STM32中断

6.1 概述

异常与中断

异常与中断的区别在于，那240个中断对Cortex-M3来说是来自核外部的请求信号，对Cortex-M3核来说是‘异步’的。而异常则是因Cortex-M3核的活动产生的，在执行指令或访问存储器时产生的，对Cortex-M3来说是同步的。

中断与外部中断

在内核层次讨论时，“中断”都是内核以外产生的，都叫做“外部中断”。

在芯片层次讨论时，通过EXTI控制的那几个中断（引脚、RTC时钟、USB唤醒、PVD输出），叫说“外部中断”。

有三个异常的优先级时负数，并且时固定的：复位、不可屏蔽中断（NMI）、硬件失效。这三个异常的级别比任何其他异常都要高。软件优先级数字越大优先级越低。

6.2 嵌套向量中断控制器（NVIC）

6.2.4 STM32中断优先级

在STM32中，有两个优先级的概念————抢占式优先级和响应优先级。

1. 具有高抢占式优先级的中断可以在具有低抢占式优先级的中断处理过程中被相应，即中断嵌套，或者说高抢占式优先级的中断可以嵌套低抢占式优先级的中断。
2. 当两个中断源的抢占式优先级相同时，这两个中断将没有嵌套关系。当一个中断到来后，如果正在处理另一个中断，这个后到来的中断就要等到前一个中断处理完之后才能处理。如果两个中断同时到达，则中断控制器根据他们的相应优先级高低来决定先处理哪一个。如果他们的抢占式优先级和相应优先级都相等，则根据他们在中断表中的排位顺序决定先处理哪一个。
3. 响应优先级不可以中断嵌套。

Cortex-M3内核允许具有较少中断源时使用较少的寄存器位指定中断源的优先级，因此STM32把指定中断优先级的寄存器位减少到4位。这4个寄存器位对应有5中分组方式：

1. 第0组：全部4位用于响应优先级（16级）
2. 第1组：最高1位用于抢占式优先级（2级），剩下3位用于响应优先级（8级）
3. 第2组：最高2位用于抢占式优先级（4级），剩下2位用于响应优先级（4级）

4. 第3组：最高3位用于抢占式优先级（8级），剩下1位用于响应优先级（2级）
5. 第4组：全部4位用于抢占式优先级（16级）

通过调用库函数void NVIC_PriorityGroupConfig(u32 NVIC_PriorityGroup)选择使用哪种分组方式：

```
NVIC_PriorityGroup_0       // 选择第0组
NVIC_PriorityGroup_1       // 选择第1组
NVIC_PriorityGroup_2       // 选择第2组
NVIC_PriorityGroup_3       // 选择第3组
NVIC_PriorityGroup_4       // 选择第4组
```

最后，使用库函数void NVIC_Init(NVIC_InitTypeDef *NVIC_InitStruct)库函数进行中断优先级设置。

6.2.5 NVIC配置过程

NVIC_PriorityGroupConfig函数

void NVIC_PriorityGroupConfig(u32 NVIC_PriorityGroup)

设置优先级分组：先占优先级和从优先级。

NVIC_Init函数

void NVIC_Init(NVIC_InitTypeDef *NVIC_InitStruct)

根据NVIC_InitSturct中指定的参数初始化外设NVIC寄存器。

NVIC配置例程：

```
// 配置NVIC，不考虑中断优先级的情况
static void nvic_init()
{
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);
    NVIC_InitTypeDef nvic_init;
    nvic_init.NVIC_IRQChannel = EXTI0_IRQChannel;
    nvic_init.NVIC_IRQChannelCmd = ENABLE;
    nvic_init.NVIC_IRQChannelPreemptionPriority = 0;
    nvic_init.NVIC_IRQChannelSubPriority = 0;
    NVIC_Init(&nvic_init);
}
```

6.3 EXTI外部中断

6.3.1 EXTI硬件结构

在STM32F103XX处理器中，外部中断/时间控制器由用于产生事件/中断请求的19个边沿检测器组成，其中16个中断通道EXTI0 ~ EXTI15对应GPIOx_Pin0 ~ GPIOx_Pin15，另外3个是EXTI16连接PVD输

出。EXTI17连接到RTC和EXTI18连接到USB唤醒事件。

中断与事件的区别

中断需要CPU参与，需要软件的中断服务函数才能完成中断后产生的结果，但是事件是靠脉冲发生器产生一个脉冲，进而由硬件自动完成这个事件产生的结果。

6.3.4 EXTI库函数

EXTI_Init函数

void EXTI_Init(EXTI_InitTypeDef *EXTI_InitStruct)

根据EXTI_InitStruct中指定的参数初始化外设EXTI寄存器。

注意，外部中断线中，Px0 ~ Px4在NVIC的channel配置中，使用EXTI0_IRQn ~ EXTI4_IRQn。而Px5 ~ Px9使用EXTI9_5_IRQn，Px10 ~ Px15使用EXTI15_10_IRQn。

EXTI中断操作过程

1. 首先设定好中断通道，具体就是配置GPIO端口的工作方式，配置GPIO和EXTI映射关系；
2. 接着是中断处理，配置EXTI触发条件，配置响应NVIC，根据中断编号对应到中断向量表查找服务函数的入口地址。
3. 中断响应，当到达中断触发条件时，内核从主程序跳到相应的中断向量处，根据中断向量的地址信息，跳转到中断服务函数入口地址执行。

EXTI中断应用实例：

```
/*
    利用外接的按钮产生外部中断信号，并实现对LED灯的控制。
*/

// 配置外部中断

static void exti_init()
{
    EXTI_InitTypeDef exti_init;
    exti_init.EXTI_Line = EXTI_Line0;                                   // PA0按钮按下产生外部中断信
    exti_init.EXTI_Mode = EXTI_Mode_Interrupt;
    exti_init.EXTI_Trigger = EXTI_Trigger_Rising_Falling;
    exti_init.EXTI_LineCmd = ENABLE;
    EXTI_init(&exti_init);
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource0);     // 配置连接引脚到PA0
}

// 配置NVIC

static void nvic_init()
{
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);
    NVIC_InitTypeDef nvic_init;
    nvic_init.NVIC_IRQChannel = EXTI0_IRQn;
    nvic_init.NVIC_IRQChannelCmd = ENABLE;
    nvic_init.NVIC_IRQChannelPeemptionPriority = 0;
    nvic_init.NVIC_IRQChannelSubPriority = 0;
    NVIC_Init(&nvic_init);
}

// 配置中断响应函数
// 中断响应函数的位置在stm32f10x_it.c中

void EXTI0_IRQHandler()
{
    if(readKey())
        led_light();
    else led_dark();

    EXTI_ClearITPendingBit(EXTI_Line0);
}

int main()
{
    led_init();
    key_init();
    exti_init();
    nvic_init();
    while(1);

    return 0;
}
```

NVIC与EXTI配置总结

NVIC配置

1. 设定NVIC的优先级分组组别：void NVIC_PriorityGroupConfig(u32 NVIC_PriorityGroup)。
2. 初始化nvic
   初始化NVIC_InitTypeDef类型变量nvic_init，并设置NVIC_IRQChannel、NVIC_IRQChannelCmd、NVIC_IRQChannelPeemptionPriority、NVIC_IRQChannelSubPriority等字段。
   调用初始化函数NVIC_Init()完成初始化。

EXTI配置

1. 初始化exti
   初始化EXTI_InitTypeDef类型变量exti_init，并设置EXTI_Line、EXTI_Mode、EXTI_LineCmd、EXTI_Trigger等字段。
   调用初始化函数EXTI_Init()完成初始化。
   如果使用了GPIO口作为外部中断入口，则还需要调用GPIO_EXTILineConfig()来进行配置。参数：GPIO_PortSourceGPIOx、GPIO_PortSourcex。
2. 编写中断处理函数
   对应NVIC哪个中断Channel则编写对应哪个中断处理函数。如若使用的是EXTI0_IRQChannel，则编写中断处理函数EXTI0_IRQHandler()。在中断处理函数的最后要清除中断标志：EXTI_ClearITPendingBit(EXTI_Linex)。