Lập trình GPIO nhấp nháy Led PC13 – STM32F103C8T6

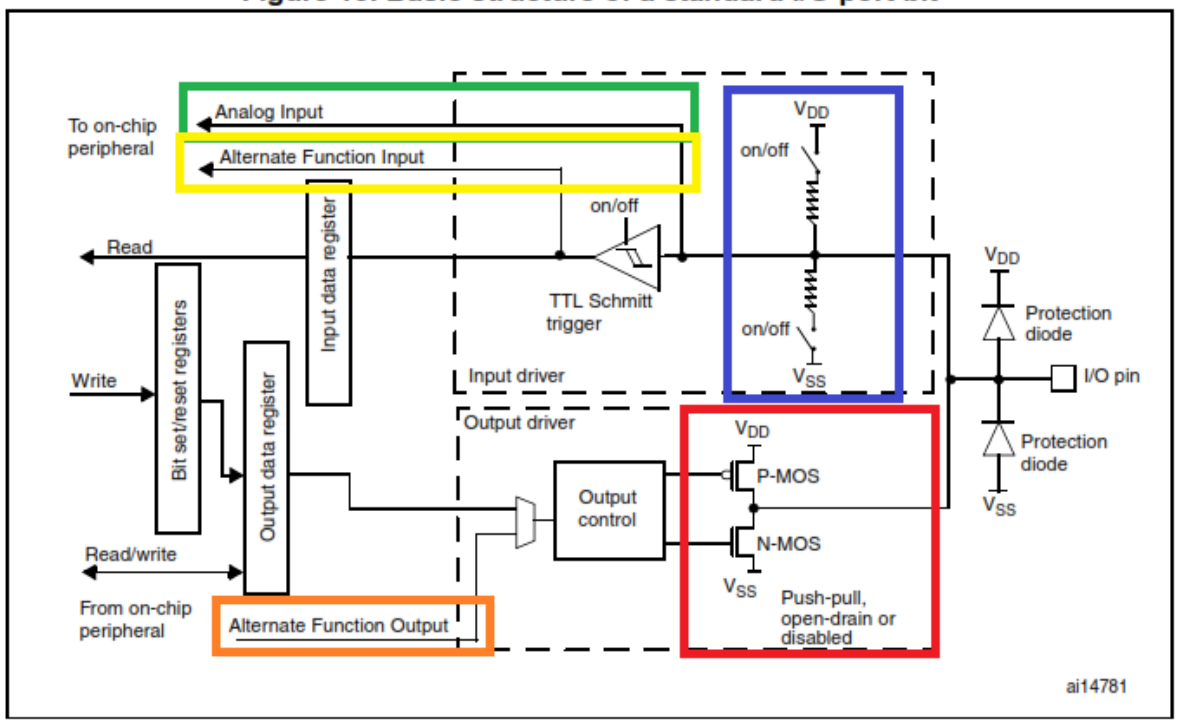
Để bắt đầu tìm hiểu các chức năng ngoại vi với bất cứ dòng vi điều khiển nào, đầu tiên các bạn cần đọc datasheet chức năng chính của nó.

Bài viết này hướng dẫn về chức năng Output push – pull sử dụng kit STM32F103C8T6.

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:

Figure 13. Basic structure of a standard I/O port bit



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

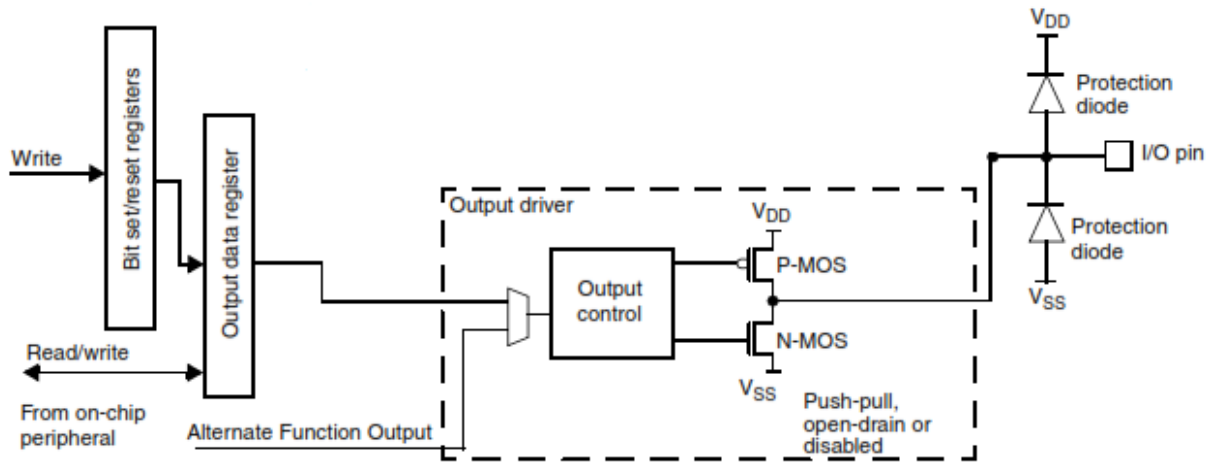
– Output driver

– Input driver

Chức năng của GPIO bao gồm:

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

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.

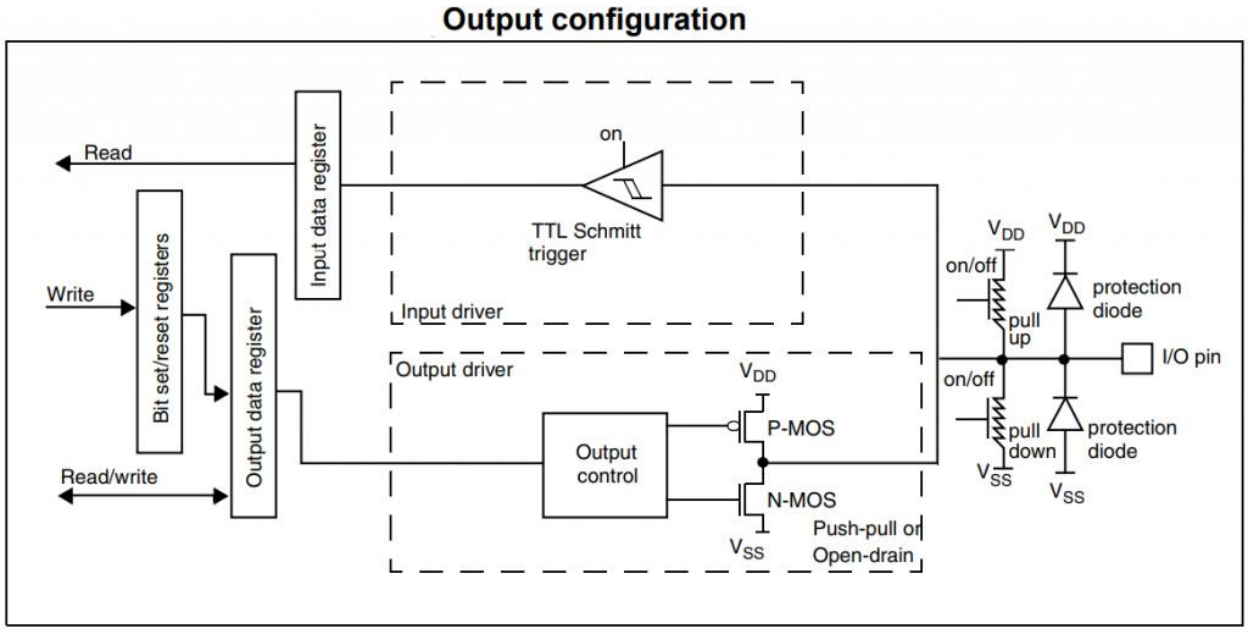
Output data register:

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:

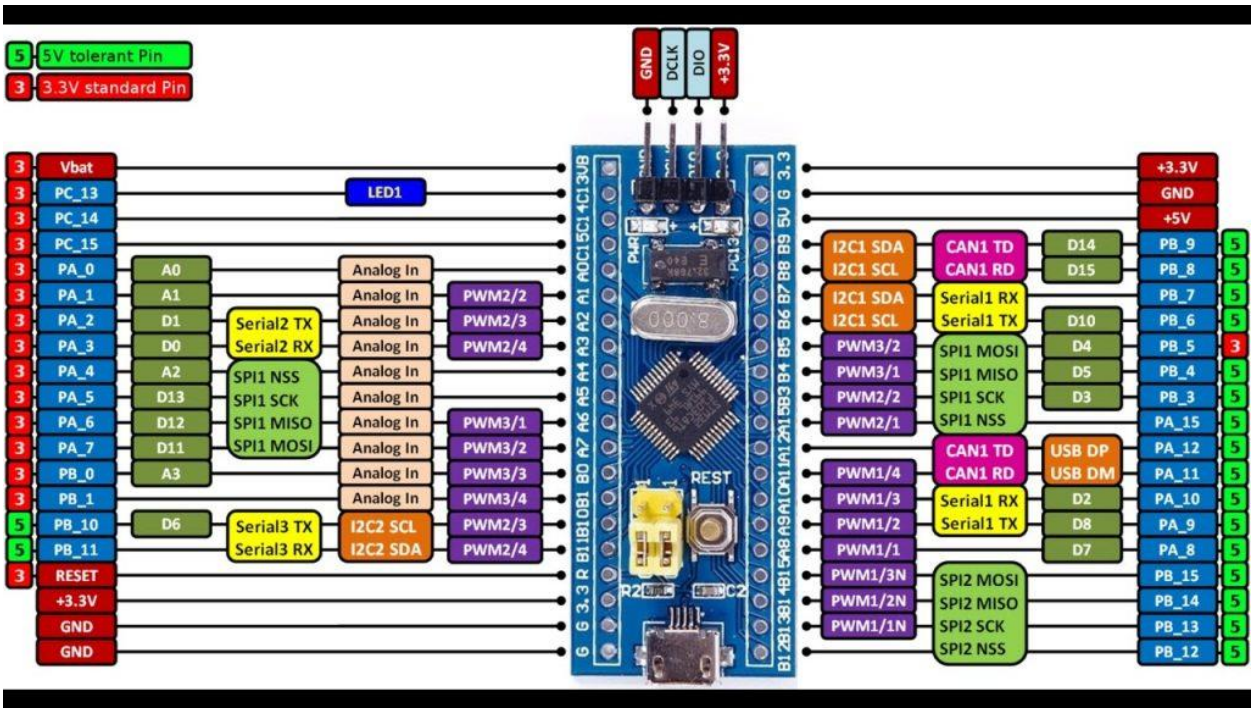
Khi một I/O Pin được cấu hình hoạt động với chức năng OUTPUT thì:
Khối điều khiển OUPUT được sử dụng với các chế độ: Open drain mode hoặc Push-pull mode.
– *Với chế độ Open drain*: Một giá trị bit bằng **"0"** ở thanh ghi ODR sẽ làm N-MOS dẫn, P-MOS ngưng dẫn, lúc này chân vi điều khiển có mức **logic 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.



Chúng ta sẽ sử dụng CubeMX để sinh code và lập trình trên môi trường KeilC-V5. 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 CubeMX 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 trên kit này bao gồm có 3 Port chính:

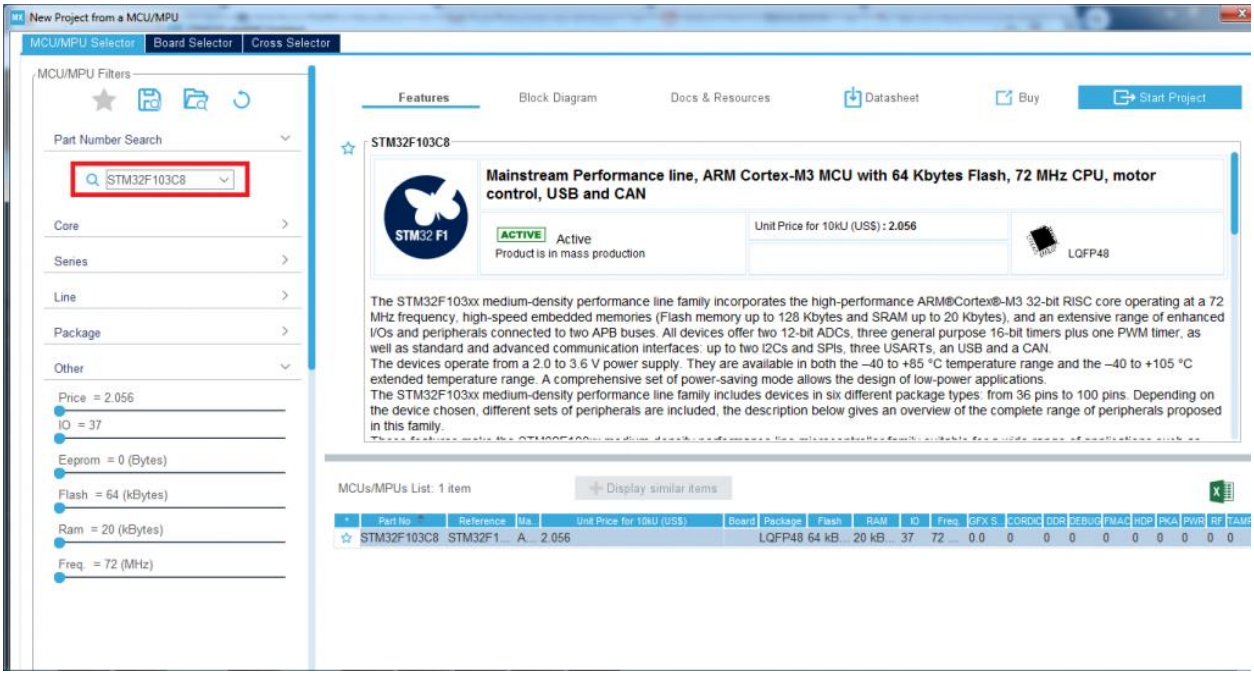
- Port A: 16 chân
- Port B: 16 chân
- Port C: 3 chân

Ở Port C chân thứ 13, trên kit đã layout sẵn một đèn led, chúng ta sẽ sử dụng đèn led này để thực hành với chức năng *output push-pull*

Cấu hình trên CubeMX:

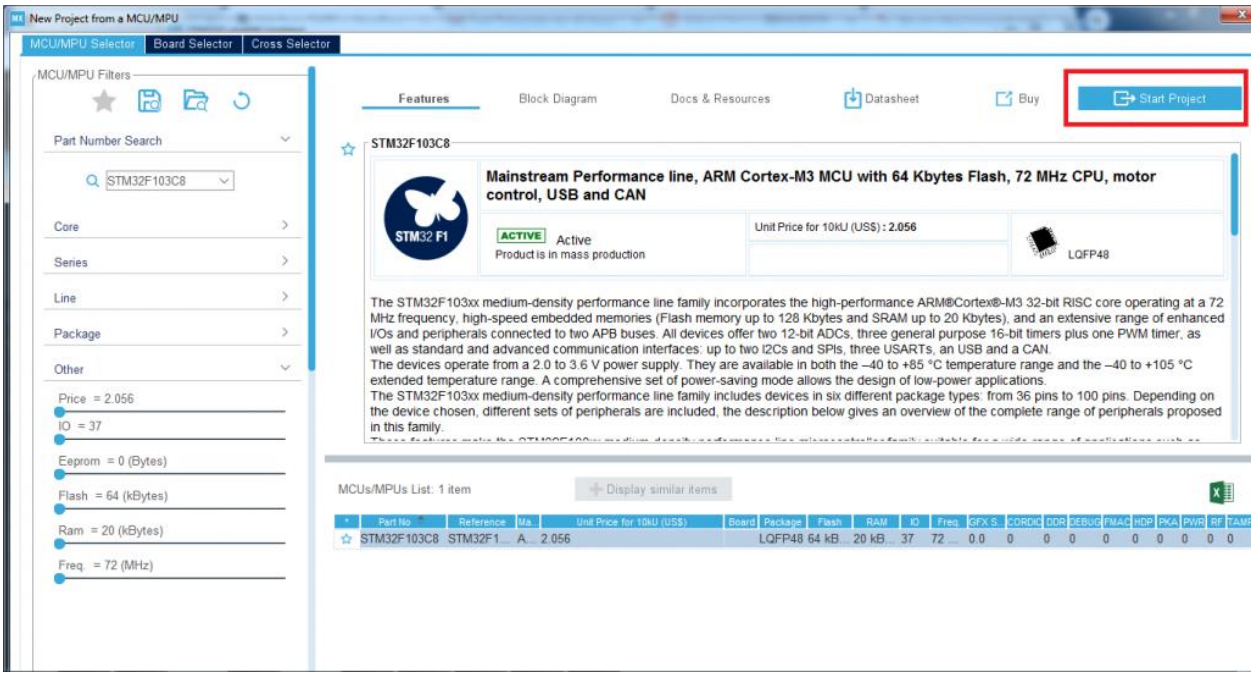
Bước 1:

Mở CubeMX chọn chip STM32F103C8T6



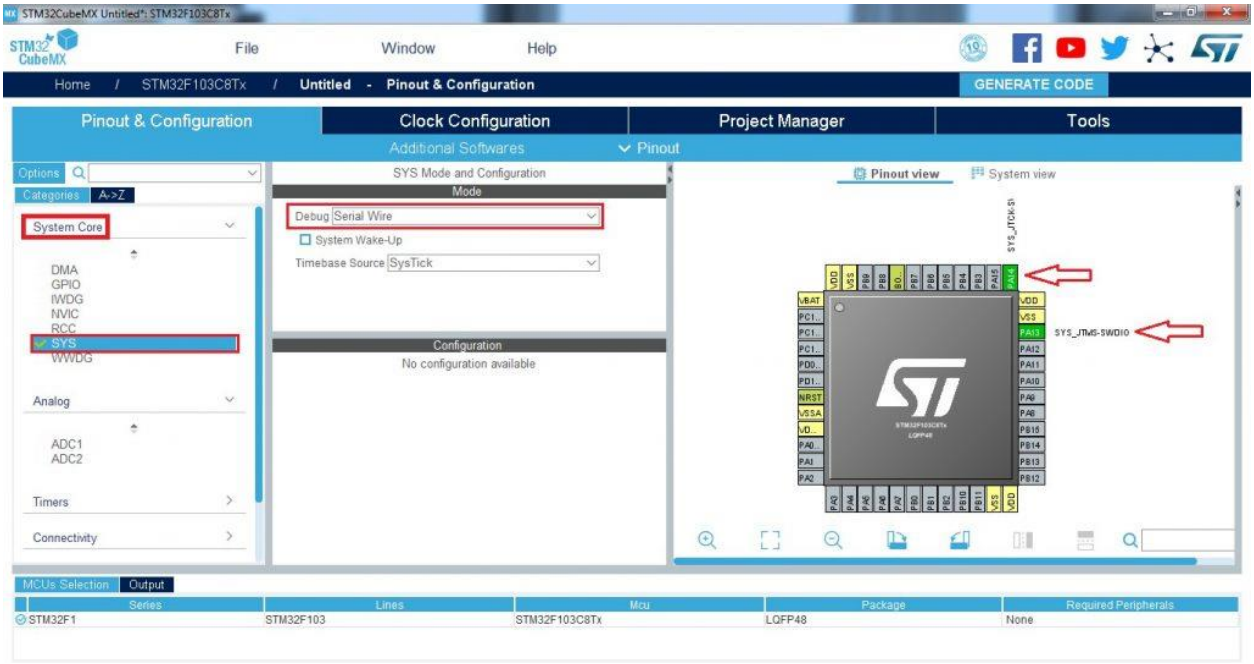
Bước 2:

Bắt đầu project



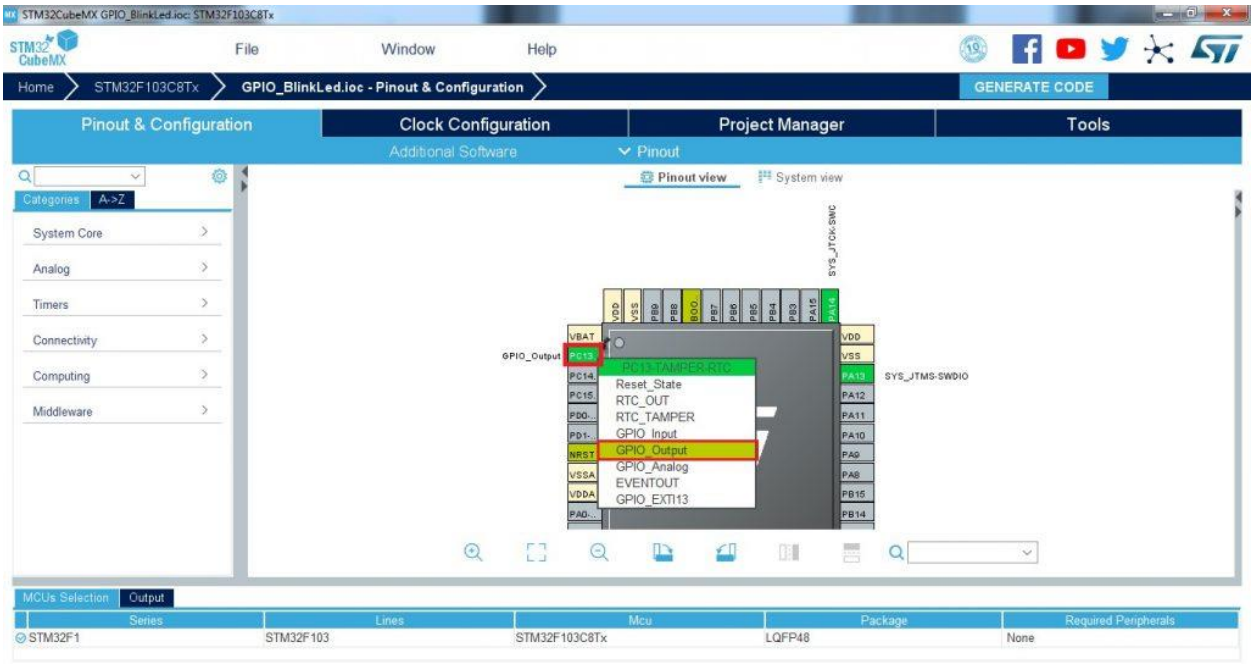
Bước 3:

Cài đặt *Serial wire* để nạp code theo chuẩn SWDIO-SWCLK



Bước 4:

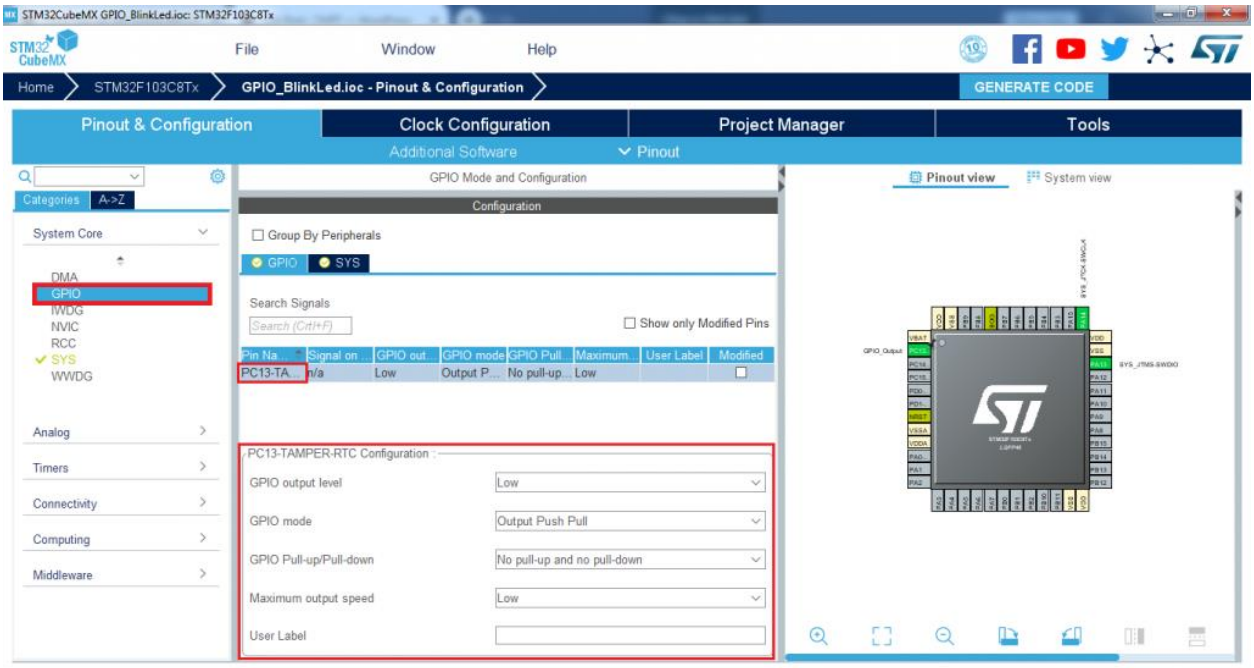
Click chuột phải vào chân PC13 sau đó tích vào ô *GPIO_Output*



Bước 5:

Cấu hình các chân GPIO như ở dưới:

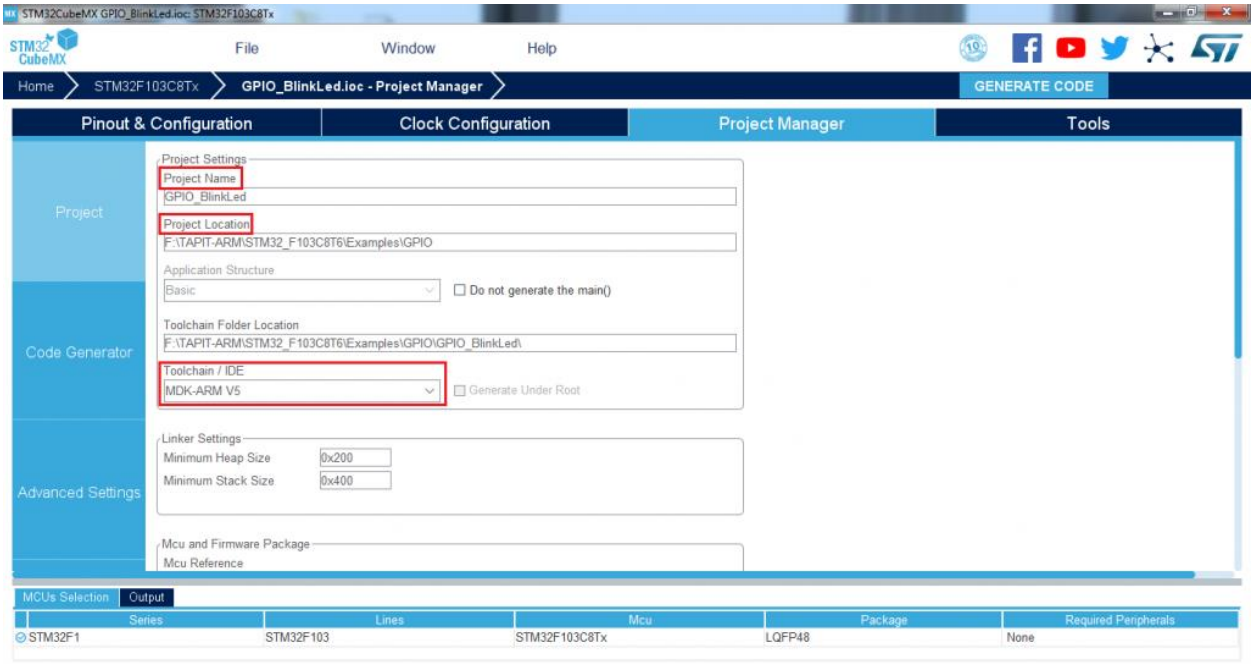
- *GPIO output level*: “LOW” (Trạng thái ban đầu của led và sẽ được kích khi xuất mức “0” tại chân IO)
- *GPIO mode*: “Output push-pull”
- *Maximum output speed*: Đối với các dòng vi điều khiển có tốc độ xử lý nhanh từ vài chục MHz trở lên, thì chúng ta phải khai báo tốc độ dao động tại chân đó để đáp 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 8MHz nên ở đây sẽ chọn “LOW”



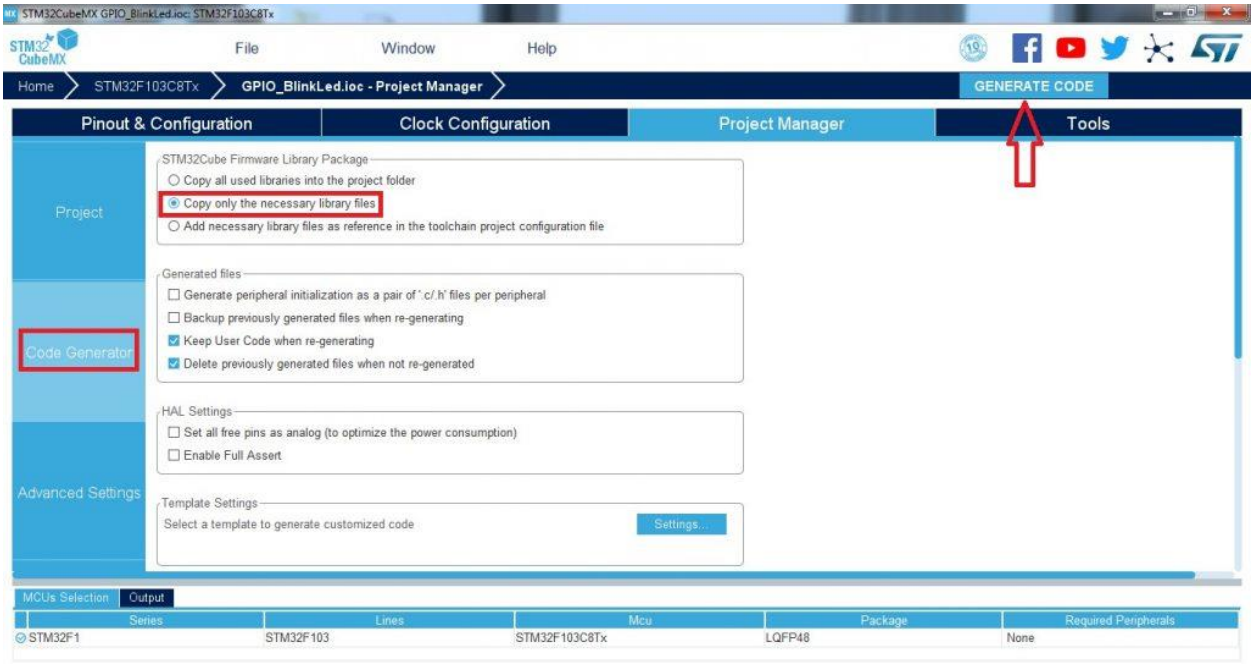
Bước 6:

Setting project và sinh code

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



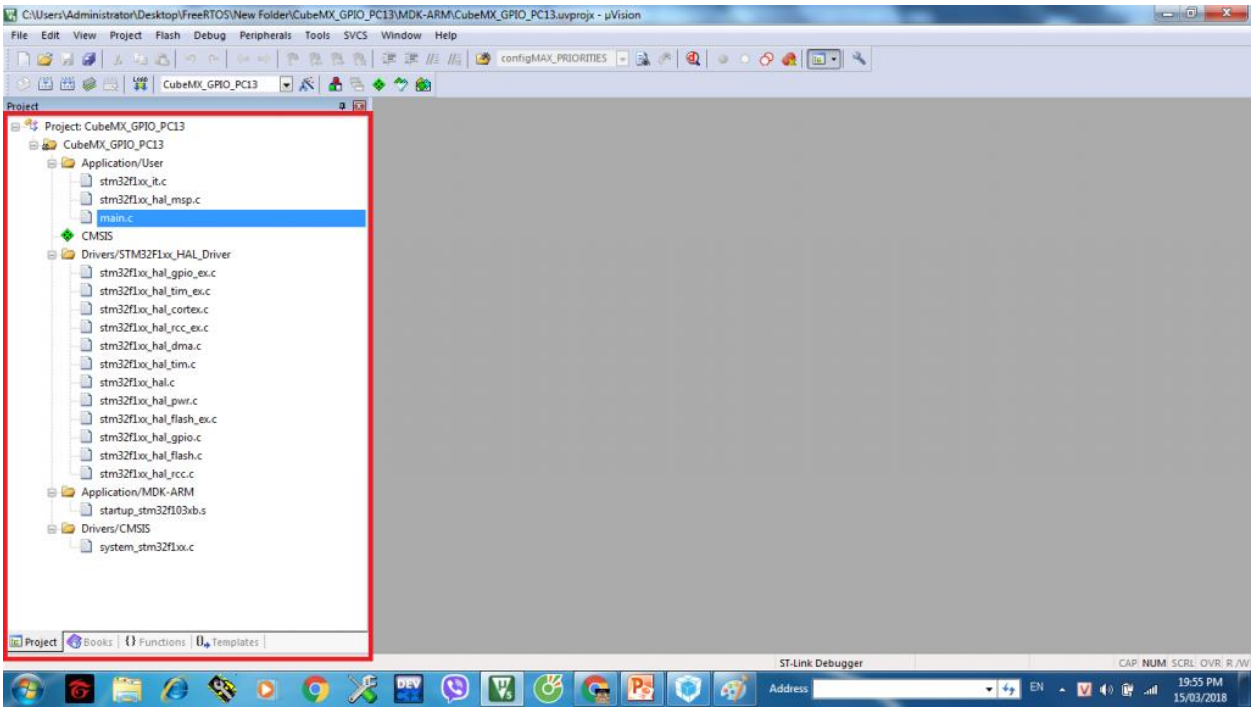
- Click vào Code Generator sau đó chọn copy các thư viện cần thiết và sinh code



Hướng dẫn lập trình nhấp nháy led PC13 sau khi đã sinh code:

Sau khi sinh code CubeMX sẽ hiển thị thông báo

- Chọn *open project*



- Chọn function:

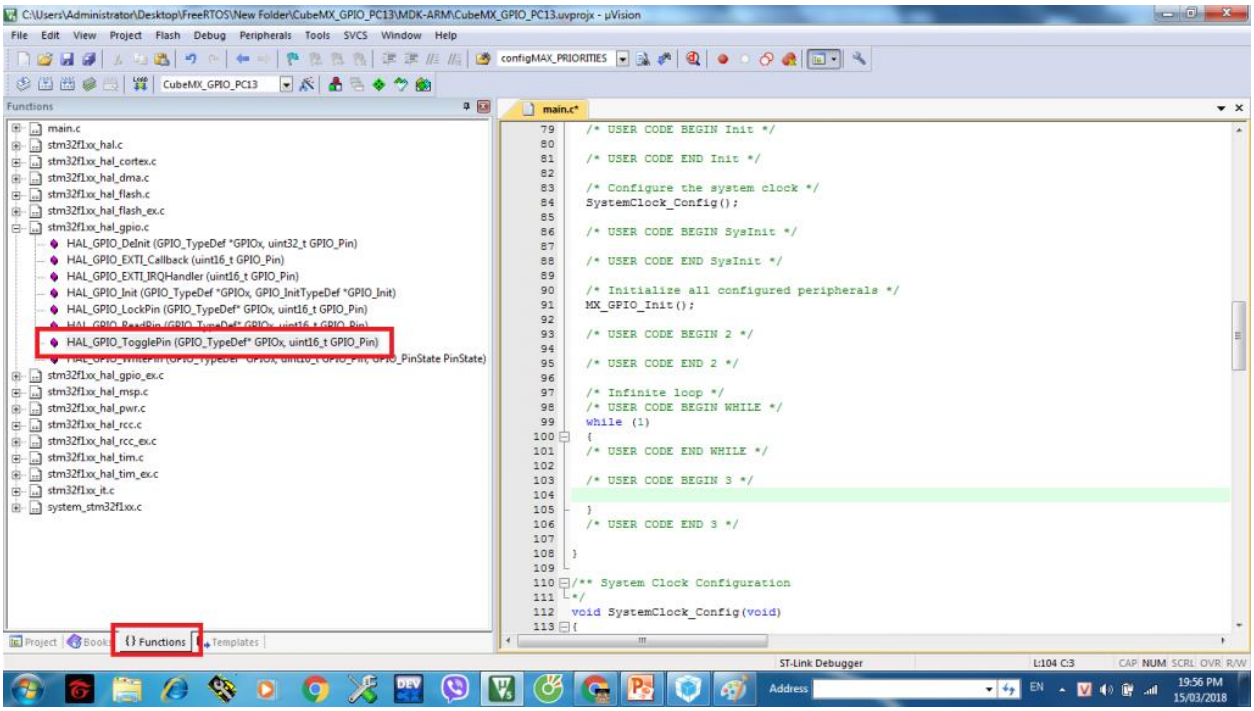
Ở đây sẽ có các hàm liên quan đến điều khiển các chân GPIO:

– HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin) để đảo trạng thái của led PC13. 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(GPIOC,GPIO_PIN_13);

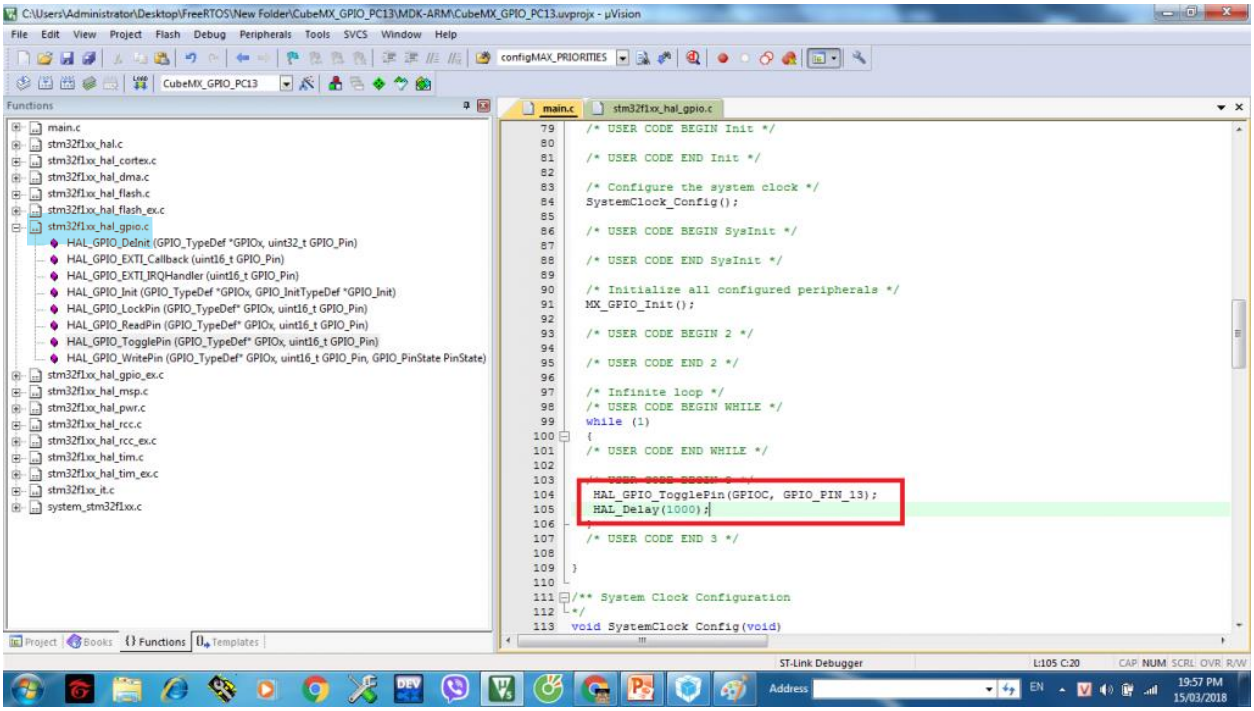
– Ngoài ra có thể xuất mức “1” hoặc mức “0” tại chân IO thông qua hàm:

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);

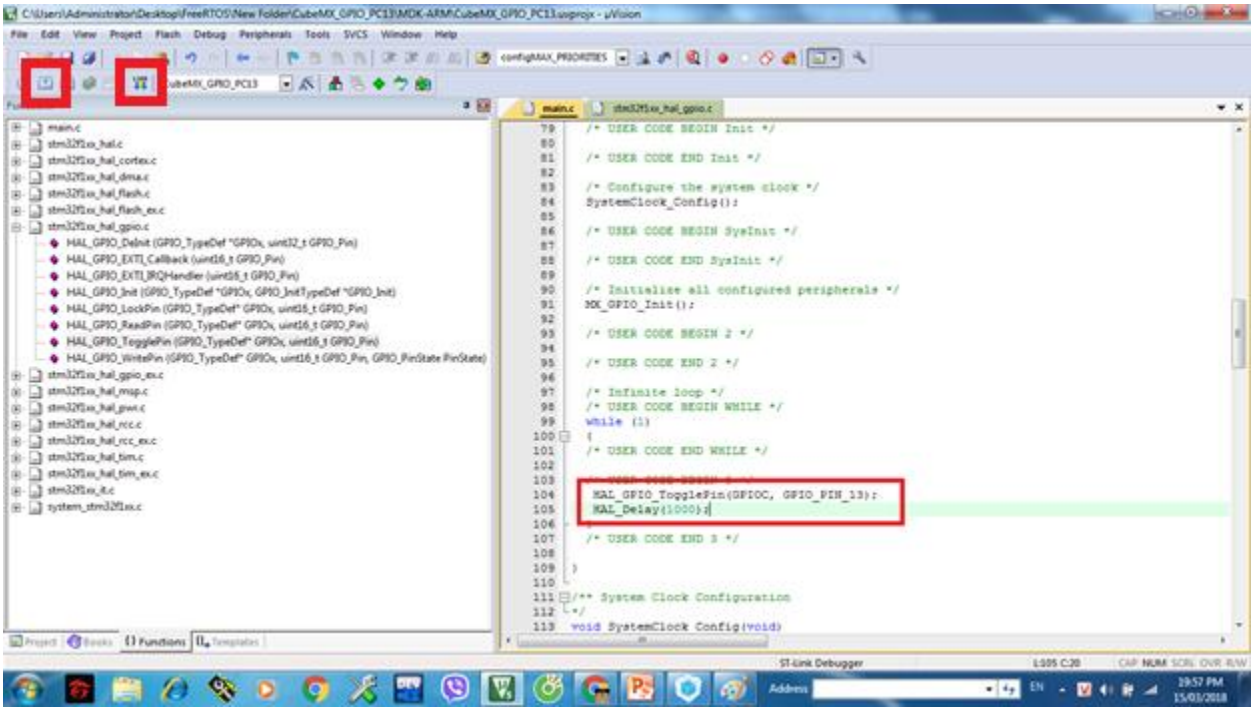
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);



- Hàm while(1) mình sẽ đảo trạng thái của led 1s 1 lần.



- Build chương trình
- Nạp code vào bộ nhớ flash của vi điều khiển


File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
main.c
stm32f1xx_hal.c
stm32f1xx_hal_cortex.c
stm32f1xx_hal_dma.c
stm32f1xx_hal_flash.c
stm32f1xx_hal_flash_ex.c
stm32f1xx_hal_gpio.c
HAL_GPIO_DeInit (GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)
HAL_GPIO_EXTI_IRQHandler (uint16_t GPIO_Pin)
HAL_GPIO_Init (GPIO_TypeDef *GPIOx, GPIO_InitTypeDef *GPIO_Init)
HAL_GPIO_LockPin (GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
HAL_GPIO_ReadPin (GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
HAL_GPIO_TogglePin (GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
HAL_GPIO_WritePin (GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)
stm32f1xx_hal_gpio_ex.c
stm32f1xx_hal_msp.c
stm32f1xx_hal_pwr.c
stm32f1xx_hal_rcc.c
stm32f1xx_hal_rcc_ex.c
stm32f1xx_hal_tim.c
stm32f1xx_hal_tim_ex.c
stm32f1xx_it.c
system_stm32f1xx.c
Functions
main.c
stm32f1xx_hal_gpio.c
/* USER CODE BEGIN Init */
/* USER CODE END Init */
/* Configure the system clock */
SystemClock_Config();
/* USER CODE BEGIN SysInit */
/* USER CODE END SysInit */
/* Initialize all configured peripherals */
MX_GPIO_Init();
/* USER CODE BEGIN 2 */
/* USER CODE END 2 */
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
/* USER CODE END WHILE */
HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
HAL_Delay(1000);
/* USER CODE END 3 */
}
/** System Clock Configuration
*/
void SystemClock_Config(void)
ST-Link Debugger
L:105 C:20
CAP NUM SCRL OVR R/W
Address
EN
19:57 PM
15/03/2018