在讲解之前，先介绍一下 s3c2440 时钟系统。一般来说，MCU 的主时钟源主要是外部晶振或外部时钟，而用的最多的是外部晶振。在正确情况下，系统内所使用的时钟都是外部时钟源经过一定的处理得到的。由于外部时钟源的频率一般不能满足系统所需要的高频条件，所以往往需要 PLL（锁相环）进行倍频处理。在 s3c2440 中，有 2 个不同的 PLL，一个是 MPLL，另一个是 UPLL。UPLL 是给 USB 提供 48MHz。在这里，我们主要介绍 MPLL。外部时钟源经过 MPLL 处理后能够得到三个不同的系统时钟：FCLK、HCLK 和 PCLK。FCLK 是主频时钟，用于 ARM920T 内核；HCLK 用于 AHB 总线设备，如 ARM920T，内存控制，中断控制，LCD 控制，DMA 以及 USB 主模块；PCLK 用于 APB 总线设备，如外围设备的看门狗，IIS，I2C，PWM，MMC 接口，ADC，UART，GPIO，RTC 以及 SPI。这三个系统时钟（FCLK、HCLK 和 PCLK）是有一定的比例关系，这种关系是通过寄存器 CLKDIVN 中的 HDIVN 位和 PDIVN 位来控制的，因此我们只要知道了 FCLK，再通过这两位的控制，就能确定 HCLK 和 PCLK。而 FCLK 是如何得到的呢？它是通过输入时钟（即外部时钟源）的频率，经过一个计算公式（具体公式请查阅数据手册）得到的，这个计算公式还需要三个参数（MDIV、PDIV、SDIV），而这三个参数是经过寄存器 MPLLCON 配置得到的。最后，我们用最清晰的线路来绘制一下时钟的产生过程：外部时钟源→通过寄存器 MPLLCON 得到 FCLK→再通过寄存器 CLKDIVN 得到 HCLK 和 PCLK。这个配置过程在启动文件中就已完成。在本开发板上，外部晶振为 12MHz，进过 MPLL 倍频以后得到 400MHz 的 FCLK，而 FCLK、HCLK、PCLK 之间的比例关系为 1：4：8，因此 HCLK 为 100MHz，PCLK 为 50MHz。

3c2440 的时钟系统就介绍到这里，我们再回到定时器的配置上来。如何才能得到精确的定时呢？那就要靠 TCFG0 和 TCFG1 这两个寄存器来配置定时器的频率，即要确定 TCNTOn 每递减一个数所需要的时间，它们之间是倒数的关系。具体的计算公式为：

定时器输出时钟频率=PCLK ÷ (prescaler+1) ÷ divider

其中 prescaler 值由 TCFG0 决定，divider 值由 TCFG1 决定，而 prescaler 只能取 0~255 之间的整数，divider 只能取 2、4、8 和 16。比如已知 PCLK 为 50MHz，而我们想得到某一定时器的输出时钟频率为 25kHz，则依据公式可以使 prescaler 等于 249，divider 等于 8。有了这个输出时钟频率，理论上我们通过设置寄存器 TCNTBn 就可以得到任意与 0.04 毫秒（1÷25000×1000）成整数倍关系的时间间隔了。例如我们想要得到 1 秒钟的延时，则使 TCNTBn 为 25000（1000÷0.04）即可。

下面我们通过一段程序来演示利用定时器得到精确延时。这里我们用到的是定时器 4。这段程序的作用是让蜂鸣器每隔 2 秒钟响一次，持续时间为 0.5 秒，蜂鸣器响的同时伴随着 LED 亮。

```
#define _ISR_STARTADDRESS 0x33ffff00
#define U32 unsigned int
#define pISR_TIMER4             (*(unsigned *)(_ISR_STARTADDRESS+0x58))
#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 rTCFG0    (*(volatile unsigned *)0x51000000)          //Timer 0 configuration
#define rTCFG1    (*(volatile unsigned *)0x51000004)          //Timer 1 configuration
#define rTCON     (*(volatile unsigned *)0x51000008)          //Timer control
#define rTCNTB4 (*(volatile unsigned *)0x5100003c)           //Timer count buffer 4
void __irq Timer4_ISR(void)
{
        static int count;
        count ++;
        rSRCPND = rSRCPND | (0x1<<14);
        rINTPND = rINTPND | (0x1<<14);
        //每隔 2 秒蜂鸣器响一次，持续时间为 0.5 秒，并伴随着 LED 亮
        if (count % 4 ==0)
                rGPBDAT = ~0x1e0;                   //蜂鸣器响，LED 亮
        else if (count % 4 ==1)

                rGPBDAT = 0x1e0;          //蜂鸣器不响，LED 灭

}
void Main(void)
{

        rGPBCON = 0x155555;                         //B0 输出，给蜂鸣器；B5~B8 输出，给 LED

        rGPBUP    = 0x7ff;
        rGPBDAT = 0x1e0;          //蜂鸣器不响，LED 灭
        rSRCPND = rSRCPND | (0x1<<14);
        rINTPND = rINTPND | (0x1<<14);
        rINTMSK = ~(0x1<<14);                   //打开定时器 4 中断
        rTCFG0 &= 0xFF00FF;
        rTCFG0 |= 0xf900;             // prescaler 等于 249
        rTCFG1 &= ~0xF0000;
        rTCFG1 |= 0x20000;        //divider 等于 8，则设置定时器 4 的时钟频率为 25kHz
        rTCNTB4 = 12500;                        //让定时器 4 每隔 0.5 秒中断一次
        rTCON &= ~0xF00000;
        rTCON |= 0x700000;
        rTCON &= ~0x200000 ;                    //定时器 4 开始工作
```

```
        pISR_TIMER4 = (U32)Timer4_ISR;
        while(1)
        {       ;

        }
}
```