出十六进制是 0X0550，因为高字节的高 5 位为 0，表明检测的温度是正温度，0X0550 对应的十进制为 1360，将这个值乘 以 12 位精度 0.0625，所以可以得到+85 度。

————以上关于温度读取的内容均摘抄自《普中51单片机开发攻略–A2》

temp.c包含了DS18B20的时序和温度转换功能函数，Ds18b20Init的作用是初始化DS18B20模块，是初始化时序的代码实现；Ds18b20WriteByte用来向DS18B20写入1字节数据，对应写时序；Ds18b20ReadByte作用是从DS18B20读取1个字节的数据，对应读时序；Ds18b20ChangTemp函数让DS18B20开始转换温度，默认精度为12bit，对应等待时间750ms，函数中屏蔽了100ms延时，不知会有什么影响（实际运行看没影响）；Ds18b20ReadTempCom函数发送读取命令给DS18B20；Ds18b20ReadTemp是上面所以函数的集合，是应用中直接调用的函数，包括了读取温度的所有过程。

temp.c

```
#include"temp.h"
/*******************************************************************************
* 函 数 名         : Delay1ms
* 函数功能		   : 延时函数
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/

void Delay1ms(uint y)
{
 uint x;
 for( ; y>0; y--)
 {
  for(x=110; x>0; x--);
 }
}
/*******************************************************************************
* 函 数 名         : Ds18b20Init
* 函数功能		   : 初始化
* 输    入         : 无
* 输    出         : 初始化成功返回1，失败返回0
*******************************************************************************/

uchar Ds18b20Init()
{
 uchar i;
 DSPORT = 0;     //将总线拉低480us~960us
 i = 70;
 while(i--);//延时642us
 DSPORT = 1;    //然后拉高总线，如果DS18B20做出反应会将在15us~60us后总线拉低
 i = 0;
 while(DSPORT) //等待DS18B20拉低总线
 {
  Delay1ms(1);
  i++;
  if(i>5)//等待>5MS
  {
   return 0;//初始化失败
  }

 }
 return 1;//初始化成功
}

/*******************************************************************************
* 函 数 名         : Ds18b20WriteByte
* 函数功能		   : 向18B20写入一个字节
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/
```

```
void Ds18b20WriteByte(uchar dat)
{
 uint i, j;

 for(j=0; j<8; j++)
 {
  DSPORT = 0;          //每写入一位数据之前先把总线拉低1us
  i++;
  DSPORT = dat & 0x01;  //然后写入一个数据，从最低位开始
  i=6;
  while(i--); //延时68us，持续时间最少60us
  DSPORT = 1; //然后释放总线，至少1us给总线恢复时间才能接着写入第二个数值
  dat >>= 1;
 }
}
/*****************************************************************************
* 函 数 名         : Ds18b20ReadByte
* 函数功能     : 读取一个字节
* 输    入         : 无
* 输    出         : 无
*****************************************************************************/


uchar Ds18b20ReadByte()
{
 uchar byte, bi;
 uint i, j;
 for(j=8; j>0; j--)
 {
  DSPORT = 0;//先将总线拉低1us
  i++;
  DSPORT = 1;//然后释放总线
  i++;
  i++;//延时6us等待数据稳定
  bi = DSPORT;  //读取数据，从最低位开始读取
  /*将byte左移一位，然后与上右移7位后的bi，注意移动之后移掉那位补0。*/
  byte = (byte >> 1) | (bi << 7);
  i = 4;  //读取完之后等待48us再接着读取下一个数
  while(i--);
 }
 return byte;
}
/*****************************************************************************
* 函 数 名         : Ds18b20ChangTemp
* 函数功能     : 让18b20开始转换温度
* 输    入         : 无
* 输    出         : 无
*****************************************************************************/

void  Ds18b20ChangTemp()
{
 Ds18b20Init();
 Delay1ms(1);
 Ds18b20WriteByte(0xcc);  //跳过ROM操作命令
 Ds18b20WriteByte(0x44);     //温度转换命令
 //Delay1ms(100); //等待转换成功，而如果你是一直刷着的话，就不用这个延时了

}
/*****************************************************************************
```

```
* 函 数 名         : Ds18b20ReadTempCom
* 函数功能     : 发送读取温度命令
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/

void  Ds18b20ReadTempCom()
{

 Ds18b20Init();
 Delay1ms(1);
 Ds18b20WriteByte(0xcc);  //跳过ROM操作命令
 Ds18b20WriteByte(0xbe);  //发送读取温度命令
}
/*******************************************************************************
* 函 数 名         : Ds18b20ReadTemp
* 函数功能     : 读取温度
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/

int Ds18b20ReadTemp()
{
 int temp = 0;
 uchar tmh, tml;
 Ds18b20ChangTemp();     //先写入转换命令
 Ds18b20ReadTempCom();   //然后等待转换完后发送读取温度命令
 tml = Ds18b20ReadByte();  //读取温度值共16位，先读低字节
 tmh = Ds18b20ReadByte();  //再读高字节
 temp = tmh;
 temp <<= 8;
 temp |= tml;
 return temp;
}
```

TEMP.H

```
#ifndef __TEMP_H_
#define __TEMP_H_

#include<reg52.h>
//---重定义关键词---//
#ifndef uchar
#define uchar unsigned char
#endif

#ifndef uint
#define uint unsigned int
#endif

//--定义使用的IO口--//
sbit DSPORT=P3^7;

//--声明全局函数--//
void Delay1ms(uint );
uchar Ds18b20Init();
void Ds18b20WriteByte(uchar com);
uchar Ds18b20ReadByte();
void  Ds18b20ChangTemp();
void  Ds18b20ReadTempCom();
int Ds18b20ReadTemp();

#endif
```

main.c实现的功能是不停读取DS18B20的温度数据并更新数码管的显示，datapros作用是把从DS18B20读取的数据按照规定的格式转换成摄氏度，再将温度值的十位、个位、十分位和百分位分别显示在1-5号数码管上，当温度低于0度，0号数码管上会显示负号‘-’。

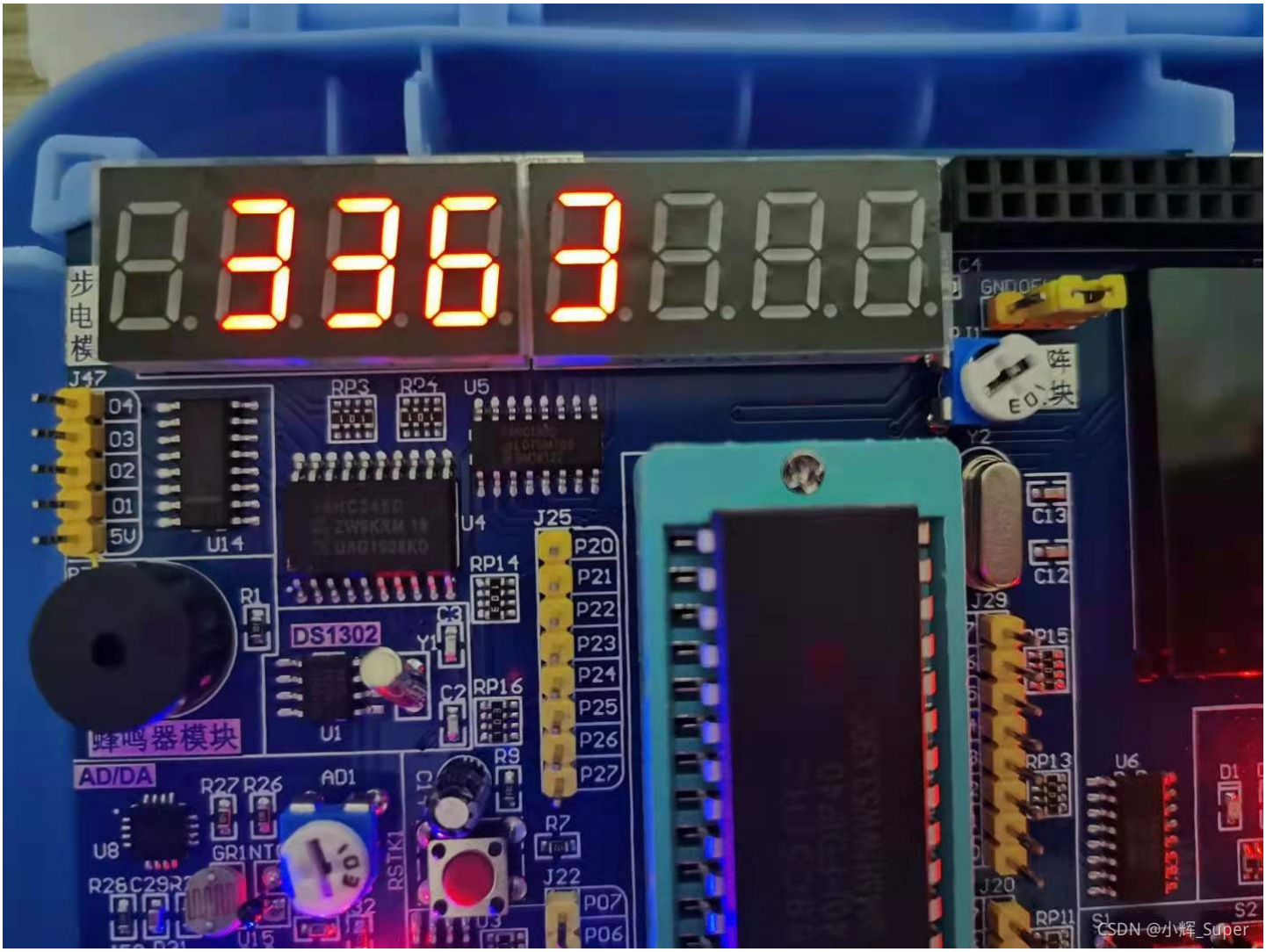
main.c

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

*
实验现象：下载程序后，在温度传感器接口处，按照丝印方向插好温度传感器，数码管就会显示
    检测的温度值

接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）


注意事项：


*********************************************************************************

*/

#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器
#include"temp.h"

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;

sbit LSA=P2^2;
sbit LSB=P2^3;
sbit LSC=P2^4;
```

```
char num=0;
u8 DisplayData[8];
u8 code smgduan[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};

/*******************************************************************************
* 函 数 名         : delay
* 函数功能     : 延时函数，i=1时，大约延时10us
*******************************************************************************/
void delay(u16 i)
{
 while(i--);
}


/*******************************************************************************
* 函 数 名         : datapros()
* 函数功能     : 温度读取处理转换函数
* 输    入         : temp
* 输    出         : 无
*******************************************************************************/

void datapros(int temp)
{
    float tp;
 if(temp< 0)    //当温度值为负数
   {
  DisplayData[0] = 0x40;    //   -
  //因为读取的温度是实际温度的补码，所以减1，再取反求出原码
  temp=temp-1;
  temp=~temp;
  tp=temp;
  temp=tp*0.0625*100+0.5;
  //留两个小数点就*100，+0.5是四舍五入，因为C语言浮点数转换为整型的时候把小数点
  //后面的数自动去掉，不管是否大于0.5，而+0.5之后大于0.5的就是进1了，小于0.5的就
  //算加上0.5，还是在小数点后面。

   }
  else
   {
  DisplayData[0] = 0x00;
  tp=temp;//因为数据处理有小数点所以将温度赋给一个浮点型变量
  //如果温度是正的那么，那么正数的原码就是补码它本身
  temp=tp*0.0625*100+0.5;
  //留两个小数点就*100，+0.5是四舍五入，因为C语言浮点数转换为整型的时候把小数点
  //后面的数自动去掉，不管是否大于0.5，而+0.5之后大于0.5的就是进1了，小于0.5的就
  //算加上0.5，还是在小数点后面。
 }
 DisplayData[1] = smgduan[temp % 10000 / 1000];
 DisplayData[2] = smgduan[temp % 1000 / 100];
 DisplayData[3] = smgduan[temp %  100 / 10];
 DisplayData[4] = smgduan[temp %  10 / 1];

}


/*******************************************************************************
* 函数名         :DigDisplay()
* 函数功能    :数码管显示函数
* 输入           : 无
* 输出           : 无
```

```
* 输出             : 无
*******************************************************************************/
void DigDisplay()
{
 u8 i;
 for(i=0;i<6;i++)
 {
  switch(i)  //位选，选择点亮的数码管，
  {
   case(0):
    LSA=1;LSB=1;LSC=1; break;//显示第0位
   case(1):
    LSA=0;LSB=1;LSC=1; break;//显示第1位
   case(2):
    LSA=1;LSB=0;LSC=1; break;//显示第2位
   case(3):
    LSA=0;LSB=0;LSC=1; break;//显示第3位
   case(4):
    LSA=1;LSB=1;LSC=0; break;//显示第4位
   case(5):
    LSA=0;LSB=1;LSC=0; break;//显示第5位
  }
  P0=DisplayData[i];//发送数据
  delay(100); //间隔一段时间扫描
  P0=0x00;//消隐
 }
}

/*******************************************************************************
* 函 数 名         : main
* 函数功能   : 主函数
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/
void main()
{
 while(1)
 {
  datapros(Ds18b20ReadTemp());  //数据处理函数
  DigDisplay();//数码管显示函数
 }
}
```

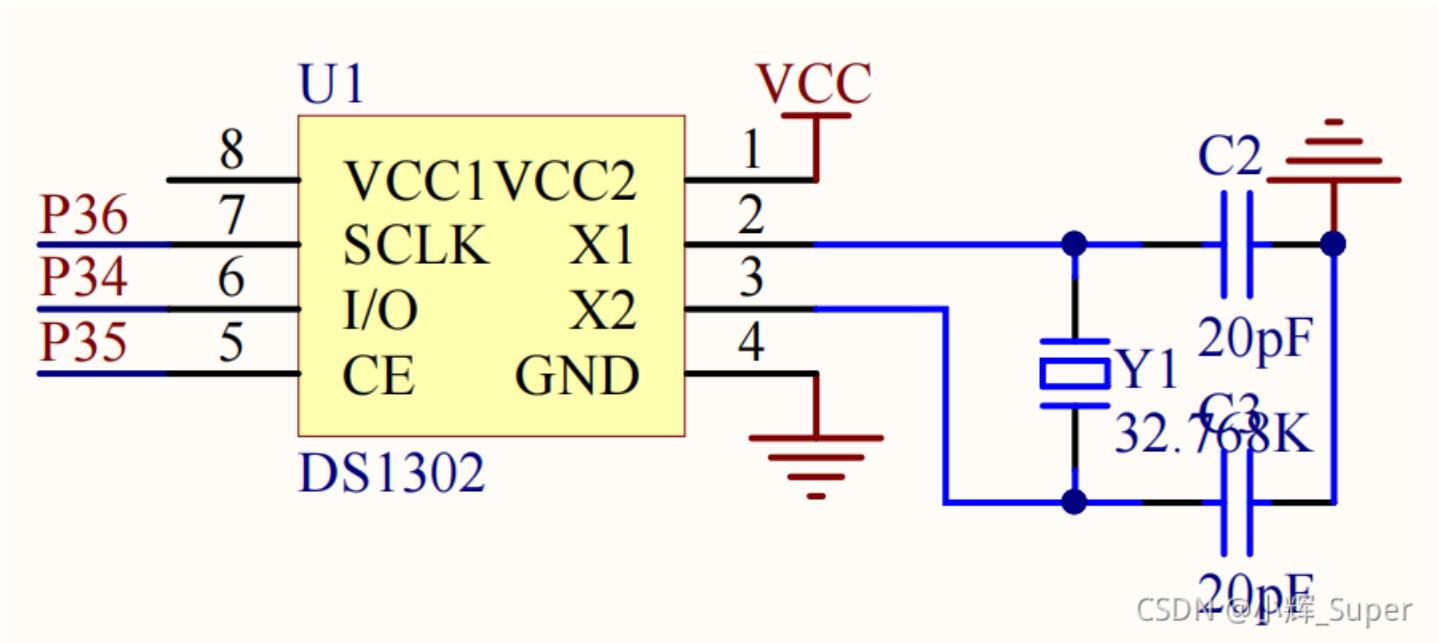
这个实验真是水过去的。。。



实验18：DS1302时钟

DS1302 是 DALLAS 公司推出的涓流充电时钟芯片，内含有一个实时时钟/日 历和 31 字节静态 RAM，通过简单的串行接口与单片机进行通信。实时时钟/日 历电路提供秒、分、时、日、周、月、年的信息，每月的天数和闰年的天数可自动调整。时钟操作可通过 AM/PM 指示决定采用 24 或 12 小时格式。

该芯片使用了一种特殊的信号线，既然这样，就直接跳到学习怎么使用这个模块的环节吧。。。



ds1302.c包含了DS1302的初始化、读地址数据、向地址写数据和读取时间等函数，原理就不管了，先拿来用。

ds1302.c

```
#include"ds1302.h"

//---DS1302写入和读取时分秒的地址命令---//
//---秒分时日月周年 最低位读写位;-------//
uchar code READ_RTC_ADDR[7] = {0x81, 0x83, 0x85, 0x87, 0x89, 0x8b, 0x8d};
uchar code WRITE_RTC_ADDR[7] = {0x80, 0x82, 0x84, 0x86, 0x88, 0x8a, 0x8c};

//---DS1302时钟初始化2016年5月7日星期六12点00分00秒。---//
//---存储顺序是秒分时日月周年,存储格式是用BCD码---//
uchar TIME[7] = {0, 0, 0x12, 0x07, 0x05, 0x06, 0x16};

/*********************************************************************************
* 函 数 名         : Ds1302Write
* 函数功能		   : 向DS1302命令（地址+数据）
* 输    入         : addr,dat
* 输    出         : 无
*********************************************************************************/

void Ds1302Write(uchar addr, uchar dat)
{
 uchar n;
 RST = 0;
 _nop_();

 SCLK = 0;//先将SCLK置低电平。
 _nop_();
 RST = 1; //然后将RST(CE)置高电平。
 _nop_();

 for (n=0; n<8; n++)//开始传送八位地址命令
 {
  DSIO = addr & 0x01;//数据从低位开始传送
  addr >>= 1;
  SCLK = 1;//数据在上升沿时，DS1302读取数据
  _nop_();
```

```
 _nop_();
 SCLK = 0;
 _nop_();
 }
 for (n=0; n<8; n++)//写入8位数据
 {
  DSIO = dat & 0x01;
  dat >>= 1;
  SCLK = 1;//数据在上升沿时，DS1302读取数据
  _nop_();
  SCLK = 0;
  _nop_();
 }

 RST = 0;//传送数据结束
 _nop_();
}

/*************************************************************************
* 函 数 名         : Ds1302Read
* 函数功能      : 读取一个地址的数据
* 输    入         : addr
* 输    出         : dat
*************************************************************************/

uchar Ds1302Read(uchar addr)
{
 uchar n,dat,dat1;
 RST = 0;
 _nop_();

 SCLK = 0;//先将SCLK置低电平。
 _nop_();
 RST = 1;//然后将RST(CE)置高电平。
 _nop_();

 for(n=0; n<8; n++)//开始传送八位地址命令
 {
  DSIO = addr & 0x01;//数据从低位开始传送
  addr >>= 1;
  SCLK = 1;//数据在上升沿时，DS1302读取数据
  _nop_();
  SCLK = 0;//DS1302下降沿时，放置数据
  _nop_();
 }
 _nop_();
 for(n=0; n<8; n++)//读取8位数据
 {
  dat1 = DSIO;//从最低位开始接收
  dat = (dat>>1) | (dat1<<7);
  SCLK = 1;
  _nop_();
  SCLK = 0;//DS1302下降沿时，放置数据
  _nop_();
 }

 RST = 0;
 _nop_(); //以下为DS1302复位的稳定时间,必须的。
 SCLK = 1;
 _nop_();
```

```
 DSIO = 0;
 _nop_();
 DSIO = 1;
 _nop_();
 return dat;
}

/*******************************************************************************
* 函 数 名         : Ds1302Init
* 函数功能     : 初始化DS1302.
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/

void Ds1302Init()
{
 uchar n;
 Ds1302Write(0x8E,0X00);   //禁止写保护，就是关闭写保护功能
 for (n=0; n<7; n++)//写入7个字节的时钟信号：分秒时日月周年
 {
  Ds1302Write(WRITE_RTC_ADDR[n],TIME[n]);
 }
 Ds1302Write(0x8E,0x80);   //打开写保护功能
}

/*******************************************************************************
* 函 数 名         : Ds1302ReadTime
* 函数功能     : 读取时钟信息
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/

void Ds1302ReadTime()
{
 uchar n;
 for (n=0; n<7; n++)//读取7个字节的时钟信号：分秒时日月周年
 {
  TIME[n] = Ds1302Read(READ_RTC_ADDR[n]);
 }

}
```

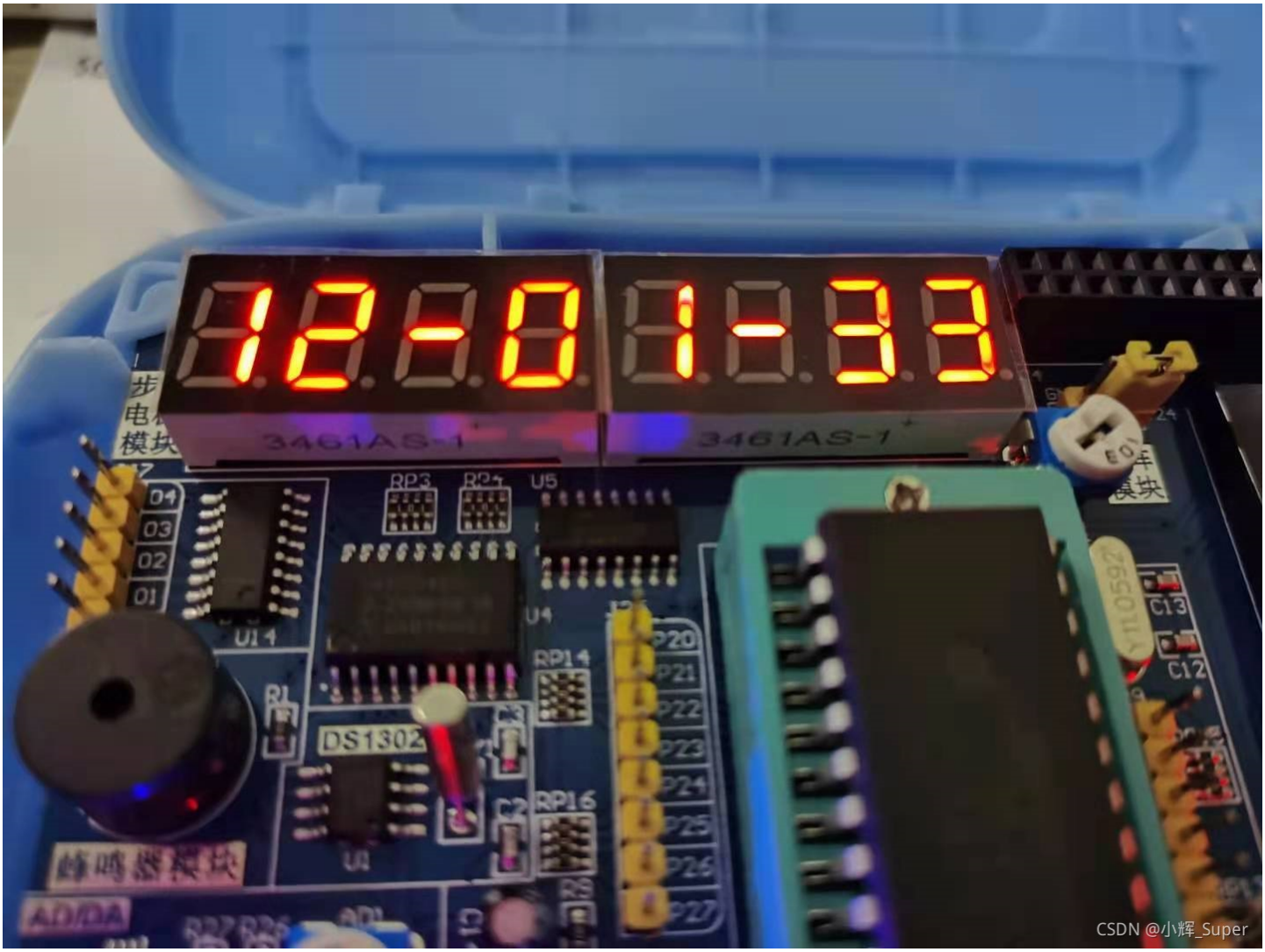
main函数只是完成了读取DS1302时间数据并显示在数码管上的操作。
TIME[7]存储顺序是秒分时日月周年,存储格式是用BCD码，datapros函数中，TIME[2]存储的时小时的数据，包含两个BCD码，假设小时数据为12，则BCD码为0001 0010，如果直接放在程序中使用，程序会把时数据当做十六进制数0x12，十进制为18，显然和原本的12不符。所以datapros里执行了TIME[2]/16，相当于将0001 0010右移4位，变成0000 0001，得出小时的十位数，TIME[2]&0x0f等于0010，得出小时的个位数。一个BCD码最大为10，而时分秒的十位和个位数都不超过9，所以时分秒BCD码高位和低位分别对应时分秒十位和个位。

main.c

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

*
实验现象：下载程序后，数码管显示时钟
```

```
接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）
     1，单片机-->DS1302时钟模块
       P34-->DIO
    P35-->CE
    P36-->CLK
     2，单片机-->动态数码管模块
       J22-->J6
    P22-->J9(A)
    P23-->J9(B)
    P24-->J9(C)

注意事项：


*******************************************************************************

*/

#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器
#include "ds1302.h"

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;

sbit LSA=P2^2;
sbit LSB=P2^3;
sbit LSC=P2^4;


char num=0;
u8 DisplayData[8];
u8 code smgduan[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};

/*******************************************************************************
* 函 数 名         : delay
* 函数功能     : 延时函数，i=1时，大约延时10us
*******************************************************************************/
void delay(u16 i)
{
 while(i--);
}


/*******************************************************************************
* 函 数 名         : datapros()
* 函数功能     : 时间读取处理转换函数
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/

void datapros()
{
    Ds1302ReadTime();
 DisplayData[0] = smgduan[TIME[2]/16];    //时
 DisplayData[1] = smgduan[TIME[2]&0x0f];
 DisplayData[2] = 0x40;
 DisplayData[3] = smgduan[TIME[1]/16];    //分
 DisplayData[4] = smgduan[TIME[1]&0x0f];
 DisplayData[5] = 0x40;
 DisplayData[6] = smgduan[TIME[0]/16];    //秒
```

```
 DisplayData[7] = smgduan[TIME[0]&0x0f];
}


/*****************************************************************************
* 函数名         :DigDisplay()
* 函数功能    :数码管显示函数
* 输入            : 无
* 输出            : 无
*****************************************************************************/
void DigDisplay()
{
 u8 i;
 for(i=0;i<8;i++)
 {
  switch(i)  //位选，选择点亮的数码管，
  {
   case(0):
    LSA=1;LSB=1;LSC=1; break;//显示第0位
   case(1):
    LSA=0;LSB=1;LSC=1; break;//显示第1位
   case(2):
    LSA=1;LSB=0;LSC=1; break;//显示第2位
   case(3):
    LSA=0;LSB=0;LSC=1; break;//显示第3位
   case(4):
    LSA=1;LSB=1;LSC=0; break;//显示第4位
   case(5):
    LSA=0;LSB=1;LSC=0; break;//显示第5位
   case(6):
    LSA=1;LSB=0;LSC=0; break;//显示第6位
    case(7):
    LSA=0;LSB=0;LSC=0; break;//显示第7位
  }
  P0=DisplayData[i];//发送数据
  delay(100); //间隔一段时间扫描
  P0=0x00;//消隐
 }
}

/*****************************************************************************
* 函 数 名       : main
* 函数功能    : 主函数
* 输    入       : 无
* 输    出       : 无
*****************************************************************************/
void main()
{
 //Ds1302Init();    //第一次初始化后就可以注释该条语句，这样下次重启就不会再次初始化了
 while(1)
 {
  datapros();  //数据处理函数
  DigDisplay();//数码管显示函数
 }
}
```



实验19：红外通信

红外遥控是一种无线、非接触控制技术，具有抗干扰能力强，信息传输可靠， 功耗低，成本低，易实现等显著优点，被诸多电子设备特别是家用电器广泛采用， 并越来越多的应用到计算机系统中。

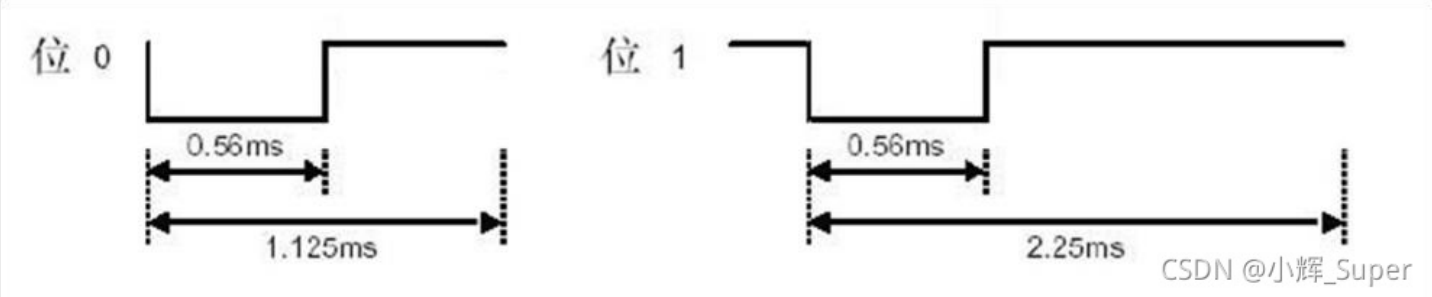
红外遥控通信系统一般由红外发射装置和红外接收设备两大部分组成。

（1）红外发射装置
红外发射装置，也就是通常我们说的红外遥控器，是由键盘电路、红外编码电 路、电源电路和红外发射电路组成。红外发射电路的主要元件为红外发光二极管。 它实际上是一只特殊的发光二极管；由于其内部材料不同于普通发光二极管，因 而在其两端施加一定电压时，它便发出的是红外线而不是可见光。目前大量的使 用的红外发光二极管发出的红外线波长为 940nm 左右，外形与普通发光二极管相同。红外发光二极管有透明的，还有不透明的，在我们的红外遥控器上可以看这个红外发光二极管。

通常红外遥控为了提高抗干扰性能和降低电源消耗，红外遥控器常用载波的 方式传送二进制编码，常用的载波频率为 38kHz，这是由发射端所使用的 455kHz 晶振来决定的。在发射端要对晶振进行整数分频，分频系数一般取 12，所以 455kHz÷12≈37.9kHz≈38kHz。也有一些遥控系统采用 36kHz、 40 kHz、 56 kHz 等，一般由发射端晶振的振荡频率来决定。所以，通常的红外遥控器是将遥控信 号（二进制脉冲码）调制在 38KHz 的载波上，经缓冲放大后送至红外发光二极 管，转化为红外信号发射出去的。

二进制脉冲码的形式有多种，其中最为常用的是 NEC Protocol 的 PWM 码 (脉冲宽度调制)和 Philips RC-5 Protocol 的 PPM 码 (脉冲位置调制码，脉冲 串之间的时间间隔来实现信号调制)。如果要开发红外接收设备，一定要知道红 外遥控器的编码方式和载波频率，我们才可以选取一体化红外接收头和制定解码 方案。本开发板使用的红外遥控器使用的是 NEC 协议。

NEC 码的位定义：一个脉冲对应 560us 的连续载波，一个逻辑 1 传输需要 2.25ms（560us 脉冲+1680us 低电平），一个逻辑 0 的传输需要 1.125ms（560us 脉冲+560us 低电平）。而红外接收头在收到脉冲的时候为低电平，在没有脉冲 的时候为高电平，这样，我们在接收头端收到的信号为：逻辑 1 应该是 560us 低 +1680us 高，逻辑 0 应该是 560us 低+560us 高。所以可以通过计算高电平时 间判断接收到的数据是 0 还是 1。NEC 码位定义时序图如下图所示



NEC 遥控指令的数据格式为：引导码、地址码、地址反码、控制码、控制反码。引导码由一个 9ms 的低电平和一个 4.5ms 的高电平组成，地址码、地址反 码、控制码、控制反码均是 8 位数据格式。按照低位在前，高位在后的顺序发 送。采用反码是为了增加传输的可靠性（可用于校验）。数据格式如下：

NEC 码还规定了连发码(由 9ms 低电平+2.5m 高电平+0.56ms 低电平 +97.94ms 高电平组成)，如果在一帧数据发送完毕之后，红外遥控器按键仍然没 有放开，则发射连发码，可以通过统计连发码的次数来标记按键按下的长短或次数。

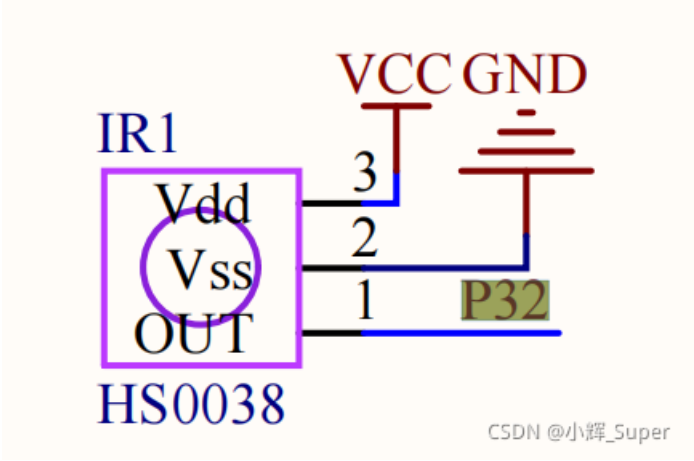
（2）红外接收设备

红外接收设备是由红外接收电路、红外解码、电源和应用电路组成。红外遥 控接收器的主要作用是将遥控发射器发来的红外光信好转换成电信号，再放大、 限幅、检波、整形，形成遥控指令脉冲，输出至遥控微处理器。近几年不论是业 余制作还是正式产品，大多都采用成品红外接收头。成品红外接收头的封装大致 有两种：一种采用铁皮屏蔽；一种是塑料封装。均有三只引脚，即电源正（ VDD）、 电源负（GND）和数据输出（VOUT）。

由于红外接收头在没有脉冲的时候为高电平，当收到脉冲的时候为低电平， 所以可以通过外部中断的下降沿触发中断，在中断内通过计算高电平时间来判断 接收到的数据是 0 还是 1。

————以上关于红外通信的内容均摘抄自《普中51单片机开发攻略–A2》

之前在外部中断0实验中提到P32为51INT0的引脚号，从原理图看红外接收器的引脚，发现1脚也是P32，所以红外接收器也可以使用外部中断0。



代码中，红外接收代码全部放在外部中断0的中断服务函数Readlr中，分别接收了地址码、地址反码、控制码和、控制反码四个数据，每个数据都是8位；main函数中将接收到的控制码高低位分别显示在2个数码管上，第三个数码管则显示‘H’。
红外接收的代码倒是不难，难的是发送，不过这个开发板上没有红外发射器。

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

*
实验现象：下载程序后，数码管显示红外遥控键值数据

接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）
     1，单片机-->红外接收模块
       P32-->J11
     2，单片机-->动态数码管模块
       J22-->J6
    P22-->J9(A)
    P23-->J9(B)
    P24-->J9(C)

注意事项：红外遥控器上的电池绝缘隔片要拿掉


*******************************************************************************

*/

#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器


typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;

sbit LSA=P2^2;
sbit LSB=P2^3;
sbit LSC=P2^4;

sbit IRIN=P3^2;

u8 IrValue[6];
u8 Time;

u8 DisplayData[8];
u8 code smgduan[17]={
0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,
0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71,0X76};
```

```
//0、1、2、3、4、5、6、7、8、9、A、b、C、d、E、F、H的显示码

/*******************************************************************************
* 函 数 名         : delay
* 函数功能		   : 延时函数，i=1时，大约延时10us
*******************************************************************************/
void delay(u16 i)
{
 while(i--);
}


/*******************************************************************************
* 函数名         :DigDisplay()
* 函数功能		 :数码管显示函数
* 输入           : 无
* 输出           : 无
*******************************************************************************/
void DigDisplay()
{
 u8 i;
 for(i=0;i<3;i++)
 {
  switch(i)  //位选，选择点亮的数码管，
  {
   case(0):
    LSA=1;LSB=1;LSC=1; break;//显示第0位
   case(1):
    LSA=0;LSB=1;LSC=1; break;//显示第1位
   case(2):
    LSA=1;LSB=0;LSC=1; break;//显示第2位
  }
  P0=DisplayData[i];//发送数据
  delay(100); //间隔一段时间扫描
  P0=0x00;//消隐
 }
}


/*******************************************************************************
* 函数名         : IrInit()
* 函数功能		 : 初始化红外线接收
* 输入           : 无
* 输出           : 无
*******************************************************************************/

void IrInit()
{
 IT0=1;//下降沿触发
 EX0=1;//打开中断0允许
 EA=1; //打开总中断

 IRIN=1;//初始化端口
}


/*******************************************************************************
* 函 数 名       : main
* 函数功能		 : 主函数
```

```
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/
void main()
{
 IrInit();
 while(1)
 {
  DisplayData[0] = smgduan[IrValue[2]/16];
  DisplayData[1] = smgduan[IrValue[2]%16];
  DisplayData[2] = smgduan[16];
     DigDisplay();
 }
}

/*******************************************************************************
* 函数名         : ReadIr()
* 函数功能     : 读取红外数值的中断函数
* 输入           : 无
* 输出           : 无
*******************************************************************************/

void ReadIr() interrupt 0
{
 u8 j,k;
 u16 err;
 Time=0;
 delay(700); //7ms
 if(IRIN==0)  //确认是否真的接收到正确的信号
 {

  err=1000;    //1000*10us=10ms,超过说明接收到错误的信号
  /*当两个条件都为真是循环，如果有一个条件为假的时候跳出循环，免得程序出错的时
  侯，程序死在这里*/
  while((IRIN==0)&&(err>0)) //等待前面9ms的低电平过去
  {
   delay(1);
   err--;
  }
  if(IRIN==1)   //如果正确等到9ms低电平
  {
   err=500;
   while((IRIN==1)&&(err>0))   //等待4.5ms的起始高电平过去
   {
    delay(1);
    err--;
   }
   for(k=0;k<4;k++)  //共有4组数据
   {
    for(j=0;j<8;j++) //接收一组数据
    {

     err=60;
     while((IRIN==0)&&(err>0))//等待信号前面的560us低电平过去
     {
      delay(1);
      err--;
     }
     err=500;
     while((IRIN==1)&&(err>0))  //计算高电平的时间长度。
```

```
    {
     delay(10);  //0.1ms
     Time++;
     err--;
     if(Time>30)
     {
      return;
     }
    }
    IrValue[k]>>=1;  //k表示第几组数据
    if(Time>=8)   //如果高电平出现大于565us，那么是1
    {
     IrValue[k]|=0x80;
    }
    Time=0;  //用完时间要重新赋值
   }
  }
 }
 if(IrValue[2]!=~IrValue[3])
 {
  return;
 }
}
}
```

实验20：AD模数转换

ADC（analog to digital converter）也称为模数转换器，是指一个将模拟信号转变为数字信号。STC89C5x 单片机内部不含 ADC 接口，所以需要外接一个 ADC 转换芯片将模拟信号转换成数字信号供单片机 处理。本开发板上集成了一个 ADC 模数转换电路，选用的 ADC 芯片是 12 位的 AD 芯片-XPT2046。

XPT2046 是一款 4 线制电阻式触摸屏控制器，内含 12 位分辨率 125KHz 转换速率逐步逼近型 A/D 转换器。XPT2046 支持从 1.5V 到 5.25V 的低电压 I/O 接口。XPT2046 能通过执行两次 A/D 转换查出被按的屏幕位置，除此之外， 还可以测量加在触摸屏上的压力。内部自带 2.5V 参考电压，可以作为辅助输入、 温度测量和电池监测之用，电池监测的电压范围可以从 0V 到 6V。XPT2046 片 内集成有一个温度传感器。

XPT2046 是一种典型的逐次逼近型模数转换器（SAR ADC），包含了采样/保 持、模数转换、串口数据 输出等功能。同时芯片集成有一个 2.5V 的内部参考 电压源、温度检测电路，工作时使用外部时钟。XPT2046 可以单电源供电，电源 电压范围为 2.7V～5.5V。参考电压值直接决定 ADC 的输入范围，参考电压可以 使用内部参考电压，也可以从外部直接输入 1V～VCC 范围内的参考电压（要求 外部参考电压源输出阻抗低）。X、Y、Z、VBAT、Temp 和 AUX 模拟信号经过片内 的控制寄存器选择后进入 ADC，ADC 可以配置为单端或差分模式。选择 VBAT、Temp 和 AUX 时应该配置为单端模式；作为触摸屏应用时，应该配置为差分模式，这可 有效消除由于驱动开关的寄生电阻及外部的干扰带来的测量误差，提高转换精 度。单端和差分模式输入配置如下图所示：

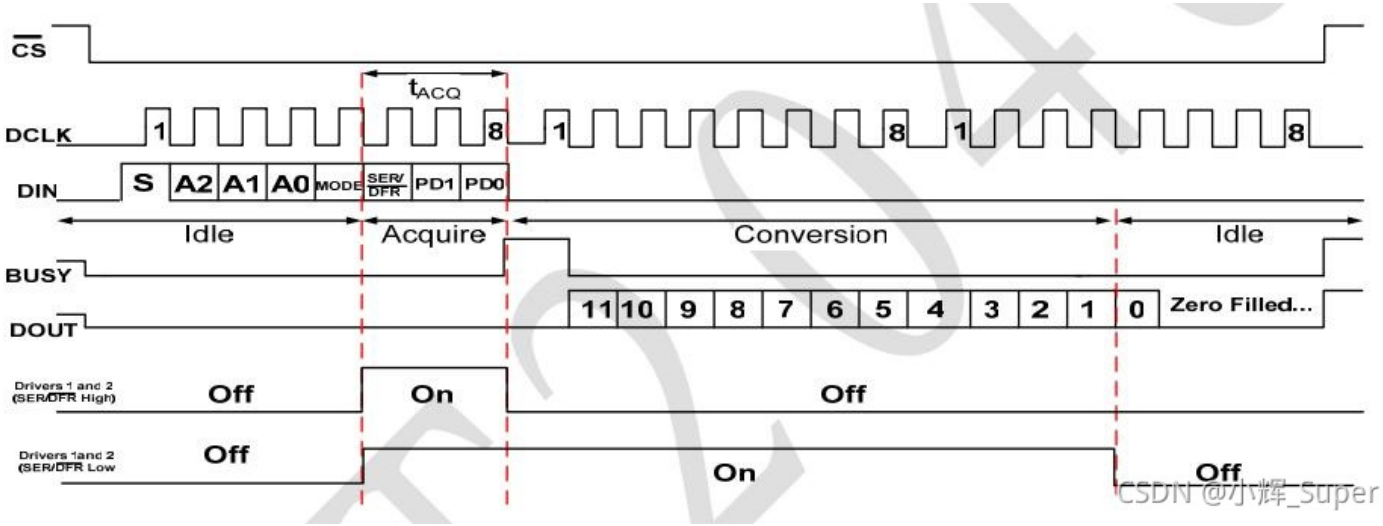
表3 单端模式输入配置（SER/$\overline{DFR}$=1）

A2	A1	A0	V_{BAT}	AUX_{IN}	TEMP	YN	XP	YP	Y-位置	X-位置	Z_1-位置	Z_2-位置	X-驱动	Y-驱动
0	0	0			+IN (TEMP0)								off	off
0	0	1					+IN		测量				Off	On
0	1	0	+IN										Off	Off
0	1	1					+IN				测量		XN, On	YP, On
1	0	0				+IN						测量	XN, On	YP, On
1	0	1						+IN		测量			On	Off
1	1	0		+IN									Off	Off
1	1	1			+IN (TEMP1)								Off	Off

表4 差分模式输入配置（SER/$\overline{DFR}$=0）

A2	A1	A0	+REF	−REF	YN	XP	YP	Y-位置	X-位置	Z_1-位置	Z_2-位置	驱动
0	0	1	YP	YN		+IN		测量				YP，YN
0	1	1	YP	XN		+IN				测量		YP，XN
1	0	0	YP	XN	+IN						测量	YP，XN
1	0	1	XP	XN			+IN		测量			

XPT2046 数据接口是串行接口，其典型工作时序如下图所示，图中展示的信 号来自带有基本串行接口的单片机或数据信号处理器。处理器和转换器之间的的 通信需要 8 个时钟周期，可采用 SPI、SSI 和 Microwire 等同步串行接口。一次完整的转换需要 24 个串行同步时钟（DCLK）来完成。 前 8 个时钟用来通过 DIN 引脚输入控制字节。当转换器获取有关下一次转 换的足够信息后，接着根据获得的信息设置输入多路选择器和参考源输入，并进入采样模式，如果需要，将启动触摸面板驱动器。3 个多时钟周期后，控制字节 设置完成，转换器进入转换状态。这时，输入采样－保持器进入保持状态，触摸面板驱动器停止工作（单端工作模式）。接着的 12 个时钟周期将完成真正的模 数转换。如果是度量比率转换方式（SER/DFR＝0），驱动器在转换过程中将一直 工作，第 13 个时钟将输出转换结果的最后一位。剩下的 3 个多时钟周期将用 来完成被转换器忽略的最后字节（DOUT 置低）。



在对 XPT2046 进行控制时，控制字节由 DIN 输入的控制字命令格式如下所示

表5 制字的控制位命令

位 7 (MSB)	位 6	位 5	位 4	位 3	位 2	位 1	位 0(LSB)
S	A2	A1	A0	MODE	SER/$\overline{\text{DFR}}$	PD1	PD0

表6 控制字节各位描述

位	名称	功能描述
7	S	开始位。为 1 表示一个新的控制字节到来，为 0 则忽略 PIN 引脚上数据
6-4	A2-A0	通道选择位。参见表 1 和表 2
3	MODE	12 位/8 位转换分辨率选择位。为 1 选择 8 位为转换分辨率，为 0 选择 12 位分辨率
2	SER/$\overline{\text{DFR}}$	单端输入方式/ 差分输入方式选择位。为 1 是单端输入方式，为 0 是差分输入方式
1-0	PD1-PD0	低功率模式选择位。若为 11，器件总处于供电状态;若为 00，器件在变换之间处于低功率模式

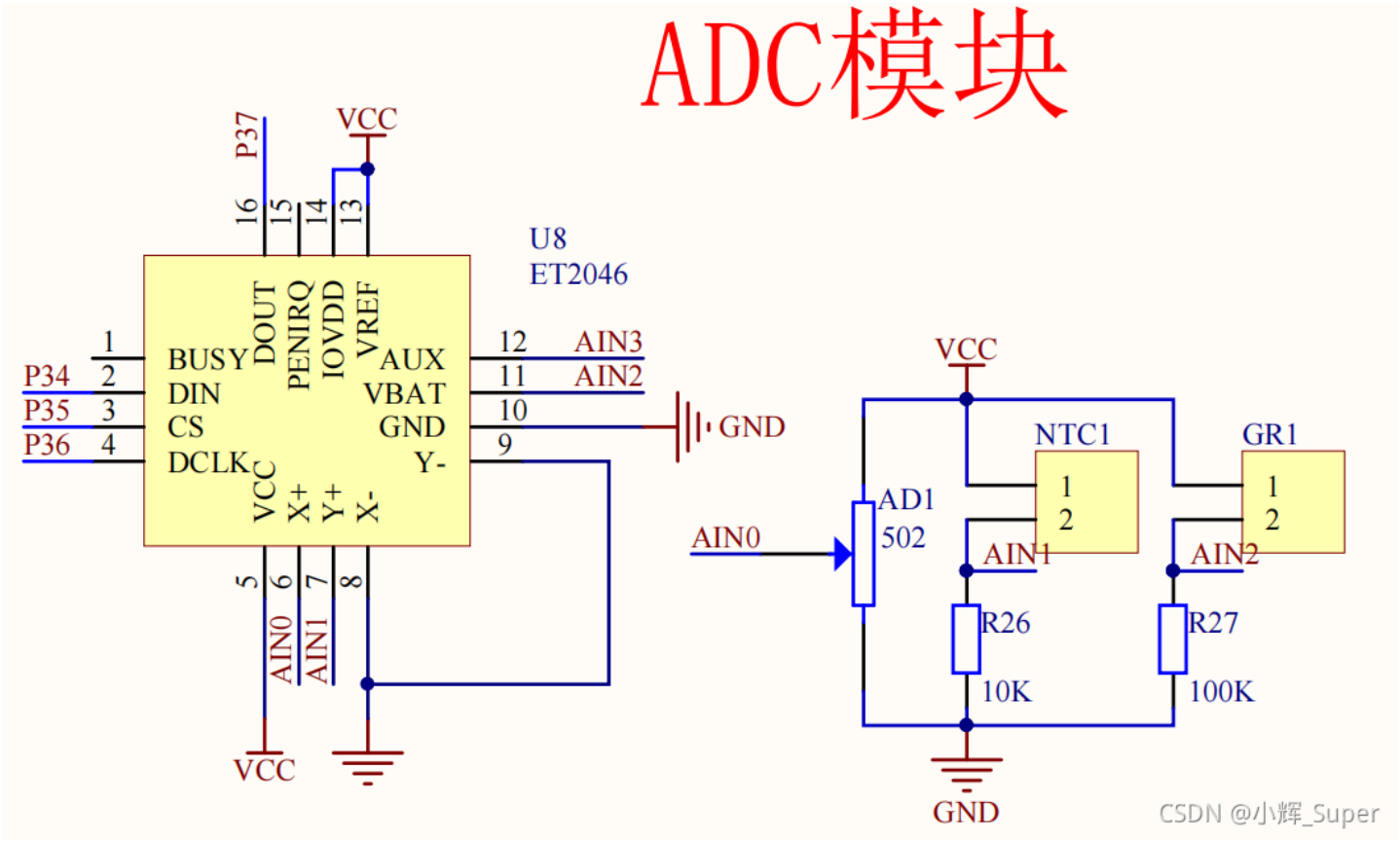
所以控制命令为1xxx 0100，6-4位用来选择输入通道。
通道为XP时，控制命令为：1001 0100 (94H)
通道为YP时，控制命令为 : 1101 0100 (D4H)
通道为AUX时，控制命令为：1110 0100 (E4H)
通道为VBAT时，控制命令为：1010 0100 (A4H)

注意：差分模式仅用于 X 坐标、Y 坐标和触摸压力的测量，其它测量要求采用单端模式。

表7 掉电和内部参考电压选择

PD1	PD0	$\overline{\text{PENIRQ}}$	功能说明
0	0	使能	在两次 A / D 转换之间掉电，下次转换一开始，芯片立即进入完全上电状态，而无需额外的延时。在这种模式下，YN 开关一直处于 ON 状态
0	1	禁止	参考电压关闭，ADC 打开
1	0	使能	参考电压打开，ADC 关闭
1	1	禁止	芯片处于上电状态，参考电压和 ADC 总是打开

下图为我所用的开发板ADC部分的原理图，AD1为电位器（可调电阻），NTC1为热敏传感器，GR1为光敏传感器，AIN3 接在 DAC（PWM）模块的 J52 端子上供外部模拟信号检测。



QFN引脚号	TSSOP引脚号	VFBGA引脚号	名称	说明
1	13	A5	BUSY	忙时信号线。当$\overline{CS}$为高电平时为高阻状态
2	14	A4	DIN	串行数据输入端。当$\overline{CS}$为低电平时，数据在DCLK上升沿锁存进来
3	15	A3	$\overline{CS}$	片选信号。控制转换时序和使能串行输入输出寄存器，高电平时ADC掉电
4	16	A2	DCLK	外部时钟信号输入
5	1	B1和C1	VCC	电源输入端
6	2	D1	XP	XP位置输入端
7	3	E1	YP	YP位置输入端
8	4	G2	XN	XN位置输入端
9	5	G3	YN	YN位置输入端
10	6	G4和G5	GND	接地
11	7	G6	V_{BAT}	电池监视输入端
12	8	E7	AUX	ADC辅助输入通道
13	9	D7	V_{REF}	参考电压输入/输出
14	10	C7	IOVDD	数字电源输入端
15	11	B7	$\overline{PENIRQ}$	笔接触中断引脚
16	12	A6	DOUT	串行数据输出端。数据在DCLK的下降沿移出，当$\overline{CS}$高电平时为高阻状态

ADC实验有4个子实验：电位器、光敏电阻、热敏电阻和外部输入。

1. 电位器AD值

XPT2046.c包含了XPT2046芯片的驱动代码：写字节、读字节和读取AD值。

XPT2046.c

```
#include"XPT2046.h"

/*****************************************************************************
*函数名：SPI_Write
*输  入：dat：写入数据
*输  出：无
*功  能：使用SPI写入数据
*****************************************************************************/

void SPI_Write(uchar dat)
{
 uchar i;
 CLK = 0;
 for(i=0; i<8; i++)
 {
  DIN = dat >> 7;   //放置最高位
  dat <<= 1;
  CLK = 0;   //上升沿放置数据

  CLK = 1;

 }
}
/*****************************************************************************
*函数名：SPI_Read
*输  入：无
*输  出：dat：读取到的数据
```

```
*输  出: dat: 读取到的数据
*功  能: 使用SPI读取数据
*****************************************************************************/

uint SPI_Read(void)
{
 uint i, dat=0;
 CLK = 0;
 for(i=0; i<12; i++)  //接收12位数据
 {
  dat <<= 1;

  CLK = 1;
  CLK = 0;

  dat |= DOUT;

 }
 return dat;
}

/*****************************************************************************
*函数名: Read_AD_Data
*输  入: cmd: 读取的X或者Y
*输  出: endValue: 最终信号处理后返回的值
*功  能: 读取触摸数据
*****************************************************************************/
uint Read_AD_Data(uchar cmd)
{
 uchar i;
 uint AD_Value;
 CLK = 0;
 CS  = 0;
 SPI_Write(cmd);
 for(i=6; i>0; i--);  //延时等待转换结果
 CLK = 1;   //发送一个时钟周期, 清除BUSY
 _nop_();
 _nop_();
 CLK = 0;
 _nop_();
 _nop_();
 AD_Value=SPI_Read();
 CS = 1;
 return AD_Value;
}
```

XPT2046.H

```
#ifndef   __XPT2046_H_
#define   __XPT2046_H_

//---包含头文件---//
#include<reg52.h>
#include<intrins.h>

//---重定义关键词---//
#ifndef uchar
#define uchar unsigned char
#endif

#ifndef uint
#define uint  unsigned int
#endif

#ifndef ulong
#define ulong  unsigned long
#endif

//---定义使用的IO口---//
sbit DOUT = P3^7;   //输出
sbit CLK  = P3^6;   //时钟
sbit DIN  = P3^4;   //输入
sbit CS   = P3^5;   //片选

uint Read_AD_Data(uchar cmd);
uint SPI_Read(void);
void SPI_Write(uchar dat);

#endif
```

main.c的功能为：轮询读取电位器AD值，并将其千位-个位显示在4个数码管上。由于XPT2046是12位的，所以读取的AD值范围为0-4095。原理图中，电位器AIN0与XPT2046的X+相连，使用Read_AD_Data函数时需要填的地址为0x94（见上文——控制字命令格式）。

main.c

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

*
实验现象：下载程序后数码管前4位显示电位器检测的AD值，范围是0-4095，一般达不到最大，这个受
   供电电压的影响

接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）
     1，单片机-->AD/DAC模块
       P34-->DI
    P35-->CS
    P36-->CL
    P37-->DO
     2，单片机-->动态数码管模块
       J22-->J6
    P22-->J9(A)
    P23-->J9(B)
    P24-->J9(C)

注意事项：
```

```
*******************************************************************************

*/

#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器
#include"XPT2046.h"

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;

sbit LSA=P2^2;
sbit LSB=P2^3;
sbit LSC=P2^4;

u8 disp[4];
u8 code smgduan[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};

/*******************************************************************************
* 函 数 名         : delay
* 函数功能     : 延时函数，i=1时，大约延时10us
*******************************************************************************/
void delay(u16 i)
{
 while(i--);
}

/*******************************************************************************
* 函数名         :datapros()
* 函数功能   :数据处理函数
* 输入           : 无
* 输出           : 无
*******************************************************************************/
void datapros()
{
 u16 temp;
 static u8 i;
 if(i==50)
 {
  i=0;
  temp = Read_AD_Data(0x94);  //   AIN0 电位器
 }
 i++;
 disp[0]=smgduan[temp/1000];//千位
 disp[1]=smgduan[temp%1000/100];//百位
 disp[2]=smgduan[temp%1000%100/10];//十位
 disp[3]=smgduan[temp%1000%100%10];
}


/*******************************************************************************
* 函数名         :DigDisplay()
* 函数功能   :数码管显示函数
* 输入           : 无
* 输出           : 无
*******************************************************************************/
void DigDisplay()
{
 u8 i;
 for(i=0;i<4;i++)
```

```
 {
  switch(i)  //位选，选择点亮的数码管，
  {
   case(0):
    LSA=1;LSB=1;LSC=1; break;//显示第0位
   case(1):
    LSA=0;LSB=1;LSC=1; break;//显示第1位
   case(2):
    LSA=1;LSB=0;LSC=1; break;//显示第2位
   case(3):
    LSA=0;LSB=0;LSC=1; break;//显示第3位
  }
  P0=disp[i];//发送数据
  delay(100); //间隔一段时间扫描
  P0=0x00;//消隐
 }
}

/*******************************************************************************
* 函 数 名         : main
* 函数功能   : 主函数
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/
void main()
{
 while(1)
 {
  datapros();  //数据处理函数
  DigDisplay();//数码管显示函数
 }
}
```

2. 光敏电阻AD值

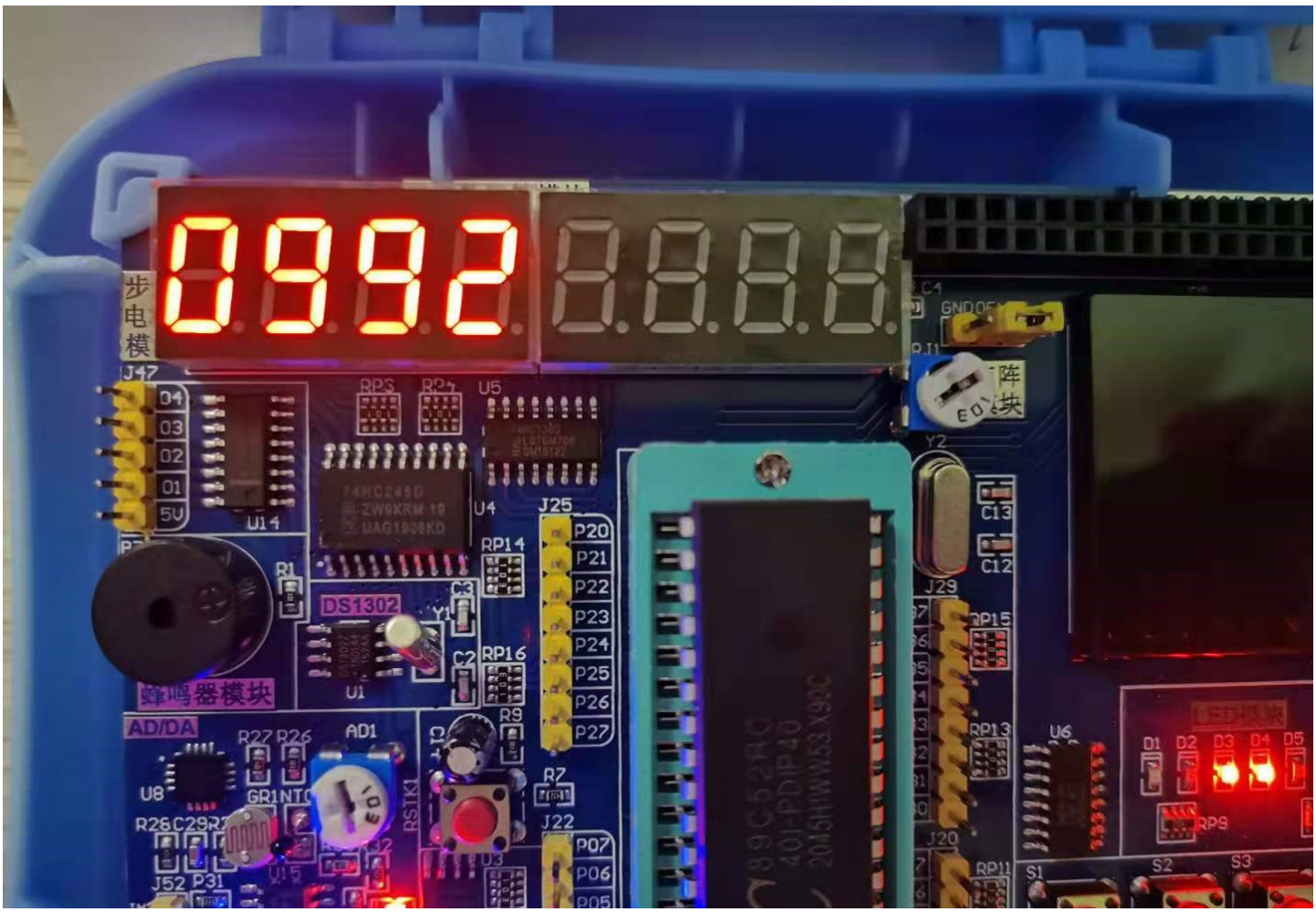
读取光敏电阻AD值的程序和读取电位器AD值的程序逻辑一致，只需将Read_AD_Data的地址改成0xA4（见上文——控制字命令格式）即可读取光敏电阻的AD值。

```
/*******************************************************************************************
实验现象：下载程序后数码管前4位显示光敏传感器检测的AD值

接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）
     1，单片机-->AD/DAC模块
       P34-->DI
    P35-->CS
    P36-->CL
    P37-->DO
     2，单片机-->动态数码管模块
       J22-->J6
    P22-->J9(A)
    P23-->J9(B)
    P24-->J9(C)

注意事项：
*******************************************************************************************/

#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器
#include"XPT2046.h"
```

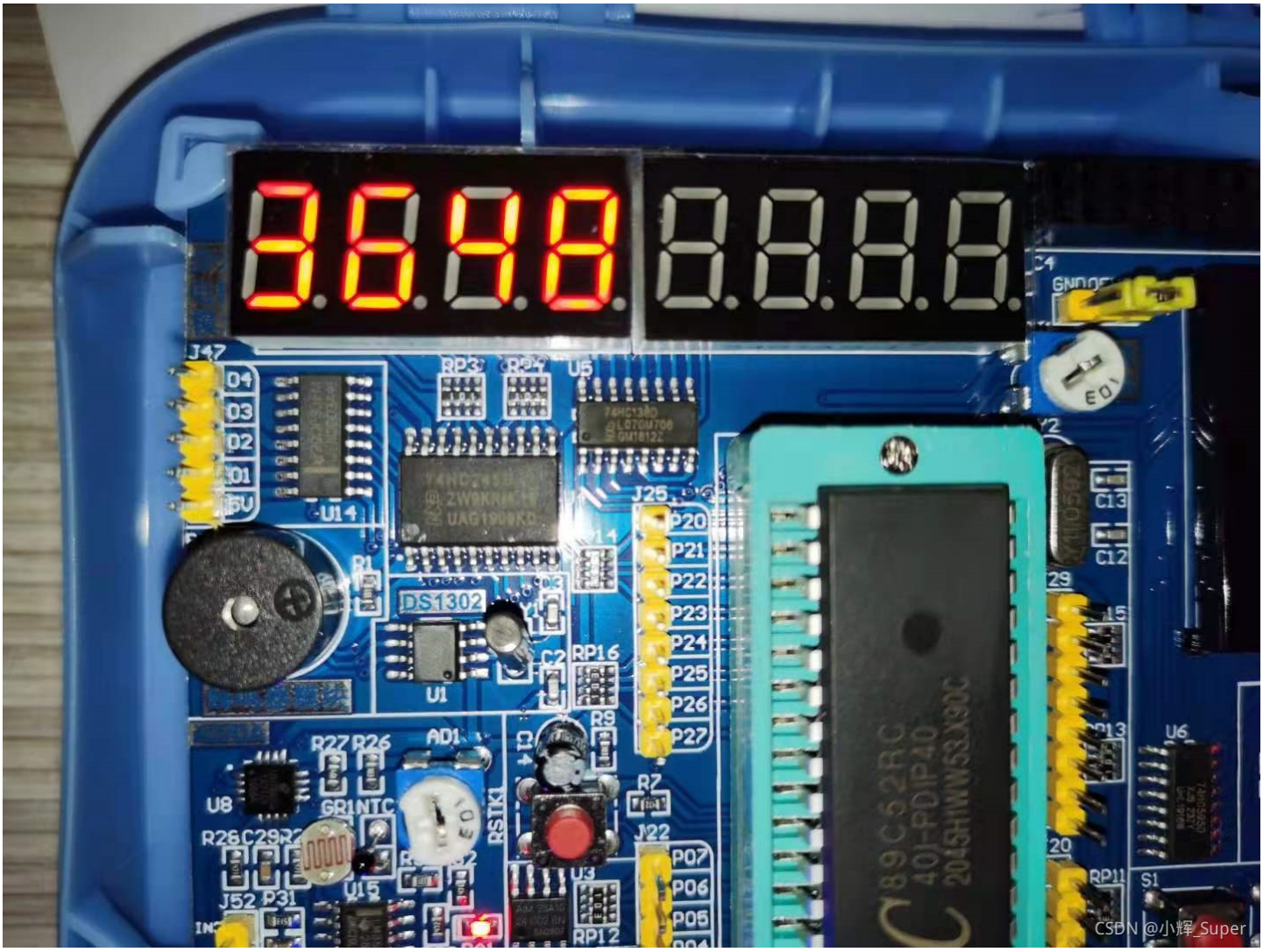
```
#include "xpt2046.h"

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;

sbit LSA=P2^2;
sbit LSB=P2^3;
sbit LSC=P2^4;

u8 disp[4];
u8 code smgduan[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};

/*******************************************************************************
* 函 数 名         : delay
* 函数功能     : 延时函数，i=1时，大约延时10us
*******************************************************************************/
void delay(u16 i)
{
 while(i--);
}

/*******************************************************************************
* 函数名         :datapros()
* 函数功能   :数据处理函数
* 输入           : 无
* 输出           : 无
*******************************************************************************/
void datapros()
{
 u16 temp;
 static u8 i;
 if(i==50)
 {
  i=0;
  temp = Read_AD_Data(0xA4);  //   AIN2 光敏电阻
 }
 i++;
 disp[0]=smgduan[temp/1000];//千位
 disp[1]=smgduan[temp%1000/100];//百位
 disp[2]=smgduan[temp%1000%100/10];//十位
 disp[3]=smgduan[temp%1000%100%10];
}


/*******************************************************************************
* 函数名         :DigDisplay()
* 函数功能   :数码管显示函数
* 输入           : 无
* 输出           : 无
*******************************************************************************/
void DigDisplay()
{
 u8 i;
 for(i=0;i<4;i++)
 {
  switch(i)  //位选，选择点亮的数码管，
  {
  case(0):
    LSA=1;LSB=1;LSC=1; break;//显示第0位
   case(1):
```

```
    LSA=0;LSB=1;LSC=1; break;//显示第1位
   case(2):
    LSA=1;LSB=0;LSC=1; break;//显示第2位
   case(3):
    LSA=0;LSB=0;LSC=1; break;//显示第3位
  }
  P0=disp[i];//发送数据
  delay(100); //间隔一段时间扫描
  P0=0x00;//消隐
 }
}

/*******************************************************************************
* 函 数 名       : main
* 函数功能   : 主函数
* 输    入       : 无
* 输    出       : 无
*******************************************************************************/
void main()
{
 while(1)
 {
  datapros();  //数据处理函数
  DigDisplay();//数码管显示函数
 }
}
```

亮度低时：



亮度高时：



3. 热敏电阻AD值

读取光敏电阻AD值的程序和读取电位器AD值的程序逻辑一致，只需将Read_AD_Data的地址改成0xD4（见上文——控制字命令格式）即可读取热敏电阻的AD值。

```
/*********************************************************************************************
实验现象：下载程序后数码管前4位显示热敏传感器检测的AD值

接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）
     1，单片机-->AD/DAC模块
       P34-->DI
    P35-->CS
    P36-->CL
    P37-->DO
     2，单片机-->动态数码管模块
       J22-->J6
    P22-->J9(A)
    P23-->J9(B)
    P24-->J9(C)

注意事项
```

```
注意事项:
*******************************************************************************/

#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器
#include"XPT2046.h"

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;

sbit LSA=P2^2;
sbit LSB=P2^3;
sbit LSC=P2^4;

u8 disp[4];
u8 code smgduan[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};

/*******************************************************************************
* 函 数 名         : delay
* 函数功能     : 延时函数，i=1时，大约延时10us
*******************************************************************************/
void delay(u16 i)
{
 while(i--);
}

/*******************************************************************************
* 函数名         :datapros()
* 函数功能   :数据处理函数
* 输入           : 无
* 输出           : 无
*******************************************************************************/
void datapros()
{
 u16 temp;
 static u8 i;
 if(i==50)
 {
  i=0;
  temp = Read_AD_Data(0xD4);  //   AIN1 热敏电阻
 }
 i++;
 disp[0]=smgduan[temp/1000];//千位
 disp[1]=smgduan[temp%1000/100];//百位
 disp[2]=smgduan[temp%1000%100/10];
 disp[3]=smgduan[temp%1000%100%10];  //个位
}


/*******************************************************************************
* 函数名         :DigDisplay()
* 函数功能   :数码管显示函数
* 输入           : 无
* 输出           : 无
*******************************************************************************/
void DigDisplay()
{
 u8 i;
 for(i=0;i<4;i++)
 {
  switch(i)  //位选，选择点亮的数码管，
```

```
  {
    case(0):
    LSA=1;LSB=1;LSC=1; break;//显示第0位
   case(1):
    LSA=0;LSB=1;LSC=1; break;//显示第1位
   case(2):
    LSA=1;LSB=0;LSC=1; break;//显示第2位
   case(3):
    LSA=0;LSB=0;LSC=1; break;//显示第3位
  }
  P0=disp[i];//发送数据
  delay(100); //间隔一段时间扫描
  P0=0x00;//消隐
 }
}

/*******************************************************************************
* 函 数 名       : main
* 函数功能   : 主函数
* 输    入       : 无
* 输    出       : 无
*******************************************************************************/
void main()
{
 while(1)
 {
  datapros();  //数据处理函数
  DigDisplay();//数码管显示函数
 }
}
```

4.外部输入AD值

读取光敏电阻AD值的程序和读取电位器AD值的程序逻辑一致，只需将Read_AD_Data的地址改成0xE4（见上文——控制字命令格式）即可读取外部输入的AD值。

```
/*******************************************************************************************
实验现象：下载程序后数码管前4位显示外部输入AIN3通道检测的AD值，模拟信号电压范围在0-5V

接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）
     1，单片机-->AD/DAC模块
       P34-->DI
    P35-->CS
    P36-->CL
    P37-->DO
     2，单片机-->动态数码管模块
       J22-->J6
    P22-->J9(A)
    P23-->J9(B)
    P24-->J9(C)

注意事项：
*******************************************************************************************/

#include "reg52.h"     //此文件中定义了单片机的一些特殊功能寄存器
#include"XPT2046.h"
```

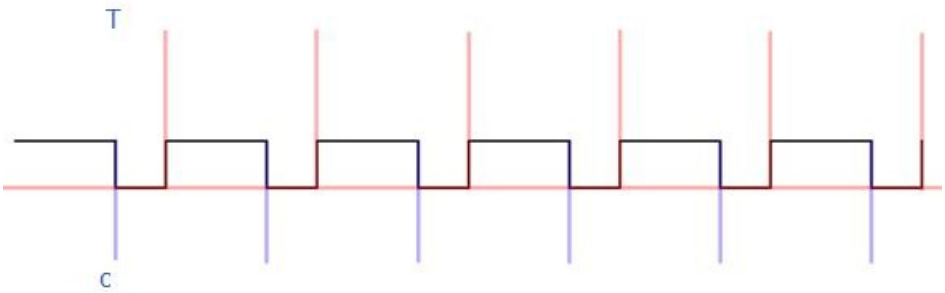
```c
typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;

sbit LSA=P2^2;
sbit LSB=P2^3;
sbit LSC=P2^4;

u8 disp[4];
u8 code smgduan[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};

/*******************************************************************************
* 函 数 名         : delay
* 函数功能     : 延时函数，i=1时，大约延时10us
*******************************************************************************/
void delay(u16 i)
{
 while(i--);
}

/*******************************************************************************
* 函数名         :datapros()
* 函数功能   :数据处理函数
* 输入           : 无
* 输出           : 无
*******************************************************************************/
void datapros()
{
 u16 temp;
 static u8 i;
 if(i==50)
 {
  i=0;
  temp = Read_AD_Data(0xE4);  //   AIN3 外部输入
 }
 i++;
 disp[0]=smgduan[temp/1000];//千位
 disp[1]=smgduan[temp%1000/100];//百位
 disp[2]=smgduan[temp%1000%100/10];
 disp[3]=smgduan[temp%1000%100%10]; //个位
}


/*******************************************************************************
* 函数名         :DigDisplay()
* 函数功能   :数码管显示函数
* 输入           : 无
* 输出           : 无
*******************************************************************************/
void DigDisplay()
{
 u8 i;
 for(i=0;i<4;i++)
 {
  switch(i)  //位选，选择点亮的数码管，
  {
   case(0):
    LSA=1;LSB=1;LSC=1; break;//显示第0位
   case(1):
    LSA=0;LSB=1;LSC=1; break;//显示第1位
   case(2):
```

```
    LSA=1;LSB=0;LSC=1; break;//显示第2位
   case(3):
    LSA=0;LSB=0;LSC=1; break;//显示第3位
  }
  P0=disp[i];//发送数据
  delay(100); //间隔一段时间扫描
  P0=0x00;//消隐
 }
}

/*******************************************************************************
* 函 数 名       : main
* 函数功能    : 主函数
* 输    入       : 无
* 输    出       : 无
*******************************************************************************/
void main()
{
 while(1)
 {
  datapros();  //数据处理函数
  DigDisplay();//数码管显示函数
 }
}
```

实验21：DA数模转换

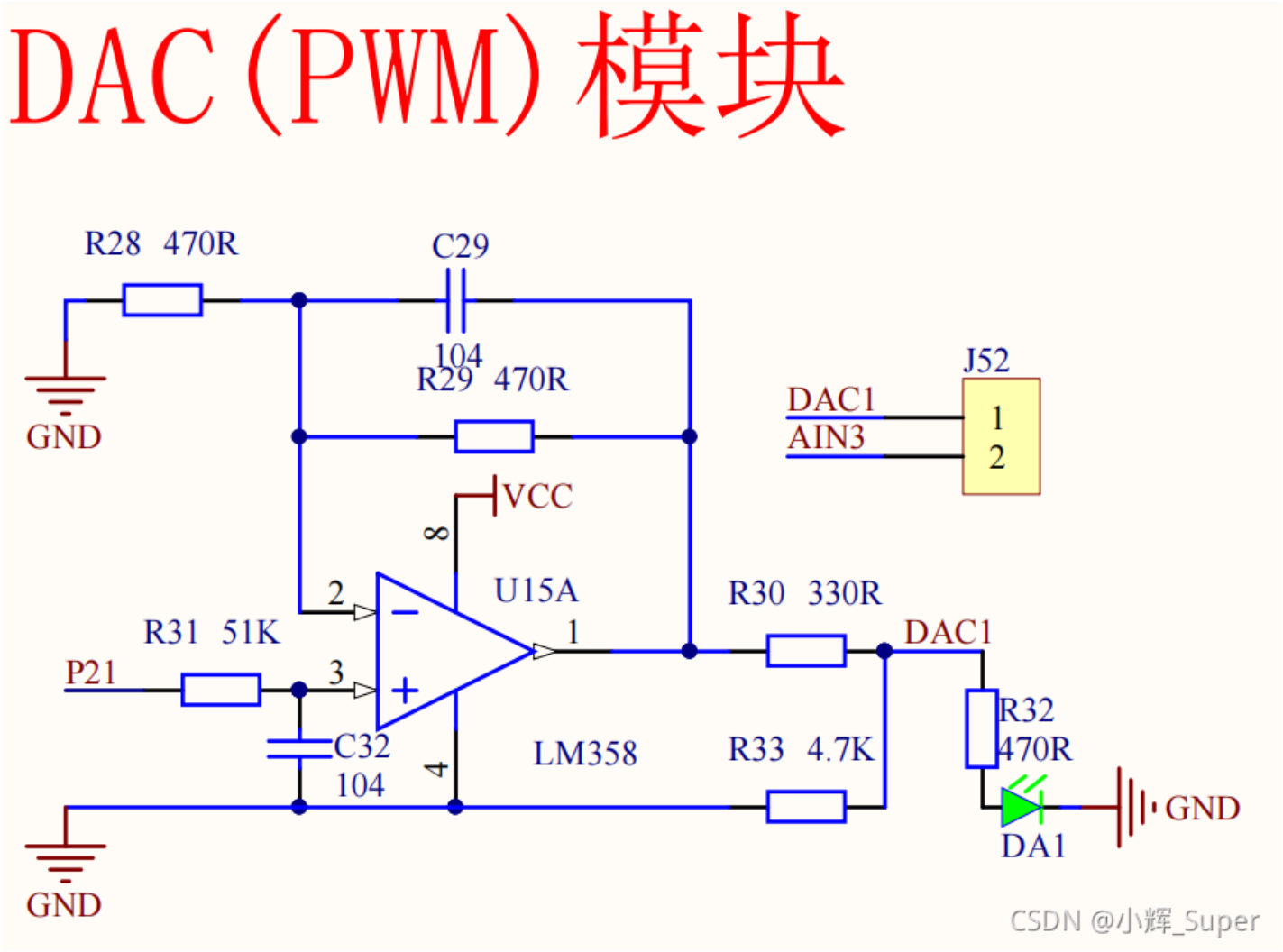
在实际开发应用中，多数使用 PWM 来模拟 DAC 输 出，PWM 是一种对模拟信号电平进行数字编码的方法。通过高分辨率计数器的使用，方波的占空比被调制用来对一个具体模拟信号的电平进行编码。PWM 信号仍然是数字的，因为在给定的任何时刻，满幅值的直流供电要么完全有(ON)，要么完全无(OFF)。电压或电流源是以一种通(ON)或断 (OFF)的重复脉冲序列被加到模拟负载上去的。通的时候即是直流供电被加到负载上的时候，断的时候即是供电被断开的时候。只要带宽足够，任何模拟值都可以使用 PWM 进行编码。

PWM 的输出其实就是对外输出脉宽可调（即占空比调节）的方波信号，信号 频率是由 T 的值决定，占空比由 C 的值决定。其



示意图如图所示：
从上图中可以看到，PWM 输出频率是不变的，改变的是 C 的值，此值的改变 将导致 PWM 输出信号占空比的改变。占空比其实就是一个周期内高电平时间与 周期的比值。而频率的话可以使用 51 单片机的定时器确定。

由原理图可知，PWM 输出控制管脚接至单片机 P21 管 脚上，DAC1 为 PWM 输出信号，将其连接一个 LED，这样可以通过指示灯的状态直观的反映出 PWM 输出电压值变化。



代码中，定时器平均每1us进一次中断，将timer和count变量加1。main函数里，产生了周期为T（1000us，timer的最大值），占空比为value的PWM信号，如果占空比value为固定值，则LED的亮度就为固定值，只有value在0-T间变化，才能动态调整LED亮度使其形成呼吸灯效果。main中的count作用是控制呼吸灯的呼吸速度。

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

*
实验现象：下载程序后AD/DAC模块上的DA1指示灯呈呼吸灯效果，由暗变亮再由亮变暗

接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）
    1，单片机-->AD/DAC模块
      P21-->J50(PWM)

注意事项：


*********************************************************************************

*/

#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;
```

```
//--定义使用的IO口--//
sbit PWM=P2^1;
bit DIR;

//--定义一个全局变量--//
u16 count,value,timer1;

/*******************************************************************************
* 函 数 名         : Timer1Init
* 函数功能		   : 定时器1初始化
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/
void Timer1Init()
{
 TMOD|=0X10;//选择为定时器1模式，工作方式1，仅用TR1打开启动。

 TH1 = 0xFF;
 TL1 = 0xff;   //1us

 ET1=1;//打开定时器1中断允许
 EA=1;//打开总中断
 TR1=1;//打开定时器
}



/*******************************************************************************
* 函 数 名       : main
* 函数功能		 : 主函数
* 输    入       : 无
* 输    出       : 无
*******************************************************************************/
void main()
{
 Timer1Init();  //定时器1初始化
 while(1)
 {
  if(count>100)
  {
   count=0;
   if(DIR==1)        //DIR控制增加或减小
   {
     value++;
   }
   if(DIR==0)
   {
     value--;
   }

  }

  if(value==1000)
  {
    DIR=0;
  }
  if(value==0)
  {
    DIR=1;
```

```
  }

  if(timer1>1000)  //PWM周期为1000*1us
  {
   timer1=0;
  }
  if(timer1 <value)
  {
   PWM=1;
  }
  else
  {
   PWM=0;
  }
 }
}

/********************************************************************************
* 函 数 名         : Time1
* 函数功能   : 定时器1的中断函数
* 输    入         : 无
* 输    出         : 无
********************************************************************************/

void Time1(void) interrupt 3    //3 为定时器1的中断号  1 定时器0的中断号 0 外部中断1 2 外部中断2  4 串口中断
{
 TH1 = 0xFF;
 TL1 = 0xff;   //1us
 timer1++;
    count++;

}
```

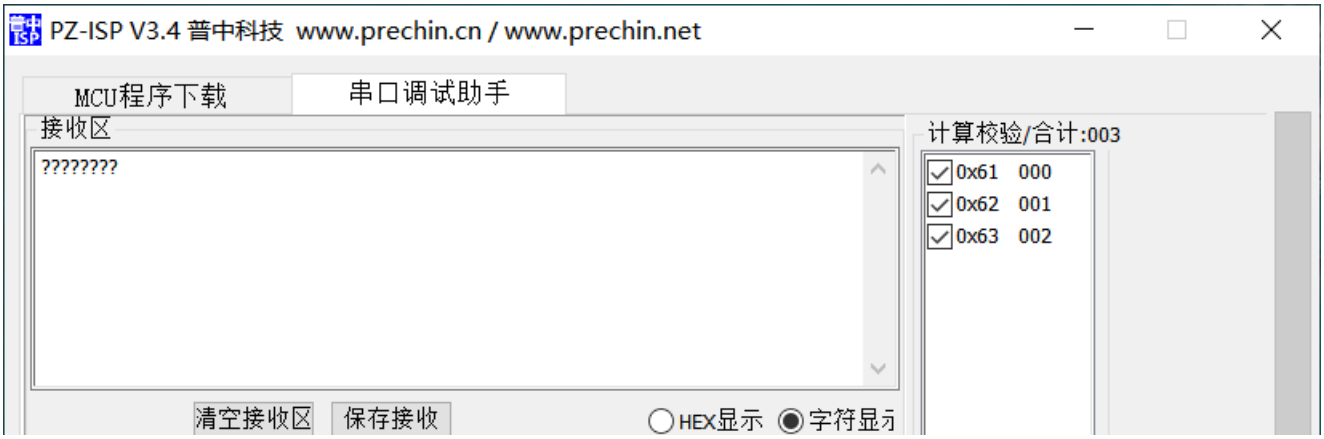
实验21：串口通信

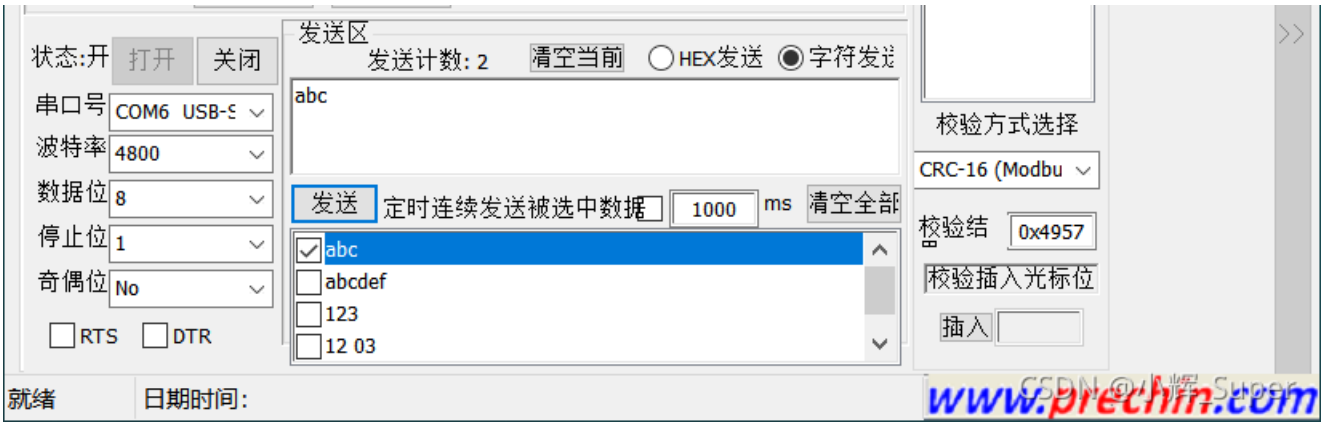
串行接口 （Serial Interface）是指数据一位一位地顺序传送。其特点是通信线路简单，只要一对传输线就可以实现双向通信。实验代码主要包括UasrtInit和串口中断子函数。UasrtInit为串口的初始化，虽然看起来很简单，但我能理解的也就波特率和中断开启了，波特率设置为4800；中断服务函数的中断号为4，receiveData=SBUF;和SBUF=receiveData;看起来挺有趣。

```
/********************************************************************************************
实验现象：下载程序后打开串口调试助手，将波特率设置为4800，选择发送的数据就可以显示

接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）

注意事项：


*********************************************************************************************/

#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;


/*******************************************************************************
* 函数名         :UsartInit()
```

```
* 函数功能      :设置串口
* 输入           : 无
* 输出           : 无
*******************************************************************************/
void UsartInit()
{
 SCON=0X50;   //设置为工作方式1
 TMOD=0X20;   //设置计数器工作方式2
 PCON=0X80;   //波特率加倍
 TH1=0XF3;    //计数器初始值设置，注意波特率是4800的
 TL1=0XF3;
 ES=1;       //打开接收中断
 EA=1;       //打开总中断
 TR1=1;      //打开计数器
}

/*******************************************************************************
* 函 数 名         : main
* 函数功能   : 主函数
* 输    入         : 无
* 输    出         : 无
*******************************************************************************/
void main()
{
 UsartInit();  // 串口初始化
 while(1);
}

/*******************************************************************************
* 函数名          : Usart() interrupt 4
* 函数功能     : 串口通信中断函数
* 输入           : 无
* 输出           : 无
*******************************************************************************/
void Usart() interrupt 4
{
 u8 receiveData;

 receiveData=SBUF;//出去接收到的数据
 RI = 0;//清除接收中断标志位
 SBUF=receiveData;//将接收到的数据放入到发送寄存器
 while(!TI);    //等待发送数据完成
 TI=0;       //清除发送完成标志位
}
```

实验结果：

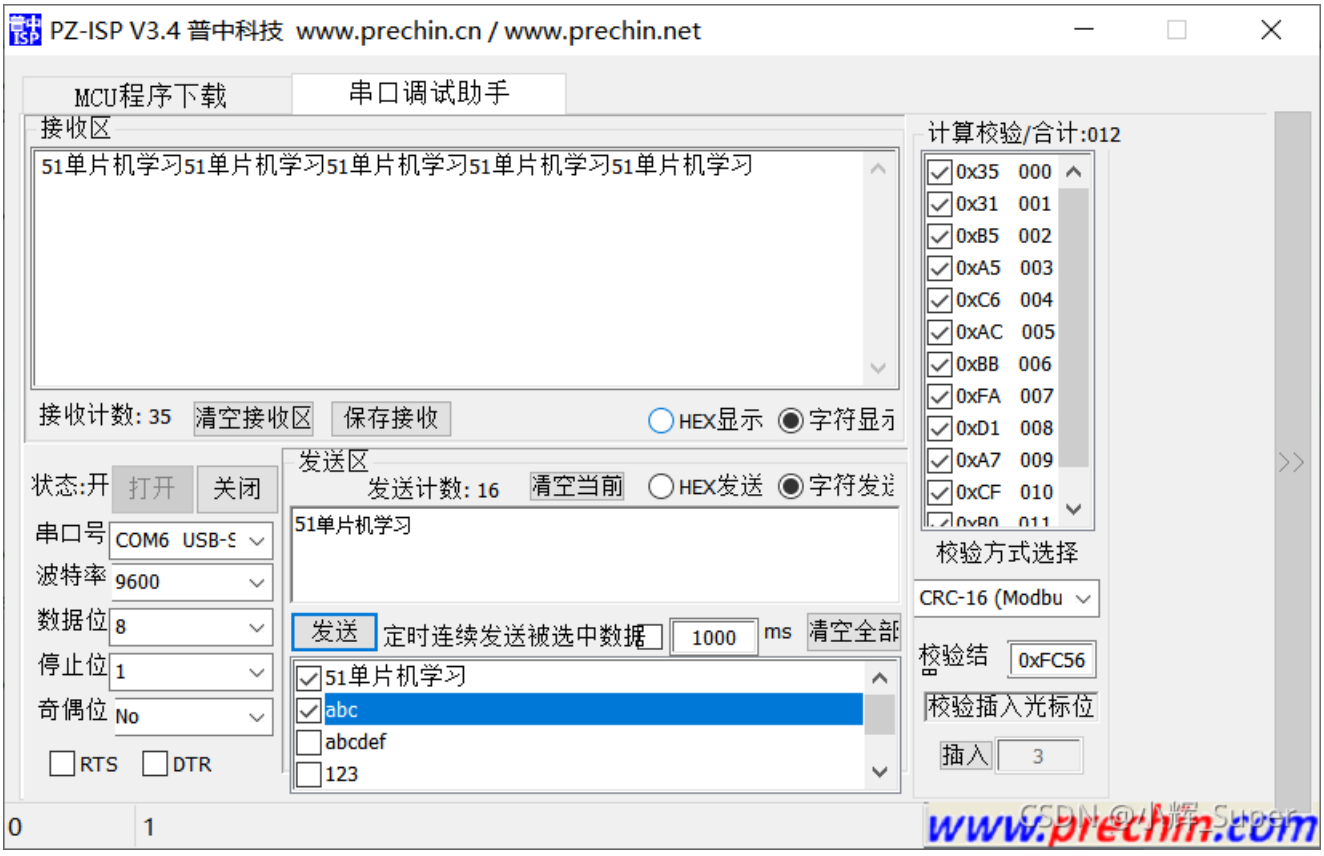




串口接收到的数据全是乱码，我试着降低波特率，乱码依然没解决，但当我把波特率设成9600时，竟然可以用了，真奇怪。

```
void UsartInit()
{
    SCON=0X50;
    TMOD=0X20;
    PCON=0X80;
    TH1=(256-6);
    TL1=(256-6);
    ES=1;
    EA=1;
    TR1=1;
}
```

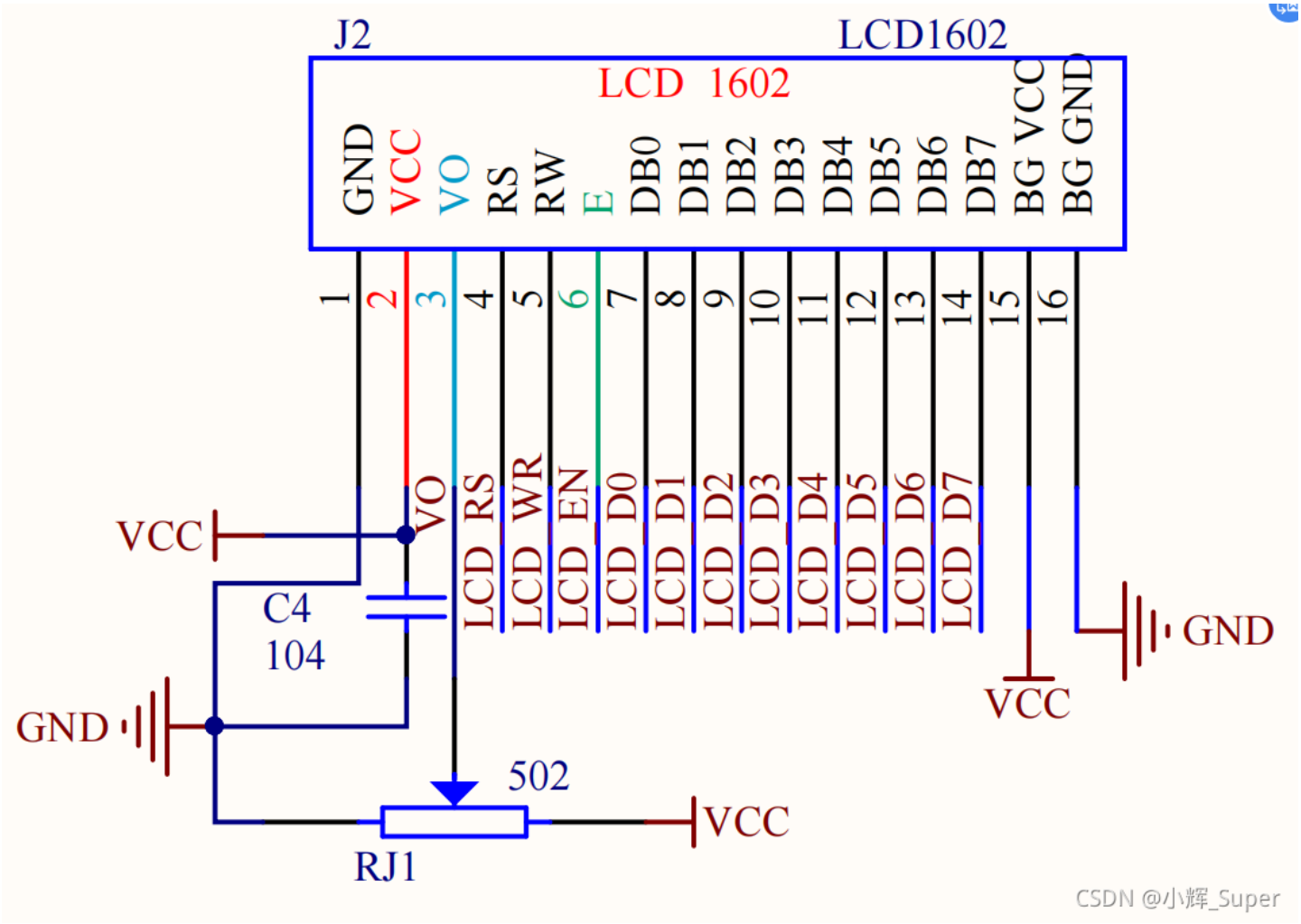
9600波特率测试：



实验22：LCD1602液晶

1602 液晶也叫 1602 字符型液晶，它能显示 2 行字符信息，每行又能显示 16 个字符。它是一种专门用来显示字母、数字、符号的点阵型液晶模块。它是由若 干个 5x7 或者 5x10 的点阵字符位组成，每个点阵字符位都可以用显示一个字符， 每位之间有一个点距的间隔，每行之间也有间隔，起到了字符间距和行间距的作 用，正因为如此，所以它不能很好的显示图片。

对于这个屏幕的细节，就不过多描述了，直接看要怎么使用吧：



要使用 LCD1602，首先需要对其初始化，即通过写入一些特定的指令实现。 然后选择要在 LCD1602 的哪个位置显示并将所要显示的数据发送到 LCD 的 DDRAM。使用 LCD1602 通常都是用于写数据进去，很少使用读功能。LCD1602 操 作步骤如下所示：

（1）初始化

（2）写命令（RS=L），设置显示坐标

（3）写数据（RS=H）

在此，不需要读出它的数据的状态或者数据本身。所以只需要看两个写时序：

① 当要写指令字，设置 LCD1602 的工作方式时：需要把 RS 置为低电平，RW 置为低电平，然后将数据送到数据口 D0~D7，最后 E 引脚一个高脉冲将数据写入。

② 当要写入数据字，在 1602 上实现显示时：需要把 RS 置为高电平，RW 置 为低电平，然后将数据送到数据口 D0~D7，最后 E 引脚一个高脉冲将数据写入。 写指令和写数据，差别仅仅在于 RS 的电平不一样而已。

————以上内容摘抄自《普中51单片机开发攻略–A2》

lcd.c中主要函数包括LcdInit、LcdWriteCom和LcdWriteData。LcdInit主要是给LCD发送一些初始化命令，让其能正常显示；LcdWriteCom用来写入命令；LcdWriteData用来写入数据。

lcd.c

```
#include "lcd.h"
```

```
/*******************************************************************************
* 函 数 名         : Lcd1602_Delay1ms
* 函数功能		   : 延时函数，延时1ms
* 输    入         : c
* 输    出         : 无
* 说    名         : 该函数是在12MHZ晶振下，12分频单片机的延时。
*******************************************************************************/

void Lcd1602_Delay1ms(uint c)   //误差 0us
{
    uchar a,b;
 for (; c>0; c--)
 {
   for (b=199;b>0;b--)
   {
     for(a=1;a>0;a--);
   }
 }

}

/*******************************************************************************
* 函 数 名         : LcdWriteCom
* 函数功能		   : 向LCD写入一个字节的命令
* 输    入         : com
* 输    出         : 无
*******************************************************************************/
#ifndef  LCD1602_4PINS  //当没有定义这个LCD1602_4PINS时
void LcdWriteCom(uchar com)   //写入命令
{
 LCD1602_E = 0;     //使能
 LCD1602_RS = 0;    //选择发送命令
 LCD1602_RW = 0;    //选择写入

 LCD1602_DATAPINS = com;     //放入命令
 Lcd1602_Delay1ms(1);  //等待数据稳定

 LCD1602_E = 1;           //写入时序
 Lcd1602_Delay1ms(5);   //保持时间
 LCD1602_E = 0;
}
#else
void LcdWriteCom(uchar com)   //写入命令
{
 LCD1602_E = 0;  //使能清零
 LCD1602_RS = 0;  //选择写入命令
 LCD1602_RW = 0;  //选择写入

 LCD1602_DATAPINS = com; //由于4位的接线是接到P0口的高四位，所以传送高四位不用改
 Lcd1602_Delay1ms(1);

 LCD1602_E = 1;  //写入时序
 Lcd1602_Delay1ms(5);
 LCD1602_E = 0;

 LCD1602_DATAPINS = com << 4; //发送低四位
 Lcd1602_Delay1ms(1);

 LCD1602_E = 1;  //写入时序
```

```
 Lcd1602_Delay1ms(5);
 LCD1602_E = 0;
}
#endif
/*******************************************************************************
* 函 数 名         : LcdWriteData
* 函数功能		   : 向LCD写入一个字节的数据
* 输    入         : dat
* 输    出         : 无
*******************************************************************************/
#ifndef  LCD1602_4PINS
void LcdWriteData(uchar dat)   //写入数据
{
 LCD1602_E = 0; //使能清零
 LCD1602_RS = 1; //选择输入数据
 LCD1602_RW = 0; //选择写入

 LCD1602_DATAPINS = dat; //写入数据
 Lcd1602_Delay1ms(1);

 LCD1602_E = 1;   //写入时序
 Lcd1602_Delay1ms(5);   //保持时间
 LCD1602_E = 0;
}
#else
void LcdWriteData(uchar dat)   //写入数据
{
 LCD1602_E = 0;   //使能清零
 LCD1602_RS = 1;   //选择写入数据
 LCD1602_RW = 0;   //选择写入

 LCD1602_DATAPINS = dat; //由于4位的接线是接到P0口的高四位，所以传送高四位不用改
 Lcd1602_Delay1ms(1);

 LCD1602_E = 1;   //写入时序
 Lcd1602_Delay1ms(5);
 LCD1602_E = 0;

 LCD1602_DATAPINS = dat << 4; //写入低四位
 Lcd1602_Delay1ms(1);

 LCD1602_E = 1;   //写入时序
 Lcd1602_Delay1ms(5);
 LCD1602_E = 0;
}
#endif
/*******************************************************************************
* 函 数 名       : LcdInit()
* 函数功能		   : 初始化LCD屏
* 输    入       : 无
* 输    出       : 无
*******************************************************************************/
#ifndef  LCD1602_4PINS
void LcdInit()        //LCD初始化子程序
{
  LcdWriteCom(0x38);  //开显示
 LcdWriteCom(0x0c);  //开显示不显示光标
 LcdWriteCom(0x06);  //写一个指针加1
 LcdWriteCom(0x01);  //清屏
 LcdWriteCom(0x80);  //设置数据指针起点
```

```
}
#else
void LcdInit()        //LCD初始化子程序
{
 LcdWriteCom(0x32);  //将8位总线转为4位总线
 LcdWriteCom(0x28);  //在四位线下的初始化
 LcdWriteCom(0x0c);  //开显示不显示光标
 LcdWriteCom(0x06);  //写一个指针加1
 LcdWriteCom(0x01);  //清屏
 LcdWriteCom(0x80);  //设置数据指针起点
}
#endif
```

LCD.H

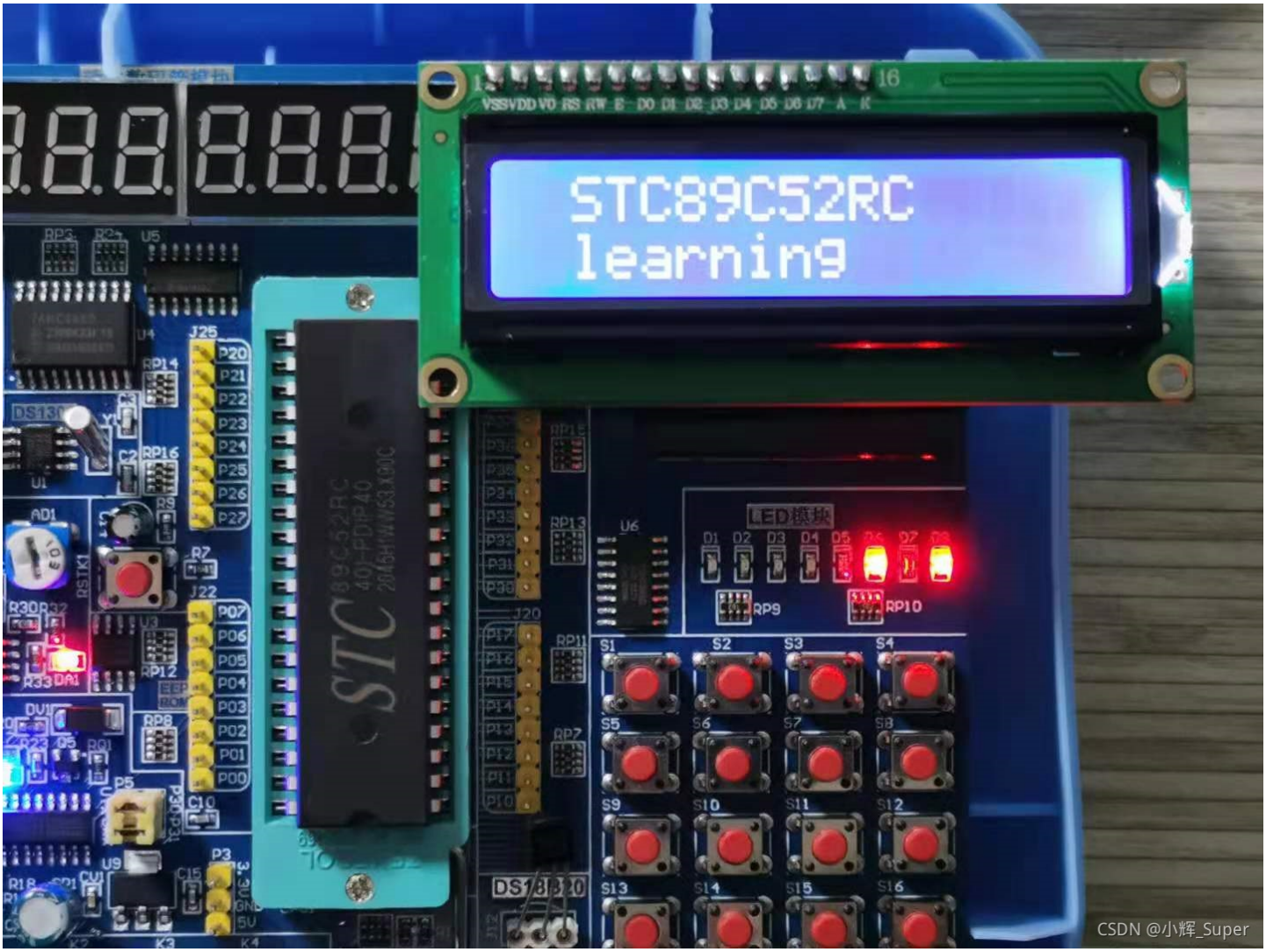
```
#ifndef __LCD_H_
#define __LCD_H_
/*********************************
当使用的是4位数据传输的时候定义，
使用8位取消这个定义
*********************************/
//#define LCD1602_4PINS

/*********************************
包含头文件
*********************************/
#include<reg52.h>

//---重定义关键词---//
#ifndef uchar
#define uchar unsigned char
#endif

#ifndef uint
#define uint unsigned int
#endif

/*********************************
PIN口定义
*********************************/
#define LCD1602_DATAPINS P0
sbit LCD1602_E=P2^7;
sbit LCD1602_RW=P2^5;
sbit LCD1602_RS=P2^6;

/*********************************
函数声明
*********************************/
/*在51单片机12MHZ时钟下的延时函数*/
void Lcd1602_Delay1ms(uint c);   //误差 0us
/*LCD1602写入8位命令子函数*/
void LcdWriteCom(uchar com);
/*LCD1602写入8位数据子函数*/
void LcdWriteData(uchar dat) ;
/*LCD1602初始化子程序*/
void LcdInit();

#endif
```

下面这个main.c我在实验例程的基础上做了一点小修改，显示两行字符串，

main.c

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

*
实验现象：下载程序后插上LCD1602液晶在开发板上，即可显示

接线说明：（具体接线图可见开发攻略对应实验的“实验现象”章节）


注意事项：根据自己使用的LCD1602是否带有转接板，如果带有转接板的即为4位，需在LCD.H头文件中
将宏#define LCD1602_4PINS打开，我们这里使用的LCD1602是8位，所以默认将该宏注释


*********************************************************************************

*/

#include "reg52.h"    //此文件中定义了单片机的一些特殊功能寄存器
#include "lcd.h"
#include "string.h"

typedef unsigned int u16;   //对数据类型进行声明定义
typedef unsigned char u8;

u8 Disp[]=" STC89C52RC  ";
u8 Disp2[] = " learning   ";

/*********************************************************************************
* 函 数 名         : main
* 函数功能     : 主函数
* 输    入         : 无
* 输    出         : 无
*********************************************************************************/
void main(void)
{
 u8 i;
 LcdInit();
 for(i = 0; i < strlen(Disp); i++)
 {
  LcdWriteData(Disp[i]);
 }
 LcdWriteCom(0x80+0x40);  //设置数据指针起点为第二行
 for(i = 0; i < strlen(Disp2); i++)
 {
  LcdWriteData(Disp2[i]);
 }
 while(1);
}
```

实验结果：



实验23：LCD12864液晶（空）

由于我没买这个LCD屏幕，所以就不管它了。结束！

> 终于过完一遍，连续钻一天，看到后面都有些不耐烦了，，但说实话，小小的51真让人着迷，麻雀虽小五脏俱全，绝对是入门单片机的不二之选。

> 本文是我学习51开发板时随手记录，肯定会有一些小错误（技术上或字面表述上）。除了代码注释，如果发现了文中存在明显的错误，随时可以指出。