#include<reg52.h> //头文件

#include"eeprom52.h" // 调用EEPROM处理程序

#define uchar unsigned char //宏定义

#define uint unsigned int

#define LCD1602\_dat P0 //LCD 1602 数据传输IO口

sbit LCD1602\_rs=P2^5; //LCD1602控制IO口 //I/O 定义

sbit LCD1602\_rw=P2^6; //I/O 定义

sbit LCD1602\_e=P2^7;

sbit key\_1=P1^3; //按键控制IO口

sbit key\_2=P1^4;

sbit beep=P1^5; //蜂鸣器控制IO口

sbit led=P1^0; //指示灯控制IO口

unsigned long shu1,shu=1500; //检测频率变量，频率限制变量

uchar ms,sec; //50ms计数变量， 秒计数变量

uint cs; //中断溢出计数变量

bit OK,beep1; //标志位

/\*

1602液晶，是常用的显示器件，一共是16个管脚，其中有八个管脚是数据传输管脚，有三个管脚是数据命令使能端管脚，还有两组电源管脚，

其中一组电源管脚是给整个液晶进行供电的，还有一组电源是单纯的背景光电源，还剩下的最后一个管脚是对比度调节管脚，一般接上一个3K电

阻再接地即可。

一般我们用的函数，无非就是 LCD1602\_write 和 LCD1602\_writebyte

LCD1602\_write(x,y); 这个函数括号里面可以填写两个数据，第一个数据只能是 0 1 ，是0就说明第二个数据对液晶来说就是命令，填1就说明

第二个数据对于液晶来说就是要显示的数据。

LCD1602\_writebyte（）； 这个函数里面直接填上要显示的字符串即可，自动进行显示

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : delay()

\* 功能 : 小延时。

\* 输入 : 无

\* 输出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void delay(uint T) //延时程序

{

while(T--);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : LCD1602\_write(uchar order,dat)

\* 功能 : 1602写如数据函数

\* 输入 : 输入的命令值

\* 输出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD1602\_write(uchar order,dat) //1602 一个字节 处理

{

LCD1602\_e=0;

LCD1602\_rs=order;

LCD1602\_dat=dat;

LCD1602\_rw=0;

LCD1602\_e=1;

delay(10);

LCD1602\_e=0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : LCD1602\_writebye(uchar \*prointer)

\* 功能 : 1602写入数据函数 指针式

\* 输入 : 输入的命令值

\* 输出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD1602\_writebyte(uchar \*prointer) //1602 字符串 处理

{

while(\*prointer!='\0')

{

LCD1602\_write(1,\*prointer);

prointer++;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : LCD1602\_cls()

\* 功能 : 初始化1602液晶

\* 输入 : 无

\* 输出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD1602\_cls() //1602 初始化

{

LCD1602\_write(0,0x01); //1602 清屏 指令

delay(1500);

LCD1602\_write(0,0x38); // 功能设置 8位、5\*7点阵

delay(1500);

LCD1602\_write(0,0x0c); //设置 光标 不显示开关、不显示光标、字符不闪烁

LCD1602\_write(0,0x06);

LCD1602\_write(0,0xd0);

delay(1500);

}

/\*

数据显示的时候一般的处理：

首先，无论是数码管显示还是液晶显示，进行显示的时候绝对都是一个一个进行显示的，那么，比如说一个数据123，一百二十三，

进行显示的时候，要先显示1，然后是2，然后是3，那么怎么把数据提取出来？？

提取百位 123/100=1

提取十位 123/10=12 12%10=2 “%”是取余的意思，像这个，就是12对10取余，换句话说，12除以10，然后取余数，就是2

提取个位 123%10=3 解释同上

取余的用法也有很多种，大家只要知道出现这个的时候，一般都是进行数据提取的就行

然后

如果您是数码管显示数据，将提取的数据放到段码数组里面送给IO即可，

如果是液晶显示，需要将数据转化成字符，因为液晶是字符屏，只能显示字符数据，数据0对应的字符是0x30，数据1对应的字符是0x31，

所以将提取出的数据直接加上0x30送给液晶即可，或者加上'0' 也是一样的

\*/

void show()

{

LCD1602\_write(0,0x80); //0发送命令

LCD1602\_writebyte("NowFreq:"); //显示检测频率

LCD1602\_write(1,0x30+shu1/100000%10); //1发送数据

LCD1602\_write(1,0x30+shu1/10000%10); //因为液晶只能显示字符，所以我们在这里加一个0x30把数字强制转换成字符，0x30是字符0

LCD1602\_write(1,0x30+shu1/1000%10);

LCD1602\_writebyte(".");

LCD1602\_write(1,0x30+shu1/100%10);

LCD1602\_writebyte("KHz");

LCD1602\_write(0,0xc0);

LCD1602\_writebyte("SetFreq:"); //显示限制频率

LCD1602\_write(1,0x30+shu/1000%10);

LCD1602\_write(1,0x30+shu/100%10);

LCD1602\_write(1,0x30+shu/10%10);

LCD1602\_writebyte(".");

LCD1602\_write(1,0x30+shu%10);

LCD1602\_writebyte("KHz");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : key()

\* 功能 : 按键控制程序

\* 输入 : 无

\* 输出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

无论是什么单片机，基本使用都是这样

关于内部存储区，EEPROM，不同的单片机使用流程基本一致，单片机内部有很多存储单元，或者说扇区，每一个扇区下面有很多地址，

数据就是存储在这些地址下面的。存储函数的程序都是官方提供好的，这些程序，咱们只需要用三个，一个是扇区擦除函数，一个是

数据写函数，还有一个就是数据读取函数。

扇区擦除函数------使用哪个扇区，先对那个扇区进行擦，函数里填写要擦除扇区的首地址 例如 SectorErase（0x2000）;就是说擦除首地址为0x2000的扇区数据

数据存储----------扇区擦除之后，就可以使用这个扇区下的地址进行存储数据 例如 byte\_write（0x2000,123）; 就是说将123存储在0x2000地址下

数据读取----------直接调用即可，例如 Dat=byte\_read（0x2000）;就是说将0x2000地址下的数据读取出来给 Dat

另外----

//51单片机存储区域是8位的，也就是说能够存下的最大数据是 255，而我们存的数据一旦大于256就会出现一些问题

//所以，如果您的设计需要存储的数据大于256，那就把数据拆开存 /256得到高位 %256得到低位，之所以是256，是因为0-255,256个数

// 例如数据257 257/256=1 257%256=1 ,这就存进去两个1，读取的时候，将高位数据乘以256加低位数据，还原数据

\*/

void key()

{

if(OK==1)

{

if(!key\_1) //判断按键是否按下

{

OK=0;

if(sec==3) //执行按键对应的控制内容

{

if(shu<3000) shu+=10; //长按处理

}else

{

if(shu<3000) shu+=1; //短按处理

}

SectorErase(0x2000); //保存数据到单片机EEPROM中

byte\_write(0x2000,shu/256);

byte\_write(0x2001,shu%256);

}

if(!key\_2) //判断按键是否按下

{

OK=0; //执行按键对应的控制内容

if(sec==3)

{

if(shu>200) shu-=10;//长按处理

}else

{

if(shu>200) shu-=1; //短按处理

}

SectorErase(0x2000); //保存数据到单片机EEPROM中

byte\_write(0x2000,shu/256);

byte\_write(0x2001,shu%256);

}

}

if(key\_1==1&&key\_2==1) //判断按键是否按下

{

sec=0; //重新计时

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : proc（）

\* 功能 : 报警指示灯处理函数

\* 输入 : 无

\* 输出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void proc()

{

if(shu1>=(shu\*100)|shu1==0)

{

led=0;

beep1=1;

}else

{

led=1;

beep1=0;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : main()

\* 功能 : 主程序

\* 输入 : 无

\* 输出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main()

{

uchar H,L;

TMOD=0x51; //定时器0配置 方式0 定时模式; 定时器1配置 方式0 计数模式

TH1=0; //定时器 赋初值

TL1=0;

TL0 = 0xB0; //设置定时初值

TH0 = 0x3C; //设置定时初值

TR0=1; //定时器0启动定时

TR1=1; //定时器0启动计数

EA=1; //打开总中断

ET0=1; //定时器0中断

ET1=1; //定时器1中断

H=byte\_read(0x2000); //读取单片机EEPROM中保存的数据

L=byte\_read(0x2001);

shu=H\*256+L;

if(shu>3000) shu=1500; //判断读出的数据是否正确 如果不正确 //则重新赋值

LCD1602\_cls(); //LCD1602初始化

while(1)

{

key(); //调用按键控制程序

show(); //调用LCD1602液晶显示程序

proc(); //调用程序处理函数

}

}

void init\_1() interrupt 1

{

TL0 = 0xB0; //设置定时初值

TH0 = 0x3C; //设置定时初值

ms++; //50ms定时计数

if(ms%4 ==0) //按键处理时间控制 没200ms处理一次

{

OK=1;

}

if(ms%5==0) //蜂鸣器报警，指示灯闪烁处理

{

if(beep1==1)

{

led=0;

beep=!beep;

}else

{

led=1;

beep=1;

}

}

if(ms>19) //1s定时 处理

{

shu1=(long)cs\*65535+TH1\*256+TL1; //计算频率

cs=TH1=TL1=0;

ms=0;

if(!key\_1||!key\_2) //按键长按处理

{

sec++;

if(sec>3) //长按3s判断

{

sec=3;

}

}else

{

sec=0;

}

}

}

void init\_3() interrupt 3 //外部中断检测当前传感器频率

{

cs++; //中断溢出计数

}

#ifndef \_EEPROM52\_H\_

#define \_EEPROM52\_H\_

#include <intrins.h>

/\*\*\*\*\*\*\*\*STC89C51扇区分布\*\*\*\*\*\*\*

第一扇区：1000H--11FF

第二扇区：1200H--13FF

第三扇区：1400H--15FF

第四扇区：1600H--17FF

第五扇区：1800H--19FF

第六扇区：1A00H--1BFF

第七扇区：1C00H--1DFF

第八扇区：1E00H--1FFF

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*STC89C52扇区分布\*\*\*\*\*\*\*

第一扇区：2000H--21FF

第二扇区：2200H--23FF

第三扇区：2400H--25FF

第四扇区：2600H--27FF

第五扇区：2800H--29FF

第六扇区：2A00H--2BFF

第七扇区：2C00H--2DFF

第八扇区：2E00H--2FFF

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define RdCommand 0x01 //定义ISP的操作命令

#define PrgCommand 0x02

#define EraseCommand 0x03

#define Error 1

#define Ok 0

#define WaitTime 0x01 //定义CPU的等待时间

sfr ISP\_DATA=0xe2; //寄存器申明

sfr ISP\_ADDRH=0xe3;

sfr ISP\_ADDRL=0xe4;

sfr ISP\_CMD=0xe5;

sfr ISP\_TRIG=0xe6;

sfr ISP\_CONTR=0xe7;

/\* ================ 打开 ISP,IAP 功能 ================= \*/

void ISP\_IAP\_enable(void)

{

EA = 0; /\* 关中断 \*/

ISP\_CONTR = ISP\_CONTR & 0x18; /\* 0001,1000 \*/

ISP\_CONTR = ISP\_CONTR | WaitTime; /\* 写入硬件延时 \*/

ISP\_CONTR = ISP\_CONTR | 0x80; /\* ISPEN=1 \*/

}

/\* =============== 关闭 ISP,IAP 功能 ================== \*/

void ISP\_IAP\_disable(void)

{

ISP\_CONTR = ISP\_CONTR & 0x7f; /\* ISPEN = 0 \*/

ISP\_TRIG = 0x00;

EA = 1; /\* 开中断 \*/

}

/\* ================ 公用的触发代码 ==================== \*/

void ISPgoon(void)

{

ISP\_IAP\_enable(); /\* 打开 ISP,IAP 功能 \*/

ISP\_TRIG = 0x46; /\* 触发ISP\_IAP命令字节1 \*/

ISP\_TRIG = 0xb9; /\* 触发ISP\_IAP命令字节2 \*/

\_nop\_();

}

/\* ==================== 字节读 ======================== \*/

unsigned char byte\_read(unsigned int byte\_addr)

{

EA = 0;

ISP\_ADDRH = (unsigned char)(byte\_addr >> 8);/\* 地址赋值 \*/

ISP\_ADDRL = (unsigned char)(byte\_addr & 0x00ff);

ISP\_CMD = ISP\_CMD & 0xf8; /\* 清除低3位 \*/

ISP\_CMD = ISP\_CMD | RdCommand; /\* 写入读命令 \*/

ISPgoon(); /\* 触发执行 \*/

ISP\_IAP\_disable(); /\* 关闭ISP,IAP功能 \*/

EA = 1;

return (ISP\_DATA); /\* 返回读到的数据 \*/

}

/\* ================== 扇区擦除 ======================== \*/

void SectorErase(unsigned int sector\_addr)

{

unsigned int iSectorAddr;

iSectorAddr = (sector\_addr & 0xfe00); /\* 取扇区地址 \*/

ISP\_ADDRH = (unsigned char)(iSectorAddr >> 8);

ISP\_ADDRL = 0x00;

ISP\_CMD = ISP\_CMD & 0xf8; /\* 清空低3位 \*/

ISP\_CMD = ISP\_CMD | EraseCommand; /\* 擦除命令3 \*/

ISPgoon(); /\* 触发执行 \*/

ISP\_IAP\_disable(); /\* 关闭ISP,IAP功能 \*/

}

/\* ==================== 字节写 ======================== \*/

void byte\_write(unsigned int byte\_addr, unsigned char original\_data)

{

EA = 0;

// SectorErase(byte\_addr);

ISP\_ADDRH = (unsigned char)(byte\_addr >> 8); /\* 取地址 \*/

ISP\_ADDRL = (unsigned char)(byte\_addr & 0x00ff);

ISP\_CMD = ISP\_CMD & 0xf8; /\* 清低3位 \*/

ISP\_CMD = ISP\_CMD | PrgCommand; /\* 写命令2 \*/

ISP\_DATA = original\_data; /\* 写入数据准备 \*/

ISPgoon(); /\* 触发执行 \*/

ISP\_IAP\_disable(); /\* 关闭IAP功能 \*/

EA =1;

}

#endif