一、实验目的

1. 学习与理解 51 单片机的结构、工作原理;
2. 掌握单片机定时器和中断系统相关知识;
3. 进一步熟悉 51 单片机开发环境，掌握调试方法。

二、实验任务

在实验板上实现实时数字钟和万年历。要求

1) “hh-mm-ss”方式显示时钟; （已在上次实验完成）
2) 可以通过按键置入时间值; （已在上次实验完成）
3) 显示日期时可以从右向左滚动显示“yyyy-mm-dd”三次。（已在上次实验完成）
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、分模块实现方案分析

3.1 音乐播放

闹钟与倒计时器功能中均需要实现音乐播放功能。

3.1.1 音乐播放原理

播放音乐使用蜂鸣器发声，主要涉及音长与音调的控制。

3.1.1.1 音调的控制

音调的控制使用 timer1。要发出频率为 440Hz 的声音，则令与蜂鸣器相接的管脚（P1.6）产生频率为 440Hz 的方波即可。为此设置 timer1 频率为 880Hz 即 0 可。可按照(65535-X)*12/11.0592us = Yms 算出 timer1 置数初值，需要播放 440Hz 音调时，置入该值即可。

3.1.1.2 音长的控制

音长的控制使用 timer0。由于 timer0 还用来计时，因此其计时长度 50ms 无法改变，因此定义四分之一拍的时间长度为 n×50ms，半拍为 2×n×50ms，四分之三拍为 3×n×50ms……以此类推，可控制音长。由于不同乐谱四分之一拍的时间长度不尽相同，为简化程序，取四分之一拍长度为 4×50=200ms。

3.1.2 音乐播放流程图

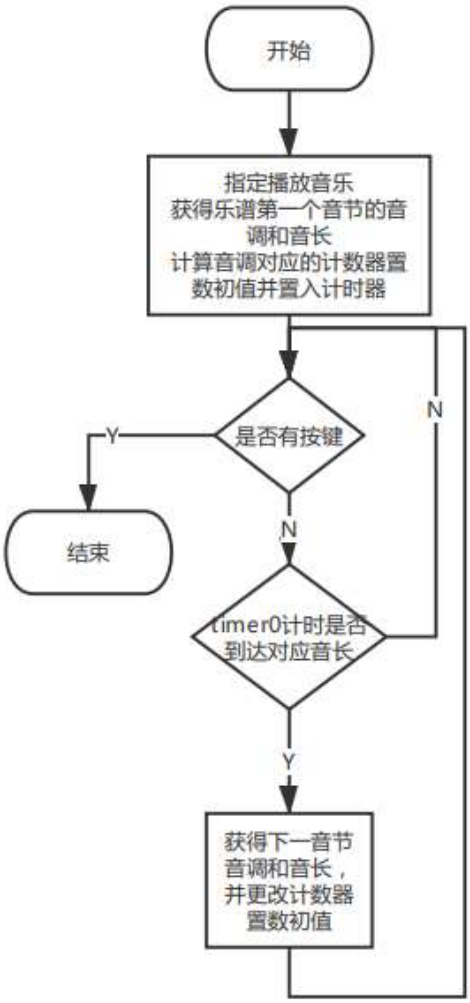

图 1 音乐播放流程图

3.2 闹钟

3.2.1 闹钟功能分析

a) 时间设置。

b) 铃声设置。

c) 闹钟开启与关闭的设置。

d) 响铃与停止。每分钟将时钟时间与闹钟设置时间对比，如果相同则响铃。

进入闹钟模式后，可以设置时间、铃声和开启与关闭。八位 LED 以“X--HH-MM”格式显示：X 由 0~9，设置闹钟的开启、关闭和铃声；HH-MM 设置闹钟时间。设置位闪烁，用两个按键 L 和 R 实现设置位切换和设置位值增加。

如果闹钟开启，则时钟时间+1s 事件发生的同时，检测时钟时间与闹钟时间是否相同，如果相同，则响铃。音乐播放后，按任意键停止闹钟。

3.2.2 闹钟流程图

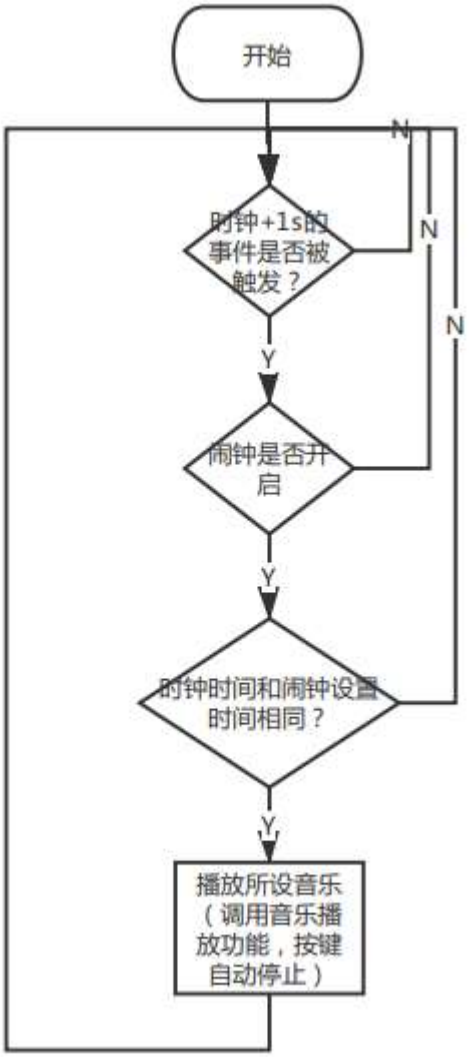

图 2 闹钟流程图

3.3 倒计时器

3.3.1 倒计时器功能分析

有两个按键可用，需要实现定时器设置和开启，采用如下显示方式：X-MMM-SS。其中 X 用来控制计时开关和选歌。0 表示停止，1~9 用来选歌；MMM 表示分钟，最高支持 250 分钟； SS 表示秒，范围 0~59。两个按键 L 和 R，具有如下功能：L 切换设置位，R 对设置位自增。倒是计时器共有两个状态：设置，计时（应支持后台计时），当 X 不为 0，即进入计时状态。

3.3.2 倒计时器误差分析

由于需要倒计时器和时钟使用同一个 timer0_count(timer0 的计数变量，timer0_count 达到 20 时，时钟+1s),而在开始计时时，timer0_count 不一定为 0，因此可能会产生 0~1 秒的误差。可认为误差服从均匀分布即 E_second~U(0,1)。

3.3.3 倒计时器流程图

倒计时器原理与时钟原理完全相同，仅将+1s 改为-1s。时间设置、音乐播放也在前文尽述，此处省略流程图。

3.4 秒表

3.4.1 秒表功能分析

进入秒表功能后有两个按键可用，需要实现:开始计时，暂停显示，恢复显示，停止计时，清零五个功能。秒表共有三个状态:闲置状态（停止状态），计时状态，暂停状态。因此，按键功能和状态切换表格如下表所示。

起始状态	按键功能 (对应按键)	到达状态	备注
闲置	开始计时(L)	计时	从当前值开始计时。
闲置	清零(R)	闲置	恢复到 0。
计时	暂停显示(L)	暂停	显示值固定，但后台继续计时。
计时	停止计时(R)	闲置	显示值固定，后台也停止计时。
暂停	暂停显示(L)	暂停	显示值恢复到后台计时值，但固定于此值，类似于读完第一名成绩后，接着直接读第二名成绩。
暂停	恢复显示(R)	计时	显示值恢复到后台计时值，并实时显示计时值。

3.4.2 秒表误差分析

timer0 为时钟所用计时 50ms 无法改变，达不到秒表所需百分之一秒的精度。若要将 timer1 设置成 5ms 或 10ms，会使 LED 显示频率过低。为了使用 timer1 的同时不影响显示功能，timer1 的中断时间 2ms 无法改变。但此处按照公式

$$X = 65535 - Y * 11059.2/12$$

计算得到的 X 并非整数。此处 Y 为中断时间 2（ms），X 为 timer1 的(TH)(TL)置数初值（十进制），计算结果为 X=63691.8，因此若将置数(TH)(TL)置为 63692，则每 2ms 产生

$$0.2/1843.2 \approx 0.00011\text{ms}$$

的误差，60s 则会产生 3.26s 误差。

该误差并非不可避免，可以通过置四个 63692 后置一个 63691 来消除此误差。但在本次验收时尚未消除此误差。

3.4.3 秒表状态机图

因秒表共三个状态，使用状态机图比流程图更易表明秒表工作方式和流程。

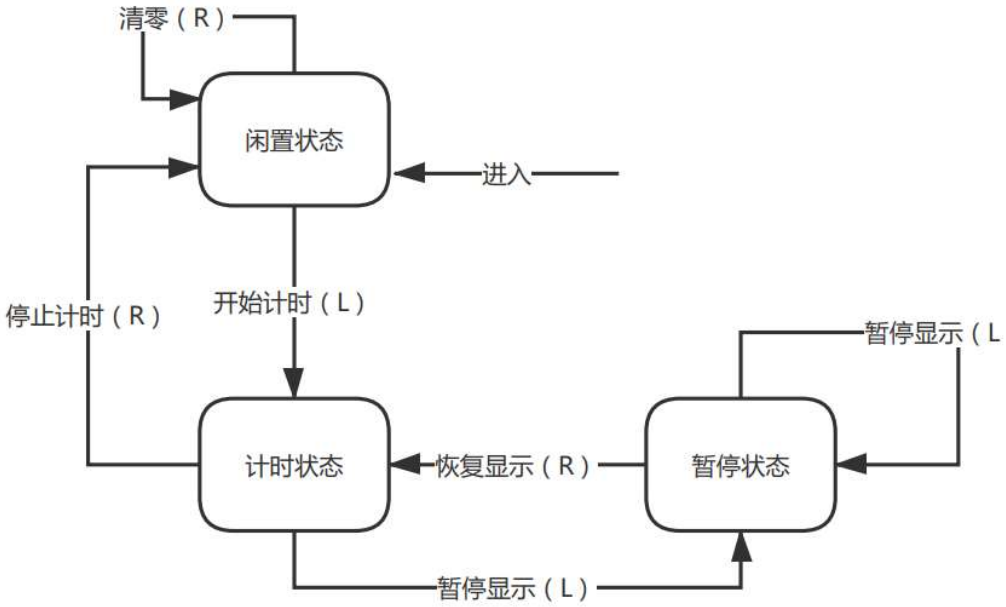

图 3 秒表状态机图

四、程序优点分析

1. **中断函数十分简洁。**除响音乐时涉及对 p1.6 管脚的输出外，没有其他诸如 LED 显示、按键检测、时钟自增一秒等功能（以上功能均写在主函数中），仅涉及 timerx 的 flag、count 值改变。
2. **没有使用 for 循环进行延迟**，即没有调用 demo 中的 delay 函数。因此即使响音乐时也不会时状态卡在某处，且因为使用 timer1 控制音乐音长，音乐音长十分准确。且响铃时除对秒表有影响外，对其他功能如时间自增、倒计时器自减等功能均无影响。
3. **秒表达到百分秒精度。**

4. **无论在什么模式下，闹钟都会发挥作用，不会因为处于日期设置模式、秒表模式等而错过闹钟。**

五、资源使用情况

5.1 存储器使用情况

Program Size: data=9.0 xdata=336 code=5743

5.2 按键使用情况：

使用三个按键，第三行第一列、第三行第二列（L 键），第三行第三列（R 键）。

5.3 定时器及中断使用情况：

使用 timer0、timer1 及对应的中断 1、中断 3。

六、源代码、开发日志

由于篇幅限制，报告中仅贴出部分代码，全部源代码过于冗长，详情可见个人 github 仓库 github.com/cantjie：
https://github.com/cantjie/EELC530705-Electronic-System-Design1

6.1 主函数

```
void main() {
    unsigned char keycode = 0x00;
    unsigned char key_input_break = 0x00;

    timerAndInterruptInit();

    while(1) {

        //1s has passed, change time and check whether it's the clock time;
        if (G_timer0_count == TIMER0_SECOND_COUNT) {
            timeIncrease(G_pointer_current_time);
            G_timer0_count = 0;

            // if clock is set, check if it's time to play music.
            if (G_pointer_clock->music != 0) {
                if (isTimeEqualsClock(G_pointer_current_time, G_pointer_clock) == 1)
{
                    startPlayMusic(G_pointer_clock->music);
                }
            }

            // if countdown timer is set and is counting, then countdown by 1 second
            // and check if it's time to play music.
            if (G_countdown_timer_state == COUNTDOWN_TIMER_COUNTING) {
                countdownTimerDecreaseOrPlayMusic(G_pointer_countdown_timer);
            }
        }

        //play music
        if (G_music_playing_ptr) {
            if (G_timer0_music_count == TIMER0_BEAT_LENGTH * G_beat_length) {
```

```
                G_timer0_music_count = 0;
                continuePlayMusic();
            }
        }

        //50ms passed, check if any key being pressed
        if (G_timer0_flag) {
        //todo. why if the next line is used instead of the former line, the program
wont work properly.
        //if ((G_timer0_count & 0x01) == 0x01) {
            keycode = getKeycode();
            if (keycode) {
                if (G_key_pressing_flag) {
                    G_keycode = 0;
                }
                else {
                    G_key_pressing_flag = 1;
                    G_keycode = keycode;
                }
            }
            else {
                G_key_pressing_flag = 0;
            }
            G_timer0_flag = 0;
        }

        // 2ms passed , excute routine work(display next 7-seg-led)
        if (G_timer1_flag) {
            execute(G_mode);
            G_timer1_flag = 0;
        }

        //if key pressed
        if (G_keycode) {
            if (G_music_playing_ptr) {
                // if music is playing, then only to stop it.
                G_music_playing_ptr = 0x00;
                beep_pin = 1;
            }
            else{
                dealWithKeyPressed(G_keycode);
            }
            G_keycode = 0;
        }
    }

    // never reaches here;
    return ;
}
```

6.2 时钟中断函数

```
void timer0() interrupt 1 {
    TH0 = 0x4B;       // load 19455D to timer0, 50ms.
    TL0 = 0xFF;
    G_timer0_count++;
    G_timer0_flag = 1;
    if (G_music_playing_ptr) {
        G_timer0_music_count++;
    }
}

void timer1() interrupt 3{
    if (G_music_playing_ptr) {
```

```
            TH1 = G_TIMER1_TH;
            TL1 = G_TIMER1_TL;
            if (G_TIMER1_TH == 0x00 && G_TIMER1_TL == 0x00) {
                beep_pin = 1;
            }
            else {
                beep_pin = !beep_pin;
            }
        }
        else {
            TH1 = 0xF8;       // load 63692 to timer1 , about 2ms
            TL1 = 0xCC;       // in stopwatch mode, the time displayed is about 3 seconds
faster than real time per minute
        }
        G_timer1_count++;
        G_timer1_flag = 1;
}
```

6.3 音乐播放功能

```
void startPlayMusic(unsigned char music_num) {

    switch (music_num)
    {
    case 1:
        G_music_playing_ptr = &music_mother;
        break;
    case 2:
        G_music_playing_ptr = &music_if_love;
    default:
        break;
    }
    G_music_playing_ptr->playing_index = 0;
    G_beat_length = G_music_playing_ptr->beat_len[0];
    G_TIMER1_TH = G_score_to_TH[G_music_playing_ptr->score[0] - 1];
    G_TIMER1_TL = G_score_to_TL[G_music_playing_ptr->score[0] - 1];
    G_timer0_music_count = 0;
}

void continuePlayMusic() {
    G_music_playing_ptr->playing_index >= G_music_playing_ptr->len ?
G_music_playing_ptr->playing_index = 0 : G_music_playing_ptr->playing_index++;

    G_beat_length = G_music_playing_ptr->beat_len[G_music_playing_ptr->playing_index];
    G_TIMER1_TH =
G_score_to_TH[G_music_playing_ptr->score[G_music_playing_ptr->playing_index] - 1];
    G_TIMER1_TL =
G_score_to_TL[G_music_playing_ptr->score[G_music_playing_ptr->playing_index] - 1];
}
```