|  |  |  |  |
| --- | --- | --- | --- |
| Họ và tên | Phạm Quốc Tiến | Nguyễn Đức Trung | Bùi Thái Toàn |
| MSSV | 22521472 | 22521564 | 22521485 |
| Họ và tên | Trịnh Thành Trung | Trần Quốc Trinh | Nhật Tin |
| MSSV | 22521570 | 22521542 | 22521479 |
| Họ và tên | Nguyễn Hữu Bảo Trọng |  |  |
| MSSV | 22521545 |  |  |

LAB 1: LÀM QUEN VỚI KIT  
STM32F4 DISCOVERY

# CHUẨN BỊ

|  |  |
| --- | --- |
| Phần cứng  * KIT STM32F4 Discovery for STM32F429 MCU.  Phần mềm  * STM32CubeIDE: sử dụng để lập trình, build, nạp và debug code.  Tài liệu User Manual: tham khảo [tại đây](https://www.st.com/en/evaluation-tools/32f429idiscovery.html).  Datasheet: tham khảo [tại đây](https://www.st.com/resource/en/datasheet/stm32f429ig.pdf). | Hình .. KIT STM32F4 Discovery |

# HƯỚNG DẪN

## Khảo sát phần cứng

Phần thực hành Thiết kế Hệ thống nhúng được thực hiện trên KIT STM32F4 Discovery. Trước khi bắt đầu, hãy khảo sát các thông tin phần cứng về kit thực hành này bằng cách điền vào bảng sau:

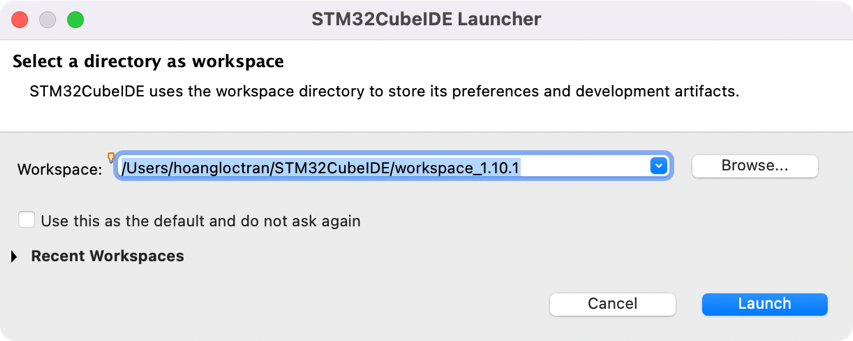
|  |  |
| --- | --- |
|  | **Thông tin phần cứng** |
| Tên KIT | STM32F429I-DISC1 |
| Tên MCU | STM32F429ZIT6 |
| Core | Arm® Cortex®‑M4 |
| Xung nhịp tối đa | 180 MHz |
| Dung lượng bộ nhớ | 2 MB bộ nhớ flash  256+4 KB SRAM  64 KB CCM |
| Nguồn cấp cho board | 1.7V – 3.6V |
| Số chân của chip | 168 I/O ports |

## Cài đặt công cụ thực hành

[STM32CubeIDE](https://www.st.com/en/development-tools/stm32cubeide.html) là một nền tảng cao cấp để phát triển code C/C++ với các chức năng cấu hình các thành phần ngoại vi, sinh code, biên dịch code và debug các chức năng của các vi điều khiển và vi xử lý thuộc dòng STM32. Phần mềm này có thể được cài đặt và sử dụng trên các hệ điều hành khác nhau như Windows, OSX và Linux.

Điểm đặc biệt của STM32CubeIDE là nó tích hợp chức năng cấu hình STM32 và khởi tạo các project của phần mềm STM32CubeMX từ đó tạo ra một phần mềm phát triển all-in-one. Sau khi lựa chọn MCU, tiền cấu hình hệ thống, chọn các project từ ví dụ mẫu, STM32CubeIDE sẽ tự động tạo ra code (sinh code) để cấu hình trước các thành phần cần thiết cho project (ví dụ tự động cấu hình các chân GPIO, các giao thức kết nối...). Chi tiết các bước cấu hình được thực hiện như sau:

* Bước 1: Khởi động STM32CubeIDE và chọn thư mục làm việc (workplace)



Hình .. Chọn thư mục làm workplace cho project

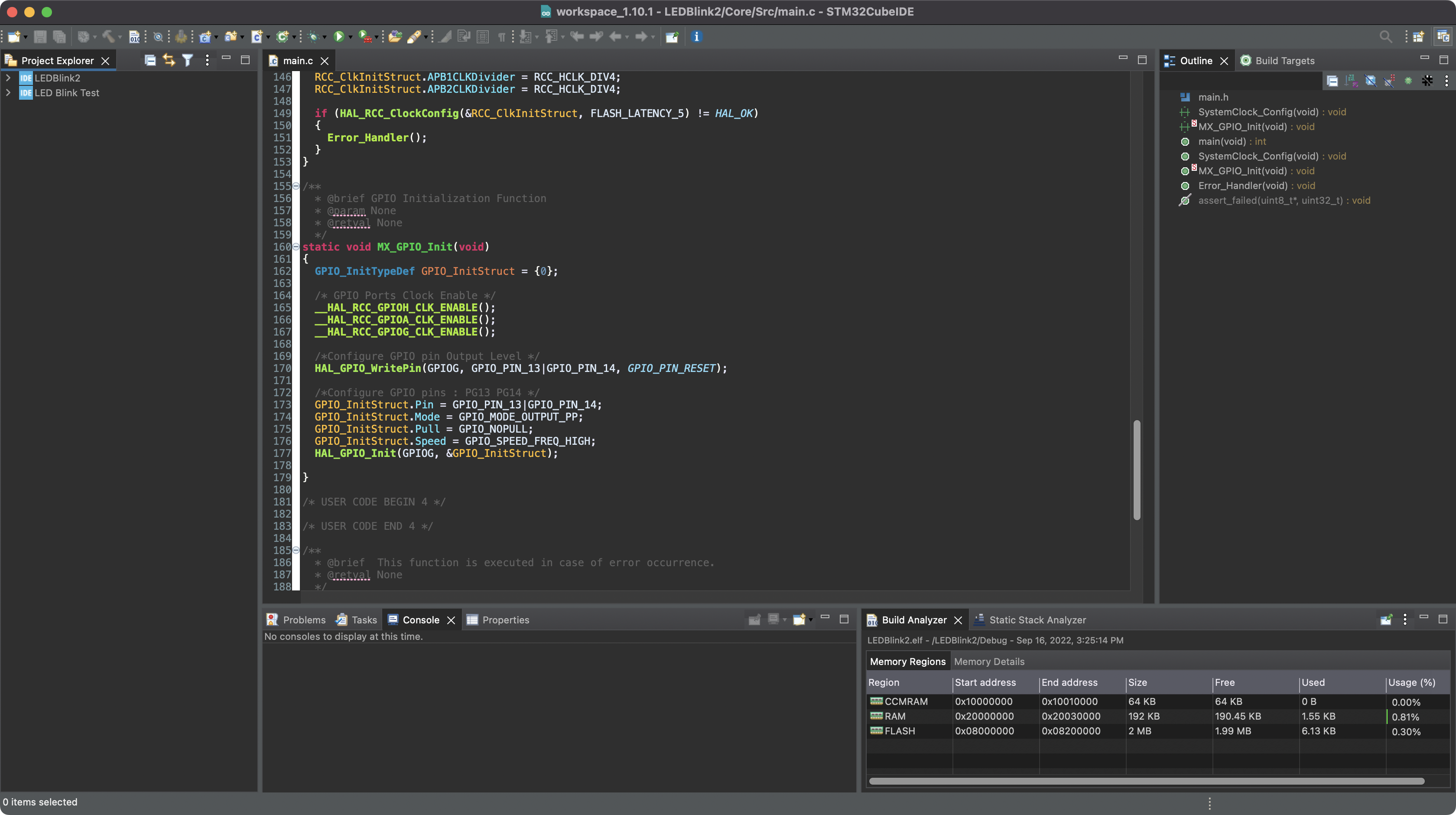
* Bước 2: Trong lần khởi động đầu tiên, STM32CubeIDE sẽ hiển thị Information Center, sinh viên có thể bắt đầu tạo project mới bằng cách chọn nút **Start new STM32 project**.

Graphical user interface, website

Description automatically generated

Hình .. Information Center xuất hiện trong lần đầu khởi động chương trình

Trong các lần khởi động tiếp theo, chương trình sẽ tự động mở project gần nhất trong workspace, để tạo project mới từ giao diện này, chúng ta chọn **File** " **New** " **STM32 Project**.



Hình .. Giao diện chính của chương trình trong các lần khởi động sau

* Bước 3: Cấu hình project sử dụng MCU **STM32F429ZIT6**

Ta đã biết MCU được sử dụng trên KIT STM32F4 Discovery là STM32F429ZIT6, do đó để tìm nhanh chóng thông tin của KIT trong phần mềm, ta gõ tên của MCU vào ô Commercial Part Number. Sau khi kiểm tra các thông tin của MCU đã đúng, ta bấm Next để sang bước tiếp theo.

A screenshot of a computer

Description automatically generated

Hình .. Cấu hình và chọn đúng MCU cho project

* Bước 4: Điền thông tin về project và bấm **Next**
* Bước 5: Lựa chọn option "**Add necessary library files as reference in the toolchain project configuration file"** và bấm **Finish**.

|  |  |
| --- | --- |
| Graphical user interface, application  Description automatically generated  Hình .. Điền thông tin project | Hình .. Lựa chọn Code Generator Options |

Sau khi đã lựa chọn MCU và khởi tạo các thông tin của project, chương trình sẽ mở giao diện để cấu hình cho project bao gồm (i) Lựa chọn và Cấu hình Pinout, (ii) Cấu hình xung Clock, (iii) Quản lý project, (iv) Các công cụ khác.

* Bước 6: Lựa chọn và Cấu hình Pinout.
  + Đầu tiên là lựa chọn và cấu hình Pinout, ta sẽ chọn các pin cần sử dụng và cấu hình cho nó. Ví dụ, trong project chớp/tắt led, ta cần xuất tín hiệu ra LED 03 thì ta cần cấu hình pin gắn với LED này là GPIO\_Output.

|  |
| --- |
| Tìm pin gắn với LED 03 và cấu hình cho pin đó là output bằng cách bấm chọn Pin và chọn GPIO\_Output? |

* + Bấm chọn GPIO trong phần System Core, chọn pin gắn với LED 03 trong danh sách các GPIO, trong bảng Configuration cài đặt:
    - GPIO Output Level: Low (mặc định đèn tắt)
    - GPIO Mode: Output Push Pull
    - GPIO Pull-up/Pull-down: No pull-up and no pull-down
    - Maximum output speed: High
  + Tiếp theo, trong phần System Core, chọn RCC, set High Speed Clock (HSE) là Crystal/Ceramic Resonator để sử dụng clock ngoài tốc độ cao.
  + Tiếp tục trong phần System Core, chọn SYS và chỉnh option Debug là Serial Wire
* Bước 7: Cấu hình xung Clock
  + STM32CubeIDE cung cấp khả năng cấu hình mạnh mẽ và dễ sử dụng thông qua giao diện đồ họa, để điều chỉnh xung Clock hệ thống, chương trình cung cấp sơ đồ về cách chọn Clock cũng như cài đặt các bộ chia, prescaler để cho ra được thông số Clock mong muốn

Diagram

Description automatically generated

Hình .. Sơ đồ cấu hình xung clock cho hệ thống

* + Thông thường, chúng ta sẽ sử dụng clock ngoại tốc độ cao (HSE) để tạo xung cho hệ thống, do đó, tại vị trí PLL Source Mux, ta chọn HSE

|  |
| --- |
| Dựa trên sơ đồ cấu hình clock, hãy cài đặt Input Frequency, /M, \*N, /P, System Clock Mux, APB1 Prescaler, APB2 Prescaler sao cho:  SYSCLK = 180MHz  APB1 = 45MHz  APB2 = 90MHz |

* + Trong hầu hết các trường hợp, nếu không quan tâm đến vấn đề năng lượng hay xung clock thì ta có thể đặt SysClock là 180MHz (tốc độ tối đa của MCU)
  + Trong các trường hợp khác như tiết kiệm năng lượng, chỉnh xung để giao tiếp với các thiết bị khác (như tốc độ baudrate) thì cần phải chỉnh lại clock để đồng bộ xung
* Bước 8: Khởi tạo hệ thống
  + Nếu không cần chỉnh sửa gì thêm về thông tin project, ta có thể bỏ qua phần Quản lý project và Các công cụ khác, sau đó bấm **File** " **Save**. Hệ thống sẽ hỏi có muốn sinh code (generate Code) không và ta sẽ chọn **Yes** để chương trình tự động tạo các đoạn code cấu hình và khởi tạo hệ thống.
  + Như vậy, ta đã có được một project đã cấu hình sẵn các thành phần cần dùng (các thành phần này có thể là các GPIO, Timers, các chuẩn giao tiếp, xung clock...) và chỉ cần tập trung vào bước lập trình.
* Bước 9: Lập trình
  + Trong tab Project Explorer ta sẽ thấy lúc này đã được thêm các file source code vào, file main.c sẽ nằm trong thư mục <Tên Project>\Core\Src\main.c. Ngoài ra, ta còn thấy một file có tên <Tên Project>.ioc, đây là file dùng để lưu cấu hình hệ thống mà ta vừa thực hiện từ Bước 6 đến Bước 8. Nếu cần cấu hình lại, ta chỉ cần dounble-click vào file này và bắt đầu chỉnh lại theo nhu cầu.
  + Trong file main.c, ta sẽ thấy các đoạn code dùng để khởi tạo (initialization) hệ thống đã được thêm vào sẵn. Để thực hiện việc chớp/tắt LED, thêm 2 dòng code sau vào vòng lặp chính của chương trình:

|  |
| --- |
| *//Thay X và Y bằng Pin tương ứng đã chọn ở trên.*  **HAL\_GPIO\_TogglePin**(GPIO*X*, GPIO\_PIN\_*Y*);  **HAL\_Delay**(1000); |

A screenshot of a computer

Description automatically generated with medium confidence

Hình .. Chương trình được sinh ra sau khi được tự động cấu hình các thành phần cần thiết

* Bước 10: Build và Nạp/Debug chương trình
  + Build chương trình: Chọn **Project** " **Build All**
  + Nạp nhanh chương trình: Right-Click vào tên project, chọn Run As " STM32 C/C++ Application. Chọn OK trong cửa sổ hiện ra.
  + Debug chương trình: Right-Click vào tên project, chọn **Debug As** " **STM32 C/C++ Application**. Chương trình sẽ hỏi về việc "**Switch to perspective**" " chọn **Switch** để chuyển sang giao diện debug. Với chức năng Debug, chúng ta có thể quan sát sự thay đổi của các biến, các thanh ghi, các GPIO và các thông số khác trên board.

# BÀI TẬP

**Bài tập 1:** Trình bày về các đặc điểm Timer trên KIT STM32F4 Discovery (số lượng Timer, các chế độ hoạt động, cách cấu hình, cách cài đặt)

* **Có 14 Timer**
* **Có các chế độ gồm: input capture, output compare, pwm generation, one-pulse mode output, DMA request generate, DAC trigger, wavefrom generation, auto-reload, up/down counter, prescaler, watchdog.**
* **Sử dụng công thức để tính giá trị của prescaler và giá trị của ARR (counter period):**

**Bài tập 2:** Sử dụng Timer để viết chương trình chạy 2 LED 3 và LED 4 cùng lúc như sau:

* LED 3 chớp/tắt với chu kỳ 01 giây
* LED 4 chớp/tắt với chu kỳ 750ms

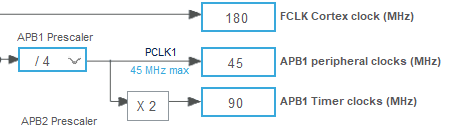
*Gợi ý:*

* *Tham khảo phần 3.22 trong Datasheet để biết chi tiết cách sử dụng Timer trên STM32F429xx.*
* *Tham khảo cách tính toán các thông số cho timer* [*tại đây*](https://deviot.vn/tutorials/stm32f4.62153513/lap-trinh-timer-che-do-counter-de-tao-delay.58829201) *(lưu ý, không thực hiện theo 100% bài hướng dẫn do công cụ và chip khác nhau).*
* *Sinh viên có thể chạy example TIM\_TimeBase khi chọn ví dụ trong phần tạo project.*

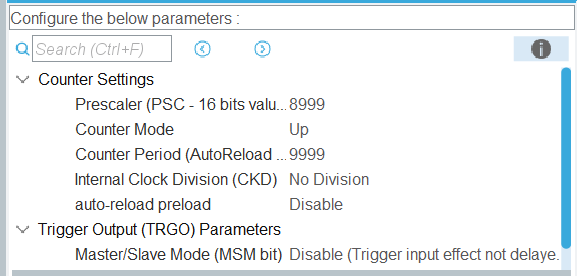
**Ý tưởng: nhóm sẽ sử dụng timer với chế độ ngắt cho mỗi led**

**Setup:**

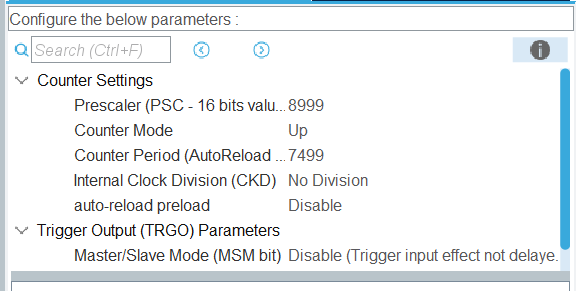
**Nhóm sẽ sử dụng timer 2 cho LED 3 và timer 3 cho LED 4**

**Do timer 2 và timer 3 đều nhận xung từ APB1 nên Prescaler của cả hai sẽ bằng 8999  
**

**Thời gian chớp tắt của LED 3 là 1s nên nhóm set timer 2 như sau**

****

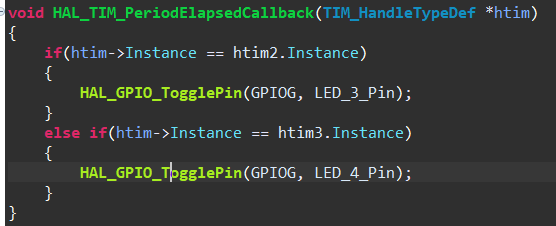
**Thời gian chớp tắt của LED 4 là 750ms nên timer 3 sẽ được set như sau**

****

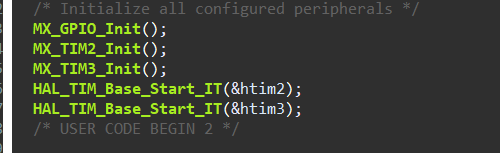
**Và kích hoạt ngắt trên hai timer**

****

**Hàm callback**

****

**Hàm main**

****