# 嵌入式系统与接口技术 EST2506 期末大作业

姓名：张文康　　学号：522021910121

## 一、功能简述

- **开机画面**
- **显示模式**
  - 日期单独显示
  - 时间单独显示
  - 左流水显示（慢）
  - 右流水显示（慢）
  - 左流水显示（快）
  - 右流水显示（快）
- **设置模式**
  - 日期设置
  - 时间设置
  - 闹钟设置
- **音乐播放**
  - 作为闹钟
  - 单独欣赏
- **串口通信**
  - 帮助指令
  - 初始化指令
  - 日期、时间、闹钟的获取
  - 日期、时间、闹钟的设置
  - 音乐的播放与停止
  - 班级、学号获取
- **BootLoader**

## 二、按键说明

> R 表示1294上的按键

- **SW1**：日期显示模式，同时亮起LED1
- **SW2**：时间显示模式，同时亮起LED2
- **SW3**：设置模式，通过SW5/SW6左右调整所设置的位，下方的LED指示所设置的位
- **SW4**：显示模式下音乐播放；时间设置模式，用于设置闹钟

- **SW5**：显示模式下，短按为慢右流水、长按为快右流水；设置模式下，右移设置位
- **SW6**：显示模式下，短按为慢左流水、长按为快左流水；设置模式下，左移设置位
- **SW7**：设置模式下，确认设置内容
- **SW8**：一键返回，停止音乐播放及设置、恢复显示模式
- **R-SW1**：设置模式下增加数值
- **R-SW2**：设置模式下减小数值
- **R-RESET**：恢复出厂设置
- 上述**按键具有较强的复用性**，如不同模式下、相同模式下不同按下时间都会导致不同的功能

## 三、具体实现

1. 设备初始化、开机画面
2. 模式选择
    - 设置模式选择函数，将解析到的键值转换为对应的模式执行，具有可扩展性

```
void  ModeSelect(void)
{
        switch (key_mode)
        {
                case Key_Return:        // 一键返回
                        key_mode = Key_Null;
                        key_right = key_left = key_enter = key_plus = key_minus= 0;
                        SetMusicOff();
                        break;
                case Key_Date:          // 显示日期
                        DisplayStatus = 1;
                        LedLight(0x1 << 0);
                        break;
                case Key_Time:          // 显示时间
                        DisplayStatus = 2;
                        LedLight(0x1 << 1);
                        break;
                case Key_Set:           // 设置日期或时间
                        SetModeOn();
                        break;
        }
}
```

3. 日期、时间、闹钟相关设计
    - 设置全局变量 `year`， `month`， `day`， `hh`， `mm`， `ss`， `AlarmH`， `AlarmM`， `AlarmS` 记录相应数值，并设置对应的字符串便于显示

- 设置函数 `DateGen()`，`ClockGen()`，`AlarmGen` 用于日期、时间、闹钟的生成，并且在显示模式、设置模式、串口通信中均可使用，**复用性强**
- **具有一定的容错性**，若设置日期为“2024-2-30”的错误日期，系统会直接将其转换为“2024-3-1”

4. 串口通信
   - 在函数 `UARTCmdProcess()` 针对多组情况进行处理
   - 具有容错机制：大小写不区分，多个空格会只保留1个
   - 具有相应的帮助指令“HELP"

## 四、亮点展示：完整音乐的播放

- 将简谱中的每一个数字转换为对应的频率及持续的时间（对应拍数）
- 在TIMER1的中断函数中，计时相应时间，利用PWM发生器产生对应频率的方波

```
void TIMER1A_Handler(void)
 {
        TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT); // 清除中断标志
        // 处理一个新音符
        PWMGenDisable(PWM0_BASE, PWM_GEN_3); // PWM 信号暂停

        if (CurPos == SizeOfSong) { CurPos = 0; }
        else {
                // 设置音符的播放时间为 TIMER1 的定时时长
                TimerLoadSet(TIMER1_BASE, TIMER_A, ui32SysClock/1000.0 * time[CurPos] * Bea
                if (freqs[CurPos] != 0) {
                        // 根据音频计算 PWM 信号的周期，并启动产生 PWM 信号
                        PWMGenPeriodSet(PWM0_BASE, PWM_GEN_3, ui32SysClock / freqs[CurPos])
                        PWMPulseWidthSet(PWM0_BASE, PWM_OUT_7,PWMGenPeriodGet(PWM0_BASE, PWI
                        if (MusicOn) PWMGenEnable(PWM0_BASE, PWM_GEN_3); // 使能 PWM 信号产生
                }
                CurPos++; // 向后推进
        }
 }
 ```
```

- 方波信号驱动无源蜂鸣器实现音乐播放
```