Bài thực hành 3

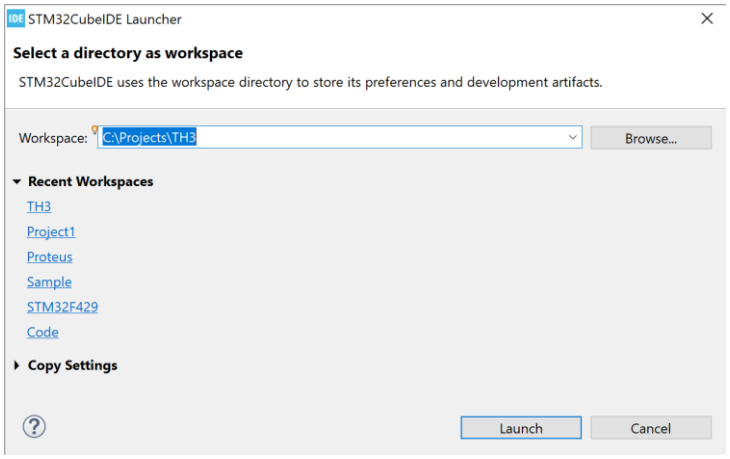
XÂY DỰNG ỨNG DỤNG TRÊN HỆ NHÚNG ARM (Online)

1. Mục đích

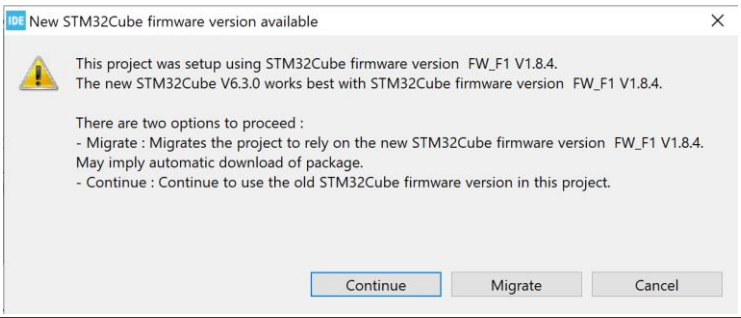
- Tìm hiểu CPU ARM Cortex M3 và STM32F103.
- Tìm hiểu bộ công cụ STM32CubeIDE.
- Lập trình các ngoại vi cơ bản của STM32F103.
- Xây dựng ứng dụng demo.

2. Chuẩn bị

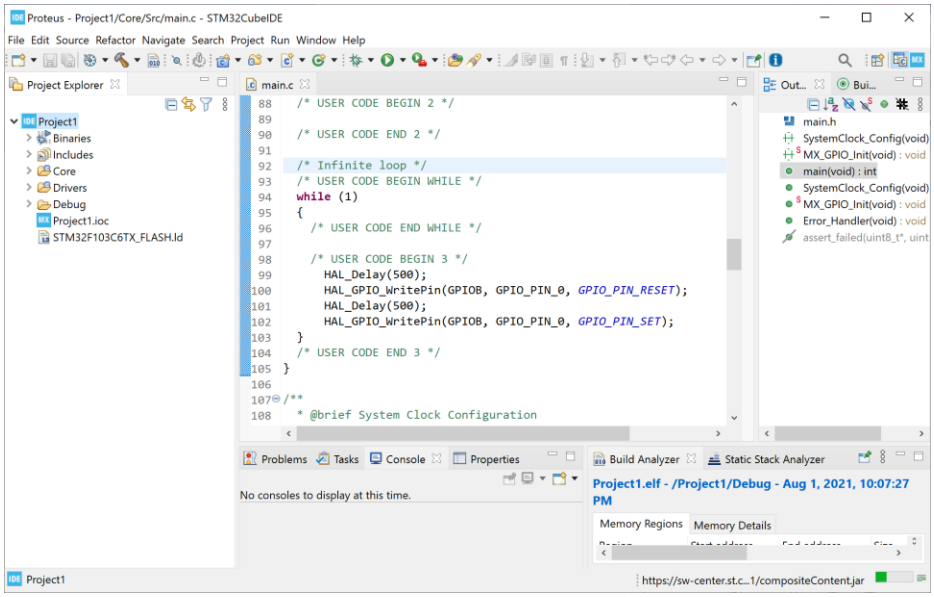
- Cài đặt môi trường làm việc:
  + Cài Java Runtime nếu chưa có.
  + Proteus 8.9 (hoặc bản mới hơn).
  + Tải STM32CubeIDE 1.7.0 (có trong thư mục **Files\Bài thực hành số 3**, hoặc download từ trang chủ https://www.st.com/).
  + Tạo 1 thư mục mới để chứa nội dung bài Thực hành 3. Tải và giải nén mã nguồn mẫu vào thư mục vừa tạo.
  + Double click file TH3\Project1\.project để mở file với STM32CubeIDE, phần mềm sẽ yêu cầu chọn thư mục chứa workspace. Chọn đường dẫn workspace tới thư mục TH3.



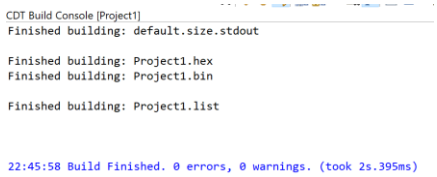
  + STM32CubeIDE sẽ chạy và load project đã có sẵn. Sau đó hộp thoại yêu cầu tải thư viện hỗ trợ cho CPU STM32F1 sẽ hiện ra. ➔ Chọn Migrate để phần mềm tự tải các thư viện cần thiết.



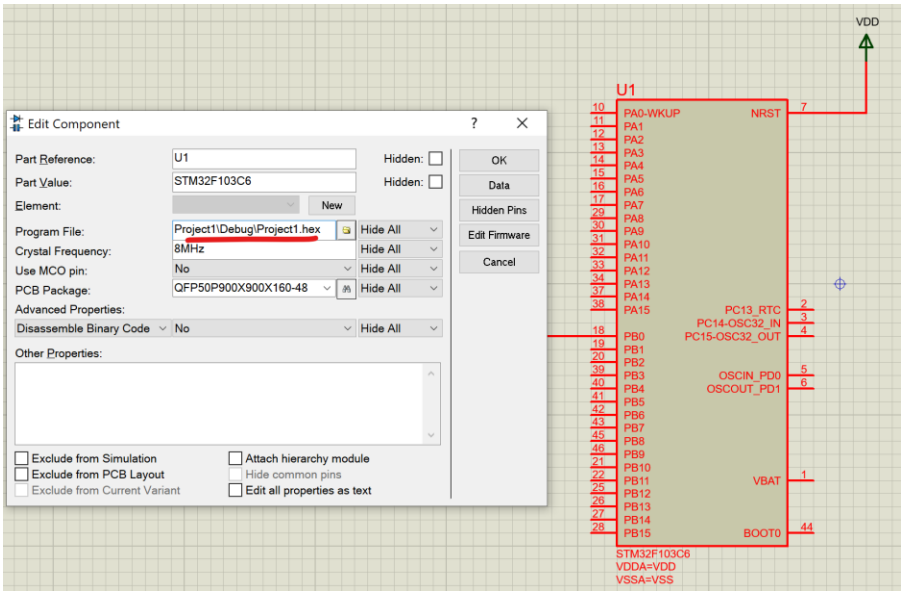
+ Sau khi tải xong thư viện, STM32CubeIDE sẽ load project. Giao diện hoàn chỉnh như hình dưới đây.



+ Bấm nút Build trên toolbar, hoặc chọn menu Project → Build project để biên dịch project ra firmware. Nếu kết quả như ở dưới là thành công và môi trường phát triển được cấu hình đúng.



+ Mở file TH3.pdsprj với Proteus. Double-click vào chip U1 rồi đặt đường dẫn đến Program file tương ứng với file Project1.hex tạo ra từ bước Build ở trên.



+ Bắt đầu chạy mô phỏng trên Proteus. Nếu thấy đèn LED trên mạch nhấp nháy là chương trình chạy đúng.

Chú ý: *mạch và chương trình mô phỏng này cần rất nhiều tài nguyên. Nếu chạy trên laptop thì cần để máy ở chế độ Best performance và dùng nguồn ngoài.*

3. Thực hành

3.1 Tìm hiểu CPU STM32F103C6

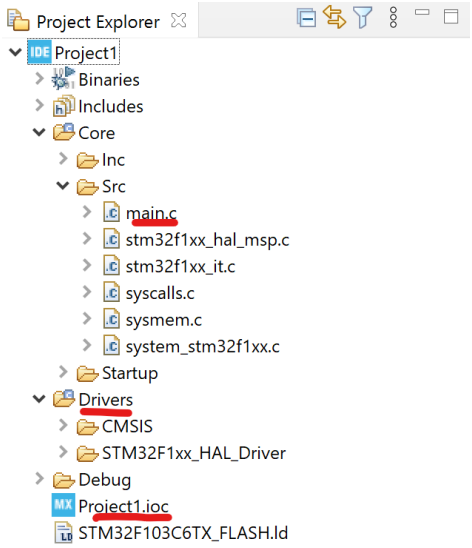
- Đọc file datasheet (stm32f103c6.pdf)

- Xác định các thông số sau

Dung lượng ROM	
Dung lượng RAM	
Tần số clock tối đa của CPU	
Điện áp hoạt động	
Số chân vào ra	
Số bộ timer	
Số cổng USART	
Dòng điện sử dụng khi CPU chạy ở 72 MHz	
Dòng điện sử dụng khi CPU chạy ở 8 MHz	
Danh sách chân ADC input	
Danh sách chân ngắt ngoài	
Các chân RX/TX của USART1	

3.2. Tìm hiểu mã nguồn mẫu

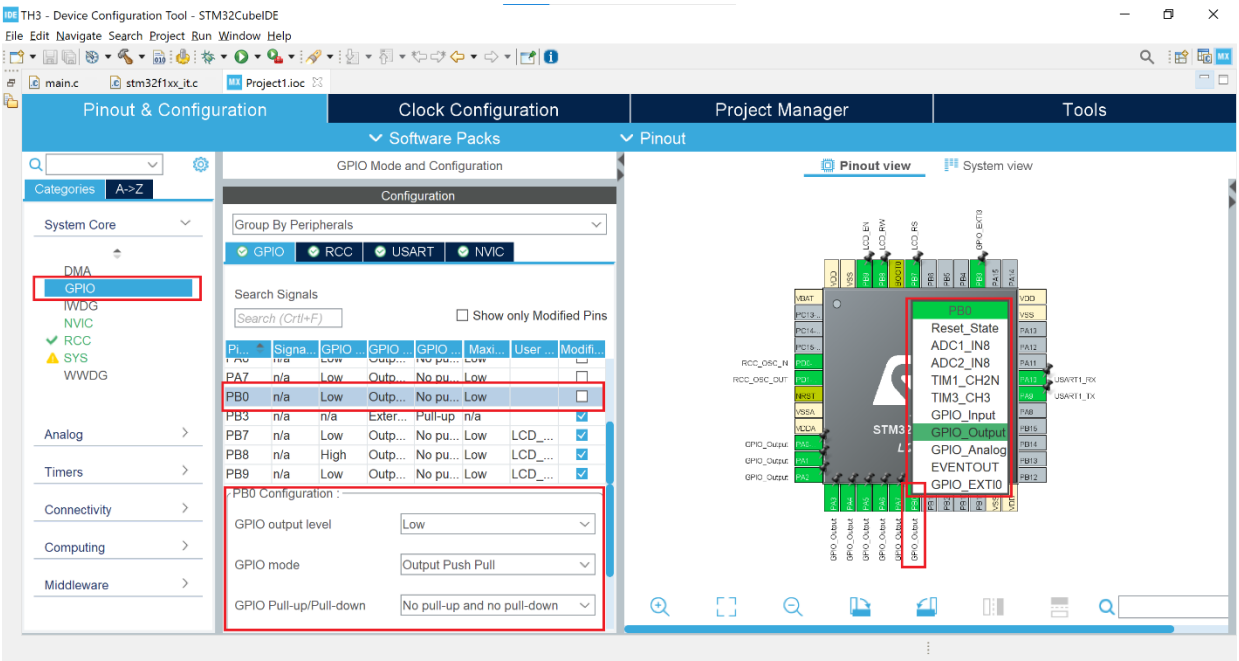
- Quan sát cấu trúc project và cho biết ý nghĩa các file/thư mục đánh dấu đỏ dưới đây:



3.3. Thiết lập và lập trình GPIO (bài mẫu).

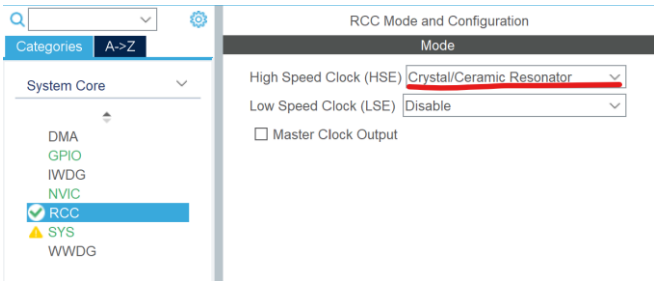
Vi điều khiển nhân ARM có thể có rất nhiều chân vào ra và mỗi chân thường được dồn kênh với rất nhiều tính năng khác nhau. Việc cấu hình chế độ hoạt động cho từng tính năng, từng chân vào ra rất phức tạp. Do đó, các nhà sản xuất (ví dụ: Texas Instrument, STMicroelectronics…) đều cung cấp các công cụ tạo project và sinh mã nguồn cấu hình thiết bị để giảm tải cho lập trình viên.

- Double-click vào file Project1.ioc trong cửa sổ STM32CubeIDE để mở công cụ cấu hình project.
- Quan sát cấu hình chân PB0 trên layout của chip được đặt là GPIO_Output (bên phải màn hình), và cấu hình chi tiết trong mục System core → GPIO bên trái màn hình.

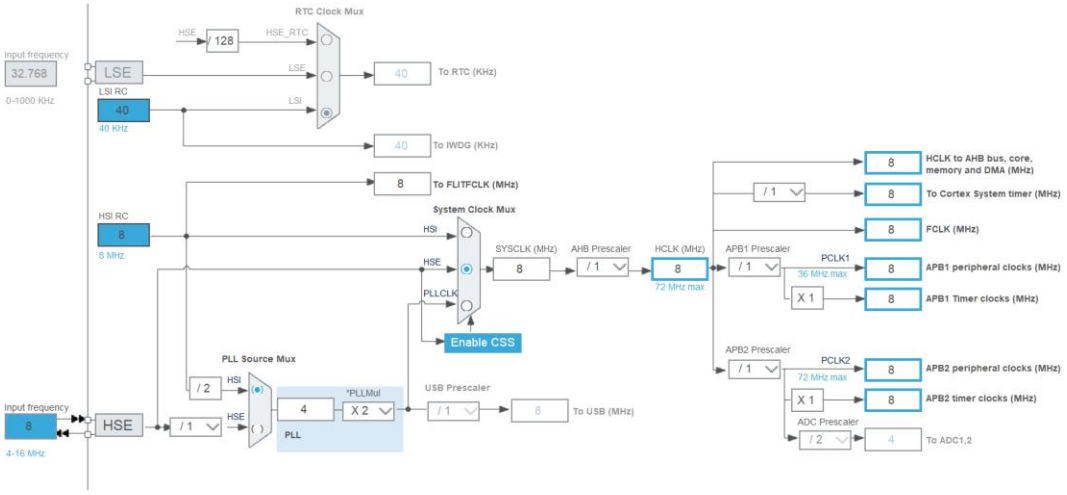


Cấu hình chế độ output cho chân PB0

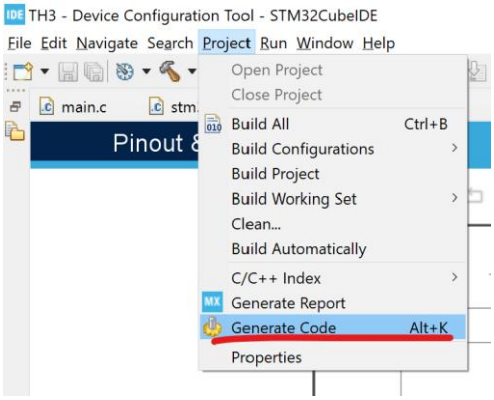
- Chọn mục System Core →RCC bên trái và đặt cấu hình nguồn clock ngoài là Crystal/Ceramic Resonator.
-



- Chuyển sang tab Clock Configuration, đặt Input frequency là 8 MHz và HCLK là 8 MHz.



- Vào menu Project → Generate Code (Alt+K), lúc này toàn bộ cấu hình sẽ được sử dụng để sinh mã nguồn tương ứng.



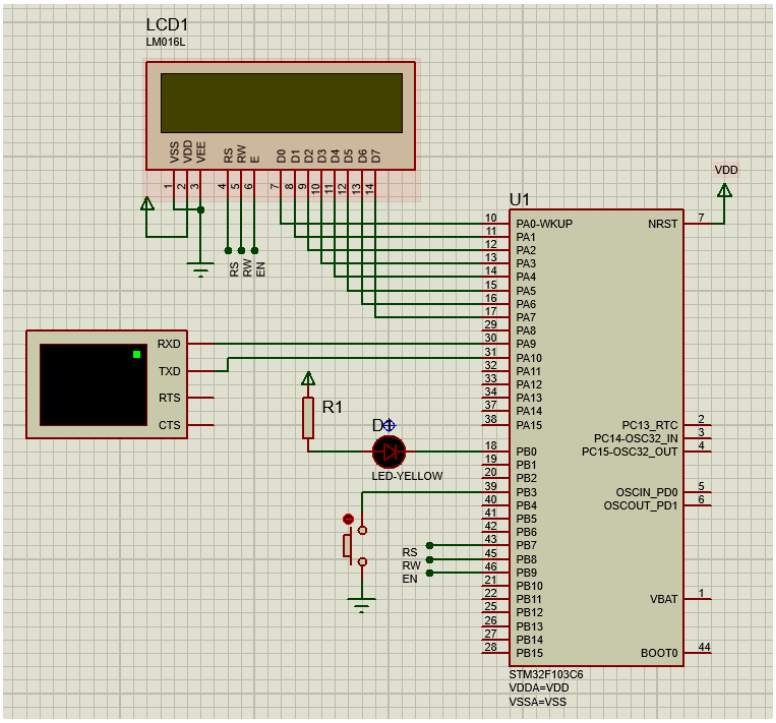
- Vào menu Project → Generate Code (Alt+K), lúc này toàn bộ cấu hình sẽ được sử dụng để sinh mã nguồn tương ứng.
- Kiểm tra file main.c. Cho biết:
  - Chân PB0 thuộc cổng B được cấu hình ở dòng code nào, trong hàm nào?
  - Chức năng của vòng lặp main loop trong hàm main() là gì?

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
  HAL_Delay(500);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET);
  HAL_Delay(500);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_SET);
}
/* USER CODE END 3 */
```

3.4. Cấu hình ngoại vi

Vẽ mạch trên Proteus như sau:

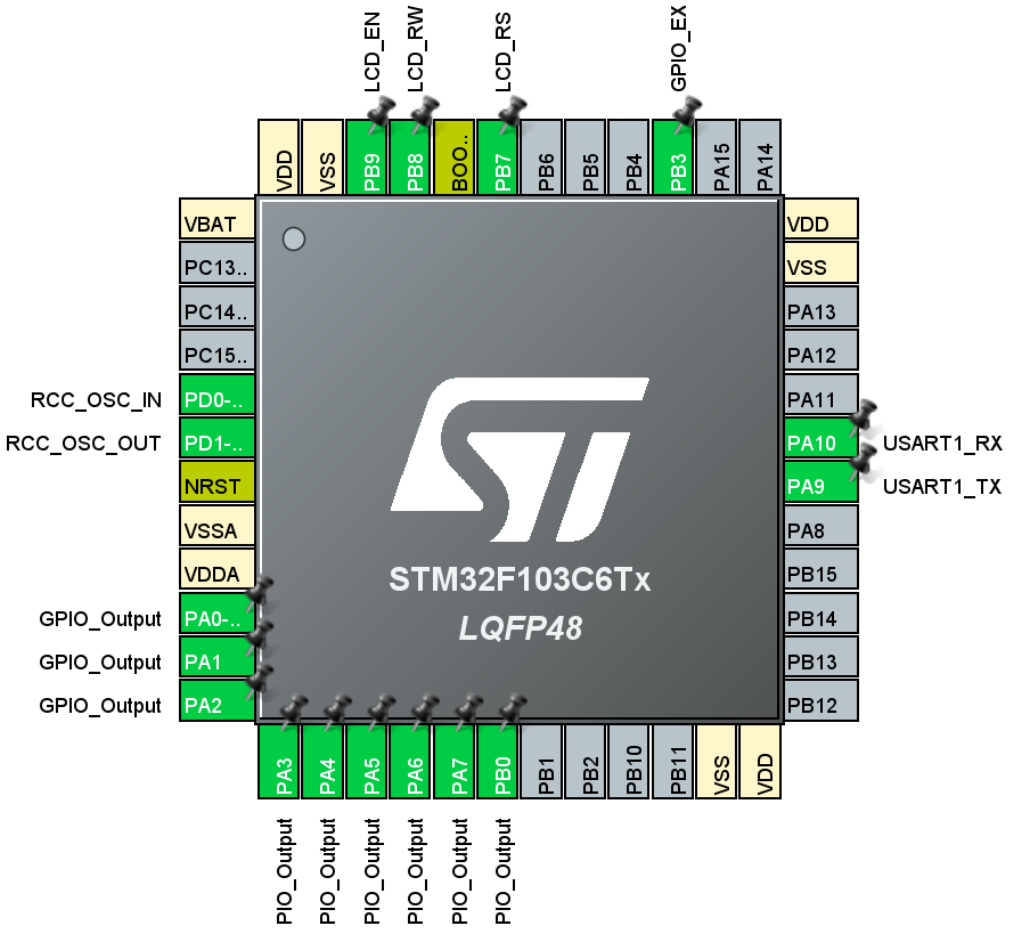


Cấu hình các ngoại vi trên STM32F103C6 gồm:

- Chân GPIO điều khiển đèn LED (đã làm ở trên)
- Các chân GPIO để nối với màn hình LCD

- Chân ngắt ngoài nối với nút bấm
- Chân RX, TX và bộ điều khiển vào ra nối tiếp USART1
- Bộ định thời TIM2

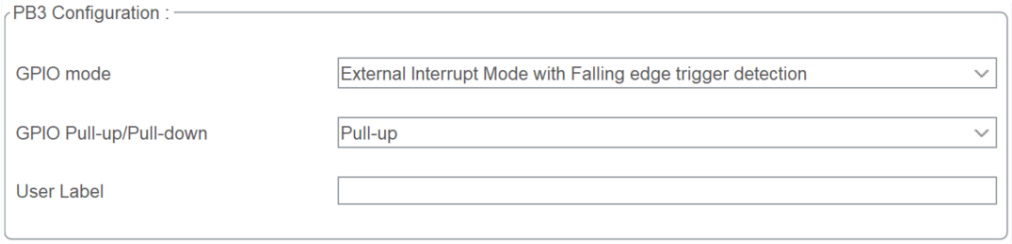
Trong phần IC layout chọn cấu hình cho các chân:



- Vào mục System Core → GPIO đặt các cấu hình:

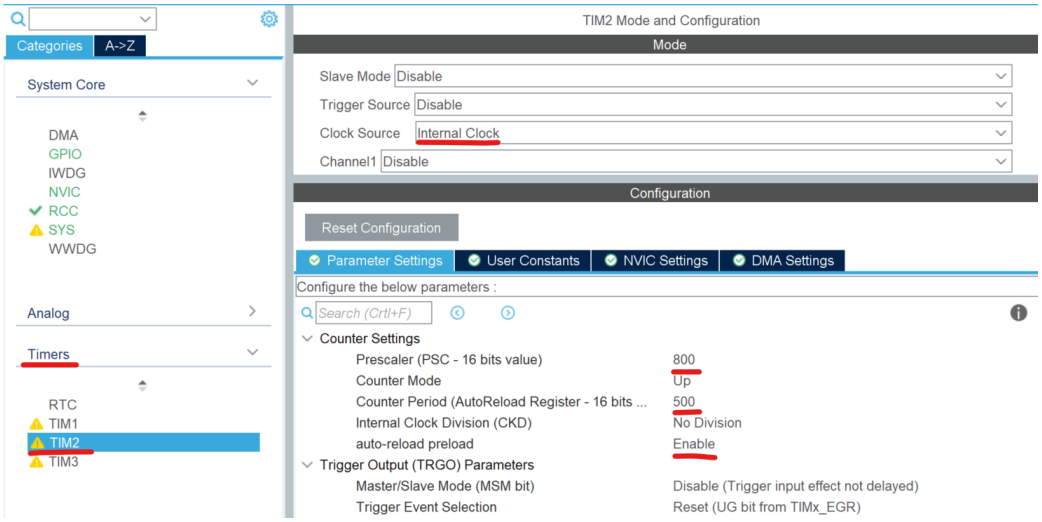
Pin N...	Signal on...	GPIO out...	GPIO mo...	GPIO Pul...	Maximum...	User Label	Modified
PA0-WK...	n/a	Low	Output P...	No pull-u...	Low		☐
PA1	n/a	Low	Output P...	No pull-u...	Low		☐
PA2	n/a	Low	Output P...	No pull-u...	Low		☐
PA3	n/a	Low	Output P...	No pull-u...	Low		☐
PA4	n/a	Low	Output P...	No pull-u...	Low		☐
PA5	n/a	Low	Output P...	No pull-u...	Low		☐
PA6	n/a	Low	Output P...	No pull-u...	Low		☐
PA7	n/a	Low	Output P...	No pull-u...	Low		☐
PB0	n/a	Low	Output P...	No pull-u...	Low		☐
PB3	n/a	n/a	External I...	Pull-up	n/a		☑
PB7	n/a	Low	Output P...	No pull-u...	Low	LCD_RS	☑
PB8	n/a	High	Output P...	No pull-u...	Low	LCD_RW	☑
PB9	n/a	Low	Output P...	No pull-u...	Low	LCD_EN	☑

Cấu hình các chân GPIO

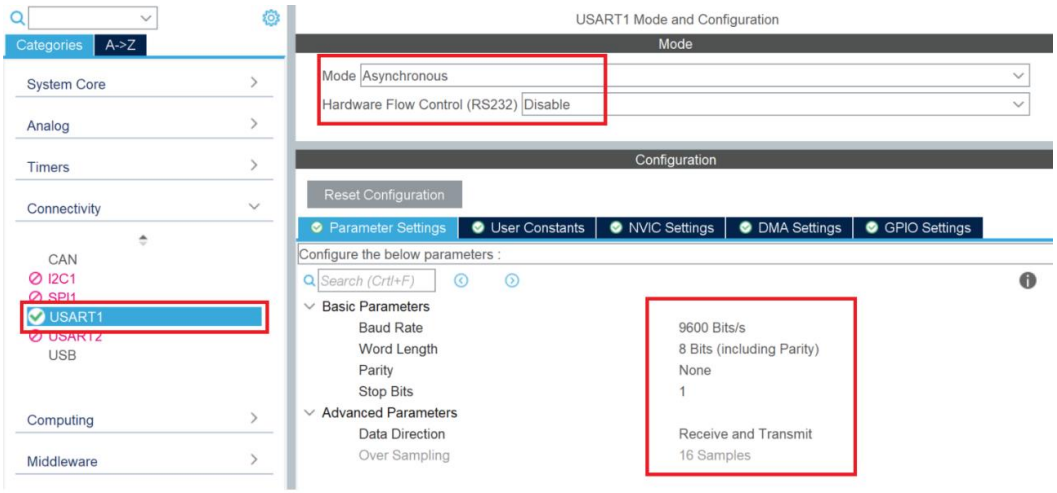


Cấu hình chân ngắt nối với nút bấm

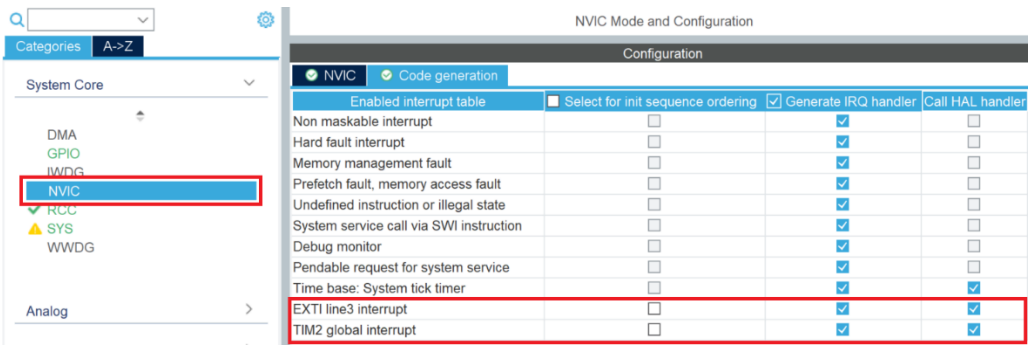
- Vào mục Timers → TIM2 đặt các cấu hình:

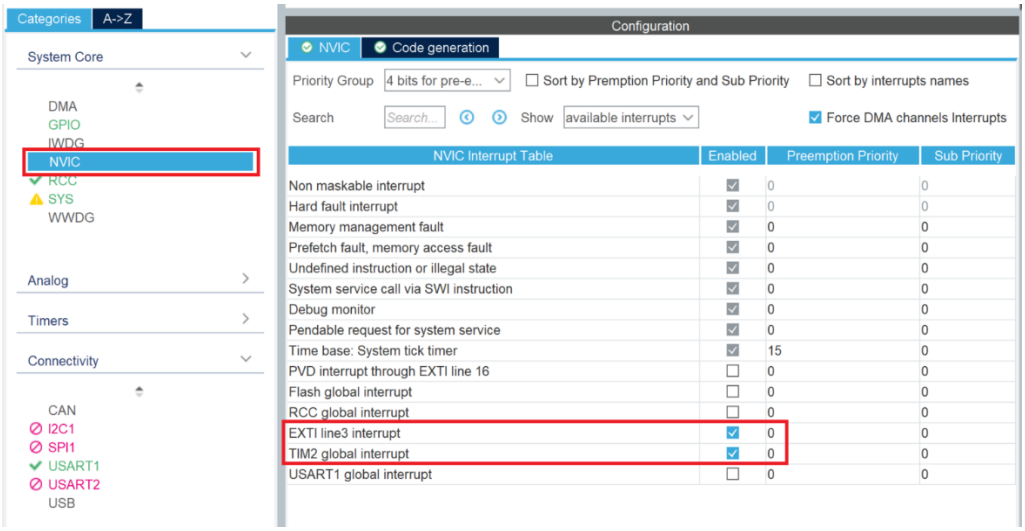


- Vào mục Connectivity → USART1 đặt các cấu hình:



- Vào mục System Core → NVIC đặt cấu hình cho các vector ngắt:





Vào menu Project →Generate Code hoặc bấm Alt+K, tất cả cấu hình vừa tạo sẽ được chuyển thành mã nguồn tương ứng.

Bấm Build và kiểm tra để xác nhận project được biên dịch thành công, không có lỗi phát sinh.

3.5. Lập trình xử lý nút bấm

- Xác định khai báo và mô tả của các hàm xử lý ngắt ngoài 3
- Lập trình để:
  - Loại bỏ phần điều khiển đèn LED khỏi main loop.
  - Bật/tắt đèn bằng cách bấm nút.

```
/* USER CODE BEGIN 4 */
/* STM32F1xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32f1xx.s).                    */
/*****************************************************************************/

/**
  * @brief This function handles EXTI line3 interrupt.
  */
void EXTI3_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI3_IRQn 0 */

  /* USER CODE END EXTI3_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_3);
  /* USER CODE BEGIN EXTI3_IRQn 1 */
  HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_0);

  /* USER CODE END EXTI3_IRQn 1 */
}
```

3.6. Lập trình xử lý ngắt timer

Khác với ngắt ngoài là cấu trúc đơn giản, chỉ cần 1 hàm xử lý, thì timer và USART cần được quản lý và điều khiển bởi các driver. Các driver này được cung cấp trong thư viện STM32Fxxx và được gọi thông qua các cấu trúc + các hàm HAL_xxx.

- Xác định đoạn code cấu hình TIM2
- Xác định handler của TIM2
- Lập trình 2 đoạn code quan trọng
  - Lệnh bắt đầu cho TIM2 hoạt động ở chế độ ngắt.
    `HAL_TIM_Base_Start_IT(&htim2);`
  - Hàm xử lý ngắt TIM2 để bật/tắt đèn

```c
/**
  * @brief This function handles TIM2 global interrupt.
  */
void TIM2_IRQHandler(void)
{
  /* USER CODE BEGIN TIM2_IRQn 0 */

  /* USER CODE END TIM2_IRQn 0 */
  HAL_TIM_IRQHandler(&htim2);
  /* USER CODE BEGIN TIM2_IRQn 1 */
  HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_0);
  /* USER CODE END TIM2_IRQn 1 */
}
```

3.7. Lập trình ghép nối cổng nối tiếp UART

USART cũng được điều khiển thông qua driver tương ứng, cung cấp sẵn bởi nhà sản xuất STMicroelectronics

- Xác định đoạn code cấu hình USART1
- Xác định handler của USART1
- Xác định 2 hàm quan trọng của USART driver: transmit và receive

```c
  * @brief  Sends an amount of data in blocking mode.
HAL_StatusTypeDef HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)

  * @brief  Receives an amount of data in blocking mode.
HAL_StatusTypeDef HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)
```

- Xây dựng hàm in một xâu ra cổng USART1

```c
void UartPrint(char *);
```

- Sử dụng hàm vừa xây dựng, in xâu "UART initialized!" ra Virtual Terminal.

3.8. Lập trình điều khiển LCD 16x2

LCD là thiết bị ngoại vi không được hỗ trợ trực tiếp từ STM32CubeIDE và các thư viện tương ứng. Do đó cần tự xây dựng driver cho LCD, gồm các cấu trúc dữ liệu, hàm vào ra mức thấp, và hàm điều khiển chức năng.

Project mẫu cung cấp các cấu trúc và hàm sau:

- Các chân điều khiển ghép nối với LCD

```c
typedef struct
{
    GPIO_TypeDef * LCD_GPIO;        //LCD port
    uint16_t D0_PIN;                //Data pins - 8 bit mode
    uint16_t D1_PIN;
    uint16_t D2_PIN;
    uint16_t D3_PIN;
    uint16_t D4_PIN;
    uint16_t D5_PIN;
    uint16_t D6_PIN;
    uint16_t D7_PIN;
}LCD_Databus_Config;
typedef struct
{
    GPIO_TypeDef * LCD_GPIO;        //LCD port
    uint16_t EN_PIN;                //EN
    uint16_t RW_PIN;                //RW (usually 0)
    uint16_t RS_PIN;                //RS
}LCD_Controlbus_Config;
```

- Các hàm thao tác với LCD

```
void LCD_Init();
void LCD_WriteChar(uint8_t);
void LCD_WriteString(char*);
void LCD_Command(uint8_t);
void LCD_Data(uint8_t);
```

Hãy lập trình để in ra 2 dòng chữ trên LCD:

“Hello 2022

Be stronger”

3.9. Bài tập tổng hợp

Hãy lập trình chức năng hiện giờ trên LCD.

- Dùng timer để đếm thời gian đã trôi qua kể từ khi mạch chạy.
- Hiện dữ liệu lên LCD theo thời gian thực, khuôn dạng HH:MM:SS.
- Đèn LED nháy theo chu kỳ 1 giây.
- Khi bấm nút thì thời gian trên LCD bị reset về 0.

- Hết -