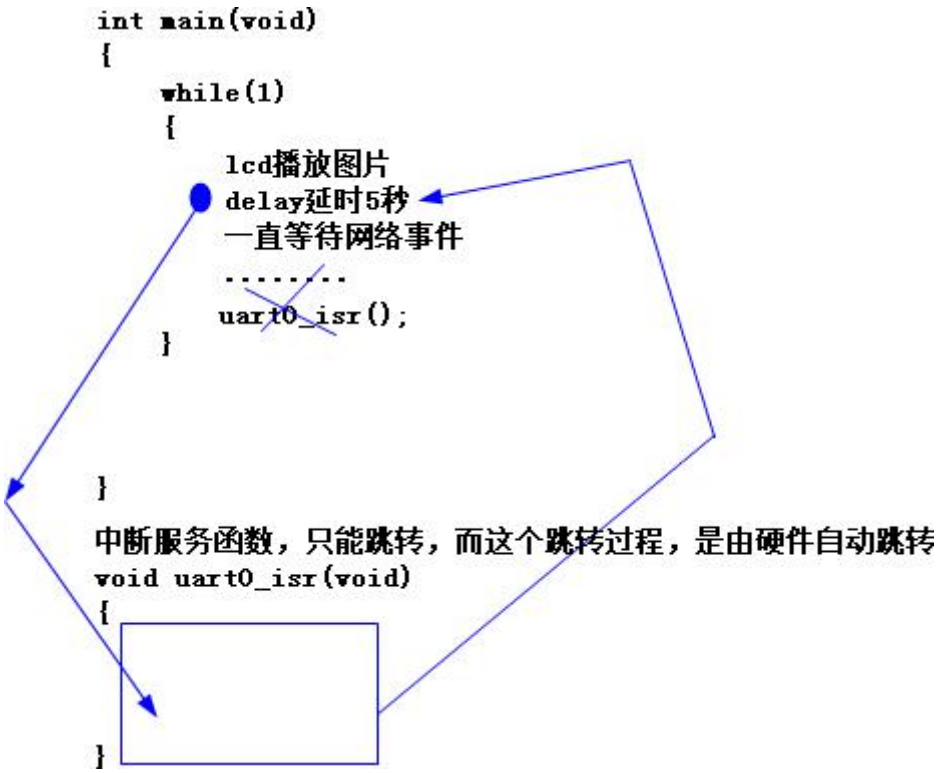
一、ARM Cortex-M4 的中断体系

1、定义

中断，意味着中途打断现在干的事情，要处理紧急的事件。

现实的例子：玩王者农药的时候，女朋友突然来电话。在编程当中还常遇到实时接收数据的请求，都使用中断服务函数，示例如下：



二、嵌套向量中断控制寄存器

预习：STM32F4xx 中文参考手册.pdf P233~P243

1、NVIC 特性

无论是 ARM Cortex M0/M3/M4 还是 ARM Cortex-A8/A53/A72/A73 等等内核，都有 NVIC。

STM32F405xx/07xx 和 STM32F415xx/17xx 具有 **82 个可屏蔽**（能够通过代码进行开和关中断）中断通道，**10 个不可屏蔽（无法通过代码关闭该中断）的中断**，16 个可编程优先级。

向量意味就是中断源。
向量表，也就是中断源表。

2、外部中断/事件控制器 (EXTI)

多达 140 个 GPIO（STM32F405xx/07xx 和 STM32F415xx/17xx）通过以下方式连接到 16 个外部中断/事件线：
另外七根 EXTI 线连接方式如下：

- EXTI 线 16 连接到 PVD 输出
- EXTI 线 17 连接到 RTC 闹钟事件
- EXTI 线 18 连接到 USB OTG FS 唤醒事件

- EXTI 线 19 连接到以太网唤醒事件
- EXTI 线 20 连接到 USB OTG HS（在 FS 中配置）唤醒事件
- EXTI 线 21 连接到 RTC 入侵和时间戳事件
- EXTI 线 22 连接到 RTC 唤醒事件

3.库函数

a.选择对应的 GPIO 引脚连接到相应的中断控制线

```
/**
  * @brief  Selects the GPIO pin used as EXTI Line.
  * @param  EXTI_PortSourceGPIOx : selects the GPIO port to be used as source for
  *                EXTI lines where x can be (A..K) for STM32F42xxx/43xxx devices, (A..I)
  *                for STM32F405xx/407xx and STM32F415xx/417xx devices or (A, B, C, D and H)
  *                for STM32401xx devices.
  *
  * @param  EXTI_PinSourcex: specifies the EXTI line to be configured.
  *                This parameter can be EXTI_PinSourcex where x can be (0..15, except
  *                for EXTI_PortSourceGPIOI x can be (0..11) for STM32F405xx/407xx
  *                and STM32F405xx/407xx devices and for EXTI_PortSourceGPIOK x can
  *                be (0..7) for STM32F42xxx/43xxx devices.
  *
  * @retval None
  */
void SYSCFG_EXTILineConfig(uint8_t EXTI_PortSourceGPIOx, uint8_t EXTI_PinSourcex)
```

b.根据 EXTI_InitTypeDef 结构体进行外部中断控制线 0 初始化

```
/**
  * @brief  Initializes the EXTI peripheral according to the specified
  *             parameters in the EXTI_InitStruct.
  * @param  EXTI_InitStruct: pointer to a EXTI_InitTypeDef structure
  *             that contains the configuration information for the EXTI peripheral.
  * @retval None
  */
void EXTI_Init(EXTI_InitTypeDef* EXTI_InitStruct)
```

c.根据 NVIC_InitTypeDef 结构体对中断向量进行配置

```
/**
  * @brief  Initializes the NVIC peripheral according to the specified
  *             parameters in the NVIC_InitStruct.
  * @note   To configure interrupts priority correctly, the NVIC_PriorityGroupConfig()  //执行 NVIC_Init 前，必须调用 NVIC_PriorityGroupConfig
  *             function should be called before.
  * @param  NVIC_InitStruct: pointer to a NVIC_InitTypeDef structure that contains
  *             the configuration information for the specified NVIC peripheral.
  * @retval None
  */
void NVIC_Init(NVIC_InitTypeDef* NVIC_InitStruct)
```

4、中断优先级

中断优先级的一个意义：出现多个中断同时触发，但是不能同时处理，所以先后顺序之分，要根据实际上的运行环境优先处理重要的中断。

a.概述

STM32 对中断进行分组，共 5 组，组 0~4。同时，对每个中断设置一个抢占优先级和一个响应优先级级。

函数原型如下：

```
/**
  * @brief  Configures the priority grouping: pre-emption priority and subpriority.
  * @param  NVIC_PriorityGroup: specifies the priority grouping bits length.
  *   This parameter can be one of the following values:
  *     @arg NVIC_PriorityGroup_0: 0 bits for pre-emption priority        //没有抢占优先级
  *                                4 bits for subpriority                 //4 位设置响应优先级
  *     @arg NVIC_PriorityGroup_1: 1 bits for pre-emption priority        //1 位抢占优先级，能设置 2 个中断抢占优先级
  *                                3 bits for subpriority                 //3 位设置响应优先级
  *     @arg NVIC_PriorityGroup_2: 2 bits for pre-emption priority        //2 位抢占优先级，能设置 4 个中断抢占优先级
  *                                2 bits for subpriority                 //2 位设置响应优先级
  *     @arg NVIC_PriorityGroup_3: 3 bits for pre-emption priority        //3 位抢占优先级，能设置 8 个中断抢占优先级
  *                                1 bits for subpriority                 //1 位设置响应优先级
  *     @arg NVIC_PriorityGroup_4: 4 bits for pre-emption priority        //4 位抢占优先级，能设置 16 个中断抢占优先级
  *                                0 bits for subpriority                 //没有响应优先级
  * @note   When the NVIC_PriorityGroup_0 is selected, IRQ pre-emption is no more possible.
  *         The pending IRQ priority will be managed only by the subpriority.
  * @retval None
  */
void NVIC_PriorityGroupConfig(uint32_t NVIC_PriorityGroup)
```

只要开机初始化一次就可以了。

b.抢占优先级与响应优先级区别

1)高抢占优先级是可以打断正在进行的低抢占优先级的中断。
2)抢占优先级相同的中断，高响应优先级不可以打断低响应优先级的中断。
3)抢占优先级相同的中断，当两个中断同时发生的情况下，哪个响应优先级高，哪个先执行。
4)抢占优先级相同且响应优先级相同的中断，假如同时发生，会按照硬件内部固定的优先级执行，如下图。
5)无论是抢占优先级还是响应优先级，优先级数值越小，就代表优先级越高。

典型生活例子：

抢占优先级是银行劫匪，响应优先级是 VIP 客户，

1：两拨劫匪同时抢劫一家银行，这就看实例了，你抢占优先级高，你老大，你先抢。

2：VIP 客户和普通客户，大家都是良好公民，虽然你有钱，那也得等当前正在窗口办理的普通客户办完，VIP 才能开始办，

3：大家都是良好公民，抢占优先级相同，一起进入银行，好吧，VIP 先被接待

4：都是良好公民，都是普通客户，这就看谁在银行有人了，

5：抢占优先级是道德素质，越高越是普通公民，土匪的道德素质都比较低，响应优先级也是道德素质，当今社会，老实人可能都没啥钱~~~

位置	优先级	优先级类型	名称	说明	地址
	-	-	-	保留	0x0000 0000
	-3	固定	Reset	复位	0x0000 0004
	-2	固定	NMI	不可屏蔽中断。RCC 时钟安全系统(CSS) 连接到 NMI 向量。	0x0000 0008
	-1	固定	HardFault	所有类型的错误	0x0000 000C
	0	可设置	MemManage	存储器管理	0x0000 0010
	1	可设置	BusFault	预取指失败，存储器访问失败	0x0000 0014
	2	可设置	UsageFault	未定义的指令或非法状态	0x0000 0018
	-	-	-	保留	0x0000 001C - 0x0000 002B
	3	可设置	SVCall	通过 SWI 指令调用的系统服务	0x0000 002C
	4	可设置	Debug Monitor	调试监控器	0x0000 0030
	-	-	-	保留	0x0000 0034
	5	可设置	PendSV	可挂起的系统服务	0x0000 0038
	6	可设置	SysTick	系统嘀嗒定时器	0x0000 003C
0	7	可设置	WWDG	窗口看门狗中断	0x0000 0040
1	8	可设置	PVD	连接到 EXTI 线的可编程电压检测(PVD) 中断	0x0000 0044
2	9	可设置	TAMP_STAMP	连接到 EXTI 线的入侵和时间戳中断	0x0000 0048
3	10	可设置	RTC_WKUP	连接到 EXTI 线的 RTC 唤醒中断	0x0000 004C
4	11	可设置	FLASH	Flash 全局中断	0x0000 0050
5	12	可设置	RCC	RCC 全局中断	0x0000 0054
6	13	可设置	EXTI0	EXTI 线 0 中断	0x0000 0058
7	14	可设置	EXTI1	EXTI 线 1 中断	0x0000 005C
8	15	可设置	EXTI2	EXTI 线 2 中断	0x0000 0060
9	16	可设置	EXTI3	EXTI 线 3 中断	0x0000 0064
10	17	可设置	EXTI4	EXTI 线 4 中断	0x0000 0068
11	18	可设置	DMA1_Stream0	DMA1 流 0 全局中断	0x0000 006C
12	19	可设置	DMA1_Stream1	DMA1 流 1 全局中断	0x0000 0070
13	20	可设置	DMA1_Stream2	DMA1 流 2 全局中断	0x0000 0074
14	21	可设置	DMA1_Stream3	DMA1 流 3 全局中断	0x0000 0078
15	22	可设置	DMA1_Stream4	DMA1 流 4 全局中断	0x0000 007C
16	23	可设置	DMA1_Stream5	DMA1 流 5 全局中断	0x0000 0080
17	24	可设置	DMA1_Stream6	DMA1 流 6 全局中断	0x0000 0084
18	25	可设置	ADC	ADC1、ADC2 和 ADC3 全局中断	0x0000 0088

如果两个中断的抢占优先级和响应优先级都是一样的话，则看哪个中断先发生就先执行；

例子 1：中断抢占演示

```
抢占优先级2
响应优先级3
void EXTI3_IRQHandler(void)
{
        //检查是否有触发中断
    if(EXTI_GetITStatus(EXTI_Line3) == SET)
    {
        //添加用户代码
        PEout(13)=0;
        delay();
        PEout(13)=1;
        delay();

        /* 清空标志位，告诉CPU我已经处理完中断请求，可以触发下一次中断 */
        EXTI_ClearITPendingBit(EXTI_Line3);
    }
}
```

当产生抢占，会看到有盏LED灯同时亮。

```
抢占优先级3
响应优先级3
void EXTI4_IRQHandler(void)
{
        //检查是否有触发中断
    if(EXTI_GetITStatus(EXTI_Line4) == SET)
    {
        //添加用户代码
        PEout(14)=0;
        delay();
        PEout(14)=1;
        delay();

        /* 清空标志位，告诉CPU我已经处理完中断请求，可以触发下一次中断 */
        EXTI_ClearITPendingBit(EXTI_Line4);
    }
}
```

当KEY3按下触发中断的时候，这个时候再按下KEY2，发现能够抢夺CPU的使用权，会优先执行EXIT3_IRQHanlder这个函数，执行完之后，再执行EXIT4_IRQHanlder.
同时也能观察到PE14引脚连接的LED灯点亮的时间更长。

编写中断服务函数
1）高效的完成，简单。
2）不要添加各种大延时，会导致其他中断的延迟。

例子 2：中断不抢占演示

如果抢占优先级一样，不允许抢占。

```
抢占优先级3
响应优先级3
void EXTI3_IRQHandler(void)
{
        //检查是否有触发中断
    if(EXTI_GetITStatus(EXTI_Line3) == SET)
    {
        //添加用户代码
        PEout(13)=0;
        delay();
        PEout(13)=1;
        delay();

        /* 清空标志位，告诉CPU我已经处理完中断请求，可以触发下一次中断 */
        EXTI_ClearITPendingBit(EXTI_Line3);
    }
}
```

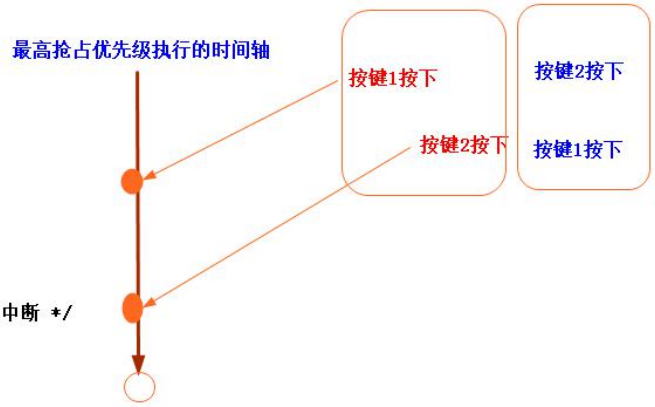
当没有发生抢占，LED灯是轮流点亮的。

```
抢占优先级3
响应优先级3
void EXTI4_IRQHandler(void)
{
        //检查是否有触发中断
    if(EXTI_GetITStatus(EXTI_Line4) == SET)
    {
        //添加用户代码
        PEout(14)=0;
        delay();
        PEout(14)=1;
        delay();

        /* 清空标志位，告诉CPU我已经处理完中断请求，可以触发下一次中断 */
        EXTI_ClearITPendingBit(EXTI_Line4);
    }
}
```

例子 3：响应优先级演示

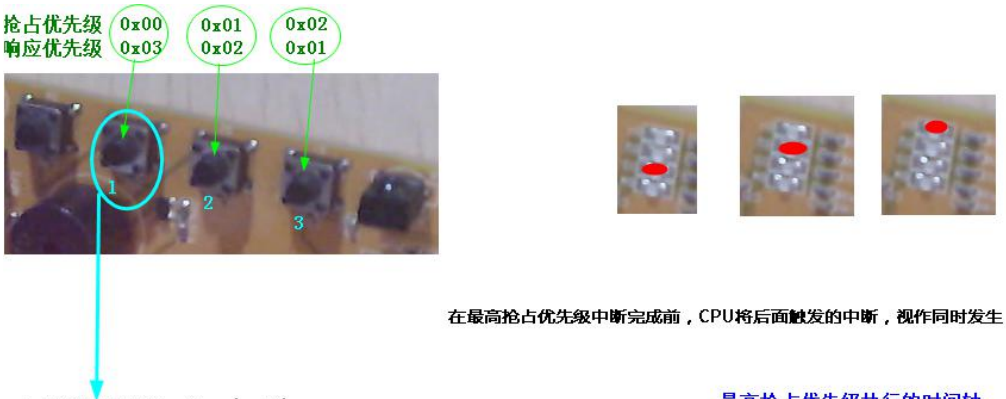
在最高抢占优先级中断完成前，CPU将后面触发的中断，视作同时发生，当抢占优先级相同时，则可根据其响应优先级进行执行

```
void EXTI3_IRQHandler(void)
{
        //检查是否有触发中断
    if(EXTI_GetITStatus(EXTI_Line3) == SET)
    {
        //添加用户代码
        PEout(13)=0;
        delay();
        PEout(13)=1;
        delay();

        /* 清空标志位，告诉CPU我已经处理完中断请求，可以触发下一次中断 */
        EXTI_ClearITPendingBit(EXTI_Line3);
    }
}
```



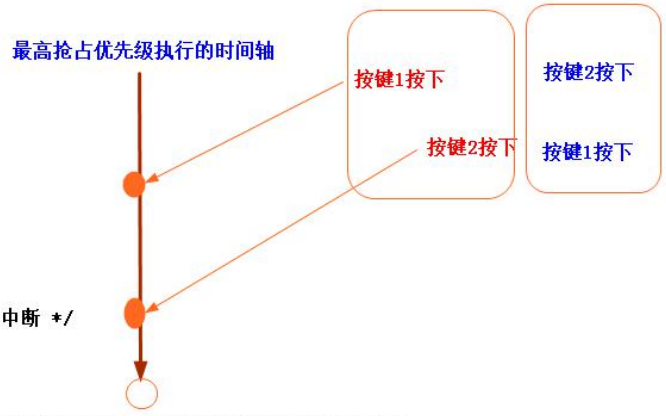
根据中断记录查询哪一个中断响应优先级高就执行谁。

例子 4：中断同时发生的时候，中断抢占优先级高的，抢先执行！



在最高抢占优先级中断完成前，CPU将后面触发的中断，视作同时发生，则先进行抢占优先级判断，再进行响应优先级判断。

```
void EXTI3_IRQHandler(void)
{
        //检查是否有触发中断
    if(EXTI_GetITStatus(EXTI_Line3) == SET)
    {
        //添加用户代码
        PEout(13)=0;
        delay();
        PEout(13)=1;
        delay();

        /* 清空标志位，告诉CPU我已经处理完中断请求，可以触发下一次中断 */
        EXTI_ClearITPendingBit(EXTI_Line3);
    }
}
```



根据中断记录查询哪一个中断抢占优先级高就执行谁！

例 5：假定设置中断优先级组为 2，然后设置中断 3(RTC 中断)的抢占优先级为 2，响应优先级为 1。
中断 6（外部中断 0）的抢占优先级为 3，响应优先级为 0。中断 7（外部中断 1）的抢占优先级为 2，响应优先级为 0。

中断 7>中断 3>中断 6。

c.注意事项

1）一般情况下，系统代码执行过程中，只设置一次中断优先级分组，比如分组 2，设置好分组之后一般不会再改变分组。
随意改变分组会导致中断管理混乱，程序出现意想不到的执行结果，如下图。

NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);

例子：

分组2	抢占优先级		响应优先级	
触摸屏中断	1	0	0	1
网络中断	1	0	1	0

往往只初始化一遍，不要随意更改，有可能导致整个中断管理紊乱！

NVIC_PriorityGroupConfig(NVIC_PriorityGroup_3);

例子：

分组3	抢占优先级		响应优先级
触摸屏中断	1 0	0	1
网络中断	1 0	1	0

2）中断优先级设置步骤

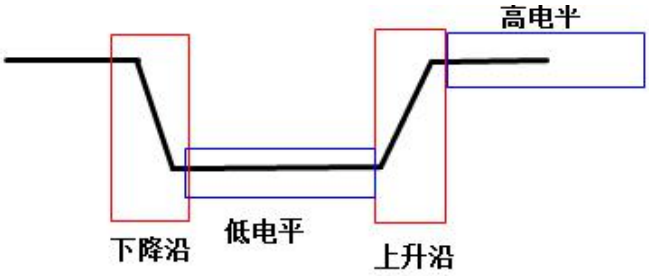
.系统运行后先设置中断优先级分组。调用函数：
void NVIC_PriorityGroupConfig(uint32_t NVIC_PriorityGroup);
整个系统执行过程中，只设置一次中断分组。

.针对每个中断，设置对应的抢占优先级和响应优先级：
void NVIC_Init(NVIC_InitTypeDef* NVIC_InitStruct);

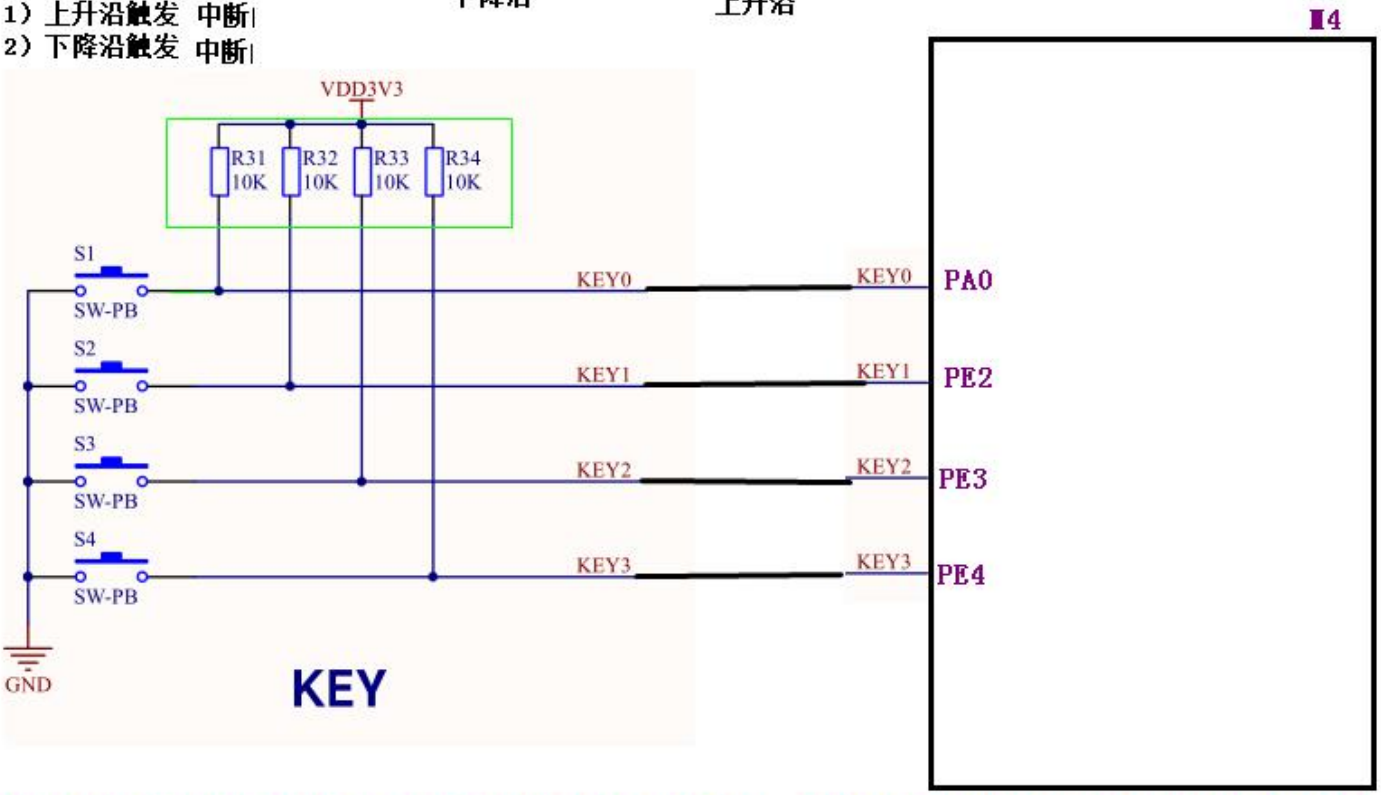
.如果需要挂起/解挂，查看中断当前激活状态，分别调用相关函数即可。

三、硬件电路

中断的触发方式：
a. 电平触发
1）高电平触发 中断|
2）低电平触发 中断|



b. 边沿触发
1）上升沿触发 中断|
2）下降沿触发 中断|



例如上拉电阻3.3K/4.7K/5.1K/10K都可以，但是电阻越小，功耗越高，10K的话就是I4能够识别的电流，同时功耗不会太高，如果是100K，电流太小的，所以引脚识别不了。

#练习

将所有的按键的检测使用外部中断进行实现。

#预习

启动文件 startup_stm32f40_41xxx.s 看汇编代码
系统时钟 SysTick,Cortex M3 与 M4 权威指南.pdf P313
定时器 TIM,STM32F4xx 中文参考手册.pdf P392