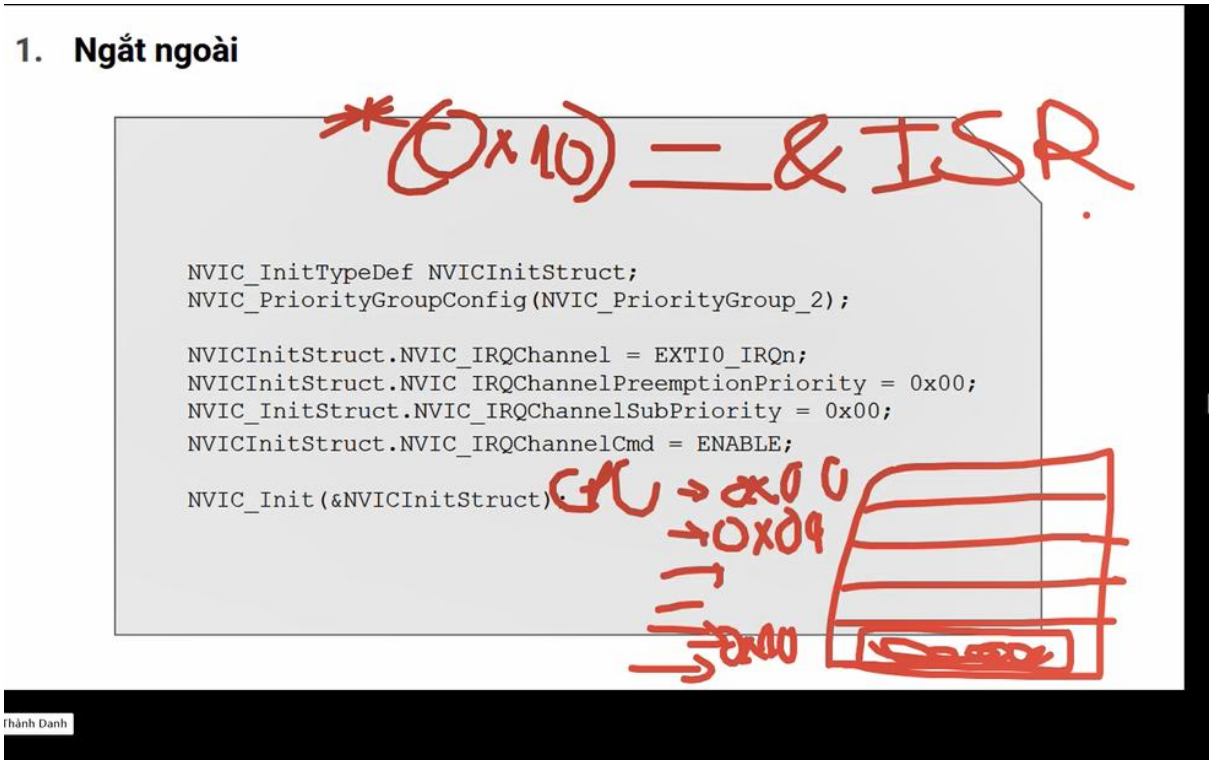
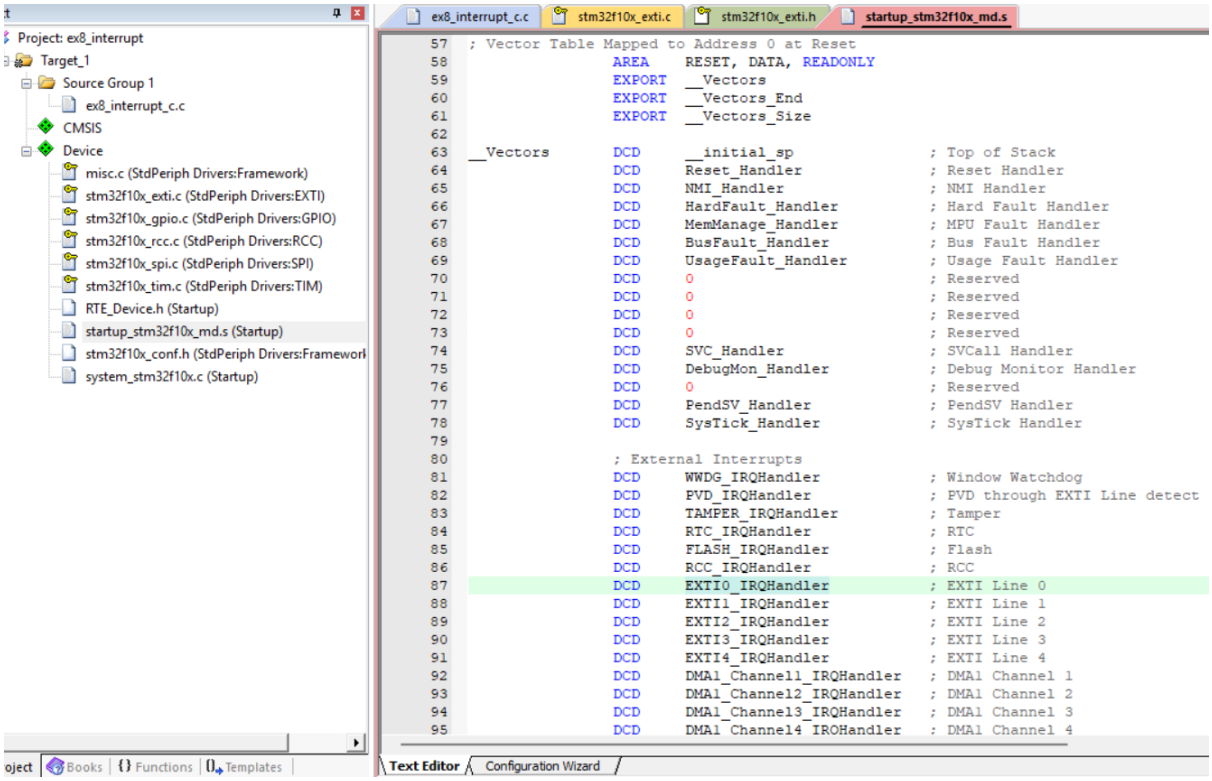
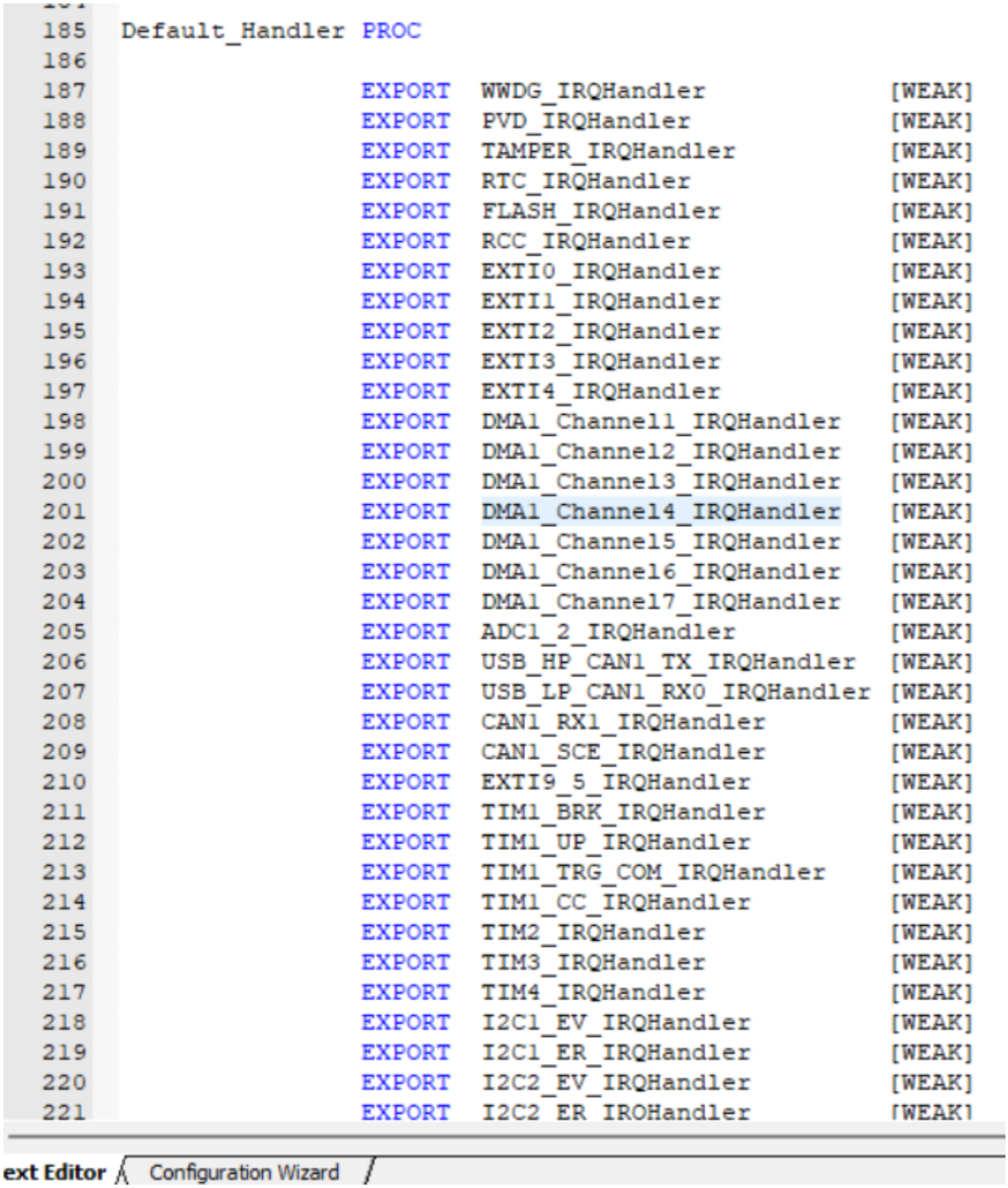
Bài 8: Ngắt ngoài, Ngắt Timer, Ngắt truyền thông

Sơ lược về ngắt

- Cấu hình ngắt cần chú ý kích hoạt chế độ ngắt của từng loại ngoại vi,
- Nguồn ngắt (ngắt ngoài-nút nhấn, xung tín hiệu, vs Timer & Ngắt truyền thông using Cờ báo trạng thái (update event, flag nhận dữ liệu, truyền dữ liệu…)),
- Chương trình ngắt ISR sẽ có tên được quy định sẵn trong bảng vector table của ngắt.
- Ngắt ngoài từ các chân GPIO cần bật clock cho AFIO. AFIO phục vụ cho việc ngắt ngoài như bấm nút nhấn…, nối điện trở treo pull up, pull down nếu cần thiết.
- Ngắt trên các ngoại vi TIMER, SPI, UART, I2C… thì đã có bộ phần cứng riêng, khi bật clock cho ngoại vi nào nó sẽ tự động nối chân gpio đến các chân của ngoại vi đó. Bật chương trình ngắt của các ngoại vi này đã có sẵn hàm điều khiển ngắt trong thư viện.
- Các chương trình ngắt được quản lý NVIC-Nested Vector Interrupt Controller. NVIC phân cấp độ ưu tiên cho ngắt và thực hiện điều khiển con trỏ PC nhảy đến chương trình ngắt ISR.
- Bảng Vector ngắt đã được NSX định nghĩa sẵn trong bộ nhớ cpu: Tên chương trình ngắt ISR, địa chỉ cố định của ISR, giá trị khởi tạo, Handler. Bạn chỉ việc viết thêm code điều khiển vào các chương trình ngắt có sẵn.
- Khi ngắt xảy ra CPU biết nguồn ngắt là từ đâu rồi, nó sẽ dò dải địa chỉ trong bảng vector ngắt để coi chỗ nào là nguồn ngắt đã báo về cho mình. Giá trị trong địa chỉ đó chính là địa chỉ của ISR. Và nó bắt thanh ghi PC trỏ đến chạy chương trình ISR đó

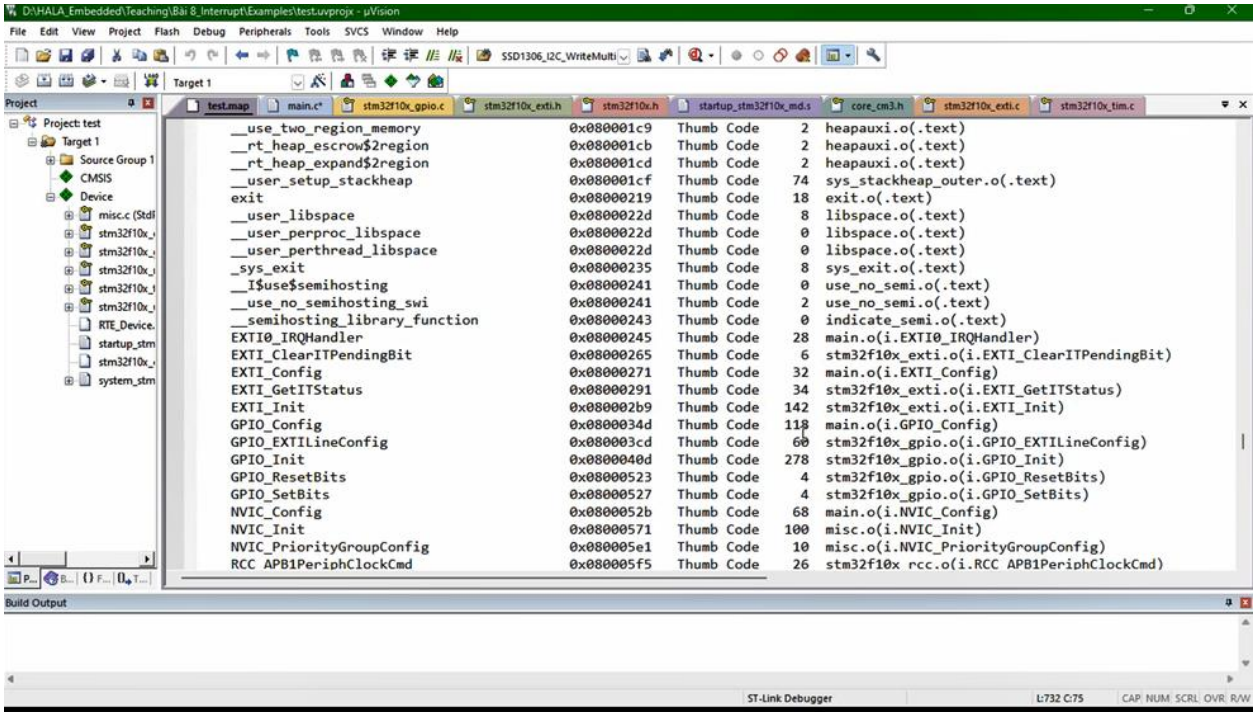



Project: ex8_interrupt
Target_1
Source Group 1
ex8_interrupt_c.c
CMSIS
Device
misc.c (StdPeriph Drivers:Framework)
stm32f10x_exti.c (StdPeriph Drivers:EXTI)
stm32f10x_gpio.c (StdPeriph Drivers:GPIO)
stm32f10x_rcc.c (StdPeriph Drivers:RCC)
stm32f10x_spi.c (StdPeriph Drivers:SPI)
stm32f10x_tim.c (StdPeriph Drivers:TIM)
RTE_Device.h (Startup)
startup_stm32f10x_md.s (Startup)
stm32f10x_conf.h (StdPeriph Drivers:Framework
system_stm32f10x.c (Startup)
oject
Books
Functions
Templates
ex8_interrupt_c.c
stm32f10x_exti.c
stm32f10x_exti.h
startup_stm32f10x_md.s
57 ; Vector Table Mapped to Address 0 at Reset
58 AREA RESET, DATA, READONLY
59 EXPORT __Vectors
60 EXPORT __Vectors_End
61 EXPORT __Vectors_Size
62
63 __Vectors DCD __initial_sp ; Top of Stack
64 DCD Reset_Handler ; Reset Handler
65 DCD NMI_Handler ; NMI Handler
66 DCD HardFault_Handler ; Hard Fault Handler
67 DCD MemManage_Handler ; MPU Fault Handler
68 DCD BusFault_Handler ; Bus Fault Handler
69 DCD UsageFault_Handler ; Usage Fault Handler
70 DCD 0 ; Reserved
71 DCD 0 ; Reserved
72 DCD 0 ; Reserved
73 DCD 0 ; Reserved
74 DCD SVC_Handler ; SVCall Handler
75 DCD DebugMon_Handler ; Debug Monitor Handler
76 DCD 0 ; Reserved
77 DCD PendSV_Handler ; PendSV Handler
78 DCD SysTick_Handler ; SysTick Handler
79
80 ; External Interrupts
81 DCD WWDG_IRQHandler ; Window Watchdog
82 DCD PVD_IRQHandler ; PVD through EXTI Line detect
83 DCD TAMPER_IRQHandler ; Tamper
84 DCD RTC_IRQHandler ; RTC
85 DCD FLASH_IRQHandler ; Flash
86 DCD RCC_IRQHandler ; RCC
87 DCD EXTI0_IRQHandler ; EXTI Line 0
88 DCD EXTI1_IRQHandler ; EXTI Line 1
89 DCD EXTI2_IRQHandler ; EXTI Line 2
90 DCD EXTI3_IRQHandler ; EXTI Line 3
91 DCD EXTI4_IRQHandler ; EXTI Line 4
92 DCD DMA1_Channel1_IRQHandler ; DMA1 Channel 1
93 DCD DMA1_Channel2_IRQHandler ; DMA1 Channel 2
94 DCD DMA1_Channel3_IRQHandler ; DMA1 Channel 3
95 DCD DMA1_Channel4_IRQHandler ; DMA1 Channel 4
Text Editor
Configuration Wizard



```
185  Default_Handler PROC
186
187                  EXPORT  WWDG_IRQHandler             [WEAK]
188                  EXPORT  PVD_IRQHandler              [WEAK]
189                  EXPORT  TAMPER_IRQHandler           [WEAK]
190                  EXPORT  RTC_IRQHandler              [WEAK]
191                  EXPORT  FLASH_IRQHandler            [WEAK]
192                  EXPORT  RCC_IRQHandler              [WEAK]
193                  EXPORT  EXTI0_IRQHandler            [WEAK]
194                  EXPORT  EXTI1_IRQHandler            [WEAK]
195                  EXPORT  EXTI2_IRQHandler            [WEAK]
196                  EXPORT  EXTI3_IRQHandler            [WEAK]
197                  EXPORT  EXTI4_IRQHandler            [WEAK]
198                  EXPORT  DMA1_Channel1_IRQHandler    [WEAK]
199                  EXPORT  DMA1_Channel2_IRQHandler    [WEAK]
200                  EXPORT  DMA1_Channel3_IRQHandler    [WEAK]
201                  EXPORT  DMA1_Channel4_IRQHandler    [WEAK]
202                  EXPORT  DMA1_Channel5_IRQHandler    [WEAK]
203                  EXPORT  DMA1_Channel6_IRQHandler    [WEAK]
204                  EXPORT  DMA1_Channel7_IRQHandler    [WEAK]
205                  EXPORT  ADC1_2_IRQHandler           [WEAK]
206                  EXPORT  USB_HP_CAN1_TX_IRQHandler   [WEAK]
207                  EXPORT  USB_LP_CAN1_RX0_IRQHandler  [WEAK]
208                  EXPORT  CAN1_RX1_IRQHandler         [WEAK]
209                  EXPORT  CAN1_SCE_IRQHandler         [WEAK]
210                  EXPORT  EXTI9_5_IRQHandler          [WEAK]
211                  EXPORT  TIM1_BRK_IRQHandler         [WEAK]
212                  EXPORT  TIM1_UP_IRQHandler          [WEAK]
213                  EXPORT  TIM1_TRG_COM_IRQHandler     [WEAK]
214                  EXPORT  TIM1_CC_IRQHandler          [WEAK]
215                  EXPORT  TIM2_IRQHandler             [WEAK]
216                  EXPORT  TIM3_IRQHandler             [WEAK]
217                  EXPORT  TIM4_IRQHandler             [WEAK]
218                  EXPORT  I2C1_EV_IRQHandler          [WEAK]
219                  EXPORT  I2C1_ER_IRQHandler          [WEAK]
220                  EXPORT  I2C2_EV_IRQHandler          [WEAK]
221                  EXPORT  I2C2_ER_IRQHandler          [WEAK]
```


Vào file maps trong listing mở ra sẽ thấy địa chỉ các chương trình và bạn có thể thấy được kích thước các hàm đã viết là bao nhiêu byte.

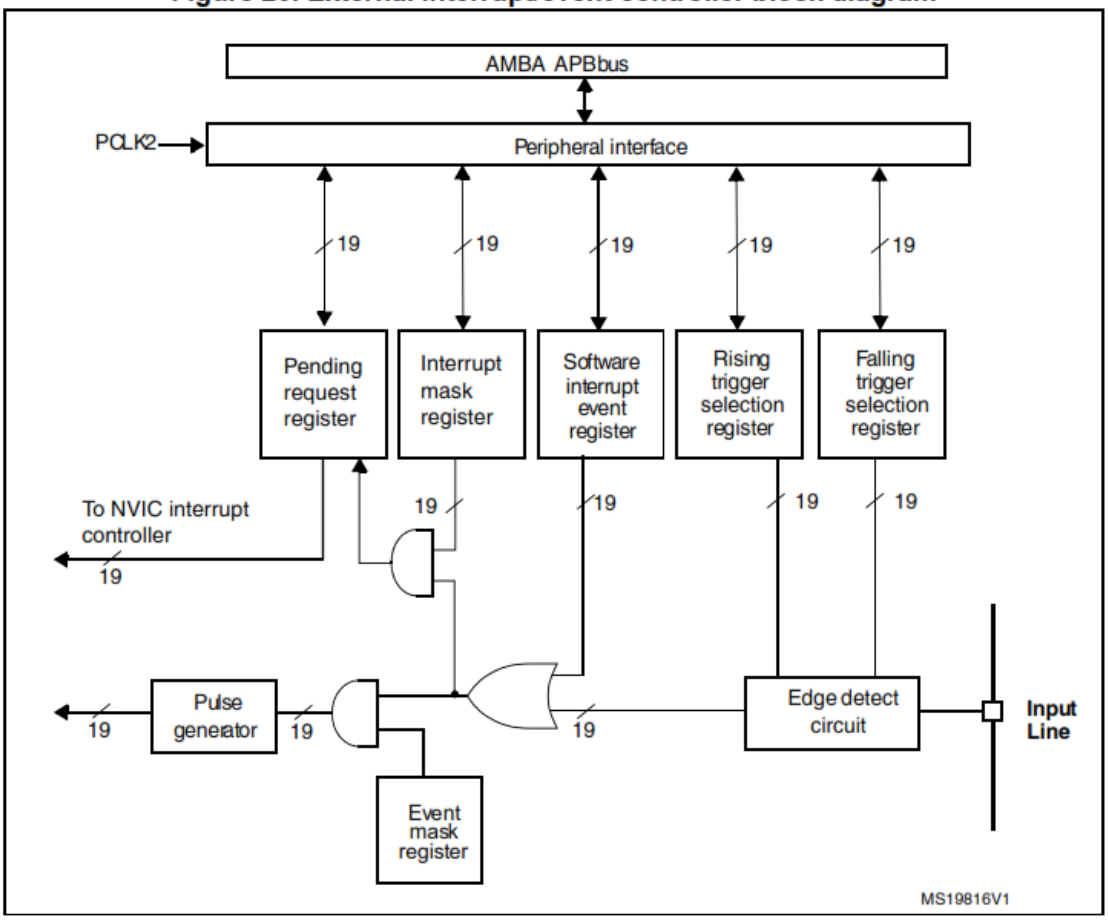

D:\HALA_Embedded\Teaching\Bài 8_Interrupt\Examples\test.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Project: test
Target 1
Source Group 1
CMSIS
Device
test.map
main.c*
stm32f10x_gpio.c
stm32f10x_exti.h
stm32f10x.h
startup_stm32f10x_md.s
core_cm3.h
stm32f10x_exti.c
stm32f10x_tim.c
__use_two_region_memory 0x080001c9 Thumb Code 2 heapauxi.o(.text)
__rt_heap_escrow$2region 0x080001cb Thumb Code 2 heapauxi.o(.text)
__rt_heap_expand$2region 0x080001cd Thumb Code 2 heapauxi.o(.text)
__user_setup_stackheap 0x080001cf Thumb Code 74 sys_stackheap_outer.o(.text)
exit 0x08000219 Thumb Code 18 exit.o(.text)
__user_libspace 0x0800022d Thumb Code 8 libspace.o(.text)
__user_perproc_libspace 0x0800022d Thumb Code 0 libspace.o(.text)
__user_perthread_libspace 0x0800022d Thumb Code 0 libspace.o(.text)
_sys_exit 0x08000235 Thumb Code 8 sys_exit.o(.text)
__Iusesemihosting 0x08000241 Thumb Code 0 use_no_semi.o(.text)
__use_no_semihosting_swi 0x08000241 Thumb Code 2 use_no_semi.o(.text)
__semihosting_library_function 0x08000243 Thumb Code 0 indicate_semi.o(.text)
EXTI0_IRQHandler 0x08000245 Thumb Code 28 main.o(i.EXTI0_IRQHandler)
EXTI_ClearITPendingBit 0x08000265 Thumb Code 6 stm32f10x_exti.o(i.EXTI_ClearITPendingBit)
EXTI_Config 0x08000271 Thumb Code 32 main.o(i.EXTI_Config)
EXTI_GetITStatus 0x08000291 Thumb Code 34 stm32f10x_exti.o(i.EXTI_GetITStatus)
EXTI_Init 0x080002b9 Thumb Code 142 stm32f10x_exti.o(i.EXTI_Init)
GPIO_Config 0x0800034d Thumb Code 118 main.o(i.GPIO_Config)
GPIO_EXTILineConfig 0x080003cd Thumb Code 60 stm32f10x_gpio.o(i.GPIO_EXTILineConfig)
GPIO_Init 0x0800040d Thumb Code 278 stm32f10x_gpio.o(i.GPIO_Init)
GPIO_ResetBits 0x08000523 Thumb Code 4 stm32f10x_gpio.o(i.GPIO_ResetBits)
GPIO_SetBits 0x08000527 Thumb Code 4 stm32f10x_gpio.o(i.GPIO_SetBits)
NVIC_Config 0x0800052b Thumb Code 68 main.o(i.NVIC_Config)
NVIC_Init 0x08000571 Thumb Code 100 misc.o(i.NVIC_Init)
NVIC_PriorityGroupConfig 0x080005e1 Thumb Code 10 misc.o(i.NVIC_PriorityGroupConfig)
RCC_APB1PeriphClockCmd 0x080005f5 Thumb Code 26 stm32f10x_rcc.o(i.RCC_APB1PeriphClockCmd)
Build Output
ST-Link Debugger
L:732 C:75
CAP NUM SCRL OVR R/W

1.NGẮT NGOÀI

External interrupt (EXTI) hay còn gọi là ngắt ngoài. Là 1 sự kiện ngắt xảy ra khi có tín hiệu can thiệp từ bên ngoài, từ phần cứng, người sử dụng hay ngoại vi. Chọn chân của GPIO để điều khiển việc ngắt.

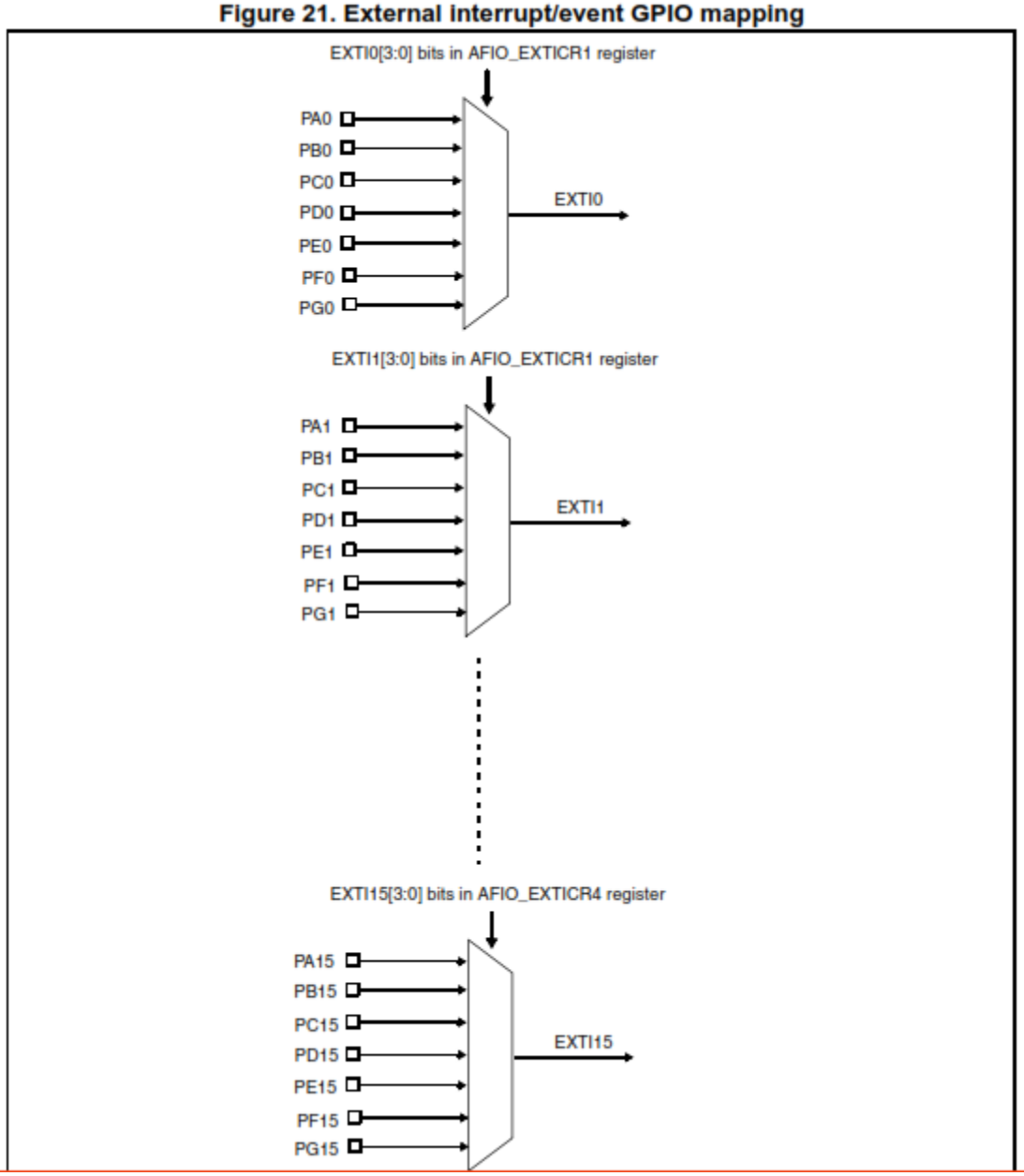
Sơ đồ khối của các khối điều khiển ngắt ngoài:

Figure 20. External interrupt/event controller block diagram



Ngắt ngoài của chip STM32F103 bao gồm có 16 line:

Figure 21. External interrupt/event GPIO mapping



Ở đây chúng ta có thể thấy chip STM32F103C8 gồm có 16 Line ngắt riêng biệt.

Ví dụ:

- Line0 sẽ chung cho tất cả chân Px0 ở tất cả các Port, với x là tên của Port A, B…
- Line0 nếu chúng ta đã chọn chân PA0 (chân 0 ở port A) làm chân ngắt thì tất cả các chân 0 ở các Port khác không được khai báo làm chân ngắt ngoài nữa
- Line1 nếu chúng ta chọn chân PB1 là chân ngắt thì tất cả chân 1 ở các Port khác không được khai báo làm chân ngắt nữa.

Tiếp theo các Line ngắt sẽ được phân vào các Vector ngắt tương ứng. Các Line ngắt của chip STM32F103 được phân bố vào các vector ngắt như sau:

6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000_0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068

23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_009C

40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00E0

Các Line0, Line1, Line2, Line3, Line4 sẽ được phân vào các vector ngắt riêng biệt EXTI0, EXTI1, EXTI2, EXTI3, EXTI4, còn từ Line5->Line9 sẽ được phân vào vector ngắt EXTI9_5, Line10->Line15 được phân vào vecotr EXTI15_10.

Bảng mức độ ưu tiên ngắt NVIC:

NVIC_PriorityGroup	NVIC_IRQChannelPreemptionPriority	NVIC_IRQChannelSubPriority	Description
NVIC_PriorityGroup_0	0	0-15	0 bits for pre-emption priority, 4 bits for subpriority
NVIC_PriorityGroup_1	0-1	0-7	1 bits for pre-emption priority, 3 bits for subpriority
NVIC_PriorityGroup_2	0-3	0-3	2 bits for pre-emption priority, 2 bits for subpriority
NVIC_PriorityGroup_3	0-7	0-1	3 bits for pre-emption priority, 1 bits for subpriority
NVIC_PriorityGroup_4	0-15	0	4 bits for pre-emption priority, 0 bits for subpriority

Có hai loại ưu tiên ngắt khác nhau trên MCU STM32F103C8T6 đó là Preemption Priorities và Sub Priorities. STM32 sử dụng 4 bit để định nghĩa cấp độ ngắt.

- Group 1: Phân bổ 1 bit cho preemption Priorities (chia được 2 cấp độ ngắt cho Pre 0-1), 3 bit cho Sub Priorities (chia được 8 cấp độ cho sub 0-7)
- Group 2: Phân bổ 2 bit cho preemption Priorities (chia được 4 cấp độ 0-3), 2 bit cho Sub Priorities (chia cấp độ ngắt 0-3)
- Tương tự cho các group 0-4.

– Mặc định thì ngắt nào có Preemtion Priority cao hơn thì sẽ được thực hiện trước.

– Khi nào 2 ngắt có cùng một mức Preemption Priority thì ngắt nào có Sub Priority cao hơn thì ngắt đó được thực hiện trước.

– Còn trường hợp 2 ngắt có cùng mức Preemption và Sub Priority luôn thì ngắt nào đến trước được thực hiện trước.

1.CẤU HÌNH NGẮT NGOÀI

1.1.Cấu hình GPIO:

Chân ngắt ngoài sẽ được cấu hình là Input. chế độ PullUp hay PullDown tùy thuộc vào cạnh ngắt. Ví dụ dưới dùng pin 0 là chân ngắt

```
void GPIO_Config(){
    GPIO_InitTypeDef GPIOInitStruct;
    GPIOInitStruct.GPIO_Mode = GPIO_Mode_IPU;// set pin 0 la Input pull up
    GPIOInitStruct.GPIO_Pin = GPIO_Pin_0;
    GPIOInitStruct.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIOInitStruct);
    GPIOInitStruct.GPIO_Mode = GPIO_Mode_Out_PP;// pin 13 la output push pull
    GPIOInitStruct.GPIO_Pin = GPIO_Pin_13;
    GPIOInitStruct.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOC, &GPIOInitStruct);
```

1.2.Cấu hình NVIC

```
void NVIC_Config(){
    NVIC_InitTypeDef NVICInitStruct; // delcare 1 struct NVIC
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2); // Chon group 2
    NVICInitStruct.NVIC_IRQChannel = EXTI0_IRQn; // chon line ngat EXTIO
        NVICInitStruct.NVIC_IRQChannelPreemptionPriority = 0x00; // Cap do uu
tien chinh = 0: khong uu tien
    NVICInitStruct.NVIC_IRQChannelSubPriority = 0x00;// Cap do uu tien phu = 0
    NVICInitStruct.NVIC_IRQChannelCmd = ENABLE;// Lenh cho ngat hoat dong
    NVIC_Init(&NVICInitStruct);}    // Nap cau hinh cho NVICInitStruct
```

Bộ NVIC cấu hình các tham số ngắt và quản lý các vecto ngắt.Các tham số được cấu hình trong NVIC_InitTypeDef, trong thư viện misc.h, bao gồm

```
  * @brief  NVIC Init Structure definition
  */

typedef struct
{
  uint8_t NVIC_IRQChannel;                    /*!< Specifies the IRQ channel to be enabled or disabled.
                                                   This parameter can be a value of @ref IRQn_Type
                                                   (For the complete STM32 Devices IRQ Channels list, please
                                                    refer to stm32f10x.h file) */

  uint8_t NVIC_IRQChannelPreemptionPriority;  /*!< Specifies the pre-emption priority for the IRQ channel
                                                   specified in NVIC_IRQChannel. This parameter can be a value
                                                   between 0 and 15 as described in the table @ref NVIC_Priority_Table */

  uint8_t NVIC_IRQChannelSubPriority;         /*!< Specifies the subpriority level for the IRQ channel specified
                                                   in NVIC_IRQChannel. This parameter can be a value
                                                   between 0 and 15 as described in the table @ref NVIC_Priority_Table */

  FunctionalState NVIC_IRQChannelCmd;         /*!< Specifies whether the IRQ channel defined in NVIC_IRQChannel
                                                   will be enabled or disabled.
                                                   This parameter can be set either to ENABLE or DISABLE */
} NVIC_InitTypeDef;
```

- NVIC_PriorityGroupConfig : Đầu tiên chọn group làm việc theo bảng dưới

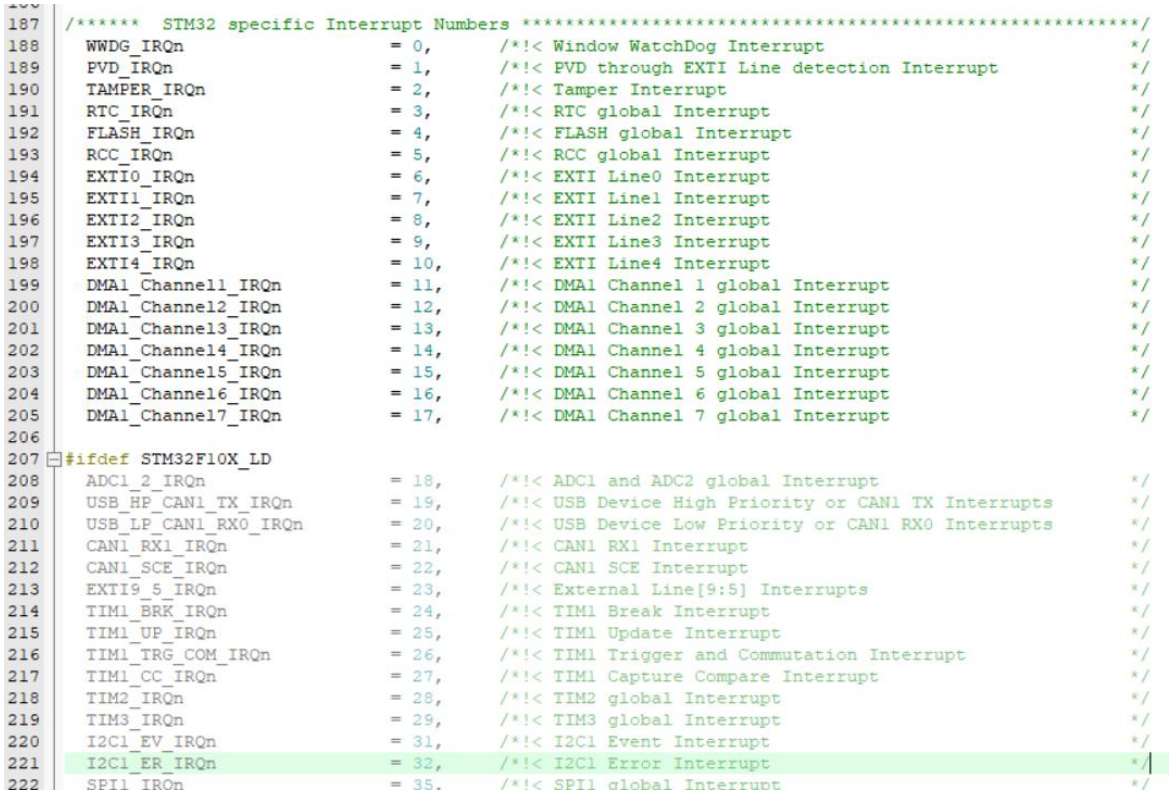
```
  void NVIC_PriorityGroupConfig(uint32_t NVIC_PriorityGroup)
{
  /* Check the parameters */
  assert_param(IS_NVIC_PRIORITY_GROUP(NVIC_PriorityGroup));

  /* 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
```

```
The table below gives the allowed values of the pre-emption priority and subpriority according
to the Priority Grouping configuration performed by NVIC_PriorityGroupConfig function
 ============================================================================================================
   NVIC_PriorityGroup   | NVIC_IRQChannelPreemptionPriority | NVIC_IRQChannelSubPriority  | Description
 ============================================================================================================
  NVIC_PriorityGroup_0  |                0                 |            0-15             |   0 bits for pre-emption priority
                        |                                  |                             |   4 bits for subpriority
 ------------------------------------------------------------------------------------------------------------
  NVIC_PriorityGroup_1  |              0-1                 |             0-7             |   1 bits for pre-emption priority
                        |                                  |                             |   3 bits for subpriority
 ------------------------------------------------------------------------------------------------------------
  NVIC_PriorityGroup_2  |              0-3                 |             0-3             |   2 bits for pre-emption priority
                        |                                  |                             |   2 bits for subpriority
 ------------------------------------------------------------------------------------------------------------
  NVIC_PriorityGroup_3  |              0-7                 |             0-1             |   3 bits for pre-emption priority
                        |                                  |                             |   1 bits for subpriority
 ------------------------------------------------------------------------------------------------------------
  NVIC_PriorityGroup_4  |             0-15                 |              0              |   4 bits for pre-emption priority
                        |                                  |                             |   0 bits for subpriority
 ============================================================================================================
```

- NVIC_IRQChannel: Cấu hình kênh Line ngắt, Enable line ngắt tương ứng với ngắt sử dụng

Các kênh ngắt định nghĩa trong thư viện #include "stm32f10x.h"

Chọn kênh tương ứng , ví dụ kênh EXTI0 = 6



```
/******  STM32 specific Interrupt Numbers **********************************************************/
  WWDG_IRQn                   = 0,      /*!< Window WatchDog Interrupt                                */
  PVD_IRQn                    = 1,      /*!< PVD through EXTI Line detection Interrupt                */
  TAMPER_IRQn                 = 2,      /*!< Tamper Interrupt                                         */
  RTC_IRQn                    = 3,      /*!< RTC global Interrupt                                     */
  FLASH_IRQn                  = 4,      /*!< FLASH global Interrupt                                   */
  RCC_IRQn                    = 5,      /*!< RCC global Interrupt                                     */
  EXTI0_IRQn                  = 6,      /*!< EXTI Line0 Interrupt                                     */
  EXTI1_IRQn                  = 7,      /*!< EXTI Line1 Interrupt                                     */
  EXTI2_IRQn                  = 8,      /*!< EXTI Line2 Interrupt                                     */
  EXTI3_IRQn                  = 9,      /*!< EXTI Line3 Interrupt                                     */
  EXTI4_IRQn                  = 10,     /*!< EXTI Line4 Interrupt                                     */
  DMA1_Channel1_IRQn          = 11,     /*!< DMA1 Channel 1 global Interrupt                          */
  DMA1_Channel2_IRQn          = 12,     /*!< DMA1 Channel 2 global Interrupt                          */
  DMA1_Channel3_IRQn          = 13,     /*!< DMA1 Channel 3 global Interrupt                          */
  DMA1_Channel4_IRQn          = 14,     /*!< DMA1 Channel 4 global Interrupt                          */
  DMA1_Channel5_IRQn          = 15,     /*!< DMA1 Channel 5 global Interrupt                          */
  DMA1_Channel6_IRQn          = 16,     /*!< DMA1 Channel 6 global Interrupt                          */
  DMA1_Channel7_IRQn          = 17,     /*!< DMA1 Channel 7 global Interrupt                          */

#ifdef STM32F10X_LD
  ADC1_2_IRQn                 = 18,     /*!< ADC1 and ADC2 global Interrupt                           */
  USB_HP_CAN1_TX_IRQn         = 19,     /*!< USB Device High Priority or CAN1 TX Interrupts           */
  USB_LP_CAN1_RX0_IRQn        = 20,     /*!< USB Device Low Priority or CAN1 RX0 Interrupts           */
  CAN1_RX1_IRQn               = 21,     /*!< CAN1 RX1 Interrupt                                       */
  CAN1_SCE_IRQn               = 22,     /*!< CAN1 SCE Interrupt                                       */
  EXTI9_5_IRQn                = 23,     /*!< External Line[9:5] Interrupts                            */
  TIM1_BRK_IRQn               = 24,     /*!< TIM1 Break Interrupt                                     */
  TIM1_UP_IRQn                = 25,     /*!< TIM1 Update Interrupt                                    */
  TIM1_TRG_COM_IRQn           = 26,     /*!< TIM1 Trigger and Commutation Interrupt                   */
  TIM1_CC_IRQn                = 27,     /*!< TIM1 Capture Compare Interrupt                           */
  TIM2_IRQn                   = 28,     /*!< TIM2 global Interrupt                                    */
  TIM3_IRQn                   = 29,     /*!< TIM3 global Interrupt                                    */
  I2C1_EV_IRQn                = 31,     /*!< I2C1 Event Interrupt                                     */
  I2C1_ER_IRQn                = 32,     /*!< I2C1 Error Interrupt                                     */
  SPI1_IRQn                   = 35,     /*!< SPI1 global Interrupt                                    */
```

- NVIC_IRQChannelPreemptionPriority: Chọn cấp độ ưu tiên chính của ngắt
- NVIC_IRQChannelSubPriority: Cấu hình độ ưu tiên phụ.
- NVIC_IRQChannelCmd: Cho phép ngắt.

Ngoài ra, NVIC_PriorityGroupConfig(); cấu hình các bit dành cho ChannelPreemptionPriority và ChannelSubPriority:

- NVIC_PriorityGroup_0: 0 bits for pre-emption priority 4 bits for subpriority
- NVIC_PriorityGroup_1: 1 bits for pre-emption priority 3 bits for subpriority
- NVIC_PriorityGroup_2: 2 bits for pre-emption priority 2 bits for subpriority
- NVIC_PriorityGroup_3: 3 bits for pre-emption priority 1 bits for subpriority
- NVIC_PriorityGroup_4: 4 bits for pre-emption priority 0 bits for subpriority

1.3.Cấu hình ngắt ngoài EXTI (chon nguồn ngắt, Line ngắt, Sườn xung kích hoạt ngắt)

```
void EXTI_Config(){
    EXTI_InitTypeDef EXTIInitStruct;
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource0); // chon GPIOA, Chan PA0 lam nguon ngat
    EXTIInitStruct.EXTI_Line = EXTI_Line0;  // do dung PA0 nen dung chuong trinh ngat ngoai Line0
    EXTIInitStruct.EXTI_Mode = EXTI_Mode_Interrupt;
    EXTIInitStruct.EXTI_Trigger = EXTI_Trigger_Falling; // ngat kich hoat luc bo nut nhan
    EXTIInitStruct.EXTI_LineCmd = ENABLE;
    EXTI_Init(&EXTIInitStruct);
    }
```

Đầu tiên, để sử dụng GPIO như ngắt ngoài, cần cấp clock cho ngoại vi AFIO.

RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);

Do ngắt ngoài kích hoạt từ chân Port của GPIO nên cần cấu hình chân thực hiện ngắt ngoài.

Sử dụng hàm sau trong thư viện gpio.h

GPIO_EXTILineConfig(uint8_t GPIO_PortSource, uint8_t GPIO_PinSource) cấu hình chân ở chế độ sử dụng ngắt ngoài:

- GPIO_PortSource: Chọn Port để sử dụng làm nguồn cho ngắt ngoài.
- GPIO_PinSource: Chọn Pin để cấu hình.

```
608  void GPIO_EXTILineConfig(uint8_t GPIO_PortSource, uint8_t GPIO_PinSource)
609  {
610    uint32_t tmp = 0x00;
611    /* Check the parameters */
612    assert_param(IS_GPIO_EXTI_PORT_SOURCE(GPIO_PortSource));
613    assert_param(IS_GPIO_PIN_SOURCE(GPIO_PinSource));
614
615    tmp = ((uint32_t)0x0F) << (0x04 * (GPIO_PinSource & (uint8_t)0x03));
616    AFIO->EXTICR[GPIO_PinSource >> 0x02] &= ~tmp;
617    AFIO->EXTICR[GPIO_PinSource >> 0x02] |= (((uint32_t)GPIO_PortSource) << (0x04 * (GPIO_PinSource & (uint8_t)0x03)));
618  }
619
620  /**
621    * @brief  Selects the Ethernet media interface.
622    * @note   This function applies only to STM32 Connectivity line devices.
623    * @param  GPIO_ETH_MediaInterface: specifies the Media Interface mode.
624    *   This parameter can be one of the following values:
625    *     @arg GPIO_ETH_MediaInterface_MII: MII mode
626    *     @arg GPIO_ETH_MediaInterface_RMII: RMII mode
627    * @retval None
628    */
```

```
249  /** @defgroup GPIO_Port_Sources
250    * @{
251    */
252
253  #define GPIO_PortSourceGPIOA       ((uint8_t)0x00)
254  #define GPIO_PortSourceGPIOB       ((uint8_t)0x01)
255  #define GPIO_PortSourceGPIOC       ((uint8_t)0x02)
256  #define GPIO_PortSourceGPIOD       ((uint8_t)0x03)
257  #define GPIO_PortSourceGPIOE       ((uint8_t)0x04)
258  #define GPIO_PortSourceGPIOF       ((uint8_t)0x05)
259  #define GPIO_PortSourceGPIOG       ((uint8_t)0x06)
260  #define IS_GPIO_EVENTOUT_PORT_SOURCE(PORTSOURCE) (((PORTSOURCE) == GPIO_PortSourceGPIOA) || \
261                                                    ((PORTSOURCE) == GPIO_PortSourceGPIOB) || \
262                                                    ((PORTSOURCE) == GPIO_PortSourceGPIOC) || \
263                                                    ((PORTSOURCE) == GPIO_PortSourceGPIOD) || \
264                                                    ((PORTSOURCE) == GPIO_PortSourceGPIOE))
```

```
/** @defgroup GPIO_Pin_sources
  * @{
  */

#define GPIO_PinSource0             ((uint8_t)0x00)
#define GPIO_PinSource1             ((uint8_t)0x01)
#define GPIO_PinSource2             ((uint8_t)0x02)
#define GPIO_PinSource3             ((uint8_t)0x03)
#define GPIO_PinSource4             ((uint8_t)0x04)
#define GPIO_PinSource5             ((uint8_t)0x05)
#define GPIO_PinSource6             ((uint8_t)0x06)
#define GPIO_PinSource7             ((uint8_t)0x07)
#define GPIO_PinSource8             ((uint8_t)0x08)
#define GPIO_PinSource9             ((uint8_t)0x09)
#define GPIO_PinSource10            ((uint8_t)0x0A)
#define GPIO_PinSource11            ((uint8_t)0x0B)
#define GPIO_PinSource12            ((uint8_t)0x0C)
#define GPIO_PinSource13            ((uint8_t)0x0D)
#define GPIO_PinSource14            ((uint8_t)0x0E)
#define GPIO_PinSource15            ((uint8_t)0x0F)
```

Tạo Struct ngắt ngoài.Cac tham số ngắt ngoài được cấu hình trong Struct EXTI_InitTypeDef, gồm:

```
typedef struct
{
  uint32_t EXTI_Line;               /*!< Specifies the EXTI lines to be enabled or disabled.
                                        This parameter can be any combination of @ref EXTI_Lines */

  EXTIMode_TypeDef EXTI_Mode;       /*!< Specifies the mode for the EXTI lines.
                                        This parameter can be a value of @ref EXTIMode_TypeDef */

  EXTITrigger_TypeDef EXTI_Trigger; /*!< Specifies the trigger signal active edge for the EXTI lines.
                                        This parameter can be a value of @ref EXTITrigger_TypeDef */

  FunctionalState EXTI_LineCmd;     /*!< Specifies the new state of the selected EXTI lines.
                                        This parameter can be set either to ENABLE or DISABLE */
}EXTI_InitTypeDef;
```

- EXTI_Line: Chọn line ngắt.

```
#define EXTI_Line0       ((uint32_t)0x00001)  /*!< External interrupt line 0 */
#define EXTI_Line1       ((uint32_t)0x00002)  /*!< External interrupt line 1 */
#define EXTI_Line2       ((uint32_t)0x00004)  /*!< External interrupt line 2 */
#define EXTI_Line3       ((uint32_t)0x00008)  /*!< External interrupt line 3 */
#define EXTI_Line4       ((uint32_t)0x00010)  /*!< External interrupt line 4 */
#define EXTI_Line5       ((uint32_t)0x00020)  /*!< External interrupt line 5 */
#define EXTI_Line6       ((uint32_t)0x00040)  /*!< External interrupt line 6 */
#define EXTI_Line7       ((uint32_t)0x00080)  /*!< External interrupt line 7 */
#define EXTI_Line8       ((uint32_t)0x00100)  /*!< External interrupt line 8 */
#define EXTI_Line9       ((uint32_t)0x00200)  /*!< External interrupt line 9 */
#define EXTI_Line10      ((uint32_t)0x00400)  /*!< External interrupt line 10 */
#define EXTI_Line11      ((uint32_t)0x00800)  /*!< External interrupt line 11 */
#define EXTI_Line12      ((uint32_t)0x01000)  /*!< External interrupt line 12 */
#define EXTI_Line13      ((uint32_t)0x02000)  /*!< External interrupt line 13 */
#define EXTI_Line14      ((uint32_t)0x04000)  /*!< External interrupt line 14 */
#define EXTI_Line15      ((uint32_t)0x08000)  /*!< External interrupt line 15 */
#define EXTI_Line16      ((uint32_t)0x10000)  /*!< External interrupt line 16 Connected to the PVD Output */
#define EXTI_Line17      ((uint32_t)0x20000)  /*!< External interrupt line 17 Connected to the RTC Alarm event */
#define EXTI_Line18      ((uint32_t)0x40000)  /*!< External interrupt line 18 Connected to the USB Device/USB OTG FS
                                                  Wakeup from suspend event */
#define EXTI_Line19      ((uint32_t)0x80000)  /*!< External interrupt line 19 Connected to the Ethernet Wakeup event */

#define IS_EXTI_LINE(LINE) ((((LINE) & (uint32_t)0xFFF00000) == 0x00) && ((LINE) != (uint16_t)0x00))
#define IS_GET_EXTI_LINE(LINE) (((LINE) == EXTI_Line0) || ((LINE) == EXTI_Line1) || \
                                ((LINE) == EXTI_Line2) || ((LINE) == EXTI_Line3) || \
                                ((LINE) == EXTI_Line4) || ((LINE) == EXTI_Line5) || \
                                ((LINE) == EXTI_Line6) || ((LINE) == EXTI_Line7) || \
                                ((LINE) == EXTI_Line8) || ((LINE) == EXTI_Line9) || \
                                ((LINE) == EXTI_Line10) || ((LINE) == EXTI_Line11) || \
                                ((LINE) == EXTI_Line12) || ((LINE) == EXTI_Line13) || \
                                ((LINE) == EXTI_Line14) || ((LINE) == EXTI_Line15) || \
                                ((LINE) == EXTI_Line16) || ((LINE) == EXTI_Line17) || \
                                ((LINE) == EXTI_Line18) || ((LINE) == EXTI_Line19))
```

- EXTI_Mode: Chọn Mode cho ngắt là Interupt hay Even.

Dùng mode event khi bạn lập trình multi-task, đa luồng. còn bình thường làm lập trình embed chủ yếu dùn interrupt.

```
49  typedef enum
50  {
51    EXTI_Mode_Interrupt = 0x00,
52    EXTI_Mode_Event = 0x04
53  }EXTIMode_TypeDef;
54
55  #define IS_EXTI_MODE(MODE) (((MODE) == EXTI_Mode_Interrupt) || ((MODE) == EXTI_Mode_Event))
```

- EXTI_Trigger: Cấu hình cạnh ngắt.

```
61  typedef enum
62  {
63    EXTI_Trigger_Rising = 0x08,
64    EXTI_Trigger_Falling = 0x0C,
65    EXTI_Trigger_Rising_Falling = 0x10
66  }EXTITrigger_TypeDef;
67
68  #define IS_EXTI_TRIGGER(TRIGGER) (((TRIGGER) == EXTI_Trigger_Rising) || \
69                                    ((TRIGGER) == EXTI_Trigger_Falling) || \
70                                    ((TRIGGER) == EXTI_Trigger_Rising_Falling))
```

- EXTI_LineCmd: Cho phép ngắt ở Line đã cấu hình.

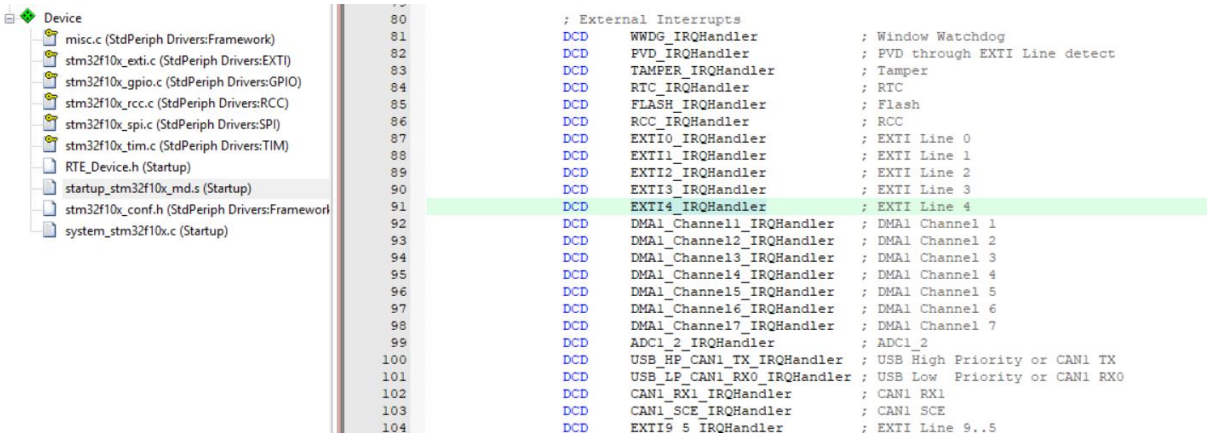
1.4 Hàm phục vụ ngắt ngoài (chương trình ngắt)

```
void EXTI0_IRQHandler(void){
    if (EXTI_GetITStatus(EXTI_Line0) != RESET) { //kiem tra flag ngat, neu
dung Line0 moi thuc hien lenh
    GPIOC->ODR ^= GPIO_Pin_13;
    EXTI_ClearITPendingBit(EXTI_Line0);
    }
    }
```

Trong hàm phục vụ ngắt ngoài, chúng ta sẽ thực hiện:

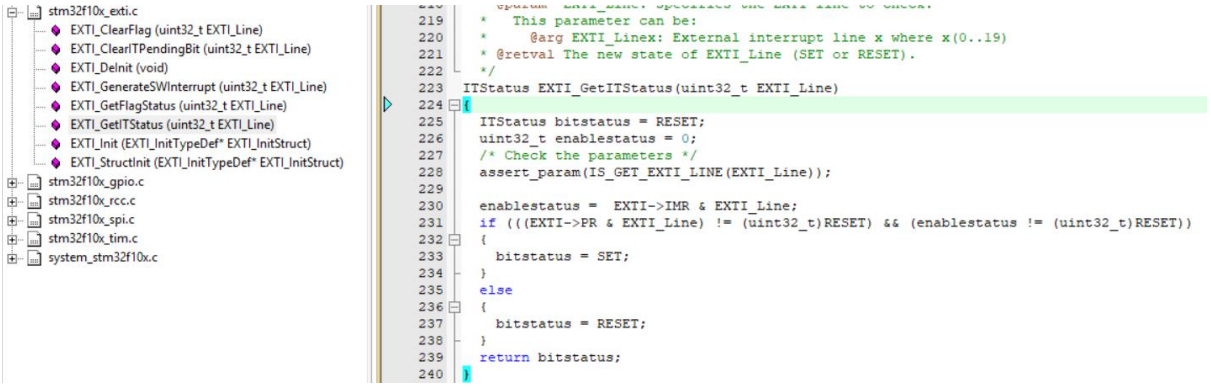
- Tìm tên hàm trong bảng vector table trong file startup_stm32f10x_md.s(Startup) tương ứng với Line trong EXTI_Config() ở trên.
- Kiểm tra ngắt đến từ line nào, có đúng là line cần thực thi hay không?
- Thực hiện các lệnh, các hàm.
- Xóa cờ ngắt ở line.

Ngắt trên từng line có hàm phục riêng của từng line. Được đăng kí và có tên cố định. Bởi vậy phải tìm đúng tên hàm ngắt tương ứng line đó để viết chương trình. Tìm kiếm tên này trong file startup_stm32f10x_md.s(Startup)

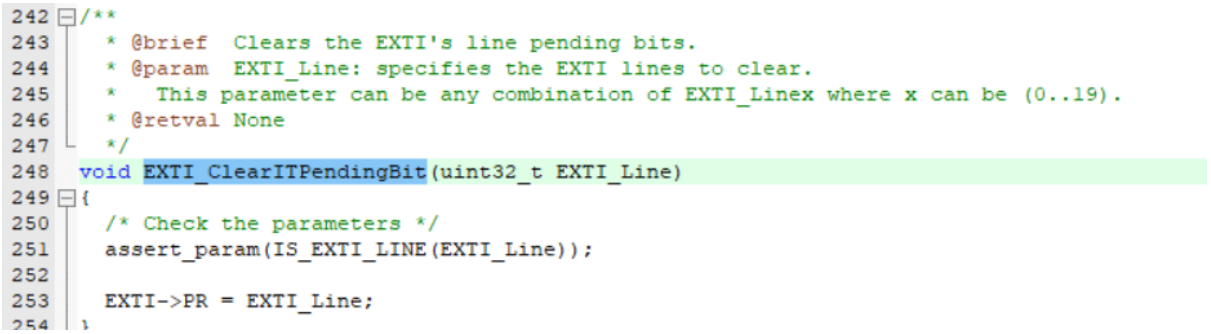


EXTIx_IRQHandler() (x là line ngắt tương ứng). Hàm này sẽ được gọi khi có ngắt tương ứng trên Line xảy ra.

Hàm EXTI_GetITStatus(EXTI_Linex)(x là Line ngắt): Kiểm tra cờ ngắt của line x tương ứng. Nếu chính xác Ngắt từ line x mới thực hiện các lệnh tiếp theo.



Hàm EXTI_ClearITPendingBit(EXTI_Linex): Xóa cờ ngắt ở line x.

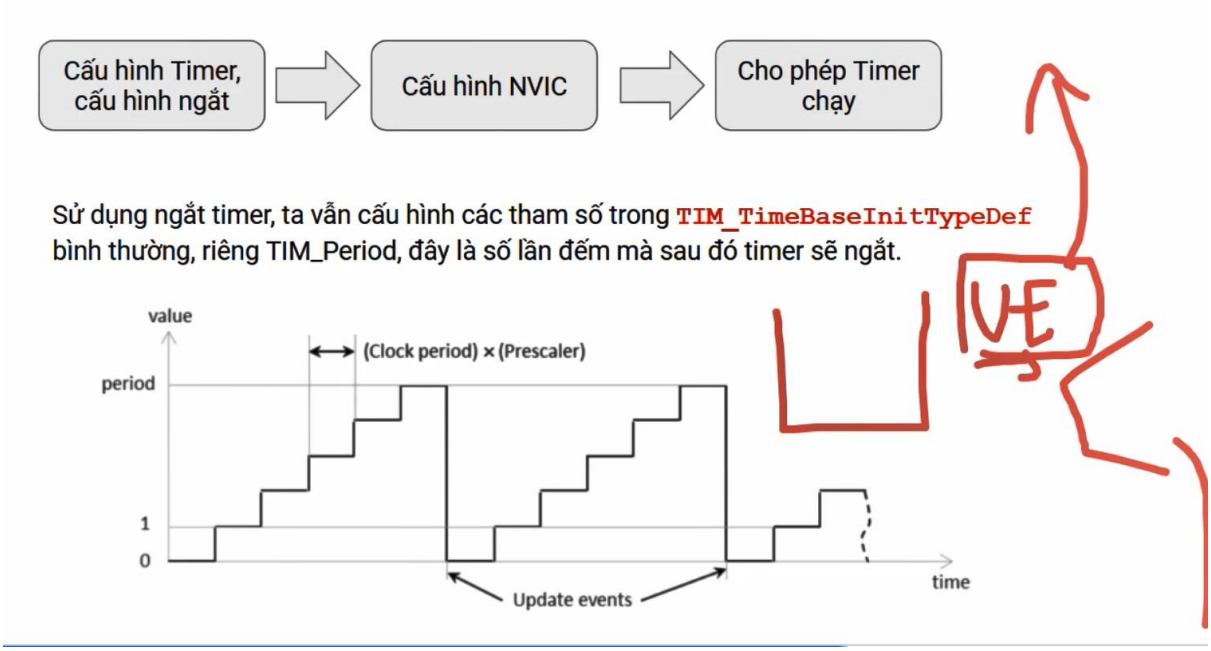


2.NGẮT TIMER

Sơ lược cấu hình ngắt Timer

Cấu hình Timer:

Sử dụng ngắt timer, ta vẫn cấu hình các tham số trong TIM_TimeBaseInitTypeDef bình thường, riêng TIM_Period, đây là số chu kì mà timer sẽ ngắt. Ta tính toán và đặt giá trị để tạo khoảng thời gian ngắt mong muốn. Khi timer đếm hết period gây ra tràn bộ đệm thì Update Event sẽ xảy ra, Timer bắt đầu đếm lại từ đầu.



2.1. Hàm kích hoạt ngắt cho TIMERx

Hàm TIM_ITConfig(TIMx, TIM_IT_Update, ENABLE)

Truyền vào tham số (Timer sử dụng, nguồn ngắt, enable-kích hoạt)

Nguồn ngắt: Interrupt Source như:

TIM_IT_Update : là Update Event.

TIM_IT_CC1..: So sánh giá trị

```
/**
  * @brief  Enables or disables the specified TIM interrupts.
  * @param  TIMx: where x can be 1 to 17 to select the TIMx peripheral.
  * @param  TIM_IT: specifies the TIM interrupts sources to be enabled or disabled.
  *   This parameter can be any combination of the following values:
  *     @arg TIM_IT_Update: TIM update Interrupt source
  *     @arg TIM_IT_CC1: TIM Capture Compare 1 Interrupt source
  *     @arg TIM_IT_CC2: TIM Capture Compare 2 Interrupt source
  *     @arg TIM_IT_CC3: TIM Capture Compare 3 Interrupt source
  *     @arg TIM_IT_CC4: TIM Capture Compare 4 Interrupt source
  *     @arg TIM_IT_COM: TIM Commutation Interrupt source
  *     @arg TIM_IT_Trigger: TIM Trigger Interrupt source
  *     @arg TIM_IT_Break: TIM Break Interrupt source
  * @note
  *   - TIM6 and TIM7 can only generate an update interrupt.
  *   - TIM9, TIM12 and TIM15 can have only TIM_IT_Update, TIM_IT_CC1,
  *      TIM_IT_CC2 or TIM_IT_Trigger.
  *   - TIM10, TIM11, TIM13, TIM14, TIM16 and TIM17 can have TIM_IT_Update or TIM_IT_CC1.
  *   - TIM_IT_Break is used only with TIM1, TIM8 and TIM15.
  *   - TIM_IT_COM is used only with TIM1, TIM8, TIM15, TIM16 and TIM17.
  * @param  NewState: new state of the TIM interrupts.
  *   This parameter can be: ENABLE or DISABLE.
  * @retval None
  */
void TIM_ITConfig(TIM_TypeDef* TIMx, uint16_t TIM_IT, FunctionalState NewState)
{
  /* Check the parameters */
  assert_param(IS_TIM_ALL_PERIPH(TIMx));
  assert_param(IS_TIM_IT(TIM_IT));
  assert_param(IS_FUNCTIONAL_STATE(NewState));

  if (NewState != DISABLE)
  {
    /* Enable the Interrupt sources */
    TIMx->DIER |= TIM_IT;
  }
  else
  {
    /* Disable the Interrupt sources */
    TIMx->DIER &= (uint16_t)~TIM_IT;
  }
}
```

2.2.Cấu hình NVIC:

Ở NVIC, ta cấu hình tương tự như ngắt ngoài EXTI, tuy nhiên NVIC_IRQChannel được đổi thành TIM_IRQn để khớp với line ngắt timer.

```
NVICInitStruct.NVIC_IRQChannel = EXTI0_IRQn;
```

2.3.Hàm phục vụ ngắt TImer

Hàm phục vụ ngắt Timer được đặt tên : TIMx_IRQHandler() với x là timer tương ứng.

.

```
void TIM2_IRQHandler()
{
    if(TIM_GetITStatus(TIM2, TIM_IT_Update)){
        count++;
        TIM_ClearITPendingBit(TIM2, TIM_IT_Update);// Clears the TIM2
interrupt pending bit
}
}
```

Bên trong hàm ngắt, ta kiểm tra cờ TIM_IT_Update bằng hàm TIM_GetITStatus()
Hàm này trả về giá trị kiểm tra xem timer đã tràn hay chưa

```
2631  ITStatus TIM_GetITStatus(TIM_TypeDef* TIMx, uint16_t TIM_IT)
2632  {
2633    ITStatus bitstatus = RESET;
2634    uint16_t itstatus = 0x0, itenable = 0x0;
2635    /* Check the parameters */
2636    assert_param(IS_TIM_ALL_PERIPH(TIMx));
2637    assert_param(IS_TIM_GET_IT(TIM_IT));
2638
2639    itstatus = TIMx->SR & TIM_IT;
2640
2641    itenable = TIMx->DIER & TIM_IT;
2642    if ((itstatus != (uint16_t)RESET) && (itenable != (uint16_t)RESET))
2643    {
2644      bitstatus = SET;
2645    }
2646    else
2647    {
2648      bitstatus = RESET;
2649    }
2650    return bitstatus;
```

Sau khi thực hiện xong, gọi TIM_ClearITPendingBit(TIMx, TIM_IT_Update); để xóa cờ ngắt này.

```
2675  void TIM_ClearITPendingBit(TIM_TypeDef* TIMx, uint16_t TIM_IT)
2676  {
2677    /* Check the parameters */
2678    assert_param(IS_TIM_ALL_PERIPH(TIMx));
2679    assert_param(IS_TIM_IT(TIM_IT));
2680    /* Clear the IT pending Bit */
2681    TIMx->SR = (uint16_t)~TIM_IT;
2682  }
```

3.NGẮT TRUYỀN THÔNG

STM32F1 hỗ trợ các ngắt cho các giao thức truyền nhận như SPI, I2C, UART…

Ở bài này ta sẽ ví dụ với UART ngắt nhận, các giao thức còn lại cũng sẽ có cách cấu hình tương tự.

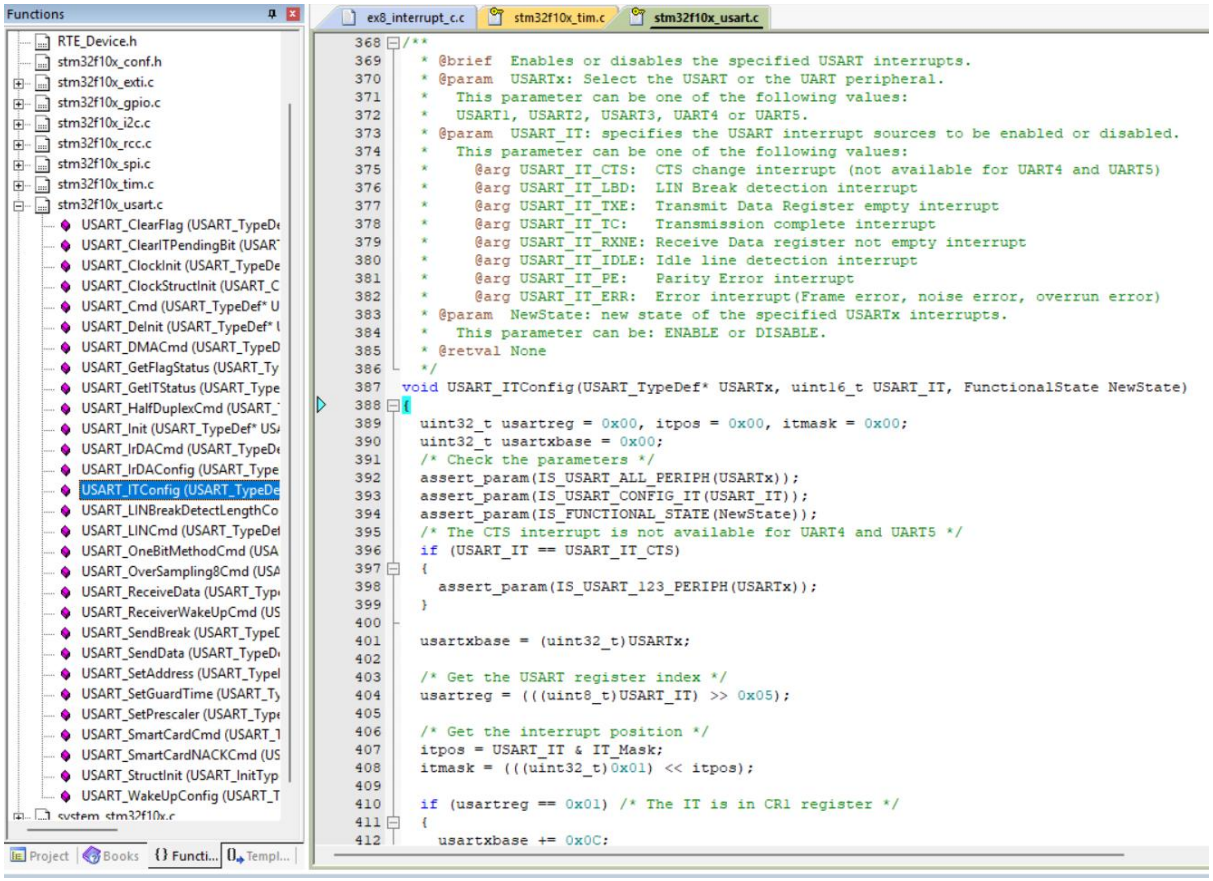
Đầu tiên, các cấu hình tham số cho UART thực hiện bình thường.

Cấu hình RCC->Cấu hình GPIO-> Cấu hình tham số UART.

```
void UART_Config(){
    USART_InitTypeDef UART_InitStruct;
    UART_InitStruct.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
    UART_InitStruct.USART_BaudRate = 9600;
    UART_InitStruct.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;
    UART_InitStruct.USART_WordLength = USART_WordLength_8b;
    UART_InitStruct.USART_StopBits = USART_StopBits_1;
    UART_InitStruct.USART_Parity = USART_Parity_No;

    USART_Init(USART1, &UART_InitStruct);
    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE); //Hàm cho phep ngat khi co
flag bao nhan data RXNE
    USART_Cmd(USART1, ENABLE);
}
```

3.1.Cấu hình USART cần cho thêm hàm kich hoạt chế độ phép ngắt UART bằng hàm USART_ITConfig(),Trước khi cho phép UART hoạt động; ham này gồm 3 tham số: USART_ITConfig(chọn Usart, Nguồn ngắt,enable),

- USART_TypeDef* USARTx: Bộ UART cần cấu hình.
- uint16_t USART_IT: Chọn nguồn ngắt UART.
- Có nhiều nguồn ngắt từ UART,bài này sử dụng cờ ngắt truyền (USART_IT_TXE) và ngắt nhận (USART_IT_RXNE).
- FunctionalState NewState: Cho phép ngắt.



3.2.Cấu hình NVIC.

```
NVICInitStruct.NVIC_IRQChannel = USART1_IRQn
```

Ở cấu hình NVIC cho UART cũng như các ngoại vi khác, cần chọn line ngắt tương ứng với bộ ngoại vi cần sử dụng. Ở đây chọn line UART là USARTx_IRQn với x là bộ Uart sử dụng.

Line ngắt định nghĩa trong thư viện stm32f10x.h

```
#ifdef STM32F10X_MD
  ADC1_2_IRQn                 = 18,     /*!< ADC1 and ADC2 global Interrupt                    */
  USB_HP_CAN1_TX_IRQn         = 19,     /*!< USB Device High Priority or CAN1 TX Interrupts    */
  USB_LP_CAN1_RX0_IRQn        = 20,     /*!< USB Device Low Priority or CAN1 RX0 Interrupts    */
  CAN1_RX1_IRQn               = 21,     /*!< CAN1 RX1 Interrupt                                */
  CAN1_SCE_IRQn               = 22,     /*!< CAN1 SCE Interrupt                                */
  EXTI9_5_IRQn                = 23,     /*!< External Line[9:5] Interrupts                     */
  TIM1_BRK_IRQn               = 24,     /*!< TIM1 Break Interrupt                              */
  TIM1_UP_IRQn                = 25,     /*!< TIM1 Update Interrupt                             */
  TIM1_TRG_COM_IRQn           = 26,     /*!< TIM1 Trigger and Commutation Interrupt            */
  TIM1_CC_IRQn                = 27,     /*!< TIM1 Capture Compare Interrupt                    */
  TIM2_IRQn                   = 28,     /*!< TIM2 global Interrupt                             */
  TIM3_IRQn                   = 29,     /*!< TIM3 global Interrupt                             */
  TIM4_IRQn                   = 30,     /*!< TIM4 global Interrupt                             */
  I2C1_EV_IRQn                = 31,     /*!< I2C1 Event Interrupt                              */
  I2C1_ER_IRQn                = 32,     /*!< I2C1 Error Interrupt                              */
  I2C2_EV_IRQn                = 33,     /*!< I2C2 Event Interrupt                              */
  I2C2_ER_IRQn                = 34,     /*!< I2C2 Error Interrupt                              */
  SPI1_IRQn                   = 35,     /*!< SPI1 global Interrupt                             */
  SPI2_IRQn                   = 36,     /*!< SPI2 global Interrupt                             */
  USART1_IRQn                 = 37,     /*!< USART1 global Interrupt                           */
  USART2_IRQn                 = 38,     /*!< USART2 global Interrupt                           */
  USART3_IRQn                 = 39,     /*!< USART3 global Interrupt                           */
  EXTI15_10_IRQn              = 40,     /*!< External Line[15:10] Interrupts                   */
  RTCAlarm_IRQn               = 41,     /*!< RTC Alarm through EXTI Line Interrupt             */
  USBWakeUp_IRQn              = 42      /*!< USB Device WakeUp from suspend through EXTI Line Interrupt */
#endif /* STM32F10X_MD */
```

1. Hàm phục vụ ngắt.

Hàm USARTx_IRQHandler() sẽ được gọi nếu xảy ra ngắt trên Line ngắt UART đã cấu hình. Trong hàm phục vụ ngắt, ta kiểm tra ngắt đến là ngắt nhận RXNE hay ngắt truyền TX, bnawgf hàm USART_GetITStatus. Tùy theo tín hiệu ngắt mà có thể lập trình tác vụ khác nhau. Sau khi thực thi xong, có thể xóa cờ ngắt để đảm bảo không còn ngắt trên line (thông thường cờ ngắt sẽ tự động xóa).

```
void USART1_IRQHandler()
{
    uint8_t data = 0x00;
    if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET){
        while(USART_GetFlagStatus(USART1, USART_FLAG_RXNE)); // Cho co du lieu
trong thanh ghi DR
        data = USART_ReceiveData(USART1);// Gan du lieu vao bien data
        if(USART_GetITStatus(USART1, USART_IT_TXE) == RESET){
            USART_SendData(USART1, data); // Tra du lieu ra man hinh
            while (USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET);
        }
    }
    USART_ClearITPendingBit (USART1, USART_IT_RXNE);
}
```