哈爾濱工業大學(深圳)
HARBIN INSTITUTE OF TECHNOLOGY,SHENZHEN

实验与创新实践教育中心

实验报告

课程名称：电子工艺实习-嵌入式系统软件设计

实验名称：单片机MSP430F5529-麦克风和拨码电位器控制音量柱实验

姓　　名：宋子洋

学　　号：190210305

专业-班级：19级电信3班

实验日期：2021 年 7 月 15 日

教师评语：

评分：

教师签字：

日　　期：

源代码要求：关键代码要求注释
电子版文件名格式：专业--姓名--学号—实验 X
实验报告提交：纸质版实验报告由班长统一收取；电子版报告按班级打包，发到主讲老师 QQ 邮箱。

1. 实验内容

参考课上实验，通过 AD 同时采集麦克风声音信号和电位器输出电压。

（1）音量强度由 LED 音量柱显示：根据声音大小，分为几档，通过 LED 灯 L1~L6 显示，LED 点亮数量与声音大小成正比。

（2）若电位器输出电压高于某一电压值，则点亮主板上的 LED1 灯。

（3）增加按键开关，控制 ADC 采集的开始和停止。

2. 实验原理（用文字或图表阐述要实现的实验现象、相关单片机模块的理论知识、寄存器使用方法等）

单片机理论知识：

1. 对应的端口需要设置的有：

输入寄存器	Input Registers (PxIN)
输出寄存器	Output Registers (PxOUT)
方向寄存器	Direction Registers (PxDIR)
上下拉电阻使能寄存器	Pullup or Pulldown Resistor Enable Registers (PxREN)
功能选择寄存器	Function Select Registers (PxSEL)
此外，对于按键 S1，还有	
中断使能寄存器	P1IES Register
中断标记寄存器	P1IFG Register

2. 寄存器使用方法：

设置寄存器对应引脚位置：	Reg |= BITx
清除寄存器对应引脚位置设置：	Reg &= ~BITx
寄存器对应位置取反：	Reg ^= BITx
判断对应位置为 0：	(!(Reg & BITx))
判断对应位置为 1：	(Reg & BITx)

3. ADC 模数转换器使用方法：

包括的模块有：

转换控制寄存器 0（ADC12CTL0 Register）
转换控制寄存器 1（ADC12CTL1 Register）
转换控制寄存器 2（ADC12CTL2 Register）

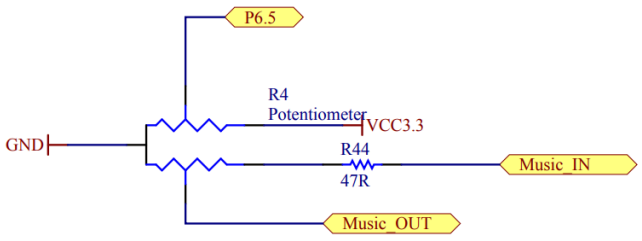
存储寄存器（ADC12MEMx Register）
存储控制寄存器（ADC12MCTLx Register）
中断使能寄存器（ADC12IE Register）
中断标志寄存器（ADC12IFG Register）
中断向量寄存器（ADC12IV Register）

初始化时，重点处理以下功能的通断：
（1）在 ADC12ENC==0 时（默认）,初始化各寄存器，后打开 ADC 转换使能（ADC12ENC==1）。
（2） 选择多路采样转换（首次需要 SHI 信号上升沿触发采样定时器）自动循环采样转换。
（3） 启动（打开）ADC12 模块。
（4） 选择作为起始的寄存器地址。
（5） 选择采样转换模式。
（6） 选择采样保持模式，代码中选择采样信号（SAMPCON）的来源是采样定时器。
（7） 配置存储寄存器的选择通道，代码中寄存器 0 选择通道 P6.2 麦克风，寄存器 1 选择通道 P6.5 连接拨码电位器（GPIO 默认方向是输入方向）。
（8） ADC 转换使能。
（9） 采样开始前，要设置开始采样转换。

4. 中断的使用方法：
首先需要使能全局中断，`_enable_interrupt();`
以 P1 的中断为例，需要有中断向量和中断的头，进入中断后退出时要清除中断标记。

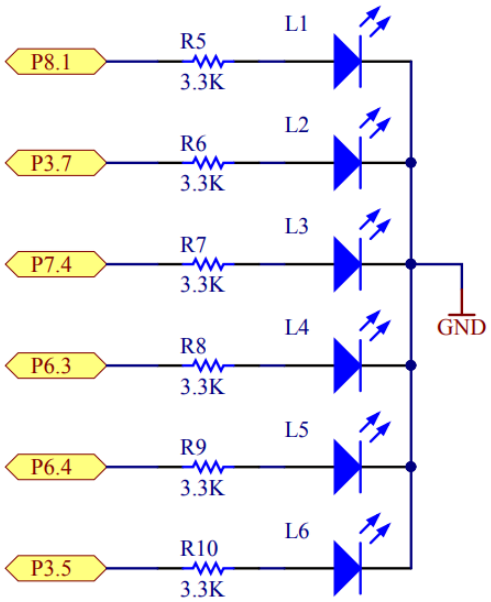
```
#pragma vector=PORT1_VECTOR //中断向量
__interrupt void Port_1(void) //中断函数
{
    ..............
    P1IFG &= ~BIT2; //清除中断标记
}
```

3. 硬件原理图（原理图、接线图等，要阐明具体引脚并与程序对应）

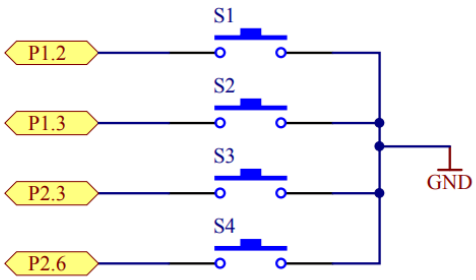


拨码电位器

P6.5 对应拨码电位器，功能设为外设。



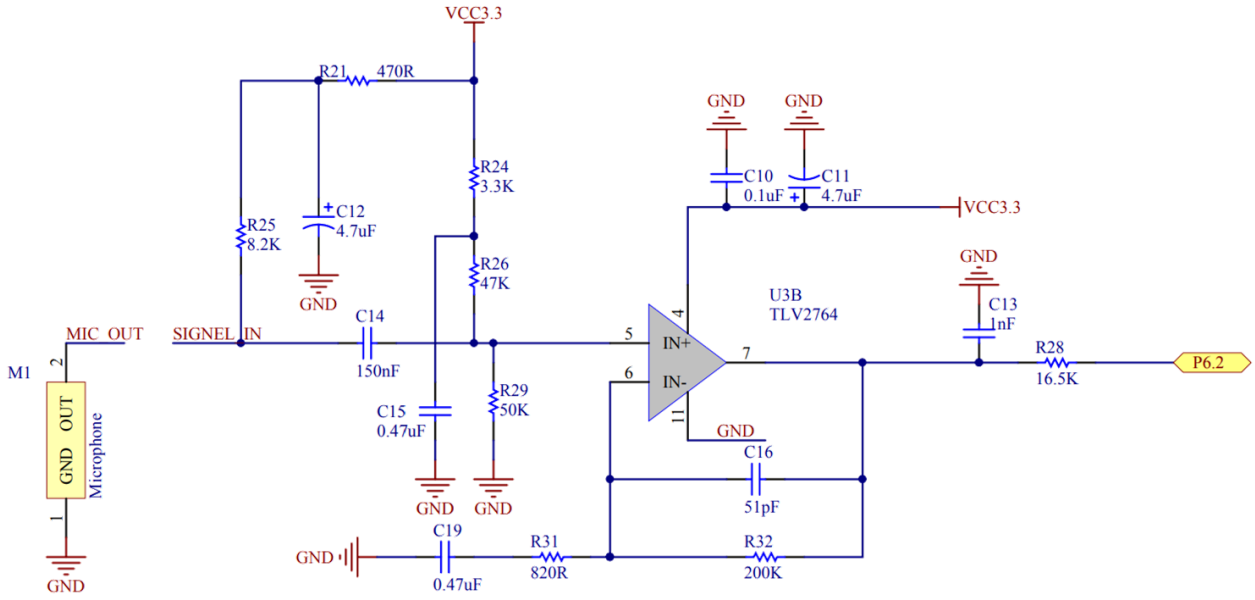
LED



USER Key

LED 灯端口输出，依次是 L1 对应 P8.1；L2 对应 P3.7；L3 对应 P7.4；L4 对应 P6.3；L5 对应 P6.4；L6 对应 P3.5，功能设为 GPIO。

用户按键 S1 对应 P1.2 端口，功能设为 GPIO，使能上下拉电阻功能且置为上拉电阻方式，使能中断。



音频输入调理

P6.2 端口对应麦克风输入，功能设为外设。

接线为口袋板 MIC 和 OPA_IN 端口用跳线帽连接。

4. 程序流程图（画出详细设计的程序流程图，具体到每个步骤）

流程图的绘制详见实验报告末尾附页。

5. 源代码

源代码（main.c）详见实验报告末尾附页。

6. 实验结果（文字阐述实验现象，实验波形可手画）

（1）开关 S1 按下，声音和拨码电位器开始采样。

（2）麦克风输入声音，且 LED 音量柱显示音量强度：根据声音大小分为七档，通过 LED 灯 L1~L6 显示，LED 点亮数量与声音大小成正比，从全灭到全亮连续变化。

（3）播放音乐时，音量高处亮灯多，音量低处亮灯少，音拍急促时灯变化快，音拍长缓时灯变化慢，呈现音量柱随音乐律动变化的特点。由于声音采集后加入了滤波环节，采用中值滤波法，即去除最值后取平均数。虽增加了一定的时延，但是音量柱变化更为平滑，LED 频繁闪烁的现象更少，且减少了噪声对音量柱的敏感程度。

（4）调整拨码电位器，若电位器输出电压高于某一电压值，则主板上的 LED1 灯为常亮，但是音量柱其他 LED 灯仍然继续正常工作。

（5）按下按键开关 S1，ADC 模数转换器采样采集随即停止，所有 LED 立即熄灭，拨码电位器不能点亮 LED1。

（6）再次按下开关 S1，声音和拨码电位器重新开始采样。

具体实验现象以及音量柱表现情况请参考附件中的测试视频。

7. 实验中遇到过哪些问题？如何解决的？

实验过程中，常常由于环境底噪的影响，LED1 处于频繁亮灭的状态；且由于音量采样值常处于一定的范围内，对应 LED 亮灭的需要设置合理的梯度和上下限；同时程序内需要对 LED 的亮灭循环设定一定的时延以减弱频繁闪烁、亮灭不稳定的现象，而上述这些参数需要大量测试由经验总结得到。

通过大量实验总结，对于这些值设定了一个合理的大小。例如，亮灭的梯度值设为一个数组，即 `g[7]={0,2005,2100,2130,2180,2220,2250}`；同时亮灭周期中也设置了

`_delay_cycles(20000)`的时延使得现象明显而又稳定。

同时，由于开关 S1 的中断亦有一定的敏感性，通过仿照老师课上所给的 Timer_A 示例代码中设置时延和二重判断的办法，使得开关 S1 更稳定地工作。

另外，由于两路采集同时进行，需要设置两个寄存器，通过查阅数据手册，再仿照之前的实验，通过对 ADC 的初始化解决了此问题。

8. 实验体会与建议

由于之前的课上实验中已经做好一部分的内容，接下来关键为整合知识，编写出一个功能完整的程序。本实验难度不大，可以作为上手实践，为接下来的大作业做充足的准备。

附页：源代码/*main.c*/

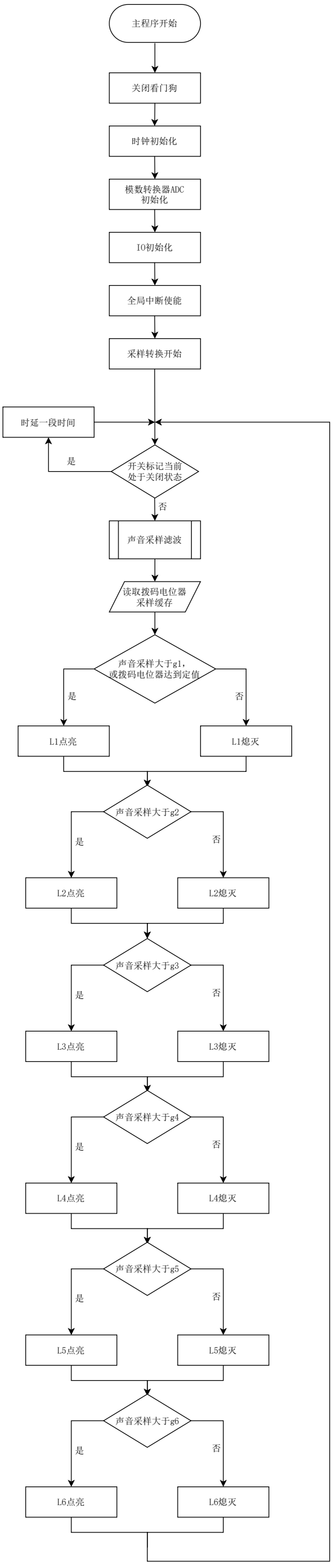
```
#include <msp430.h>
#define N 12

int Filter()//中值滤波法
{
    int i, j, t;
    int value_buf[N];
    int sum = 0;
    for (i = 0; i < N; i++)
    {
        value_buf[i] = ADC12MEM0;
        _delay_cycles(1000);
    }
    for (j=0; j<N-1; j++)
    {
        for (i=0; i<N-j-1; i++)
        {
            if (value_buf[i] > value_buf[i+1])
            {
                int temp = value_buf[i];
                value_buf[i] = value_buf[i+1];
                value_buf[i+1] = temp;
            }
        }
    }
    for(i=1; i<N-1; i++) sum += value_buf[i];
    t = sum/(N-2);
    return t;
}
void light_off()//LED灯全灭
{
    P8OUT &= ~BIT1;
    P3OUT &= ~BIT7;
    P7OUT &= ~BIT4;
    P6OUT &= ~BIT3;
    P6OUT &= ~BIT4;
    P3OUT &= ~BIT5;
}
void ioinit()
{//LED1~LED6的初始化
    P8DIR |= BIT1;  //p8.1 output
    P3DIR |= BIT7;  //p3.7 output
    P7DIR |= BIT4;  //p7.4 output
```

```
    P6DIR |= BIT3;  //p6.3 output
    P6DIR |= BIT4;  //p6.4 output
    P3DIR |= BIT5;  //p3.5 output
    light_off();
 //拨盘电位器对应IO初始化
    P6DIR |= BIT5;
    P6SEL |= BIT5;
//麦克风对应IO初始化
    P6DIR |= BIT2;
    P6SEL |= BIT2;

    P1DIR &= ~BIT2;
    P1REN |= BIT2;              // 使能P1.2上下拉电阻功能
    P1OUT |= BIT2;              // 置P1.2为上拉电阻方式
    P1IE  |= BIT2;              // 使能P1.2中断
    P1IES |= BIT2;              // 高低电平转换
}

void AD_Init()
{// 在ADC12ENC==0时（默认）,初始化各寄存器，后打开ADC转换使能（ADC12ENC==1）
    //多路采样转换（首次需要SHI信号上升沿触发采样定时器）自动循环采样转换
    ADC12CTL0 |= ADC12MSC;

    //启动（打开）ADC12模块
    ADC12CTL0 |= ADC12ON;

    //设置采样起始地址
    ADC12CTL1 |= ADC12CSTARTADD_0;

    //选择序列通道多次循环采样转换
    ADC12CTL1 |= ADC12CONSEQ_3;

    //采样保持模式选择，选择采样信号（SAMPCON）的来源是采样定时器
    ADC12CTL1 |= ADC12SHP;

    //存储寄存器0选择通道P6.2麦克风,P6.5连接拨码电位器（GPIO默认方向是输入方向）
    ADC12MCTL0 |= ADC12INCH_2;
    ADC12MCTL1 |= ADC12INCH_5;

    //ADC转换使能
    ADC12CTL0 |= ADC12ENC;
    ADC12CTL1 |= ADC12ENC;
}
```

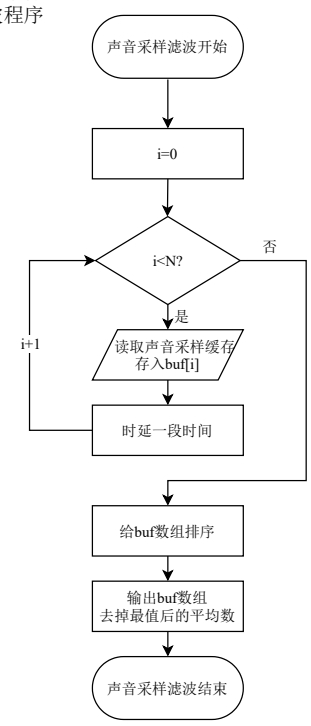
```
void initClock()
{//最终效果：MCLK:16MHZ,SMCLK:8MHZ,ACLK:32KHZ
     UCSCTL6 &= ~XT1OFF;          //启动XT1
     P5SEL |= BIT2 + BIT3;        //XT2引脚功能选择
     UCSCTL6 &= ~XT2OFF;          //打开XT2

     //设置系统时钟生成器1,FLL control loop关闭SCG0=1,关闭锁频环，用户自定义UCSCTL0~1工作模式
     __bis_SR_register(SCG0);

     //手动选择DCO频率阶梯（8种阶梯），确定DCO频率大致范围。
     UCSCTL0 = DCO0+DCO1+DCO2+DCO3+DCO4;
     //DCO频率范围在28.2MHz以下，DCO频率范围选择（三个bit位，改变直流发生器电压，进而改变DCO输出频率）
     UCSCTL1 = DCORSEL_4;
     //fDCOCLK/32，锁频环分频器
     UCSCTL2 = FLLD_5;

     //n=8,FLLREFCLK时钟源为XT2CLK
     //DCOCLK=D*(N+1)*(FLLREFCLK/n)
     //DCOCLKDIV=(N+1)*(FLLREFCLK/n)
     UCSCTL3 = SELREF_5 + FLLREFDIV_3;
     //ACLK的时钟源为DCOCLKDIV,MCLK\SMCLK的时钟源为DCOCLK
     UCSCTL4 = SELA_4 + SELS_3 +SELM_3;
     //ACLK由DCOCLKDIV的32分频得到，SMCLK由DCOCLK的2分频得到
     UCSCTL5 = DIVA_5 +DIVS_1;
}
static volatile unsigned int ADon = 0;
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

    initClock();
    ioinit();
    AD_Init();

    _enable_interrupt();

    ADC12CTL0 |= ADC12SC;                          //开始采样转换

    volatile unsigned int value[2] = {0,0}, g[7]={0,2005,2100,2130,2180,2220,2250};
    //g[1]~g[6]为声音采样值阶梯，存为数组
    for(;;)
    {
```

```
        if (ADon == 0) {                            //若开关标记为关，则不进入采样模块
            _delay_cycles(50000);
            continue;
        }
        value[0] = Filter(), value[1] = ADC12MEM1;  //value[0]记录声音滤波采
样,value[1]记录拨码电位器采样
        _delay_cycles(20000);                       //时延控制
        //完成LED1~LED6灯的亮灭判断，设置value的有效范围
        if ((value[0] >= g[1] || value[1] > 1500) && ADon)  {P8OUT |= BIT1;} else
{P8OUT &= ~BIT1;}
        if (value[0] >= g[2] && ADon) {P3OUT |= BIT7;} else {P3OUT &= ~BIT7;}
        if (value[0] >= g[3] && ADon) {P7OUT |= BIT4;} else {P7OUT &= ~BIT4;}
        if (value[0] >= g[4] && ADon) {P6OUT |= BIT3;} else {P6OUT &= ~BIT3;}
        if (value[0] >= g[5] && ADon) {P6OUT |= BIT4;} else {P6OUT &= ~BIT4;}
        if (value[0] >= g[6] && ADon) {P3OUT |= BIT5;} else {P3OUT &= ~BIT5;}

    }
}

#pragma vector=PORT1_VECTOR //中断向量
__interrupt void Port_1(void) //中断函数
{
    if (P1IFG & BIT2) {
        _delay_cycles(50);
        if (P1IFG & BIT2) {
            ADon ^= 1;
            light_off();
        }
    }
    P1IFG &= ~BIT2; //清除中断标记
}
```

主程序



声音采样
滤波程序



按键中断程序