Ngắt ngoài và ưu tiên ngắt trên STM32F4

NVIC – Nested vectored interrupt controller là bộ điều khiển xử lý ngắt có trong MCU STM32F407VG. Việc lập trình sử dụng ngắt là một kĩ năng rất quan trọng đối với lập trình vi điều khiển. Nếu không có ngắt, chương trình của chúng ta sẽ thực hiện tuần tự từ trên xuống dưới, ngắt sẽ giúp chương trình xử lý theo sự việc, đáp ứng được các sự kiện như thay đổi mức logic từ 1 chân I/O (ngắt ngoài), nhận 1 ký tự (ngắt nhận UART),…

Trong bài viết này chúng ta cùng tìm hiểu về ngắt ngoài (**EXTI – External interrupt**) cùng với vi điều khiển STM32F407VG.

I, Lý thuyết

Một số tính năng của NVIC với STM32F407 :

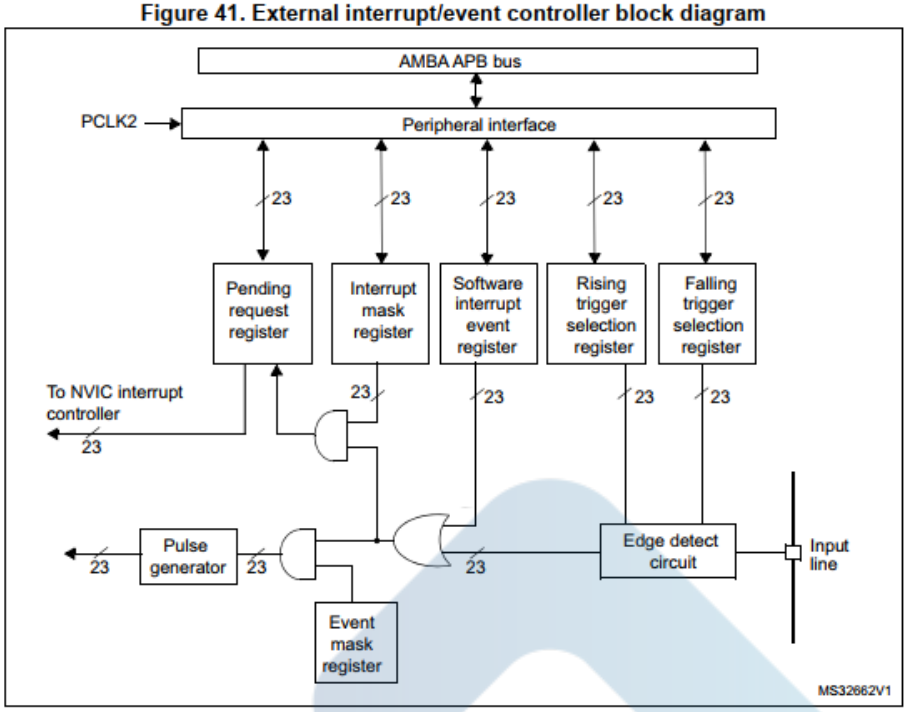
- 82 kênh ngắt
- 16 mức ưu tiên ngắt (có thể lập trình được)
- Quản lý, điều khiển năng lượng cho vector ngắt
- Thực hiện trên các thanh ghi điều khiển hệ thống
- Đỗ trễ thấp, xử lý ngắt cực kỳ nhanh

Một số tính năng của ngắt ngoài trên STM32F407:
Kích hoạt độc lập và mask cho mỗi line sự kiện/ngắt.

- Có bit trạng thái (status) riêng cho mỗi line ngắt
- Có thể có tối đa 23 sự kiện/ ngắt
- Kiểm tra tín hiệu ngoài có độ rộng xung nhỏ hơn clock trên APB2

Sơ đồ khối của các khối điều khiển EXTI

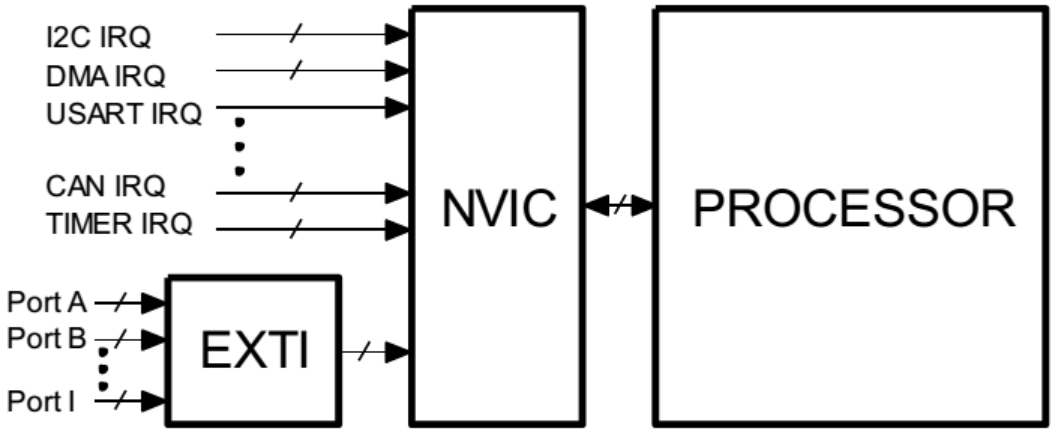
Figure 41. External interrupt/event controller block diagram



Cấu hình với thư viện chuẩn của ST. Có 2 loại ngắt ngoài chính đó là ngắt ngoài trên các chân điều khiển ở dạng thông thường và ngắt ngoài trên các ứng dụng như : PVD, RTC, USB, Ethernet.

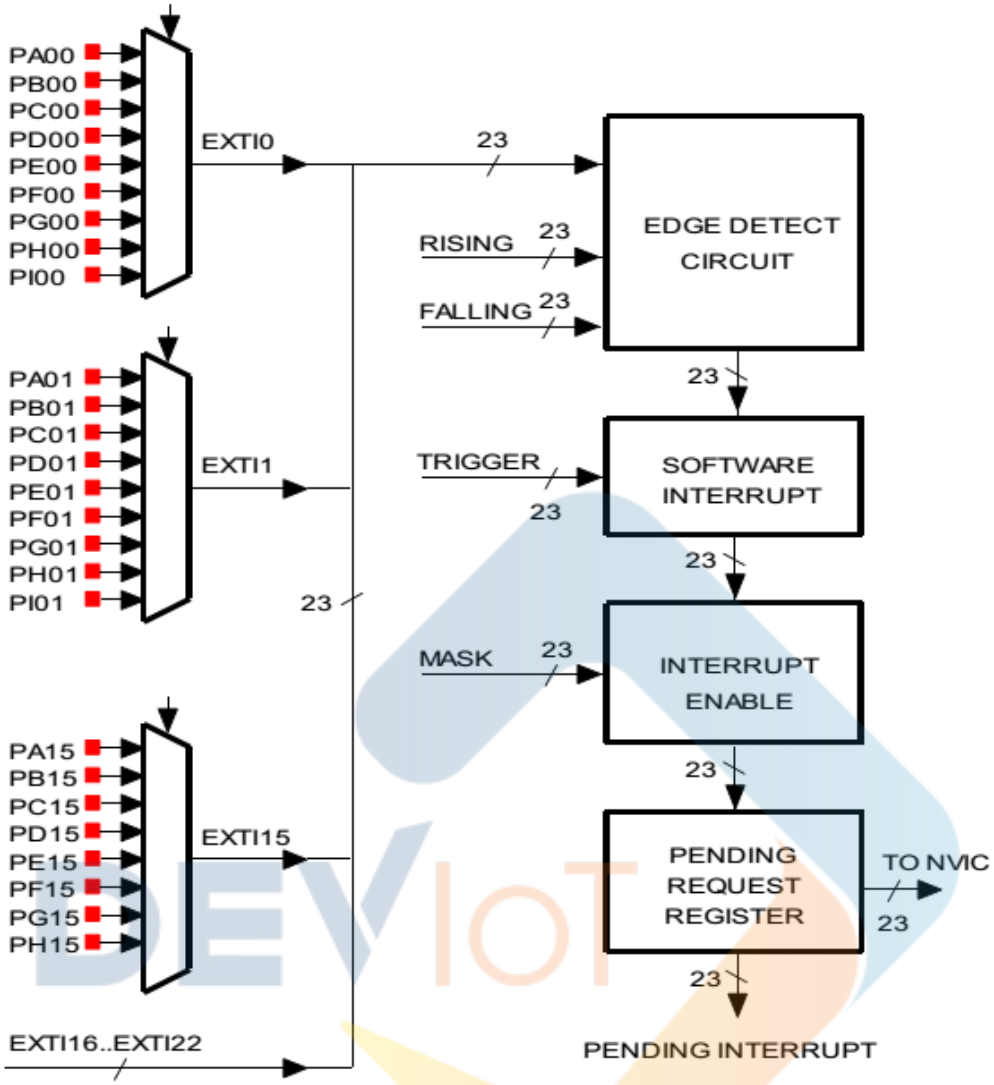
EXTI nằm trong 1 phần của NVIC, bao gồm 23 bộ phát hiện sự kiện, từ đó khởi tạo nên các yêu cầu ngắt. Mỗi đường đầu vào có thể được cấu hình độc lập để lựa chọn kiểu là interrupt hay event và trigger event tương ứng (rising, falling hoặc cả 2). Mỗi đường ngắt cũng có thể được che một cách độc lập.

EXTI được kết nối với bộ xử lý ngắt lồng nhau NVIC như sau:



Bộ điều khiển ngắt ngoại EXTI xử lý tất cả các tín hiệu yêu cầu ngắt đến từ tất cả các chân của vi điều khiển. Ngoài ra nó còn xử lý các yêu cầu ngắt đến từ các nguồn khác. Các yêu cầu ngắt được phân thành 23 đường ngắt khác nhau, trong đó các yêu cầu đến từ chân 0 của tất cả các port được xử lý trên line 0, các yêu cầu đến từ chân 1 của tất cả các port được xử lý trên line 1…

Các bạn theo dõi hình dưới đây để hiểu rõ hơn nhé!



7 đường ngắt EXTI còn lại được nối như sau:

- EXTI line 16 được nối vào PVD output
- EXTI line 17 được nối vào RTC Alarm event
- EXTI line 18 được nối vào USB OTG FS Wakeup event
- EXTI line 19 được nối vào Ethernet Wakeup event
- EXTI line 20 được nối vào USB OTG HS (configured in FS) Wakeup event
- EXTI line 21 được nối vào RTC Tamper and TimeStamp events
- EXTI line 22 được nối vào RTC Wakeup event

Một số thanh ghi quan trọng với EXTI:

EXTI_IMR – Interrupt mask register:

Thanh ghi này cài đặt cho phép có yêu cầu ngắt trên Line tương ứng. (cho phép ngắt)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									MR22	MR21	MR20	MR19	MR18	MR17	MR16
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **MRx:** Interrupt mask on line x
0: Interrupt request from line x is masked
1: Interrupt request from line x is not masked

EXTI_RTSR – Rising trigger selection register:
Thanh ghi này được sử dụng để cấu hình chọn sườn lên làm tín hiệu kích hoạt ngắt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									TR22	TR21	TR20	TR19	TR18	TR17	TR16
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **TRx:** Rising trigger event configuration bit of line x
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

EXTI_FTSR – Falling trigger selection register:
Thanh ghi này được sử dụng để cấu hình chọn sườn xuống làm tín hiệu kích hoạt ngắt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									TR22	TR21	TR20	TR19	TR18	TR17	TR16
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **TRx:** Falling trigger event configuration bit of line x
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line.

Note: The external wakeup lines are edge triggered, no glitch must be generated on these lines. If a falling edge occurs on the external interrupt line while writing to the EXTI_FTSR register, the pending bit is not set.

Rising and falling edge triggers can be set for the same interrupt line. In this configuration, both generate a trigger condition.

EXTI_SWIER – Software interrupt even register:

Thanh ghi này cho phép kích hoạt Line ngắt tương tứng bằng phần mềm.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIER 15	SWIER 14	SWIER 13	SWIER 12	SWIER 11	SWIER 10	SWIER 9	SWIER 8	SWIER 7	SWIER 6	SWIER 5	SWIER 4	SWIER 3	SWIER 2	SWIER 1	SWIER 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **SWIERx:** Software Interrupt on line x

If interrupt are enabled on line x in the EXTI_IMR register, writing '1' to SWIERx bit when it is set at '0' sets the corresponding pending bit in the EXTI_PR register, thus resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit in EXTI_PR (by writing a 1 to the bit).

EXTI_PR – Pending register:

Đây là thanh ghi chờ xử lý ngắt, khi có yêu cầu ngắt được tạo ra trên một Line ngắt thì bit tương tứng của thanh ghi này được bật lên cho đến khi ngắt này được xử lý. Nhiều trước hợp có sự thay đổi sườn tín hiệu tạo ra yêu cầu ngắt nhưng ngắt không được thực thi như: độ ưu tiên thấp, chưa cho phép ngắt toàn cục.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PR22	PR21	PR20	PR19	PR18	PR17	PR16
									rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **PRx:** Pending bit

0: No trigger request occurred

1: selected trigger request occurred

This bit is set when the selected edge event arrives on the external interrupt line.

This bit is cleared by programming it to '1'.

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 2 loại ưu tiên ngắt khác nhau, đó là **Preemption Priorities** và **Sub Priorities**:

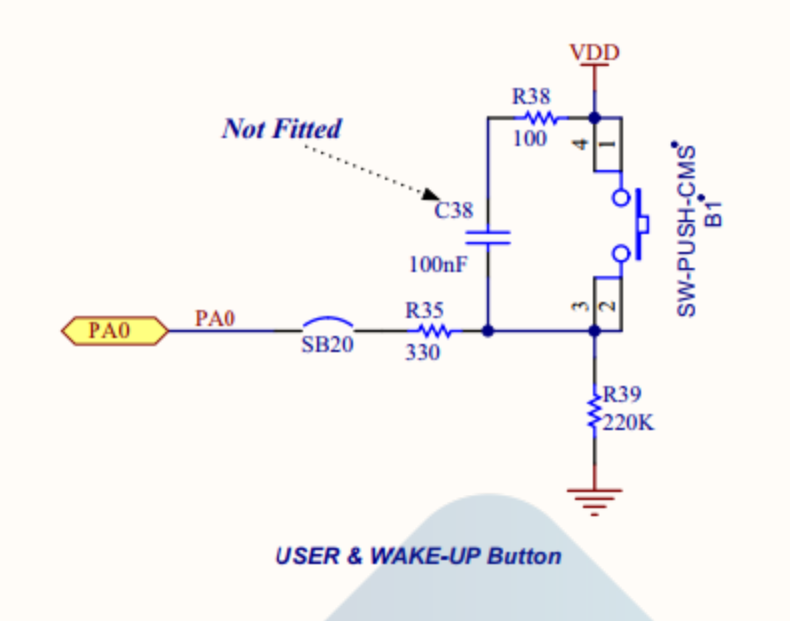
- Mặc định thì ngắt nào có **Preemtion Priority** cao hơn thì sẽ được thực hiện trước.
- Khi nào 2 ngắt có cùng một **mức Preemption Priority** thì ngắt nào có **Sub Priority** cao hơn thì ngắt đó được thực hiện trước.
- Còn trường hợp 2 ngắt có cùng mức **Preemption** và **Sub Priority** luôn thì ngắt nào đến trước được thực hiện trước.

Lưu ý: Ngắt có mức cao hơn thì có số thứ tự bé hơn.

II, Thực hành

Sau khi tìm hiểu lý thuyết về EXTI, chúng ta cùng thực hành 1 project sử dụng ngắt ngoài trên KIT STM32F407VG. Trên MCU này, nhà sản xuất đã kết nối sẵn cho chúng ta chân PA0 với User button (nút nhấn màu xanh trên KIT). Vì vậy chúng ta sẽ tận dụng nút nhấn này để thực hành.

Trước hết, chúng ta cùng tìm hiểu sơ qua về cấu trúc của User button thông qua hình dưới đây:



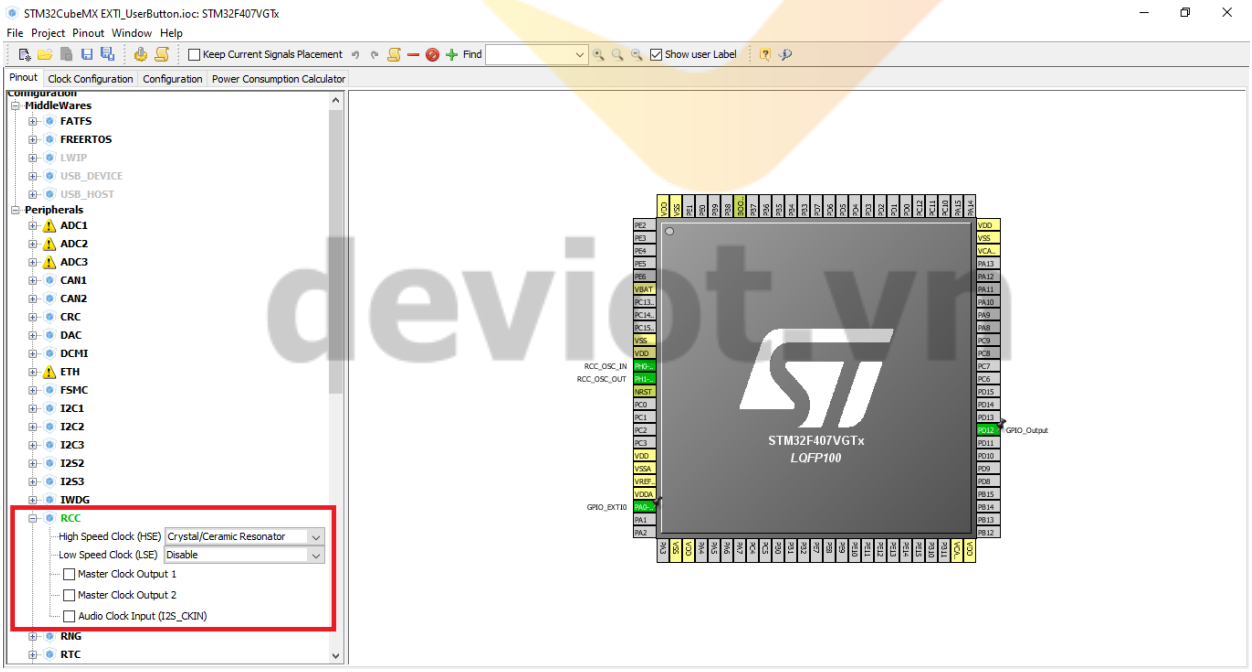
1, Khởi tạo Project với CubeMX

Khởi tạo **New Project** với CubeMX, chọn dòng chip chúng ta sử dụng.

Tiếp theo, chúng ta cấu hình thạch anh và xung Clock cho Chip:

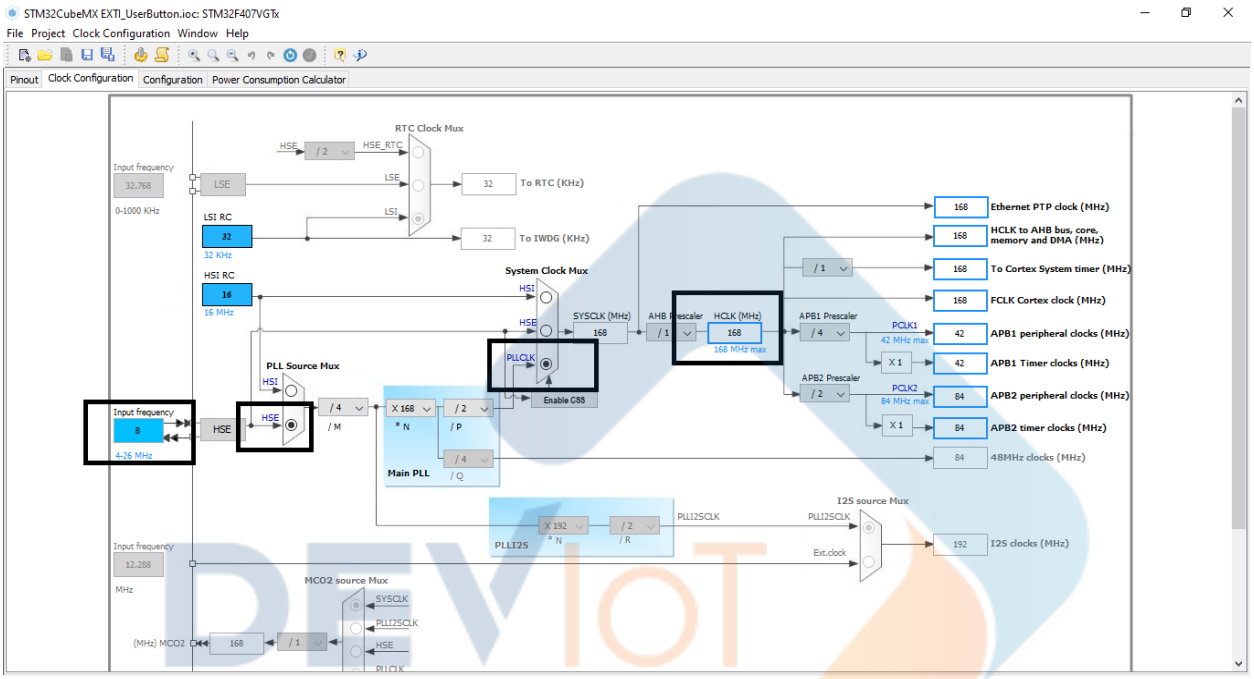
Tại mục **RCC → High Speed Clock(HSE)** và chọn "**Crystal/Ceramic Resonator**"

Chức năng này sẽ giúp chip hoạt động với thạch anh ngoại được gắn sẵn trên board mạch.

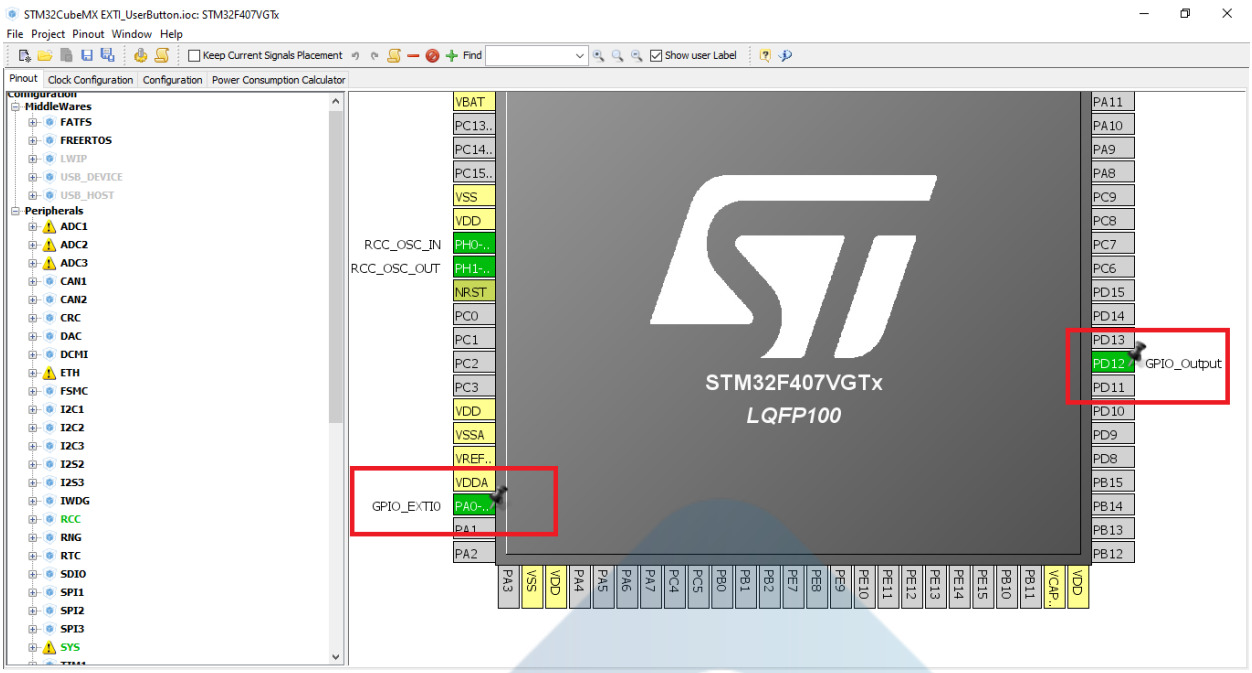


Tiếp theo, chúng ta chuyển sang Tab "**Clock Configuration**" và tích chọn mục **HSE**, tín hiệu Clock đi qua bộ nhân tần PLLCLK giúp chip đạt được tần số hoạt động tối đa.

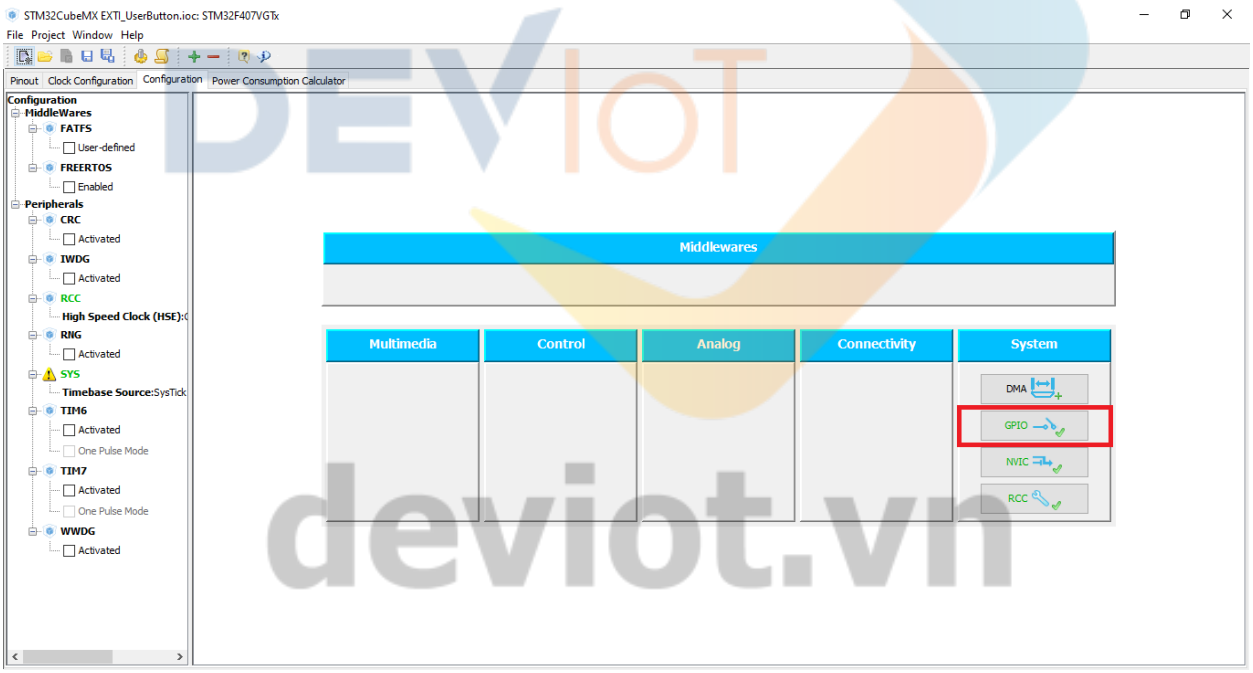
Tại mục **Input frequency**, các bạn điền "8" (thạch anh hàn sẵn trên board là loại 8Mhz). Sau đó chúng ta điền "168" tại mục HCLK (đây là tần số hoạt động tối đa của chip) và ấn Enter, đợi cho CubeMX tự tính toán các thông số.



Trong pinout, mình sẽ cấu hình cho chân **PA0** hoạt động với chức năng **GPIO_EXTI0** và cấu hình cho chân **PD12** ở chế độ **GPIO_Output** để quan sát sự hoạt động của ngắt

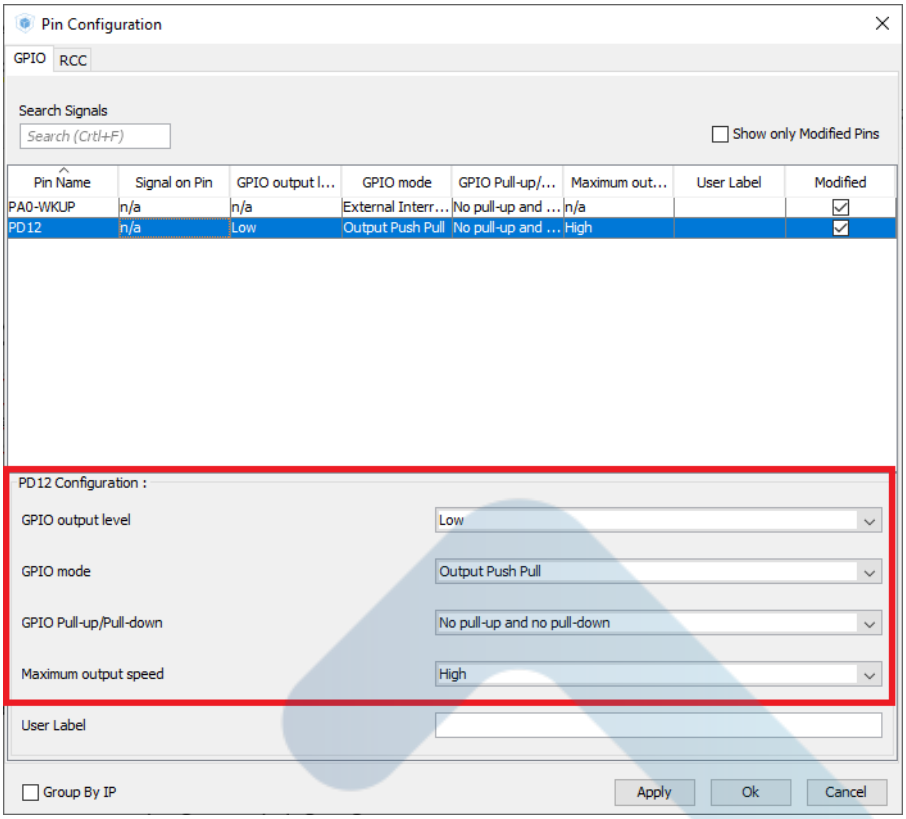


Cấu hình **GPIO** : chuyển sang Tab "**Configuration**", chúng ta chọn mục GPIO.

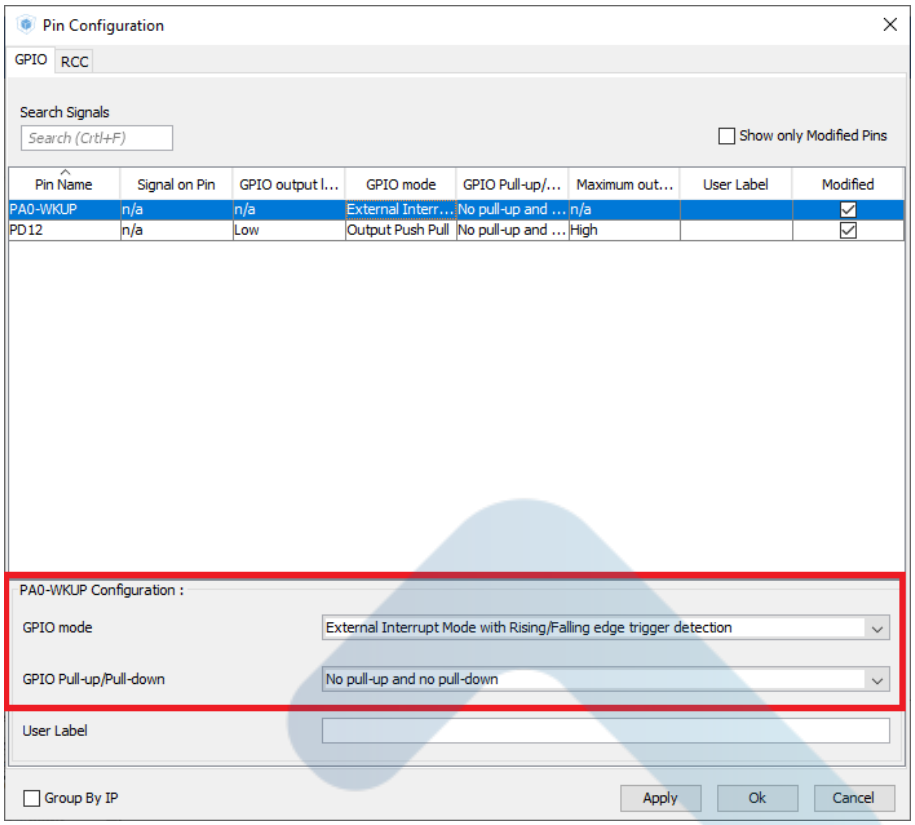


Tại đây, mình sẽ thiết lập các thông số như sau:

Đối với **PD12** :



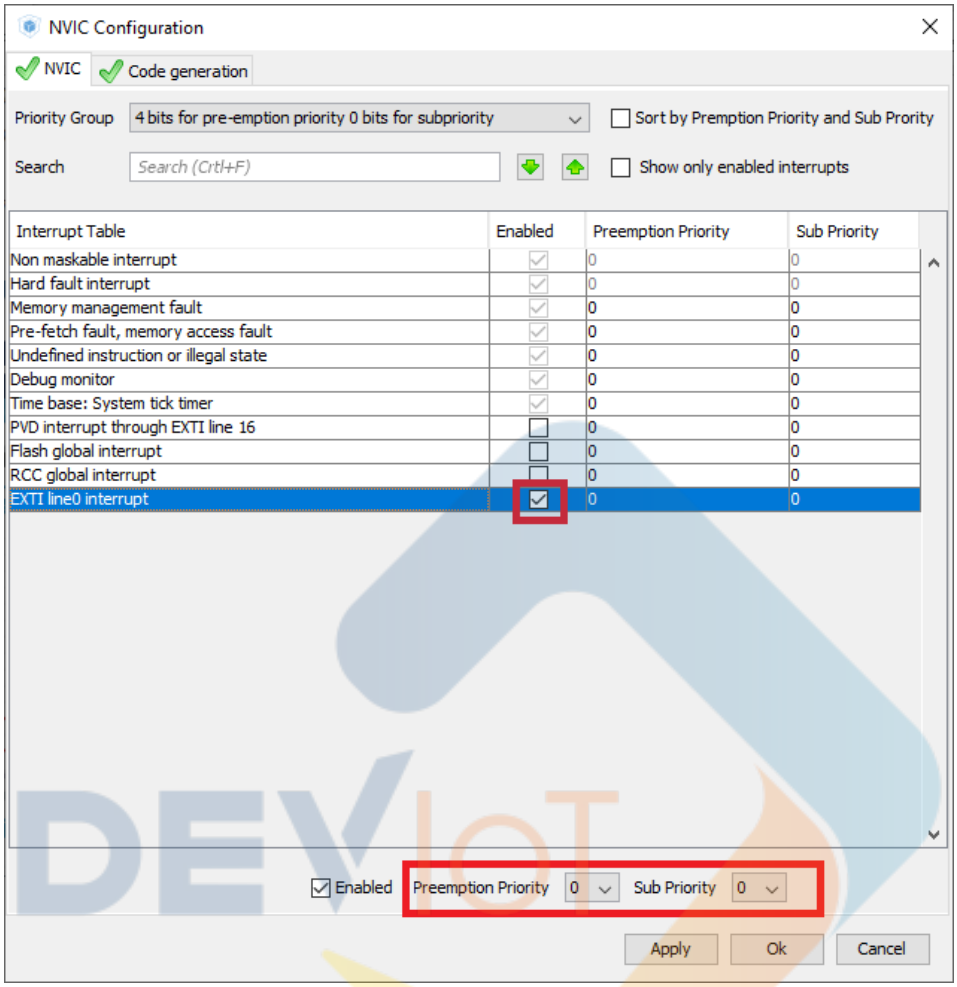
Đối với **PA0** :



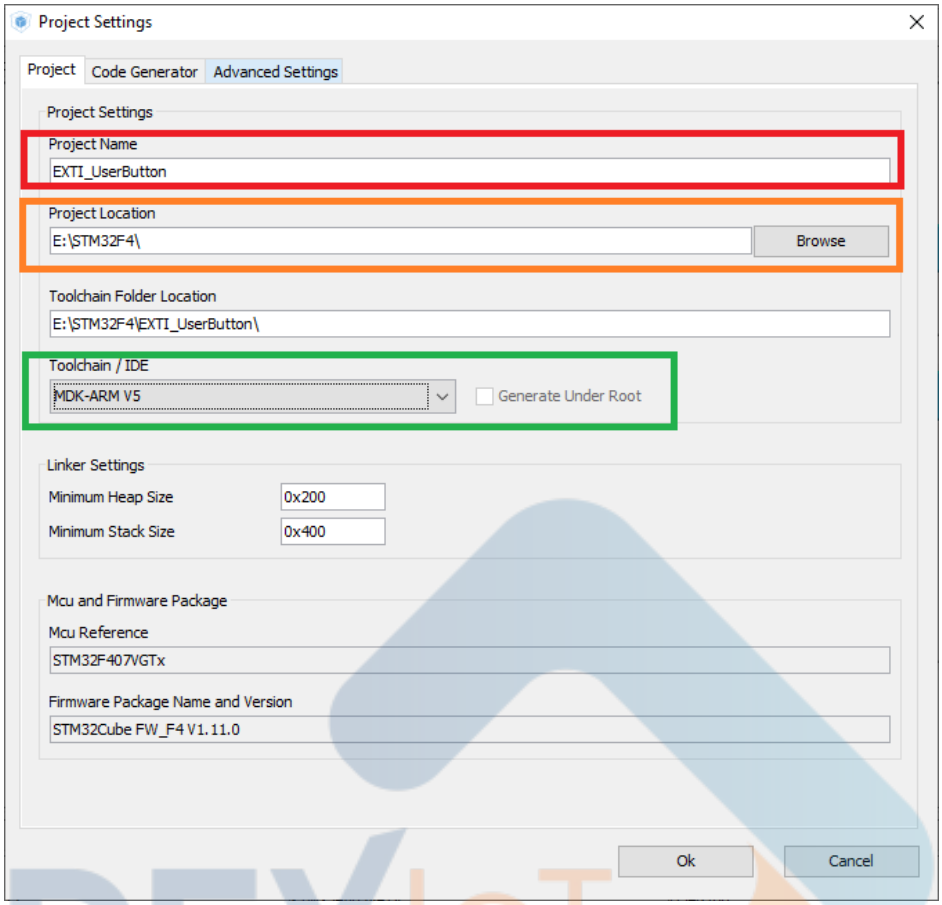
Tiếp đến, các bạn chọn mục NVIC để cấu hình ngắt.

Khi cửa sổ NVIC Configuration hiện lên, chúng ta sẽ Enable cho EXTI line0 interrupt.

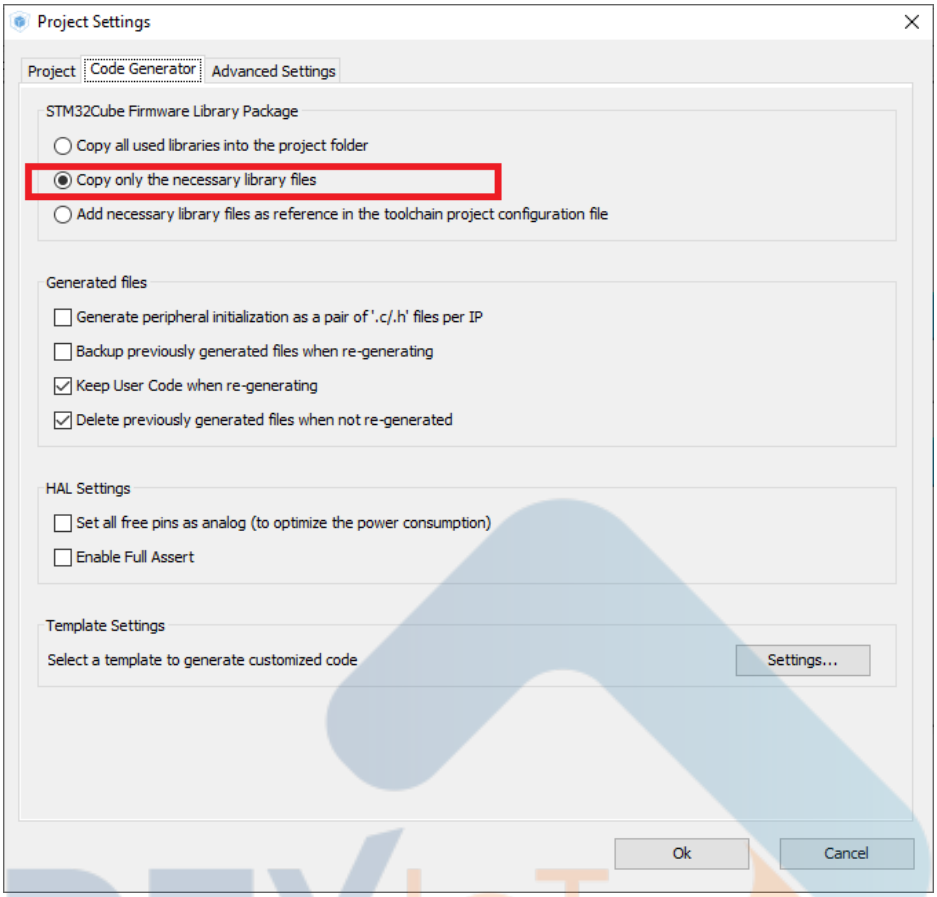
Tại mục Preemption Priority và Sub Priority, chúng ta có thể thay đổi mức 2 thông số cho mức ưu tiên ngắt. Nhưng trong ví dụ này, các bạn hãy để mặc định là “0” và “0”, và cùng theo dõi tiếp để hiểu rõ hơn **mức ưu tiên ngắt là gì, tác dụng như thế nào** ở cuối bài viết nhé !



Cuối cùng là **Setting Project** và tạo code.



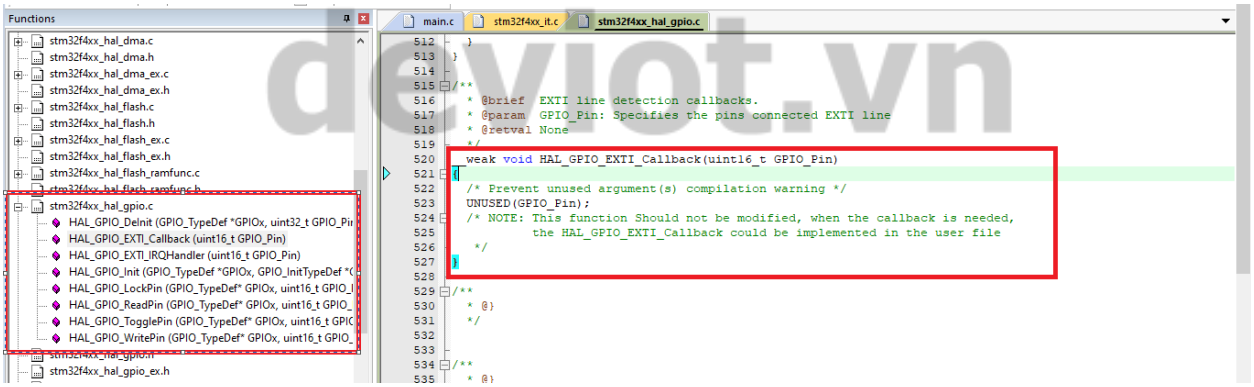
Ở Tab "**Code Generator**", hãy chọn "**Copy only necessary library files**" để chương trình sinh ra chỉ với các thư viện cần thiết, tiết kiệm dung lượng và thời gian build code nhé!



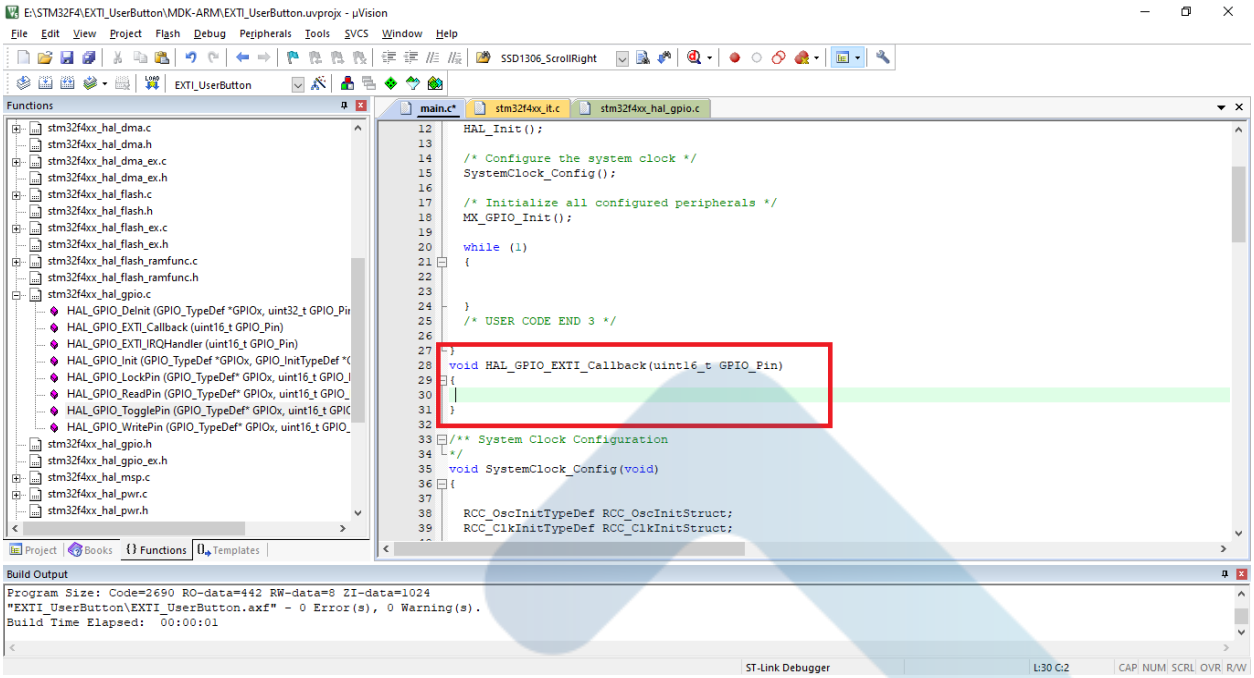
2, Lập trình với KeilC

Sau khi CubeMX sinh code xong, chúng ta chọn Open Project để mở chương trình trên KeilC.

Tại mục Functions, chúng ta mở file "**stm32f4xx_hal_gpio.c**", tìm đến hàm **HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin).** Hàm này sẽ phát hiện có sự kiện ngắt và xử lý yêu cầu ngắt đó.



Lưu ý : hàm này không nên chỉnh sửa vì được khai báo với **__weak** , nếu muốn sử dụng đến nó, chúng ta phải khai báo ở 1 file khác, ở đây mình sẽ khai báo trong file "main.c".



Trong hàm `void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)` chúng ta sẽ viết chương trình như sau :

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  if(GPIO_Pin == GPIO_PIN_0)
    {
        HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12);
    }
}
```

Câu lệnh `if(GPIO_Pin == GPIO_PIN_0)` sẽ giúp kiểm tra, phân luồng, phát hiện ngắt có đúng đang sinh ra có phải ở chân 0 hay không.

Build chương trình (**F7**) và nạp code xuống kit (**F8**)

Nhấn Reset button trên kit để reset lại KIT, thực hiện thao tác nhấn User button để quan sát led xanh lá cây trên KIT.

III, Mức ưu tiên ngắt trên vi điều khiển STM32

1, Đặt vấn đề

Như đã tìm hiểu ở trên, vi điều khiển STM32 hỗ trợ rất nhiều ngắt khác nhau và chúng được quản lý bởi NVIC (Nested Vector Interrupt Controller).

Vậy điều gì sẽ xảy ra nếu có 2 yêu cầu ngắt đang chờ để xử lý hoặc nếu 1 ngắt đang được xử lý thì 1 ngắt khác xuất hiện và 2 ngắt này có cùng mức ưu tiên cả Preemption Priority,Sub Priority?

Để làm rõ vấn đề này, chúng ta hãy làm 1 phép thử sau : thêm 1 dòng code `HAL_Delay(1000);`

vào trong hàm void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) trong file "main.c"

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  if(GPIO_Pin == GPIO_PIN_0)
  {
    HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12);
    HAL_Delay(1000);
  }
}
```

Build lại chương trình và nạp code xuống kit, sau đó reset lại KIT.

Nào, giờ hay nhấn User button như lúc nãy. Chắc chắn đèn led xanh lá sẽ sáng hẳn hoặc tối hẳn.

Điều này có nghĩa là gì?

Chương trình của bạn đã bị "treo", vì 2 yêu cầu ngắt có cùng mức ưu tiên đồng thời xuất hiện, điều này khiến chương trình bị đứng tại đây.

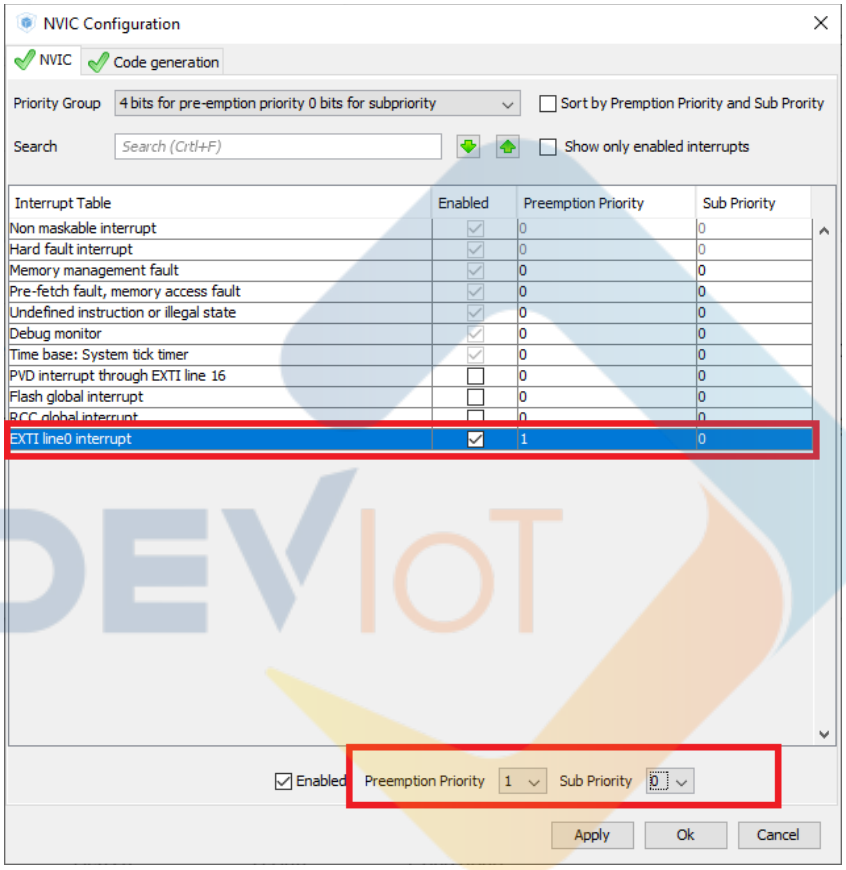
Hàm HAL_Delay() chúng ta hay sử dụng cũng là 1 kiểu ngắt, và nó có mức ưu tiên là Preemption Priority : 0, Sub Priority : 0

Vì vậy, mức ưu tiên của hàm HAL_Delay() ngang bằng với mức ưu tiên của EXTI line0 mà chúng ta đang sử dụng.

2, Cách khắc phục

Để khắc phục điều này, chúng ta cần xử lý như thế nào?

Cách 1:
Mở lại CubeMX và thiết lập lại thông số cho Preemption Priority, Sub Priority của EXTI line0, tạo lại code mới.

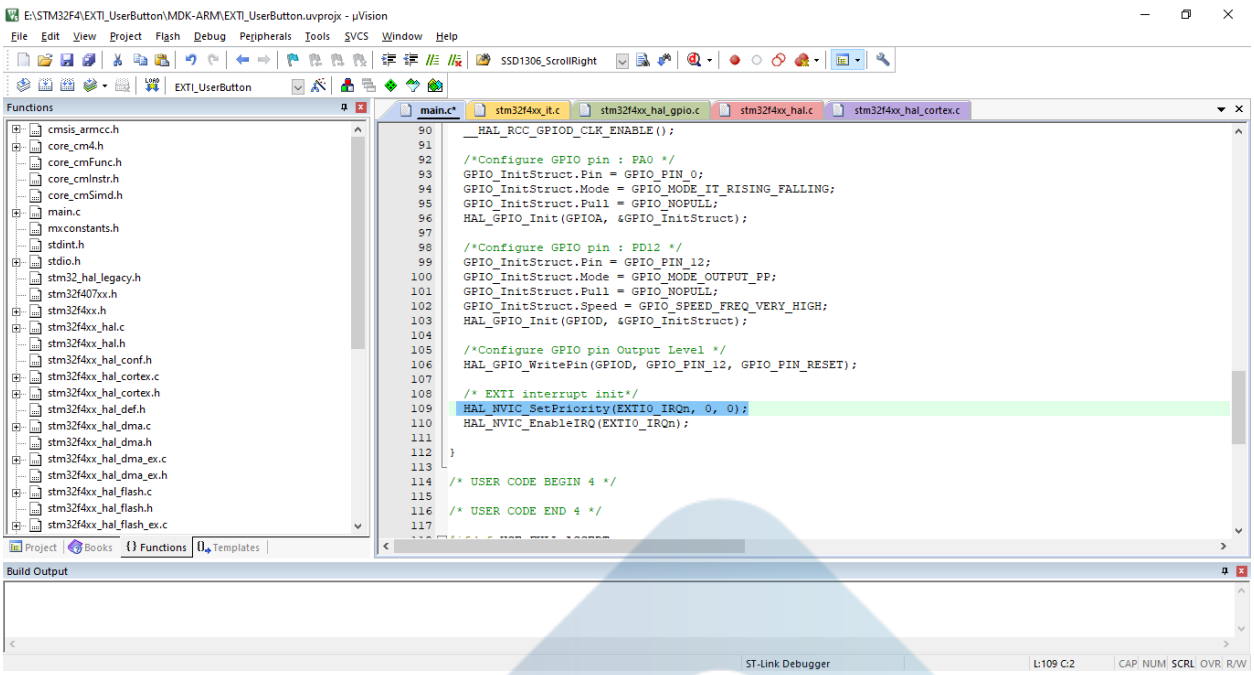


Lưu ý : Cách này chỉ nên thực hiện trước khi sinh code ra KeilC, vì nếu bạn khởi tạo lại 2 thông số này, sau đó remake project, toàn bộ code trong chương trình cũ sẽ bị reset lại.

Cách 2 : Chỉnh sửa mức ưu tiên ngắt ngay trong chương trình của mình

Trong file “main.c”, chúng ta kéo xuống và tìm đến hàm.

```
HAL_NVIC_SetPriority(EXTI0_IRQn, 0, 0);
```



Hàm này cho phép chúng ta thiết lập mức ưu tiên cho các line ngắt. Chúng ta sẽ sửa hàm này thành như sau :

```
HAL_NVIC_SetPriority(EXTI0_IRQn, 1, 0);
```

Điều này có nghĩa chúng ta đã thay đổi Preemption Priority cho EXTI line0 từ “0” xuống “1”, và Sub Priority vẫn giữ là “0” . Như vậy mức ưu tiên của EXTI line0 sẽ không ngang bằng với ngắt của hàm HAL_Delay() nữa.

Sau đó build lại chương trình và nạp code xuống kit. Reset lại board và nhấn User button, quan sát sự thay đổi của led nhé!

Trên đây là bài giới thiệu về ngắt ngoài (EXTI) cho vi điều khiển STM32F4 và cách cấu hình mức ưu tiên ngắt. Hy vọng bài viết này sẽ giúp các bạn có một cái nhìn trực quan về ngắt và sử dụng tốt được ngắt.

DEVIOT - CÙNG NHAU HỌC LẬP TRÌNH IOT

📌 Website: deviot.vn

📌 FanPage: Deviot - Thời sự kỹ thuật & IoT

📌 Group: Deviot - Cùng nhau học lập trình IOT

📌 Hotline: 0969.666.522

📌 Address: Số 101C, Xã Đàn 2

📌 Đào tạo thật, học thật, làm thật