第六天笔记

1. 程序的运行方式

* 轮询系统

指的是在程序运行时，首先对所有的硬件进行初始化，然后在主程序中写一个死循环，需要运行的功能按照顺序进行执行，轮询系统是一种简单可靠的方式，一般适用于在只需要按照顺序执行的并且没有外部事件的影响的情况下。

程序的运行过程中出现如按键等需要外部检测的事件，轮询系统的实时响应能力变得很差。

int main(void)

{

//对硬件进行初始化

//编写死循环

while(1)

{

//程序1

//程序2

.....

}

}

* 前后台系统

相比于轮询系统，前后台系统增加**中断**的概念，如果外部事件发生，则在中断中进行处理，主程序在轮询系统中运行，**中断被称为前台**，**主程序中的while(1)就称为后台**。中断会终止后台程序的运行，然后跳转到对应的中断服务函数中去处理，处理完成后，在继续执行后台的程序。

如果使用前后台系统，会大大的提高程序的实时响应能力，避免造成外部事件的缺失。

int main(void)

{

//硬件的初始化

//注册中断，设置中断的触发条件

//在死循环进行执行后台程序

while(1)

{

//程序1

//程序2

.....

}

}

//中断服务函数

void XXX\_IQR(void)

{

//处理和响应中断

//返回主程序进行运行

}

* 多任务系统

相比于前后台系统，多任务系统的外部事件也是在中断中进行响应，但是外部事件的处理是任务中进行处理。任务具有优先级，优先级高的任务先处理，所以程序就会被分割为一个个的任务，任务是一个独立的死循环，并且不能返回，可以由操作系统进行任务的调度，程序段的实时响应能力又得到提升。

void task1(void) //任务1

{

while(1)

{

//执行任务1的功能

}

}

void task2(void) //任务2

{

while(1)

{

//执行任务2的功能

}

}

int main(void)

{

//对硬件进行初始化

//注册中断，设置中断的触发条件

//任务调度，由操作系统进行任务调度

//在死循环进行执行后台程序

while(1)

{

}

}

1. 外部中断的概述
2. 中断的概念

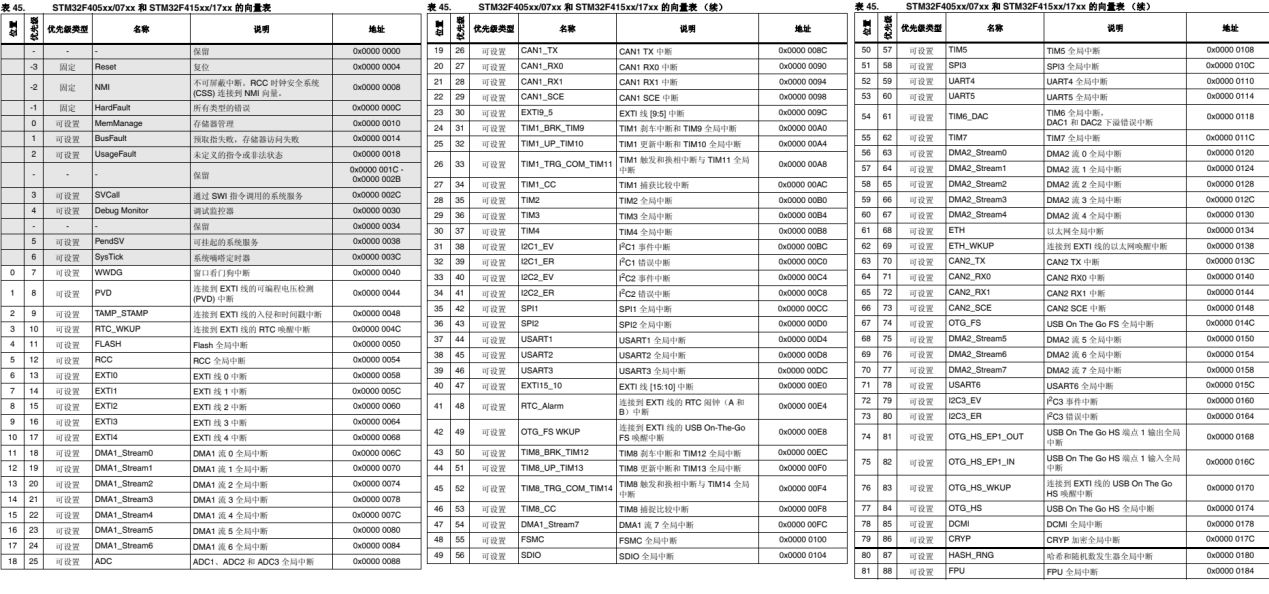
中断指的是CPU来处理和响应外部发生的异常，中断也就意味着打断，比如打断正在做的事，然后去处理一个紧急的事，处理完成后在继续做刚才没做完的事。比如打游戏，女朋友来电话。

1. 中断源分析

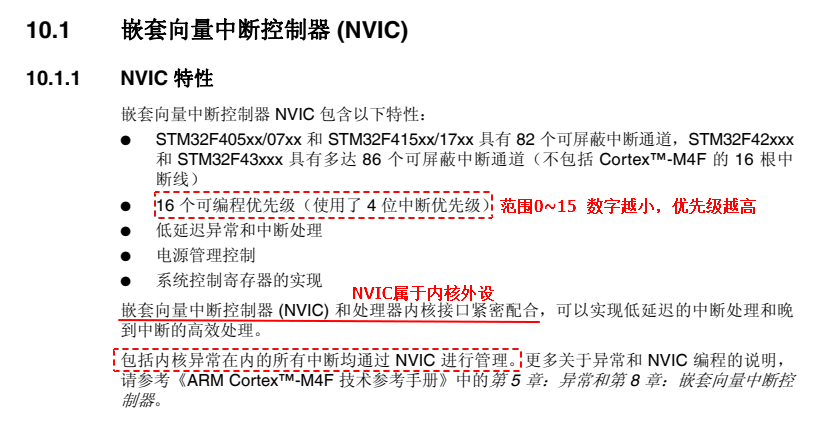
中断源指的是中断发生的源头，中断源在内核中已经定义好了，也称为向量表，向量表在STM32F4中文参考手册参考。

Cortex-M4内核一共支持256个中断，其中有16个内核中断，240个外部中断，只不过对于STM32F407系列来说，只用到了一部分，包含了**10个内核中断**（不可屏蔽中断，无法通过软件进行控制）、**82个外部中断**（可屏蔽中断，可以通过软件进行控制）。 **共92个。**

对于STM32的中断异常而言，分为两类：**内核异常** + **外部异常**，都可以参考下图进行分析



1. NVIC的概述

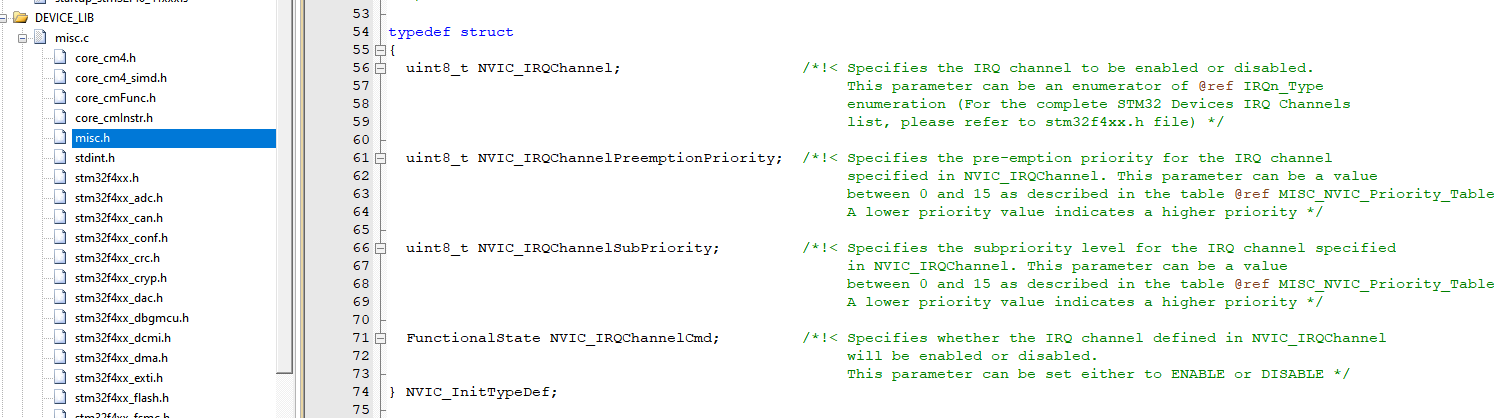


NVIC指的是嵌套向量中断控制器，属于内核中的外设，作用是管理所有的中断，比如中断的使能或失能、中断的优先级.....。

不管是Cortex A系列还是Cortex M系列的内核内部均有NVIC，通过NVIC来管理内核异常和外部异常。

* 中断的使能与失能

NVIC管理中断通道的打开与关闭，可以把NVIC理解为所有中断的开关，想要使用中断发送中断请求，就必须提前打开中断的通道。关于NVIC的使用都存储在一个结构体中，这个结构体和NVIC的函数接口都定义在misc.c和misc.h中。



* 中断的优先级设置

NVIC利用4bit的优先级来管理所有的中断通道，STM32中断的优先级分为两种：抢占式优先级（主优先级） + 响应式优先级（次优先级），每种都有16个优先级（0~15），**数字越小，优先级越高。**

意义：如果同时发生多个中断请求，但是又不能同时处理，就根据中断请求的优先级来处理和响应中断。比如：**女朋友问：我和你妈掉水里，你先救谁？**

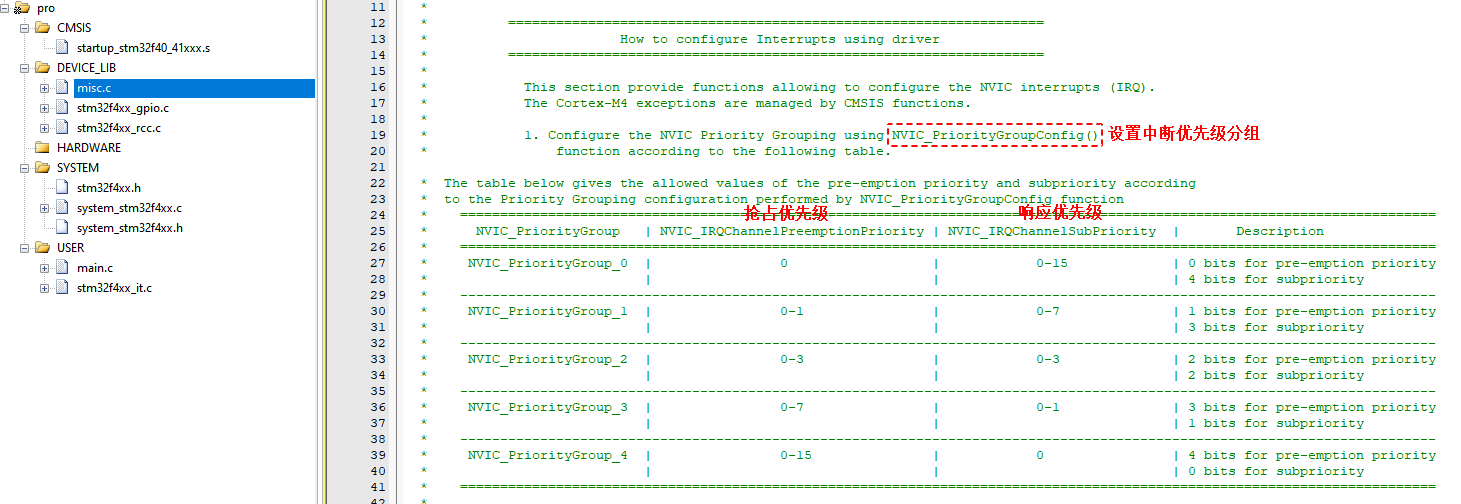
抢占优先级（主优先级）：抢占优先级高的中断可以打断抢占优先级低的中断的执行。

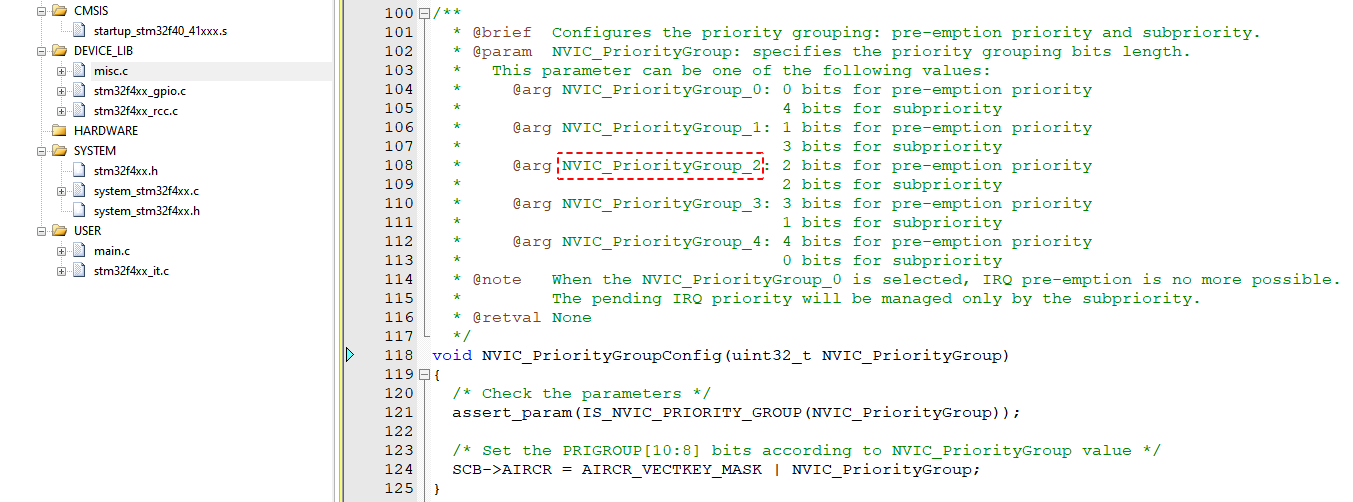
响应优先级（次优先级）：在同时发生多个中断的情况下，响应优先级高的先执行。

**思考：抢占优先级和响应优先级之间的区别？？？？？？**

1. 抢占优先级高的中断可以打断抢占优先级低的中断的执行
2. 抢占优先级一样高的中断，响应优先级高的中断不可以打断响应优先级低的中断
3. 抢占优先级一样高的中断，如果同时发生的情况下响应优先级高的先执行
4. 抢占优先级和响应优先级一样高的中断同时发生，则按照向量表中的优先级执行

为了方便用户管理和响应中断，NVIC提供一个函数接口可以对中断优先级进行分组





函数原型

void NVIC\_PriorityGroupConfig(uint32\_t NVIC\_PriorityGroup)

函数参数

参数一：NVIC\_PriorityGroup 打算设置的NVIC优先级分组 一般为NVIC\_PriorityGroup\_2

返回值 None

**注意：设置中断优先级分组应该在主程序运行的开头部分进行，并且不能随意修改分组，否则会出现中断管理混乱，导致程序出现未知问题。**

int main()

{

//设置优先级分组

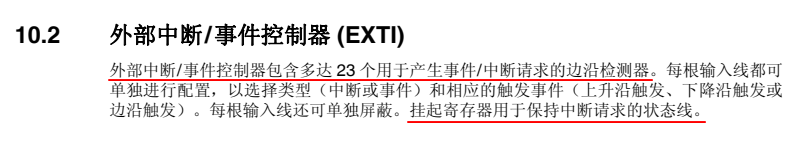
NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);

//对硬件进行初始化

.......

}

1. EXTI的概述

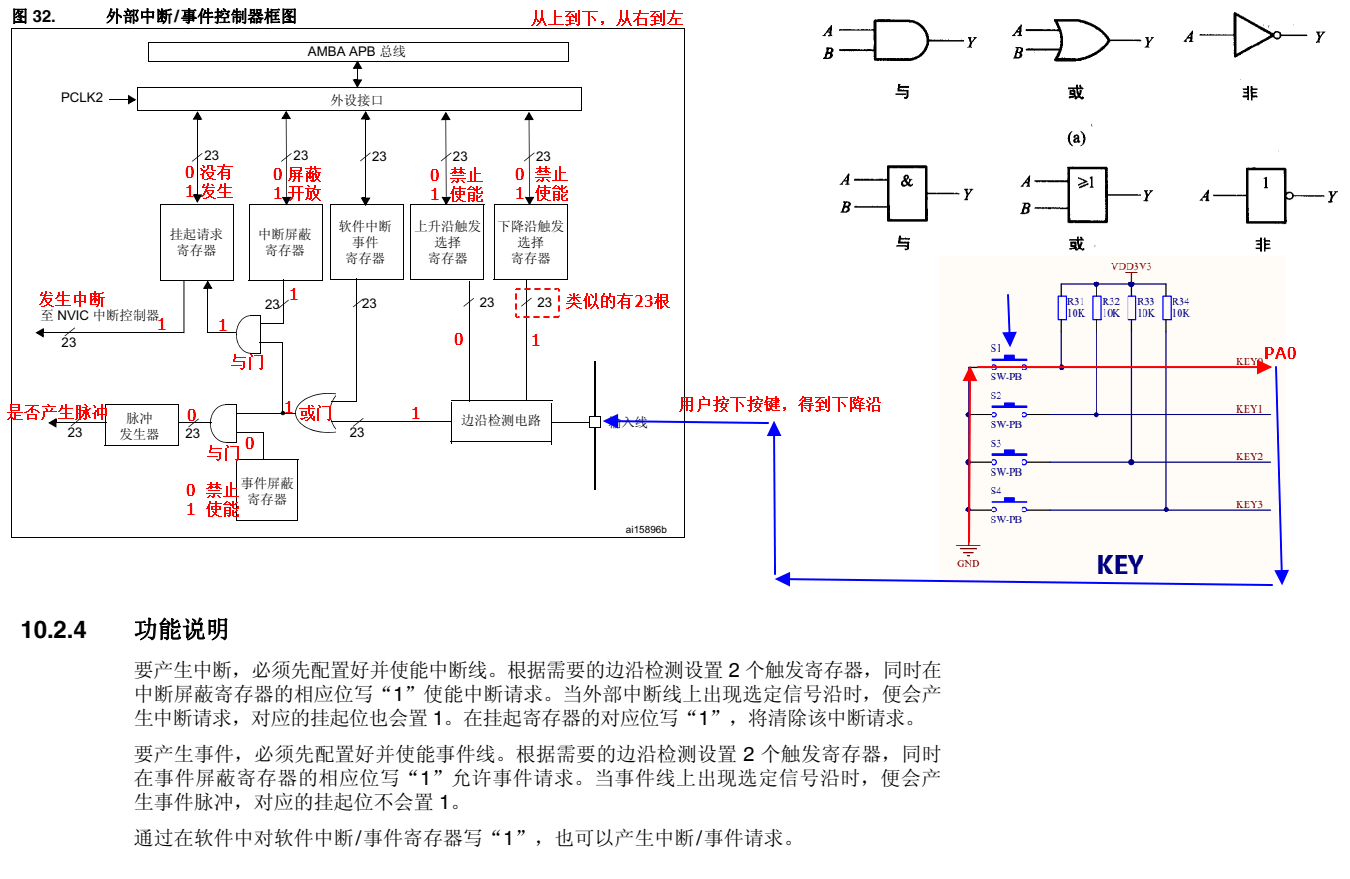


EXTI指的是外部中断/事件控制器，一共有23个，每个都有一个内部的边沿检测器，可以检测上升沿或者下降沿，每根线都可以产生事件或者中断。

上升沿：指的是电平信号由低变高的那一刻

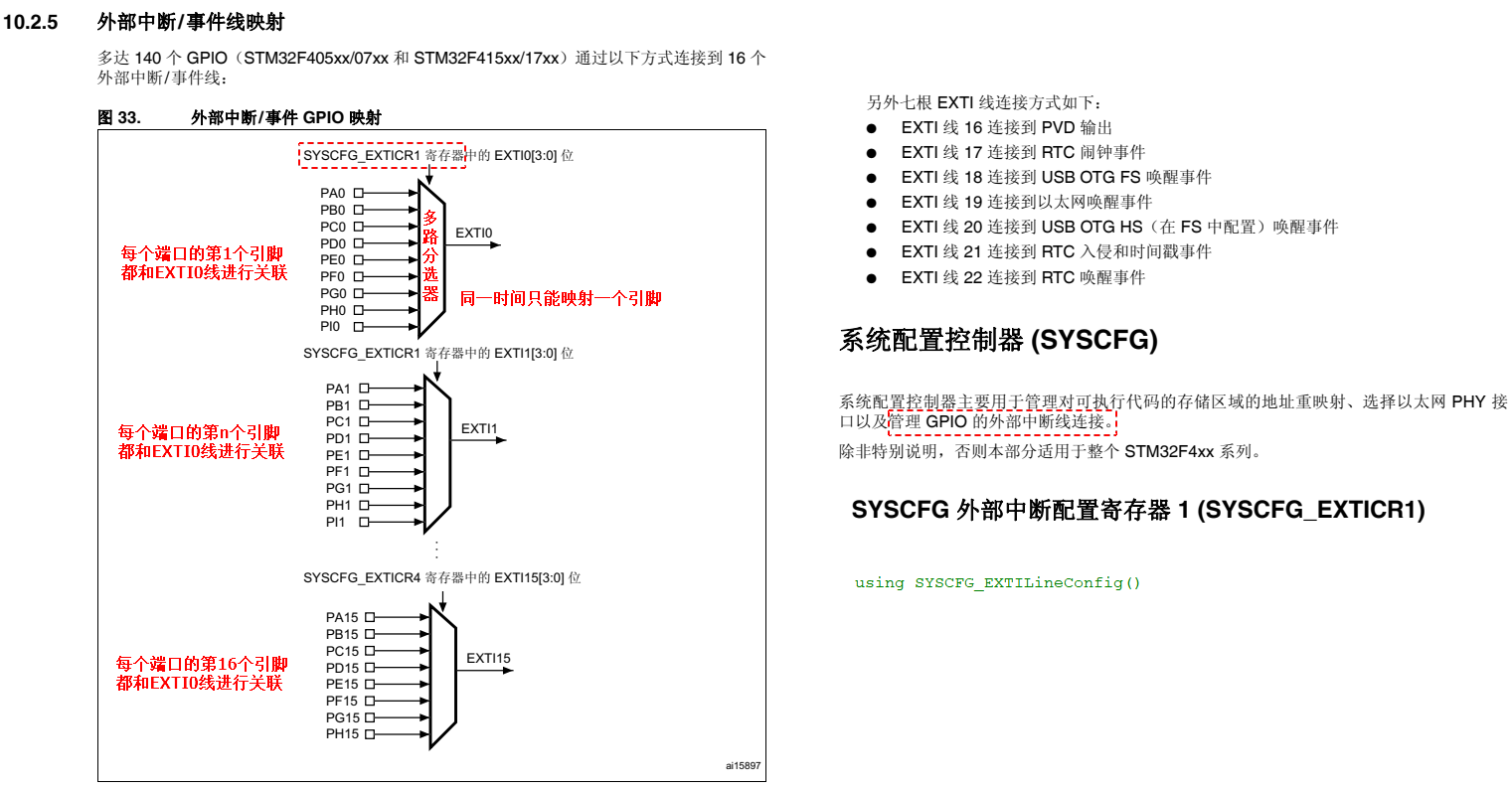
下降沿：指的是电平信号由高变低的那一刻

想要搞清楚EXTI的使用流程，必须要看懂EXTI的框图

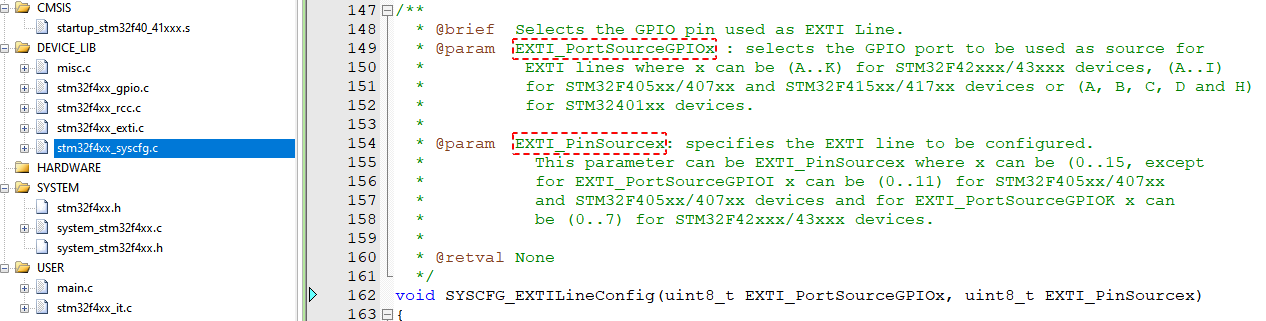


注意：每个GPIO引脚都可以配置为外部中断，但是和GPIO相关的外部中断线一共有16根，分别为EXTI0~EXTI15。

**思考：STM32F407系列有114个GPIO口，那如何和外部中断线进行关联？通过映射的方式**



如果想要让外部中断线和GPIO口进行映射，需要使用一个函数接口SYSCFG\_EXTILineConfig()



函数原型

void SYSCFG\_EXTILineConfig(uint8\_t EXTI\_PortSourceGPIOx, uint8\_t EXTI\_PinSourcex)

函数参数

参数一：EXTI\_PortSourceGPIOx 想要映射的GPIO端口 如 EXTI\_PortSourceGPIOA

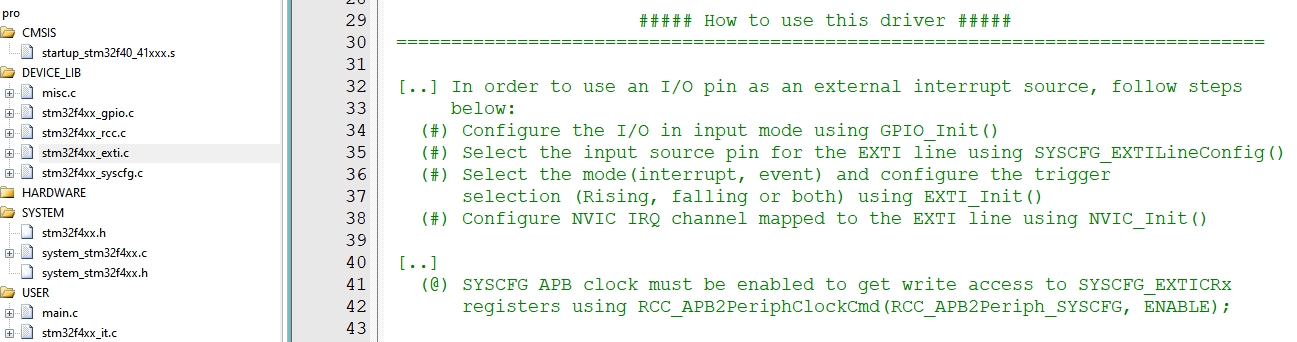
参数二：EXTI\_PinSourcex 想要映射的GPIO引脚 如 EXTI\_PinSource0

**注意：因为建立映射的时候需要使用系统配置控制器（SYSCFG），所以在编写代码的时候必须打开SYSCFG外设的时钟，挂载在APB2总线下 调用RCC\_APB2PeriphClockCmd()函数**

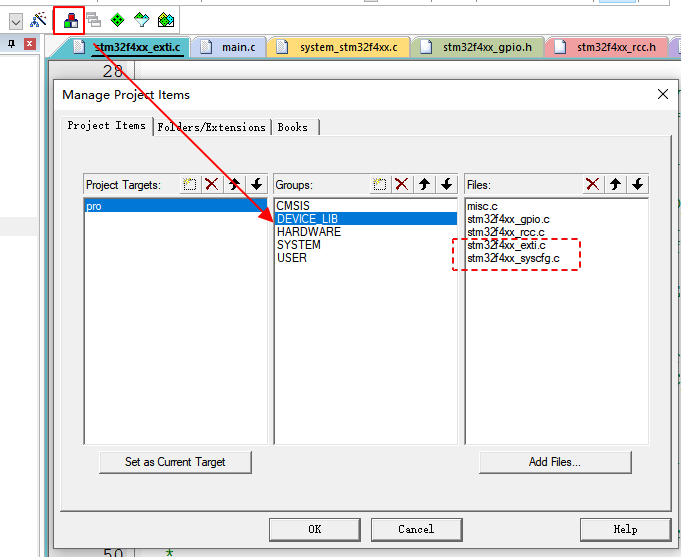
1. 外部中断的代码编写

**注意：所有端口都具有外部中断功能。要使用外部中断线，必须将端口配置为输入模式**

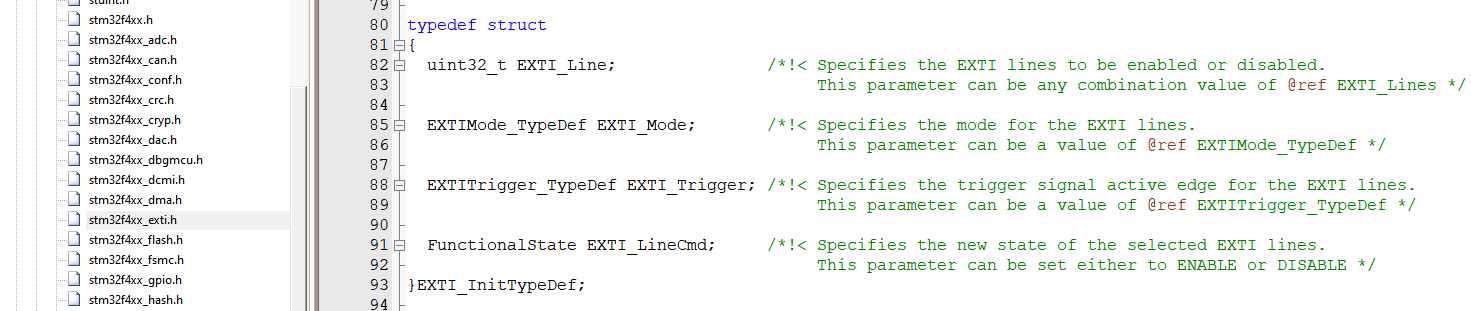
想要使用EXTI进行外部事件的检测，可以参考stm32f4xx\_exti.c源文件中的开头注释部分



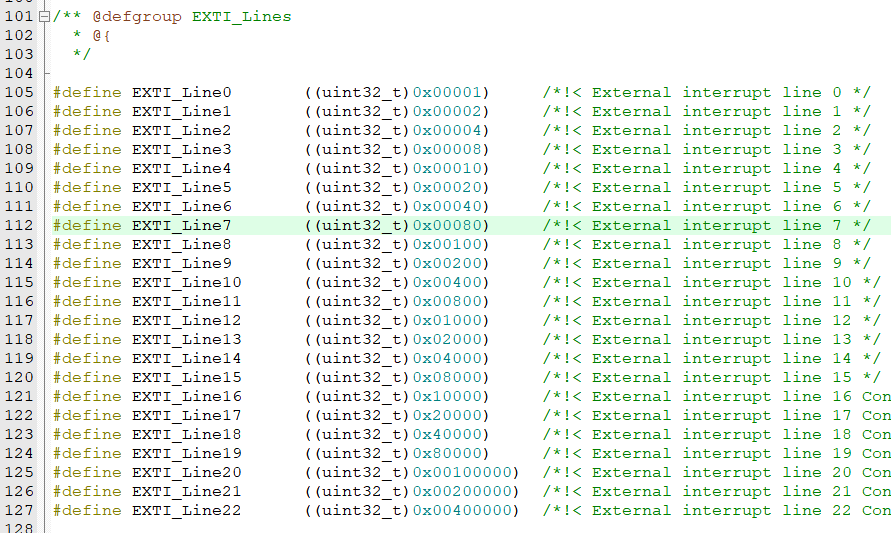
1. 应该在工程中添加stm32f4xx\_exti.c以及stm43f4xx\_syscfg.c两个源文件 步骤如下



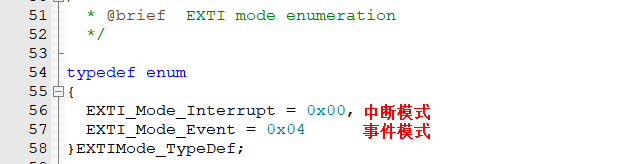
1. 打开GPIO外设时钟 + SYSCFG外设时钟 GPIO挂载AHB1总线 SYSCFG挂载在APB2总线
2. 定义GPIO初始化结构体，结构体配置的时候需要把引脚模式配置为输入模式 + 初始化
3. 调用SYSCFG\_EXTILineConfig()函数来对外部中断线和GPIO引脚建立映射关系
4. 定义EXTI初始化结构体，需要配置外部中断线（模式、触发方式.....） + 初始化



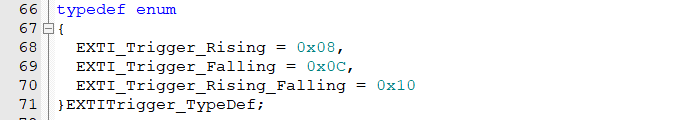
* EXTI\_Line 需要使用的外部中断线 参考EXTI\_Lines



* EXTI\_Mode 外部中断线的模式 （中断 or 事件） 参考EXTIMode\_TypeDef

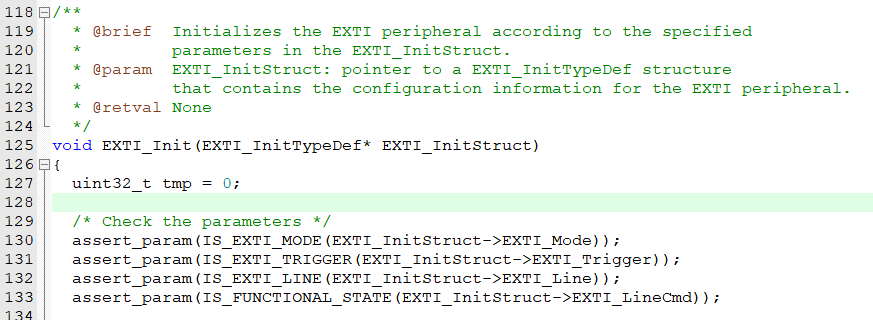


* EXTI\_Trigger 边沿检测方式（上升沿、下降沿、边沿） 参考EXTITrigger\_TypeDef

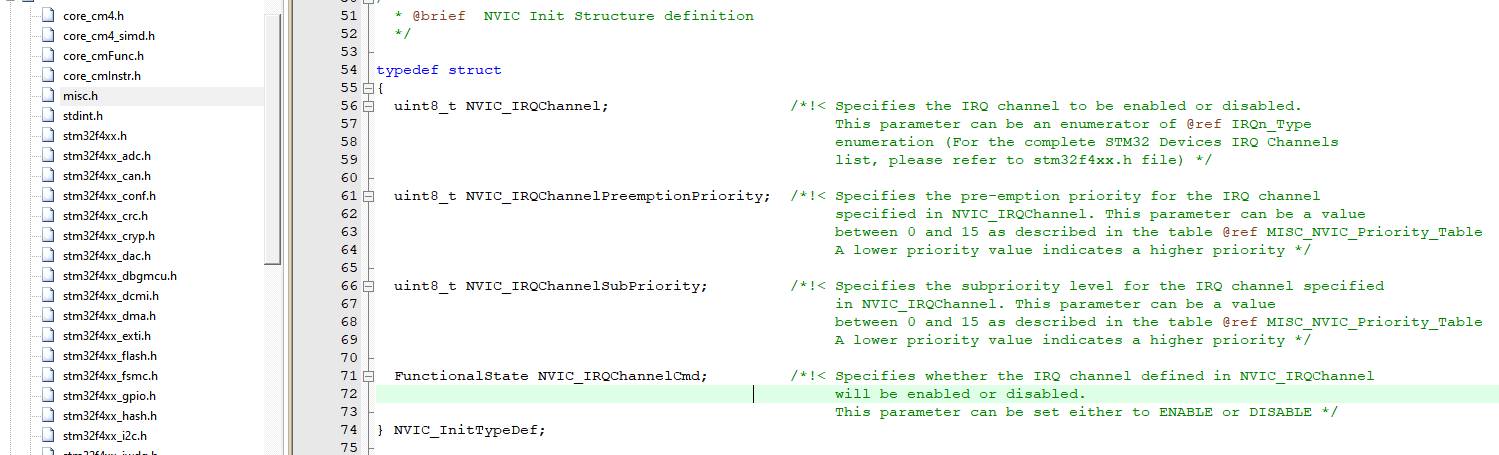


* EXTI\_LineCmd 外部中断线使能 ENABLE or DISABLE

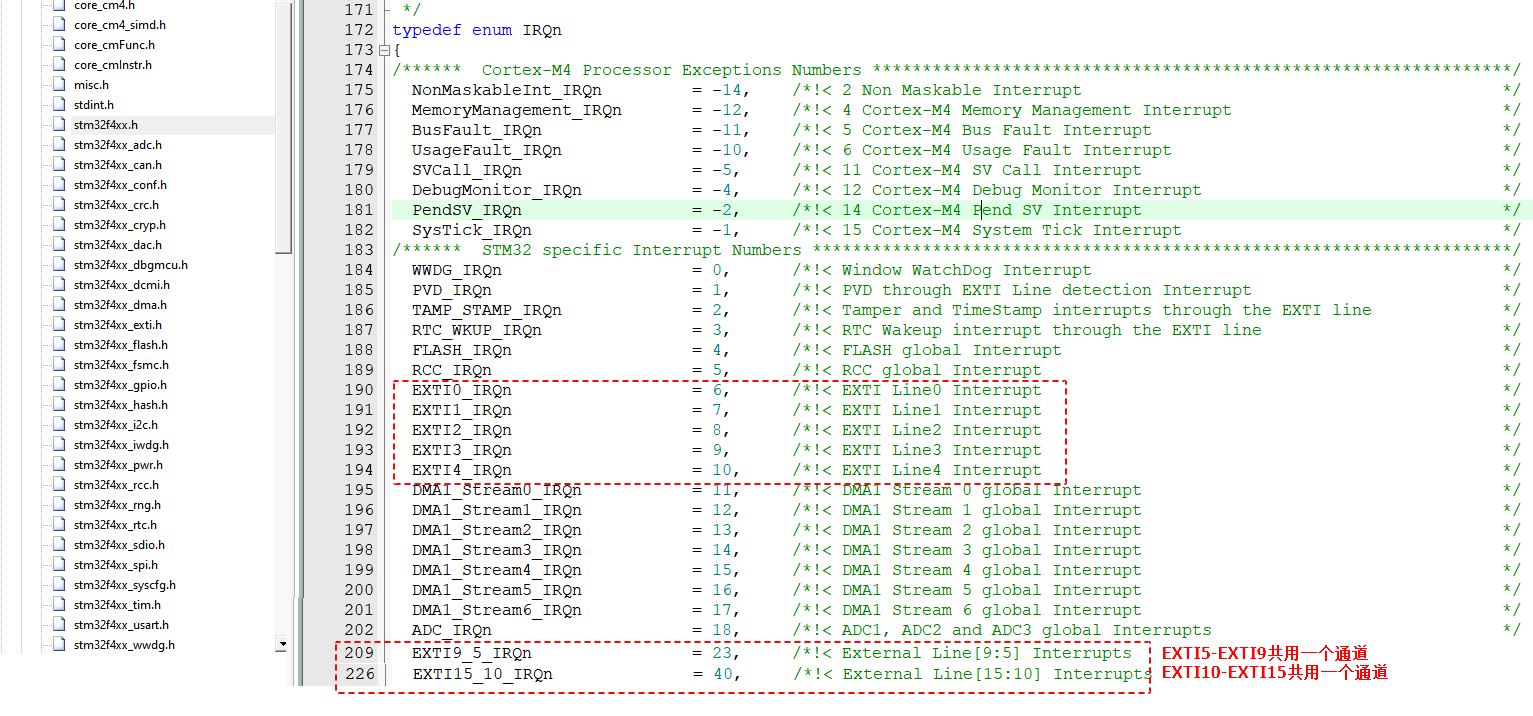
初始化EXTI



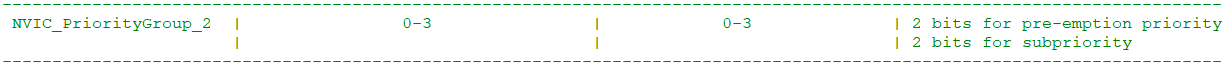
1. 定义NVIC的初始化结构体，并进行赋值（中断通道....） + 初始化



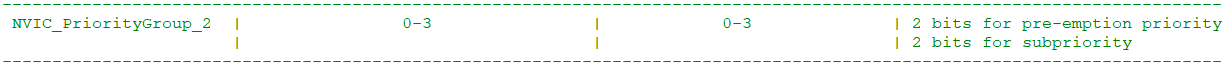
* NVIC\_IRQChannel 需要打开的中断通道 参考stm32f4xx.h



* NVIC\_IRQChannelPreemptionPriority 抢占优先级 需要根据分组进行填写

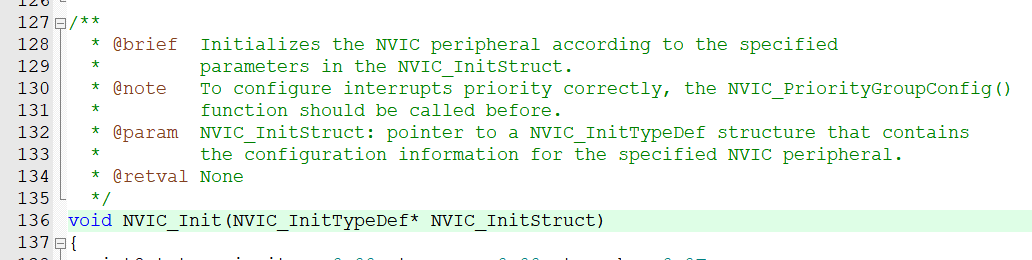


* NVIC\_IRQChannelSubPriority 响应优先级 需要根据分组进行填写



* NVIC\_IRQChannelCmd 中断通道的使能 ENABLE or DISABLE

初始化NVIC



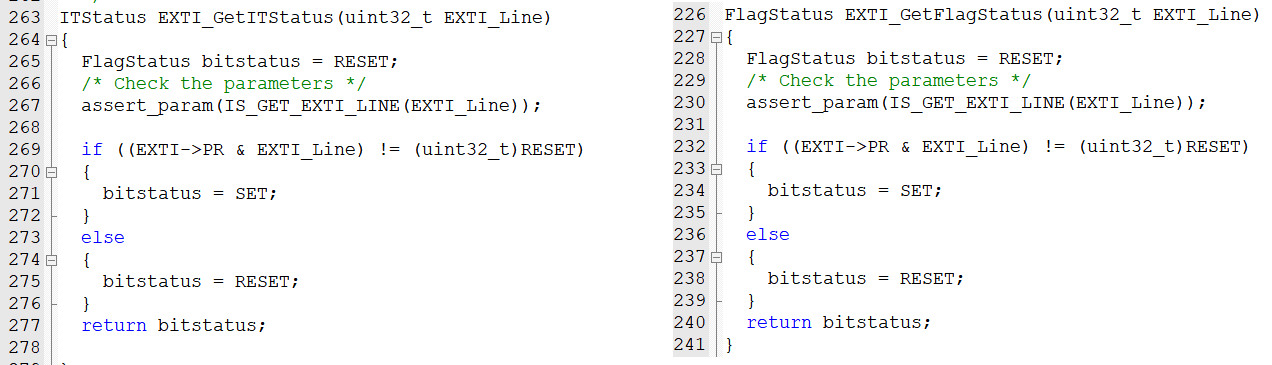
1. 编写中断服务函数 中断服务函数的名字必须从启动文件中进行拷贝

注意：中断服务函数的格式是固定的（没有返回值、没有参数） 名字也是固定的

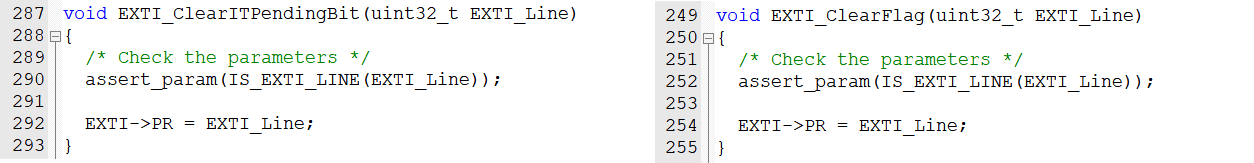
void EXTI0\_IRQHandler(void)

{

//检测是否发生了中断 if( EXTI\_GetITStatus(EXTI\_Line0) != RESET )



//清除中断标志 EXTI\_ClearITPendingBit(EXTI\_Line0)



}

注意：中断服务函数是不需要手动调用的，并且不要再中断中添加过长的延时时间，会导致系统的响应能力降低，所以如果打算处理比较复杂的事件，可以在中断中定义一个标志位，然后回到主程序中进行处理。

**作业：根据所学内容，编写代码，要求把四个按键都通过外部中断实现按键检测，当按下按键后控制LED进行亮灭**

预习：STM32F4中文参考手册的第14-17章节 关于定时器