[**STM32 中断**](https://www.cnblogs.com/zhoubatuo/p/6134166.html)**优先级实验**

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 |

1. 实验情景：

 在“红绿黄流水灯”基础上，按“按键1”，蓝色LED立即闪亮5次，每次闪亮为亮500ms,灭500ms。按“按键2”，蓝色LED立即闪亮5次，每次闪亮为亮100ms,灭1000ms。

当按“按键1”后，立即按“按键2”，中断“按键1”程序执行，马上执行“按键2”的程序，执行完“按键2”，再继续执行“按键1”程序。

1. 实验情景分析

 “红绿黄流水灯”是后台程序，按下“按键1”，触发中断，打断后台程序执行，去执行“按键1”的中断处理程序，当执行完中断处理程序，继续执行后台程序。

 当按下“按键2”时，立即中断“按键1”的中断处理程序，说明 "按键2”的**抢占优先级**比“按键1”高。

1. 电路连接

 PB9——“按键1”—— “接地”

 PA0——“按键2”—— “接地”

1. 代码编写

/\*

\* 功能: 初始化PB9,PA0为上拉输入模式

\*/

void initKey(void)

{

 GPIO\_InitTypeDef GPIO\_InitStructure;

/\*\*\*\*\*\*\*\*\*\*\*\*\*初始化PB9为上拉输入模式\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

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

 GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU; //上拉输入模式

 GPIO\_Init(GPIOB, &GPIO\_InitStructure); //初始化PB9

/\*\*\*\*\*\*\*\*\*\*\*\*\*初始化PA0为上拉输入模式\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE); //使能 GPIOA 时钟

 GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0; //PA0

 GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU; //上拉输入模式

 GPIO\_Init(GPIOA, &GPIO\_InitStructure); //初始化PA0

}

/\*\*

 \* 功能：初始化外部中断0,9

 \* 参数：None

 \* 返回值：None

 \*/

void initEXTI(void)

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*初始化PB9作为EXTI9\_Line中断线\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 EXTI\_InitTypeDef EXTI\_InitStructure; //定义外部中断初始化结构体

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

 GPIO\_EXTILineConfig(GPIO\_PortSourceGPIOB, GPIO\_PinSource9); //使用PB9作为EXTI9信号输入源

 EXTI\_InitStructure.EXTI\_Line = EXTI\_Line9; //设置外部中断线9

 EXTI\_InitStructure.EXTI\_Mode = EXTI\_Mode\_Interrupt; //外部中断模式

 EXTI\_InitStructure.EXTI\_Trigger = EXTI\_Trigger\_Falling; //下降沿触发中断

 EXTI\_InitStructure.EXTI\_LineCmd = ENABLE; //使能外部中断屏蔽寄存器

 EXTI\_Init(&EXTI\_InitStructure); //设置生效

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*初始化PA0作为EXTI0\_Line中断线\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 GPIO\_EXTILineConfig(GPIO\_PortSourceGPIOA, GPIO\_PinSource0); //使用PA0作为EXTI0信号输入源

 EXTI\_InitStructure.EXTI\_Line = EXTI\_Line0; //设置外部中断线0

 EXTI\_InitStructure.EXTI\_Mode = EXTI\_Mode\_Interrupt; //外部中断模式

 EXTI\_InitStructure.EXTI\_Trigger = EXTI\_Trigger\_Falling; //下降沿触发中断

 EXTI\_InitStructure.EXTI\_LineCmd = ENABLE; //使能外部中断屏蔽寄存器

 EXTI\_Init(&EXTI\_InitStructure); //设置生效

}

/\*\*

 \* 功能：外部中断9\_5中断服务函数

 \*/

void EXTI9\_5\_IRQHandler(void)

{

 int time = 500;

 if(EXTI\_GetITStatus(EXTI\_Line9) == SET) //EXTI\_Line9中断已发生

 {

 EXTI\_ClearITPendingBit(EXTI\_Line9); //清除 EXTI\_Line9上的中断标志位

 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);

 }

}

/\*\*

 \* 功能：外部中断0中断服务函数

 \*/

void EXTI0\_IRQHandler(void)

{

 int time=1000;

 int time1=100;

 if(EXTI\_GetITStatus(EXTI\_Line0) == SET) //EXTI\_Line9中断已发生

 {

 EXTI\_ClearITPendingBit(EXTI\_Line0); //清除 EXTI\_Line9上的中断标志位

 B\_LED\_OFF;

 Delay\_ms(time);

 B\_LED\_ON;

 Delay\_ms(time1);

 B\_LED\_OFF;

 Delay\_ms(time);

 B\_LED\_ON;

 Delay\_ms(time1);

 B\_LED\_OFF;

 Delay\_ms(time);

 B\_LED\_ON;

 Delay\_ms(time1);

 B\_LED\_OFF;

 Delay\_ms(time);

 B\_LED\_ON;

 Delay\_ms(time1);

 B\_LED\_OFF;

 Delay\_ms(time);

 B\_LED\_ON;

 Delay\_ms(time1);

 }

}

/\*\*

\*

\*功能: 设置中断抢占优先级

\*

\*/

void setNVIC(void)

{

//设置中断优先级分组

 NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);

/\*\*\*\*\*\*\*\*\*\*设置EXTI9\_5中断优先级\*\*\*\*\*\*\*\*\*\*\*\*\*/

 NVIC\_InitTypeDef NVIC\_InitStructure; //定义NVIC初始化结构体

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

 NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 2; //设置抢占优先级

 NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 2; //设置子优先级

 NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE; //开启中断

 NVIC\_Init(&NVIC\_InitStructure); //设置生效

/\*\*\*\*\*\*\*\*\*\*设置EXTI0中断优先级\*\*\*\*\*\*\*\*\*\*\*\*\*/

 NVIC\_InitStructure.NVIC\_IRQChannel = EXTI0\_IRQn; //设置中断号

 NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 1; //设置抢占优先级

 NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 1; //设置子优先级

 NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE; //开启中断

 NVIC\_Init(&NVIC\_InitStructure); //设置生效

}

注意：如果缺少中断服务程序函数EXTI0\_IRQHandler或EXTI9\_5\_IRQHandler，则按下按键2或按键1时，系统将会出现死机状态。