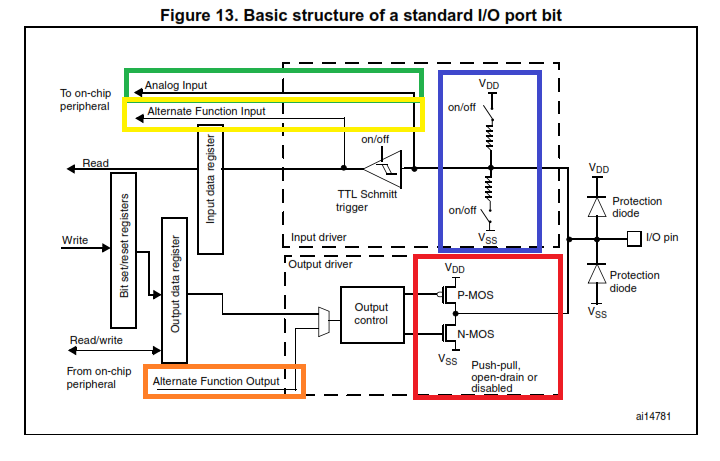
Hướng dẫn sử dụng kit STM32F103C8T6 bản Start

Điều khiển LED đơn với STM32 start kit

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 hoặc PIC. Không giống như những dòng vi điều khiển 8bit, chỉ có 8 chân IO trên 1 port thì ở các vi điều khiển 32bit, có đến 16 chân IO trên 1 port. Cụ thể đối với chip STM32F103C8Tx gồm có 3 Port chính đó là GPIOA, GPIOB, GPIOC. Không phải tất cả các port đều có 16 chân, chỉ riêng GPIOA, GPIOB trên kit thì có đủ 16 chân GPIO.

Tiếp theo chúng ta đến phần cấu trúc 1 chân GPIO của chip:

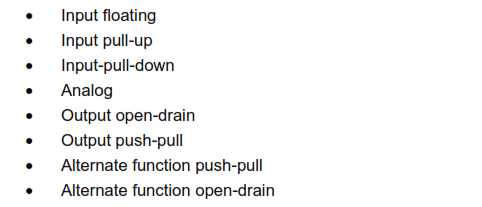


Có 2 khối điều khiển khác nhau (khung hình nét đứt) như ở hình trên:

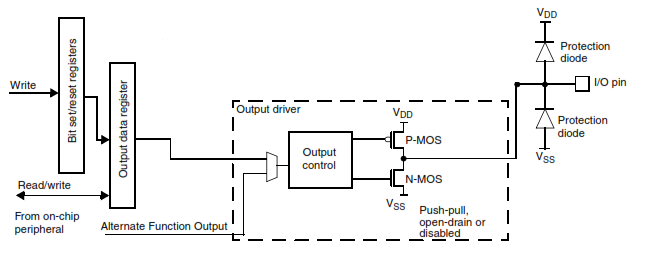
– Output driver

– Input driver

Chức năng của GPIO bao gồm:



Mặc định khi người dùng không cấu hình, trạng thái của các chân I/O pins là input floating. Bài viết này sẽ hướng dẫn về chức năng của khối Output. Sau đây sẽ là sơ lược về cấu trúc phần cứng và khối điều khiển Output.



**1. Các thanh ghi dữ liệu:**

**Bit set/reset register:**

Ở mỗi chân general-purpose I/O port đều có 2 thanh ghi cấu hình 32 bit (GPIOx\_CRL – Control Register Low, GPIO\_CRH – Control Register High):

–  Thanh ghi 32 bit dùng để set/reset các bit ở các chân IO (GPIOx\_BSRR: Bit Set Reset Register)

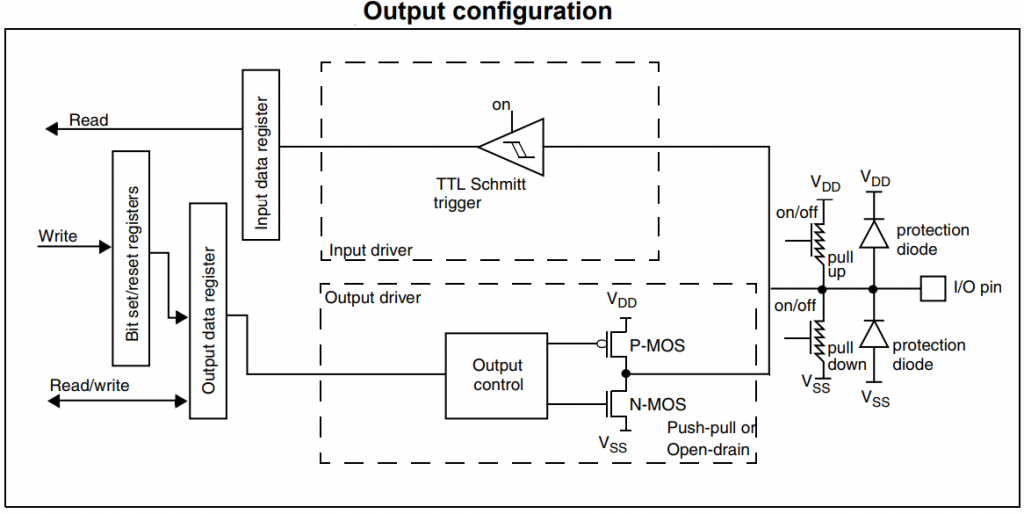
–  Thanh ghi 16 bit reset các bit ở các chân IO (GPIOx\_BRR: Bit Reset Register) với x là các port của vi điều khiển.

**Output data register:**

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 IO. 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”.

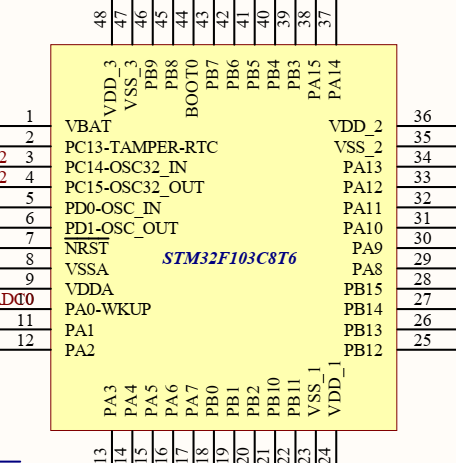
**2.Cách xuất mức tín hiệu output thông qua khối CMOS:**

Khi một I/O Pin được cấu hình hoạt động với chức năng OUTPUT thì:  
Khối điều khiển OUPUT được sử dụng với các chế độ: Open drain mode hoặc Push-pull mode.  
– Với chế độ Open drain: Một giá trị bit bằng “**0″** ở thanh ghi ODR sẽ làm N-MOS dẫn, P-MOS ngưng dẫn, lúc này chân vi điều khiển có mức **logic 0** (được nối với GND); Một giá trị bit bằng **“1”** ở thanh ghi ORD sẽ làm ngưng dẫn cả N-MOS và P-MOS, chân tương ứng sẽ ở trạng thái **Hi-Z** (trở kháng cao).   
– Với chế độ Push-pull:  Một giá trị bit bằng **0** ở thanh ghi ODR sẽ làm N-MOS dẫn và P-MOS ngưng dẫn, lúc này chân vi điều khiển có mức **logic 0** (được nối với GND); Một giá trị bit bằng **“1”** ở thanh ghi ODR sẽ làm N-MOS ngưng dẫn và P-MOS dẫn. Lúc này chân vi điều khiển có mức **logic 1** (được nối với VDD).  
Như vậy, để  điều khiển giá trị logic của một pin được cấu hình hoạt động với chức năng OUPUT thì chúng ta cần ghi giá trị logic vào thanh ghi Ouput Data (GPIOx\_ODR). Bit tương ứng của thanh ghi sẽ điều khiển pin ở vị trí tương ứng. Ví dụ: BIT thứ 0 của thanh ghi GPIOA\_ODR sẽ điều khiển pin tương ứng là PA0.

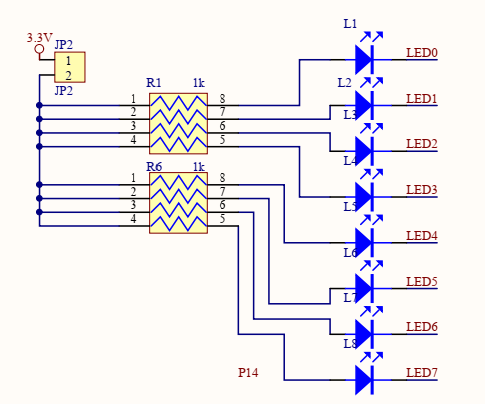


Ta sẽ sử dụng CubeMX để sinh code và lập trình trên môi trường KeilC-V5, hoặc ta có thể sử dụng CubeIDE để vừa cấu hình và vừa sinh code trên đó. Ở hướng dẫn này, mình sẽ sử dụng CubeIDE.  Khi sinh code, chúng ta sẽ làm việc dựa trên lớp thư viện HAL-Hardware Abstraction Layer. Thư viện này được xây dựng dựa trên các thư viện tiêu chuẩn (Std) của STM32, để tiếp cận được nhanh chóng dòng vi điều khiển này thì chúng ta sẽ không đi quá sâu vào việc set/reset các bit ở trong thanh ghi. Mà sử dụng phần mềm để sinh code và sử dụng các hàm GPIO có sẵn trong thư viện HAL.

Tiếp theo chúng ta sẽ đi sơ lược các chân GPIO của kit STM32F103C8T6:



Ở bài hướng dẫn này, ta sẽ đi thực hành nháy LED đơn bằng delay trên STM32 START KIT. Ở Port B chân thứ 0 đến 7 trên kit đã layout sẵn các đèn led, chúng ta sẽ sử dụng đèn led này để thực hành với chức năng output push-pull. 8 đèn LED được mắc chung Anode với nhau như sơ đồ dưới đây:



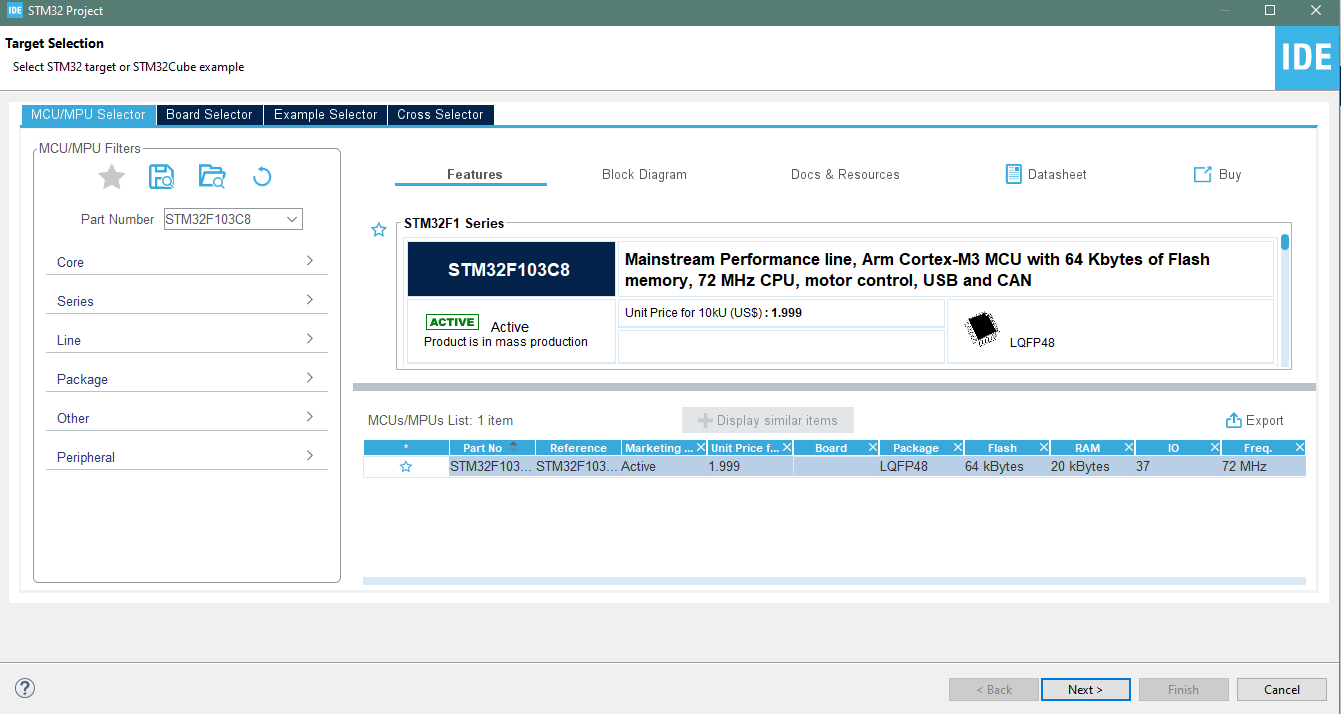
\*lưu ý: nếu muốn thực hiện nháy với led L3 thì ta phải gạt switch **BOOT1** ở trên board về trạng thái 1, nếu không thì L3 sẽ luôn sáng trong suốt cả quá trình:



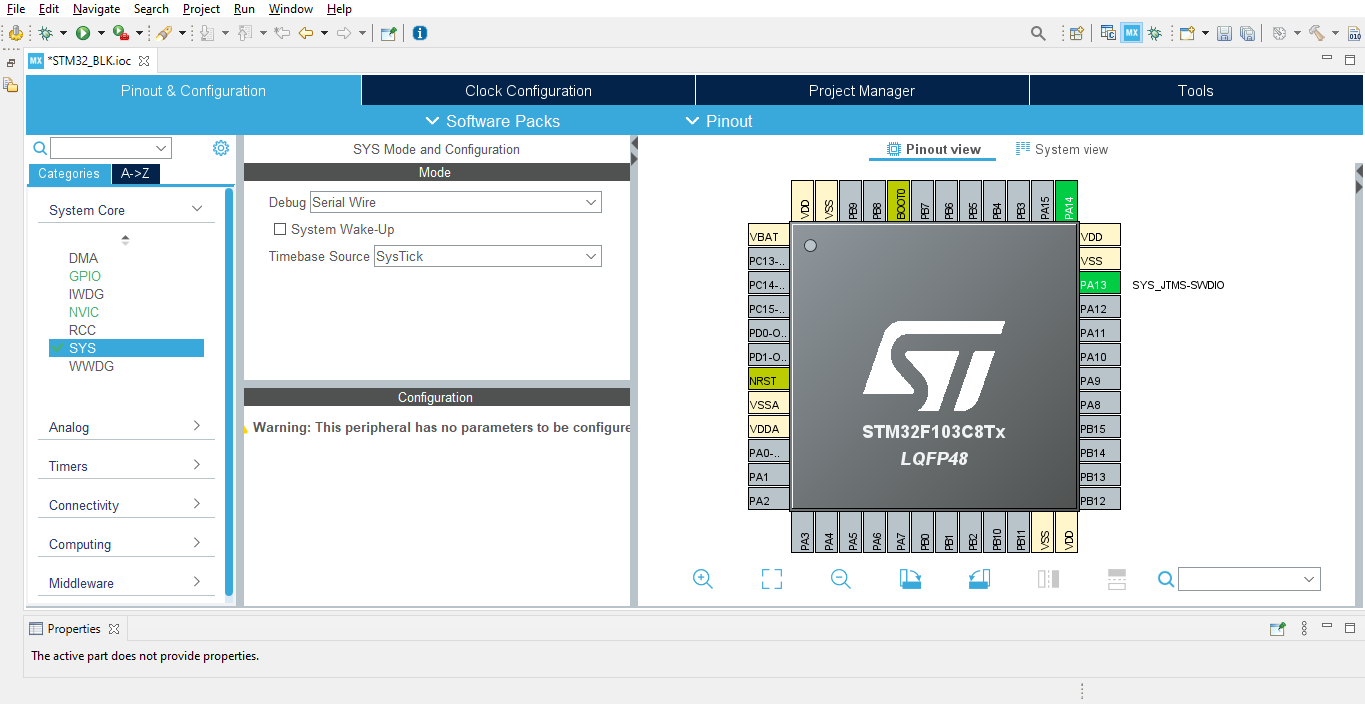
**3. hướng dẫn cấu hình và sinh code**

**Bước 1**:

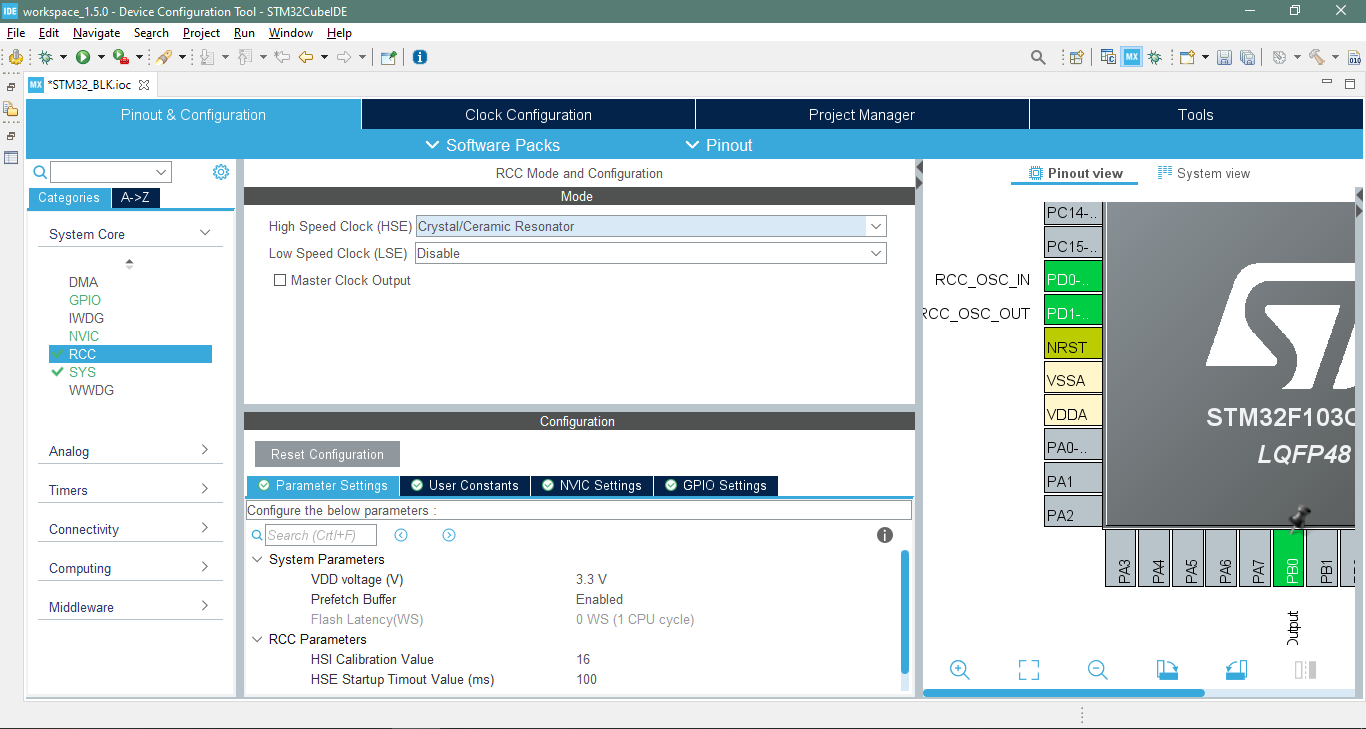
Mở CubeIDE, chọn **Start new STM32 project,** chọn chip STM32F103C8T6. Sau khi chọn xong chip thì ấn **next** để tiếp tục đặt tên và vị trí mà bạn muốn lưu project:



Bước 2: Vào phần system core, chọn cài đặt **Serial wire** để nạp code theo chuẩn SWDIO-SWCLK

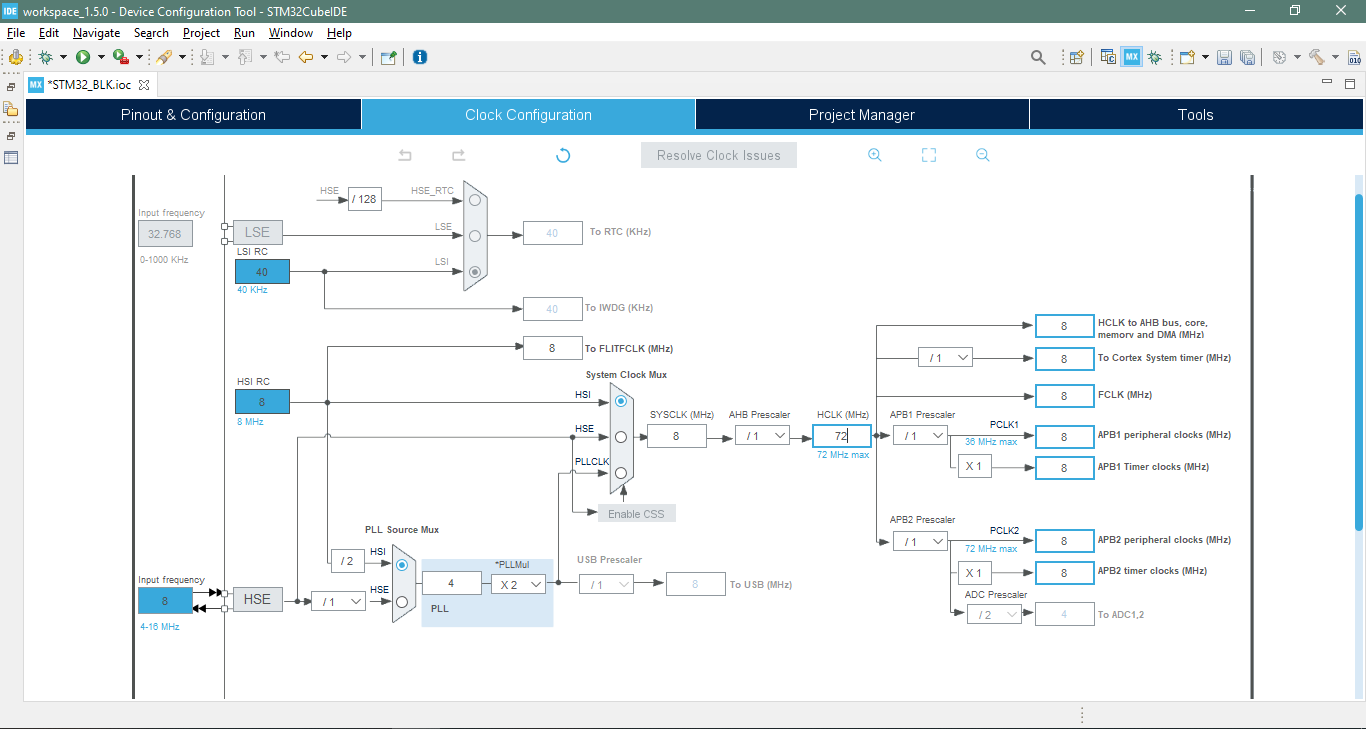


Bước 3: Vào phần RCC, High Speed Clock, chọn chức năng “Crystal/Ceramic Resonator”. Chức năng này giúp cho chip hoạt động với thạch anh ngoại gắn sẵn trên chip

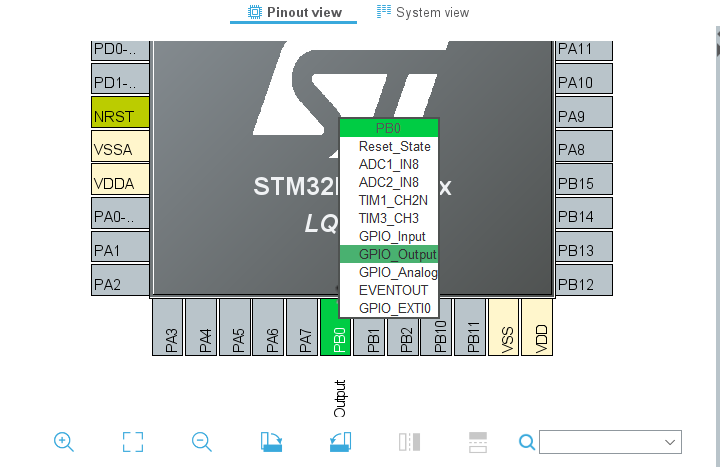


sau đó, vào tab **Clock configuration,** chọnHSE (nguồn thạch anh ngoại), tín hiệu sẽ đi qua bộ phân tần PLLCLK giúp chip hoạt động ở tần số tối đa

Tại mục **Input frequency**, các bạn điền “8” (thạch anh hàn sẵn trên board là loại 8Mhz). Sau đó chúng ta điền “72” tại mục “HCLK” (đây là tần số hoạt động tối đa của chip) và ấn Enter, đợi cho CubeIDE tự tính toán các thông số còn lại.



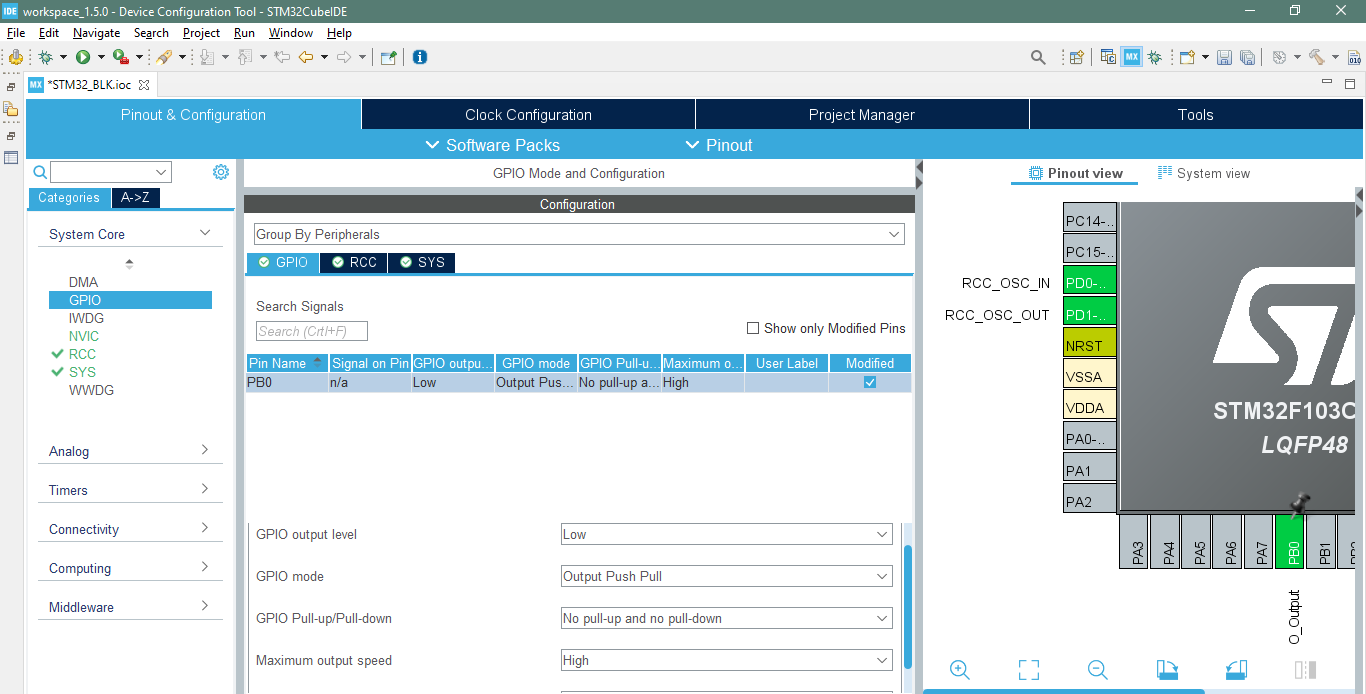
Bước 4: click chuột trái vào các chân từ PB0 đến PB7, chọn ô *GPIO\_Output*

******

Bước 5: click vào ô GPIO trong phần system core để cấu hình chân cho output

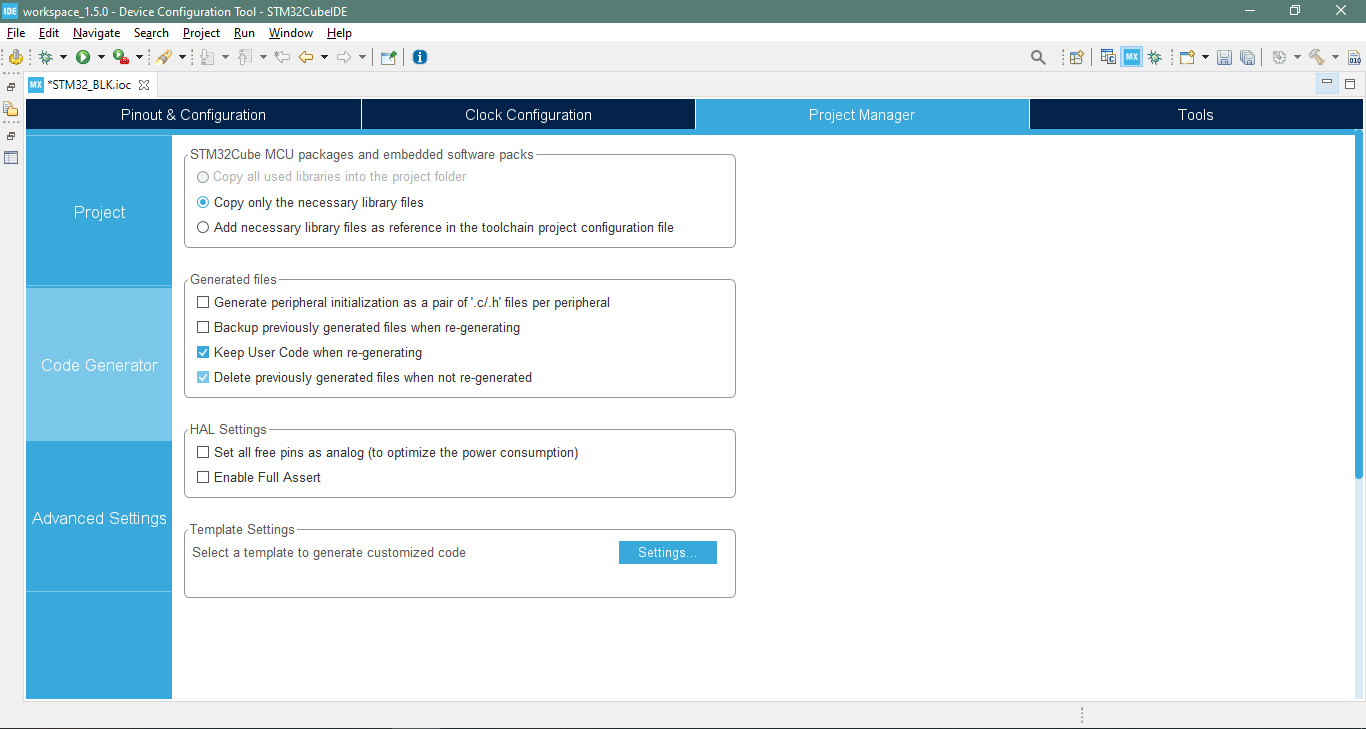
Cấu hình các chân GPIO như ở hình dưới:

* GPIO output level: “LOW” (Trạng thái ban đầu của led và sẽ được kích khi xuất mức “0” tại chân IO)
* 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: Đối với các dòng vi điều khiển có tốc độ xử lý nhanh từ vài chục MHz trở lên, thì chúng ta phải khai báo tốc độ dao động tại chân đó để đáp ứng với tốc độ xử lý của vi điều khiển. Hiện tại, chúng ta đang sử dụng nguồn dao động nội với tốc độ là 72MHz nên ở đây sẽ chọn “HIGH”



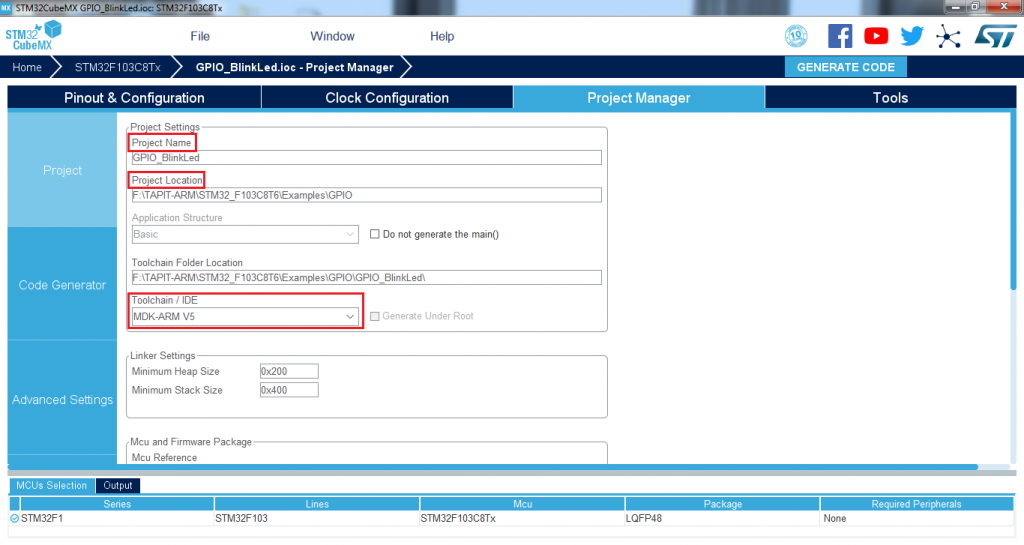
Bước 6: vào phần Project Manager để setting project và sinh code:

Ở cubeIDE thì ta không cần chọn môi trường sinh code nên sẽ vào thẳng phần **Code Generator**, chọn **“copy only necessary library files**” rồi bấm **“ctrl + s”** để sinh code



Còn nếu bạn sử dụng CubeMX và keilC để sinh code, thì ta sẽ làm như sau:

* Click project chọn Setting
* Đặt tên project và công cụ sử dụng để lập trình



Click vào Code Generator sau đó chọn copy các thư viện cần thiết và sinh code

4. hướng dẫn chương trình nhấp nháy led đơn

sau khi sinh code, ta sẽ bắt đầu viết chương trình nhấp nháy led. Ta sẽ thực hiện nhấp nháy 1 led đơn trước.

Ở đây sẽ có các hàm liên quan đến điều khiển các chân GPIO:

– **HAL\_GPIO\_TogglePin(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)** để đảo trạng thái của led PB0. Ở đây mình sẽ truyền vào 2 tham số, thứ nhất là Port cần sử dụng và tham số thứ 2 là chân IO cần sử dụng cụ thể:  HAL\_GPIO\_TogglePin(GPIOB,GPIO\_PIN\_0);

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

**HAL\_GPIO\_WritePin (GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_SET);**

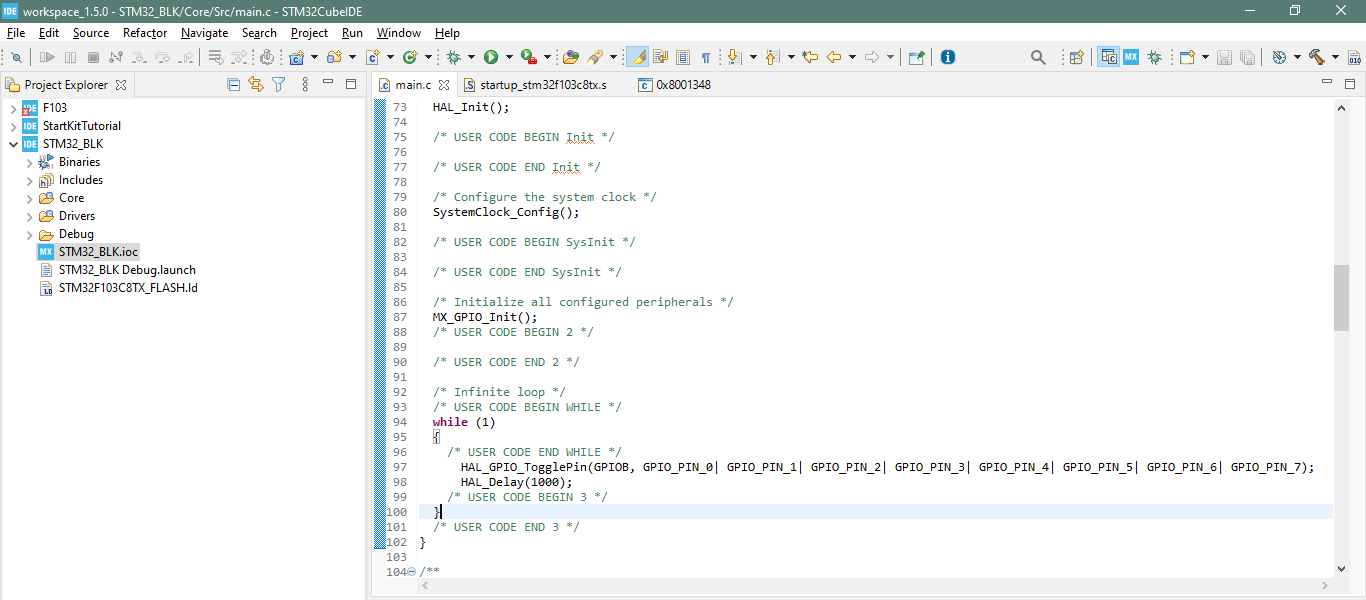
**HAL\_GPIO\_WritePin (GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);**

Trong vòng lặp **while (1)** ta sẽ tiến hành đảo trạng thái của các led 1s 1 lần:

****

Sau khi viết xong code, ta sẽ tiến hành build và nạp code vào vi điều khiển.

Ở đây mình sử dụng mạch nạp **ST-LINK V2** để kết nối với vi điều khiển. Sau khi kết nối thành công, ta bấm vào phần **Debug** có trên thanh công cụ và tiến hành debug. Nếu không có lỗi gì thì bấm F8 để vi điều khiển chạy chương trình

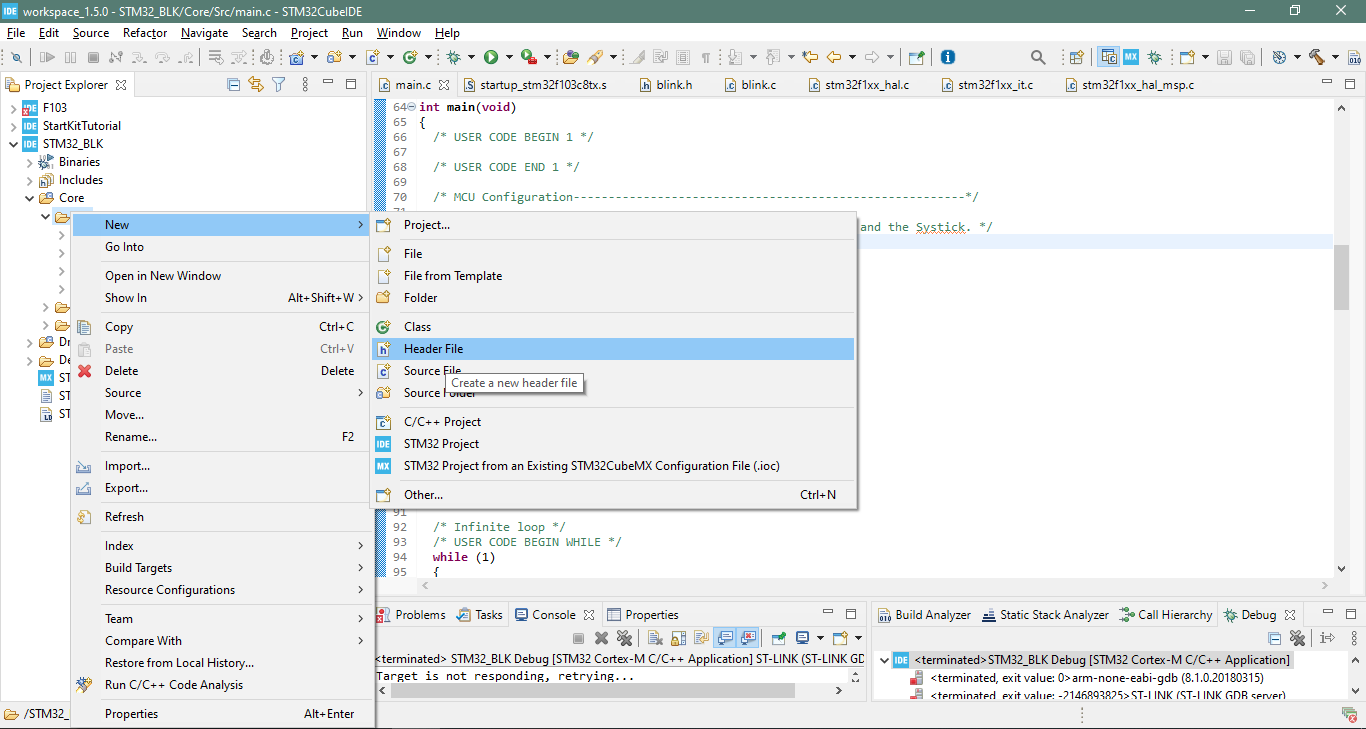


Sau khi thực hiện nhấp nháy được các led đơn thì ta có thể thực hành nháy led với nhiều hiệu ứng khác nhau.

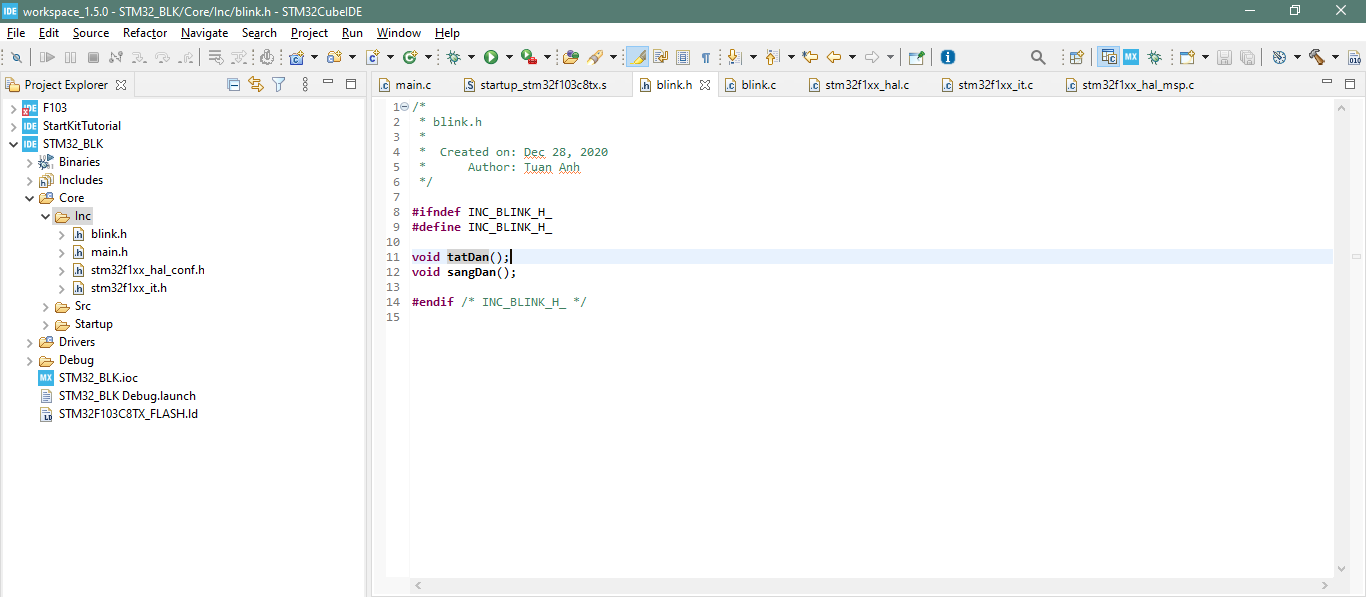
5. Ví dụ về hiệu ứng sáng dần và tắt dần các led

Ở ví dụ này mình sẽ hướng dẫn các bạn tạo 1 thư viện trong CubeIDE để chứa các hàm hiệu ứng nhấp nháy led.

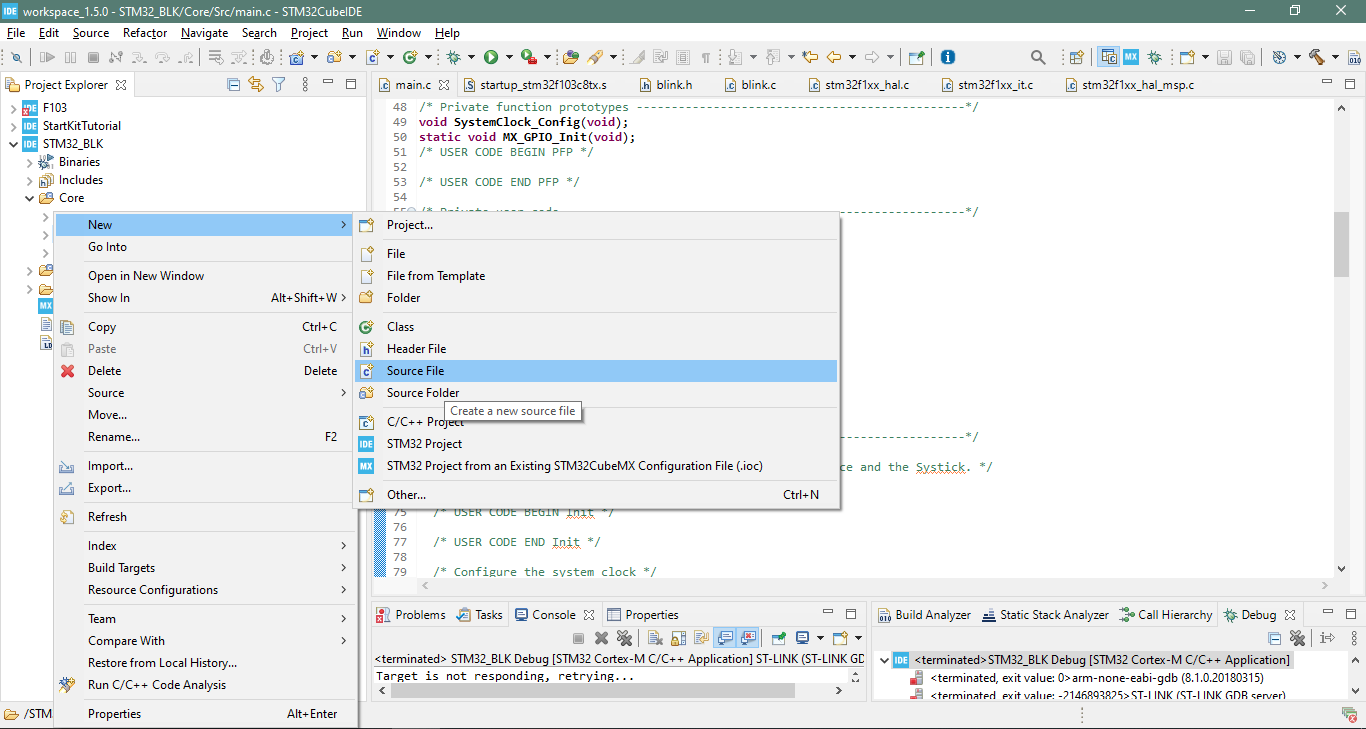
đầu tiên, click vào phần **Core → Inc**. Click chuột phải vào và chọn **New → Header file**



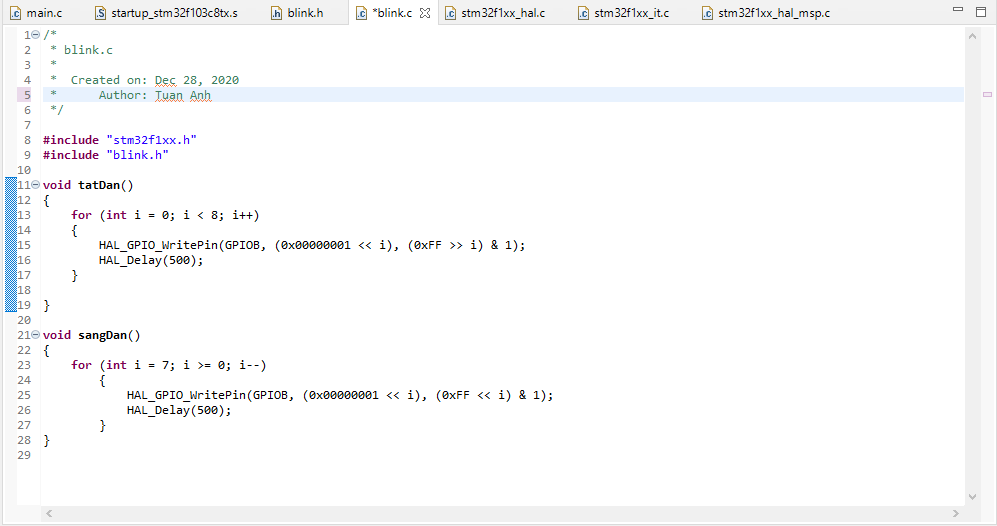
ở đây mình đặt tên là **blink.h.** Sau khi tạo xong file thư viện, mình sẽ viết các prototype của hàm sangDan() và tatDan() ở đây.



Tiếp theo, click chuột phải vào Core → Src, chọn New → Source file

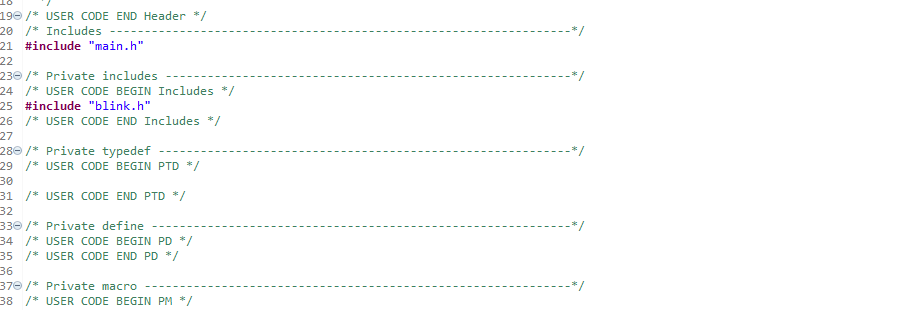


Sau khi tạo xong file **blink.c**, mình tiến hành viết 2 hàm sáng dần và tắt dần các led



ở đây mình chọn cách viết 0x00000001 thay vì GPIO\_PIN\_x là để thuận tiện cho việc sử dụng vòng for. 0x00000001 ở đây chính là giá trị địa chỉ 32bits của chân GPIO\_PIN\_0, việc dịch bit ở đây là tăng giá trị để trỏ đến chân tiếp theo.

Sau khi viết xong các hàm cần thiết, ta quay trở lại file **main.c,** ta thêm thư viện **blink.h** vừa viết vào



Trong vòng lặp **while(1)** ta gọi các hàm **sangDan()** và **tatDan(),** thêm hàm HAL\_Delay(500) ở giữa để tạo độ trễ



Như vậy là chứng ta đã có hiệu ứng tắt dần và sáng dần của 8 led đơn trên kit STM32 start. Từ những kiến thức cơ bản này, ta có thể làm nhiều hiệu ứng thú vị khác nữa như là hiệu ứng sáng từng led, sáng xen kẽ,…