Chương 2

HƯỚNG DẪN SỬ DỤNG PHẦN MỀM LẬP TRÌNH KEIL CHO ARM CORTEX – M3 STM32F103VET

CÁC BÀI THỰC HÀNH LED ĐƠN, NÚT NHẤN

- **GIỚI THIỆU**
- **CÁCH TẠO PROJECT CHO ARM STM32F103**
- **GIẢI THÍCH CHƯƠNG TRÌNH ĐIỀU KHIỂN LED ĐƠN**
- **CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN LED ĐƠN**
- **CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN LED ĐƠN VÀ NÚT NHẤN**

I. GIỚI THIỆU

Chương này trình bày cách sử dụng phần mềm KEIL – ARM để tạo project lập trình cho ARM cortex – M3 STM32F.

Sau khi biết cách tạo project thì có thể thực hiện các bài điều khiển khác.

II. CÁCH TẠO PROJECT CHO ARM STM32F103

Cách tạo project lập trình cho ARM cũng giống như lập trình cho vi điều khiển AT89Sxx, chỉ có 1 vài khác biệt, chi tiết tiến hành như sau.

Yêu cầu: tạo một project điều khiển 1 led đơn chớp tắt.

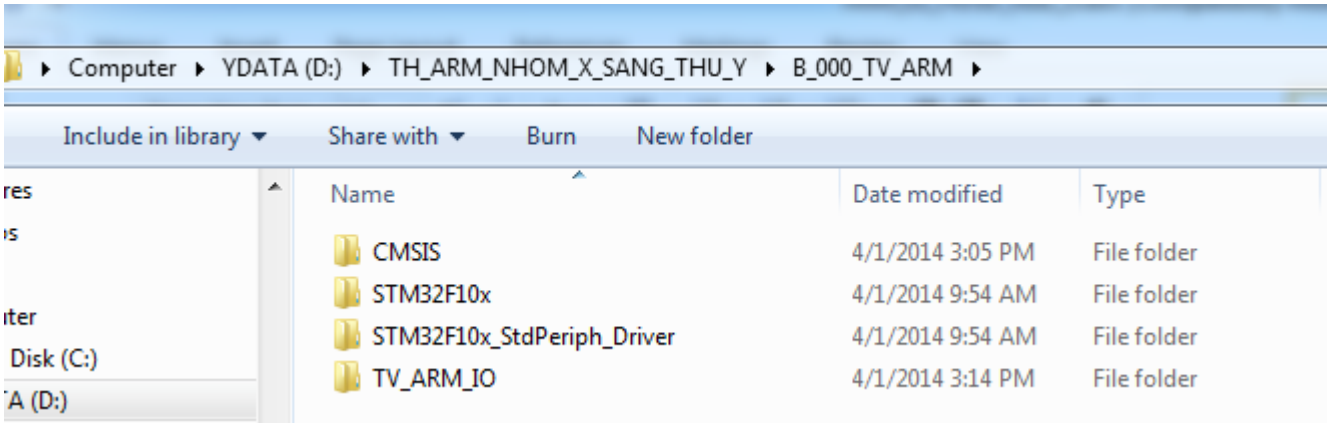
Thực hiện: ARM là vi điều khiển 32 bit với tài nguyên và cấu trúc lớn hơn nhiều so với vi điều khiển 8 bit và được hỗ trợ thư viện nhiều nên khi lập trình ta phải khai báo các thư viện đã cho.

Mỗi một project chứa nhiều file nên phải tạo nhiều thư mục để lưu các loại file khác nhau ví dụ thư mục lưu thư viện, thư mục lưu file phát sinh sau biên dịch, thư mục lưu file biên dịch trung gian, …

Bước 1: Tạo thư mục cha có tên là "TH_ARM_NHOM_X_SANG_THU_Y" để lưu tất cả các project.

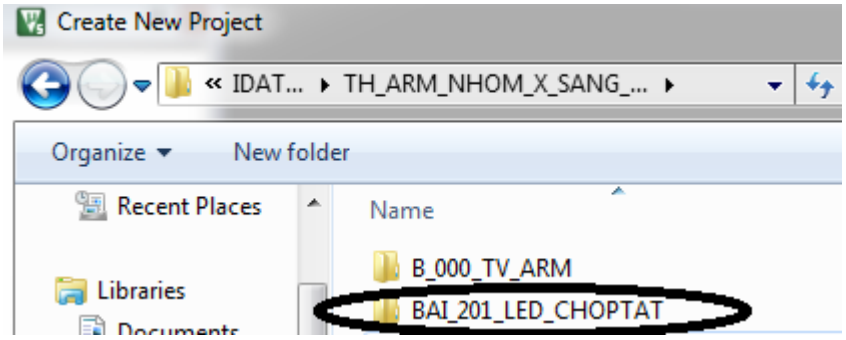
Bước 2: Tạo thư mục con nằm trong thư mục cha có tên "B_000_TV_ARM" để lưu các file thư viện dùng cho các bài thực hành.

Bước 3: Tiến hành copy các file thư viện vào thư mục "B_000_TV_ARM", kết quả sau khi copy như sau:



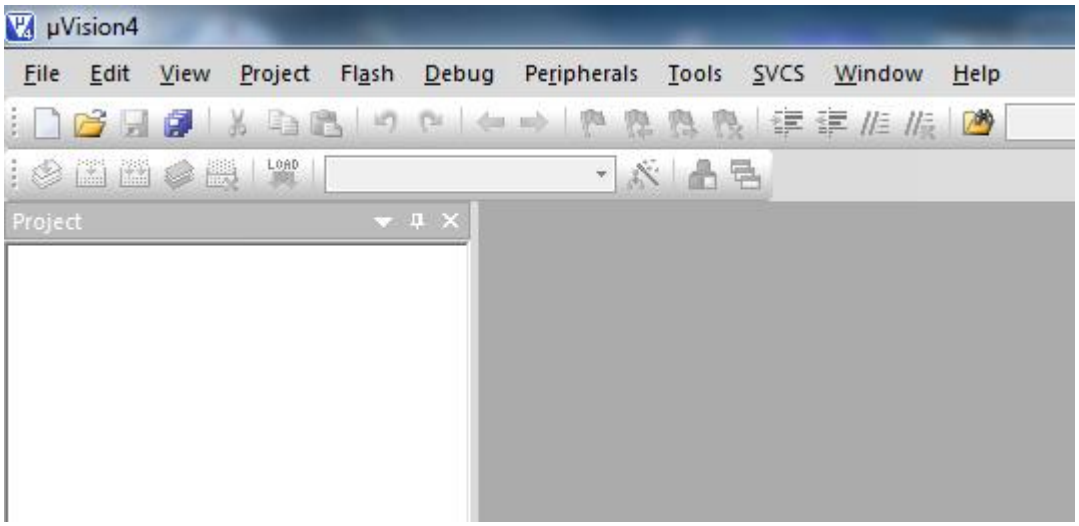
Hình 2-1. Các thư mục chứ các file thư viện.

Bước 4: Tạo thư mục con "BAI_201_LED_CHOPTAT" để lưu tất cả các file của project. Kết quả sau khi tạo thư mục như hình sau



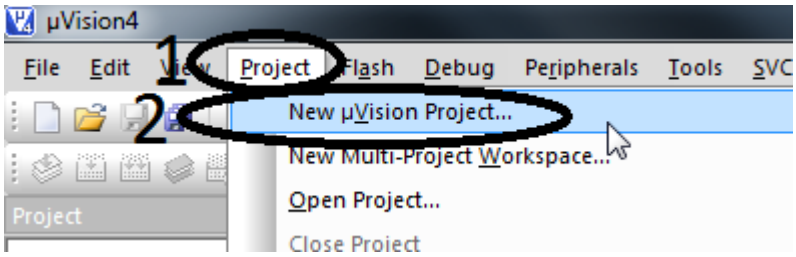
Hình 2-2. Thư mục cha và thư mục con.

Bước 5: Khởi động phần mềm KEIL như sau:



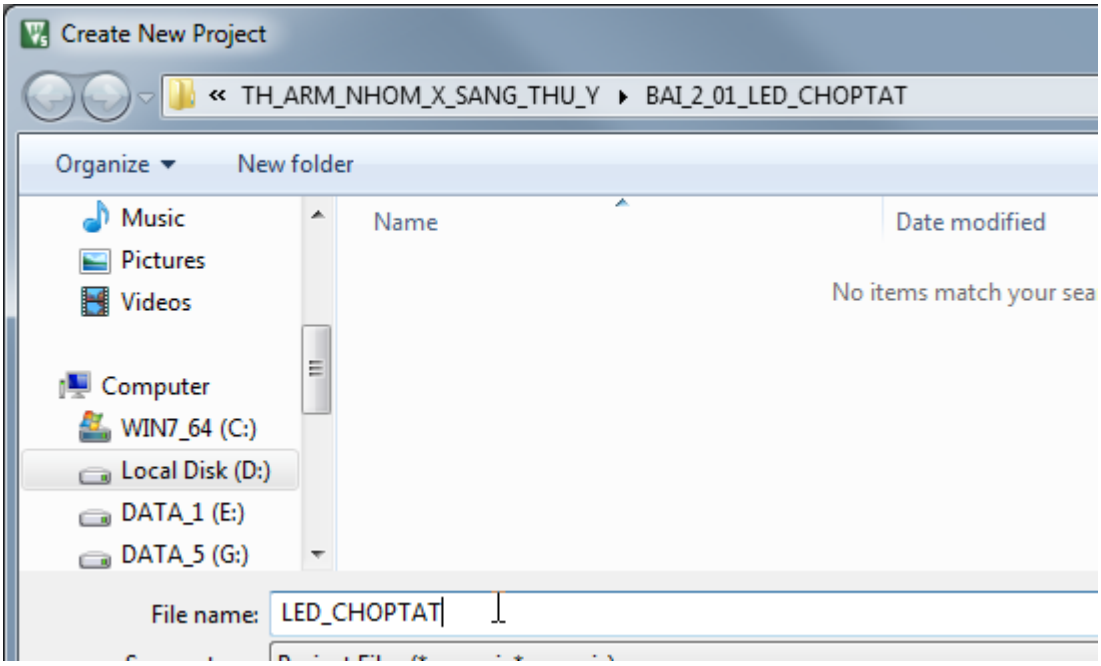
Hình 2-3. Giao diện phần mềm Keil.

Bước 6: Tạo project mới theo 2 trình tự 1 và 2 như hình sau:



Hình 2-4. Tiến hành tạo project mới.

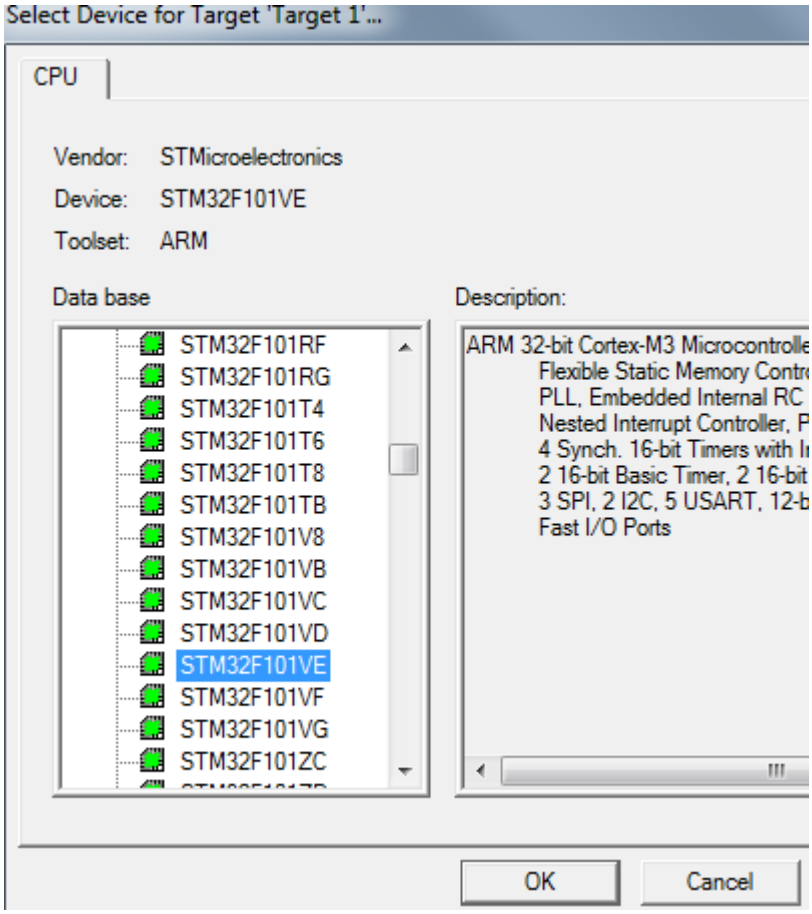
Một giao diện mới xuất hiện, bạn tiến hành chọn thư mục lưu project rồi đánh tên cho project tùy ý nhưng ở đây đặt tên là LED_CHOPTAT, kết quả như hình sau:



Hình 2-5. Đặt tên, chọn đường dẫn lưu project mới.

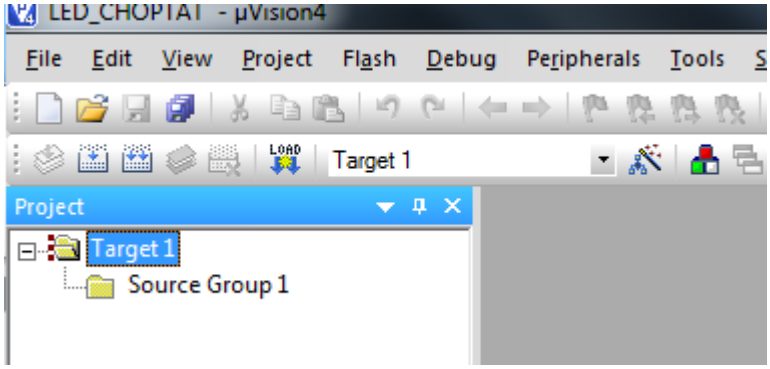
Một giao diện mới xuất hiện yêu cầu bạn chọn loại ARM, bạn tiến hành chọn mục ARM "STMicroelectronis" rồi chọn chip cho đúng với kit đang sử dụng, kết quả như hình sau:

CPU cho thiết bị thực hành có tên là STM32F103VE.



Hình 2-6. Chọn đúng thông số cho chip.

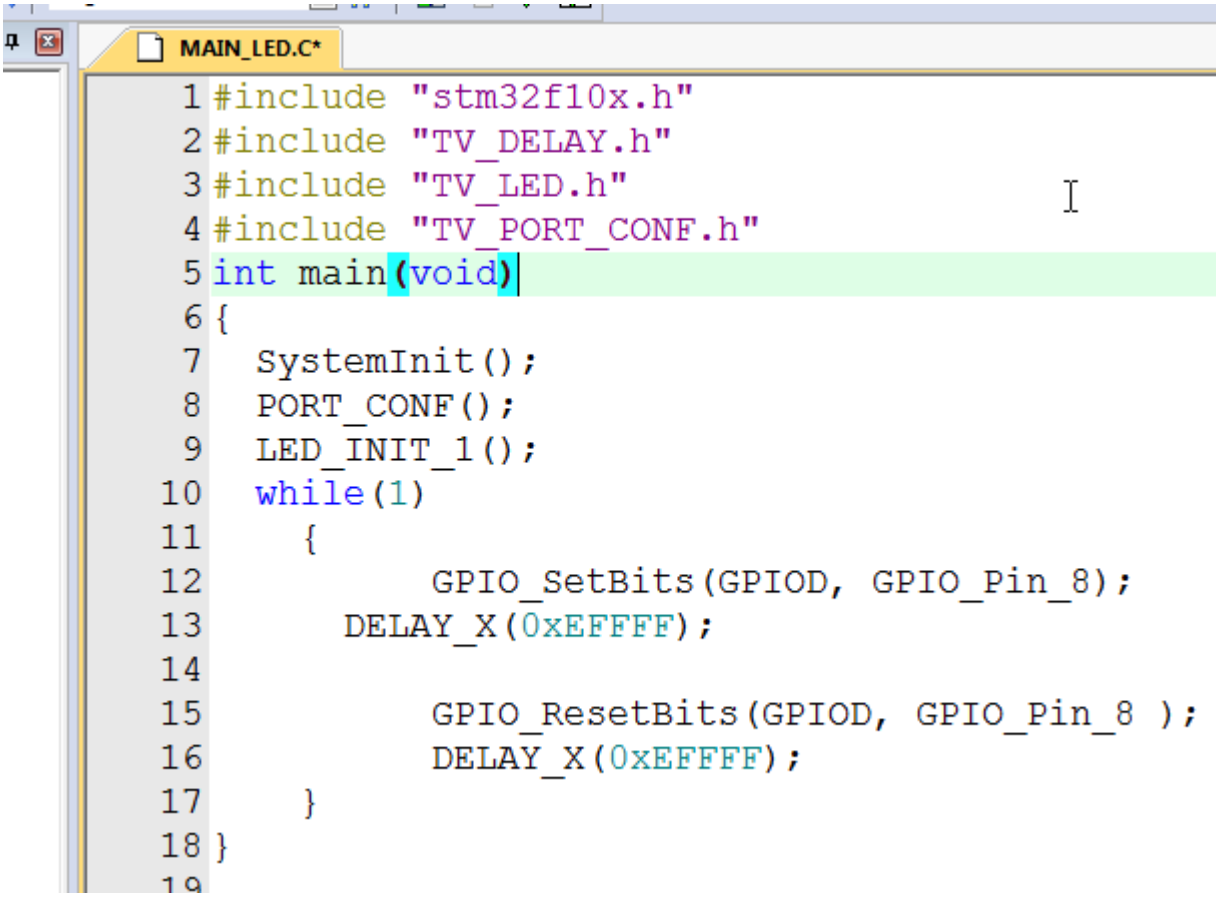
Tiến hành nhấn Ok, một giao diện như sau:



Hình 2-7. Màn hình chưa có gì.

Bước 7: Tiến hành bấm New rồi tiến hành viết đoạn code như hình sau:

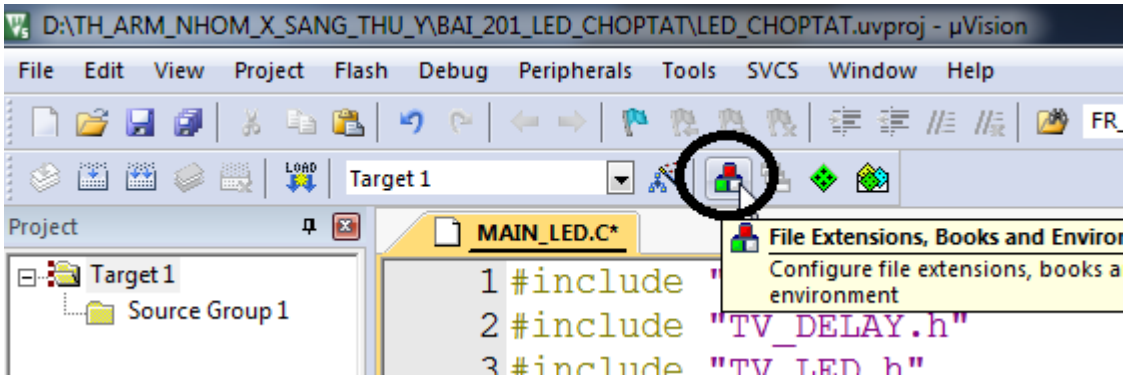
Chú ý khi đánh được 1 hàng bạn nên tiến hành lưu tên file là "MAIN_LED.C" để phần mềm hỗ trợ các tính năng cảnh báo khi bạn đánh sai, nếu chưa lưu với phần mở rộng ".C" thì không hỗ trợ.



Hình 2-8. Chọn biểu tượng gán.

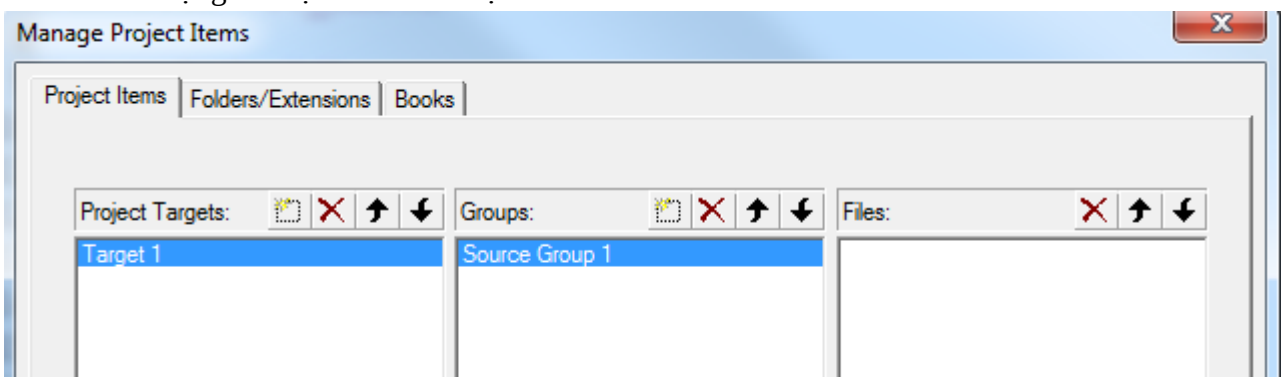
Bước 8: Tiến hành thiết lập các nhóm lưu các file cho Project, gán thư viện cho các nhóm, gán file nguồn:

Chọn biểu tượng như trong hình sau:



Hình 2-9. Chọn biểu tượng gán.

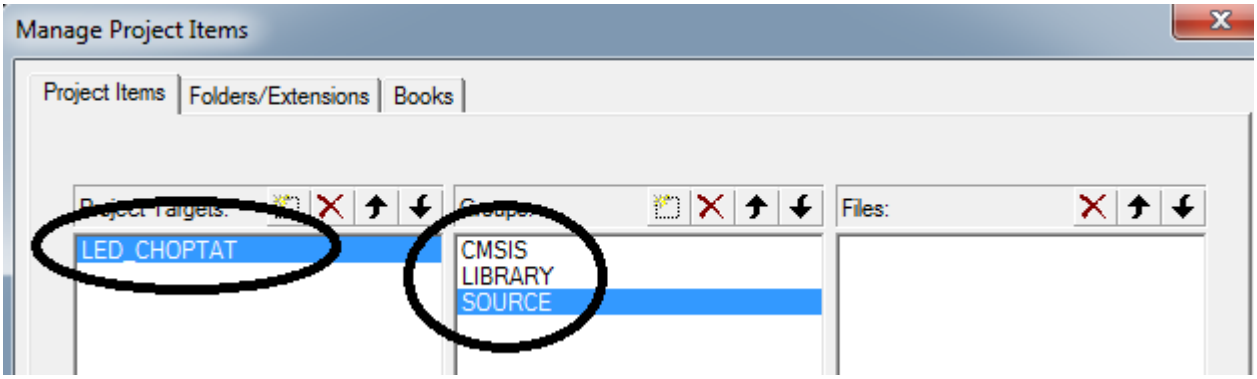
Khi đó một giao diện mới xuất hiện như hình sau:



Hình 2-10. Giao diện thay đổi tên và đường dẫn.

Trong giao diện này bạn thấy có 3 cột, cột thứ nhất tên của project là target1 và cột thứ 2 có tên là Source Group1, các tên này là mặc nhiên do phần mềm tạo ra, ta cần phải thay thế các tên này.

Kết quả thay đổi tên project và thành lập 3 group mới như hình sau:



Hình 2-11. Sau khi thay đổi tên.

Tên trong group có thể đặt tùy ý.

Cách thực hiện: bạn để dấu nháy ngay cột thứ nhất và tiến hành đánh đúng tên là xong.

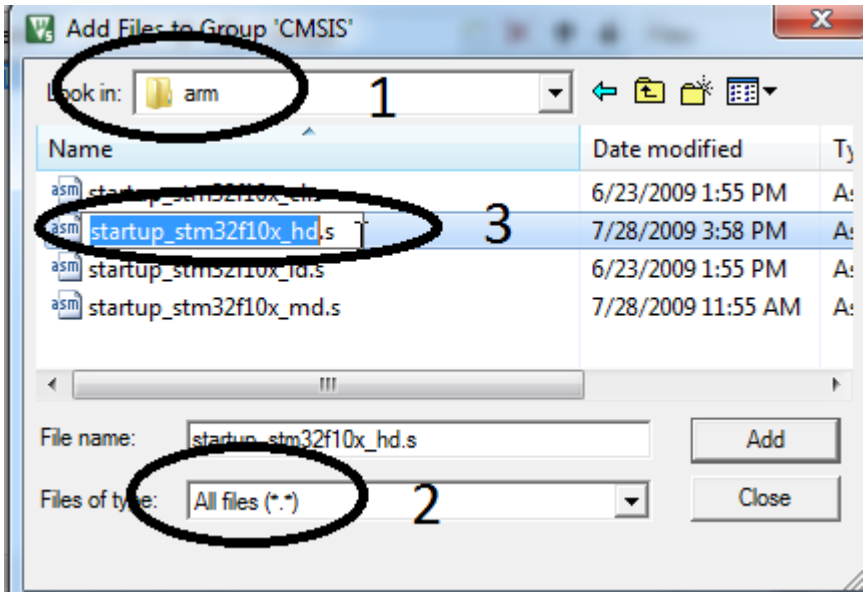
Chuyển sang cột thứ 2 rồi đánh hàng thứ nhất xong thì nhấn enter và bấm vào ô vuông gần ô xóa để tạo hàng mới và đánh tên cho group thứ 2, tương tự cho group thứ 3.

Bước 9: Tiếp theo là gán file cho từng group:

Gán các file cho group CMSIS: chọn group CMSIS rồi chọn Add Files nằm ở cột thứ 3, tiến hành tìm file theo đường dẫn:

"D:\TH_ARM_NHOM_X_SANG_THU_Y\B_000_TV_ARM\CMSIS\Core\CM3\startup\arm"

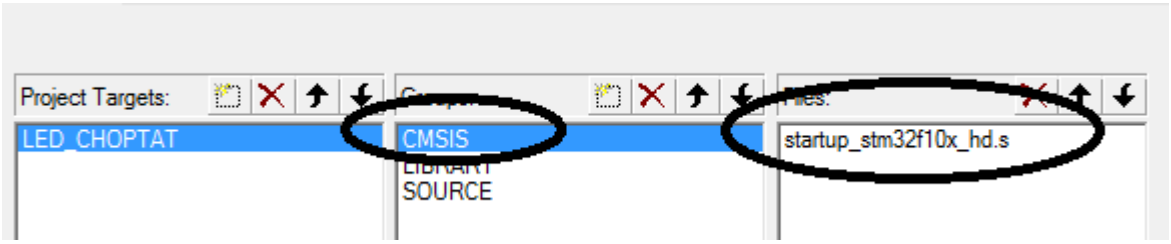
Như hình sau:



Hình 2-12. Kết quả sau khi add cho group CMSIS.

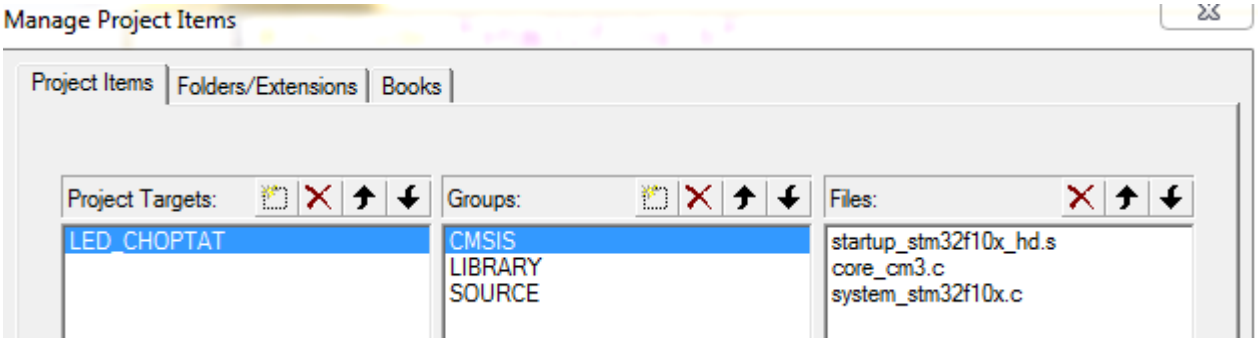
Chú ý chọn loại file mục số 2 chọn loại file là *.*

Màn hình sau khi add cho group CMSIS như hình sau:



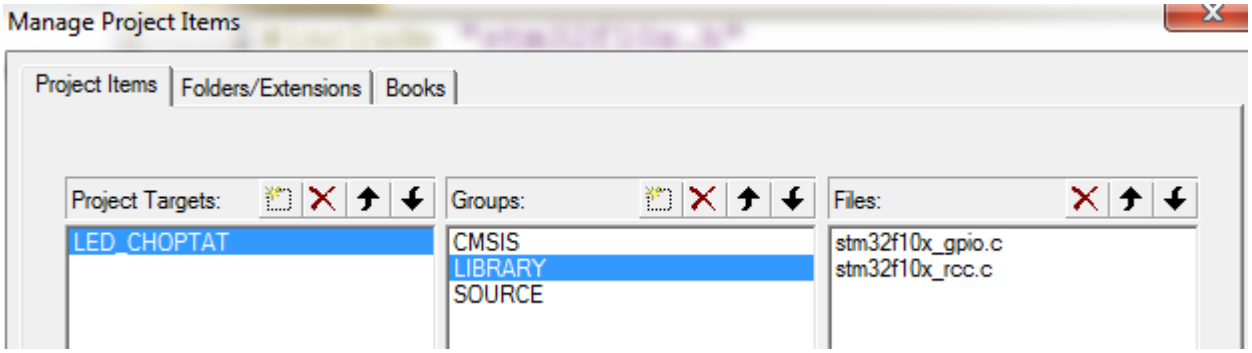
Hình 2-13. Kết quả sau khi add cho group CMSIS.

Tương tự bạn tìm file ở đường dẫn:
"D:\TH_ARM_NHOM_X_SANG_THU_Y\B_000_TV_ARM\CMSIS\Core\CM3"
rồi add, kết quả của group CMSIS như hình sau:



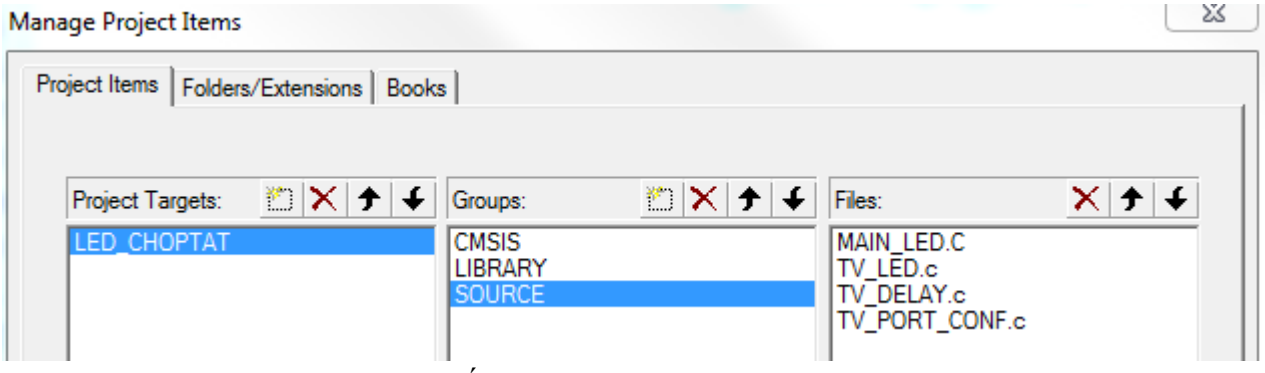
Hình 2-14. Kết quả sau khi add cho group LIBRARY.

Tương tự bạn chọn group LIBRARY và tìm file rồi add, kết quả của group LIBRARY như hình sau:



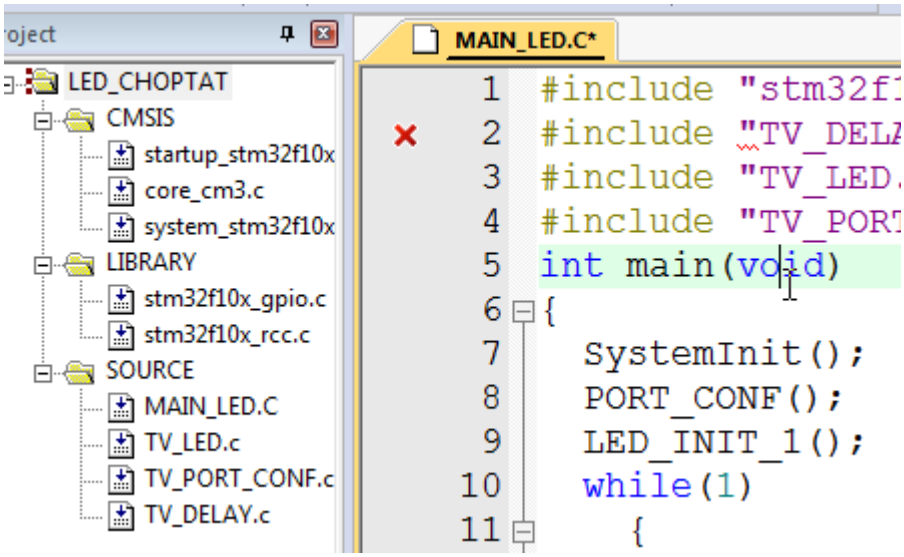
Hình 2-15. Kết quả sau khi add cho group LIBRARY.

Tương tự bạn chọn group SOURCE và tìm file mà bạn biên soạn rồi add, kết quả của group SOURCE như hình sau:



Hình 2-16. Kết quả sau khi add cho group SOURCE.

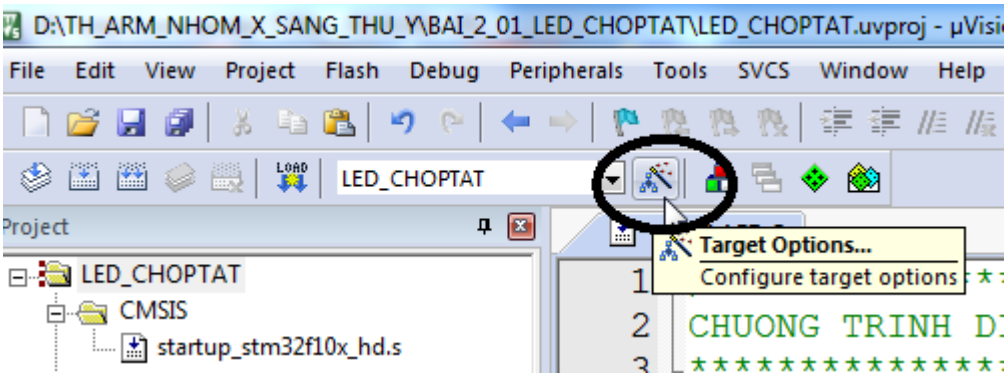
Sau đó nhấn OK để đóng phần này, giao diện màn như sau:



Hình 2-17. Giao diện màn hình mới sau khi gán cho các group.

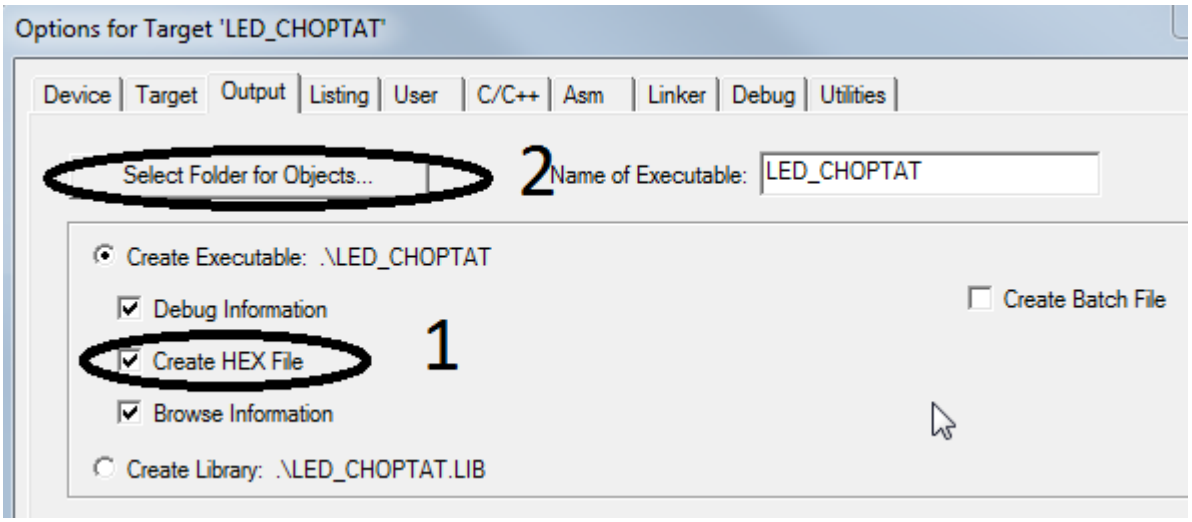
Chú ý: Với bài minh hoạ đơn giản này thì ta chỉ chọn các file thư viện cần thiết có lien quan như trên, bạn chọn dư thêm các thư viện khác cũng không ảnh hưởng chỉ làm tăng thời gian biên dịch mà thôi. Nếu bạn chưa rõ các thành thư viện này thì ở cuối bài có giải thích chi tiết cho bạn hiểu.

Bước 10: Tiến hành thiết lập các lựa chọn cho project bằng cách bấm vào biểu tượng có tên "Target Option" như hình sau:



Hình 2-18. Vị trí "Target Option".

Sau khi chọn thì một giao diện mới xuất hiện tiến hành chọn tab có tên là "Output" rồi tick vào mục tạo file hex như hình sau:

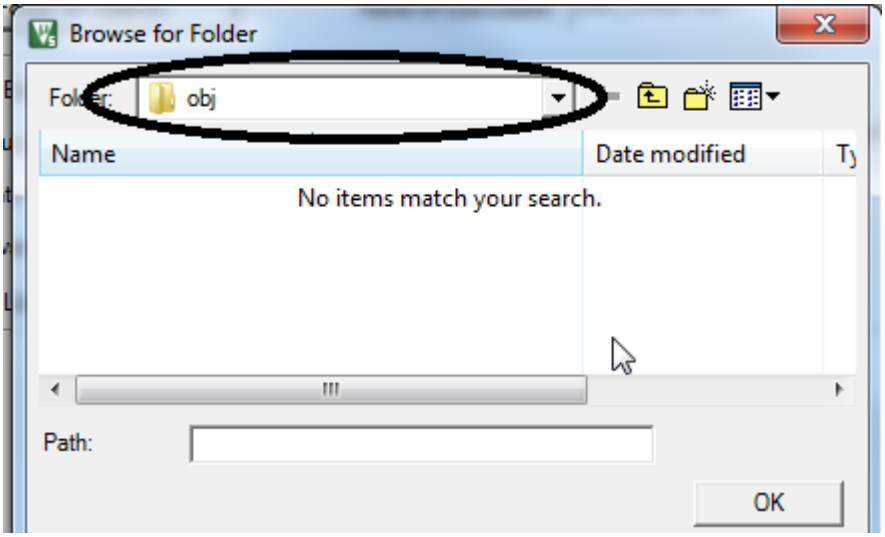


Hình 2-19. Tick ô xây dựng file hex.

Sau đó bấm vào ô tương ứng với số 2 để tạo thư mục lưu các file phát sinh khi biên dịch.

Bạn tiến hành tạo thêm thư mục có tên là obj và mở thư mục đó ra để thiết lập đường dẫn cho phần mềm biết các file đối tượng sau khi tạo ra do trình biên dịch sẽ lưu vào thư mục này cho dễ quản lý.

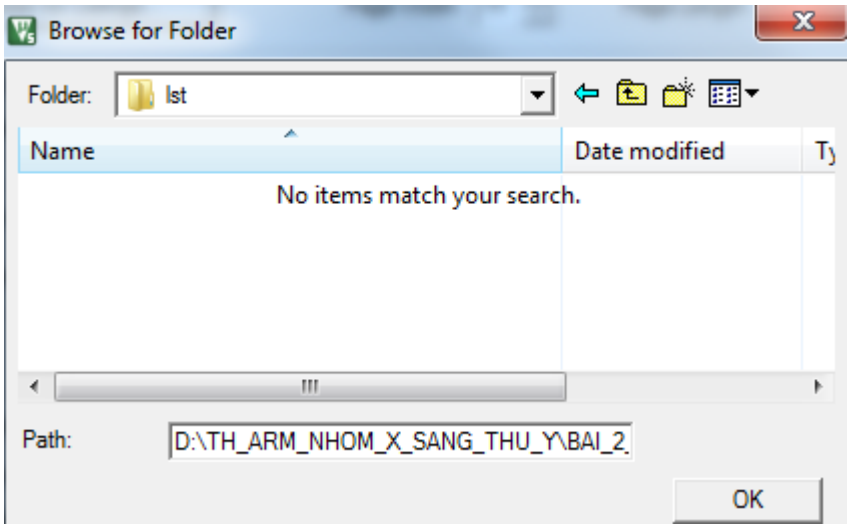
Kết quả như hình sau:



Hình 2-20. Sau khi thiết lập thư mục obj.

Tiến hành chọn tab Listing rồi bấm vào ô để thiết lập thư mục lưu các file lst do trình biên dịch tạo ra giống như mới vừa làm cho obj.

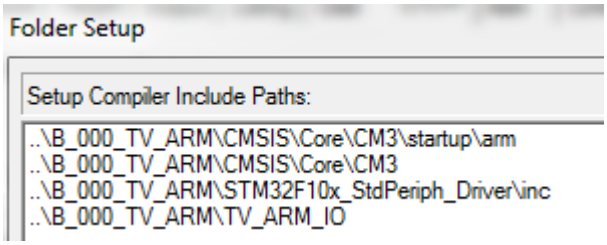
Kết quả sau khi thực hiện như hình sau:



Hình 2-21. Sau khi thiết lập thư mục lst.

Tiến hành chọn tab C/C++ rồi bấm vào ô dấu 3 chấm của hàng "Include Paths" để thiết lập các đường dẫn cho các file cho trình biên dịch biết để biên dịch.

Một giao diện mới xuất hiện chưa có đường dẫn, bạn tiến hành tạo mới và trỏ các đường dẫn và kết quả thiết lập 4 đường dẫn như sau:



Hình 2-22. Thiết lập đường dẫn cho các file để biên dịch.

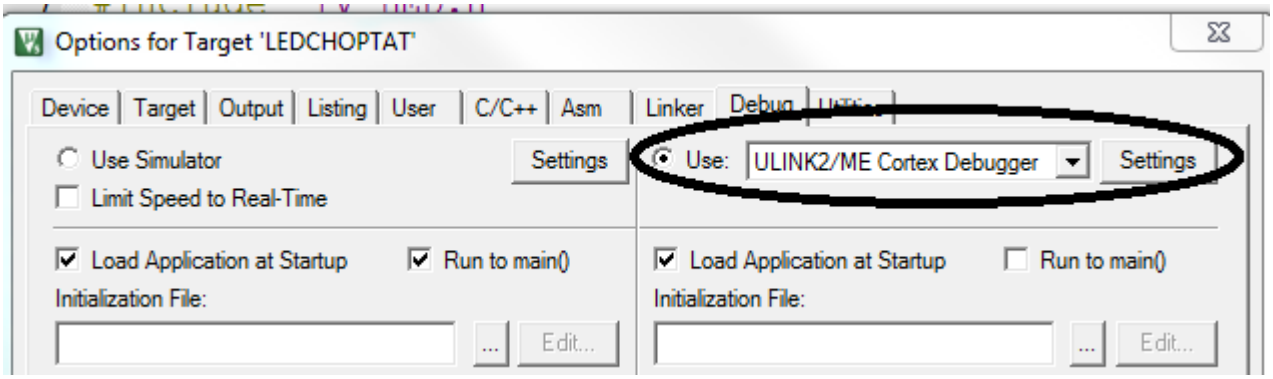
Giải thích tại sao khai báo các đường dẫn:

Hai đường dẫn đầu tiên liên quan đến 2 thư mục lưu các file "stm32f10x_gpio.c" và "stm32f10x_rcc.c".

- Đường dẫn thứ 1 liên quan đến thư mục lưu file "startup_stm32f10x_hd.s".
- Đường dẫn thứ 2 liên quan đến thư mục lưu các file "system_stm32f10x.c" và "core_cm3".
- Đường dẫn thứ 3 liên quan đến thư mục lưu các file "stm32f10x_gpio.c" và "stm32f10x_rcc.c".
- Đường dẫn thứ 4 liên quan đến thư mục lưu các file thư viện "TV_LED.c", "TV_DELAY.C", "TV_PORT_CONF.C".

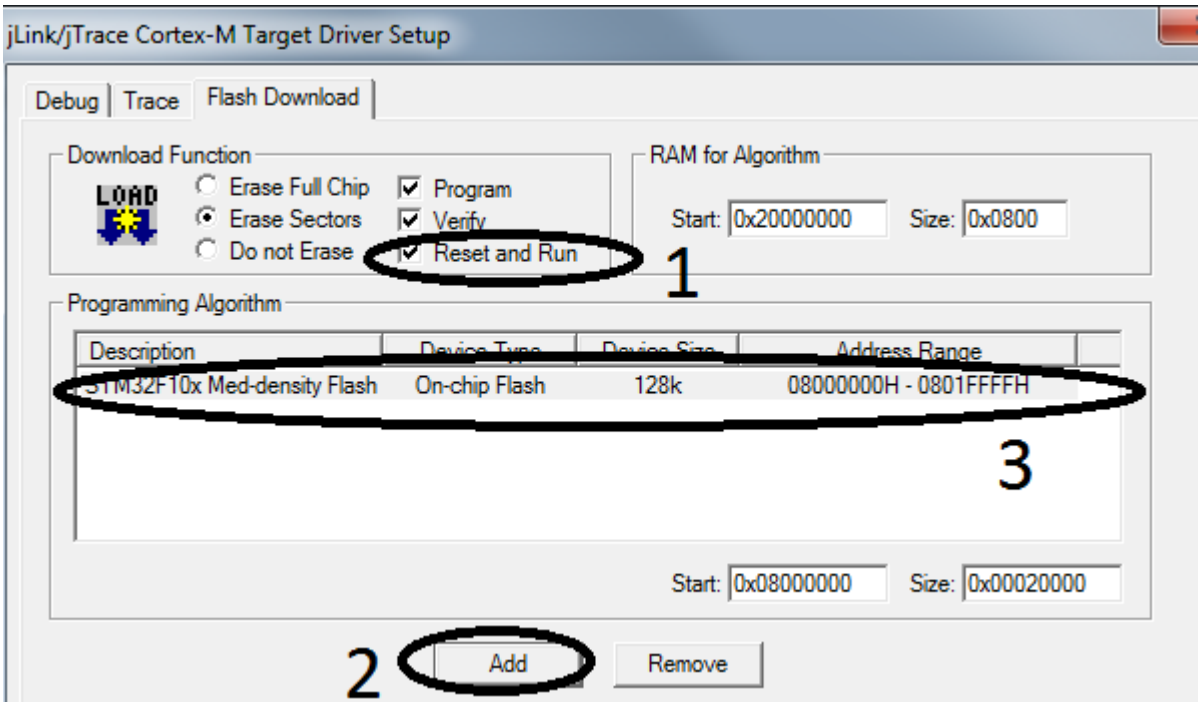
Chú ý: các đường dẫn này không đúng thì trình biên dịch sẽ báo lỗi.

Tiến hành chọn tab "Debug" rồi chọn mục Setting để chọn mạch nạp như hình sau:



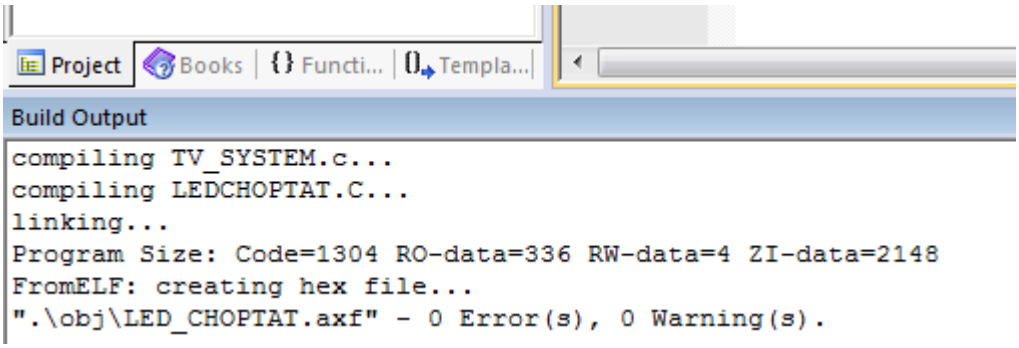
Hình 2-23. Chọn bộ nạp loại "ULINK2".

Tiến hành chọn tab "Utilities" rồi chọn mục Setting, một giao diện mới xuất hiện, tiến hành tick vào ô "Reset and Run" (số 1), bấm vào nút "Add" (số 2) thì một danh sách xuất hiện, bạn hãy chọn thông số giống như ở mục số 3 rồi nhấn ok là xong.



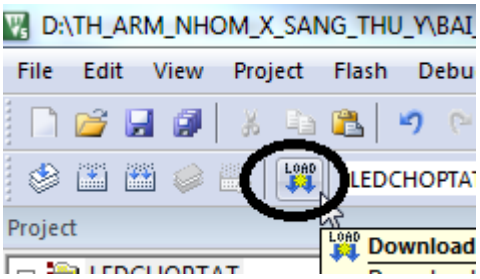
Hình 2-24. Chọn chế độ reset để nạp xong thì chạy và chọn loại bộ nhớ cho phù hợp chip.

Bước 10: Tiến hành biên dịch bằng cách và kết quả sau khi biên dịch như sau:



Hình 2-25. Kết quả sau khi biên dịch.

Bước 11: Tiến hành nạp bằng cách bấm vào biểu tượng LOAD.



Hình 2-26. Biểu tượng để nạp.

Bước 12: Sau khi nạp xong thì quan sát kết quả là 1 led chớp tắt.

Chú ý nhớ chọn chế độ nạp xong thì chạy thì bạn mới nhìn thấy led chớp tắt, nếu không chọn thì phải nhấn nút reset nếu viết đúng thì led chớp tắt.

Nội dung toàn bộ chương trình nguồn:

```
/************************************************************
CHUONG TRINH DIEU KHIEN 1 LED CHOP TAT
```

```
**********************************************************/
#include "stm32f10x.h"
#include "TV_DELAY.h"
#include "TV_LED.h"
#include "TV_PORT_CONF.h"
int main(void)
{
	SystemInit();
	PORT_CONF();
	LED_INIT_1();
	while(1)
	{
		GPIO_SetBits(GPIOD, GPIO_Pin_8);
		DELAY_X(0xEFFFF);

		GPIO_ResetBits(GPIOD, GPIO_Pin_8 );
		DELAY_X(0xEFFFF);
	}
}
```

III. GIẢI THÍCH CHƯƠNG TRÌNH ĐIỀU KHIỂN LED ĐƠN

Phần trên trình bày cho các bạn cách viết chương trình điều khiển 1 led chớp tắt.

Phần này giải thích các lệnh của chương trình chính, chương trình con, các kiểu dữ liệu có liên quan và các định nghĩa.

1. Giải thích các lệnh chương trình chính:

TT	LỆNH	GIẢI THÍCH
1	#include "stm32f10x.h" #include "TV_DELAY.h" #include "TV_LED.h" #include "TV_PORT_CONF.h"	Khai báo 4 file thư viện .h và kèm theo là .c
2	*Chương trình chính gọi hàm* SystemInit();	*Hàm này viết trong thư viện "system_stm32f10x.c"* *Nằm ở hàng thứ 180.* *Chức năng: cấu hình cho các thanh ghi liên quan đến reset, các bộ dao động HSE, LSE, LSI, PLL tạo xung clock cho hệ thống, nguồn xung clock cho các port và các ngoại vi.* ***Xem chi tiết bảng theo sau.***
3	*Chương trình chính tiếp tục gọi chương trình con:* PORT_CONF();	*Hàm này viết trong thư viện "PORT_CONF.c"* *Chức năng: cấu hình cho các ngoại vi là USART1, SPI1, ADC, các port A, B, C, D, E, AFIO ở chế độ cho phép.* *Lệnh này liên quan đến thanh ghi:* *APB2 peripheral clock enable register (RCC_APB2ENR)* ***Xem chi tiết bảng theo sau.***
4	*Chương trình chính tiếp tục gọi hàm:* LED_INIT_1();	*Hàm này viết trong thư viện "TV_LED.c".* *Chức năng: cấu hình cho port D có 4 bit 8, 9, 10, 11 kết nối với 4 led với cấu hình là port xuất, chế độ push-pull, tần số clock là 50MHz.*
5	*Sau khi khởi tạo xong thì vòng lặp*	

	while thực hiện lệnh GPIO_SetBits(GPIOD, GPIO_Pin_8);	*Hàm có chức năng làm chân thứ 8 của port D bằng 1 làm led sáng. Hàm này ở file "stm32f10x _gpio.h" hàng 351.* ***Xem chi tiết bảng theo sau.***
6	*Hàm delay:* Delay(0xEFFFF);	*Hàm có chức năng làm delay để nhìn thấy led sáng.*
7	*Hàm:* GPIO_ResetBits(GPIOD, GPIO_Pin_8););	*Hàm có chức năng làm chân thứ 8 của port D bằng 0 làm led tắt. Hàm này ở file "stm32f10x _gpio.h"hàng 367.* ***Xem chi tiết bảng theo sau.***
8	*Hàm delay:* Delay(0xEFFFF);	*Hàm có chức năng làm delay để nhìn thấy led sáng.*
9	*Các lệnh trong vòng lặp while thực hiện liên tục làm led chớp tắt.*	

2. Các chương trình con chi tiết trong thư viện:

TT	HÀM	GIẢI THÍCH
1	*Chi tiết chương trình con:* PORT_CONF();	
	RCC_APB2PeriphClockCmd (RCC_APB2Periph_USART1 RCC_APB2Periph_GPIOA |RCC_APB2Periph_GPIOB | RCC_APB2Periph_GPIOC |RCC_APB2Periph_GPIOD | RCC_APB2Periph_GPIOE |RCC_APB2Periph_ADC1 | RCC_APB2Periph_AFIO |RCC_APB2Periph_SPI1, ENABLE);	*Hàm này viết trong thư viện "stm32f10x_rcc.c"* *Nằm ở hàng 1076.* *Chức năng: cấu hình cho các ngoại vi là USART1, SPI1, ADC, các port A, B, C, D, E, AFIO ở chế độ cho phép.* *Lệnh này liên quan đến thanh ghi:* *APB2 peripheral clock enable register (RCC_APB2ENR)* ***Xem chi tiết bảng theo sau.***
2	*Chi tiết chương trình con* LED_INIT_1(void)	
	{ GPIO_InitTypeDef GPIO_InitStructure;	*Khai báo biến "GPIO_InitStructure" thuộc kiểu: GPIO_InitTypeDef được khai báo ở file "stm32f10x_gpio.h" ở hàng thứ 90.* *Bạn có thể thay đổi tên biến.* ***Xem chi tiết bảng theo sau.***
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8 | GPIO_Pin_9 | GPIO_Pin_10 | GPIO_Pin_11;	*Thuộc tính chân "GPIO_Pin" của biến là biến 16 bit sẽ được gán các giá trị của các chân lại bao gồm các chân thứ 8, 9, 10, 11 or lại với nhau.* *Kết quả GPIO_Pin=0x0F00* *Do các bit 8, 9, 10, 11 bằng 1.* *Các chân được định nghĩa ở hàng thứ 126 của file "stm32f10x_gpio.h".* ***Xem chi tiết bảng theo sau.***
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;	*Thuộc tính mode "GPIO_Mode" của biến được gán là mode xuất dữ liệu, chế độ push-pull.* *Thuộc tính "GPIO_Mode" được khai báo ở file "stm32f10x _gpio.h"hàng 98 với kiểu dữ liệu* GPIOMode_TypeDef. ***Xem chi tiết bảng theo sau.***
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;	

	GPIO_Init(GPIOD,&GPIO_InitStructure); }	*Thuộc tính mode "GPIO_Speed" của biến được gán là xung clock có tần số là 50MHz.* *Thuộc tính "GPIO_ Speed" được khai báo ở file "stm32f10x _gpio.h" hàng 95 với kiểu dữ liệu GPIOSpeed_TypeDef.* ***Xem chi tiết bảng theo sau.*** *Gọi hàm GPIO_Init(*GPIOD,&GPIO_InitStructure*) để thực hiện các yêu cầu.* *Thông số thứ nhất là chọn port để định cấu hình, và port D được chọn vì 4 led kết nối.* *Thông số thứ 2 dùng để cấu hình cho port.* ***Hàm** GPIO_Init **được giải thích chi tiết bên dưới.***
3	Hàm khởi tạo hệ thống ***SystemInit – hàm này luôn được thực hiện*** LƯU TRONG FILE *"system_stm32f10x.c"*	
	/** * @brief Setup the microcontroller system * Initialize the Embedded Flash Interface, the PLL and update the SystemFrequency variable. * @note This function should be used only after reset. * @param None * @retval None */ void SystemInit (void) { /* Reset the RCC clock configuration to the default reset state(for debug purpose) */ /* Set HSION bit */ RCC->CR |= (uint32_t)0x00000001; /* Reset SW, HPRE, PPRE1, PPRE2, ADCPRE and MCO bits */ #ifndef STM32F10X_CL RCC->CFGR &= (uint32_t)0xF8FF0000; #else RCC->CFGR &= (uint32_t)0xF0FF0000; #endif /* STM32F10X_CL */ /* Reset HSEON, CSSON and PLLON bits */ RCC->CR &= (uint32_t)0xFEF6FFFF; /* Reset HSEBYP bit */ RCC->CR &= (uint32_t)0xFFFBFFFF; /* Reset PLLSRC, PLLXTPRE, PLLMUL and USBPRE/OTGFSPRE bits */ RCC->CFGR &= (uint32_t)0xFF80FFFF; #ifndef STM32F10X_CL /* Disable all interrupts and clear pending bits */ RCC->CIR = 0x009F0000; #else /* Reset PLL2ON and PLL3ON bits */	

	RCC->CR &= (uint32_t)0xEBFFFFFF; /* Disable all interrupts and clear pending bits */ RCC->CIR = 0x00FF0000; /* Reset CFGR2 register */ RCC->CFGR2 = 0x00000000; #endif /* STM32F10X_CL */ /* Configure the System clock frequency, HCLK, PCLK2 and PCLK1 prescalers */ /* Configure the Flash Latency cycles and enable prefetch buffer */ SetSysClock(); }	
4	Hàm cho phép clock của ngoại vi: **RCC_APB2PeriphClockCmd**. LƯU TRONG FILE *"stm32f10x_rcc.c"*	

	/**
	* @brief Enables or disables the High Speed APB (APB2) peripheral clock.
	* @param RCC_APB2Periph: specifies the APB2 peripheral to gates its clock.
	* This parameter can be any combination of the following values:
	* @arg RCC_APB2Periph_AFIO, RCC_APB2Periph_GPIOA, RCC_APB2Periph_GPIOB,
	* RCC_APB2Periph_GPIOC, RCC_APB2Periph_GPIOD, RCC_APB2Periph_GPIOE,
	* RCC_APB2Periph_GPIOF, RCC_APB2Periph_GPIOG, RCC_APB2Periph_ADC1,
	* RCC_APB2Periph_ADC2, RCC_APB2Periph_TIM1, RCC_APB2Periph_SPI1,
	* RCC_APB2Periph_TIM8, RCC_APB2Periph_USART1, RCC_APB2Periph_ADC3
	* @param NewState: new state of the specified peripheral clock.
	* This parameter can be: ENABLE or DISABLE.
	* @retval None
	*/
	void RCC_APB2PeriphClockCmd(uint32_t RCC_APB2Periph, FunctionalState NewState)
	{
	/* Check the parameters */
	
	assert_param(IS_RCC_APB2_PERIPH(RCC_APB2Periph));
	assert_param(IS_FUNCTIONAL_STATE(NewState));
	if (NewState != DISABLE)
	{
	RCC->APB2ENR |= RCC_APB2Periph;
	}
	else
	{
	RCC->APB2ENR &= ~RCC_APB2Periph;
	}	

	}	
5	Hàm reset bit của port GPIO. LƯU TRONG FILE "*stm32f10x _gpio.c*"	

	/**
	* @brief Clears the selected data port bits.
	* @param GPIOx: where x can be (A..G) to select the GPIO peripheral.
	* @param GPIO_Pin: specifies the port bits to be written.
	* This parameter can be any combination of GPIO_Pin_x where x can be (0..15).
	*/
	void GPIO_ResetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
	{
	/* Check the parameters */
	assert_param(IS_GPIO_ALL_PERIPH(GPIOx));
	assert_param(IS_GPIO_PIN(GPIO_Pin));
	
	GPIOx->BRR = GPIO_Pin;
	}	

6	Hàm set bit của port GPIO. LƯU TRONG FILE "*stm32f10x _gpio.c*"	

	/**
	* @brief Sets the selected data port bits.
	* @param GPIOx: where x can be (A..G) to select the GPIO peripheral.
	* @param GPIO_Pin: specifies the port bits to be written.
	* This parameter can be any combination of GPIO_Pin_x where x can be (0..15).
	* @retval None
	*/
	void GPIO_SetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
	{
	/* Check the parameters */
	assert_param(IS_GPIO_ALL_PERIPH(GPIOx));
	assert_param(IS_GPIO_PIN(GPIO_Pin));
	
	GPIOx->BSRR = GPIO_Pin;
	}	

3. Các kiểu dữ liệu đã định nghĩa trong thư viện

TT	ĐỊNH NGHĨA CÁC DỮ LIỆU CÓ CẤU TRÚC	GIẢI THÍCH
1	ĐỊNH NGHĨA CÁC CHÂN CỦA 1 PORT 16 BIT LƯU TRONG FILE "*stm32f10x _gpio.h*"	
	#define GPIO_Pin_0 ((uint16_t)0x0001) /*!< Pin 0 selected */ #define GPIO_Pin_1 ((uint16_t)0x0002) /*!< Pin 1 selected */ #define GPIO_Pin_2 ((uint16_t)0x0004) /*!< Pin 2 selected */ #define GPIO_Pin_3 ((uint16_t)0x0008) /*!< Pin 3 selected */ #define GPIO_Pin_4 ((uint16_t)0x0010) /*!< Pin 4 selected */ #define GPIO_Pin_5 ((uint16_t)0x0020) /*!< Pin 5 selected */ #define GPIO_Pin_6 ((uint16_t)0x0040) /*!< Pin 6 selected */ #define GPIO_Pin_7 ((uint16_t)0x0080) /*!< Pin 7 selected */	

```
#define GPIO_Pin_8              ((uint16_t)0x0100)  /*!< Pin 8 selected */
#define GPIO_Pin_9              ((uint16_t)0x0200)  /*!< Pin 9 selected */
#define GPIO_Pin_10             ((uint16_t)0x0400)  /*!< Pin 10 selected */
#define GPIO_Pin_11             ((uint16_t)0x0800)  /*!< Pin 11 selected */
#define GPIO_Pin_12             ((uint16_t)0x1000)  /*!< Pin 12 selected */
#define GPIO_Pin_13             ((uint16_t)0x2000)  /*!< Pin 13 selected */
#define GPIO_Pin_14             ((uint16_t)0x4000)  /*!< Pin 14 selected */
#define GPIO_Pin_15             ((uint16_t)0x8000)  /*!< Pin 15 selected */
#define GPIO_Pin_All            ((uint16_t)0xFFFF)  /*!< All pins selected */
```

2 ĐỊNH NGHĨA GPIO_InitTypeDef

LƯU TRONG FILE "*stm32f10x _gpio.h"*

```
/** @brief  GPIO Init structure definition  */
typedef struct
{
        uint16_t                    GPIO_Pin;
        GPIOSpeed_TypeDef    GPIO_Speed;
        GPIOMode_TypeDef     GPIO_Mode;
} GPIO_InitTypeDef;
```

3 ĐỊNH NGHĨA GPIOMode_TypeDef

LƯU TRONG FILE "*stm32f10x _gpio.h"*

```
typedef enum
{
        GPIO_Mode_AIN = 0x0,
        GPIO_Mode_IN_FLOATING = 0x04,
        GPIO_Mode_IPD = 0x28,
        GPIO_Mode_IPU = 0x48,
        GPIO_Mode_Out_OD = 0x14,
        GPIO_Mode_Out_PP = 0x10,
        GPIO_Mode_AF_OD = 0x1C,
        GPIO_Mode_AF_PP = 0x18
} GPIOMode_TypeDef;
```

4 ĐỊNH NGHĨA DỮ LIỆU CÓ CẤU TRÚC: GPIOSpeed_TypeDef

LƯU TRONG FILE "*stm32f10x _gpio.h"*

```
typedef enum
{
        GPIO_Speed_10MHz = 1,
        GPIO_Speed_2MHz,
        GPIO_Speed_50MHz
}GPIOSpeed_TypeDef;
```

5 ĐỊNH NGHĨA ĐỊA CHỈ CÁC THANH GHI NỀN CỦA CÁC NGOẠI VI

LƯU TRONG FILE "stm32f10x.h"

```
** @addtogroup Peripheral_memory_map
  * @{
  */
#define PERIPH_BB_BASE        ((uint32_t)0x42000000)
        /*!< Peripheral base address in the alias region */
#define SRAM_BB_BASE          ((uint32_t)0x22000000)
        /*!< SRAM base address in the alias region */
#define SRAM_BASE             ((uint32_t)0x20000000)
        /*!< Peripheral base address in the bit-band region */
#define PERIPH_BASE           ((uint32_t)0x40000000)
        /*!< SRAM base address in the bit-band region */
```

```
#define FSMC_R_BASE          ((uint32_t)0xA0000000)
        /*!< FSMC registers base address */
```

6 ĐỊNH NGHĨA ĐỊA CHỈ CÁC THANH GHI NỀN CỦA CÁC NGOẠI VI LƯU TRONG FILE "stm32f10x.h"

```
 /*!< Peripheral memory map */    CCVGJK132SDBBBBBBBBBFOIEP
#define APB1PERIPH_BASE       PERIPH_BASE
#define APB2PERIPH_BASE       (PERIPH_BASE + 0x10000)
#define AHBPERIPH_BASE        (PERIPH_BASE + 0x20000)

#define TIM2_BASE             (APB1PERIPH_BASE + 0x0000)
#define TIM3_BASE             (APB1PERIPH_BASE + 0x0400)
#define TIM4_BASE             (APB1PERIPH_BASE + 0x0800)
#define TIM5_BASE             (APB1PERIPH_BASE + 0x0C00)
#define TIM6_BASE             (APB1PERIPH_BASE + 0x1000)
#define TIM7_BASE             (APB1PERIPH_BASE + 0x1400)
#define RTC_BASE              (APB1PERIPH_BASE + 0x2800)
#define WWDG_BASE             (APB1PERIPH_BASE + 0x2C00)
#define IWDG_BASE             (APB1PERIPH_BASE + 0x3000)
#define SPI2_BASE             (APB1PERIPH_BASE + 0x3800)
#define SPI3_BASE             (APB1PERIPH_BASE + 0x3C00)
#define USART2_BASE           (APB1PERIPH_BASE + 0x4400)
#define USART3_BASE           (APB1PERIPH_BASE + 0x4800)
#define UART4_BASE            (APB1PERIPH_BASE + 0x4C00)
#define UART5_BASE            (APB1PERIPH_BASE + 0x5000)
#define I2C1_BASE             (APB1PERIPH_BASE + 0x5400)
#define I2C2_BASE             (APB1PERIPH_BASE + 0x5800)
#define CAN1_BASE             (APB1PERIPH_BASE + 0x6400)
#define CAN2_BASE             (APB1PERIPH_BASE + 0x6800)
#define BKP_BASE              (APB1PERIPH_BASE + 0x6C00)
#define PWR_BASE              (APB1PERIPH_BASE + 0x7000)
#define DAC_BASE              (APB1PERIPH_BASE + 0x7400)

#define AFIO_BASE             (APB2PERIPH_BASE + 0x0000)
#define EXTI_BASE             (APB2PERIPH_BASE + 0x0400)

#define GPIOA_BASE            (APB2PERIPH_BASE + 0x0800)
#define GPIOB_BASE            (APB2PERIPH_BASE + 0x0C00)
#define GPIOC_BASE            (APB2PERIPH_BASE + 0x1000)
#define GPIOD_BASE            (APB2PERIPH_BASE + 0x1400)
#define GPIOE_BASE            (APB2PERIPH_BASE + 0x1800)
#define GPIOF_BASE            (APB2PERIPH_BASE + 0x1C00)
#define GPIOG_BASE            (APB2PERIPH_BASE + 0x2000)

#define ADC1_BASE             (APB2PERIPH_BASE + 0x2400)
#define ADC2_BASE             (APB2PERIPH_BASE + 0x2800)
#define TIM1_BASE             (APB2PERIPH_BASE + 0x2C00)
#define SPI1_BASE             (APB2PERIPH_BASE + 0x3000)
#define TIM8_BASE             (APB2PERIPH_BASE + 0x3400)
#define USART1_BASE           (APB2PERIPH_BASE + 0x3800)
#define ADC3_BASE             (APB2PERIPH_BASE + 0x3C00)

#define SDIO_BASE             (PERIPH_BASE + 0x18000)

#define DMA1_BASE             (AHBPERIPH_BASE + 0x0000)
#define DMA1_Channel1_BASE    (AHBPERIPH_BASE + 0x0008)
#define DMA1_Channel2_BASE    (AHBPERIPH_BASE + 0x001C)
```

```
#define DMA1_Channel3_BASE    (AHBPERIPH_BASE + 0x0030)
#define DMA1_Channel4_BASE    (AHBPERIPH_BASE + 0x0044)
#define DMA1_Channel5_BASE    (AHBPERIPH_BASE + 0x0058)
#define DMA1_Channel6_BASE    (AHBPERIPH_BASE + 0x006C)
#define DMA1_Channel7_BASE    (AHBPERIPH_BASE + 0x0080)
#define DMA2_BASE           (AHBPERIPH_BASE + 0x0400)
#define DMA2_Channel1_BASE    (AHBPERIPH_BASE + 0x0408)
#define DMA2_Channel2_BASE    (AHBPERIPH_BASE + 0x041C)
#define DMA2_Channel3_BASE    (AHBPERIPH_BASE + 0x0430)
#define DMA2_Channel4_BASE    (AHBPERIPH_BASE + 0x0444)
#define DMA2_Channel5_BASE    (AHBPERIPH_BASE + 0x0458)
#define RCC_BASE            (AHBPERIPH_BASE + 0x1000)
#define CRC_BASE            (AHBPERIPH_BASE + 0x3000)

#define FLASH_R_BASE        (AHBPERIPH_BASE + 0x2000)
#define OB_BASE             ((uint32_t)0x1FFFF800)
#define ETH_BASE            (AHBPERIPH_BASE + 0x8000)
#define ETH_MAC_BASE        (ETH_BASE)
#define ETH_MMC_BASE        (ETH_BASE + 0x0100)
#define ETH_PTP_BASE        (ETH_BASE + 0x0700)
#define ETH_DMA_BASE        (ETH_BASE + 0x1000)

#define FSMC_Bank1_R_BASE    (FSMC_R_BASE + 0x0000)
#define FSMC_Bank1E_R_BASE   (FSMC_R_BASE + 0x0104
#define FSMC_Bank2_R_BASE    (FSMC_R_BASE + 0x0060)
#define FSMC_Bank3_R_BASE    (FSMC_R_BASE + 0x0080)
#define FSMC_Bank4_R_BASE    (FSMC_R_BASE + 0x00A0)

#define DBGMCU_BASE         ((uint32_t)0xE0042000)
```

7 ĐỊNH NGHĨA CÁC NGOẠI VI: ĐỊNH NGHĨA NGOẠI VI CÓ ĐỊA CHỈ NỀN TƯƠNG ỨNG VÀ SỬ DỤNG CÁC THÀNH PHẦN KHAI BÁO TRONG CẤU TRÚC TƯƠNG ỨNG.
LƯU TRONG FILE “stm32f10x.h”

```
/** @addtogroup Peripheral_declaration
  * @{
  */

#define TIM2                ((TIM_TypeDef *) TIM2_BASE)
#define TIM3                ((TIM_TypeDef *) TIM3_BASE)
#define TIM4                ((TIM_TypeDef *) TIM4_BASE)
#define TIM5                ((TIM_TypeDef *) TIM5_BASE)
#define TIM6                ((TIM_TypeDef *) TIM6_BASE)
#define TIM7                ((TIM_TypeDef *) TIM7_BASE)
#define RTC                 ((RTC_TypeDef *) RTC_BASE)
#define WWDG                ((WWDG_TypeDef *) WWDG_BASE)
#define IWDG                ((IWDG_TypeDef *) IWDG_BASE)
#define SPI2                ((SPI_TypeDef *) SPI2_BASE)
#define SPI3                ((SPI_TypeDef *) SPI3_BASE)
#define USART2              ((USART_TypeDef *) USART2_BASE)
#define USART3              ((USART_TypeDef *) USART3_BASE)
#define UART4               ((USART_TypeDef *) UART4_BASE)
#define UART5               ((USART_TypeDef *) UART5_BASE)
#define I2C1                ((I2C_TypeDef *) I2C1_BASE)
#define I2C2                ((I2C_TypeDef *) I2C2_BASE)
#define CAN1                ((CAN_TypeDef *) CAN1_BASE)
```

```
#define CAN2              ((CAN_TypeDef *) CAN2_BASE)
#define BKP               ((BKP_TypeDef *) BKP_BASE)
#define PWR               ((PWR_TypeDef *) PWR_BASE)
#define DAC               ((DAC_TypeDef *) DAC_BASE)
#define AFIO              ((AFIO_TypeDef *) AFIO_BASE)
#define EXTI              ((EXTI_TypeDef *) EXTI_BASE)
#define GPIOA             ((GPIO_TypeDef *) GPIOA_BASE)
#define GPIOB             ((GPIO_TypeDef *) GPIOB_BASE)
#define GPIOC             ((GPIO_TypeDef *) GPIOC_BASE)
#define GPIOD             ((GPIO_TypeDef *) GPIOD_BASE)
#define GPIOE             ((GPIO_TypeDef *) GPIOE_BASE)
#define GPIOF             ((GPIO_TypeDef *) GPIOF_BASE)
#define GPIOG             ((GPIO_TypeDef *) GPIOG_BASE)
#define ADC1              ((ADC_TypeDef *) ADC1_BASE)
#define ADC2              ((ADC_TypeDef *) ADC2_BASE)
#define TIM1              ((TIM_TypeDef *) TIM1_BASE)
#define SPI1              ((SPI_TypeDef *) SPI1_BASE)
#define TIM8              ((TIM_TypeDef *) TIM8_BASE)
#define USART1            ((USART_TypeDef *) USART1_BASE)
#define ADC3              ((ADC_TypeDef *) ADC3_BASE)
#define SDIO              ((SDIO_TypeDef *) SDIO_BASE)
#define DMA1              ((DMA_TypeDef *) DMA1_BASE)
#define DMA2              ((DMA_TypeDef *) DMA2_BASE)
#define DMA1_Channel1     ((DMA_Channel_TypeDef *) DMA1_Channel1_BASE)
#define DMA1_Channel2     ((DMA_Channel_TypeDef *) DMA1_Channel2_BASE)
#define DMA1_Channel3     ((DMA_Channel_TypeDef *) DMA1_Channel3_BASE)
#define DMA1_Channel4     ((DMA_Channel_TypeDef *) DMA1_Channel4_BASE)
#define DMA1_Channel5     ((DMA_Channel_TypeDef *) DMA1_Channel5_BASE)
#define DMA1_Channel6     ((DMA_Channel_TypeDef *) DMA1_Channel6_BASE)
#define DMA1_Channel7     ((DMA_Channel_TypeDef *) DMA1_Channel7_BASE)
#define DMA2_Channel1     ((DMA_Channel_TypeDef *) DMA2_Channel1_BASE)
#define DMA2_Channel2     ((DMA_Channel_TypeDef *) DMA2_Channel2_BASE)
#define DMA2_Channel3     ((DMA_Channel_TypeDef *) DMA2_Channel3_BASE)
#define DMA2_Channel4     ((DMA_Channel_TypeDef *) DMA2_Channel4_BASE)
#define DMA2_Channel5     ((DMA_Channel_TypeDef *) DMA2_Channel5_BASE)
#define RCC               ((RCC_TypeDef *) RCC_BASE)
#define CRC               ((CRC_TypeDef *) CRC_BASE)
#define FLASH             ((FLASH_TypeDef *) FLASH_R_BASE)
#define OB                ((OB_TypeDef *) OB_BASE)
#define ETH               ((ETH_TypeDef *) ETH_BASE)
#define FSMC_Bank1        ((FSMC_Bank1_TypeDef *) FSMC_Bank1_R_BASE)
#define FSMC_Bank1E       ((FSMC_Bank1E_TypeDef *) FSMC_Bank1E_R_BASE)
#define FSMC_Bank2        ((FSMC_Bank2_TypeDef *) FSMC_Bank2_R_BASE)
#define FSMC_Bank3        ((FSMC_Bank3_TypeDef *) FSMC_Bank3_R_BASE)
#define FSMC_Bank4        ((FSMC_Bank4_TypeDef *) FSMC_Bank4_R_BASE)
#define DBGMCU            ((DBGMCU_TypeDef *) DBGMCU_BASE)
```

8 ĐỊNH NGHĨA CÁC THÀNH PHẦN TRONG CẤU TRÚC CỦA GPIO CÓ TÊN LÀ "GPIO_TypeDef".
LƯU TRONG FILE "stm32f10x.h"

```
typedef struct
{
  __IO uint32_t    CRL;
  __IO uint32_t    CRH;
  __IO uint32_t    IDR;
  __IO uint32_t    ODR;
```

	__IO uint32_t BSRR; __IO uint32_t BRR; __IO uint32_t LCKR; } GPIO_TypeDef;	
	SAU KHI ĐỊNH NGHĨA XONG THÌ CÁCH BẠN TRUY XUẤT PHẦN TỬ CỦA MỘT NGOẠI VI VÍ NHƯ SAU: GPIOD->BRR = GPIO_Pin; CÓ CHỨC NĂNG GÁN GIÁ TRỊ CỦA 'GPIO_Pin' CHO THANH GHI BRR CỦA PORT D.	

IV. CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN LED ĐƠN

Sau khi thực hiện xong bài 2-1 thì tiến hành làm các bài tập sau.

Bài tập 202. Hãy viết chương trình điều khiển 4 LED đơn sáng tắt dần.

Tạo thư mục "BAI_202_4LED_STD" để lưu project.

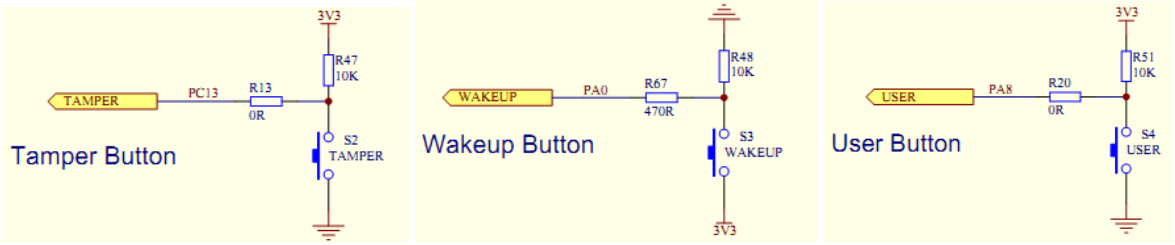
Bài tập 203. Hãy viết chương trình điều khiển 4 LED đơn sáng dần từ phải sang trái rồi tắt dần, sau đó đảo chiều.

Tạo thư mục "BAI_203_4LED_STD_2C" để lưu project.

V. CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN LED ĐƠN VÀ NÚT NHẤN

Kit thực hành có 4 nút nhấn có tên là S1, S2, S3, S4. S1 là nút reset CPU nên chỉ sử 3 nút nhấn còn lại để điều khiển.

Sơ đồ mạch giao tiếp 3 nút nhấn:



Hình 2-26. Sơ đồ mạch các nút nhấn.

Nút nhấn S2 còn có tên là Tamper nối đến port C bit thứ 13, khi không nhấn thì mức logic vào là 1, khi nhấn thì lên mức logic 0.

Nút nhấn S3 còn có tên là Wakeup nối đến port A bit thứ 0, khi không nhấn thì mức logic vào là 0, khi nhấn thì lên mức logic 1.

Nút nhấn S4 còn có tên là User nối đến port A bit thứ 8, khi không nhấn thì mức logic vào là 0, khi nhấn thì lên mức logic 1.

Chức năng Tamper và Wakeup sẽ được đề cập sau, phần này sử dụng cả 3 nút nhấn như các nút nhấn bình thường.

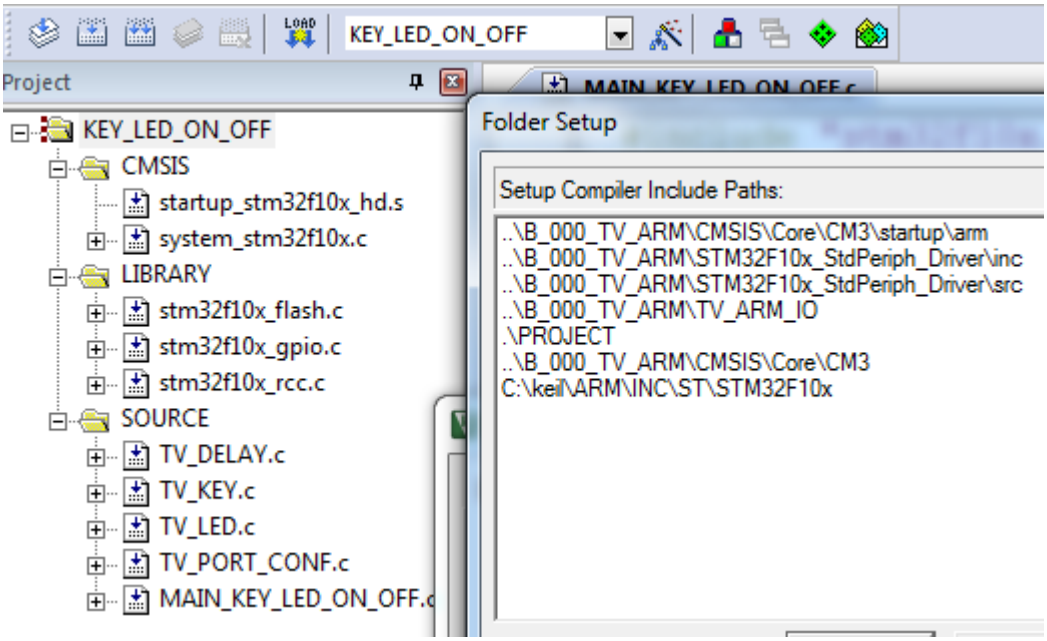
Chú ý mức logic của S3 và S4 ngược với S2.

Bài mẫu 211. Chương trình điều khiển LED bằng 2 nút nhấn ON và OFF, khi nhấn ON thì led sáng, khi nhấn OFF thì led tắt.

Tạo thư mục "BAI_211_KEY_LED_ON_OFF" để lưu project.

a. Lưu đồ:

b. Hình các file chính và thư viện:



Hình 2-27. Khai báo các nhóm và khai báo đường dẫn.

c. Chương trình:

```
#include "stm32f10x.h"
#include "TV_DELAY.h"
#include "TV_LED.h"
#include "TV_KEY.h"
#include "TV_PORT_CONF.h"
uint16_t KEYON,KEYOFF;
int main(void)
{
        SystemInit();   PORT_CONF();    LED_INIT_1();   KEY_INIT_1();

        while(1)
        {
          do
          {
            KEYON = GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_0);
          }while    (KEYON ==0);
          GPIO_SetBits(GPIOD, GPIO_Pin_8);
          do
          {
            KEYOFF = GPIO_ReadInputDataBit(GPIOC,GPIO_Pin_13);
          }while    (KEYOFF ==1);
          GPIO_ResetBits(GPIOD, GPIO_Pin_8);
        }
}
```

d. Tiến hành biên dịch và nạp.
e. Quan sát kết quả: nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.
f. Giải thích chương trình: Chương trình chính gọi hàm khởi tạo hệ thống "SystemInut();" nằm trong file system_stm32f10x.c, khởi tạo port, khởi tạo port nối với led, khởi tạo port nối 3 nút nhấn. Vòng lặp do while thực hiện kiểm tra nút nhấn ON, nếu nhấn ON thì làm led sáng và chỉ thoát khi buông phím nhấn, vòng lặp do while thứ 2 kiểm tra nút OFF nếu có nhấn thì làm led tắt.
Chú ý trạng thái của các nút nhấn ngược nhau.
g. Cho phép thay đổi: bằng led khác hoặc phím nhấn khác.
h. Chương trình thư viện "TV_KEY.c"

```
#include "stm32f10x.h"
#include "TV_KEY.h"
void KEY_INIT_1(void)
{
	GPIO_InitTypeDef GPIO_KEY;

	GPIO_KEY.GPIO_Pin = GPIO_Pin_0;
	GPIO_KEY.GPIO_Mode = GPIO_Mode_IPD;
	GPIO_KEY.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_Init(GPIOA,&GPIO_KEY);

	GPIO_KEY.GPIO_Pin = GPIO_Pin_8;
	GPIO_KEY.GPIO_Mode = GPIO_Mode_IPU;
	GPIO_Init(GPIOA,&GPIO_KEY);

	GPIO_KEY.GPIO_Pin = GPIO_Pin_13;
	GPIO_KEY.GPIO_Mode = GPIO_Mode_IPU;
	GPIO_Init(GPIOC, &GPIO_KEY);
}
```

Trong chương trình thư viện nút nhấn có chức năng khởi tạo các port tương ứng với 3 nút nhấn là ngõ vào, điện trở kéo xuống và kéo lên, tần số.

Chương trình thư viện led có chức năng khởi tạo các port tương ứng với 4 led là ngõ ra, điện trở kéo xuống và kéo lên, tần số.

i. Chương trình thư viện "TV_LED.c"

```
#include "stm32f10x.h"
#include "TV_LED.h"
void LED_INIT_1(void)
{
	GPIO_InitTypeDef GPIO_4LED;
	GPIO_4LED.GPIO_Pin = GPIO_Pin_8 | GPIO_Pin_9 | GPIO_Pin_10 | GPIO_Pin_11;
	GPIO_4LED.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO_4LED.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_Init(GPIOD, &GPIO_4LED);
}
void LED_INIT_2(void)
```

```
{
	RCC->APB2ENR|=1<<5;
	GPIOD->CRH&=0XFFFF0000;
	GPIOD->CRH|=0X00003333;//PD8~PD11 LA OUT VI NOI VOI 4 LED
}
void BUZZER_RB5(void)
{
	GPIO_InitTypeDef GPIO_BUZZER;
	GPIO_BUZZER.GPIO_Pin = GPIO_Pin_5 ;
	GPIO_BUZZER.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO_BUZZER.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_Init(GPIOB, &GPIO_BUZZER);
}
```

Bài tập 212. Hãy viết chương trình điều khiển LED bằng 1 nút nhấn ON_OFF dùng S3: khi led tắt thì nút nhấn có chức năng ON nếu nhấn thì làm led sáng, khi led sáng thì nút có chức năng OFF nếu nhấn sẽ làm led tắt.

Tạo thư mục "BAI_212_1KEY_LED_ON_OFF" để lưu project

Bài tập 213. Giống bài 2-12 nhưng 3 nút nhấn S1, S2, S3 cho 3 led tương ứng LD1, LD2 và LD3.

Tạo thư mục "BAI_213_3KEY_3LED_ON_OFF" để lưu project