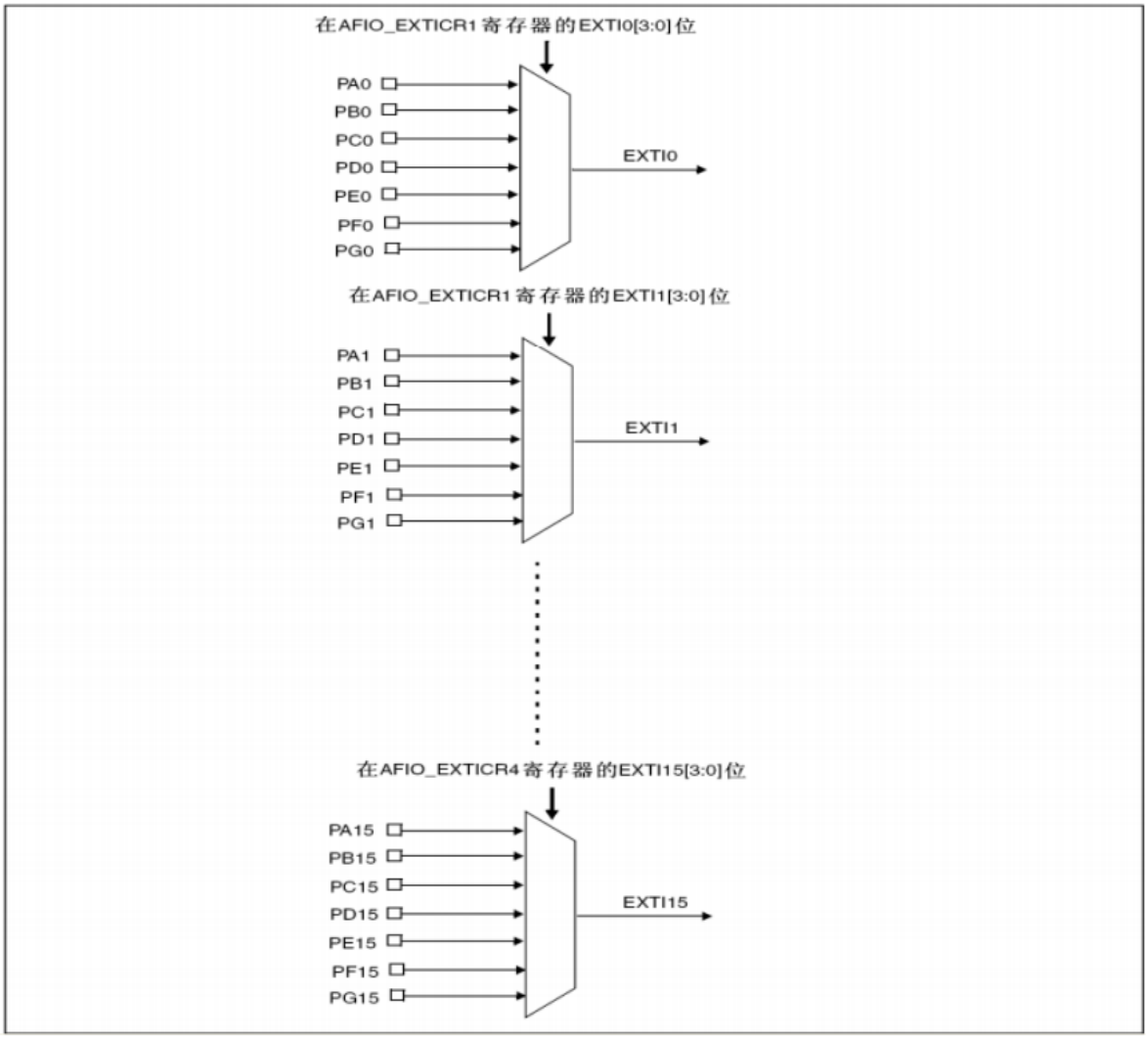
STM32 共提供了 16 根中断线，但外部引脚数量远远大于中断线的数量，ST 公司的设计如下图：



可以看到 PA0，PB0，……PG0 都对应第一根中断线，由图所示，所有引脚都有自己对应的中断线。但是每个中断线只能选择他的候选项中的一条，这样不固定引脚，大大提升了使用的自由。

GPIO_EXTILineConfig(GPIO_PortSourceGPIOA,GPIO_PinSource3)；　//将中断 3 和 GPIOA 连接。

GPIO_EXTILineConfig 函数负责将你选用的引脚和他的中断线绑定起来。上例即使用 3 号中断，GPIOA，即 PA3。

在使用外部中断之前，记得开启引脚复用时钟：

RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE)；　　//使能管脚复用时钟

同样要初始化中断使用的 GPIO，初始化为上拉输入状态。

中断的配置和 GPIO 的配置一样，需要设定自己的配置信息结构体。(该图引脚号与原理图不同，请对照原理图使用)

```
//中断线和中断初始化配置
EXTI_InitStructure.EXTI_Line = EXTI_Line3;              //因为引脚是GPIO3，使用3号中断线
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;     //配置模式为中断
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling; //下降沿触发中断
EXTI_InitStructure.EXTI_LineCmd = ENABLE;               //使能中断配置
EXTI_Init(&EXTI_InitStructure);                          //传入中断信息，初始化
```

从定义可以看出，有 4 个参数需要设置 。 第一个参数是中断线的标号，范围为 EXTI_Line0……EXTI_Line15。第二个参数是中断模式， 可选值为中断 EXTI_Mode_Interrupt 和事件 EXTI_Mode_Event。第三个参数是触发方式，可以是下降沿触发 EXTI_Trigger_Falling，上升沿触发 EXTI_Trigger_Rising，或者任意电平（上升沿和下降沿） 触发 EXTI_Trigger_Rising_Falling，相信学过 51 的对这个不难理解。 最后一个参数就是使能中断线了。

既然涉及到中断，就必须要考虑中断的优先级问题。

中断优先级的分配表如下

组	AIRCR[10：8]	bit[7：4]分配情况	分配结果
0	111	0：4	0 位抢占优先级，4 位响应优先级
1	110	1：3	1 位抢占优先级，3 位响应优先级
2	101	2：2	2 位抢占优先级，2 位响应优先级
3	100	3：1	3 位抢占优先级，1 位响应优先级
4	011	4：0	4 位抢占优先级，0 位响应优先级

在例程中我们采用 2 组分配方式，即 2：2。

```
NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
```

二进制共四位，对半分配则抢占优先级的可选范围为 0 到 3，相应优先级为 0 到

1）如果两个中断的抢占优先级和响应优先级都一样，则哪个中断先发生执行哪个。

2）高抢占优先级可以打断正在执行的低抢占优先级的中断，抢占优先级相同，高响应优先级的中断无法打断低响应优先级的中断。接下来我们配置中断的优先级。

```
//中断优先级配置
    NVIC_InitStructure.NVIC_IRQChannel=EXTI3_IRQn; //使能按键外部中断通道
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority=2;//抢占优先级为
2
    NVIC_InitStructure.NVIC_IRQChannelSubPriority=2; //子优先级为 2
    NVIC_InitStructure.NVIC_IRQChannelCmd=ENABLE; //使能配置
    NVIC_Init(&NVIC_InitStructure);  //传入配置信息，初始化
```

这里的外部中断通道其实就是使能该中断，中断号都在 stm32f10x.h 中定义。外部中断号如下：

EXTI0_IRQn

EXTI1_IRQn

EXTI2_IRQn
EXTI3_IRQn

EXTI4_IRQn
EXTI9_5_IRQn
EXTI15_10_IRQn
可以看到 5 号到 9 号中断共用一个，10 号到 15 号共用一个。因为中断号和中断函数是一一对应的。中断函数名可以在启动文件*.s 的汇编文件中找到。
中断函数即在中断发生后执行的函数。因为 15 到 10 号中断函数比较特殊，拿此举例：

```
void EXTI15_10_IRQHandler()
{

    if(EXTI_GetITStatus(EXTI_Line12) != RESET)
    {
        //添加你自己的代码
        EXTI_ClearITPendingBit(EXTI_Line12);
    }
}
```

该句判断 12 号中断是否发生

```
if(EXTI_GetITStatus(EXTI_Line12) != RESET);
```

当你在 10 号到 15 号设定了多个中断时，你就需要好几个判断语句来判定到底是哪个中断发生了。当执行完自己的代码以后，通过清除中断占用标志位来恢复中断的使能 EXTI_ClearITPendingBit(EXTI_Line12);该句清除 12 号中断的中断标志位。