5	12	可设置	RCC	复位和时钟控制(RCC)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通道1	DMA1通道1全局中断	0x0000_006C
12	19	可设置	DMA1通道2	DMA1通道2全局中断	0x0000_0070
13	20	可设置	DMA1通道3	DMA1通道3全局中断	0x0000_0074
14	21	可设置	DMA1通道4	DMA1通道4全局中断	0x0000_0078
15	22	可设置	DMA1通道5	DMA1通道5全局中断	0x0000_007C
16	23	可设置	DMA1通道6	DMA1通道6全局中断	0x0000_0080
17	24	可设置	DMA1通道7	DMA1通道7全局中断	0x0000_0084
18	25	可设置	ADC1_2	ADC1和ADC2的全局中断	0x0000_0088
19	26	可设置	USB_HP_CAN_TX	USB高优先级或CAN发送中断	0x0000_008C
20	27	可设置	USB_LP_CAN_RX0	USB低优先级或CAN接收0中断	0x0000_0090
21	28	可设置	CAN_RX1	CAN接收1中断	0x0000_0094
22	29	可设置	CAN_SCE	CAN SCE中断	0x0000_0098
23	30	可设置	EXTI9_5	EXTI线[9:5]中断	0x0000_009C
24	31	可设置	TIM1_BRK	TIM1刹车中断	0x0000_00A0
25	32	可设置	TIM1_UP	TIM1更新中断	0x0000_00A4
26	33	可设置	TIM1_TRG_COM	TIM1触发和通信中断	0x0000_00A8
27	34	可设置	TIM1_CC	TIM1捕获比较中断	0x0000_00AC
28	35	可设置	TIM2	TIM2全局中断	0x0000_00B0
29	36	可设置	TIM3	TIM3全局中断	0x0000_00B4
30	37	可设置	TIM4	TIM4全局中断	0x0000_00B8
31	38	可设置	I2C1_EV	I^2C1事件中断	0x0000_00BC
32	39	可设置	I2C1_ER	I^2C1错误中断	0x0000_00C0
33	40	可设置	I2C2_EV	I^2C2事件中断	0x0000_00C4
34	41	可设置	I2C2_ER	I^2C2错误中断	0x0000_00C8
35	42	可设置	SPI1	SPI1全局中断	0x0000_00CC
36	43	可设置	SPI2	SPI2全局中断	0x0000_00D0
37	44	可设置	USART1	USART1全局中断	0x0000_00D4
38	45	可设置	USART2	USART2全局中断	0x0000_00D8
39	46	可设置	USART3	USART3全局中断	0x0000_00DC
40	47	可设置	EXTI15_10	EXTI线[15:10]中断	0x0000_00E0
41	48	可设置	RTCAlarm	连到EXTI的RTC闹钟中断	0x0000_00E4
42	49	可设置	USB唤醒	连到EXTI的从USB待机唤醒中断	0x0000_00E8
43	50	可设置	TIM8_BRK	TIM8刹车中断	0x0000_00EC
44	51	可设置	TIM8_UP	TIM8更新中断	0x0000_00F0
45	52	可设置	TIM8_TRG_COM	TIM8触发和通信中断	0x0000_00F4
46	53	可设置	TIM8_CC	TIM8捕获比较中断	0x0000_00F8
47	54	可设置	ADC3	ADC3全局中断	0x0000_00FC

（图片来自 STM32 使用手册，只需看，不需要熟记，知道大概这么多中断就好）

三、配置中断相关寄存器

在讲解寄存器之前，我们先要对寄存器有一定的了解。何为寄存器呢？我们可以这样理解，它们是位于芯片内部的一间又一间房子，房子的装潢由我们编写程序的人决定，每间房子进行了串联，信息流按照一定的顺序经过每间房间，这样芯片就知道总体的安排是如何的，以此进行相关的操作。虽然如此，但是不同的寄存器内也有不同的家具，而且是固定的，在 32 中我们就可以认为每间屋子中有 32（[31:0] 意味着从第一位到第 32 位）件家具。对每个家具进行配置，然后达到焕然一新的状态，就近似于我们在书写功能程序前进行的准备工作。

```
/*
cortex - m3内核分组方式（8组）结构体表达方式:
*/
typedef struct
{
  __IO uint32_t ISER[8];              中断使能设置寄存器          /*!< 偏移量: 0x000  Interrupt Set Enable Register           */
       uint32_t RESERVED0[24];
  __IO uint32_t ICER[8];               中断清除使能寄存器         /*!<偏移量: 0x080  Interrupt Clear Enable Register        */
       uint32_t RSERVED1[24];
  __IO uint32_t ISPR[8];               中断挂起设置寄存器        /*!< 偏移量: 0x100  Interrupt Set Pending Register          */
       uint32_t RESERVED2[24];
  __IO uint32_t ICPR[8];               中断清除挂起寄存器        /*!<偏移量: 0x180  Interrupt Clear Pending Register        */
       uint32_t RESERVED3[24];
  __IO uint32_t IABR[8];                中断激活状态位寄存器       /*!< 偏移量: 0x200  Interrupt Active bit Register           */
       uint32_t RESERVED4[56];
  __IO uint8_t  IP[240];               中断优先级寄存器       /*!< 偏移量: 0x300  Interrupt Priority Register (8Bit wide) */
       uint32_t RESERVED5[644];        软件触发方式寄存器
  __O  uint32_t STIR;                        /*!< 偏移量: 0xE00  Software Trigger Interrupt Register     */
}  NVIC_Type;
```

P 2-4-92

```
/*
STM32分组（5组）方式结构体表达方式
typedef struct
{
  vu32 ISER[2];
  u32  RESERVED0[30];
  vu32 ICER[2];
  u32  RSERVED1[30];
  vu32 ISPR[2];
  u32  RESERVED2[30];
  vu32 ICPR[2];
  u32  RESERVED3[30];
  vu32 IABR[2];
  u32  RESERVED4[62];
  vu32 IPR[15];
} NVIC_TypeDef;
*/
```

P 2-4-93

以下寄存器介绍：

寄存器	寄存器符号	位和功能	备注特点
ISER0 和 ISER1	中断配置使能寄存器	ISER0[31:0]:置 1 可使能中断向量表中从位置编号为 0~31 的 32 个中断源； ISER1[31:0]:置 1 可使能中断向量表中从位置编号为 32~59 的 29 个中断源	1. 在 3.5 库中的 stm32f10x.h 文件中对 64 个中断都有定义； 2. 两个寄存器为写 1 有效，写 0 无效
ICER0 和 ICER1	中断清除使能寄存器	功能和 ICER 相反，并且每一位与 ICER 中的位一一对应	两个寄存器为写 1 有效，写 0 无效
ISPR0 和 ISPR1	中断挂起清除寄存器	ISPR0[31:0]:置 1 可挂起中断向量表中从位置编号为 0~31 的 32 个中断源； ISPR1[31:0]:置 1 可挂起中断向量表中从位置编号为 32~59 的 29 个中断源	两个寄存器写 1 有效，写 0 无效
ICPR0 和 ICPR1	中断挂起清除寄存器	功能和 ISPR 相反，并且每一位与 ISPR 中的位一一对应	两个寄存器写 1 有效，写 0 无效
IABR 和 IABR1	中断激活标志寄存器	可以说就是中断正在执行标志位	1. 只读 2. 执行中断时硬件置 1，中断执行完毕硬件置 0
IPR 0~14	中断优先级控制寄存器	[31:28]:为向量表中的中断分配优先级（IPRx+3） [23:20]：为向量表中的中断分配优先级（IPRx+2） [15:12]：为向量表中的中断分配优先级（IPRx+1） [7:4]：为向量表中的中断分配优先级（IPRx+0）	1. 编号从 0-14 共 15 个寄存器为 60 个中断源分配优先级； 2. STM32 把 60 个中断分成 5 组；即 0~4 组，分组设置由 AIRC 的 bit10~8 来决定；

http://blog.csdn.net/wuyuzun

P 2-4-94

另外要提到的一点是，尽管默认每个寄存器都是 32 位，但是有些寄存器内的某些位却是“保留”不动的，可以进行位操作的只有部分位而已。关于此点，等待同学们遇到时，再作更加详细的了解。

对一些概念的解释：

1. 挂起：当置位中断挂起寄存器的时候，相应的中断将被挂起，这是这个中断将不会立即执行，而是等待可执行的时候再执行；比如高低级别的中断同时产生，就先挂起低级别的中断，等高级别的中断执行完毕，解除并执行低级中断。

2. 对中断优先级控制寄存器的解释：

组	AIRC[10:8]	IPR[7:4]分配情况	分配结果	备注
0	111	0：4	0 位抢占优先级， 4 位响应优先级	2^0=1 个抢占优先级， 2^4=16 位响应优先级
1	110	1：3	1 位抢占优先级， 3 位响应优先级	2^1=2 个抢占优先级， 2^3=8 个响应优先级
2	101	2：2	2 位抢占优先级， 2 位响应优先级	2^2=4 个抢占优先级， 2^2=4 个响应优先级
3	100	3：1	3 位抢占优先级， 1 位响应优先级	2^3=8 个抢占优先级， 2^1=2 个响应优先级
5	011	4：0	4 位抢占优先级， 0 位响应优先级	2^4=16 个抢占优先级， 2^0=1 个响应优先级

http://blog.csdn.net/wuyuzun

P 2-4-95

四、编程步骤

1、选择优先级分组

（1）此函数在库文件 misc.h 文件下；

（2）参数可参照下面图片：

NVIC_PriorityGroup	先占优先级	从优先级	描述
NVIC_PriorityGroup_0	0	0-15	先占优先级0位从优先级4位
NVIC_PriorityGroup_1	0-1	0-7	先占优先级1位从优先级3位
NVIC_PriorityGroup_2	0-3	0-3	先占优先级2位从优先级2位
NVIC_PriorityGroup_3	0-7	0-1	先占优先级3位从优先级1位
NVIC_PriorityGroup_4	0-15	0	先占优先级4位从优先级0位

http://blog.csdn.net/wuyuzun

P 2-4-96

（3）功能：选择分组方式；

void NVIC_PriorityGroupConfig(uint32_t NVIC_PriorityGroup);

举个栗子：NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);

//抢占优先级可选 0~3，响应优先级可选 0~3；

2、选择，配置，并使能中断

```
NVIC_InitStructure.NVIC_IRQChannel = EXTI2_IRQn; // 选择 EXTI2 中断
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x02; // 抢占优先级为 2
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x02; // 响应优先级为 2
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE; // 使能 EXTI2 中断;
NVIC_Init(&NVIC_InitStructure); // 初始化以上参数;
```

3、写出相应中断函数

例子：

```
void EXTI2_IRQHandler(void)
{
// 逻辑代码部分;
EXTI_ClearITPendingBit(EXTI_Line2);
}
```

（摘自 CSDN 博客“wuyuzun”的原创博文《STM32 的 NVIC 和中断的总结》https://blog.csdn.net/wuyuzun/article/details/72783152）

关于中断的介绍，我们就进行到这里。

以上诸多文字，无论是我个人皆有经验书写，或是摘自学习网站上的介绍清晰且易懂的资料，都是希望大家可以对于 32 有个全面且初步的了解。学习之路，道阻且长，尽力而为即是。

注：之前提到的资料，我会进行整理，然后使用合适的平台发放给大家。