ARM 异常之中断

中断：程序运行过程中，系统外部、系统内部或程序本身出现紧急事件需处理器干预时，处理器立即中止现行程序的运行，自动转入相应的处理程序(中断处理程序)，待处理完后，再返回原来被暂停的程序继续运行，这整个过程称为程序中断。

分类：　硬件中断、　软件中断（软中断，即 SWI 异常）

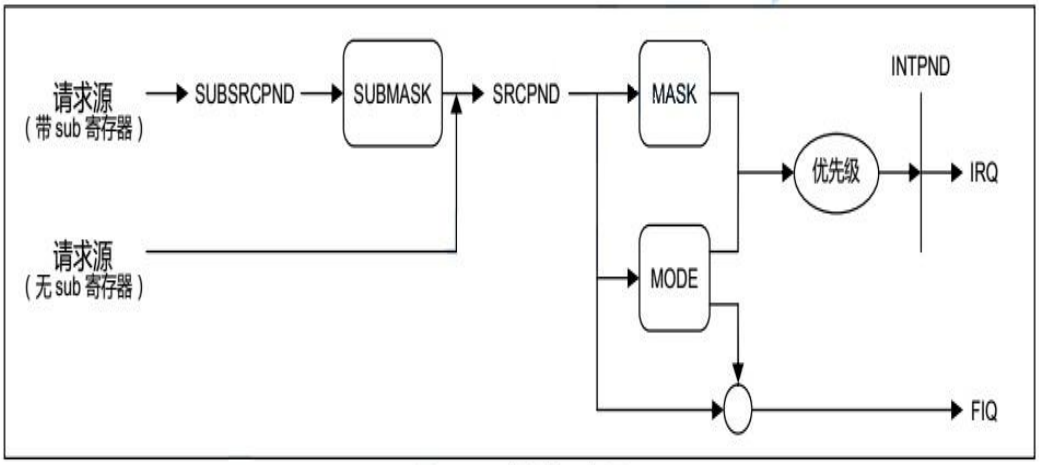


图 14-1. 中断处理框图

看门狗定时器（WDT，Watch Dog Timer）是一套简单的电路，它实际上是一个计数器。

有 2 个功能：

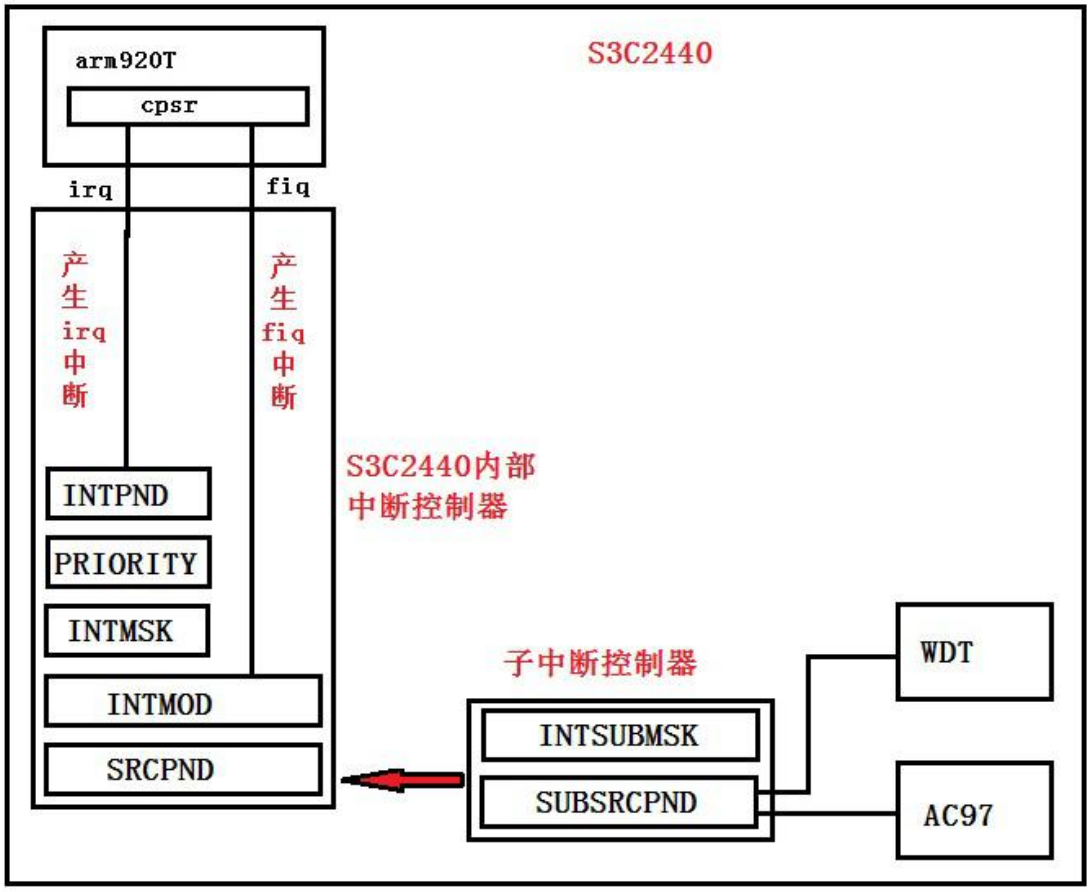
（1）复位。

电路的内部机制：给看门狗一个初始值，程序开始运行后看门狗开始倒计数减一。如果程序运行正常，当看门狗计数器快到 0 值时，CPU 会自动发出指令让看门狗复位（再给计数器填充一个值），重新开始倒计数。如果看门狗减到 0，没有被填值，我们就认为程序没有正常工作，此时，看门狗强制整个系统复位。

（2）普通的定时器

普通的定时器主要是用来计时。当时间到了后，会产生一个中断来通知 CPU 设定的时间到了。

例：假设设置定时器的工作频率 1000HZ，定时器的计数值设置成 5000；这样当打开定时器中断，并开启定时器后，计数值就每 1ms 减一次 1,当 5 秒后，计数值减到 0，就会生成一个中断。至于中断怎么处理，就看程序的需要了。



s3c2440 内部中断控制器框图

SRCPND： 中断源寄存器。标记哪些中断产生。

INTMOD：中断方式。 0：IRQ 1：FIQ

INTMSK：中断屏蔽寄存器。可屏蔽一些不需要的中断。

PRIORITY： 中断优先级寄存器。如果有多个中断，可设置其优先级。

INTPND： 中断寄存器。用来给 ARM 产生 IRQ 中断。

SUBSRCPND：子中断源寄存器，用来区别是具体的哪种中断。

例如：在 SRCPND 中无法区分到底是 WDT 还是 AC97。因此，SUBSRCPND 寄存器用来区别 WDT 中断还是 AC97 中断。

中断处理的流程：（以 watchdog 为例）

1、设置 arm 内部开启中断响应。

```
__asm__ __volatile__ (
            "mrs r0, cpsr\n"
            "bic r0, r0, #0x80\n"
            "msr cpsr, r0\n"
            :
            :
            :"r0"
);
```

2、设置 s3c2440 中断控制器，开启中断响应。

```
SUBSRCPND |= (1 << 13);
INTSUBMSK &= ~(1 << 13);
SRCPND |= (1 << 9);
INTMOD &= ~(1 << 9);
INTMSK &= ~(1 << 9);
INTPND |= (1 << 9);
```

3、配置设备为中断模式。以 watchdog 为例，就是设置 watchdog 为产生中断功能，而不是复位功能。

```
WTCON |= (1 << 2) | (3 << 3) | (1 << 5) | (0xfe << 8);
WTCNT = 0x1000;
WTDAT = 0x500;
```

4、当中断产生后，会是 arm 进入 IRQ 异常模式，arm 会跳转到异常向量表地址为 0x18 处执行中断处理程序，使 arm 处于 IRQ 异常模式下。中断处理程序中一般需要清除本次中断操作。

```
SUBSRCPND |= 1 << 13;
SRCPND |= 1 << 9;
INTPND |= 1 << 9;
```

技巧：从汇编中想跳转到没有一起编译链接的 C 文件中的某个函数，并执行该函数的操作方法：

main.c 中：

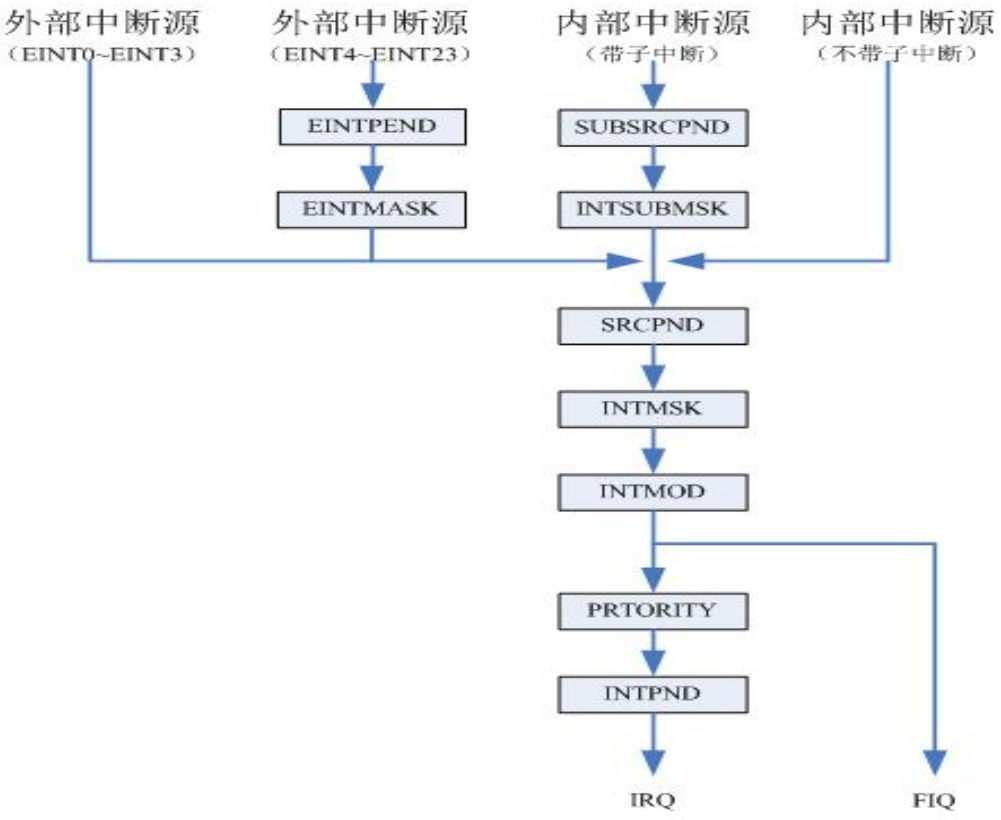
```
unsigned long* p = 0x32000000;
*p = do_irq;
void do_irq()
{
   中断处理程序
}
```

vector.S 中：

```
mov r0, #0x32000000
ldr r1, [r0]
mov lr, pc
mov pc, r1
```

四、外部中断

外部中断到达 cpu 的流程框图：



中断的开启（中断初始化，INTMOD 和 PRTORITY 使用默认值）

（a）如果是外部中断（EINT0~EINT3）和内部中断（不带子中断），需设置 INTMSK，让它不屏蔽中断即可；

（b）如果是带子中断的内部中断，需设置 INTSUBMSK 和 INTMSK，让它们不屏蔽中断即可；

（c）如果是外部中断(EINT4~EINT23)，需设置 EINTMASK 和 INTMSK，让它们不屏蔽中断即可；

中断的开启，以外部按键 S4、S5 为例：

```
EXTINT1 &= ~(7 << 12);
EXTINT1 |= (3 << 12);
EXTINT2 &= ~(7 << 12);
EXTINT2 |= (3 << 12);
EINTMASK &= (~(1 << 11) & ~(1 << 19));
GPGCON &= ~(3 << 6) & (~(3 << 22));
GPGCON |= (2 << 6) | (2 << 22);
```

清除中断：

```
SRCPND |= (1 << 5);
INTPND |= (1 << 5);
EINTPEND |= (1 << 11) | (1 << 19);
```