Hướng dẫn lập trình giao tiếp GPIO cơ bản - STM32F407VG

Khi mới bắt đầu tìm hiểu, nghiên cứu bất kỳ dòng vi điều khiển nào, GPIO luôn là phần kiến thức đầu tiên mà lập trình viên sử dụng, nghiên cứu.

Trong bài viết này, mình sẽ hướng dẫn các bạn lập trình giao tiếp GPIO cơ bản với kit STM32F407VG Discovery. Cụ thể sẽ là nháy led (Blink – 1 ví dụ kinh điển đối với bất kỳ vi điều khiển nào)

I, Lý thuyết

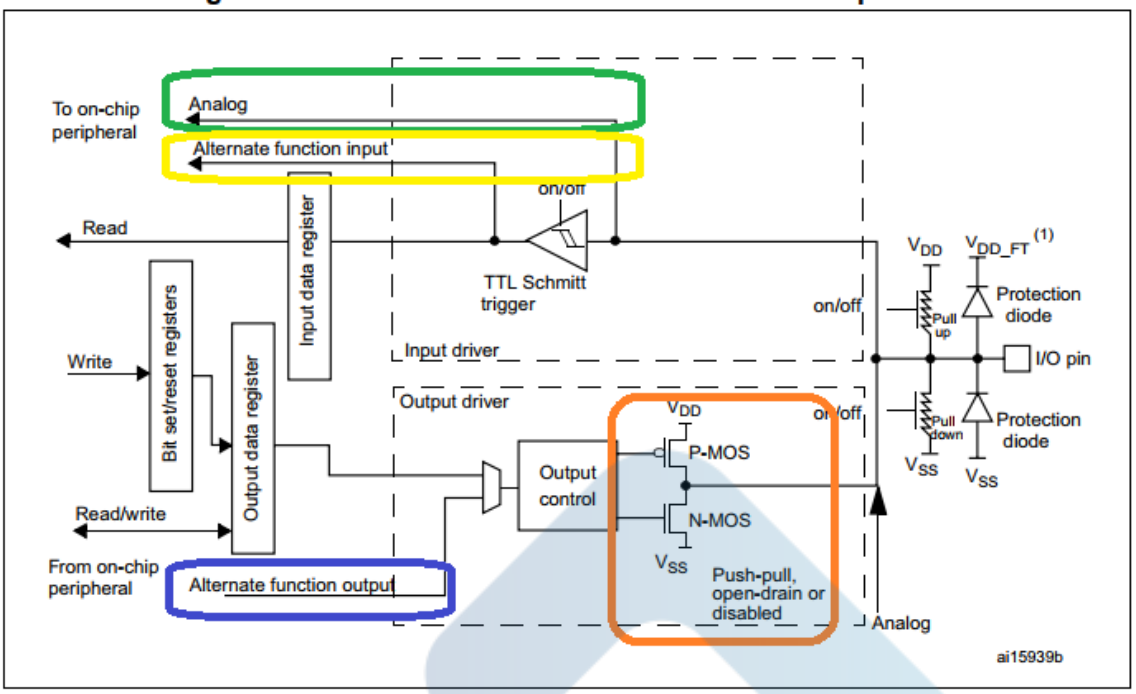
General-purpose Input/Output (GPIO) rất phổ biến, là một chức năng ngoại vi cơ bản của mỗi loại vi điều khiển, bao gồm các chân đầu vào và chân đầu ra, có thể được điều khiển bởi người dùng. Nó tương tự với các dòng vi điều khiển 8bit như AVR, 8051, PIC.

Không như các dòng vi điều khiển 8bit, chỉ có 8 chân IO trên 1 port, ở các vi điều khiển 32bit có đến 16 chân IO trên 1 port.

Cụ thể đối với kit STM32F407VG, có 5 port chính là GPIOA, GPIOB, GPIOC, GPIOD, GPIOE, trên mỗi port có các chân I/O được ký hiệu 0 đến 15.

Tiếp theo, chúng ta cùng đi sâu tìm hiểu phần cấu trúc mỗi chân GPIO của chip:

Figure 25. Basic structure of a five-volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

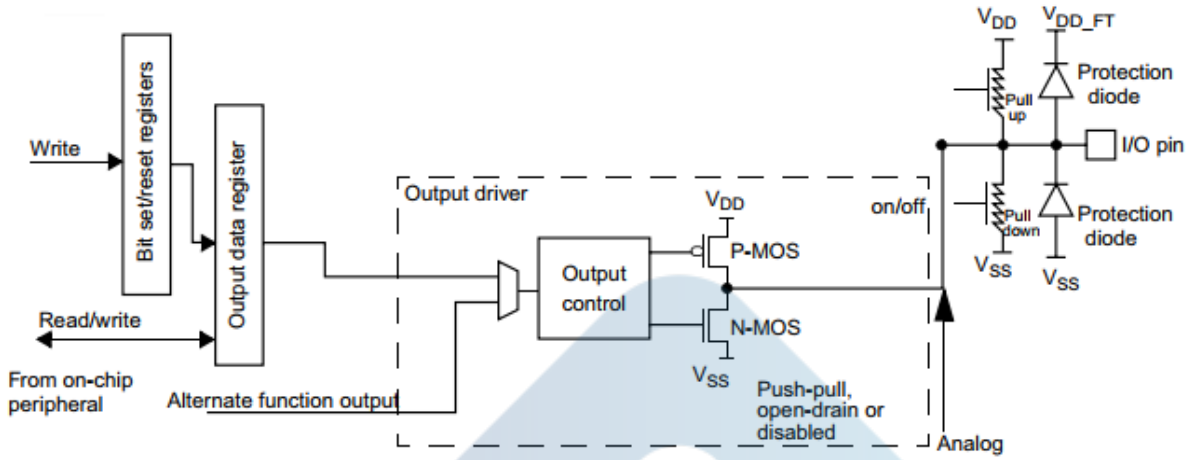
Có 2 khối điều khiển chính của mỗi GPIO (2 khối được vẽ đứt trong hình), đó chính là :

- Input driver
- Output driver

GPIO bao gồm 8 chứng năng chính sau đây :

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Mặc định khi lập trình viên không cấu hình gì, trạng thái của các chân I/O sẽ là Input Floating. Trong bài viết này, chúng ta sẽ sử dụng chức năng Output của GPIO, các bạn cùng tìm hiểu sơ lược về cấu trúc phần cứng của khối Output nhé.



1. Các thanh ghi quan trọng của GPIO

Mỗi chân GPIO đều có 2 thanh ghi cơ bản cấu hình 32 bit là (GPIOx_CRL – Control Register Low, GPIO_CRH – Control Register High)

Chúng ta quan tâm đến 2 thanh ghi sau:

GPIO port bit set/reset register (GPIOx_BSRR):

Thanh ghi này dùng để cấu hình các chân ở mức set (mức cao) hoặc reset (mức thấp)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset bit y (y = 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit
1: Resets the corresponding ODRx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy:** Port x set bit y (y= 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit
1: Sets the corresponding ODRx bit

GPIO port output data register (GPIOx_ODR)

Dữ liệu sau khi các bit đã được set/reset ở thanh ghi trên sẽ được truyền sang thanh ghi dữ liệu đầu ra 32bit (GPIOx_ODR: Output data register) và truyền đến khối điều khiển để xuất mức tín hiệu cho chân I/O. Ngoài ra đối với thanh ghi này, chúng ta có thể đọc dữ liệu để xem trạng thái hiện tại của các chân IO đang ở mức "1" hoặc mức "0".

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODRy**: Port output data (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..I/J/K).

2. Xuất tín hiệu output thông qua khối CMOS

Khi một I/O pin được cấu hình hoạt động với chức năng Output thì khối điều khiển Output driver được sử dụng với các chế độ **: Open drain mode** hoặc **Push-Pull mode**:

- **Open drain mode**: Ở chế độ này, mạch sẽ không sử dụng P-MOS (luôn khóa) và chỉ sử dụng N-MOS. Khi một giá trị bit của thanh ghi ODR bằng 0 sẽ làm N-MOS dẫn, lúc

này chân vi điều khiển được kéo xuống GND và có mức logic thấp (mức 0). Một giá trị bit của thanh ghi ODR bằng 1 sẽ làm N-MOS đóng, chân tương ứng sẽ ở trạng tháng Hi-Z (trở kháng cao).

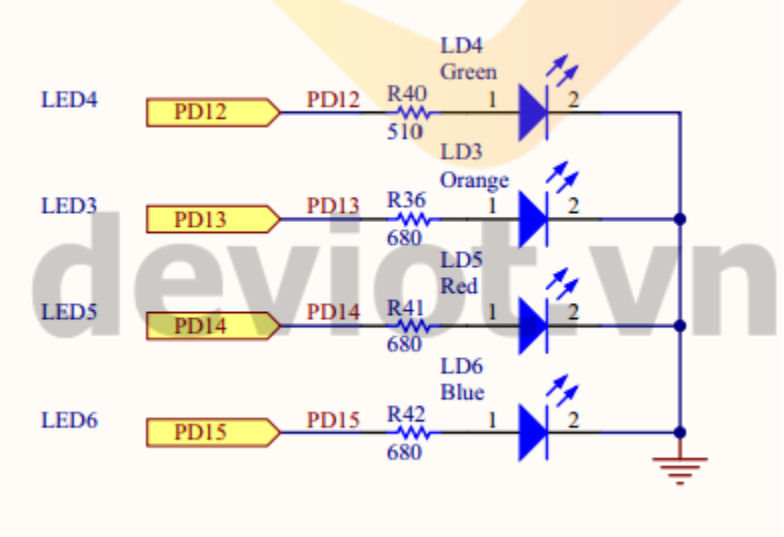
- Với **Push-pull mode** : Ở chế độ này mạch sẽ sử dụng cả P-MOS và N-MOS. Một giá trị bit của thanh ghi ODR bằng 0 sẽ làm N-MOS dẫn, P-MOS ngưng dẫn, lúc này chân vi điều khiển có mức thấp (được nối với GND). Một giá trị bit của thanh ghi ODR bằng 1 sẽ làm N-MOS ngưng dẫn và P-MOS dẫn. Lúc này chân vi điều khiển có mức cao (mức logic 1 – được nối với VDD).

Như vậy, để điều khiển giá trị logic của 1 I/O pin được cấu hình hoạt động với chức năng Output, chúng ta cần ghi giá trị logic vào Output Data Register (GPIOx_ODR). Bit tương ứng của thanh ghi sẽ điều khiển pin ở vị trí tương ứng.

VD: bit thứ 0 của thanh ghi GPIOB-ODR sẽ điều khiển chân PB0.

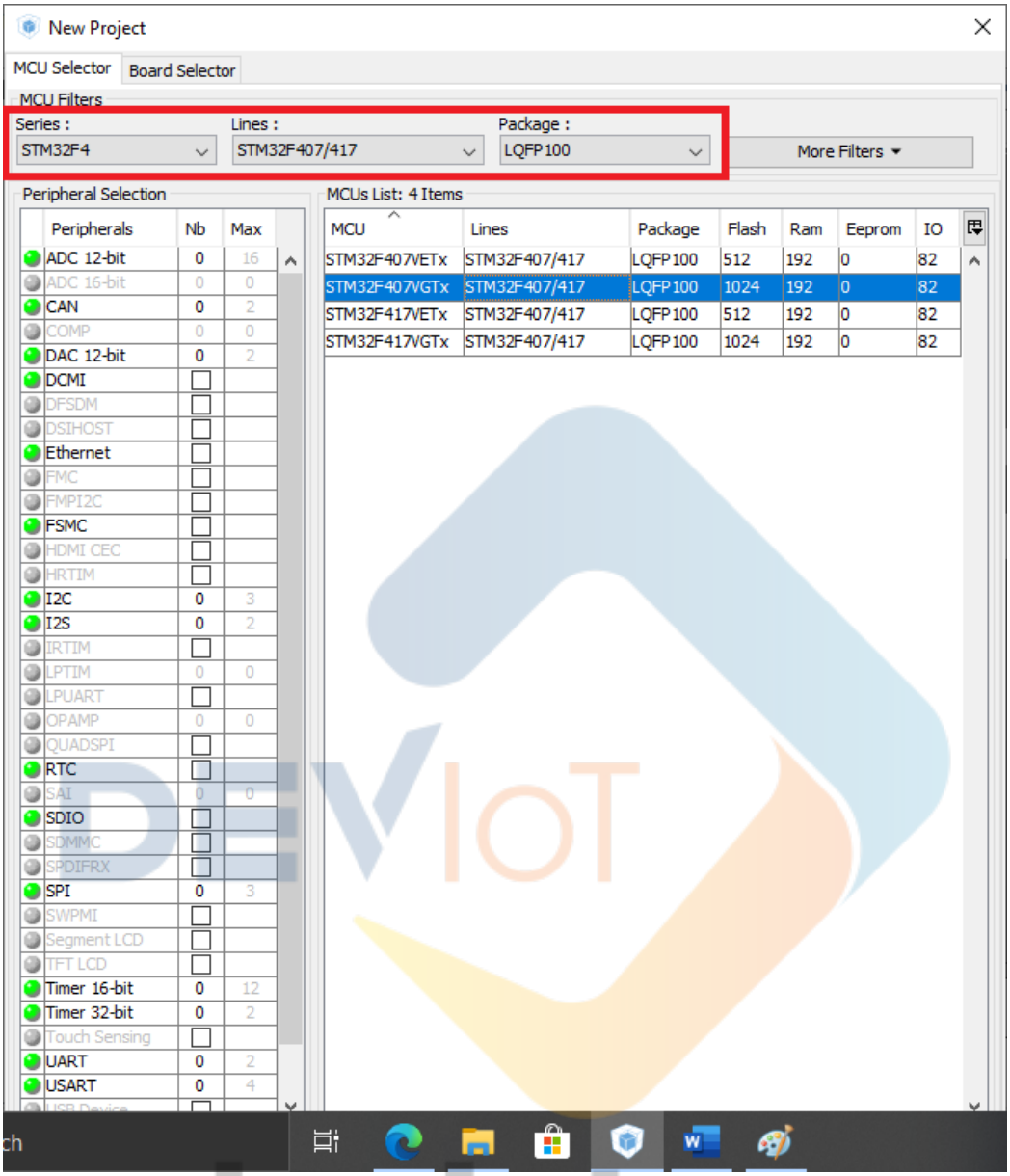
II, Lập trình

Như đã giới thiệu ở trên, kit **STM32F407VG** có 5 Port chính **A, B, C, D, E**, mỗi port này có 16 chân và được ký hiệu từ 0 đến 15. Để quan sát được sự thay đổi tín hiệu trên các chân, cách đơn giản nhất chúng ta kết nối chân với l đèn led. Trên kit này, nhà sản xuất đã kết nối sẵn cho chúng ta 4 chân với 4 đèn Led khác nhau. Đó là các chân **PD12, PD13, PD14** và **PD15**.

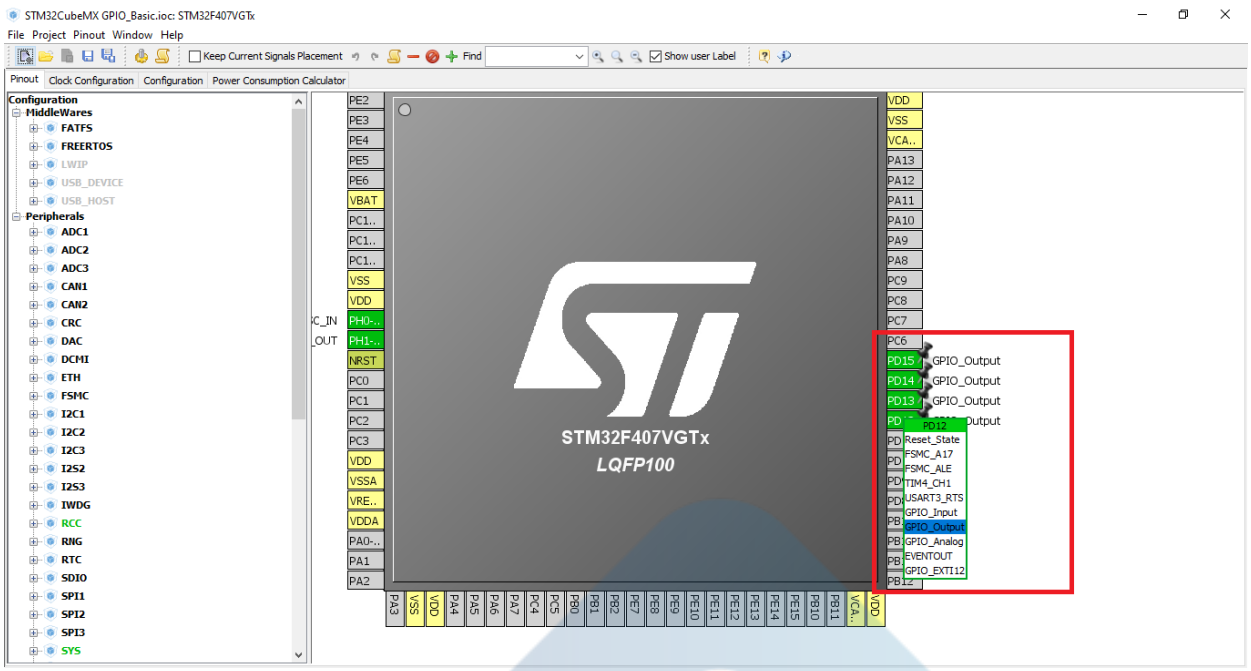


1, Cấu hình với CubeMX:

Khởi động CubeMX và chọn dòng vi điều khiển bạn muốn sử dụng



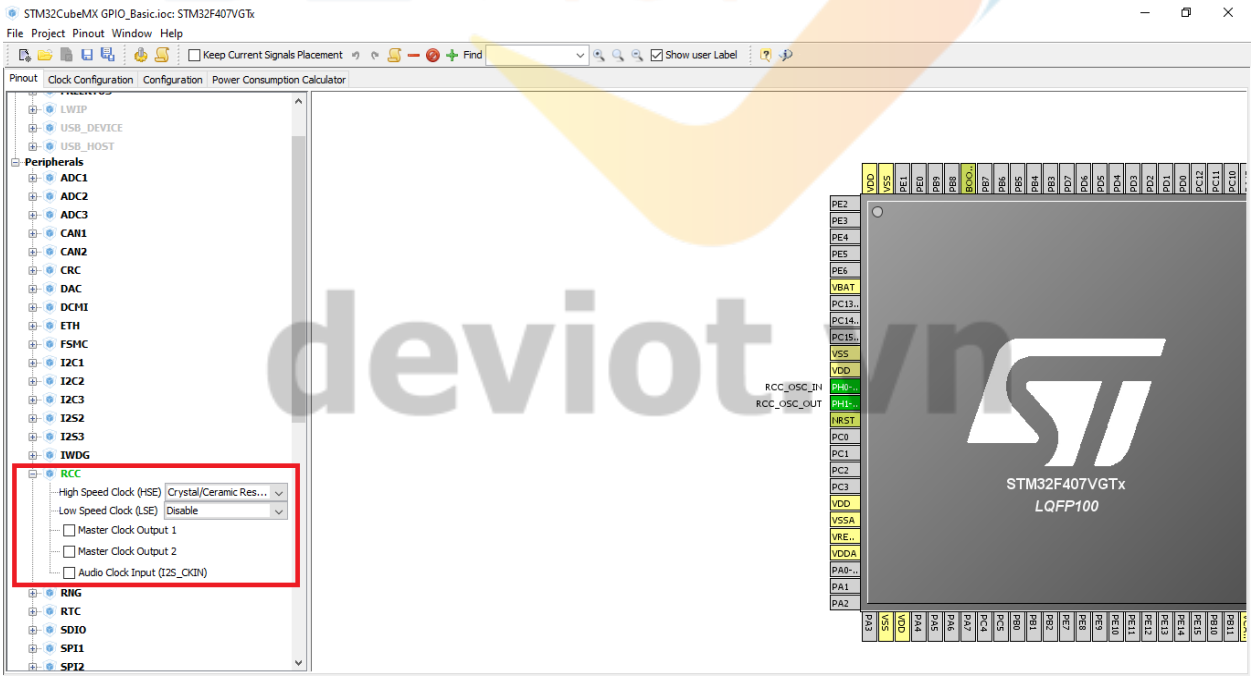
Trong **PINOUT**, chúng ta tìm đến các chân **PD12, PD13, PD14, PD15** và chọn chức năng "**GPIO_Output**".



Cấu hình thạch anh và xung clock cho chip:

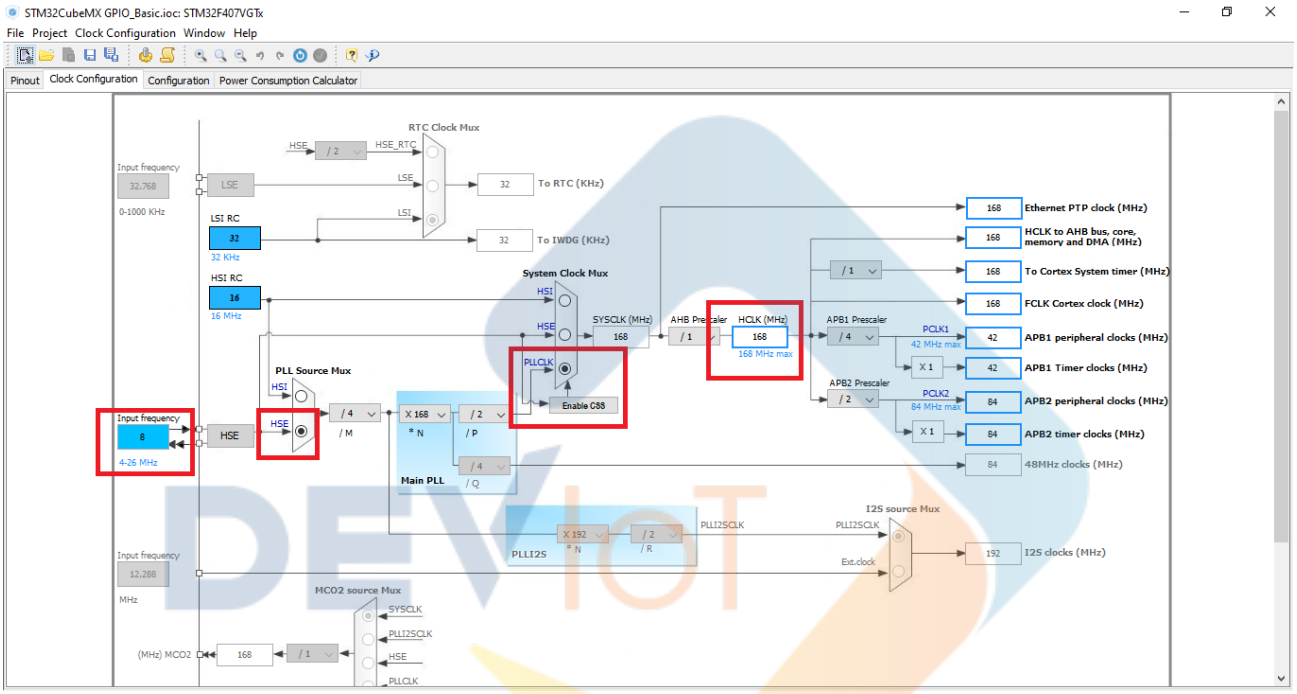
Các bạn tìm đến phần **RCC → High Speed Clock (HSE)** và chọn "**Crystal/Ceramic Resonator**".

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



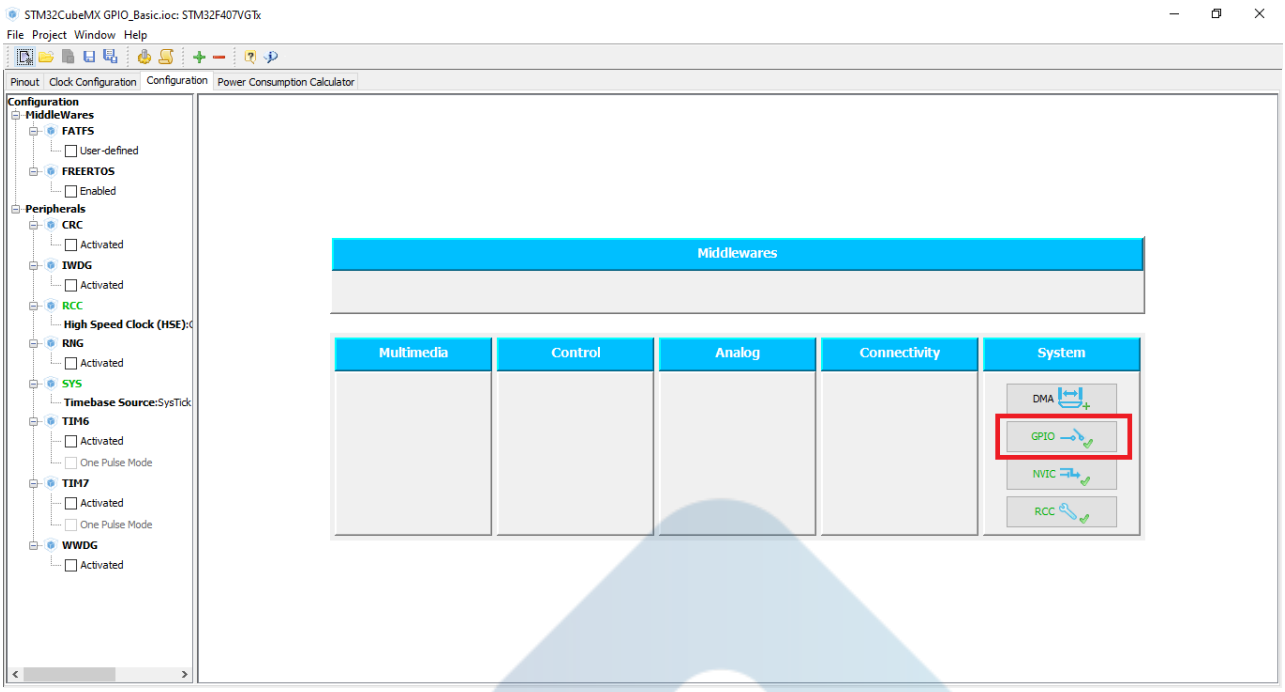
Tiếp theo, chúng ta tìm đến mục “Clock Configuration” và tích chọn mục HSE (nguồn thạch anh ngoài), tín hiệu clock sẽ đi qua bộ nhân tần PLLCLK giúp chip đạt được ở tần số hoạt động tối đa.

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 còn lại.



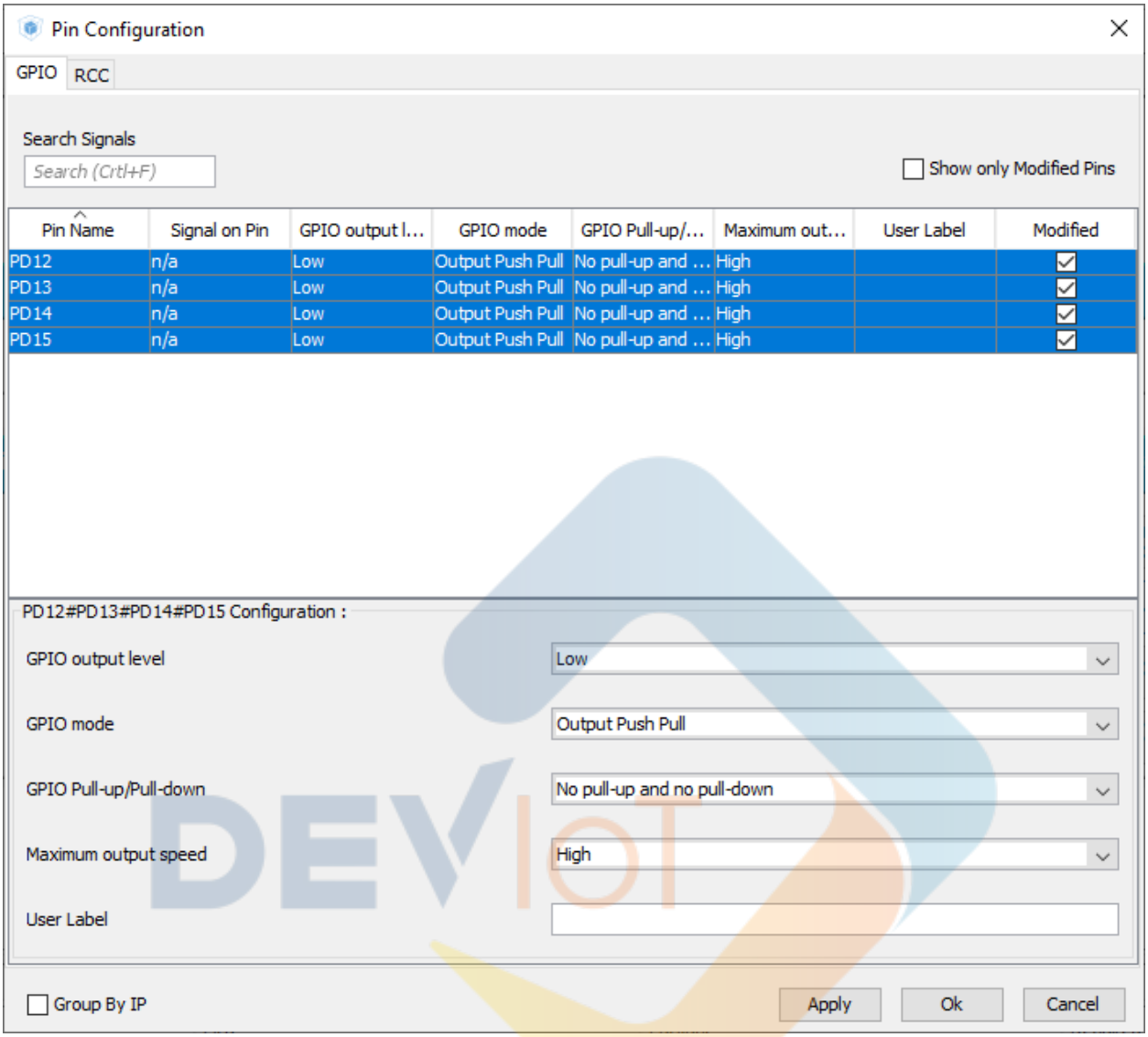
Cấu hình cho các chân GPIO

Trong Tab “**Configuration**”, các bạn chọn mục GPIO



Chúng ta chọn các thông số cho các GPIO như dưới đây :

- **GPIO output level**: Low (cấu hình ban đầy cho các chân đang ở mức thấp)
- **GPIO mode**: Output Push Pull
- **GPIO Pull-up/Pull-down**: No pull-up and no pull-down (không cần điện trở kéo lên và kéo xuống)
- **Maximum output speed**: High (với các phiên bản CubeMX mới sẽ có Very High, các bạn nên chọn Very High nhé)



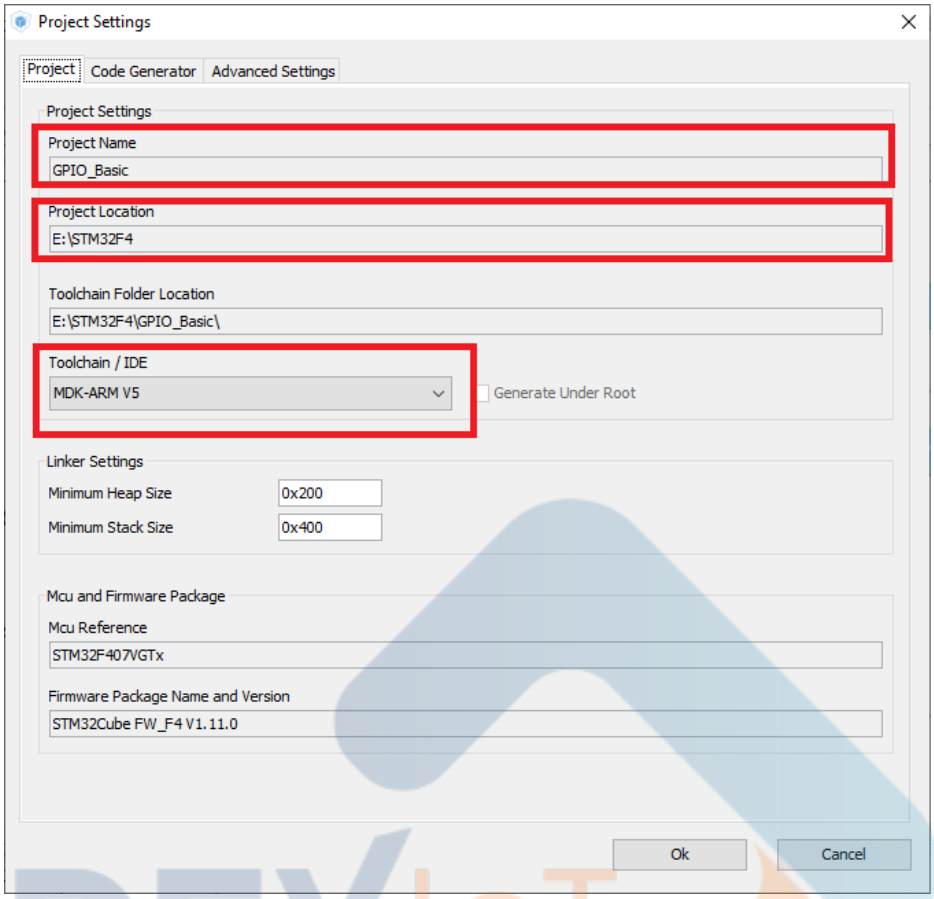
Cuối cùng là Setting cho Project và sinh code:

Chúng ta cùng cài đặt 1 số phần cơ bản trước khi tạo code nhé.

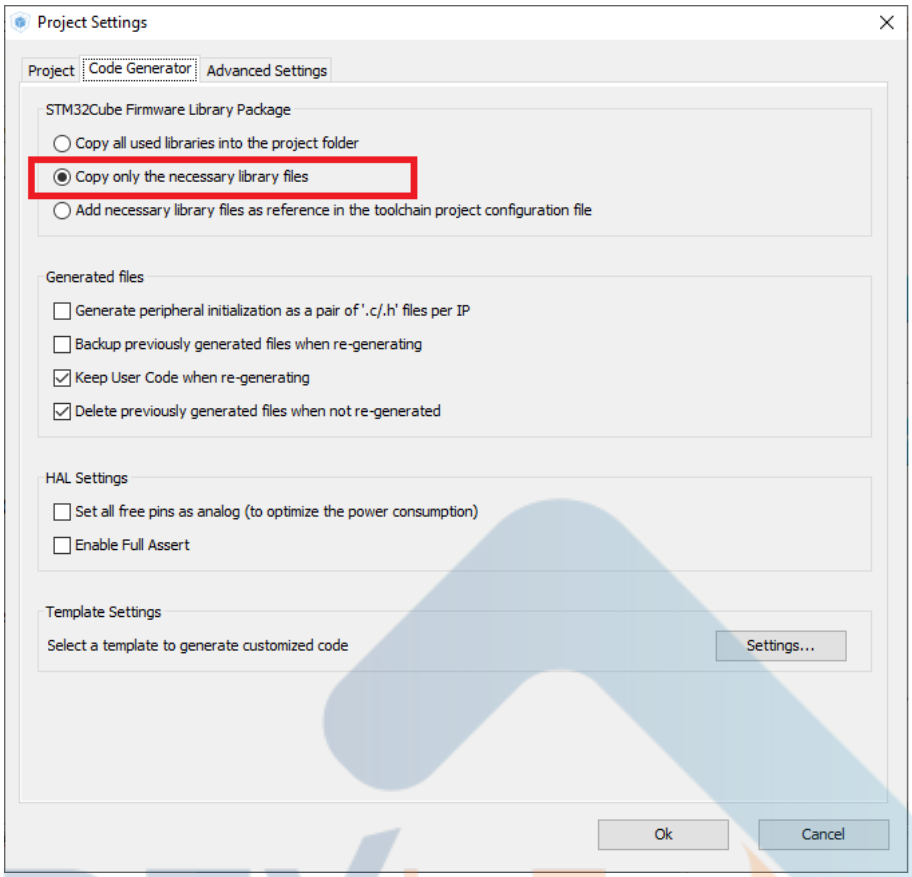
Tại mục Project Name, điền tên bạn muốn đặt cho project.

Tại mục Project Location, click “Browse” để chọn thư mục lưu project

Trong mục Toolchain/IDE, các bạn chọn MDK-ARM V5 nhé



Tiếp tục chuyển sang Tab "**Code Generator**", các bạn chọn "**Copy only the necessary library files**" để trong project của chúng ta chỉ có những thư viện cần thiết, điều này sẽ giúp tiết kiệm đáng kể dung lượng.



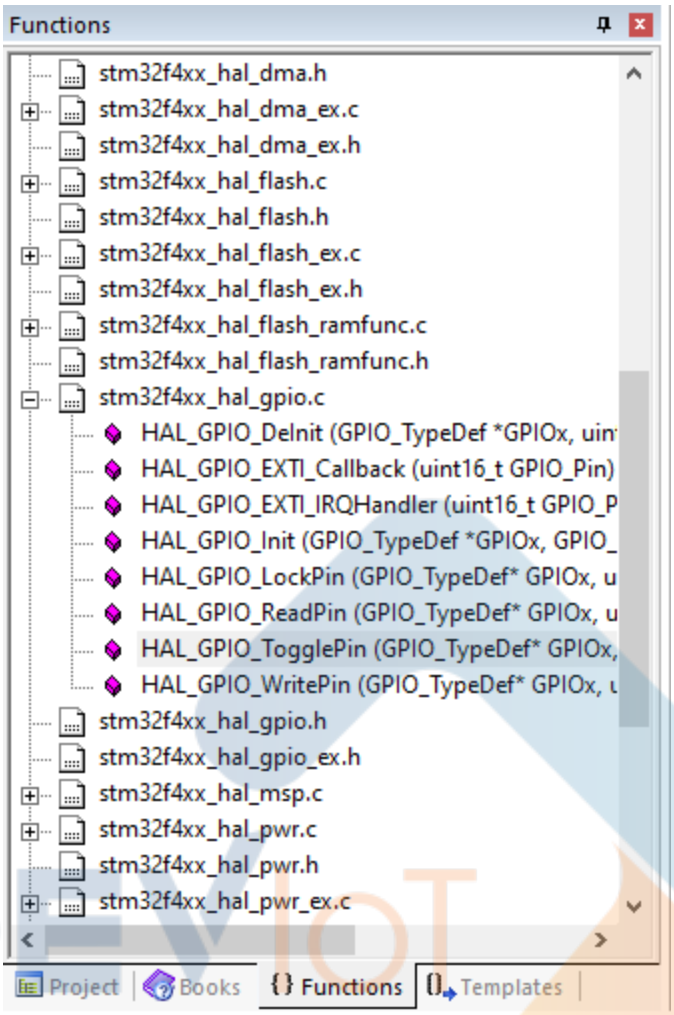
Cuối cùng là Generate Code và Open Project sau khi CubeMX sinh code xong

2, Lập trình với KeilC

Tại mục Functions, trong file "**stm32f4xx_hal_gpio.c**" sẽ chứa các hàm cơ bản để điều khiển GPIO.

HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin) cho phép đảo trạng thái của l chân bất kỳ. Ở đây mình sẽ truyền vào 2 tham số, thứ nhất là Port cần sử dụng (GPIOx) và tham số thứ 2 là chân IO cần sử dụng (GPIO_Pin) cụ thể là:

HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15);



Ngoài ra có thể xuất mức "1" hoặc mức "0" tại chân IO thông qua hàm:

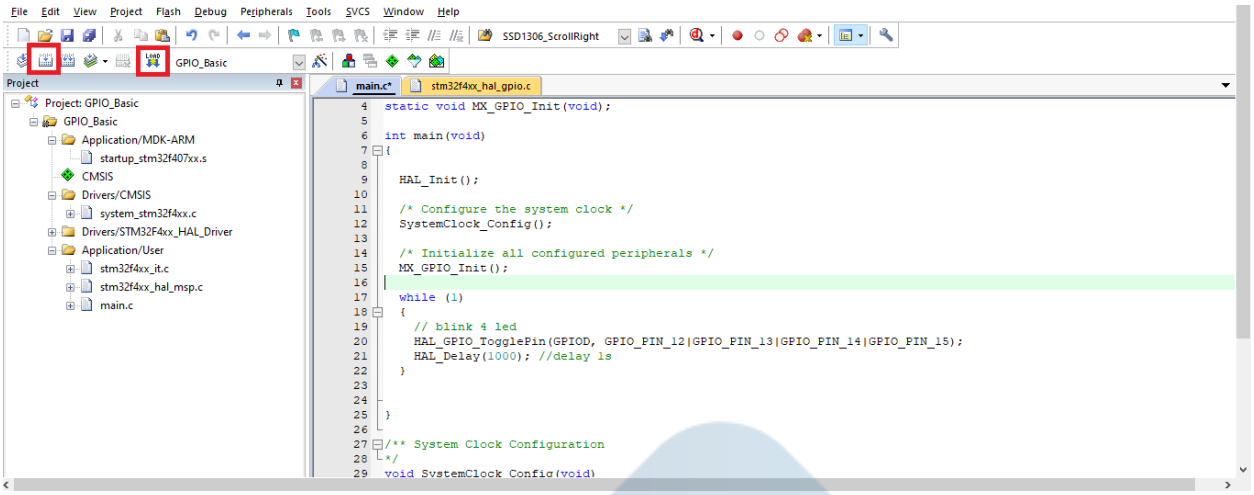
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);

Hai hàm này sẽ đặt trạng thái cho 1 chân bất kỳ, "**GPIO_PIN_SET**" là mức "1", "**GPIO_PIN_RESET**" là mức "0".

Hàm while(1) sẽ là 1 vòng lặp vô hạn, trong hàm này mình sẽ viết code để đảo trạng thái chớp tắt led liên tục chu kỳ 1s.

```
while (1)
{
// đảo trạng thái 4 led
  HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15);
  HAL_Delay(1000); //delay 1s
}
```

Chúng ta build chương trình (**F7**) và nạp code xuống kit (**F8**)



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