嵌入式处理器编程#F实验

班级：2021211316

学号：2021211419

姓名：张映文

1 作业要求

实现一个可以设置的电子钟功能，具体要求如下：

- 系统加电的RTC初始时刻设置为：2023-01-01 00:00:00，星期日。
- 支持多个定时闹钟（同时通过LED闪烁和蜂鸣器发声进行闹铃），闹铃时长为5秒，5秒后自动关闭闹铃。
- 支持手动设置：通过实验板的按键或触摸屏，实现对电子钟的参数设置（包括：设置当前年月日时分秒、设置和取消闹钟）。考虑到实验板只有4个按键（有的同学的实验板的按键可能损坏），也可以采用触摸屏实现。
- 支持连接串口设置：在连接UART串口的情况下，从PC一侧发送指令给电子钟，电子钟收到指令后进行处理，参考指令格式：

  设置当前时刻：now 2023-01-02 13:02:45 1，最后一个1表示星期1

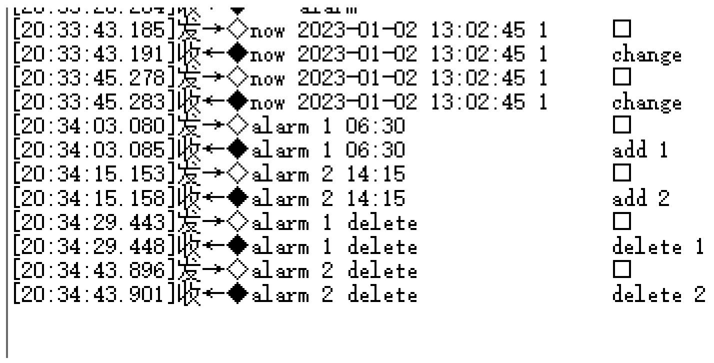
  设置闹钟1为6点30分：alarm 1 06:30

  设置闹钟2为14点15分：alarm 2 14:15

  取消闹钟1：alarm 1 delete

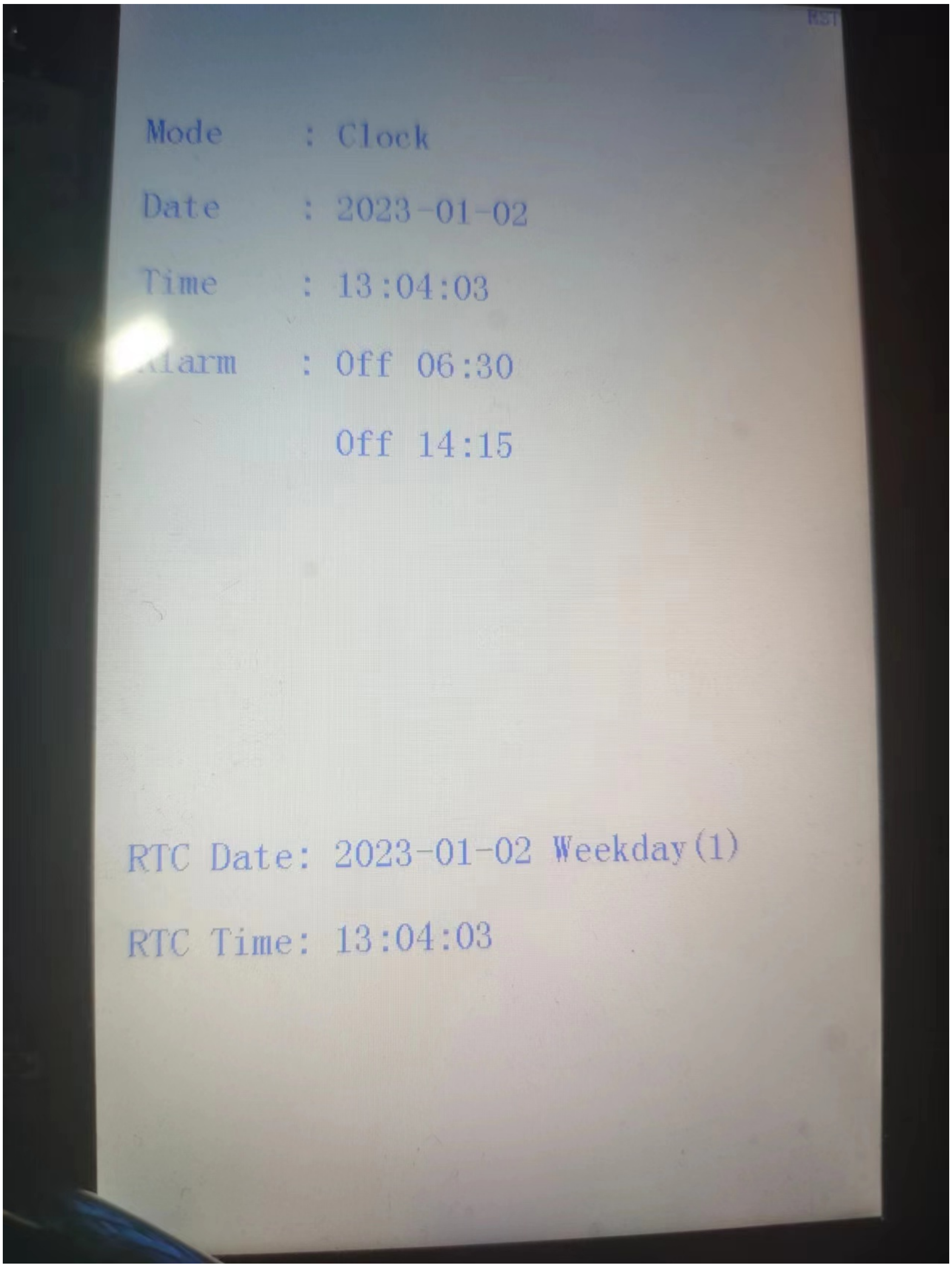
  取消闹钟2：alarm 2 delete
- 在液晶屏上显示时间和手动设置操作的界面。

2 作业内容



```
[20:33:43.185]发→◇now 2023-01-02 13:02:45 1      □
[20:33:43.191]收←◆now 2023-01-02 13:02:45 1      change
[20:33:45.278]发→◇now 2023-01-02 13:02:45 1      □
[20:33:45.283]收←◆now 2023-01-02 13:02:45 1      change
[20:34:03.080]发→◇alarm 1 06:30                  □
[20:34:03.085]收←◆alarm 1 06:30                  add 1
[20:34:15.153]发→◇alarm 2 14:15                  □
[20:34:15.158]收←◆alarm 2 14:15                  add 2
[20:34:29.443]发→◇alarm 1 delete                 □
[20:34:29.448]收←◆alarm 1 delete                 delete 1
[20:34:43.896]发→◇alarm 2 delete                 □
[20:34:43.901]收←◆alarm 2 delete                 delete 2
```

输入完后



```
Mode      : Clock

Date      : 2023-01-02

Time      : 13:04:03

Alarm     : Off  06:30

            Off  14:15




RTC Date: 2023-01-02 Weekday(1)

RTC Time: 13:04:03
```

3 代码说明

（以下仅展示添加代码部分，详细代码见附件）

```
///////省略代码//////
uint8_t Buff[30]; // 用于接收数据的缓冲区

//闹钟可设置数量，增加此处也要增加alarmFocus和初始化
#define ALARM_NUMBER                2
int alarmEnableInt[ALARM_NUMBER], alarmHourInt[ALARM_NUMBER],
alarmMinuteInt[ALARM_NUMBER];
uint8_t alarmEnableStr[ALARM_NUMBER][5];
```

```
uint8_t alarmHourStr[ALARM_NUMBER][5];
uint8_t alarmMinuteStr[ALARM_NUMBER][5];
uint16_t alarmFocus[][3] = {
    {origX + lineLabelWidth,                                    origY+150+20,
alarmEnableWidth},
    {origX + lineLabelWidth + alarmEnableWidth,                 origY+150+20,
alarmHourWidth},
    {origX + lineLabelWidth + alarmEnableWidth + alarmHourWidth,origY+150+20,
alarmMinuteWidth},
    {origX + lineLabelWidth,                                    origY+200+20,
alarmEnableWidth},
    {origX + lineLabelWidth + alarmEnableWidth,                 origY+200+20,
alarmHourWidth},
    {origX + lineLabelWidth + alarmEnableWidth + alarmHourWidth,origY+200+20,
alarmMinuteWidth}
};
uint8_t setAlarmFlag[ALARM_NUMBER] = {0};
int main(void)
{
    //初始化日期 时间
    RTC_DateTypeDef date;
    date.Year = 23;
    date.Month = 1;
    date.Date = 1;
    date.WeekDay = 7;
    HAL_RTC_SetDate(&hrtc, &date, RTC_FORMAT_BIN);

    RTC_TimeTypeDef time;
    time.Hours = 0;
    time.Minutes = 0;
    time.Seconds = 0;
    HAL_RTC_SetTime(&hrtc, &time, RTC_FORMAT_BIN);

    alarmEnableInt[0] = 1;
    alarmHourInt[0] = 0;
    alarmMinuteInt[0] = 0;

    alarmEnableInt[1] = 0;
    alarmHourInt[1] = 0;
    alarmMinuteInt[1] = 0;

    for(int i = 0; i < ALARM_NUMBER; i++) {
        sprintf((char*)alarmEnableStr[i], "%s", alarmEnableInt[i] ? "On
":"Off");
        sprintf((char*)alarmHourStr[i], "%02d", alarmHourInt[i]);
        sprintf((char*)alarmMinuteStr[i], ":%02d", alarmMinuteInt[i]);
    }
    while (1)
  {
        //主循环 判断flag 启用对应变化
        if (setTimeFlag) {
            RTC_TimeTypeDef time;
            time.Hours = timeHourInt;
            time.Minutes = timeMinuteInt;
            time.Seconds = timeSecondInt;
            HAL_RTC_SetTime(&hrtc, &time, RTC_FORMAT_BIN);
            setTimeFlag = 0;
        }
        if (setDateFlag) {
            RTC_DateTypeDef date;
            date.Year = dateYearInt;
            date.Month = dateMonthInt;
            date.Date = dateDayInt;
            date.WeekDay = dateWeekdayInt;
            HAL_RTC_SetDate(&hrtc, &date, RTC_FORMAT_BIN);
            setDateFlag = 0;
        }


        if (focusPrev) {
            LCD_Fill(focusPrev[0], focusPrev[1], focusPrev[0]+focusPrev[2],
focusPrev[1]+1, WHITE);
            focusPrev = NULL;
        }
        if (MODE_CLOCK != modeInt) {
            LCD_Fill(focusCurr[0], focusCurr[1], focusCurr[0]+focusCurr[2],
focusCurr[1]+1, BLACK);
        }

        //////代码将数据显示到led屏//////

        //HAL_UART_Receive_IT(&huart1, Buff, 1);//中断方式接收，中断回调函数里发送Buff
        HAL_UART_Receive_IT(&huart1, Buff, 30);

        delay_ms(5);

  }

}
//用户中断接收回调函数如下
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
  HAL_UART_Transmit(&huart1,Buff , sizeof(Buff), 100);
    int alarmNum;
    int hour;
    int minute;
    //判断接收串口数据的类型
    if(sscanf((char*)Buff, "now 20%d-%d-%d %d:%d:%d %d", &dateYearInt,
&dateMonthInt, &dateDayInt, &timeHourInt, &timeMinuteInt, &timeSecondInt,
&dateWeekdayInt)){
        printf("change");
        setDateFlag=1;
        setTimeFlag=1;
        sprintf((char*)dateDayStr, "-%02d", dateDayInt);
        sprintf((char*)dateMonthStr, "-%02d", dateMonthInt);
        sprintf((char*)dateYearStr, "20%02d", dateYearInt);

        sprintf((char*)timeHourStr, "%02d", timeHourInt);
        sprintf((char*)timeMinuteStr, ":%02d", timeMinuteInt);
        sprintf((char*)timeSecondStr, ":%02d", timeSecondInt);
    }

    else if(sscanf((char*)Buff, "alarm %d %d:%d",&alarmNum, &hour, &minute)==3){
        printf("add %d",alarmNum);
        alarmEnableInt[alarmNum-1] = 1;
        alarmHourInt[alarmNum-1] = hour;
        alarmMinuteInt[alarmNum-1] = minute;
        sprintf((char*)alarmEnableStr[alarmNum-1], "%s",
alarmEnableInt[alarmNum-1] ? "On ":"Off");
        sprintf((char*)alarmHourStr[alarmNum-1], "%02d", alarmHourInt[alarmNum-
1]);
        sprintf((char*)alarmMinuteStr[alarmNum-1], ":%02d",
alarmMinuteInt[alarmNum-1]);

    }
    else if(sscanf((char*)Buff, "alarm %d delete",&alarmNum)){
        printf("delete %d",alarmNum);
        alarmEnableInt[alarmNum-1] = 0;
        sprintf((char*)alarmEnableStr[alarmNum-1], "%s",
alarmEnableInt[alarmNum-1] ? "On ":"Off");
    }

}
//按键操作
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if (GPIO_Pin == KEY1_Pin) {
        if (HAL_GPIO_ReadPin(KEY1_GPIO_Port, KEY1_Pin) == GPIO_PIN_SET) {
            printf("Key1\r\n");
            //设置flag
            if (MODE_SET_TIME == modeInt) {
                setTimeFlag = 1;
            }
            if (MODE_SET_DATE == modeInt) {
                setDateFlag = 1;
            }

            modeInt++;
            if (MODE_MAX < modeInt) {
                modeInt = 0;
            }
            focusInt = 0;
            focusPrev = focusCurr;
            if (MODE_SET_DATE == modeInt) {
                focusCurr = dateFocus[0];
                focusMax = 3;
            }
            else if (MODE_SET_TIME == modeInt) {
                focusCurr = timeFocus[0];
                focusMax = 3;
            }
            else if (MODE_SET_ALARM == modeInt) {
                focusCurr = alarmFocus[0];
                focusMax = 3 * ALARM_NUMBER;
            }
        }
    }
    else if (GPIO_Pin == KEY2_Pin)
    {
        if (HAL_GPIO_ReadPin(KEY2_GPIO_Port, KEY2_Pin) == GPIO_PIN_SET) {
            printf("Key2\r\n");
            if (MODE_CLOCK != modeInt) {
                focusPrev = focusCurr;
                focusInt++;
                if (focusMax == focusInt) {
                    focusInt = 0;
                }
                if (MODE_SET_DATE == modeInt) {
                    focusCurr = dateFocus[focusInt];
                }
                else if (MODE_SET_TIME == modeInt) {
                    focusCurr = timeFocus[focusInt];
                }
                else if (MODE_SET_ALARM == modeInt) {
                    focusCurr = alarmFocus[focusInt];
                }
            }
        }
    }
    else if (GPIO_Pin == KEY3_Pin)
    {
        if (HAL_GPIO_ReadPin(KEY3_GPIO_Port, KEY3_Pin) == GPIO_PIN_SET) {
            printf("Key3\r\n");
            //////修改日期时间的代码//////
            //修改闹钟
            if (MODE_SET_ALARM == modeInt) {
                if (MODE_SET_ALARM == modeInt) {
                for(int i = 0; i < ALARM_NUMBER; i++) {
                    if ((0 + 3*i) == focusInt){
                        alarmEnableInt[i] ^= 1;
                    }
                    if ((1 + 3*i) == focusInt && 0 < alarmHourInt[i]){
                        alarmHourInt[i]--;
                    }
                    if ((2 + 3*i) == focusInt && 0 < alarmMinuteInt[i]){
                        alarmMinuteInt[i]--;
                    }
                    sprintf((char*)alarmEnableStr[i], "%s", alarmEnableInt[i] ?
"On ":"Off");
                    sprintf((char*)alarmHourStr[i], "%02d", alarmHourInt[i]);
                    sprintf((char*)alarmMinuteStr[i], ":%02d",
alarmMinuteInt[i]);
                }

            }
        }
    }
    else if (GPIO_Pin == KEY4_Pin)
    {
        //类似key3一样修改
    }
}
//比对时间 开始闹钟 添加闪灯情况led1
void HAL_SYSTICK_Callback(void)
{
    ////////////
        for(int i = 0; i < ALARM_NUMBER; i++) {
            if (time.Hours == alarmHourInt[i] && time.Minutes ==
alarmMinuteInt[i] && 00 == time.Seconds) {
                alarmFlag = 5;
            }
            ////////////
```

```
                }
            }
        }
    }

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    static int tim2Cnt = 0;
    static int secondCnt = 0;
    static int led_flash = 0;
    if (htim->Instance == TIM2) {
        if (alarmFlag) {
            led_flash++;
            if(led_flash==500){
                led_flash=0;
                //printf("----alarm----");
                HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
            }
            HAL_GPIO_TogglePin(BEEP_GPIO_Port, BEEP_Pin);
        }
        //添加闪灯
        tim2Cnt++;
        if (tim2Cnt == 1000) {
            tim2Cnt = 0;
            secondCnt++;
            if (alarmFlag) {
                alarmFlag--;
                if(!alarmFlag){
                    HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_SET);
                }
            }
            //printf("TIM second=%d\r\n", secondCnt);
        }
    }
}
```

4 存在问题

- 串口发送过来的数据只有到30字节（设置的uint8_t Buff[30]）才接收一次，低于30字节不接收而是存着，待凑齐30字节后接收；超出30字节的部分补充到下一30字节，所有要发送一个命令时，我使用空格补全到30字节后再发送。

　可能的解决办法：一字节一字节接收（我试过但是效果不好，数据顺序大概率要乱）；或是换一个函数或不用回调函数。

- 闹钟数量只能修改ALARM_NUMBER 宏

  可能的解决办法：不用宏，用全局变量或是其他，串口输入添加闹钟指令再加闹钟并显示。