Realview MDK 中编译器对中断处理的过程详解

在 ARM 程序的开发过程中，对中断的处理是很普遍的、也是相当重要的。Realview MDK 使用的 RVCT 编译器提供了__irq 关键字，用此关键字修饰的函数被作为中断出来函数编译，即在编译的过程中，编译器会自动添加中断处理过程中现场保护和恢复的代码，减小程序的开发难度，加快软件的开发过程。

在理解__irq 关键字的作用之前，先看一下 ARM 核对异常的处理过程。当产生异常时, ARM 核拷贝 CPSR 寄存器的内容 SPSR_<mode>寄存器中，同时设置适当的 CPSR 位、改变处理器状态进入 ARM 态和处理器模式，从而进入相应的异常模式。在设置中断禁止位禁止相应中断(如果需要)后，ARM 核保存返回地址到 LR_<mode>，同时设置 PC 为相应的异常向量。当异常返回时, 异常处理需要从 SPSR_<mode>寄存器中恢复 CPSR 的值，同时从 LR_<mode>恢复 PC，具体的异常返回指令如下：

- 从 SWI 和 Undef 异常返回时使用：

```
movs pc, LR;
```

- 从 FIQ、IRQ 和预取终止返回时使用：

```
SUBS PC, LR，#4;
```

- 从数据异常返回时使用：

```
SUBS PC, LR，#8
```

在使用上述指令异常返回时，如果 LR 之前被压栈的话使用 LDM “^”, 例如：

```
LDMFD SP!, {PC}^
```

理解了 ARM 异常处理的过程以后，Realview MDK 中__irq 关键字的作用就容易理解了。下面的函数为一个中断处理函数，其前面加了__irq 关键字。

```
__irq void pwm0_irq_handler(void)
{
    //Deassert PWM0 interrupt signal
    unsigned int i=AT91F_PWMC_GetInterruptStatus(AT91C_BASE_PWMC);

    // Clear the LED's. On the Board we must apply a "1" to turn off LEDs
    AT91F_PIO_SetOutput(AT91C_BASE_PIOA, led_mask[0]);
    AT91F_PWMC_StopChannel(AT91C_BASE_PWMC,AT91C_PWMC_CHID1);

    AT91F_AIC_ClearIt(AT91C_BASE_AIC,AT91C_ID_PWMC);
    AT91F_AIC_AcknowledgeIt(AT91C_BASE_AIC);
}
```

当编译器器编译这个函数时，除了保存 ATPCS 规则规定的寄存器以外，还保存了 CPSR 及 PC 的值。在函数的返回时，还自动添加了 SUBS PC, LR, #4 和从 SPSR 寄存器恢复 CPSR 寄存器值的指令。用这种方式处理以后，中断处理函数可以和普通函数一样的使用。

注意：中断处理都是在 ARM 模式下进行的，当源程序欲编译成 Thumb 指令时，这时，用__irq 关键字修饰的函数仍然会被编译成 ARM 指令。如果源程序编译成在 CORTEX M3 上运行的指令时，关键字__irq对函数的编译没有任何影响，即编译器不会自动保存CPSR及PC的值，也不会添加 SUBS PC, LR, #4 和从 SPSR 寄存器恢复 CPSR 寄存器值的指令，因为 CORTEX M3 处理器硬件会自动处理这些问题，无需软件开发人员关心。