西安交通大學

XI'AN JIAOTONG UNIVERSITY

电子系统设计

实验报告

班级：xxxx

姓名：xxxx

学号：xxxxxxxxx

数字钟设计

一、实验目的

1.了解单片机的基本概念和工作原理。
2.掌握定时器的结构及工作原理。
3.掌握中断系统的概念与工作原理。
4.能够正确应用数码管、蜂鸣器、按键等外设。

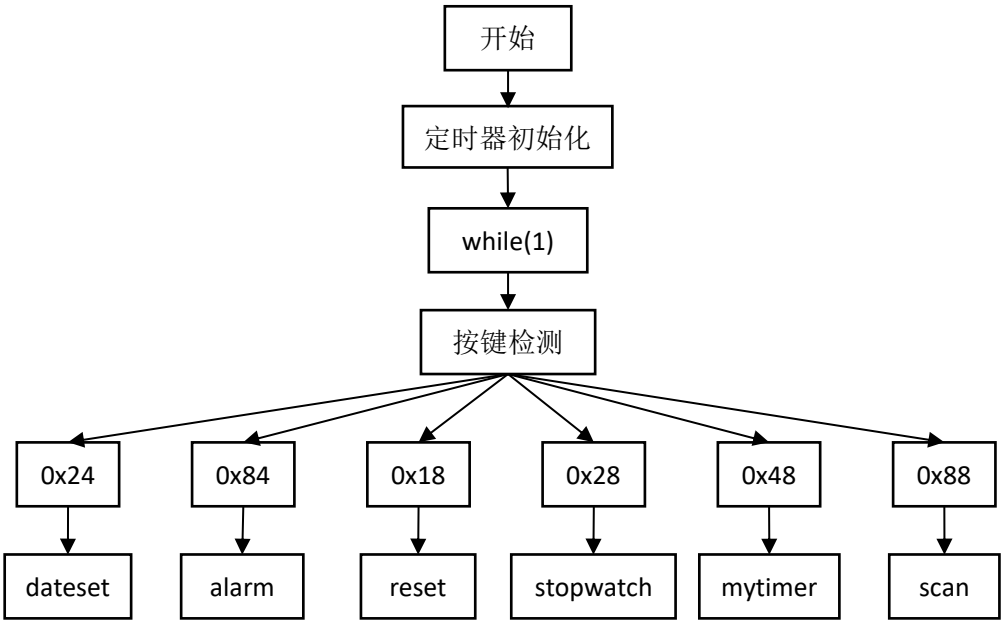
二、实验要求

1.时-分-秒（2 位-2 位-2 位）显示。
2.可通过按键置入时间值(参照电子表设置时间工作模式)。
3.可通过按键控制在 LED 上从右向左滚动显示年_月_ 日 3 次，如：2013_01_20 空空 2013_01_20。
4.实现每日闹铃提醒功能，闹铃时间可用按键设置。闹铃采用提示音表示。
5.实现秒表功能。
6.实现定时器功能（预置定时时间，按键启动，倒计时，计到 0 响提示音。
7.设计实现音乐提示音。

三、系统方案设计

1.总体方案设计

1）系统总体框图



2）系统设计思路

本设计以 51 单片机实验板为载体，实现了实时数字钟、时间显示、时间设置、日期设置、日期滚动显示、秒表、定时器、闹钟、音乐提示音等功能。程序中对要求实现的不同的功能采用分模块处理，进入主函数后，首先进行定时器的

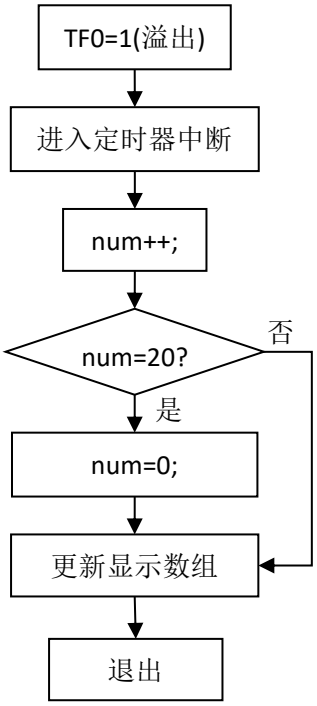
初始化配置，接着利用 while(1)循环实现按键的实时检测。当检测到有效按键按下后，执行相应功能对应的子函数。

采用定时器 timer0 和 timer1 的两个中断，其中 timer0 中断用于为实时时钟定时，timer1 中断用于秒表、定时器、闹钟、音乐提示音等其他需要定时的功能。考虑到可利用的中断有限，且若每个功能均单独使用一个定时器中断，则定时器的配置和编程将会变得复杂，故在本设计中采用了定时器中断的复用，即设置一个标志变量，根据标志变量取值的不同来选择在中断内执行不同的中断函数。

2. 各模块方案设计

1）实时时钟设计

实时时钟的定时器中断模块流程图如下：



程序中，为实现 1s 的定时，采用定时器 0、设定为工作方式 1（16 位）。定时器的计数初值设定为 TH0=0x3C、TL0=0xB0，即 15536。由于计满时为 65536，故计数器溢出一次时，计数次数为 50000，对应 50ms。这样，设置计数变量为 num，每计满一次进入定时器中断函数，则 num++，当 num 计满 20 时即为 1s。此时，秒计时变量 sec++，同时根据秒计时变量更新分、时的计数变量 min、hour，并根据当前对应的时刻值更新显示数组，即可实现时钟的实时显示。

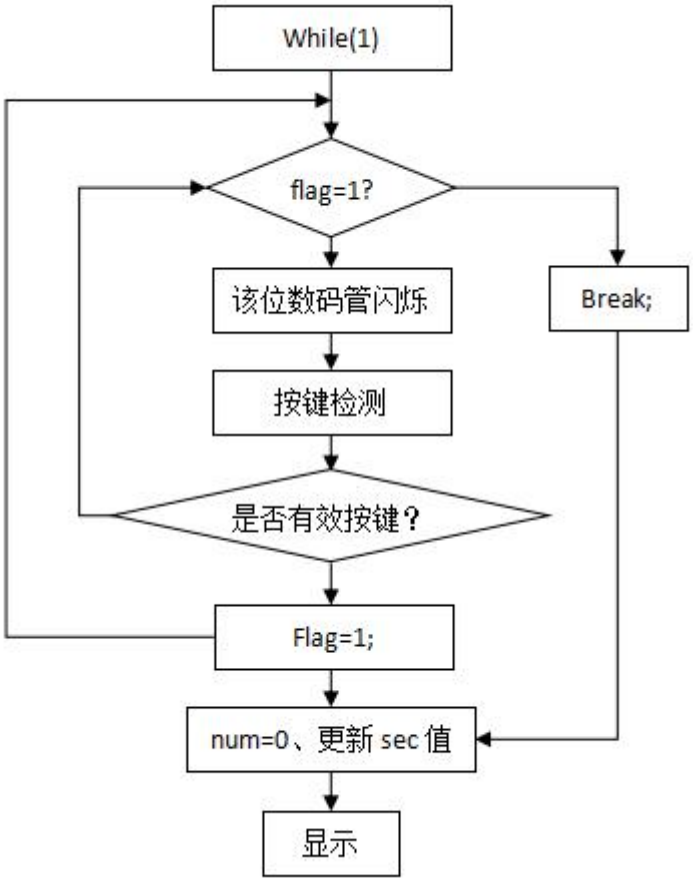
2）时间校准模块设计

本设计中，可实现时、分、秒按位顺序调整，用闪烁模式来提醒用户当前调整位。由于每一位都遵循了同样的程序设计步骤，故在此以调整秒的个位为例，说明算法流程。流程图如下：

用 while(1)循环来实现实时按键检测和数码管显示，当无按键按下或者按下的按键无效时，数码管实时显示当前时间，并在当前调整位上采取闪烁模式，以提醒用户当前正在设置此位的时间。当有效的校准按键按下时，将 flag 赋值为

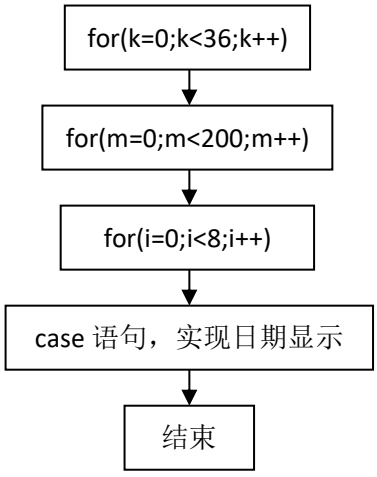
1，并跳出循环。此时，根据校准时间重新设定时、分、秒的值，并在数码管上显示。

需要注意的是，由于设定时间时计时器还在计数，故计时器中断中的 num 为历史累计值，这直接导致了置入的秒计时个位值不准，故在对秒进行校准时，需将 num 计数值清零。



3）日期滚动模式设计

日期滚动显示的流程图如下：

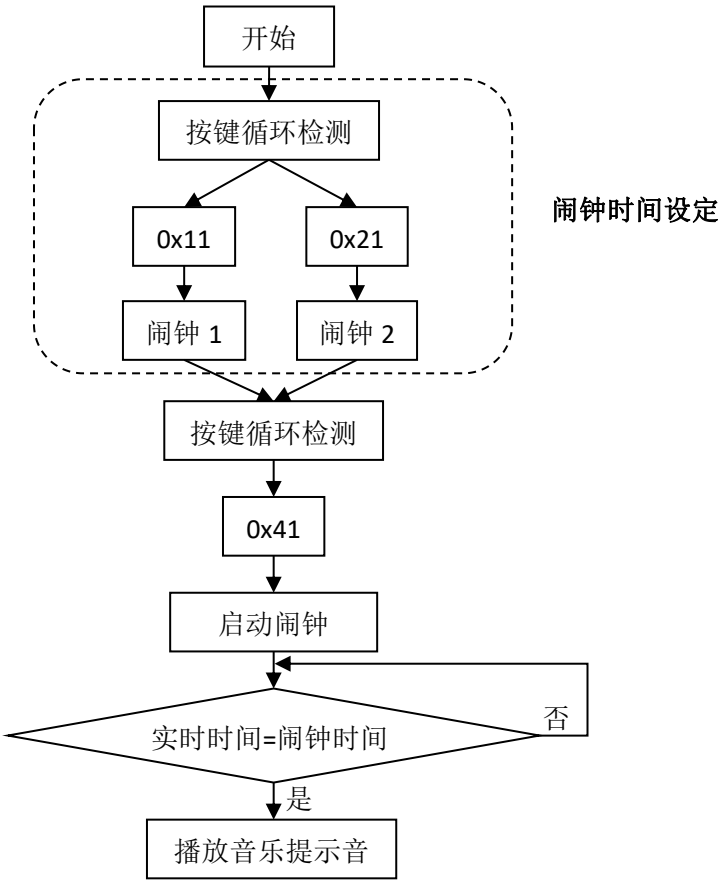


其中，最外层的 k 循环用于控制字母滚动位置，第二层的 m 循环用于延时，控制字幕滚动的速度，最内层的 i 循环用于动态扫描，使八个数码管能同时显示

不同的数字。日期信息以“年—月—日”+“空格”的形式存储在无符号字符数组 date 中，该数组决定了显示的内容。

4）闹钟提醒功能设计

闹钟提醒功能的流程图如下：



本设计中，除了原有的闹钟功能外，还增加了一个闹钟。两个闹钟对应播放的音乐提示音为不同的音乐。闹钟 1 和闹钟 2 可单独使用，也可共同使用。

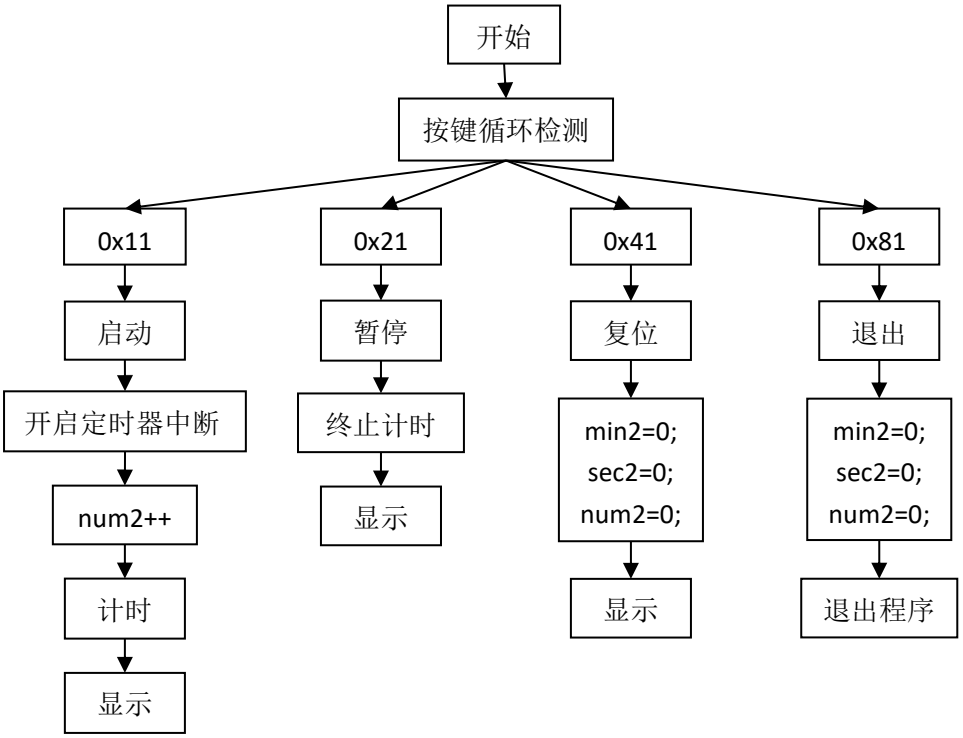
进入函数后，首先通过按键循环检测来实现闹钟时间的设定。其中按下按键 0x11 进行第一个闹钟时间的设定，若想单独使用闹钟 1，则时间设定完毕后按下 0x41，即启动闹钟；若想同时使用闹钟 1 和闹钟 2，则继续按下 0x21 进行闹钟 2 的设定，当设定完毕后，按下按键 0x41，即可同时启动两个闹钟。

闹钟启动后，即进入比较和判断部分的程序。以闹钟 1 为例，闹钟 1 设定时间的时、分、秒分别用变量 hour_alm、min_alm、sec_alm 表示，实时时钟的时、分、秒分别用变量 hour、min、sec 表示，当实时时钟的时间等于闹钟的设定时间，即满足条件(hour==hour_alm&&min==min_alm&&sec==sec_alm)时，启动音乐提示音程序作为闹铃。

关闭音乐提示音后，继续进行闹钟 2 的比较和判断，若到达闹钟 2 的设定时间，则播放闹钟 2 对应的音乐提示音并退出闹钟程序。

5）秒表功能的设计

秒表的流程图如下：



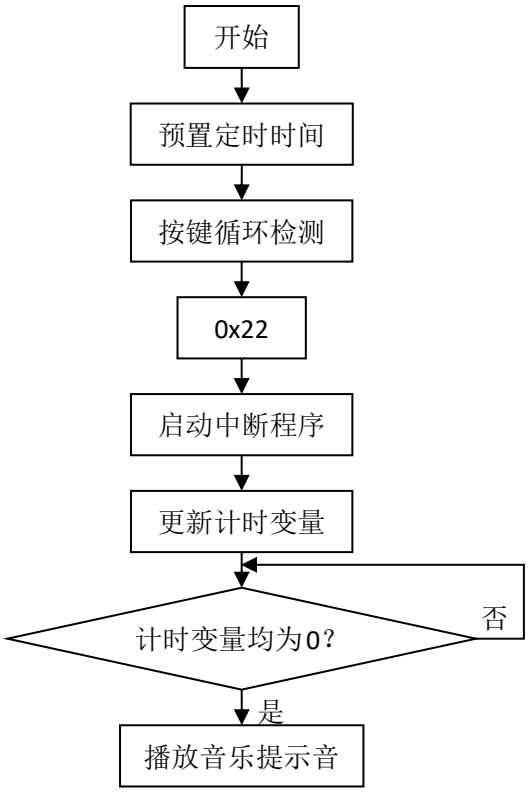
本设计实现了最大计时时间 1 小时、计时精度 0.01s 的秒表。进入秒表函数后，首先进行按键的循环检测，当按下 0x11 时执行定时器相应的中断程序，启动秒表。秒表的计时变量为 min2、sec2、num2。加载单片机定时器初值为 0xd8f0，对应的计时时间为 10ms，定时器每计满溢出一次，则 num2++,当 num2 满 100 时，sec2++，依此类推，即实现了秒表时间的更新；当按下 0x21 时，关闭定时器中断，仅显示不更新，实现了暂停功能；当按下 0x41 时，将各个计时变量置零，实现了复位功能；当按下 0x81 时，将各计时变量置零并退出程序，实现按键控制退出秒表。

6）定时器功能的设计

定时器设计流程图如下：

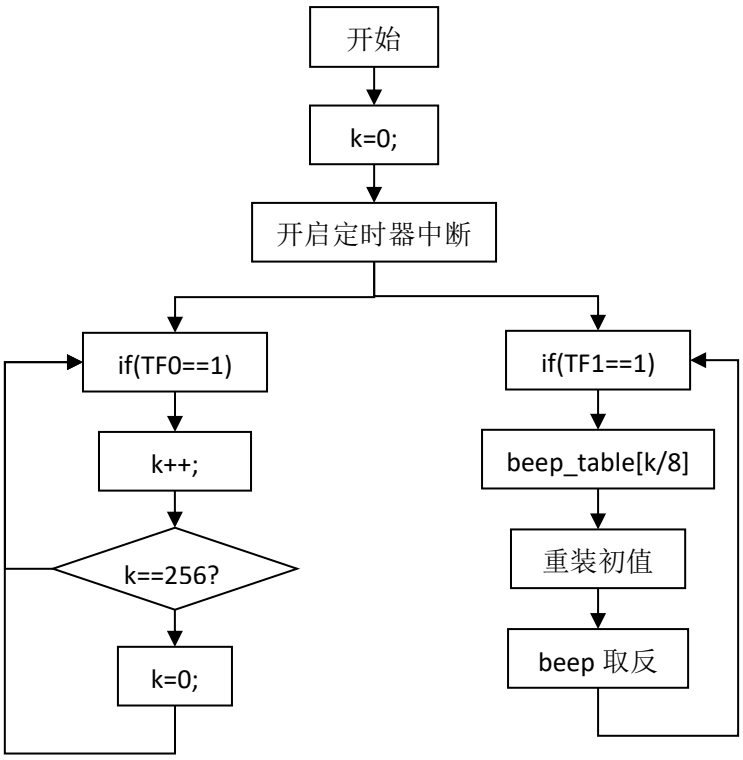
进入函数后，首先进入预置定时时间的部分，通过按键键入时间值后，进入按键循环检测，当检测到按键 0x22 按下后，启动中断程序，开启定时器。中断程序中，计时变量为 num_tmr、hour_tmr、min_tmr、sec_tmr。定时器 timer1 每计满溢出一次即 10ms，则 num_tmr++，当 num_tmr 计到 100 时，向秒计时借位，sec_tmr--，依此类推，实现倒计时功能。

计时变量的判断与更新同步进行，当 hour_tmr、min_tmr、sec_tmr 均为 0 时，退出程序，同时开始播放音乐提示音，否则继续进行检测。



7）音乐提示音的设计

音乐提示音部分的设计流程图如下：



在本设计中，采用定时器 0 来控制每个音符发声的时长，用定时器 1 来控制蜂鸣器发声的频率。其中音乐中的每一个音符对应的定时器计数初始值均按次序

排放在音乐存储表中，用 k 来代表它们在表中的位置。进入音乐播放程序后，首先将 k 置为 0，接着进入相应的定时器中断程序。

对于时长的控制，timer0 每计满溢出一次，k++。由于 timer0 的计数初值是 0x3cb0，对应的时间间隔是 50ms，而这里采用的音乐为儿歌《洋娃娃和小熊跳舞》，其合适的播放速度约为每个音符 400ms，故程序中 k 为 8 的倍数时，实现音符的跳转。

对于频率的控制，由于声音是靠物体的振动发出的，则要使蜂鸣器发出不同频率的声音，只需要产生对应频率的 PWM 波即可。在程序中利用数组 beep_table[] 对 timer1 的计数初值进行更新，timer1 每计满溢出一次，beep 取反，将 beep 的值输入蜂鸣器，即可使蜂鸣器发出对应频率的声音。

四、实验结果

1. 时-分-秒（2 位-2 位-2 位）显示

上电，即可实现时钟功能，并实时显示时间。

2. 可通过按键置入时间值(参照电子表设置时间工作模式）

按下按键 0x80，即进入时间校准模式，进入后不需再按其他按键，只需按位调整时间。当前所调整的数码管会闪烁数字，以提示用户当前调整位。当最后一位校准结束后，自动恢复实时时钟模式。

3. 可通过按键控制在 LED 上从右向左滚动显示年_月_ 日 3 次，如：2013_01_20 空空 2013_01_20

按下按键 0x88，即进入日期滚动模式，日期从右向左按规定的格式滚动 3 次，结束后自动恢复实时时钟模式 。

4. 可实现每日闹铃提醒功能，闹铃时间可用按键设置，闹铃采用音乐提示音表示。另外增加了附加功能，即可同时设置播放不同音乐的两个闹钟。

5. 可实现秒表功能。秒表最长计时时间为一小时，计时精度为 0.01s。可通过按键控制秒表的启动、暂停、复位、退出。

6. 实现定时器功能（预置定时时间，按键启动，倒计时，计到 0 响提示音）。

7. 实现了音乐提示音，并能通过按键关闭提示音。

五、资源使用情况

资源使用情况为：

Data=9.0
Xdata=273
Code=14745

说明占用的内部 RAM 为 9 字节，外部 RAM 为 273 字节，程序占用的程序存储器空间为 14745 字节。

附录

完整代码如下：

```
#include <REG51.H>          //special function register declarations
#include <absacc.h>
#include <stdio.h>

Unsigned char code
seg_table[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
//查找表
unsigned char code
seg_table_stw[10]={0xbf,0x86,0xdb,0xcf,0xe6,0xed,0xfd,0x87,0xff,0xef};
//秒表查找表

unsigned char show_table[8]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
//时钟显示
unsigned char show_table_stw[7]={0x00,0x00,0x00,0x00,0x00,0x00,0x00};
//秒表显示
unsigned char show_table_tmr[8]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
//定时器显示
unsigned char show_table_alm[8]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
//闹钟显示
unsigned char show_table_date[8]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
//日期显示

unsigned char
date[]={0x00,0x00,0x00,0x00,0x40,0x00,0x00,0x40,0x00,0x00,0x00,0x00};
//日期信息


unsigned int beep_table[32]={
0xf88b,0xf95b,0xfa15,0xfa67,0xfb05,0xfb05,0xfb05,0xfa15,
0xfa67,0xfa67,0xfa15,0xf95b,0xf88b,0xfa15,0xfb05,0xfb05,
0xf88b,0xf95b,0xfa15,0xfa67,0xfb05,0xfb05,0xfb05,0xfa15,
0xfa67,0xfa67,0xfa15,0xf95b,0xf88b,0xfa15,0xf88b,0xf88b};
//音乐存储表 1

unsigned int beep_table_2[24]={
0xfb05,0xfb8c,0xfb05,0xfc44,0xfc0c,0xfc0c,
0xfb05,0xfb8c,0xfb05,0xfcac,0xfc44,0xfc44,
0xfb05,0xfd82,0xfd09,0xfc44,0xfc0c,0xfb8c,
0xfd34,0xfd09,0xfc44,0xfcac,0xfc44,0xfc44};
 //音乐存储表 2
```

```
sbit beep =P1^6;

int num=0,num2=0,num_tmr=0;                    //计时变量
int year=2019,month=2,day=28;                  //日期变量
int k=0,k_2=0,flag=0,flag_music=0;             //flag 控制定时器 1 中断程序的选择

int hour=0,min=0,sec=0;                        //时钟
int min2=0,sec2=0;                             //秒表
int hour_tmr=0,min_tmr=0,sec_tmr=0;            //定时器
int hour_alm=0,min_alm=0,sec_alm=0;            //闹钟
int hour_alm_2=0,min_alm_2=0,sec_alm_2=0;      //闹钟 2
int stw=0,tmr=0,alm=0,alm_2=0;                 //各模式的标志变量

void Delaymini();                              //函数声明
void timer0();                                 //定时器中断函数
void timer1();
                                               //定时器初始化
void timer0_init();
void timer1_init();

void twinkle(int sel,int n);                   //设置时间时闪烁
void twinkle_tmr(int sel,int n);
void twinkle_alm(int sel,int n);

void display();                                //数码管显示
void display_tmr();
void display_alm();
void display_date();

unsigned char getkeycode();                    //按键扫描
void reset();                                  //功能函数
void scan();
void music();
void music_2();
void stopwatch();
void mytimer();
void alarm();
void show_alm();
void show_alm_2();
void update();
void dateset();

/**********************************主函数**************************************/
void  main()
```

```
{
    beep=1;
    timer0_init();
    timer1_init();

    TR1=0;
    while(1)
    {
        switch(getkeycode())
        {
            case 0x24:dateset();break;            //第三行第 2 列
            case 0x84:alarm();break;              //第三行第 4 列
            case 0x18:reset();break;              //第四行第 1 列
            case 0x28:stopwatch();break;          //第四行第 2 列
            case 0x48:mytimer();break;            //第四行第 3 列
            case 0x88:scan();break;               //第四行第 4 列

            default: break;
        }
        display();
    }
}

/**********************************延时函数**********************************/
void Delaymini()
{
    unsigned char i,j;
    for(i=1;i>0;i--)
        for(j=30;j>0;j--);
}

/**********************************定时器初始化**********************************/
void timer0_init()
{
    TMOD=0x11;      //模式控制 方式 1（16 位）
    TH0=0x3c;       //初始值 3CB0：15536   终值 65535   计数值 50000    计数周期
0.05s
    TL0=0xb0;
    TR0=1;          //启动定时器工作
    EA=1;           //CPU 中断允许位
    ET0=1;          //T0 中断允许位
}

void timer1_init()
```

```
{
    TMOD=0x11;
    TH1=0xd8;       //10ms
    TL1=0xf0;
    EA=1;
    ET1=1;
}
/*********************************定时器中断*********************************/
void timer0() interrupt 1
{
    TMOD=0x11;
    TH0=0x3c;
    TL0=0xb0;
    num++;
    if(flag==4)
    {
        k++;
        if(k==256)
            k=0;
    }
    if(flag==3)
    {
        k_2++;
        if(k_2==192)
            k_2=0;
    }
    if(num==20)
    {
        num=0;
        sec++;
            if(sec==60)
            {
                sec=0;
                min++;
                if(min==60)
                {
                    min=0;
                    hour++;
                    if(hour==24)
                        hour=0;
                        update();
                }
            }
    }
```

```
    show_table[0]=seg_table[hour/10];
    show_table[1]=seg_table[hour%10];
    show_table[2]=0x40;
    show_table[3]=seg_table[min/10];
    show_table[4]=seg_table[min%10];
    show_table[5]=0x40;
    show_table[6]=seg_table[sec/10];
    show_table[7]=seg_table[sec%10];

}

void timer1() interrupt 3
{
    if(flag==1)
    {
        TMOD=0x11;
        TH1=0xd8;
        TL1=0xf0;
        if(stw==1)
            num2++;
        if(stw==3||stw==4)
            {
                num2=0;
                min2=0;
                sec2=0;
            }

        if(num2==100)
        {
            num2=0;
            sec2++;
                if(sec2==60)
                {
                    sec2=0;
                    min2++;
                    if(min2==60)
                        min2=0;
                }
        }
        show_table_stw[0]=seg_table[min2/10];
        show_table_stw[1]=seg_table[min2%10];
        show_table_stw[2]=0x40;
        show_table_stw[3]=seg_table[sec2/10];
        show_table_stw[4]=seg_table_stw[sec2%10];
```

```
    show_table_stw[5]=seg_table[num2/10];
    show_table_stw[6]=seg_table[num2%10];
}

if(flag==2)
{
    TMOD=0x11;
    TH1=0xd8;
    TL1=0xf0;

    if(tmr==2)
    {
        num_tmr++;
        if(num_tmr==100)
        {
            num_tmr=0;
            if(sec_tmr==0)
                if(min_tmr==0)
                    if(hour_tmr==0)
                    {
                        flag=0;
                        flag_music=1;
                        tmr=0;
                    }
                    else
                    {
                        hour_tmr--;
                        min_tmr=59;
                        sec_tmr=59;
                    }
                else
                {
                    min_tmr--;
                    sec_tmr=59;
                }
            else
                sec_tmr--;
        }
    }
    show_table_tmr[0]=seg_table[hour_tmr/10];
    show_table_tmr[1]=seg_table[hour_tmr%10];
    show_table_tmr[2]=0x40;
    show_table_tmr[3]=seg_table[min_tmr/10];
    show_table_tmr[4]=seg_table[min_tmr%10];
```

```
        show_table_tmr[5]=0x40;
        show_table_tmr[6]=seg_table[sec_tmr/10];
        show_table_tmr[7]=seg_table[sec_tmr%10];
    }

    if(flag==3)
    {
        TMOD=0x11;
        TH1=(beep_table_2[k_2/8]>>8)&0xff;
        TL1=beep_table_2[k_2/8]&0xff;
        beep=~beep;
    }

    if(flag==4)
    {
        TMOD=0x11;
        TH1=(beep_table[k/8]>>8)&0xff;
        TL1=beep_table[k/8]&0xff;
        beep=~beep;
    }

    if(flag==5)
    {
        TMOD=0x11;
        TH1=0xd8;
        TL1=0xf0;
        show_table_date[0]=seg_table[year/1000];
        show_table_date[1]=seg_table[(year/100)%10];
        show_table_date[2]=seg_table[(year/10)%10];
        show_table_date[3]=seg_table[year%10];
        show_table_date[4]=seg_table[month/10];
        show_table_date[5]=seg_table[month%10];
        show_table_date[6]=seg_table[day/10];
        show_table_date[7]=seg_table[day%10];
    }

}

/**********************************更新日期**********************************/
void update()
{
    if(month==1||month==3||month==5||month==7||month==8||month==10||month==12)
        if(day==31)
            if(month==12)
```

```
        {
            month=1;
            day=1;
            year++;
        }
        else
        {
            month++;
            day=1;
        }
    else
        day++;
else if(month==4||month==6||month==9||month==11)
    if(day==30)
    {
        month++;
        day=1;
    }
    else
        day++;
else if(month==2)                       //2月
    if(year%4)            //非闰年
        if(day==28)
        {
            month++;
            day=1;
        }
        else
            day++;
    if(year%4==0)         //闰年
        if(day==29)
        {
            month++;
            day=1;
        }
        else
            day++;

date[0]=seg_table[year/1000];
date[1]=seg_table[(year/100)%10];
date[2]=seg_table[(year/10)%10];
date[3]=seg_table[year%1000];
date[4]=0x40;
date[5]=seg_table[month/10];
```

```
    date[6]=seg_table[month%10];
    date[7]=0x40;
    date[8]=seg_table[day/10];
    date[9]=seg_table[day%10];
}

/**********************************日期设置**********************************/
void dateset()
{
    int i,m1=0,n1=0,m2=0,n2=0,m3=0,n3=0,m4=0,n4=0;
    int flag1=0,flag2=0,flag3=0,flag4=0,flag5=0,flag6=0,flag7=0,flag8=0;
    year=0;
    month=0;
    day=0;
    flag=5;
    TR1=1;
    for(i=0;i<10000;i++);
    while(1)              //时校准
    {
        if(flag1==1)
            break;
        display_date();
        switch(getkeycode())
        {
            case 0x11:m1=1;flag1=1;break;              //第一行第1列
            case 0x21:m1=2;flag1=1;break;              //第一行第2列
            case 0x41:m1=3;flag1=1;break;              //第一行第3列
            case 0x81:m1=4;flag1=1;break;              //第一行第4列

            case 0x12:m1=5;flag1=1;break;              //第二行第1列
            case 0x22:m1=6;flag1=1;break;              //第二行第2列
            case 0x42:m1=7;flag1=1;break;              //第二行第3列
            case 0x82:m1=8;flag1=1;break;              //第二行第4列

            case 0x14:m1=9;flag1=1;break;              //第三行第1列
            case 0x24:m1=0;flag1=1;break;              //第三行第2列
            default:break;
        }
    }
    year=m1*1000;
    for(i=0;i<1000;i++)
        display_date();

    while(1)              //时校准
```

```
{
    if(flag2==1)
        break;
    display_date();
    switch(getkeycode())
    {
        case 0x11:n1=1;flag2=1;break;              //第一行第1列
        case 0x21:n1=2;flag2=1;break;              //第一行第2列
        case 0x41:n1=3;flag2=1;break;              //第一行第3列
        case 0x81:n1=4;flag2=1;break;              //第一行第4列

        case 0x12:n1=5;flag2=1;break;              //第二行第1列
        case 0x22:n1=6;flag2=1;break;              //第二行第2列
        case 0x42:n1=7;flag2=1;break;              //第二行第3列
        case 0x82:n1=8;flag2=1;break;              //第二行第4列

        case 0x14:n1=9;flag2=1;break;              //第三行第1列
        case 0x24:n1=0;flag2=1;break;              //第三行第2列
        default:break;
    }
}
year=year+n1*100;
for(i=0;i<1000;i++)
    display_date();

while(1)            //分校准
{
    if(flag3==1)
        break;
    display_date();
    switch(getkeycode())
    {
        case 0x11:m2=1;flag3=1;break;              //第一行第1列
        case 0x21:m2=2;flag3=1;break;              //第一行第2列
        case 0x41:m2=3;flag3=1;break;              //第一行第3列
        case 0x81:m2=4;flag3=1;break;              //第一行第4列

        case 0x12:m2=5;flag3=1;break;              //第二行第1列
        case 0x22:m2=6;flag3=1;break;              //第二行第2列
        case 0x42:m2=7;flag3=1;break;              //第二行第3列
        case 0x82:m2=8;flag3=1;break;              //第二行第4列

        case 0x14:m2=9;flag3=1;break;              //第三行第1列
        case 0x24:m2=0;flag3=1;break;              //第三行第2列
```

```
        default:break;
    }
}
year=year+m2*10;
for(i=0;i<1000;i++)
    display_date();

while(1)            //分校准
{
    if(flag4==1)
        break;
    display_date();
    switch(getkeycode())
    {
        case 0x11:n2=1;flag4=1;break;              //第一行第1列
        case 0x21:n2=2;flag4=1;break;              //第一行第2列
        case 0x41:n2=3;flag4=1;break;              //第一行第3列
        case 0x81:n2=4;flag4=1;break;              //第一行第4列

        case 0x12:n2=5;flag4=1;break;              //第二行第1列
        case 0x22:n2=6;flag4=1;break;              //第二行第2列
        case 0x42:n2=7;flag4=1;break;              //第二行第3列
        case 0x82:n2=8;flag4=1;break;              //第二行第4列

        case 0x14:n2=9;flag4=1;break;              //第三行第1列
        case 0x24:n2=0;flag4=1;break;              //第三行第2列
        default:break;
    }
}
year=year+n2;
for(i=0;i<1000;i++)
    display_date();

while(1)            //分校准
{
    if(flag5==1)
        break;
    display_date();
    switch(getkeycode())
    {
        case 0x11:m3=1;flag5=1;break;              //第一行第1列
        case 0x21:m3=2;flag5=1;break;              //第一行第2列
        case 0x41:m3=3;flag5=1;break;              //第一行第3列
        case 0x81:m3=4;flag5=1;break;              //第一行第4列
```

```
        case 0x12:m3=5;flag5=1;break;              //第二行第 1 列
        case 0x22:m3=6;flag5=1;break;              //第二行第 2 列
        case 0x42:m3=7;flag5=1;break;              //第二行第 3 列
        case 0x82:m3=8;flag5=1;break;              //第二行第 4 列

        case 0x14:m3=9;flag5=1;break;              //第三行第 1 列
        case 0x24:m3=0;flag5=1;break;              //第三行第 2 列
        default:break;
    }
}
month=m3*10;
for(i=0;i<1000;i++)
    display_date();

while(1)            //分校准
{
    if(flag6==1)
        break;
    display_date();
    switch(getkeycode())
    {
        case 0x11:n3=1;flag6=1;break;              //第一行第 1 列
        case 0x21:n3=2;flag6=1;break;              //第一行第 2 列
        case 0x41:n3=3;flag6=1;break;              //第一行第 3 列
        case 0x81:n3=4;flag6=1;break;              //第一行第 4 列

        case 0x12:n3=5;flag6=1;break;              //第二行第 1 列
        case 0x22:n3=6;flag6=1;break;              //第二行第 2 列
        case 0x42:n3=7;flag6=1;break;              //第二行第 3 列
        case 0x82:n3=8;flag6=1;break;              //第二行第 4 列

        case 0x14:n3=9;flag6=1;break;              //第三行第 1 列
        case 0x24:n3=0;flag6=1;break;              //第三行第 2 列
        default:break;
    }
}
month=month+n3;
for(i=0;i<1000;i++)
    display_date();

while(1)            //分校准
{
    if(flag7==1)
```

```
        break;
    display_date();
    switch(getkeycode())
    {
        case 0x11:m4=1;flag7=1;break;              //第一行第1列
        case 0x21:m4=2;flag7=1;break;              //第一行第2列
        case 0x41:m4=3;flag7=1;break;              //第一行第3列
        case 0x81:m4=4;flag7=1;break;              //第一行第4列

        case 0x12:m4=5;flag7=1;break;              //第二行第1列
        case 0x22:m4=6;flag7=1;break;              //第二行第2列
        case 0x42:m4=7;flag7=1;break;              //第二行第3列
        case 0x82:m4=8;flag7=1;break;              //第二行第4列

        case 0x14:m4=9;flag7=1;break;              //第三行第1列
        case 0x24:m4=0;flag7=1;break;              //第三行第2列
        default:break;
    }
}
day=m4*10;
for(i=0;i<1000;i++)
    display_date();

while(1)            //分校准
{
    if(flag8==1)
        break;
    display_date();
    switch(getkeycode())
    {
        case 0x11:n4=1;flag8=1;break;              //第一行第1列
        case 0x21:n4=2;flag8=1;break;              //第一行第2列
        case 0x41:n4=3;flag8=1;break;              //第一行第3列
        case 0x81:n4=4;flag8=1;break;              //第一行第4列

        case 0x12:n4=5;flag8=1;break;              //第二行第1列
        case 0x22:n4=6;flag8=1;break;              //第二行第2列
        case 0x42:n4=7;flag8=1;break;              //第二行第3列
        case 0x82:n4=8;flag8=1;break;              //第二行第4列

        case 0x14:n4=9;flag8=1;break;              //第三行第1列
        case 0x24:n4=0;flag8=1;break;              //第三行第2列
        default:break;
    }
```

```
    }
    day=day+n4;
    for(i=0;i<1000;i++)
        display_date();
    flag=0;
    TR1=0;
}

/*********************************闪烁与显示*********************************/
void twinkle(sel,n)
{
    int i;
    for(i=0;i<8;i++)
    {
        XBYTE[0x8000]=  0x01<<i;
        if(i==sel)
        {
            if(n%600<300)
                XBYTE[0x9000]= 0x00;
            else
                XBYTE[0x9000]= show_table[7-i];
        }
        else
            XBYTE[0x9000]= show_table[7-i];
        Delaymini();
        XBYTE[0x9000]= 0x00; //消隐
    }
}

void twinkle_tmr(sel,n)
{
    int i;
    for(i=0;i<8;i++)
    {
        XBYTE[0x8000]=  0x01<<i;
        if(i==sel)
        {
            if(n%600<300)
                XBYTE[0x9000]= 0x00;
            else
                XBYTE[0x9000]= show_table_tmr[7-i];
        }
        else
            XBYTE[0x9000]= show_table_tmr[7-i];
```

```
        Delaymini();
        XBYTE[0x9000]= 0x00; //消隐
    }
}

void twinkle_alm(sel,n)
{
    int i;
    for(i=0;i<8;i++)
    {
        XBYTE[0x8000]=  0x01<<i;
        if(i==sel)
        {
            if(n%600<300)
                XBYTE[0x9000]= 0x00;
            else
                XBYTE[0x9000]= show_table_alm[7-i];
        }
        else
            XBYTE[0x9000]= show_table_alm[7-i];
        Delaymini();
        XBYTE[0x9000]= 0x00; //消隐
    }
}

void display()
{
    int i;
    for(i=0;i<8;i++)
    {
        XBYTE[0x8000]=  0x01<<i;
        XBYTE[0x9000]= show_table[7-i];
        Delaymini();
        XBYTE[0x9000]= 0x00; //消隐
    }
}

void display_tmr()
{
    int i;
    for(i=0;i<8;i++)
    {
        XBYTE[0x8000]=  0x01<<i;
        XBYTE[0x9000]= show_table_tmr[7-i];
```

```
        Delaymini();
        XBYTE[0x9000]= 0x00; //消隐
    }
}

void display_alm()
{
    int i;
    for(i=0;i<8;i++)
    {
        XBYTE[0x8000]=  0x01<<i;
        XBYTE[0x9000]= show_table_alm[7-i];
        Delaymini();
        XBYTE[0x9000]= 0x00; //消隐
    }
}

void display_date()
{
    int i;
    for(i=0;i<8;i++)
    {
        XBYTE[0x8000]=  0x01<<i;
        XBYTE[0x9000]= show_table_date[7-i];
        Delaymini();
        XBYTE[0x9000]= 0x00; //消隐
    }
}

/*********************************按键扫描************************************/
unsigned char getkeycode()                  //键盘扫描函数，返回获得键码
{
    unsigned char line=0x00;                //行码
    unsigned char col=0x00;                 //列码
    unsigned char scancode=0x01;            //行扫描码
    unsigned char keycode;                  //键号

    XBYTE[0x8000]=0xff;
    col=XBYTE[0x8000]&0x0f;                 //从列端口读入四位列码
    if (col==0x00)
        keycode=0x00;
    else
    {
        while((scancode&0x0f)!=0)           //取 scancode 的低四位，没变为全 0，
```

```
循环
        {
            line=scancode;                  //行号
            XBYTE[0x8000]=scancode;         //给行赋扫描码，第一行为 0x01
            if((XBYTE[0x8000]&0x0f)==col)   //检测按键所在的行跳出循环
                break;
            scancode=scancode<<1;           //行扫描码左移一位，转下一行
        }
        col=col<<4;                         //把行码移到高四位
        keycode=col|line;
    }
    return keycode;
}

/**********************************校准函数**********************************/
void reset(void)
{
    int i,n,m1=0,n1=0,m2=0,n2=0,m3=0,n3=0;
    int flag1=0,flag2=0,flag3=0,flag4=0,flag5=0,flag6=0;
    while(1)              //时校准
    {
        if(flag1==1)
        {
            n=0;
            break;
        }
        if(n==6000)                                     //定期清零，防止 n 过大而
溢出
            n=0;
        twinkle(7,n);
        switch(getkeycode())
        {
            case 0x11:m1=1;flag1=1;break;               //第一行第 1 列
            case 0x21:m1=2;flag1=1;break;               //第一行第 2 列
            case 0x24:m1=0;flag1=1;break;               //第三行第 2 列
            default:break;
        }
        n++;
    }
    hour=m1*10+(hour%10);
    for(i=0;i<1000;i++)
        display();

    while(1)
```

```
{
    if(flag2==1)
    {
        n=0;
        break;
    }
    if(n==6000)
        n=0;
    twinkle(6,n);
    switch(getkeycode())
    {
        case 0x11:n1=1;flag2=1;break;              //第一行第 1 列
        case 0x21:n1=2;flag2=1;break;              //第一行第 2 列
        case 0x41:n1=3;flag2=1;break;              //第一行第 3 列
        case 0x81:n1=4;flag2=1;break;              //第一行第 4 列

        case 0x12:n1=5;flag2=1;break;              //第二行第 1 列
        case 0x22:n1=6;flag2=1;break;              //第二行第 2 列
        case 0x42:n1=7;flag2=1;break;              //第二行第 3 列
        case 0x82:n1=8;flag2=1;break;              //第二行第 4 列

        case 0x14:n1=9;flag2=1;break;              //第三行第 1 列
        case 0x24:n1=0;flag2=1;break;              //第三行第 2 列
        default:break;
    }
    n++;
}
hour=(hour/10)*10+n1;
for(i=0;i<1000;i++)
    display();

while(1)            //分校准
{
    if(flag3==1)
    {
        n=0;
        break;
    }
    if(n==6000)
        n=0;
    twinkle(4,n);
    switch(getkeycode())
    {
        case 0x11:m2=1;flag3=1;break;              //第一行第 1 列
```

```
        case 0x21:m2=2;flag3=1;break;              //第一行第 2 列
        case 0x41:m2=3;flag3=1;break;              //第一行第 3 列
        case 0x81:m2=4;flag3=1;break;              //第一行第 4 列
        case 0x12:m2=5;flag3=1;break;              //第二行第 1 列
        case 0x24:m2=0;flag3=1;break;              //第三行第 2 列
        default:break;
    }
    n++;
}
min=m2*10+(min%10);
for(i=0;i<1000;i++)
    display();

while(1)
{
    if(flag4==1)
    {
        n=0;
        break;
    }
    if(n==6000)
        n=0;
    twinkle(3,n);
    switch(getkeycode())
    {
        case 0x11:n2=1;flag4=1;break;              //第一行第 1 列
        case 0x21:n2=2;flag4=1;break;              //第一行第 2 列
        case 0x41:n2=3;flag4=1;break;              //第一行第 3 列
        case 0x81:n2=4;flag4=1;break;              //第一行第 4 列

        case 0x12:n2=5;flag4=1;break;              //第二行第 1 列
        case 0x22:n2=6;flag4=1;break;              //第二行第 2 列
        case 0x42:n2=7;flag4=1;break;              //第二行第 3 列
        case 0x82:n2=8;flag4=1;break;              //第二行第 4 列

        case 0x14:n2=9;flag4=1;break;              //第三行第 1 列
        case 0x24:n2=0;flag4=1;break;              //第三行第 2 列
        default:break;
    }
    n++;
}
min=(min/10)*10+n2;
for(i=0;i<1000;i++)
    display();
```

```
while(1)              //秒校准
{
    if(flag5==1)
    {
        n=0;
        break;
    }
    if(n==6000)
        n=0;
    twinkle(1,n);
    switch(getkeycode())
    {
        case 0x11:m3=1;flag5=1;break;                //第一行第 1 列
        case 0x21:m3=2;flag5=1;break;                //第一行第 2 列
        case 0x41:m3=3;flag5=1;break;                //第一行第 3 列
        case 0x81:m3=4;flag5=1;break;                //第一行第 4 列

        case 0x12:m3=5;flag5=1;break;                //第二行第 1 列
        case 0x24:m3=0;flag5=1;break;                //第三行第 2 列
        default:break;
    }
    n++;
}
sec=m3*10+(sec%10);
for(i=0;i<1000;i++)
    display();

while(1)
{
    if(flag6==1)
    {
        n=0;
        break;
    }
    if(n==6000)
        n=0;
    twinkle(0,n);
    switch(getkeycode())
    {
        case 0x11:n3=1;flag6=1;break;                //第一行第 1 列
        case 0x21:n3=2;flag6=1;break;                //第一行第 2 列
        case 0x41:n3=3;flag6=1;break;                //第一行第 3 列
        case 0x81:n3=4;flag6=1;break;                //第一行第 4 列
```

```
            case 0x12:n3=5;flag6=1;break;                //第二行第 1 列
            case 0x22:n3=6;flag6=1;break;                //第二行第 2 列
            case 0x42:n3=7;flag6=1;break;                //第二行第 3 列
            case 0x82:n3=8;flag6=1;break;                //第二行第 4 列

            case 0x14:n3=9;flag6=1;break;                //第三行第 1 列
            case 0x24:n3=0;flag6=1;break;                //第三行第 2 列
            default:break;
        }
        n++;
    }
    sec=(sec/10)*10+n3;
    num=0;                  //将之前的秒计时清零，从校准结束时刻算起
    for(i=0;i<1000;i++)
        display();
}

/*********************************日期滚动显示*********************************/
void scan(void)
{
    int i,m,k=0;

    date[0]=seg_table[year/1000];
    date[1]=seg_table[(year/100)%10];
    date[2]=seg_table[(year/10)%10];
    date[3]=seg_table[year%10];
    date[4]=0x40;
    date[5]=seg_table[month/10];
    date[6]=seg_table[month%10];
    date[7]=0x40;
    date[8]=seg_table[day/10];
    date[9]=seg_table[day%10];

    for(k=0;k<36;k++)
        for(m=0;m<200;m++)
            for(i=0;i<8;i++)
            {
                XBYTE[0x8000]=0x01<<i;
                switch(i)
                {
                    case(0):if((k-i)<0) XBYTE[0x9000]= 0x00; else XBYTE[0x9000]=
date[(k+i)%12];Delaymini();break;
                    case(1):if((k-i)<0) XBYTE[0x9000]= 0x00; else XBYTE[0x9000]=
```

```
date[(k+i-2)%12];Delaymini();break;
                    case(2):if((k-i)<0) XBYTE[0x9000]= 0x00; else XBYTE[0x9000]=
date[(k+i-4)%12];Delaymini();break;
                    case(3):if((k-i)<0) XBYTE[0x9000]= 0x00; else XBYTE[0x9000]=
date[(k+i-6)%12];Delaymini();break;
                    case(4):if((k-i)<0) XBYTE[0x9000]= 0x00; else XBYTE[0x9000]=
date[(k+i-8)%12];Delaymini();break;
                    case(5):if((k-i)<0) XBYTE[0x9000]= 0x00; else XBYTE[0x9000]=
date[(k+i-10)%12];Delaymini();break;
                    case(6):if((k-i)<0) XBYTE[0x9000]= 0x00; else XBYTE[0x9000]=
date[(k+i-12)%12];Delaymini();break;
                    case(7):if((k-i)<0) XBYTE[0x9000]= 0x00; else XBYTE[0x9000]=
date[(k+i-14)%12];Delaymini();break;
                }
                XBYTE[0x9000]= 0x00;
            }
}

/*************************************秒表*************************************/
void stopwatch()
{
    int i=0;
    flag=1;
    TR1=1;
    num2=0;
    min2=0;
    sec2=0;
    while(1)
    {
        switch(getkeycode())
        {
            case 0x11: stw=1;break;  //第一行第1列      开始
            case 0x21: stw=2;break;  //第一行第2列      暂停
            case 0x41: stw=3;break;  //第一行第3列      复位
            case 0x81: stw=4;break;  //第一行第4列      退出
            default:break;
        }
        for(i=0;i<7;i++)
        {
            XBYTE[0x8000]=0x01<<i;
            XBYTE[0x9000]=show_table_stw[6-i];
            Delaymini();
            XBYTE[0x9000]=0x00; //消隐
        }
```

```
        if(stw==4)
        {
            TR1=0;
            stw=0;
            flag=0;
            break;
        }
    }
}

/*************************************定时器*************************************/
void mytimer()
{
    int i,n,m1=0,n1=0,m2=0,n2=0,m3=0,n3=0;
    int flag1=0,flag2=0,flag3=0,flag4=0,flag5=0,flag6=0;
    flag=2;
    tmr=1;
    TR1=1;

    while(1)
    {
        switch(getkeycode())
        {
            case 0x12: tmr=1;break;  //第二行第 1 列      复位(跳回预置时间模式)
            case 0x22: tmr=2;break;  //第二行第 2 列      启动
            case 0x42: tmr=3;break;  //第二行第 3 列      暂停
            case 0x82: flag=0;tmr=0;break;   //第二行第 4 列      退出
            default:break;
        }
        if(tmr==0)
            break;
        if(tmr==1)
        {
            while(1)              //时校准
            {
                if(flag1==1)
                {
                    n=0;
                    flag1=0;
                    break;
                }
                if(n==6000)     //定期清零，防止 n 过大而溢出
                    n=0;
                twinkle_tmr(7,n);
```

```
switch(getkeycode())
            {
                case 0x11:m1=1;flag1=1;break;              //第一行第 1 列
                case 0x21:m1=2;flag1=1;break;              //第一行第 2 列
                case 0x24:m1=0;flag1=1;break;              //第三行第 2 列
                default:break;
            }
            n++;
        }
        hour_tmr=m1*10+(hour_tmr%10);
        for(i=0;i<1000;i++)
            display_tmr();

        while(1)
        {
            if(flag2==1)
            {
                n=0;
                flag2=0;
                break;
            }
            if(n==6000)
                n=0;
            twinkle_tmr(6,n);
            switch(getkeycode())
            {
                case 0x11:n1=1;flag2=1;break;              //第一行第 1 列
                case 0x21:n1=2;flag2=1;break;              //第一行第 2 列
                case 0x41:n1=3;flag2=1;break;              //第一行第 3 列
                case 0x81:n1=4;flag2=1;break;              //第一行第 4 列

                case 0x12:n1=5;flag2=1;break;              //第二行第 1 列
                case 0x22:n1=6;flag2=1;break;              //第二行第 2 列
                case 0x42:n1=7;flag2=1;break;              //第二行第 3 列
                case 0x82:n1=8;flag2=1;break;              //第二行第 4 列

                case 0x14:n1=9;flag2=1;break;              //第三行第 1 列
                case 0x24:n1=0;flag2=1;break;              //第三行第 2 列
                default:break;
            }
            n++;
        }
        hour_tmr=(hour_tmr/10)*10+n1;
        for(i=0;i<1000;i++)
```

```
    display_tmr();

while(1)            //分校准
{
    if(flag3==1)
    {
        n=0;
        flag3=0;
        break;
    }
    if(n==6000)
        n=0;
    twinkle_tmr(4,n);
    switch(getkeycode())
    {
        case 0x11:m2=1;flag3=1;break;              //第一行第 1 列
        case 0x21:m2=2;flag3=1;break;              //第一行第 2 列
        case 0x41:m2=3;flag3=1;break;              //第一行第 3 列
        case 0x81:m2=4;flag3=1;break;              //第一行第 4 列
        case 0x12:m2=5;flag3=1;break;              //第二行第 1 列
        case 0x24:m2=0;flag3=1;break;              //第三行第 2 列
        default:break;
    }
    n++;
}
min_tmr=m2*10+(min_tmr%10);
for(i=0;i<1000;i++)
    display_tmr();

while(1)
{
    if(flag4==1)
    {
        n=0;
        flag4=0;
        break;
    }
    if(n==6000)
        n=0;
    twinkle_tmr(3,n);
    switch(getkeycode())
    {
        case 0x11:n2=1;flag4=1;break;                //第一行第 1 列
        case 0x21:n2=2;flag4=1;break;                //第一行第 2 列
```

```
            case 0x41:n2=3;flag4=1;break;                //第一行第 3 列
            case 0x81:n2=4;flag4=1;break;                //第一行第 4 列

            case 0x12:n2=5;flag4=1;break;                //第二行第 1 列
            case 0x22:n2=6;flag4=1;break;                //第二行第 2 列
            case 0x42:n2=7;flag4=1;break;                //第二行第 3 列
            case 0x82:n2=8;flag4=1;break;                //第二行第 4 列

            case 0x14:n2=9;flag4=1;break;                //第三行第 1 列
            case 0x24:n2=0;flag4=1;break;                //第三行第 2 列
            default:break;
        }
        n++;
    }
    min_tmr=(min_tmr/10)*10+n2;
    for(i=0;i<1000;i++)
        display_tmr();

    while(1)              //秒校准
    {
        if(flag5==1)
        {
            n=0;
            flag5=0;
            break;
        }
        if(n==6000)
            n=0;
        twinkle_tmr(1,n);
        switch(getkeycode())
        {
            case 0x11:m3=1;flag5=1;break;                //第一行第 1 列
            case 0x21:m3=2;flag5=1;break;                //第一行第 2 列
            case 0x41:m3=3;flag5=1;break;                //第一行第 3 列
            case 0x81:m3=4;flag5=1;break;                //第一行第 4 列

            case 0x12:m3=5;flag5=1;break;                //第二行第 1 列
            case 0x24:m3=0;flag5=1;break;                //第三行第 2 列
            default:break;
        }
        n++;
    }
    sec_tmr=m3*10+(sec_tmr%10);
    for(i=0;i<1000;i++)
```

```
            display_tmr();

        while(1)
        {
            if(flag6==1)
            {
                n=0;
                flag6=0;
                break;
            }
            if(n==6000)
                n=0;
            twinkle_tmr(0,n);
            switch(getkeycode())
            {
                case 0x11:n3=1;flag6=1;break;                //第一行第 1 列
                case 0x21:n3=2;flag6=1;break;                //第一行第 2 列
                case 0x41:n3=3;flag6=1;break;                //第一行第 3 列
                case 0x81:n3=4;flag6=1;break;                //第一行第 4 列

                case 0x12:n3=5;flag6=1;break;                //第二行第 1 列
                case 0x22:n3=6;flag6=1;break;                //第二行第 2 列
                case 0x42:n3=7;flag6=1;break;                //第二行第 3 列
                case 0x82:n3=8;flag6=1;break;                //第二行第 4 列

                case 0x14:n3=9;flag6=1;break;                //第三行第 1 列
                case 0x24:n3=0;flag6=1;break;                //第三行第 2 列
                default:break;
            }
            n++;
        }
        sec_tmr=(sec_tmr/10)*10+n3;
        num_tmr=0;                  //将之前的秒计时清零，从校准结束时刻算起
        for(i=0;i<1000;i++)
            display_tmr();
    tmr=3;
    }

    display_tmr();
}
if(flag_music==1)
{
    flag_music=0;
    music();
```

```
    }
}
/*****************************显示闹钟设定值*****************************/
void show_alm()                                    //为闹钟 1 的显示数组赋值
{
    show_table_alm[0]=seg_table[hour_alm/10];
    show_table_alm[1]=seg_table[hour_alm%10];
    show_table_alm[2]=0x40;
    show_table_alm[3]=seg_table[min_alm/10];
    show_table_alm[4]=seg_table[min_alm%10];
    show_table_alm[5]=0x40;
    show_table_alm[6]=seg_table[sec_alm/10];
    show_table_alm[7]=seg_table[sec_alm%10];
}

void show_alm_2()                                  //为闹钟 2 的显示数组赋值
{
    show_table_alm[0]=seg_table[hour_alm_2/10];
    show_table_alm[1]=seg_table[hour_alm_2%10];
    show_table_alm[2]=0x40;
    show_table_alm[3]=seg_table[min_alm_2/10];
    show_table_alm[4]=seg_table[min_alm_2%10];
    show_table_alm[5]=0x40;
    show_table_alm[6]=seg_table[sec_alm_2/10];
    show_table_alm[7]=seg_table[sec_alm_2%10];
}

/*********************************闹钟*********************************/
void alarm()
{
    int i,n,m=0,l=0,m1=0,n1=0,m2=0,n2=0,m3=0,n3=0;
    int p1=0,q1=0,p2=0,q2=0,p3=0,q3=0;
    int flag1=0,flag2=0,flag3=0,flag4=0,flag5=0,flag6=0;
    int flag11=0,flag22=0;
    int mode=0;
    hour_alm=0;min_alm=0;sec_alm=0;
    hour_alm_2=0;min_alm_2=0;sec_alm_2=0;

    while(1)
    {
        for(i=0;i<15000;i++);
        switch(getkeycode())
        {
            case 0x11: mode=1;break;      //闹钟 1
```

```
        case 0x21: mode=2;break;      //闹钟 2
        case 0x41: mode=3;break;      //启动
        case 0x81: mode=4;break;      //取消
        default: break;
    }


    if(mode==1)                       //模式 1 用于设置闹钟 1 的时间
    {
        for(i=0;i<15000;i++);
        show_alm();
        display_alm();
        while(1)             //时校准
        {
            if(flag1==1)
            {
                n=0;
                flag1=0;
                break;
            }
            if(n==6000)     //定期清零，防止 n 过大而溢出
                n=0;
            show_alm();
            twinkle_alm(7,n);
switch(getkeycode())
            {
                case 0x11:m1=1;flag1=1;break;              //第一行第 1 列
                case 0x21:m1=2;flag1=1;break;              //第一行第 2 列
                case 0x24:m1=0;flag1=1;break;              //第三行第 2 列
                default:break;
            }
            n++;
        }
        hour_alm=m1*10;
        show_alm();
        for(i=0;i<1000;i++)
            display_alm();

        while(1)
        {
            if(flag2==1)
            {
                n=0;
                flag2=0;
```

```
            break;
        }
        if(n==6000)
            n=0;
        twinkle_alm(6,n);
        switch(getkeycode())
        {
            case 0x11:n1=1;flag2=1;break;              //第一行第 1 列
            case 0x21:n1=2;flag2=1;break;              //第一行第 2 列
            case 0x41:n1=3;flag2=1;break;              //第一行第 3 列
            case 0x81:n1=4;flag2=1;break;              //第一行第 4 列

            case 0x12:n1=5;flag2=1;break;              //第二行第 1 列
            case 0x22:n1=6;flag2=1;break;              //第二行第 2 列
            case 0x42:n1=7;flag2=1;break;              //第二行第 3 列
            case 0x82:n1=8;flag2=1;break;              //第二行第 4 列

            case 0x14:n1=9;flag2=1;break;              //第三行第 1 列
            case 0x24:n1=0;flag2=1;break;              //第三行第 2 列
            default:break;
        }
        n++;
    }
    hour_alm=hour_alm+n1;
    show_alm();
    for(i=0;i<1000;i++)
        display_alm();

    while(1)            //分校准
    {
        if(flag3==1)
        {
            n=0;
            flag3=0;
            break;
        }
        if(n==6000)
            n=0;
        twinkle_alm(4,n);
        switch(getkeycode())
        {
            case 0x11:m2=1;flag3=1;break;              //第一行第 1 列
            case 0x21:m2=2;flag3=1;break;              //第一行第 2 列
            case 0x41:m2=3;flag3=1;break;              //第一行第 3 列
```

```
        case 0x81:m2=4;flag3=1;break;              //第一行第 4 列
        case 0x12:m2=5;flag3=1;break;              //第二行第 1 列
        case 0x24:m2=0;flag3=1;break;              //第三行第 2 列
        default:break;
    }
    n++;
}
min_alm=m2*10;
show_alm();
for(i=0;i<1000;i++)
    display_alm();

while(1)
{
    if(flag4==1)
    {
        n=0;
        flag4=0;
        break;
    }
    if(n==6000)
        n=0;
    twinkle_alm(3,n);
    switch(getkeycode())
    {
        case 0x11:n2=1;flag4=1;break;              //第一行第 1 列
        case 0x21:n2=2;flag4=1;break;              //第一行第 2 列
        case 0x41:n2=3;flag4=1;break;              //第一行第 3 列
        case 0x81:n2=4;flag4=1;break;              //第一行第 4 列

        case 0x12:n2=5;flag4=1;break;              //第二行第 1 列
        case 0x22:n2=6;flag4=1;break;              //第二行第 2 列
        case 0x42:n2=7;flag4=1;break;              //第二行第 3 列
        case 0x82:n2=8;flag4=1;break;              //第二行第 4 列

        case 0x14:n2=9;flag4=1;break;              //第三行第 1 列
        case 0x24:n2=0;flag4=1;break;              //第三行第 2 列
        default:break;
    }
    n++;
}
min_alm=min_alm+n2;
show_alm();
for(i=0;i<1000;i++)
```

```
    display_alm();

while(1)              //秒校准
{
    if(flag5==1)
    {
        n=0;
        flag5=0;
        break;
    }
    if(n==6000)
        n=0;
    twinkle_alm(1,n);
    switch(getkeycode())
    {
        case 0x11:m3=1;flag5=1;break;                //第一行第 1 列
        case 0x21:m3=2;flag5=1;break;                //第一行第 2 列
        case 0x41:m3=3;flag5=1;break;                //第一行第 3 列
        case 0x81:m3=4;flag5=1;break;                //第一行第 4 列

        case 0x12:m3=5;flag5=1;break;                //第二行第 1 列
        case 0x24:m3=0;flag5=1;break;                //第三行第 2 列
        default:break;
    }
    n++;
}
sec_alm=m3*10;
show_alm();
for(i=0;i<1000;i++)
    display_alm();

while(1)
{
    if(flag6==1)
    {
        n=0;
        flag6=0;
        break;
    }
    if(n==6000)
        n=0;
    twinkle_alm(0,n);
    switch(getkeycode())
    {
```

```
                case 0x11:n3=1;flag6=1;break;                //第一行第 1 列
                case 0x21:n3=2;flag6=1;break;                //第一行第 2 列
                case 0x41:n3=3;flag6=1;break;                //第一行第 3 列
                case 0x81:n3=4;flag6=1;break;                //第一行第 4 列

                case 0x12:n3=5;flag6=1;break;                //第二行第 1 列
                case 0x22:n3=6;flag6=1;break;                //第二行第 2 列
                case 0x42:n3=7;flag6=1;break;                //第二行第 3 列
                case 0x82:n3=8;flag6=1;break;                //第二行第 4 列

                case 0x14:n3=9;flag6=1;break;                //第三行第 1 列
                case 0x24:n3=0;flag6=1;break;                //第三行第 2 列
                default:break;
            }
            n++;
        }
        sec_alm=sec_alm+n3;
        show_alm();
        for(i=0;i<1000;i++)
            display_alm();
        mode=0;
        flag11=1;
    }

    if(mode==2)                                  //模式 2 用于设置闹钟 2 的时间
    {
        for(i=0;i<15000;i++);
        show_alm_2();
        display_alm();
        while(1)              //时校准
        {
            if(flag1==1)
            {
                n=0;
                flag1=0;
                break;
            }
            if(n==6000)
                n=0;
            show_alm_2();
            twinkle_alm(7,n);
switch(getkeycode())
            {
                case 0x11:p1=1;flag1=1;break;                //第一行第 1 列
```

```
            case 0x21:p1=2;flag1=1;break;              //第一行第 2 列
            case 0x24:p1=0;flag1=1;break;              //第三行第 2 列
            default:break;
        }
        n++;
    }
    hour_alm_2=p1*10;
    show_alm_2();
    for(i=0;i<1000;i++)
        display_alm();

    while(1)
    {
        if(flag2==1)
        {
            n=0;
            flag2=0;
            break;
        }
        if(n==6000)
            n=0;
        twinkle_alm(6,n);
        switch(getkeycode())
        {
            case 0x11:q1=1;flag2=1;break;              //第一行第 1 列
            case 0x21:q1=2;flag2=1;break;              //第一行第 2 列
            case 0x41:q1=3;flag2=1;break;              //第一行第 3 列
            case 0x81:q1=4;flag2=1;break;              //第一行第 4 列

            case 0x12:q1=5;flag2=1;break;              //第二行第 1 列
            case 0x22:q1=6;flag2=1;break;              //第二行第 2 列
            case 0x42:q1=7;flag2=1;break;              //第二行第 3 列
            case 0x82:q1=8;flag2=1;break;              //第二行第 4 列

            case 0x14:q1=9;flag2=1;break;              //第三行第 1 列
            case 0x24:q1=0;flag2=1;break;              //第三行第 2 列
            default:break;
        }
        n++;
    }
    hour_alm_2=hour_alm_2+q1;
    show_alm_2();
    for(i=0;i<1000;i++)
        display_alm();
```

```
while(1)            //分校准
{
    if(flag3==1)
    {
        n=0;
        flag3=0;
        break;
    }
    if(n==6000)
        n=0;
    twinkle_alm(4,n);
    switch(getkeycode())
    {
        case 0x11:p2=1;flag3=1;break;              //第一行第 1 列
        case 0x21:p2=2;flag3=1;break;              //第一行第 2 列
        case 0x41:p2=3;flag3=1;break;              //第一行第 3 列
        case 0x81:p2=4;flag3=1;break;              //第一行第 4 列
        case 0x12:p2=5;flag3=1;break;              //第二行第 1 列
        case 0x24:p2=0;flag3=1;break;              //第三行第 2 列
        default:break;
    }
    n++;
}
min_alm_2=p2*10;
show_alm_2();
for(i=0;i<1000;i++)
    display_alm();

while(1)
{
    if(flag4==1)
    {
        n=0;
        flag4=0;
        break;
    }
    if(n==6000)
        n=0;
    twinkle_alm(3,n);
    switch(getkeycode())
    {
        case 0x11:q2=1;flag4=1;break;               //第一行第 1 列
        case 0x21:q2=2;flag4=1;break;               //第一行第 2 列
```

```
            case 0x41:q2=3;flag4=1;break;                //第一行第 3 列
            case 0x81:q2=4;flag4=1;break;                //第一行第 4 列

            case 0x12:q2=5;flag4=1;break;                //第二行第 1 列
            case 0x22:q2=6;flag4=1;break;                //第二行第 2 列
            case 0x42:q2=7;flag4=1;break;                //第二行第 3 列
            case 0x82:q2=8;flag4=1;break;                //第二行第 4 列

            case 0x14:q2=9;flag4=1;break;                //第三行第 1 列
            case 0x24:q2=0;flag4=1;break;                //第三行第 2 列
            default:break;
        }
        n++;
    }
    min_alm_2=min_alm_2+q2;
    show_alm_2();
    for(i=0;i<1000;i++)
        display_alm();

    while(1)              //秒校准
    {
        if(flag5==1)
        {
            n=0;
            flag5=0;
            break;
        }
        if(n==6000)
            n=0;
        twinkle_alm(1,n);
        switch(getkeycode())
        {
            case 0x11:p3=1;flag5=1;break;                //第一行第 1 列
            case 0x21:p3=2;flag5=1;break;                //第一行第 2 列
            case 0x41:p3=3;flag5=1;break;                //第一行第 3 列
            case 0x81:p3=4;flag5=1;break;                //第一行第 4 列

            case 0x12:p3=5;flag5=1;break;                //第二行第 1 列
            case 0x24:p3=0;flag5=1;break;                //第三行第 2 列
            default:break;
        }
        n++;
    }
    sec_alm_2=p3*10;
```

```
    show_alm_2();
    for(i=0;i<1000;i++)
        display_alm();

    while(1)
    {
        if(flag6==1)
        {
            n=0;
            flag6=0;
            break;
        }
        if(n==6000)
            n=0;
        twinkle_alm(0,n);
        switch(getkeycode())
        {
            case 0x11:q3=1;flag6=1;break;                //第一行第 1 列
            case 0x21:q3=2;flag6=1;break;                //第一行第 2 列
            case 0x41:q3=3;flag6=1;break;                //第一行第 3 列
            case 0x81:q3=4;flag6=1;break;                //第一行第 4 列

            case 0x12:q3=5;flag6=1;break;                //第二行第 1 列
            case 0x22:q3=6;flag6=1;break;                //第二行第 2 列
            case 0x42:q3=7;flag6=1;break;                //第二行第 3 列
            case 0x82:q3=8;flag6=1;break;                //第二行第 4 列

            case 0x14:q3=9;flag6=1;break;                //第三行第 1 列
            case 0x24:q3=0;flag6=1;break;                //第三行第 2 列
            default:break;
        }
        n++;
    }
    sec_alm_2=sec_alm_2+q3;
    show_alm_2();
    for(i=0;i<1000;i++)
        display_alm();
    mode=0;
    flag22=1;
}

if(mode==3)                                  //模式 3 用于启动闹钟
{
mode=0;
```

```
    if((flag11==1)&&(flag22==0))            //判断启动两个闹钟中的哪一个
        while(1)
        {
            display();
            if(hour==hour_alm&&min==min_alm&&sec==sec_alm)
            {
                music();
                break;
            }
        }

    if((flag11==0)&&(flag22==1))
        while(1)
        {
            display();
            if(hour==hour_alm_2&&min==min_alm_2&&sec==sec_alm_2)
            {
                music_2();
                break;
            }
        }
    if((flag11==1)&&(flag22==1))
        while(1)
        {
            display();
            if(m==1&&l==1)
                break;
            if(hour==hour_alm&&min==min_alm&&sec==sec_alm&&l==0)
            {
                l=1;
                music();
            }
            if(hour==hour_alm_2&&min==min_alm_2&&sec==sec_alm_2&&m==0)
            {
                m=1;
                music_2();
            }
        }
    break;
    }

    if(mode==4)                         //退出
        break;
}
```

```
}


/*********************************音乐提示音*********************************/
void music(void)
{
    flag=4;
    k=0;
    TH1=0xf8;
    TL1=0x8b;
    TR1=1;
    while(1)
    {
        display();
        if(getkeycode()==0x82)
        {
            flag=0;
            break;
        }
    }
    beep=1;
    TR1=0;
}

void music_2(void)
{
    flag=3;
    k_2=0;
    TH1=0xfb;
    TL1=0x05;
    TR1=1;
    while(1)
    {
        display();
        if(getkeycode()==0x82)
        {
            flag=0;
            break;
        }
    }
    beep=1;
    TR1=0;
}
```