看了一下《STM32 不完全手册》上的内容发现从按键后面的内容基本上都要涉及到中断。所以决定先把中断搞定。以前使用中断很是混乱基本都是对着示例程序改改实现功能就好了。但从回过头来看看发现中断还是很复杂的。

这里先把我学习的中断的相关知识梳理一下：

1. 中断是 CM3 核自带的。所以中断要在《CM3 权威指南》中阅读。《STM32 中文参考手册》里介绍的不全。
2. CM3 内核中规定的可编程中断源有 240 个，但 STM32F103 只使用了前 68 个，这 68 个中断源和编号在 STM32F10X.H 文件 170 行有定义
3. STM32 中断和 51 中断不同的地方有:
   a) STM32 多了一个中断分组的概念，这个分组操作在程序中建议只操作一次就可以。中断分组用来决定抢占优先级和子优先级的取值范围。
   b) STM32 的每个中断源都有两种优先级，抢占优先级和子优先级。这两个优先级的取值范围由中断分组决定。抢占优先级用来中断低抢占优先级的程序。子优先级用来在抢占优先级相同时决定谁先执行。这两个优先级就好比汽车在过收费站，在汽车过收费站时有两个参数决定你什么时候过。第一你的汽车是什么类型的车，第二你前面还有多少车。第一个属性就好比抢占优先级。警察的车比普通的车的优先级高就能插到前面先过收费站。第二个属性好比子优先级。都是普通的小车说明抢占优先级都一样，谁先过就看你现在排队排的几号。
4. 中断服务函数是根据函数名来决定的，这个函数名是在.s 的启动文件中定义好的。
5. 其他的和 51 就没什么差别。初始化，等待触发中断，进中断服务函数，执行完中断函数清除中断标志，返回继续执行其他程序。

下面就根据外部中断实现一个按键点亮 LED

按照以下步骤进行

1. 中断初始化
   a) 设置中断分组 SCB->AIRCR
   b) 使能外部中断 NVIC->ISER
   c) 对外部中断的优先级设置 NVIC->IP
2. 按键初始化
   a) 按键时钟使能 RCC->APB2ENR
   b) 复用时钟使能 RCC->APB2ENR
   c) 按键相关 IO 口初始化 GPIOB->CRL
   d) 把按键引脚和对应的外部中断线连接 AFIO->EXTICR
   e) 外部中断使能 EXTI->IMR
   f) 设置触发方式 EXTI->RTSR
3. LED 初始化
4. 编写中断服务函数

第一步 中断初始化

这里先举个例子来理解中断分组和中断源，抢占优先级，子优先级的关系

比如一个公司有 68 个人 ，先给这 68 个人编了 0~68 的编号(表示 68 个可编程中断源，如串口中断，外部中断 0，外部中断 1 等)

现在公司要求每人都要被编个号，这个编号分成两个部分字母部分和数字部分，编号的

大小决定了这个人在公司的地位。并规定编号越小地位越高。但是这个编号的取值范围是根据公司的规章制度决定的。比如规章制度上规定了字母编号只能在 2^3（2 的 3 次方）以内（即只能是 A~H），数字编号只能是 2^1 以内（即 0 或者 1）。 （表示中断分组，字母是抢占，数字是子优先级） 又有一个规定在打饭时如果字母大的在打饭但字母小的也来打饭了则字母大让字母小的打完了自己在继续打饭。如果字母相同的一起来打饭，就看谁的数字编号小谁就打饭。但是如果字母相同但数字编号大的先打饭，那么后来的编号小的不能插到编号大的前面打饭。

换句话说就是字母的等级比编号的等级高，字母小的可以插队到字母大的前面。但是相同字母的不能根据数字编号大小来插队。

例如：现在有几个人编号是这样的

个人编号	工作编号
0	A00
3	A01
5	B01
7	C01
65	B00

1. 比如个人编号为 7 号的在打饭，3 号和 0 号一起去了，那么 7 号要让 3 号和 0 号插队，由于 3 号数字编号比 0 号的高说以要让 0 号在前 3 号在后，7 号在最后。
3. 现在 5 号来了 5 号的编号字母比 0 和 3 小比 7 大则 5 号插队到 7 号之前。
3. 65 号也来了，65 号字母比 7 号小，和 5 号一样。65 号的数字编号比 5 号的小但是 65 号还是排在 7 号之前 5 号之后。

总结起来就是：

1. 判断一个中断谁先执行先根据抢占优先级来判断是否比现在执行的小。是则中断当前执行抢占优先级小的中断
2. 两个相同的抢占优先级同时到来，谁的子优先级小就执行谁
3. 两个相同的抢占优先级一个先执行另一个后执行，后执行的不管子优先级高和低都不能中断先执行的那个。

中断分组

中断分组是用啦规定 IP 中抢占优先级和子优先级所占的位位数的

IP 是一个数组共 240 个元素，每个元素 8bit 表示一个中断源。STM32 中用到了前面的 68 个

我们通过寄存器 SCB->AIRCR 中的 8~10 位来中断分组

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

根据上表的中断分组我们有以下结论

黄色表示抢占优先级　蓝色表示子优先级

组 0

组 1

组 2

组 3

组 4

注：这里的中断分组是对 IP 寄存器中的所有元素都适用的。IP[0]是这样，IP[68]也是这样，且中断分组一经设置尽量不要修改

位段	名称	类型	复位值	描述
31:16	VECTKEY	RW	-	访问钥匙：任何对该寄存器的写操作，都必须同时把 0x05FA 写入此段，否则写操作被忽略。若读取此半字，则 0xFA05
15	ENDIANESS	R	-	指示端设置。1＝大端(BE8)，0＝小端。此值是在复位时确定的，不能更改。
10:8	PRIGROUP	R/W	0	优先级分组
2	SYSRESETREQ	W	-	请求芯片控制逻辑产生一次复位
1	VECTCLRACTIVE	W	-	清零所有异常的活动状态信息。通常只在调试时用，或者在 OS 从错误中恢复时用。
0	VECTRESET	W	-	复位 CM3 处理器内核（调试逻辑除外），但是此复位不影响芯片上在内核以外的电路

110

在程序中设置的就是 SCB->AIRCR

从上面的描述中这个寄存器需要写入访问钥匙才能操作

所以在程序中的语句就是

SCB->AIRCR&=0X05FAF8FF//清除分组

SCB->AIRCR|=0X05FA0400//对 10~8 位写入 100 即分组 3

下面就是对中断的使能和优先级的分配

先看使能

```
typedef struct
{
  __IO uint32_t ISER[8];                /*!< Interrupt Set Enable Register           */
       uint32_t RESERVED0[24];
  __IO uint32_t ICER[8];                /*!< Interrupt Clear Enable Register         */
       uint32_t RSERVED1[24];
  __IO uint32_t ISPR[8];                /*!< Interrupt Set Pending Register          */
       uint32_t RESERVED2[24];
  __IO uint32_t ICPR[8];                /*!< Interrupt Clear Pending Register        */
       uint32_t RESERVED3[24];
  __IO uint32_t IABR[8];                /*!< Interrupt Active bit Register           */
       uint32_t RESERVED4[56];
  __IO uint8_t  IP[240];                /*!< Interrupt Priority Register, 8Bit wide  */
       uint32_t RESERVED5[644];
  __O  uint32_t STIR;                   /*!< Software Trigger Interrupt Register     */
}  NVIC_Type;
```

这里用到的就两个成员，ISER[8]和 IP[240]

ISER 用来使能中断，这里数据有 8 个元素每个元素有 32 位 STM32 有 68 个中断，说以用到了前 3 个元素就行。这里的的每一位对应一个中断的使能。例如 6 位就表示外部中断 0 的中断使能，7 位就表示外部中断 1 的使能。具体请看 STM32F10X.H 中 170 行

这里用到外部中断 0 则

NVIC->ISER[0]|=0X40;//使能外部中断 0

IP 是用来中断优先级分组的,一个元素对应一个中断源

根据前面的图我们用了组 2 就是抢占优先级有 3 位即在 IP 的 7~5 位上设置的都是抢占优先级。4 位设置的则是子优先级

我们用到了外部中断 0 是 6 号中断源则

NVIC->IP[6]=0x30;//0X30 展开是 001 1 0000 表示抢占优先级为 1 子优先级为 1

到这里中断设置部分结束

第二步 按键初始化

按键初始化前面都一样，只是要多初始化 RCC->APB2ENR 中的位 0 的复用管脚时钟就好

比前面按键扫描程序多的是关于外部中断的部分

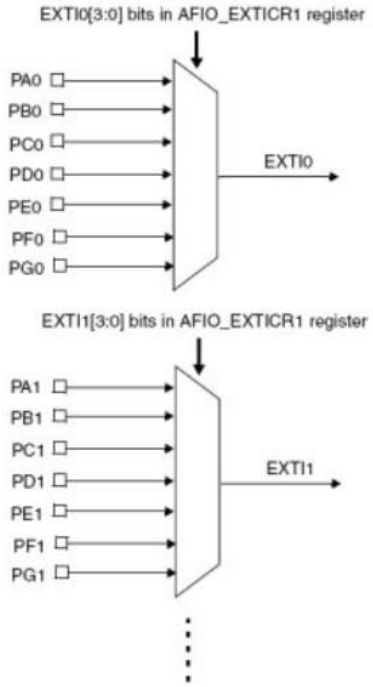
分成三步

1. 把引脚和外部中断线相连接
2. 使能外部中断相应中断线
3. 设置触发方式，上升沿还是下降沿

根据下图可以理解一下什么叫把引脚和中断线相连

STM32 中断线有 0~15 而引脚有很多，怎么办？

用分组的方法把引脚数字编号一样的分成一组通过选择器把引脚和中断线相连。



通过寄存器

AFIO->EXTICR[]来实现的

这是一个数组有 4 个元素，其中元素 0 的结构如下

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

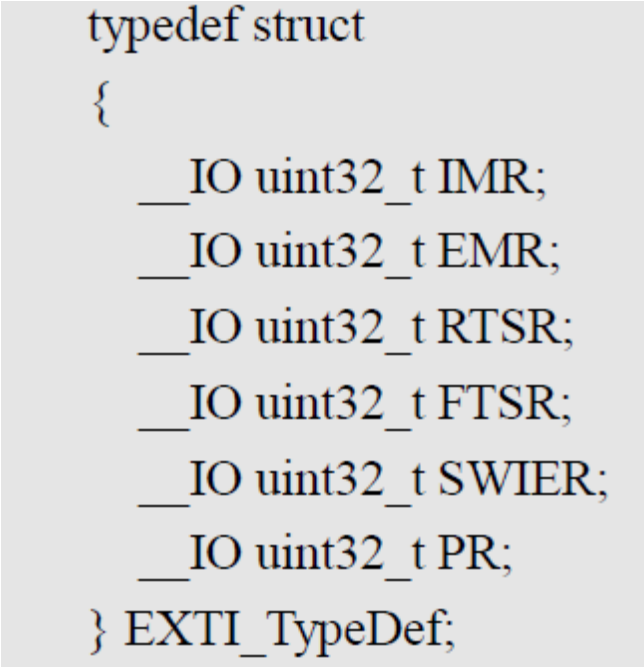
位31:16	保留。
位15:0	**EXTIx[3:0]**：EXTIx配置(x = 0 … 3) 这些位可由软件读写，用于选择EXTIx外部中断的输入源。 0000：PA[x]管脚　　0100：PE[x]管脚 0001：PB[x]管脚　　0101：PF[x]管脚 0010：PC[x]管脚　　0110：PG[x]管脚 0011：PD[x]管脚

4 位一组用来选择 ABCDEFG 中的哪个一个

我们这根据硬件连接 PB0 说以选择 EXTI0[3..0]并写入 0001 来连接 PB0 的引脚

AFIO->EXTI[0]=0X0001;//连接 PB0　如果连接 PC0 则写入 0x0002;

下面是使能对应的外部中断线



用到这个结构体里的第一个成员 IMR

IMR 有 19 位有效 15~0 是表示中断线 0~15 的使能 16 是 PVD，17 是 RTC,18 是 USB

我们是 EXTI0 所以

EXTI->IMR|=0X00000001;

下面是触发方式

是上图结构体中的 RTSR 或者 FTSR

RTSR 上升沿触发

FTSR 下降沿触发

这个也是 19 位在对饮位写 1 就设置好了触发方式

我们采用 EXTI0 上升沿触发

EXTI->RTSR|=0X00000001;

到这就初始化好了按键和中断

第三步 中断服务函数

在 51 中有一个中断号比如外部中断 0 就是 interrupt 0 函数名可以随便，在 STM32 没有了中断入口地址的中断号，但函数名固定，在.S 的启动文件中有定义

如外部中断 0 的中断服务函数要写成

```
void EXTI0_IRQHandler(void)
{
    ……..
}
```

写出这个函数后在里面就是让灯点亮就可以了

最后要把中断请求标志删除就是上面结构体的 PR

也是一个 19 位有效的基础器,对应位写 1 就清除了

EXTI->PR|=0x00000001;