哈爾濱工業大學 (深圳)

嵌入式系统软件设计基础
实验报告

实验所属课程：电子工艺实习

姓名：李木晗

学号：SZ170210119

专业：电子信息

评分：

批阅老师：

实验内容：　　定时器 A

实验代码要求：关键代码要注释，整个工程打包发到指定邮箱：jingjing.yang@163.com
邮件名称要求：专业_姓名_学号_软件基础实验报告

1.1 实验目的

（1） 了解嵌入式系统软件设计与开发流程，熟悉CCS 的基本使用方法；
（2） 掌握MSP430 系列单片机程序开发的基本步骤；
（3） 掌握对 IO 口的查询操作和 IO 基本操作的流程；
（4） 了解 MSP430F5529LP 时钟系统；
（5） 学习看门狗定时器原理，熟练掌握看门狗定时中断；
（6） 学习 Timer_A 定时器原理，掌握 Timer_A 的定时中断；
（7） 掌握蜂鸣器的使用方法，熟练应用 GPIO 控制 LED 亮灭。

1.2 实验原理及用户手册查找

（文字、图表等阐述要实现的实验效果，阐述相关模块的使用方法和有关理论知识和寄存器使用方法等）

名称	缩写	BIT=1	BIT=0
方向寄存器	PxDIR	输出模式	输入模式
输入寄存器	PxIN	输入高电平	输入低电平
输出寄存器	PxOUT	输出高电平	输出低电平
上下拉电阻使能寄存器	PxREN	使能	禁用
功能选择寄存器	PxSEL	外设功能	IO端口
驱动强度寄存器	PxDS	高强度	低强度
中断使能寄存器	PxIE	允许中断	禁止中断
中断触发沿寄存器	PxIES	下降沿置位	上升沿置位
中断标志寄存器	PxIFG	有中断请求	无中断请求

1.2.1 LED 灯的开关

以 L1（P8.1）为例，首先用 `P8DIR |= 0x02;`初始化为输出， `P8OUT &= ~0x02;`使其熄灭，`P8OUT |= 0x02;`使其点亮，`P8OUT ^= 0x02;`使其状态翻转。

1.2.2 按键使能

以 S1（P1.2）为例，首先用 `P1DIR |= 0x04;`初始化为输出， `P1REN |= 0x04;`初始化上下拉电阻，`P1OUT |= 0x04;`设置为高电平，通过`(P1IN & 0x04)`即可读取按键状态，高电平松开，低电平按下。

1.2.3 按键中断

以 S1（P1.2）为例，首先用 `P1IE |= 0x04;`进行中断使能， `P1IES |= 0x04;`选择中断边沿为下边沿，`P1IFG &= ~0x04;`归零中断标志，通过`#pragma vector=PORT1_VECTOR __interrupt void Port_1_Key(void) {}`函数即可进入中断，在中断中需要手动执行 `P1IFG &= ~0x04;`归零中断标志。

1.2.4 蜂鸣器的开关

蜂鸣器以交流电发声，其操作与 LED 相似。

1.2.5 Time_A 定时器

1.2.5.1 TA0CTL

[1] TACTL　　TIMER_A 控制寄存器

15	14	13	12	11	10	9	8
Unused						TASSELx	

7	6	5	4	3	2	1	0
IDx		MCx		Unused	TACLR	TAIE	TAIFG

查阅手册了解其各位的含义，通过 TASSELx 和 MCx 位设置计数时钟和模式。例如，使用 `TA0CTL = 0x0110;`选择 ACLK 时钟及增计数模式，使用 `TA0CTL = 0x0210;`

选择 MCLK 时钟及增计数模式。

1.2.5.2 TA0CCTLx

[3] TACCTLx TIMER_A 捕获/比较控制寄存器 x

15	14	13	12	11	10	9	8
CMx		CCISx		SCS	SCCI	Unused	CAP
7	6	5	4	3	2	1	0
OUTMODx			CCIE	CCI	OUT	COV	CCIFG

查阅手册了解其各位的含义，通过 CCIE 设置是否允许中断，0 禁止中断，1 允许中断。

1.2.5.3 TA0CCRx

[4] TACCRx TIMER_A 捕获/比较寄存器 0

15	14	13	12	11	10	9	8
TACCRx							
7	6	5	4	3	2	1	0
TACCRx							

这一个寄存器指定计数大小，通常作为周期寄存器。

1.2.5.4 TA0IV

[5] TAIV TIMER_A 中断向量寄存器

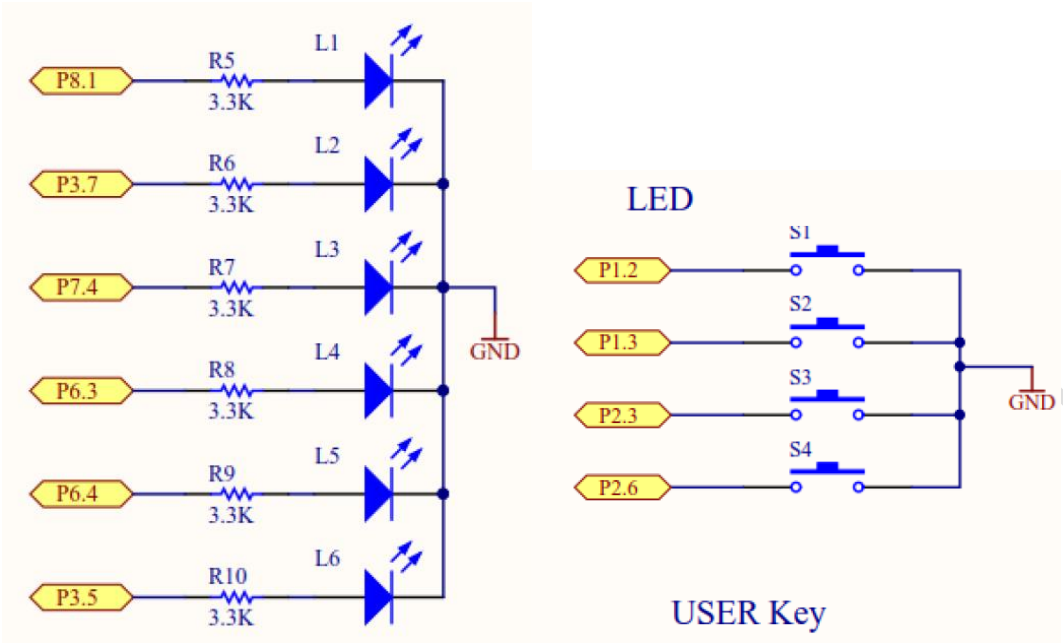
15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
0	0	0	0	TAIVx			0

对于 CCR0 的中断，Timer_A 分配了一个专门的寄存器，而对于其他中断，公用一个寄存器，并以此表明中断来源：

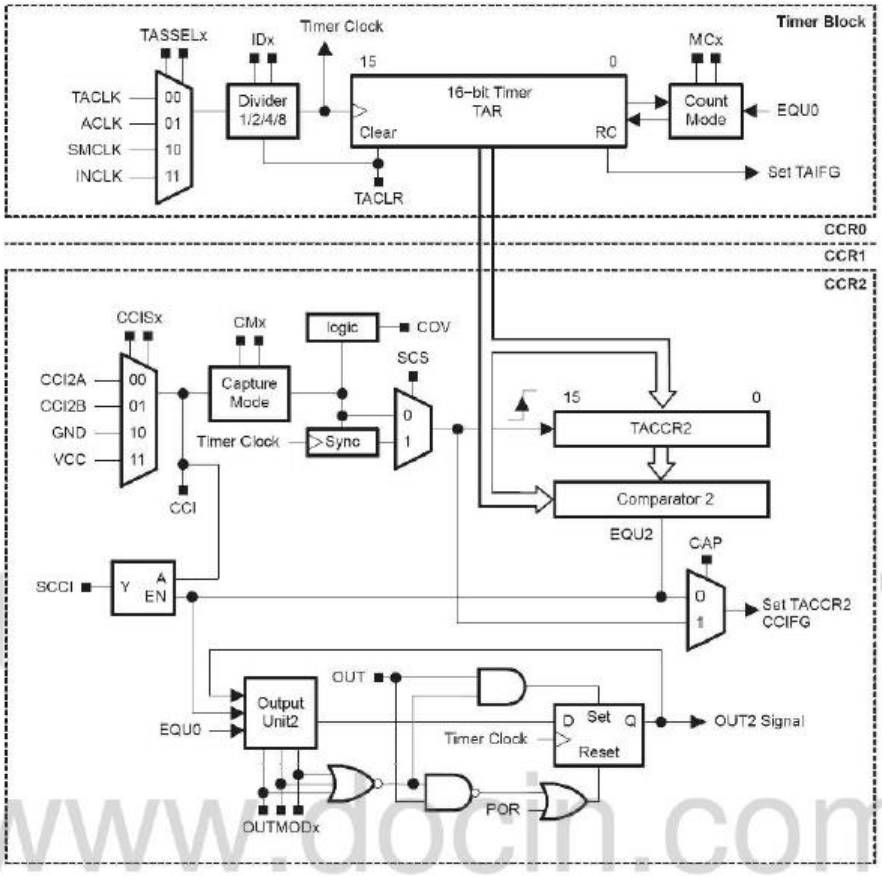
TAIV Contents	Interrupt Source	Interrupt Flag	Interrupt Priority
00h	No interrupt pending	–	
02h	Capture/compare 1	TACCR1 CCIFG	Highest
04h	Capture/compare 2	TACCR2 CCIFG	
06h	Reserved	–	
08h	Reserved	–	
0Ah	Timer overflow	TAIFG	
0Ch	Reserved	–	
0Eh	Reserved	–	Lowest

1.3 单片机硬件

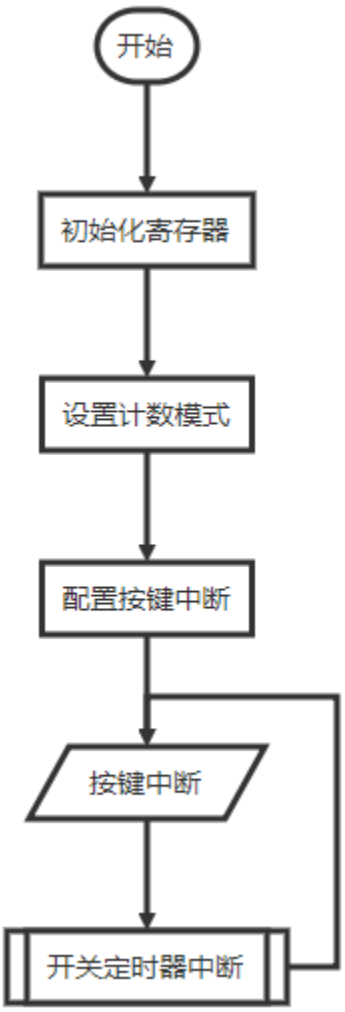
（原理图、接线图等，要阐明具体引脚并与程序对应）



TIMER_A 的结构原理图。



1.4 程序设计思路



（详细叙述程序设计思路和流程图）

利用按键中断实现定时器中断的启停，再通过定时器中断控制 LED 发光与蜂鸣器发声。

其中，S1 控制 CCR1 中断状态翻转，S2 控制 CCR2 中断状态翻转，S3 控制 CCR0 中断状态翻转，S4 禁止所有定时器中断。

1.5 实验结果

（实验结果文字阐述，按键、灯效果展示，示波器图形展示等，手画亦可）

示例参数 1：TA0CTL = 0x0110 选择 ACLK 时钟（f = 32768 Hz）及增计数模式，TA0CCR0 = 32768，TA0CCR1 = 21845，TA0CCR2 = 10923。

示例参数 2：TA0CTL = 0x0210 选择 MCLK 时钟（f = 1048576 Hz）及增计数模式，TA0CCR0 = 32768，TA0CCR1 = 21845，TA0CCR2 = 10923。

按下 S1，允许 CCR1 中断，当计数器数到 21845 时进入 TIMER0_A1_VECTOR 中断，对 L1（CCR1 指示灯），L6（总指示灯），蜂鸣器电位翻转，单独作用表现为每秒（示例参数 2：1/32 秒）翻转一次。再按下 S1，禁止 CCR1 中断，此现象停止。

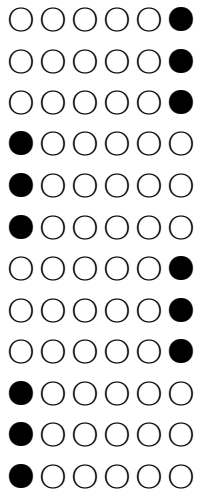
按下 S2，允许 CCR2 中断，当计数器数到 10923 时进入 TIMER0_A1_VECTOR 中断，对 L2（CCR2 指示灯），L6（总指示灯），蜂鸣器电位翻转，单独作用表现为每秒（示例参数 2：1/32 秒）翻转一次。再按下 S2，禁止 CCR2 中断，此现象停止。

按下 S3，允许 CCR0 中断，当计数器数到 32768 时进入 TIMER0_A0_VECTOR 中断，对 L4（CCR0 指示灯），L6（总指示灯），蜂鸣器电位翻转，单独作用表现为每秒（示例参数 2：1/32 秒）翻转一次。再按下 S3，禁止 CCR0 中断，此现象停止。

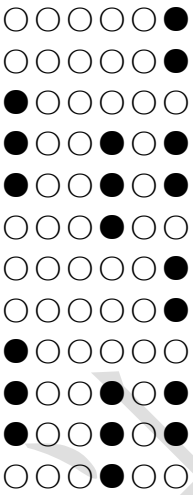
按下 S4，禁止所有中断，现象停止。

当以上效果叠加时，各自中断指示灯独立闪烁，L6 及蜂鸣器频率为其叠加。

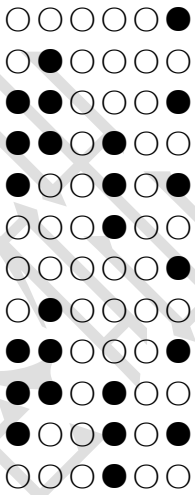
仅 CCR1：



CCR0+CCR1：



CCR0+CCR1+CCR2：



1.6 实验中遇到的问题和解决方法？

1.6.1 对各寄存器使用方式和作用不清楚

解决方法主要是参考了《MSP430 中文手册》中的内容，对每个寄存器的作用和每个位的参数设置有了了解，再编写程序实际验证。遇到还是不能理解的，比如 TAIV 中断向量寄存器，我上网搜索了相关论坛中的问题，并且参考了一些例程。

1.6.2 对程序执行性能的改进

在 TAIV 中断中，首先我采用的是 if 语句判断中断位置，但是经过测试，只能运行第一个 if 语句的内容，Google 相关问题后发现可能原因是执行效率不高，改进为 switch 语句后程序正常运行。此外还了解到更多的一些改进性能的方法，比如提前判断类型等等。

1.7 实验体会与建议

在实验过程中，我首先学会了定时器和中断的使用方法，并且通过各类尝试和运用有了进一步的了解和体会。此外，参考一些资料和手册能让我对 MSP430 的运行方式和状态有了更深的认识，并且逐渐了解到一些编程过程中可以加以提高性能的注意点。

1.8 参考代码

```
#include <msp430.h>

#define CPU_F        ((double)1000000)
#define delay_ms(x)   _delay_cycles((long)(CPU_F*(double)x/1000.0))

int main(void)
{
   WDTCTL = WDTPW | WDTHOLD;           // 关闭看门狗
   _enable_interrupts();               // 打开全局中断

   TA0CTL = 0x0110;                    // 选择 ACLK 时钟及增计数模式
   TA0CCR0 = 32768;                    // CCR0 计数（总计数）
   TA0CCR1 = 21845;                    // CCR1 计数
   TA0CCR2 = 10923;                    // CCR2 计数
   TA0CCTL0 &= ~ 0x0011;               // 禁止计数器中断
   TA0CCTL1 &= ~ 0x0011;
   TA0CCTL2 &= ~ 0x0011;
                                       // 初始化 LED 及蜂鸣器
   P8DIR |= 0x02;                      // L1（P8.1）
   P3DIR |= 0xE0;                      // L2（P3.7），L6（P3.5），蜂鸣器（P3.6）
   P6DIR |= 0x08;                      // L4（P6.3）

   P8OUT &= ~ 0x02;
   P3OUT &= ~ 0xE0;
   P6OUT &= ~ 0x08;

   P1DIR &= ~ 0x0C;                    // 配置中断
   P1REN |= 0x0C;                      // S1（P1.2），S2（P1.3）
   P1OUT |= 0x0C;

   P1IE |= 0x0C;
   P1IES |= 0x0C;
   P1IFG &= ~ 0x0C;

   P2DIR &= ~ 0x48;
   P2REN |= 0x48;                    // S3（P2.3），S4（P2.6）
```

```
    P2OUT |= 0x48;

    P2IE |= 0x48;
    P2IES |= 0x48;
    P2IFG &= ~ 0x48;

    P3OUT |= 0x20;                          // 运行状态指示灯 L6
    return 0;
}

#pragma vector=PORT1_VECTOR
__interrupt void Port_1_Key(void) {
    if (P1IFG & 0x04) {                     // S1 中断
        TA0CCTL1 ^= 0x0010;                 // 允许/禁止 CCR1 中断状态翻转
        P8OUT &= ~ 0x02;
        P3OUT |= 0x20;

        P1IFG &= ~ 0x04;                    // 关中断标志
        delay_ms(100);
    }
    if (P1IFG & 0x08) {                     // S1 中断
        TA0CCTL2 ^= 0x0010;                 // 允许/禁止 CCR2 中断状态翻转
        P3OUT &= ~ 0x80;
        P3OUT |= 0x20;

        P1IFG &= ~ 0x08;                   // 关中断标志
        delay_ms(100);
    }
}

#pragma vector=PORT2_VECTOR
__interrupt void Port_2_Key(void) {
    if (P2IFG & 0x08) {                     // S1 中断
        TA0CCTL0 ^= 0x0010;                 // 允许/禁止 CCR0 中断状态翻转
        P6OUT &= ~ 0x08;
        P3OUT |= 0x20;

        P2IFG &= ~ 0x08;                   // 关中断标志
```

```
        delay_ms(100);
    }
    if (P2IFG & 0x40) {                 // S1 中断
        TA0CCTL0 &= ~ 0x0010;           // 全部禁止计数器中断
        TA0CCTL1 &= ~ 0x0010;
        TA0CCTL2 &= ~ 0x0010;

        P8OUT &= ~ 0x02;
        P3OUT &= ~ 0xE0;
        P6OUT &= ~ 0x08;
        P3OUT |= 0x20;

        P2IFG &= ~ 0x40;                // 关中断标志
        delay_ms(100);
    }
}

#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer_A0(void) {       // CCR0 中断
    P6OUT ^= 0x08;                      // 翻转 CCR0 指示灯 L4
    P3OUT ^= 0x60;                      // 翻转运行状态指示灯 L6 及蜂鸣器
}

#pragma vector=TIMER0_A1_VECTOR
__interrupt void Timer_A1(void) {
    switch(TA0IV) {                     // 判断中断向量
        case 2:                         // 0x0002 为 CCR1
            P8OUT ^= 0x02;              // 翻转 CCR1 指示灯 L1
            P3OUT ^= 0x60;              // 翻转运行状态指示灯 L6 及蜂鸣器
            break;
        case 4:                         // 0x0004 为 CCR2
            P3OUT ^= 0x80;              // 翻转 CCR2 指示灯 L2
            P3OUT ^= 0x60;              // 翻转运行状态指示灯 L6 及蜂鸣器
            break;
        default:
            break;
    }
}
```