PWM（Pulse Width Modulation）——脉宽调制，它是利用微控制器的数字输出来对模拟电路进行控制的一种非常有效的技术，广泛应用于测量、通信、功率控制与变换等许多领域。

s3c2440 芯片中一共有 5 个 16 位的定时器，其中有 4 个定时器（定时器 0~定时器 3）具有脉宽调制功能，因此用 s3c2440 可以很容易地实现 PWM 功能。下面就具体介绍如何实现 PWM 功能。

1、PWM 是通过引脚 TOUT0~TOUT3 输出的，而这 4 个引脚是与 GPB0~GPB3 复用的，因此要实现 PWM 功能首先要把相应的引脚配置成 TOUT 输出。

2、再设置定时器的输出时钟频率，它是以 PCLK 为基准，再除以用寄存器 TCFG0 配置的 prescaler 参数，和用寄存器 TCFG1 配置的 divider 参数。

3、然后设置脉冲的具体宽度，它的基本原理是通过寄存器 TCNTBn 来对寄存器 TCNTn（内部寄存器）进行配置计数，TCNTn 是递减的，如果减到零，则它又会重新装载 TCNTBn 里的数，重新开始计数，而寄存器 TCMPBn 作为比较寄存器与计数值进行比较，当 TCNTn 等于 TCMPBn 时，TOUTn 输出的电平会翻转，而当 TCNTn 减为零时，电平会又翻转过来，就这样周而复始。因此这一步的关键是设置寄存器 TCNTBn 和 TCMPBn，前者可以确定一个计数周期的时间长度，而后者可以确定方波的占空比。由于 s3c2440 的定时器具有双缓存，因此可以在定时器运行的状态下，改变这两个寄存器的值，它会在下个周期开始有效。

4、最后就是对 PWM 的控制，它是通过寄存器 TCON 来实现的，一般来说每个定时器主要有 4 个位要配置（定时器 0 多一个死区位）：启动/终止位，用于启动和终止定时器；手动更新位，用于手动更新 TCNTBn 和 TCMPBn，这里要注意的是在开始定时时，一定要把这位清零，否则是不能开启定时器的；输出反转位，用于改变输出的电平方向，使原先是高电平输出的变为低电平，而低电平的变为高电平；自动重载位，用于 TCNTn 减为零后重载 TCNTBn 里的值，当不想计数了，可以使自动重载无效，这样在 TCNTn 减为零后，不会有新的数加载给它，那么 TOUTn 输出会始终保持一个电平（输出反转位为 0 时，是高电平输出；输出反转位为 1 时，是低电平输出），这样就没有 PWM 功能了，因此这一位可以用于停止 PWM。

PWM 有很多用途，在这里我利用开发板的资源，用它来驱动蜂鸣器，并通过改变脉宽来改变蜂鸣器发声的频率。下面的程序就是利用 PWM 来驱动蜂鸣器，脉宽从低到高，再从高到低，周而复始。我们还利用 4 个 LED 来指示频率的高低，最高时 LED 全亮，最低时 LED 全灭。并且我们用两个按钮来分别暂停蜂鸣器和重新开启蜂鸣器：

```
#define _ISR_STARTADDRESS 0x33ffff00

#define U32 unsigned int

typedef unsigned char BOOL;

#define TRUE            1

#define FALSE           0

#define pISR_EINT0                  (*(unsigned *)(_ISR_STARTADDRESS+0x20))

#define pISR_EINT1                  (*(unsigned *)(_ISR_STARTADDRESS+0x24))

#define rSRCPND         (*(volatile unsigned *)0x4a000000)          //Interrupt request status

#define rINTMSK         (*(volatile unsigned *)0x4a000008)          //Interrupt mask control

#define rINTPND         (*(volatile unsigned *)0x4a000010)          //Interrupt request status

#define rGPBCON         (*(volatile unsigned *)0x56000010)          //Port B control

#define rGPBDAT         (*(volatile unsigned *)0x56000014)          //Port B data

#define rGPBUP          (*(volatile unsigned *)0x56000018) //Pull-up control B

#define rGPFCON         (*(volatile unsigned *)0x56000050)          //Port F control

#define rEXTINT0        (*(volatile unsigned *)0x56000088) //External interrupt control register 0
```

```
#define rTCFG0    (*(volatile unsigned *)0x51000000)          //Timer    configuration

#define rTCFG1    (*(volatile unsigned *)0x51000004)          //Timer    configuration

#define rTCON     (*(volatile unsigned *)0x51000008)          //Timer control

#define rTCNTB0 (*(volatile unsigned *)0x5100000c)            //Timer count buffer 0

#define rTCMPB0 (*(volatile unsigned *)0x51000010)          //Timer compare buffer 0


BOOL stop;


static void __irq Key1_ISR(void)                //暂停键，关闭蜂鸣器

{

        rSRCPND = rSRCPND | (0x1<<1);

        rINTPND = rINTPND | (0x1<<1);


        rTCON &= ~0x8;                  //禁止定时器自动重载，即关闭定时器

        stop = TRUE;

}


void __irq Key4_ISR(void)                //重启键，开启蜂鸣器

{

        rSRCPND = rSRCPND | 0x1;

        rINTPND = rINTPND | 0x1;
```

```
        stop = FALSE;
}



void delay(int a)
{
        int k;
        for(k=0;k<a;k++)
                ;
}


void Main(void)
{
        int freq;

        rGPBCON = 0x155556;             //B0 为 TOUT0，B5~B8 为输出，给 LED
        rGPBUP    = 0x7ff;
        rGPFCON = 0xaaaa;                   //F 口为 EINT，给按钮
        //按钮的一些必要配置
        rSRCPND = 0x07;
        rINTMSK = ~0x07;
        rINTPND =0x07;
```

```
    rEXTINT0 = 0x22;

    freq = 2500;

    rTCFG0 &= 0xFFFF00;
    rTCFG0 |= 0x31;       //prescal  是 49
    rTCFG1 &= ~0xF;       //1/2，因为 PCLK 为 50MHz，所以 50MHz/50/2=500kHz
    rTCNTB0 = 5000;
    rTCMPB0 = freq;
    rTCON &= ~0x1F;
    rTCON |= 0xf;                    //死区无效，自动装载，电平反转，手动更新，定时器开启
    rTCON &= ~0x2 ;   //手动更新位清零，PWM 开始工作

    pISR_EINT0 = (U32)Key4_ISR;
    pISR_EINT1 = (U32)Key1_ISR;

    stop = FALSE;

    rGPBDAT = ~0x60;                 //两个 LED 亮

    while(1)
    {
```

```
//频率递增
for ( ; freq<4950 ; )
{
    freq+=10;
    rTCMPB0 = freq;                //重新赋值
    delay(20000);

    while (stop == TRUE)           //暂停
    {
        delay(1000);
        if (stop ==FALSE)          //判断是否重启
        {
            rTCON &= ~0x1F;
            rTCON |= 0xf;
            rTCON &= ~0x2 ;                    //恢复 PWM 功
能

        }
    }
    //4 个 LED 随着频率的高低，时灭时亮
    if(freq == 100)
        rGPBDAT = ~0x1e0;
    if(freq == 1300)
```

```
            rGPBDAT = ~0xe0;
            if(freq == 2500)
                        rGPBDAT = ~0x60;
            if(freq == 3700)
                        rGPBDAT = ~0x20;
            if(freq == 4900)
                        rGPBDAT = ~0x0;

}

//频率递减
for( ; freq>50 ; )
{
            freq-=10;
            rTCMPB0 = freq;
            delay(20000);
            while (stop == TRUE)
            {
                        delay(1000);
                        if (stop ==FALSE)
                        {
                                    rTCON &= ~0x1F;
```

```
                    rTCON |= 0xf;
                    rTCON &= ~0x2 ;
                }
            }
            if(freq == 100)
                rGPBDAT = ~0x1e0;
            if(freq == 1300)
                rGPBDAT = ~0xe0;
            if(freq == 2500)
                rGPBDAT = ~0x60;
            if(freq == 3700)
                rGPBDAT = ~0x20;
            if(freq == 4900)
                rGPBDAT = ~0x0;
        }
    }
}
```

这里还需要说明几点：

1、开发板上的蜂鸣器是高电平发声，低电平停止，而 TOUT0 定时无效时，是高电平输出，因此为了使 PWM 无效时，蜂鸣器不发声，我把输出电平进行了反转处理（置 TCON 中的输出反转位）；

2、在这里，我是通过按键把 stop 标志变量置为 FALSE 来跳出 while 循环，重新开始蜂鸣，但不知什么原因，如果在 while 循环内不加一段等待时间，则永远不能跳出循环体，因此我不得不加了一个 delay 函数，让它等待一段时间。关于这个问题，我还给不出一个满意的解释，也不知是哪里出了问题！