**[STM32 GPIO外部中断](https://www.cnblogs.com/zhoubatuo/p/6134166.html)**

一、什么是中断  
 中断是单片机当中非常重要的一种机制，它可以提高单片机处理异步事件的实时响应速

度。中断，顾名思义就是打断正在执行的任务，去执行优先级更高的任务。

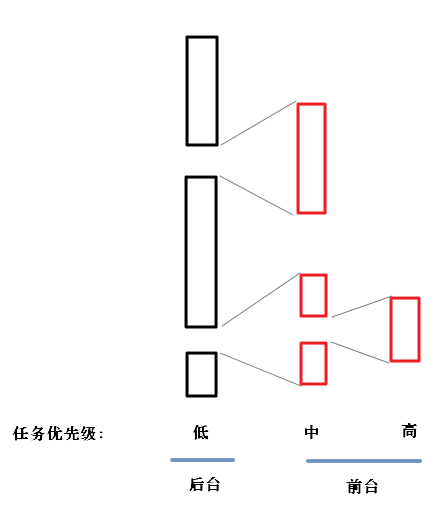
这里先介绍一个概念——前后台任务：

在main主程序中 while 死循环中的代码称为 "后台任务"，中断则称为 "前台任务"。

在大部分时间后台任务得以执行，只有在需要紧急处理的中断发生时才会切换到前台任务即

中断服务函数，前台任务执行完之后会自动切换到后台任务。另外也可以把后台任务叫做任

务级，把前台任务叫做中断级。 我们看一幅图加深理解：



图中黑色部分就是前面讲的后台任务，红色部分就是前台任务。另外可以看到，红色部

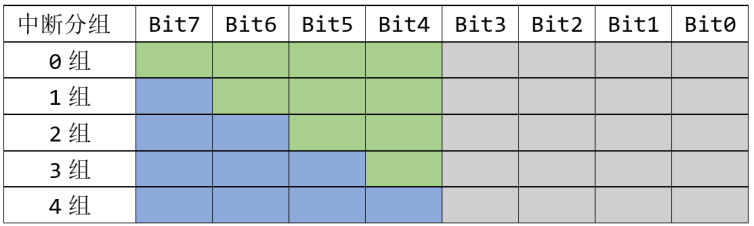
分又进行了一次嵌套，也就是说高优先级的中断同样也会打断低优先级的中断。

1. STM32中断优先级管理

STM32中断优先级是通过NVIC来管理的。  
 NVIC中文名：嵌套中断向量控制器。

1. STM32系列总共有84个中断，其中我用的STM32F103系列的板子有60个中断，可想而知STM32一定有一个中断的管理机制来管理这么多中断，使能中断并且配置相关中断的优先级，这就是NVIC的工作
2. 优先级高的中断可以打断优先级低的中断，这样一个中断执行的时候被另外中断给打断，这个中断执行完再执行上一个中断，形成中断之间的嵌套，或许这就是NVIC 名字为内嵌向量中断控制器来源
3. NVIC中对于每个中断需要设置 抢占优先级 和 响应优先级(又称子优先级)，很明显多个中断会先比较 抢占优先级，抢占优先级相同的比较响应优先级，这里值得注意的是 高抢占优先级能够打断低抢占优先级的，但是相同抢占优先级的高响应优先级不能打断低响应优先级，只能排队。
4. NVIC利用四个位来保存抢占优先级和响应优先级

中断优先级分组表





|  |  |  |  |
| --- | --- | --- | --- |
| 中断优先级分组 | 分配结果 | 抢占优先级取值 | 响应优先级取值 |
| 0 | 0位抢占优先级，4位响应优先级 | 0~0 | 0~15 |
| 1 | 1位抢占优先级，3位响应优先级 | 0~1 | 0~7 |
| 2 | 2位抢占优先级，2位响应优先级 | 0~3 | 0~3 |
| 3 | 3位抢占优先级，1位响应优先级 | 0~7 | 0~1 |
| 4 | 4位抢占优先级，0位响应优先级 | 0~15 | 0~0 |

优先级的数值越小，优先级越高。

一般在main主程序中设置中断优先级分组。系统默认分组为0组。

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_0);

三、STM32外部中断：

STM32 的每一个GPIO都能配置成一个外部中断触发源。

STM32 通过根据引脚的序号不同将众多中断触发源分成不同的组，比如：PA0，PB0，PC0，PD0，PE0，PF0，PG0为第一组，那么依此类推，我们能得出一共有16 组，STM32 规定，每一组中同时只能有一个中断触发源工作，那么，最多工作的也就是16个外部中断。STM32F103 的中断控制器支持 19 个外部中断/事件请求。每个中断设有状态位，每个中断/事件都有独立的触发和屏蔽设置。STM32F103 的19 个外部中断为：

线 0~15：对应外部 IO 口的输入中断。

|  |  |  |
| --- | --- | --- |
| GPIO引脚 | 中断标志位 | 中断处理函数名 |
| PA0~PG0 | EXTI0 | EXTI0\_IRQHandler |
| PA1~PG1 | EXTI1 | EXTI1\_IRQHandler |
| PA2~PG2 | EXTI2 | EXTI2\_IRQHandler |
| PA3~PG3 | EXTI3 | EXTI3\_IRQHandler |
| PA4~PG4 | EXTI4 | EXTI4\_IRQHandler |
| PA5~PG5 | EXTI5 | EXTI9\_5\_IRQHandler |
| PA6~PG6 | EXTI6 |  |
| PA7~PG7 | EXTI7 |  |
| PA8~PG8 | EXTI8 |  |
| PA9~PG9 | EXTI9 |  |
| PA10~PG10 | EXTI10 | EXTI15\_10\_IRQHandler |
| PA11~PG11 | EXTI11 |  |
| PA12~PG12 | EXTI12 |  |
| PA13~PG13 | EXTI13 |  |
| PA14~PG14 | EXTI14 |  |
| PA15~PG15 | EXTI15 |  |

EXTI16： PVD 输出。

EXTI17： RTC 闹钟事件。

EXTI18： USB 唤醒事件。

四：外部中断的配置过程：

　　1、配置触发源GPIO口：

　　因为GPIO口作为触发源使用，所以将GPIO口配置成输入模式，触发模式有以下几种：

　　a.GPIO\_Mode\_AIN ，模拟输入（ADC模拟输入，或者低功耗下省电）

　　b.GPIO\_Mode\_IN\_FLOATING ，浮空输入

　　c.GPIO\_Mode\_IPD ，带下拉输入

　　d.GPIO\_Mode\_IPU ，带上拉输入

初始化GPIO口，作为输入模式

void initKey(void)

{

　　GPIO\_InitTypeDef GPIO\_InitStructure;//定义结构体

　　RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB，ENABLE);//使能时钟

　　GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_9;//选择PB9

　　GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU;//设置成上拉输入

GPIO\_Init(GPIOB， &GPIO\_InitStructure);//使用结构体信息进行初始化IO口

}

　　2、中断线上进行中断初始化：

void initEXTI(void)

{

EXTI\_InitTypeDef EXTI\_InitStructure;//定义初始化结构体

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO, ENABLE); //开启外设复用时钟

GPIO\_EXTILineConfig(GPIO\_PortSourceGPIOB，GPIO\_PinSource9); //将GPIO口与中断线映射

　　EXTI\_InitStructure.EXTI\_Line=EXTI\_Line9; //中断线,取值范围为EXTI\_Line0~EXTI\_Line15

　　EXTI\_InitStructure.EXTI\_Mode = EXTI\_Mode\_Interrupt;//中断模式，可选值为中断 EXTI\_Mode\_Interrupt 和事件 EXTI\_Mode\_Event。

　　EXTI\_InitStructure.EXTI\_Trigger = EXTI\_Trigger\_Falling;//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

　　EXTI\_InitStructure.EXTI\_LineCmd = ENABLE; //开启中断

　　EXTI\_Init(&EXTI\_InitStructure);//根据结构体信息进行初始化

 }

　　3、外部中断服务函数的编写：

　　外部中断函数分别为：

　　EXPORT   EXTI0\_IRQHandler

　　EXPORT   EXTI1\_IRQHandler

　　EXPOR T   EXTI2\_IRQHandler

　　EXPORT   EXTI3\_IRQHandler

　　EXPORT   EXTI4\_IRQHandler

　　EXPORT   EXTI9\_5\_IRQHandler

　　EXPORT   EXTI15\_10\_IRQHandler

　　中断线 0-4 每个中断线对应一个中断函数，中断线 5-9 共用中断函数 EXTI9\_5\_IRQHandler，中断线 10-15 共用中断函数 EXTI15\_10\_IRQHandler。

　　void EXTI9\_5\_IRQHandler(void)

{

　　　　if(EXTI\_GetITStatus(EXTI\_Line9)!=RESET)//判断某个线上的中断是否发生

　　　　{

EXTI\_ClearITPendingBit(EXTI\_Line9); //清除 LINE 上的中断标志位，否则不断重复发生中断。

B\_LED\_OFF;

Delay\_ms(time);

B\_LED\_ON;

Delay\_ms(time);

B\_LED\_OFF;

Delay\_ms(time);

B\_LED\_ON;

Delay\_ms(time);

B\_LED\_OFF;

Delay\_ms(time);

B\_LED\_ON;

Delay\_ms(time);

B\_LED\_OFF;

Delay\_ms(time);

B\_LED\_ON;

Delay\_ms(time);

B\_LED\_OFF;

Delay\_ms(time);

B\_LED\_ON;

Delay\_ms(time);

　　　　}

　　}

　　4、中断优先级配置：

void setNVIC(void)

{

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2); //设置中断分组

　　NVIC\_InitTypeDef NVIC\_InitStructure;//定义结构体

　　NVIC\_InitStructure.NVIC\_IRQChannel = EXTI9\_5\_IRQn; //中断号

　　NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0x02; //抢占优先级 2，

　　NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 0x02; //响应优先级 2

　　NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE; //使能外部中断通道

NVIC\_Init(&NVIC\_InitStructure); //根据结构体信息进行优先级初始化

}