s3c2440 实时时钟（RTC）中，定义了两个中断源：报警中断和时间节拍中断。前面有网友问到了这两个中断的用法，最近我抽出时间对这两个中断研究了一番，发现这两个中断都很实用。现在就给大家介绍一下它们的用法。

时间节拍中断，顾名思义，就像一个节拍器，可以等时性的控制节拍。因此它类似于定时器中断。但时间节拍中断是毫秒级的，而定时器中断可以达到微秒，甚至更小级别。时间节拍中断的周期公式为：(n+1)÷128，单位是秒，即每隔这么长时间，会中断一次。其中 n 的值为 1~127，它存储在寄存器 TICNT 的低 6 位中，当寄存器 TICNT 的第 7 位被置 1 时，表示开启时间节拍中断，这时 n 递减，当减为 0 时，进入时间节拍中断。

报警中断可以实现当实时时间达到预置的时间后，引起报警。预置的时间是存储在报警时间数据寄存器中的，包括 ALMYEAR（年）、ALMMON（月）、ALMDATE（日）、ALMHOUR（小时）、ALMMIN（分）和 ALMSEC（秒）。而如何报警，是由报警控制寄存器 RTCALM 控制的。它的第 6 位置 1 表示全局报警，而第 5 位到第 0 位置 1 分别表示年、月、日、小时、分和秒报警。比如，我们想要在 2010 年 4 月 5 日 22 时 30 分 0 秒报警，那么把这个时间分别存储到相应的报警时间数据寄存器中，然后设置 RTCALM 为 0x7F，这样当实时时钟到达这个时刻时，会引起报警中断；又比如我们想要系统具有闹钟的功能，让它每天早上 6 点提醒我们起床，那么我们可以设置 ALMHOUR 为 6，RTCALM 为 0x44。如果我们只想让系统在 4 月份的时候提醒我们 6 点起床，那该怎么办呢？这个问题对于 s3c2440 来说就是小菜一碟，只要我们再在 ALMMON 里写入 4，然后把 RTCALM 改为 0x54 即可。总之，就是系统根据 RTCALM 所置 1 的相应位来比较相对应的当前时间与报警时间数据寄存器中的值，如果相等就进入中断。

我们对上一篇的程序进行改写，加入报警中断和时间节拍中断。PC 机通过 UART 不仅可以对 s3c2440 的实时时钟进行修改，还可以设置报警时间。其中设置报警时间的通信协议与设置实时时钟的相似，即：第一个字节为 0xBB，表示命令，后面的 6 个字节分别是设置报警时间的年、月、日、小时、分和秒，最后一个字节用于设置 RTCALM。当报警时间到时，我们利用时间节拍中断来控制 LED 闪烁，闪烁 15 秒后自动停止，也可以通过一个按键来中止 LED 闪烁。下面的程序只列出了主要的部分：

```
……    ……

unsigned char alarm_buffer[7];                    //报警缓存数组

……    ……
```

```
//设置报警时间
void set_alarm(void)
{
	rALMYEAR = alarm_buffer[0];		//年
	rALMMON = alarm_buffer[1];		//月
	rALMDATE = alarm_buffer[2];		//日
	rALMHOUR = alarm_buffer[3];		//小时
	rALMMIN = alarm_buffer[4];		//分
	rALMSEC = alarm_buffer[5];		//秒
	rRTCALM = alarm_buffer[6];		//报警控制
}

//按键外部中断，用于禁止时间节拍中断，中止 LED 闪烁
void __irq Key1_ISR(void)
{
	rSRCPND = rSRCPND | (0x1<<1);
	rINTPND = rINTPND | (0x1<<1);

	rGPBDAT = 0x1e0;			//LED 灭
	rTICNT = 0x0;		//禁止时间节拍中断
}
```

```
//UART 中断，与上一篇文章中的相关内容相比，进行了改写和完善
void __irq uartISR(void)
{
        char ch;
        static char command;
        static char count;
        rSUBSRCPND |= 0x3;
        rSRCPND |= 0x1<<28;
        rINTPND |= 0x1<<28;

        if(rUTRSTAT0 & 1) //接收数据处理部分
        {

                ch = rURXH0; //接收字节数据
                if(command==0)                  //判断命令信息
                {
                        switch(ch)
                        {
                        case 0xaa:              //设置实时时钟时间
                                command = 0xaa;
                                count=0;
```

```
            break;
            case 0xbb:                  //设置报警时间
                command = 0xbb;
                count=0;
            break;
            default:                    //其余命令
                command = 0;
                count =0;
                rUTXH0=ch;
            break;
            }
}
else                        //接收实时时钟时间或报警时间
{
            if(command == 0xaa)             //实时时钟时间
            {
                date_buffer[count]=ch;
                count++;
                if(count==7)
                {
                    set_date();
                    count=0;
```

```
                    command=0;
                    flag=1;
                    rUTXH0=0xaa;
                }
            }
            else if(command ==0xbb)                    //报警时间
            {
                alarm_buffer[count]=ch;
                count++;
                if(count==7)
                {
                    set_alarm();
                    count=0;
                    command=0;
                    rUTXH0=0xbb;
                }
            }
        }
    }
}

//报警中断
```

```
void __irq Alarm_ISR(void)
{
        rSRCPND |= 0x1<<30;
        rINTPND |= 0x1<<30;

        rTICNT = 0xbf;        //开启时间节拍中断，周期为 500 毫秒
}

//时间节拍中断，用于 LED 闪烁 15 秒
void __irq RTCTick_ISP(void)
{
        static char count;
        rSRCPND |= 0x1<<8;
        rINTPND |= 0x1<<8;

        if(count%2==0)                //LED 亮 0.5 秒
                rGPBDAT = ~0x1e0;
        else                          //LED 灭 0.5 秒
                rGPBDAT = 0x1e0;
        count++;
        if(count==30)
        {
```

```
            rTICNT = 0x0;                   //禁止时间节拍中断
            rGPBDAT = 0x1e0;                //LED 灭
            count=0;
        }
}

void Main(void)
{
        //初始化
        ……      ……
        //中断源
pISR_UART0 = (U32)uartISR;
pISR_EINT0 = (U32)Key4_ISR;
        pISR_EINT1 = (U32)Key1_ISR;
pISR_RTC = (U32)Alarm_ISR;
        pISR_TICK = (U32)RTCTick_ISP;
        ……      ……
        Brush_Background(0xffffff);
        show_date();

        flag=0;
```

```
        while(1)
        {
            if(flag)                    //显示实时时间
            {
                Brush_Background(0xffffff);
                show_date();
                flag=0;
            }
}
}
```