优先级可以被分为高低两个位段，分别是抢占优先级和亚优先级

Cortex-M3允许具有较少中断源时使用较少的寄存器位指定中断源的优先级，因此STM32把指定中断优先级的寄存器位减少到4位 AIRC 中 PRIGROUP 的值规定了设置和确定每个外部中断通道优先级的格式。例如，在上面将 0x05 写入了 AIRC 中 PRIGROUP，也就规定了当前系统中只能有 4 个抢先式优先级，相同的抢先式优先级下还可以有 4 个不同级别的子优先级，他们分别为：   
  
位[7：6]                   位[5：4]                  位[3：0]   
00         0 号抢先优先级   00        0 号子优先级    无效   
01         1 号抢先优先级   01        1 号子优先级    无效   
10         2 号抢先优先级   10        2 号子优先级    无效   
11         3 号抢先优先级   11        3 号子优先级    无效

在Cortex—M3的优先级分配中，较低的优先级值具有较高的优先级。

数值小的优先级高, 数值大的优先级低

a 只能高抢先优先级的中断可以打断低抢先优先级的中断服务，构成中断嵌套。

b 当 2（n）个相同抢先优先级的中断出现，它们之间不能构成中断嵌套，但 STM32 首先响应子优先级高的中断。

c 当 2（n）个相同抢先优先级和相同子优先级的中断出现，STM32 首先响应中断通道所对应的中断向量地址低的那个中断。

可以通过调用STM32的固件库中的函数NVIC\_PriorityGroupConfig()选择使用哪种优先级分组方式，这个函数的参数有下列5种：  
  
NVIC\_PriorityGroup\_0 => 选择第0组  
NVIC\_PriorityGroup\_1 => 选择第1组  
NVIC\_PriorityGroup\_2 => 选择第2组  
NVIC\_PriorityGroup\_3 => 选择第3组  
NVIC\_PriorityGroup\_4 => 选择第4组

第0组：所有4位用于指定响应优先级   ( 从不嵌套 )  
第1组：最高1位用于指定抢占式优先级，最低3位用于指定响应优先级  
第2组：最高2位用于指定抢占式优先级，最低2位用于指定响应优先级  
第3组：最高3位用于指定抢占式优先级，最低1位用于指定响应优先级  
第4组：所有4位用于指定抢占式优先级 ( 最大的嵌套可能性 ) ( 类似于 STR710 )

subpriority 开始的位号 :

如果从 7 号开始 那么没有 pre-emption priority, [ b7 b6 b5 b4 ] 全部为 subpriority

如果从 6 号开始 那么 [ b7 ]为 pre-emption priority, [ b6 b5 b4 ] 为 subpriority

如果从 3 号开始 那么 [ b7 b6 b5 b4 ]为 pre-emption priority, 没有 subpriority

[pre-emption priority] :: [ subpriority ] ( STM32 共计 4 位 ) 所以无论如何配置做多只能有 16 级, 数值最小的具有最高的优先级

在程序中合理按排中断优先级可以达到具有高抢占式优先级的中断可以在具有低抢占式优先级的中断处理过程中被响应，即中断嵌套，或者说高抢占式优先级的中断可以嵌套低抢占式优先级的中断。

要注意的几点是：  
  
1）如果指定的抢占式优先级别或响应优先级别超出了选定的优先级分组所限定的范围，将可能得到意想不到的结果；  
  
2）抢占式优先级别相同的中断源之间没有嵌套关系；  
  
3）如果某个中断源被指定为某个抢占式优先级别，又没有其它中断源处于同一个抢占式优先级别，则可以为这个中断源指定任意有效的响应优先级别。

#define NVIC\_PriorityGroup\_0         ((uint32\_t)0x700) /\*!< 0 bits for pre-emption priority  
                                                            4 bits for subpriority \*/  
#define NVIC\_PriorityGroup\_1         ((uint32\_t)0x600) /\*!< 1 bits for pre-emption priority  
                                                            3 bits for subpriority \*/  
#define NVIC\_PriorityGroup\_2         ((uint32\_t)0x500) /\*!< 2 bits for pre-emption priority  
                                                            2 bits for subpriority \*/  
#define NVIC\_PriorityGroup\_3         ((uint32\_t)0x400) /\*!< 3 bits for pre-emption priority  
                                                            1 bits for subpriority \*/  
#define NVIC\_PriorityGroup\_4         ((uint32\_t)0x300) /\*!< 4 bits for pre-emption priority  
                                                            0 bits for subpriority \*/

/\*\*  
  \* @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  
  \*                                4 bits for subpriority  
  \*     @arg NVIC\_PriorityGroup\_1: 1 bits for pre-emption priority  
  \*                                3 bits for subpriority  
  \*     @arg NVIC\_PriorityGroup\_2: 2 bits for pre-emption priority  
  \*                                2 bits for subpriority  
  \*     @arg NVIC\_PriorityGroup\_3: 3 bits for pre-emption priority  
  \*                                1 bits for subpriority  
  \*     @arg NVIC\_PriorityGroup\_4: 4 bits for pre-emption priority  
  \*                                0 bits for subpriority  
  \* @retval None  
  \*/  
void NVIC\_PriorityGroupConfig(uint32\_t NVIC\_PriorityGroup)  
{  
  /\* Check the parameters \*/  
  assert\_param(IS\_NVIC\_PRIORITY\_GROUP(NVIC\_PriorityGroup));  
    
  /\* Set the PRIGROUP[10:8] bits according to NVIC\_PriorityGroup value \*/  
  SCB->AIRCR = AIRCR\_VECTKEY\_MASK | NVIC\_PriorityGroup;  
} 

/\*\*  
  \* @brief  Initializes the NVIC peripheral according to the specified  
  \*   parameters in the NVIC\_InitStruct.  
  \* @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)  
{  
  uint32\_t tmppriority = 0x00, tmppre = 0x00, tmpsub = 0x0F;  
    
  /\* Check the parameters \*/  
  assert\_param(IS\_FUNCTIONAL\_STATE(NVIC\_InitStruct->NVIC\_IRQChannelCmd));  
  assert\_param(IS\_NVIC\_PREEMPTION\_PRIORITY(NVIC\_InitStruct->NVIC\_IRQChannelPreemptionPriority));    
  assert\_param(IS\_NVIC\_SUB\_PRIORITY(NVIC\_InitStruct->NVIC\_IRQChannelSubPriority));  
      
  if (NVIC\_InitStruct->NVIC\_IRQChannelCmd != DISABLE)  
  {  
    /\* Compute the Corresponding IRQ Priority --------------------------------\*/      
    tmppriority = (0x700 - ((SCB->AIRCR) & (uint32\_t)0x700))>> 0x08;  
    tmppre = (0x4 - tmppriority);  
    tmpsub = tmpsub >> tmppriority;

    tmppriority = (uint32\_t)NVIC\_InitStruct->NVIC\_IRQChannelPreemptionPriority << tmppre;  
    tmppriority |=  NVIC\_InitStruct->NVIC\_IRQChannelSubPriority & tmpsub;  
    tmppriority = tmppriority << 0x04;  
          
    NVIC->IP[NVIC\_InitStruct->NVIC\_IRQChannel] = tmppriority;  
      
    /\* Enable the Selected IRQ Channels --------------------------------------\*/  
    NVIC->ISER[NVIC\_InitStruct->NVIC\_IRQChannel >> 0x05] =  
      (uint32\_t)0x01 << (NVIC\_InitStruct->NVIC\_IRQChannel & (uint8\_t)0x1F);  
  }  
  else // if (NVIC\_InitStruct->NVIC\_IRQChannelCmd == DISABLE)  
  {  
    /\* Disable the Selected IRQ Channels -------------------------------------\*/  
    NVIC->ICER[NVIC\_InitStruct->NVIC\_IRQChannel >> 0x05] =  
      (uint32\_t)0x01 << (NVIC\_InitStruct->NVIC\_IRQChannel & (uint8\_t)0x1F);  
  }  
}

# 2 stm32如何开关中断

\_\_disable\_irq(); //disable interrupts  
\_\_enable\_irq(); //enable interrupts