The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

EIE3105 Integrated Project (Part I)

Laboratory Exercise 5: ARM Programming

(Deadline: Check the course information)

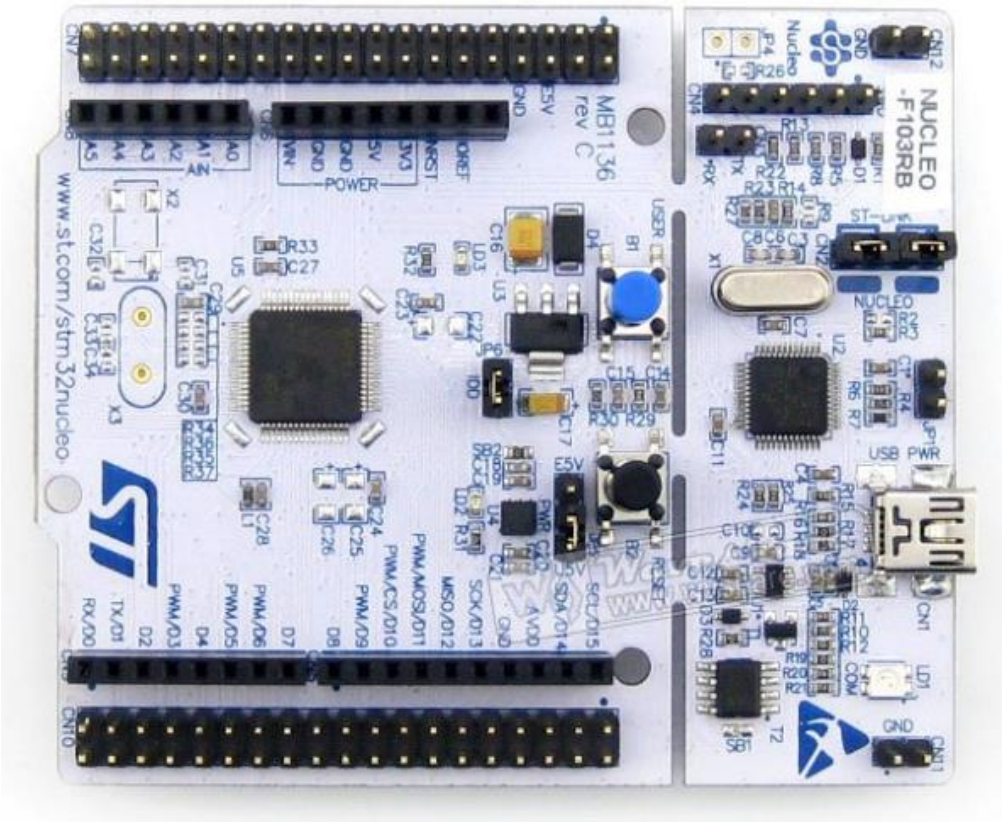
Objective:

At the end of the lab exercise, students will be able to

1. Create a Keil project using standard peripherals library to program STM32.
2. Configure general purpose I/O.
3. Use simple counting loop for delay.
4. Use STM32 system core clock for delay.
5. Use the on-board user button and LED.

Equipment:

Keil uVision5 with ARM support (software)
STM32F103RBT6 (hardware)



Important Notices:

1. You must read STM32F103RBT6 pinout diagrams very carefully before you do the laboratory exercises. You can download the pdf file from the course web site.
2. You must read R0008 Reference Manual very carefully before you do the laboratory exercises. You can download the pdf file from the course web site.
3. You leave your student identity card to our technicians to borrow one box of the kit. They will return your card to you when you return the box. Note that you need to return the box 5 minutes before the end of the laboratory session; otherwise, our technicians will pass your card to the instructor and you will have penalties when it happens.

Introduction:

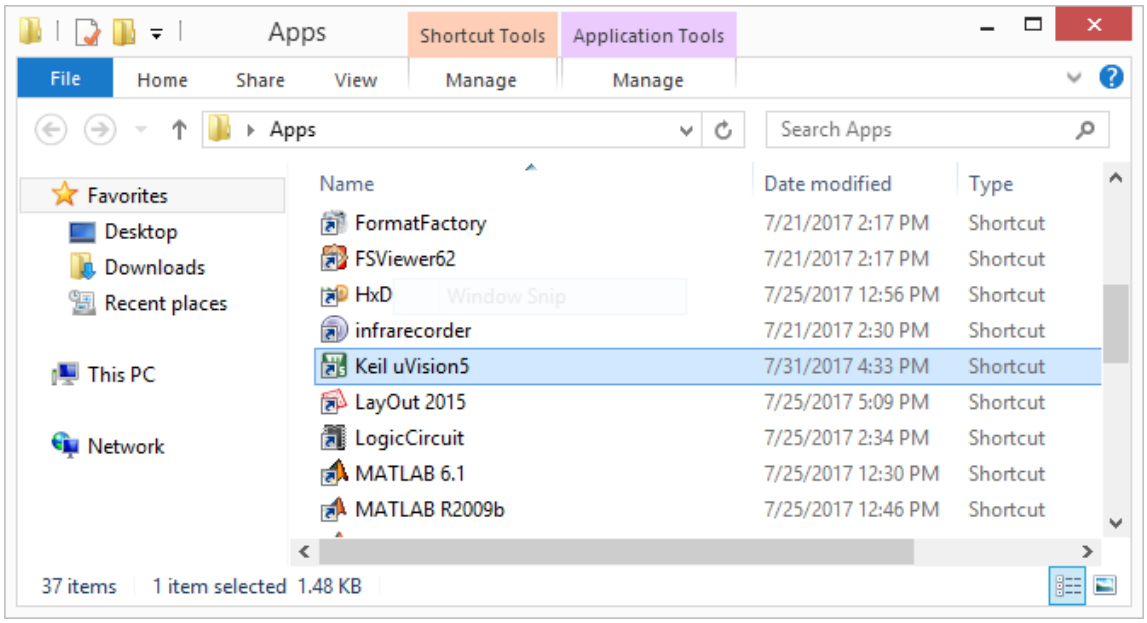
The STM32F103RB medium-density performance line family incorporates the high-performance ARM®Cortex®-M3 32-bit RISC core operating at a 72 MHz frequency, high-speed embedded memories (Flash memory up to 128 Kbytes and SRAM up to 20 Kbytes), and an extensive range of enhanced I/Os and peripherals connected to two APB buses. The STM32F103RB offers ADCs, general purpose 16-bit timers plus one PWM timer, as well as standard and advanced communication interfaces: I2Cs and SPIs, three USARTs, an USB and a CAN.

This lab exercise introduces STM32F103RB using Keil and C language. Students will be familiar with the basic I/Os and operate the on-board components.

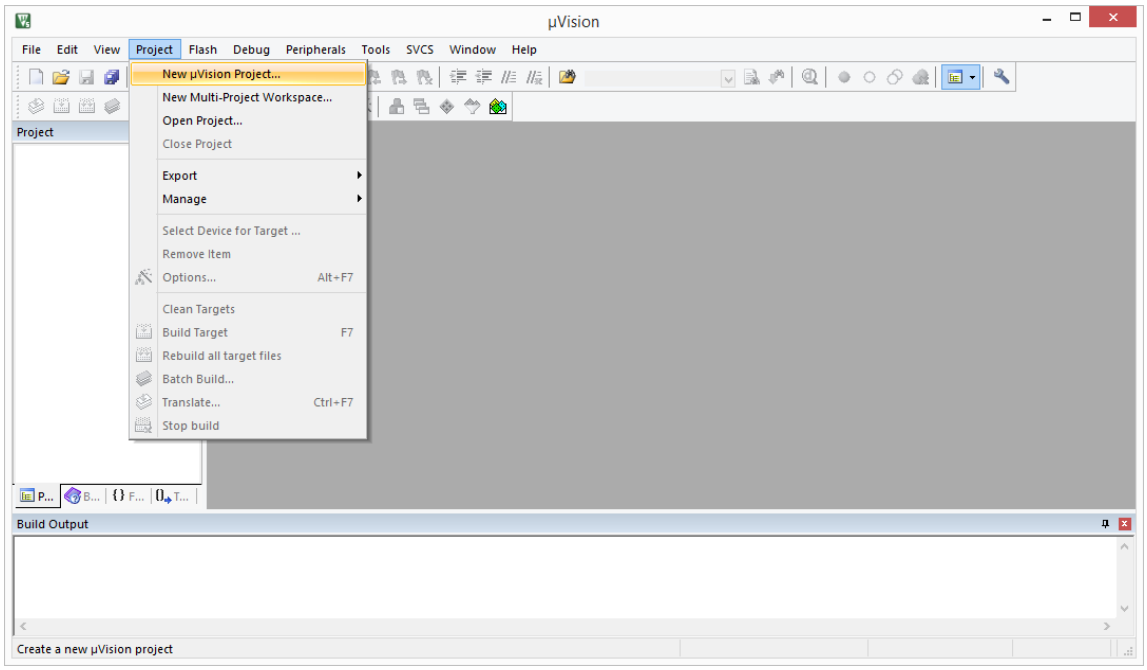
Procedure:

Section A: Create a Keil project.

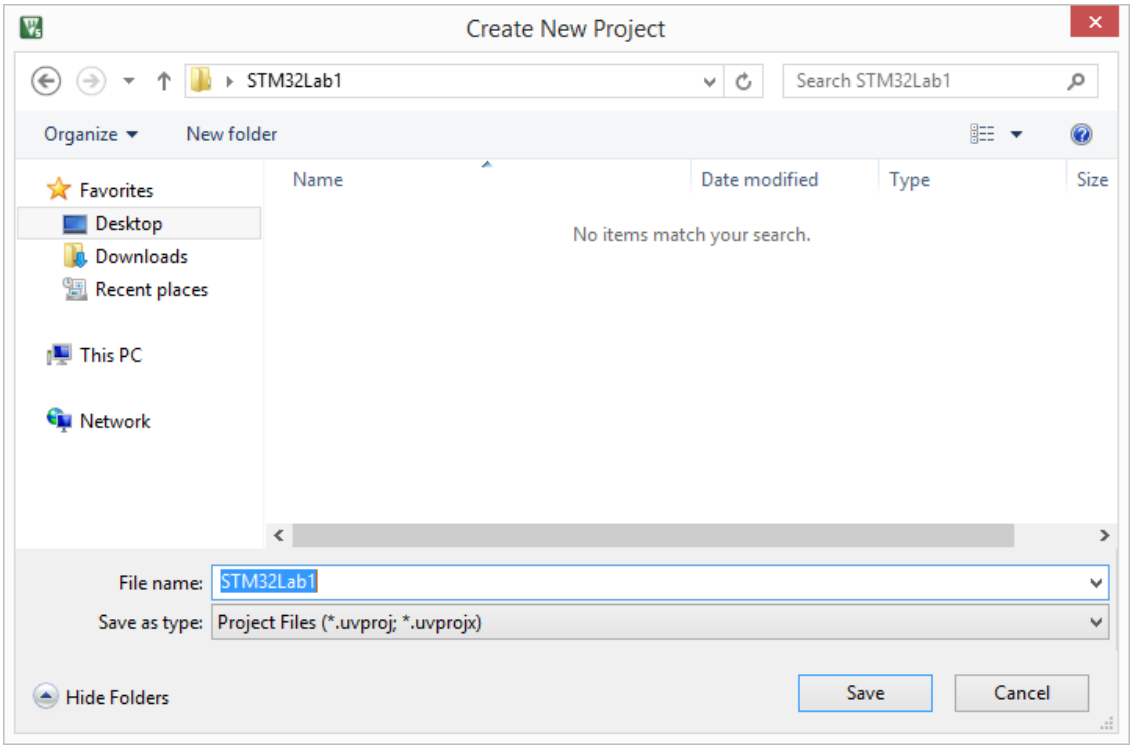
1. Open the **App** folder on Desktop and open **Keil uVision 5**.



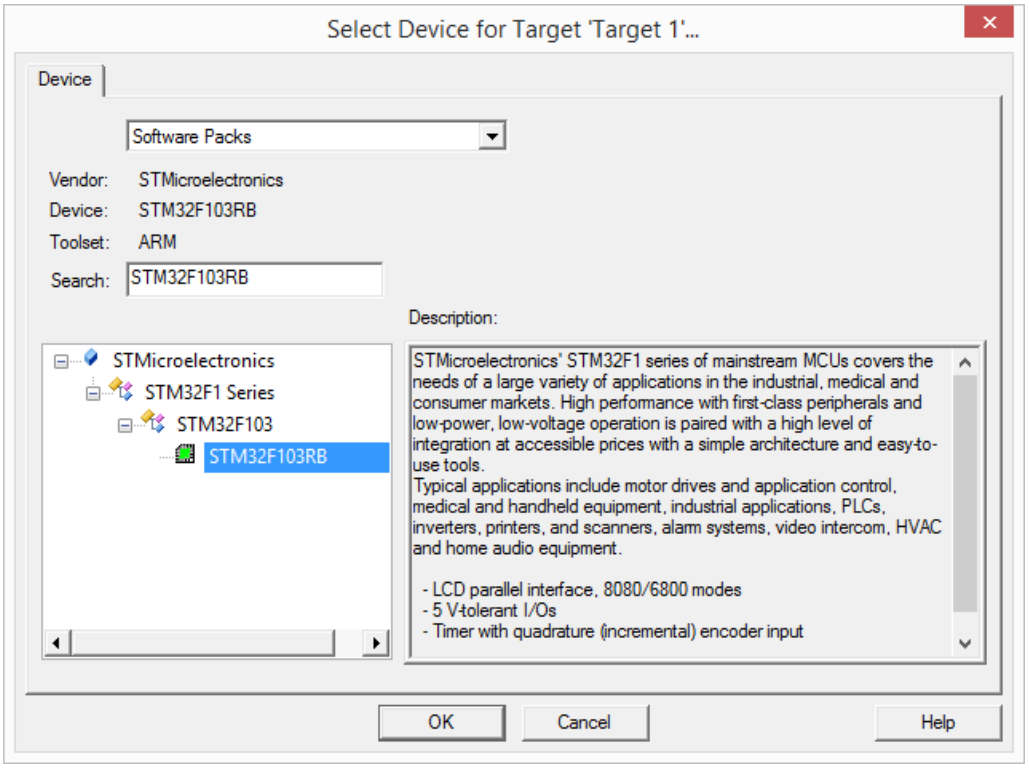
2. Click **Project** and then **New uVision Project**.



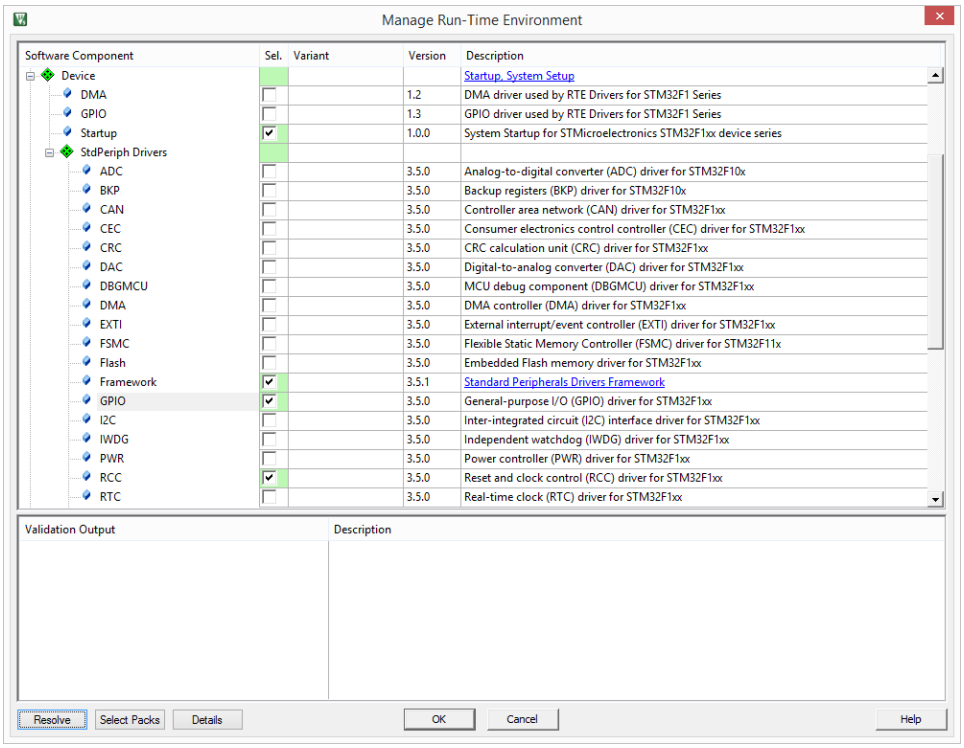
3. Create **a new folder** on Desktop first. Then choose the folder as the location to create a new project. You may name the project **STM32Lab1**.



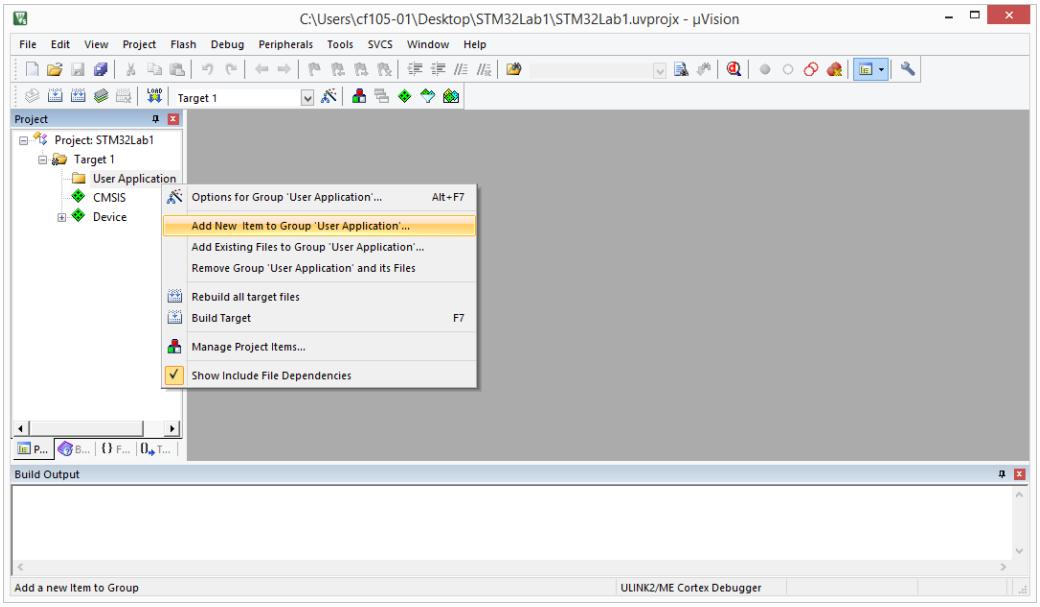
4. Select Device by searching "**STM32F103RB**". Choose **STM32F103RB** and click **OK**.



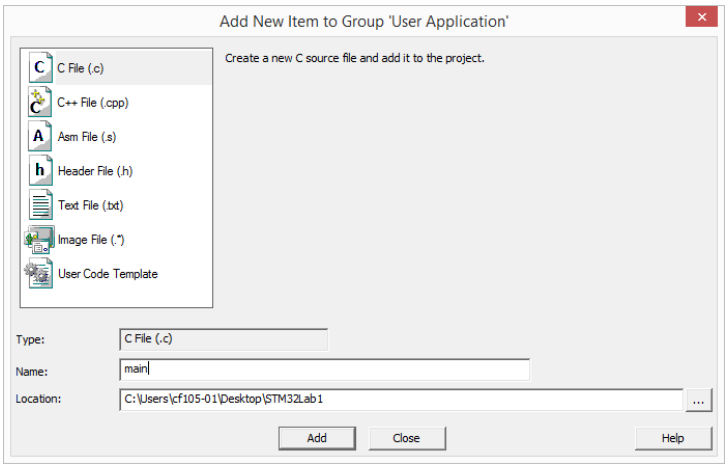
5. Select **Manage Run-Time Environment**. Check **Device ➔ Startup** and **Device ➔ StdPeriphDrivers ➔** select **Startup**, **Framework**, **GPIO**, and **RCC**. Click **Resolve** at the lower left corner to include all the necessary libraries for the drivers you selected. Validation Output should be **empty** and click OK. This is to select devices for your program. Later, you may add more devices in your laboratory exercises.



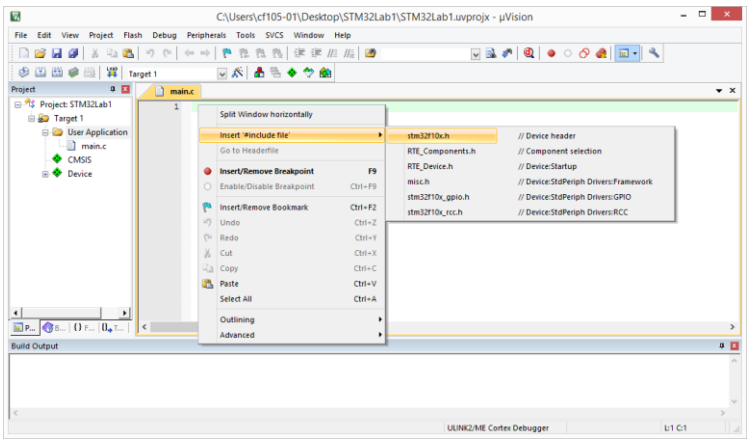
6. Organize the project explorer. Rename **Source Group 1** to **User Application**. Right click on **User Application ➔ Add New item to Group "User Application"**.



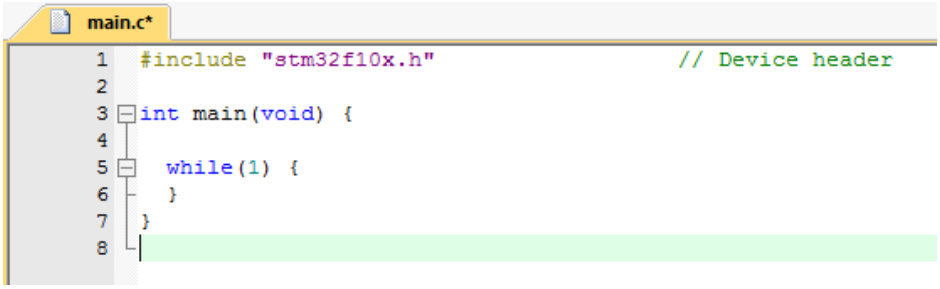
7. Choose **C File(.c)** and name it **main**. Then click **Add**.



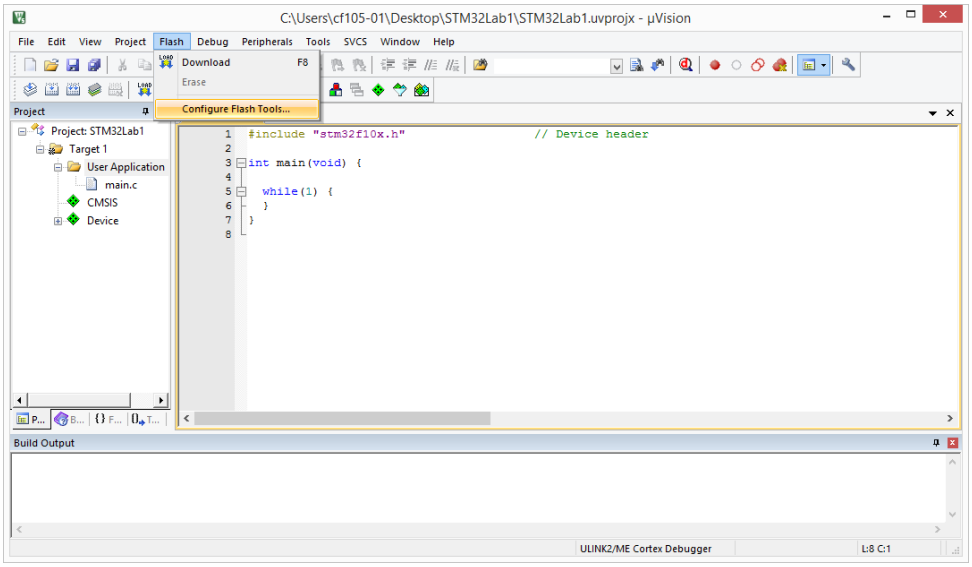
8. Open **main.c**. **Right click** the first line ➔ **Insert "#include file"** ➔ **stm32f10x.h**. The header file stm32f10x.h defines all the registers and constants for the system.



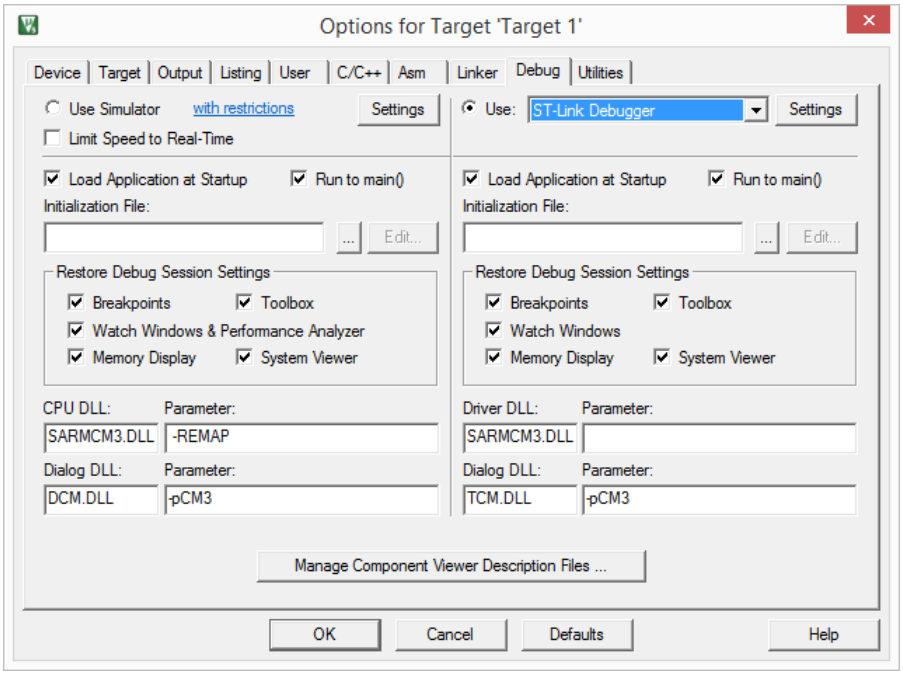
9. Create a main function.



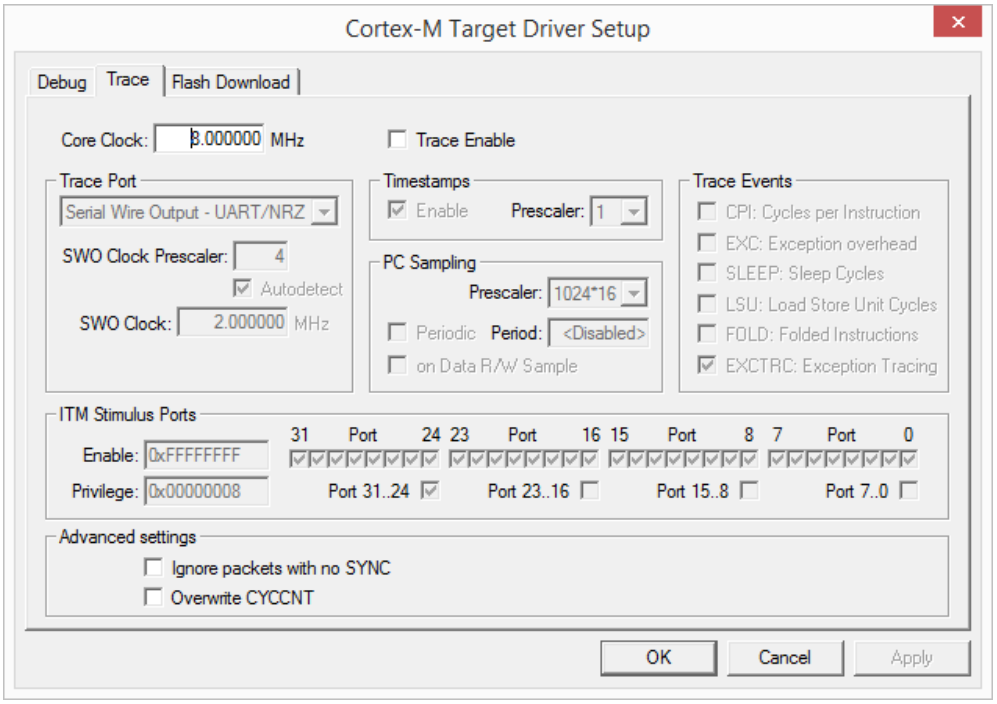
10. **Flash ➔ Configure Flash Tools**



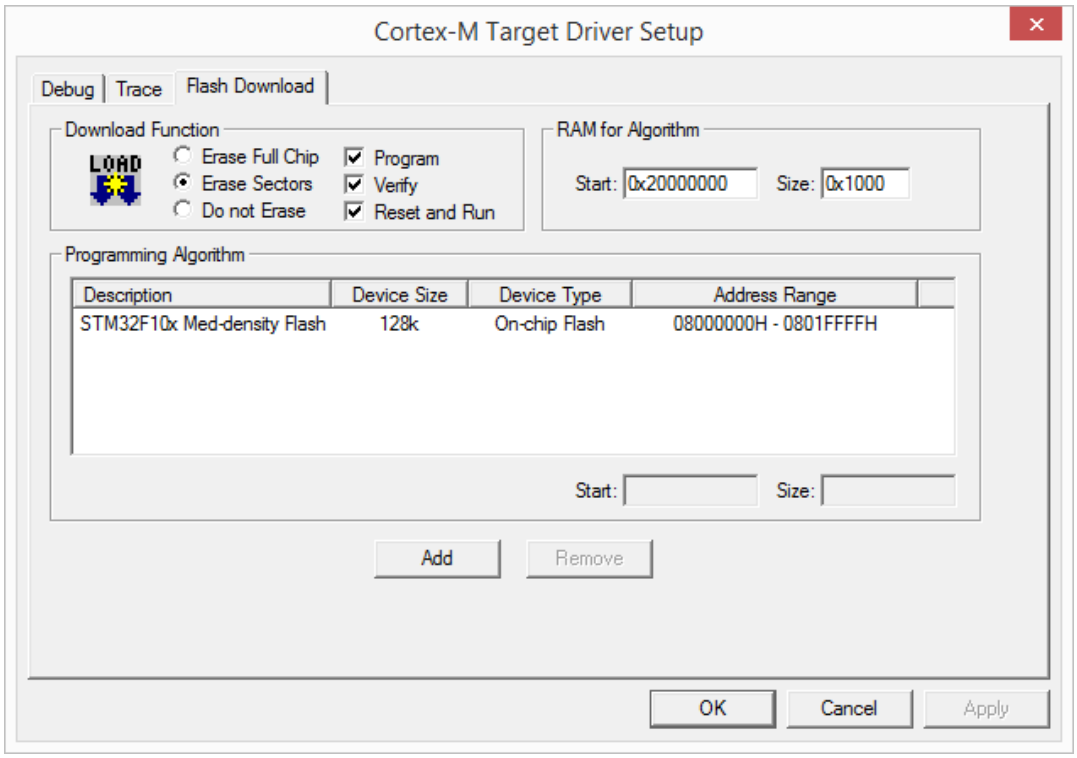
11. **Debug ➔** choose **ST-Link Debugger** (In your PC, you need to install ST Link Utility.)



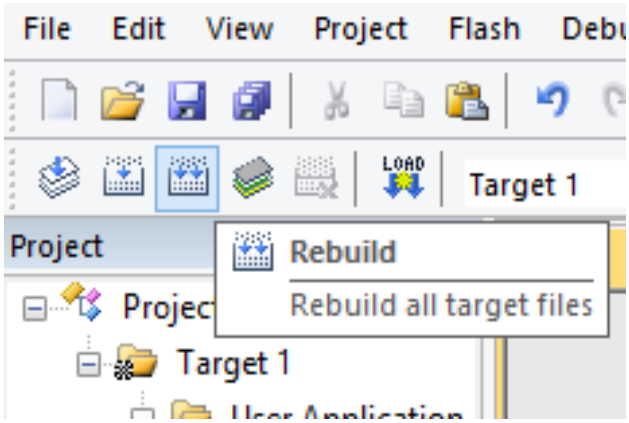
12. On the same **Debug** tab ➔ **Settings** ➔ **Trace**. Change Core Clock to **8** MHz.



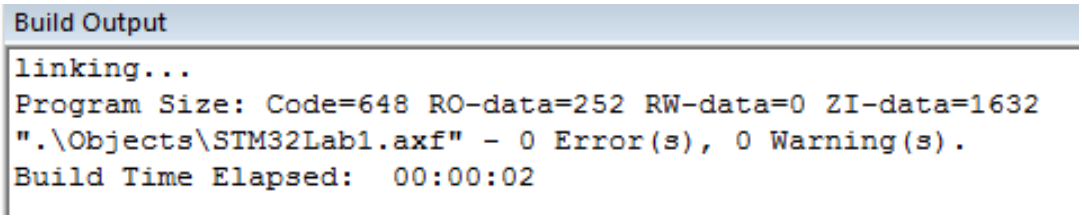
13. Go to **Flash Download** tab. Check **Reset and Run**. Then click **OK**.



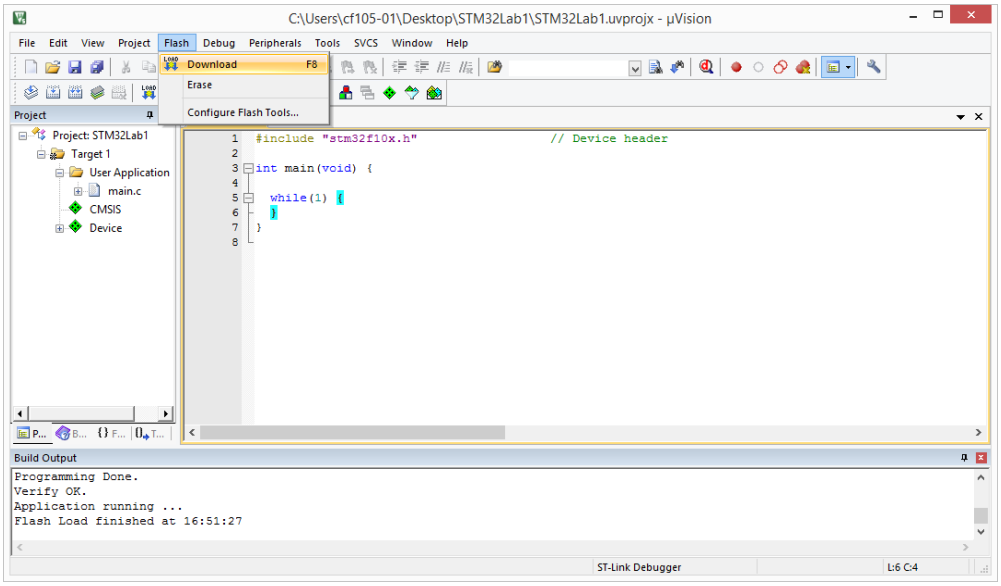
14. Click the **Rebuild** button to rebuild the project, or **Project ➔ Rebuild all target files**



15. Fix any errors and warnings.



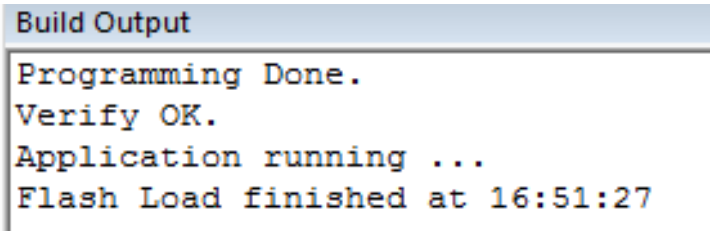
16. Download the program to the ARM microcontroller. **Flash ➔ Download**



Note:

1. You may need to go to Keil_v5/ARM/STLink/USBDriver to execute dpinst_amd64 installer for your ST-Link debugger.
2. Upgrade the ST-LINK firmware if a prompt message is requested. It is also fine if you do not upgrade it.

17. Build Output for successful download. Of course, you will not see anything happen on the board for an empty main function.



18. Type in the following statements in Line 4. Build the solution and download the program to the microcontroller. You should see the on-board LED is ON.

```
//GPIO set up for PA5 (on-board LED)
RCC->APB2ENR |= RCC_APB2Periph_GPIOA; //Enable APB2 peripheral clock
GPIOA->CRL &= ~0x00F00000;  //clear the setting
GPIOA->CRL |=  0 << 22 | 2 << 20; //GPIO_Mode_Out_PP, GPIO_Speed_2MHz
GPIOA->BSRR |= 0x20;
```

Section B: Flash the on-board LED (PA5) by using a delay loop.

Flash the on-board LED (PA5) by using a delay loop. The duty cycle should be 50%. The delay should be around one to two seconds. You may consider the following delay loop to generate the delay.

```
void delay(int t) {
     int i;
     for(i = 0; i < t; i++)
          GPIOA->BSRR |= 0x01; // do something to PA0
}
```

Section C: Flash the on-board LED (PA5) by using Systick.

Repeat Section B but this time you should use the standard peripheral function Systick.

Section D: Use the on-board button (PC13) to switch on and off the on-board LED (PA5).

When the button is pressed, the LED is on. When the button is released, the LED is off.

Section E: Use the on-board button (PC13) to change the state of the on-board LED (PA5).

There are two states in the button: State 0 and 1. When it is in State 0, the LED is off. When it is in State 1, the LED is on. At the beginning, the button is in State 0. When the button is pressed and it is in State 0, it goes to State 1. When the button is pressed and it is in State 1, it goes to State 0.

Section F: Write a C Program to simulate the traffic lights

Write a C program to simulate the traffic lights by using different pins. You can use any pins to simulate the traffic lights. You must use interrupt to implement the application.

A set of traffic lights for cars (Light 3, 3 LEDs)
A set of traffic lights for cars (Light 2, 3 LEDs)
A set of traffic lights for people (Light 1, 2 LEDs)

Repeat the following:
Light 1 (RED), Light 2 (GREEN), Light 3 (RED), period (around 5s)
Light 1 (RED), Light 2 (YELLOW), Light 3 (RED), period (around 1s)
Light 1 (RED), Light 2 (RED), Light 3 (RED), period (around 1s)
Light 1 (RED), Light 2 (RED), Light 3 (RED+YELLOW), period (around 1s)
Light 1 (GREEN), Light 2 (RED), Light 3 (GREEN), period (around 5s)
Light 1 (GREEN Blinking), Light 2 (RED), Light 3 (YELLOW), period (around 1s)
Light 1 (RED), Light 2 (RED), Light 3 (RED), period (around 1s)
Light 1 (RED), Light 2 RED+YELLOW), Light 3 (RED), period (around 1s)

Section G: Write a C program to count a switch

Connect a switch to a pin and a LED to another pin. There are two states in the switch: State 0 and 1. When it is in State 0, the LED is off. When it is in State 1, the LED is on. At the beginning, the switch is in State 0. When the switch is pressed three times and it is in State 0, it goes to State 1. When the switch is pressed three times and it is in State 1, it goes to State 0. You must use interrupt to implement the application.

Section H: Use an external hardware interrupt to enable the simulation of the traffic lights.

Connect a switch to an external hardware interrupt pin. Write a C program so that the simulation of the traffic lights in Section A can be started by pressing the switch once. If the switch is pressed again, the simulation of the traffic lights will be stopped (i.e., all LEDs are OFF).

Section I: Write a C program to keep sending and receiving characters

Write a C program to complete the following tasks by using interrupts:

1. Before you press any keys, character ‘a’ is printed continuously.

2. When you press a key (say ‘b’), 10 characters of this key (i.e., ‘b’) are printed out and then stop.

3. After that when you press a key other than the first key (i.e., ‘b’), nothing happens.

4. When you press the key again (i.e., ‘b’), character ‘a’ is printed continuously (i.e., resume).

Set the baud rate of the PC terminal (i.e., Tera Term) to 9600.

If your program runs successfully and the setting of Tera Term is correct, you should see the following output:

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaabbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacccccccccc
```

<u>Demonstrate Section B to I to our tutors or technicians.</u>

Instructions:

1. You are required to demonstrate your programs to our tutor or technicians.
2. Zip all programs (including the whole projects) from Section B to I into a single file. Submit it to Blackboard.
3. Deadline: **<u>Check the course information</u>**.

Lawrence Cheung
November 2021