The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

EIE3105 Integrated Project (Part I)

Laboratory Exercise 1: Introduction to AVR

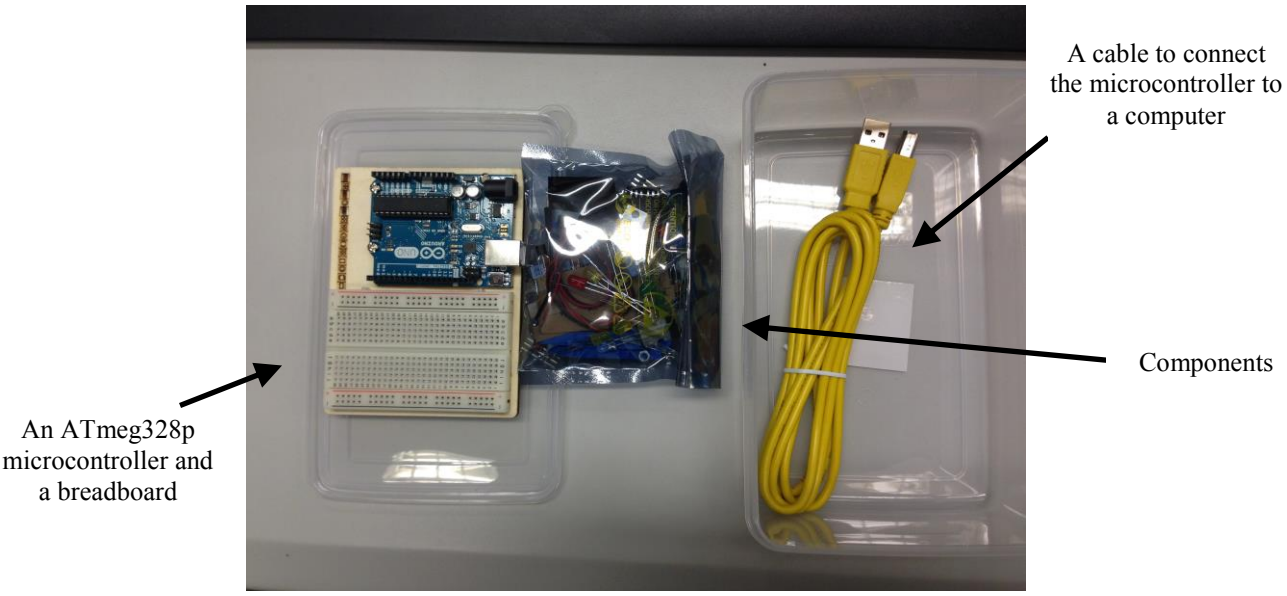
(Deadline: Check the course information)

Objective:

1. To familiarize students with the use of an AVR microcontroller.
2. To develop assembly and C programs under the Arduino platform.

Equipment:

Atmel Studio 6 (software)
The Arduino Starter Kit (hardware)



Important Notices:

1. You must read the ATmega328p datasheet carefully before you do the laboratory exercises. You have three ways to access the datasheet:
   a. Download the pdf file from Blackboard.
   b. Use a web searching engine (e.g., Google) to search the keyword "ATmega328p datasheet" and get the pdf file.
2. 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:

This experiment introduces some simple applications of using an AVR microcontroller to perform some arithmetic and I/O operations. Students are required to develop assembly and C programs and build up some simple electronic circuits for such simple applications.

Atmel Studio 6 is the integrated development platform (IDP) for developing and debugging Atmel AVR microcontroller (MCU) applications. The Atmel Studio 6 IDP gives you a seamless and easy-to-use environment to write, build and debug your applications written in C/C++ or assembly code. Atmel Studio 6 supports all 8- and 32-bit AVR MCUs. It also connects seamlessly to Atmel debuggers and development kits.

Arduino is an open-source physical computing platform based on a simple I/O board and a development environment that implements the Processing/Wiring language. Arduino can be used to develop stand-alone interactive objects or can be connected to software on your computer.

The Arduino Start Kit walks you through the basics of using the Arduino in a hands-on way. You will learn through building several creative projects. The kit includes a selection of the most common and useful electronic components.

Procedure:

Section A: Create your first simple application

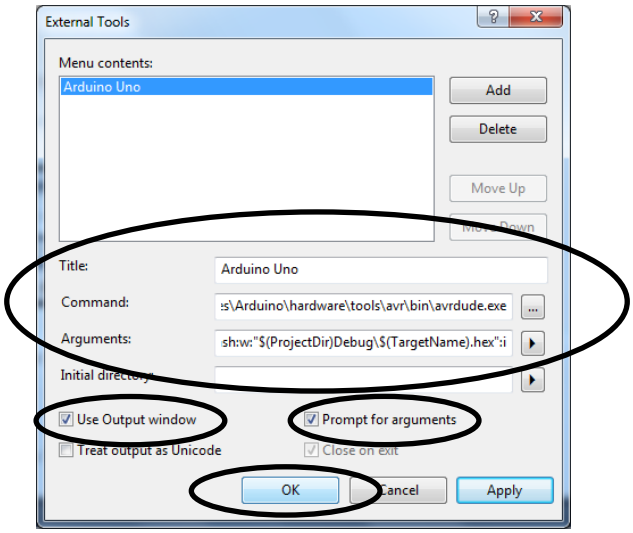
1. Open Atmel Studio: Double click Atmel Studio 6.2. 

2. Go to main menu. Select "Tools" → "External Tools". Type in "Arduino Uno" in the field "Title". Moreover, type in the following paths into the fields "Command" and "Arguments":

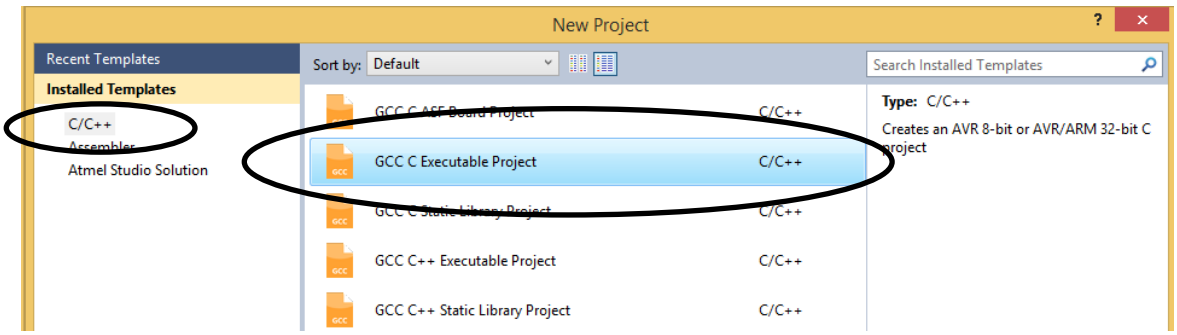

```
C:\Program Files\Arduino\hardware\tools\avr\bin\avrdude.exe

-C"C:\Program Files\Arduino\hardware\tools\avr\etc\avrdude.conf"
-v -v -v -patmega328p -carduino -P\\.\COM4 -b115200 -D -
Uflash:w:"$(ProjectDir)Debug\$(TargetName).hex":i
```

Note that "COM4" in the above path shows the port that the Arduino Uno plugs in your computer. Thus, later, when you plug in the Arduino Uno to your computer, you need to check the port is "COM4" or not (Double click the icon "Computer" and Select "Properties" → "Device Manager" → "Ports". Check the position of the port for the Arduino Uno). You may need to change the port number if it is not "COM4". Moreover, you may go to the course web site to download the text file "ArduinoUno.txt" which stores these two paths. Then you can copy and paste them into such two fields. Note that this file can also be found in "K:\Arduino". Finally, in some computers, the directory "Arduino" may not be installed in the directory "Program Files" but "Program Files (x86)". If it happens, please change the name of the directory "Program Files" in the command and the argument accordingly.

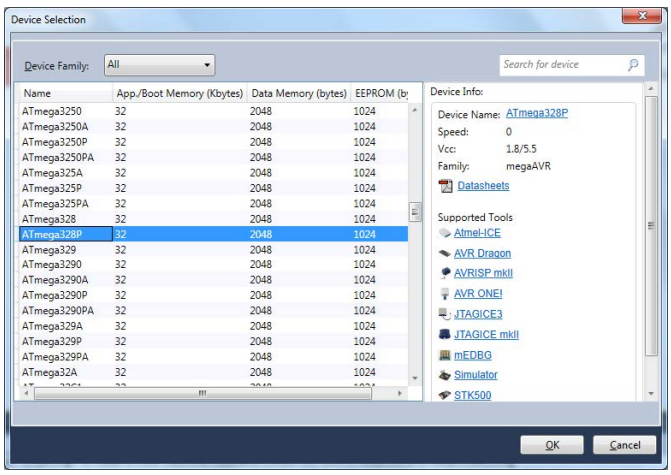
After that, click the checkboxes “Use Output window” and “Prompt for arguments” and then click the button “OK”. When you finish the setting, you will see a new field “Arduino Uno” is created under the menu item “Tools”. Note that the above setting is required to go through once it is the first time to open Atmel Studio.



3. Go to the main menu and create a new project. Then select “C/C++” and then “GCC C Executable Project” to create a new project. Type in the project name “Lab1A”. After that, select “ATmega328p”.



Note that the I/O ports of the AVR microcontroller are bit-accessible but the AVR C compiler does not support this feature. Thus, you should use a mask to set, clear and check some bits in the I/O ports.

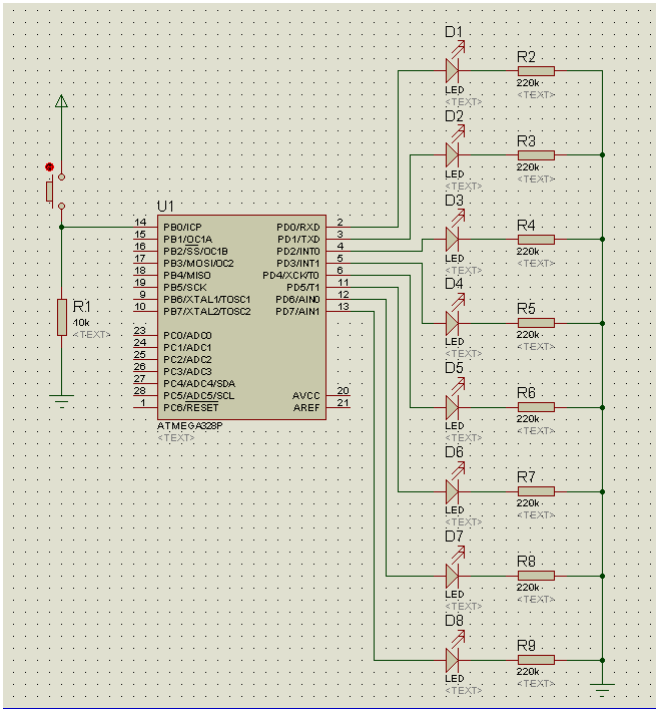


4. Type in the following program into “Lab1A.c”.

```
#include <avr/io.h>

int main(void)
{
    DDRD = 0xFF;
    while(1)
    {
            PORTD = 0x55;
    }
}
```

5. Go to the main menu. Select “Build” → “Build Solution” to compile the program.

6. Build the circuit below on the Arduino Uno and the breadboard. Refer to the pinout diagram in the appendix. Note that “220k” should be “220”. It means 220Ω.



7. Go to the main menu. Select “Tools” → “Arduino Uno” to burn the program into the Arduino Uno. You should see some LEDs are on. Note that you should remove any connections to pin 0 (RX) and pin 1 (TX) of Arduino Uno before you burn your program into it; otherwise, you may not burn your program successfully.

8. Repeat the above steps but this time Port D is toggled between 0x55 and 0xAA with 1 to 2 seconds time delay. You should use looping to generate the delay and set the optimization level to one. Show the calculation how to generate the delay. Note that the clock frequency of the Arduino Start Kit is 16 MHz. When your program works properly, you should see all LEDs connected to Port D are flashing.

Section B: Simple applications

1. Connect a switch to PB0 and the pin diagram is shown in Step 6 of Section A. Connect a LED to the switch. When the switch is pressed, the LED is on. When the switch is released, the LED is off.

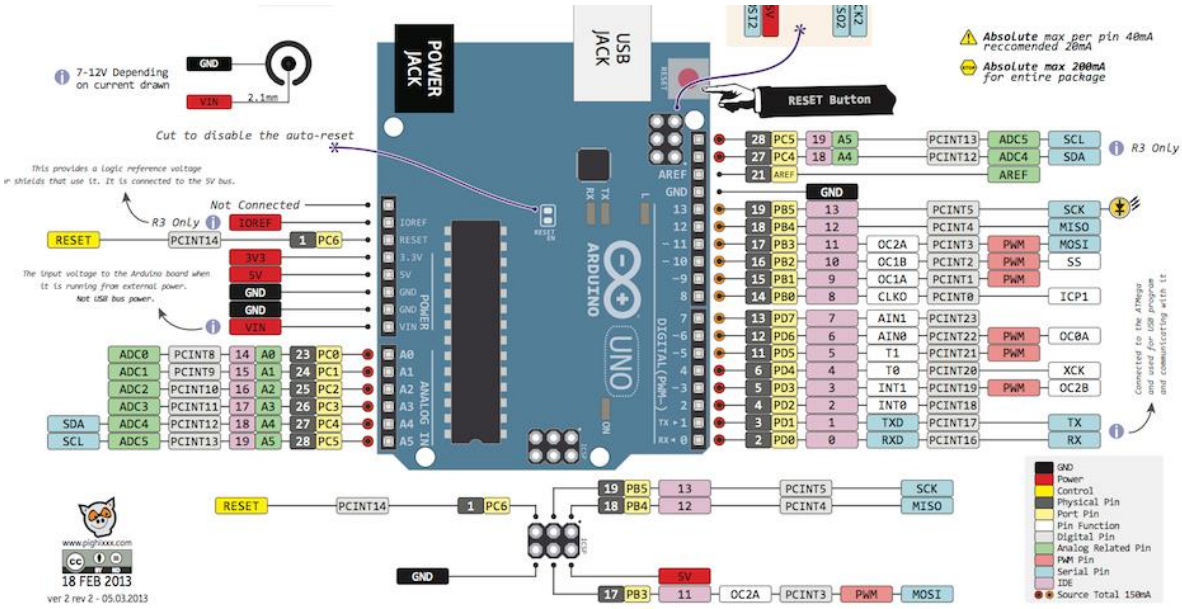
2. Use the circuit in Section A and the connection in Step 6. There are two states in the switch: State 0 and 1. When it is in State 0, all LEDs connected to Port D are off. When it is in State 1, all LEDs are on. At the beginning, the switch is in State 0. When the switch is pressed and it is in State 0, it goes to State 1. When the switch is pressed and it is in State 1, it goes to State 0.

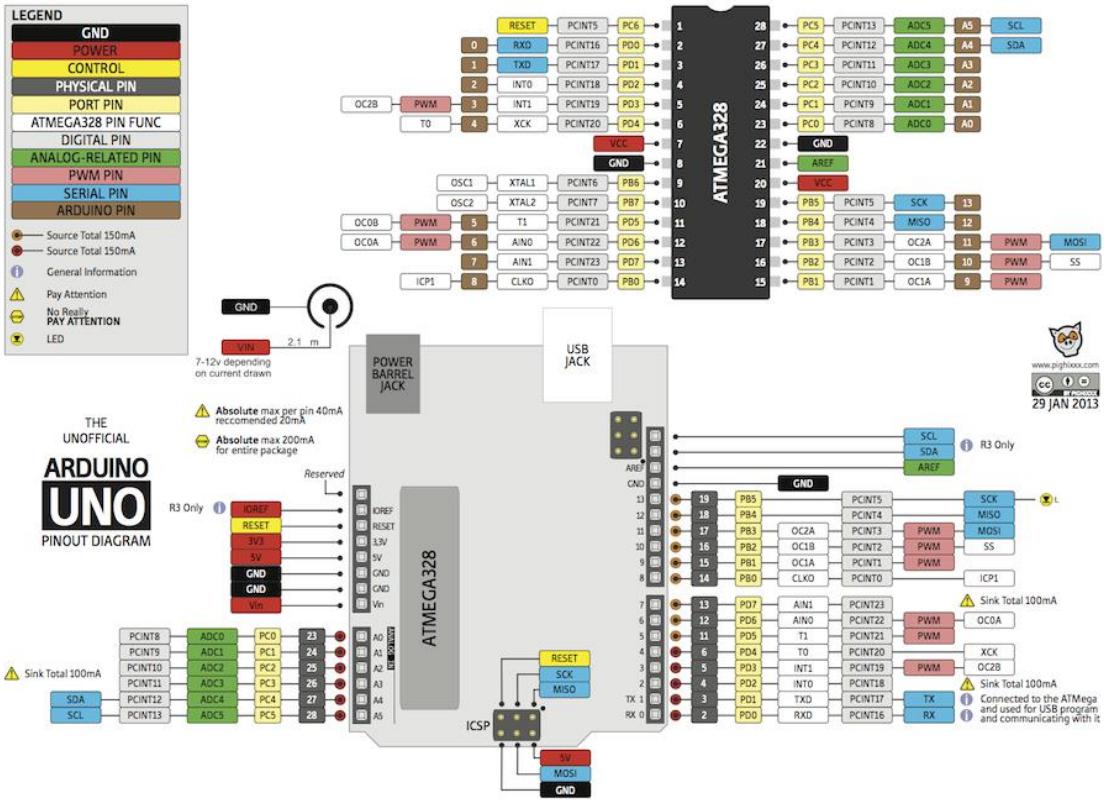
<u>Demonstrate Section A (Step 7 and 8) and B (Step 1 and 2) to our tutors or technicians.</u>

Instructions:

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

Appendix: Arduino Uno Pinout Diagrams





Ivan Lau
Lawrence Cheung
August 2017

The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

EIE3105 Integrated Project (Part I)

Laboratory Exercise 2: AVR Timer/Counter Programming

(Deadline: Check the course information)

Objective:

1. To develop C programs with timers under the Arduino platform.
2. To develop a C program with counters under the Arduino platform.

Introduction:

This experiment introduces an application of using timers and counters inside the AVR microcontroller. ***Students MUST use the AVR timers in the delay function to finish this lab. Using simple loops to count the delay is not acceptable.***

Equipment:

Atmel Studio 6
The Arduino Starter Kit

Procedure:

Section A: Write a C Program to toggle a LED by using a timer

1. Write a C program to toggle a LED connected to PB0 (you should connect the LED to PB0 with a resistor) in every second. You should use the timer with Normal mode to generate the time delay of one second. Note that the clock frequency of the Arduino Start Kit is 16 MHz.

2. Repeat Step 1 but this time the timer is in CTC mode.

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

1. Write a C program to simulate the traffic lights by using different pins. You should use a timer in Normal mode. Note that the clock frequency of the Arduino Start Kit is 16 MHz.

   A set of traffic lights for cars (Light 3): PB0, PB1, PB2 (3 LEDs)
   A set of traffic lights for cars (Light 2): PB3, PB4, PB5 (3 LEDs)
   A set of traffic lights for people (Light 1): PC4, PC5 (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)

2. Repeat Step 1 but this time the timer is in CTC mode.

Section C: Write a C program to count a switch

Connect a switch to pin T0 (PD4, Counter 0) or T1 (PD5, Counter 1) and a LED to PC0. 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 should use counter programming in CTC mode to implement this application.

Demonstrate Section B and C to our tutors or technicians.

Instructions:
1. You are required to demonstrate your programs to our tutor or technicians.
2. Zip all programs (including the whole projects) in Section A, B, and C into a single file, and submit it to Blackboard.
3. Deadline: **Check the course information**.

Ivan Lau
Lawrence Cheung
August 2017

The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

EIE3105 Integrated Project (Part I)

Laboratory Exercise 3: AVR Interrupt Programming

(Deadline: Check the course information)

Objective: To develop C programs with interrupts under the Arduino platform.

Equipment: Atmel Studio 6 and the Arduino Starter Kit

Procedure:

Section A: Write a single C program to implement two applications in Lab 2.

Implement Section B and C of Lab 2 in the Arduino Uno so that two applications (traffic light and counting) can be executed at the same time. Note that CTC mode should be used. Moreover, you must use interrupts to implement these two applications (i.e., timer and counter programming). The clock frequency of the Arduino Start Kit is 16 MHz. Note that you should write a simple application (e.g., flash one LED only) to check whether you can setup a timer properly by using an interrupt.

Section B: Replace the counting application by using an external hardware interrupt.

In Section A, counter programming is used to implement the application. In this section, an external hardware interrupt INT0 is used as a counter to implement the same application.

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

Connect a switch to an external hardware interrupt INT1 pin. Write a C program so that the simulation of the traffic lights 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).

Demonstrate your applications in Section A, B and C to our tutors or technicians.

Instructions:

1. You are required to demonstrate your programs to our tutor or technicians.
2. Zip all programs (including the whole projects) in Section A, B and C to a single file. Submit it to Blackboard.
3. Deadline: **Check the course information**.

Ivan Lau
Lawrence Cheung
August 2017

The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

EIE3105 Integrated Project (Part I)

Laboratory Exercise 4: AVR Serial Port Programming

(Deadline: Check the course information)

Objective:

To develop C programs with serial port communication under the Arduino platform.

Introduction:

This experiment introduces an application of using the serial port communication inside the AVR microcontroller.

Equipment:

Atmel Studio 6
The Arduino Starter Kit

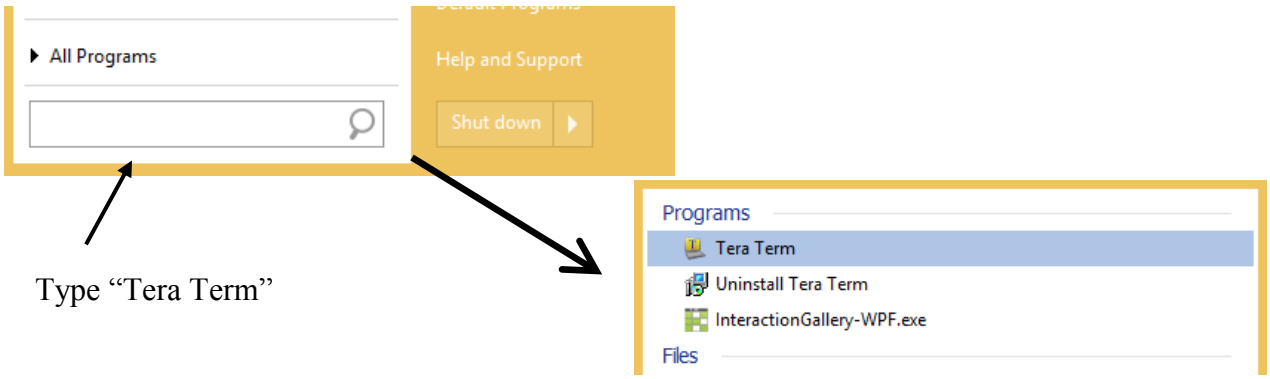
Procedure:

Section A: Write a C program to echo a character

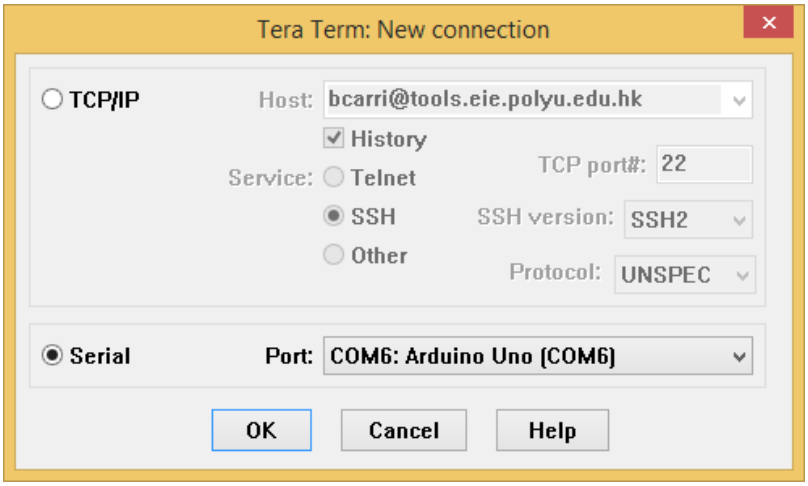
Write a C program to complete the following task:

Receive a character from the serial port by using the polling method and then send it to a PC terminal through the serial port. The baud rate should be 4800.

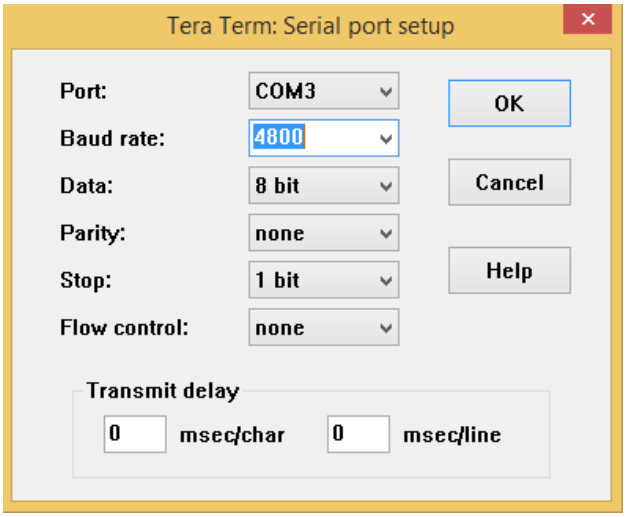
To view the serial port communication, you should use the freeware "Tera Term", which is a PC terminal. You can use the search function to locate this software in your computer.



Select “Serial” and then an appropriate port for the serial port communication. The port number should be the same as the port that the Arduino connects to your computer.



Select “Setup” from the main menu and then “Serial port”. Set the baud rate to 4800.



Note that you must close Tera Term before you burn your program into Arduino; otherwise, Tera Term holds the serial port and you cannot burn your program successfully.

A character can be sent to the Arduino by typing it (i.e., using the keyboard). When you type a character, Tera Term gets the input and sends it to the Arduino through the serial port. When the Arduino gets a character, it will send the character to the serial port and you can read it through Tera Term.

Section B: Write a C program to send and receive strings

Write a C program to complete the following tasks:

1. (Do it once at the beginning) Send “We Are Ready” to the serial port by using the polling method. Note that the clock frequency of the Arduino Start Kit is 16 MHz.

2. (Do it repeatedly) Receive data from the serial port by using the polling method. If the received string is “Hi”, your program should send “Bye” to the serial port. (Hint: To get a string from the serial port, your program should receive it character by character. For example, if your program receives a string “abc”, the characters received from the serial port should be “a” first, and then “b”, and finally “c”.)

The string “Hi” can be sent to the Arduino by typing it (i.e., using the keyboard). When you type “Hi”, Tera Term gets the input and sends it to the Arduino through the serial port. When the Arduino gets the string, it will send a message “Bye!” to the serial port and you can read it through Tera Term (see the figure below):

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

Write a C program to complete the following tasks using the polling method:

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

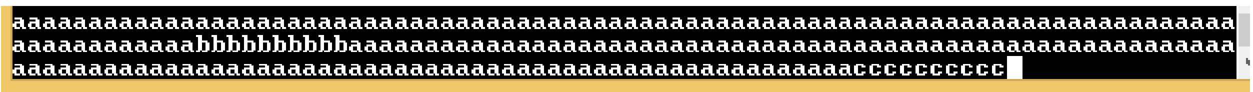
2. When you press a key (say ‘b’), 10 characters of this key (i.e., ‘b’) are printed out and then stop.

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:



Section D: Write C programs by using the serial port interrupt

Repeat Section B and C but this time you use the serial port interrupt (not the polling method).

Demonstrate your applications in Section B, C and D to our tutors or technicians.

Instructions:

1. You are required to demonstrate your programs to our tutor or technicians.
2. Zip all programs (including the whole projects) in Section A to D to a single file. Submit it to Blackboard.
3. Deadline: **Check the course information**.

Ivan Lau
Lawrence Cheung
August 2017

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)

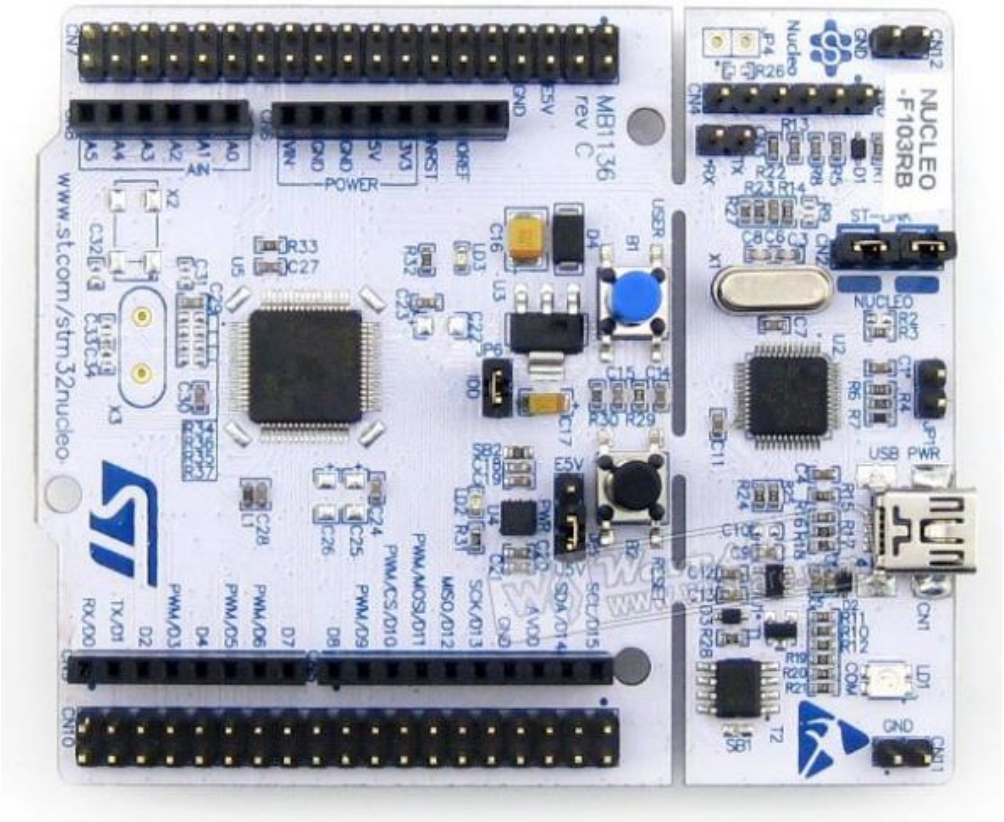
Objective:

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. The following web site may be useful for you to know ARM program codes: **http://www.longlandclan.yi.org/~stuartl/stm32f10x_stdperiph_lib_um/**
4. 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:

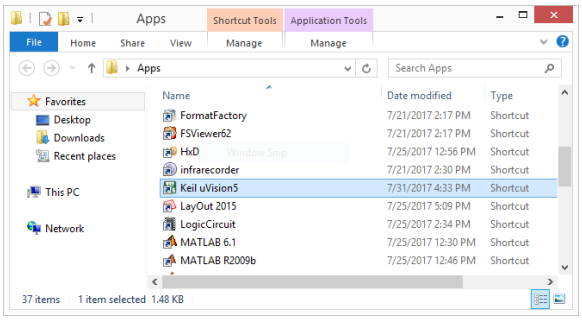
The STM32F103RB medium-density performance line family incorporates the high-performance ARM®Cortex®-M3 32-bit RISC core operating at a 72 MHz frequency, high-speed embedded memories (Flash memory up to 128 Kbytes and SRAM up to 20 Kbytes), and an extensive range of enhanced I/Os and peripherals connected to two APB buses. TheSTM32F103RB 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 provides an introduction to STM32F103RB using Keil and C language. Students will be familiar with the basic I/Os and operate the on-board components.

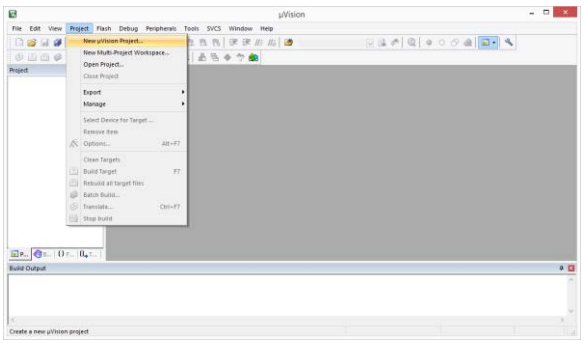
Procedure:

Section A: Create a Keil project.

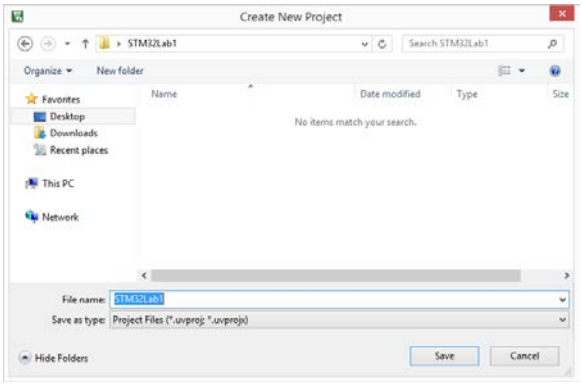
1. Open the **App** folder on Desktop and open **Keil uVision 5**.



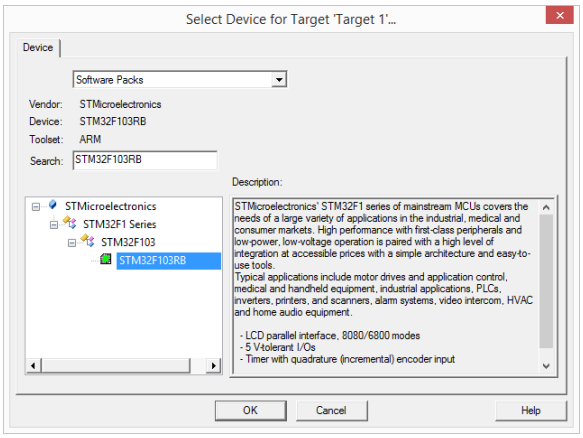
2. **Project -> New uVision Project**



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**.



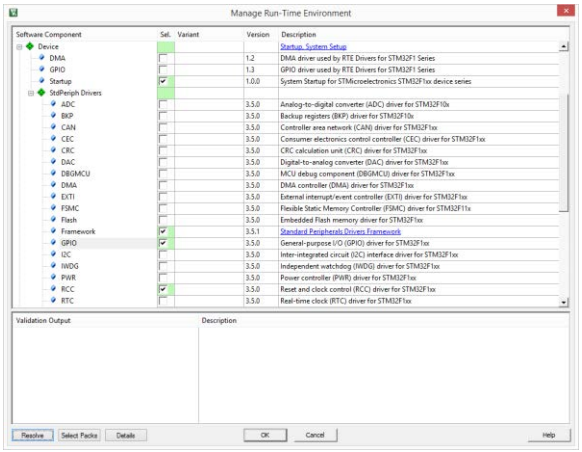
4. Select Device by searching "**STM32F103RB**". Choose **STM32F103RB** and click **OK**.



5. Manage Run-Time Environment. Check **Device** -> **Startup** and **Device** -> **StdPeriphDrivers** -> **GPIO**.

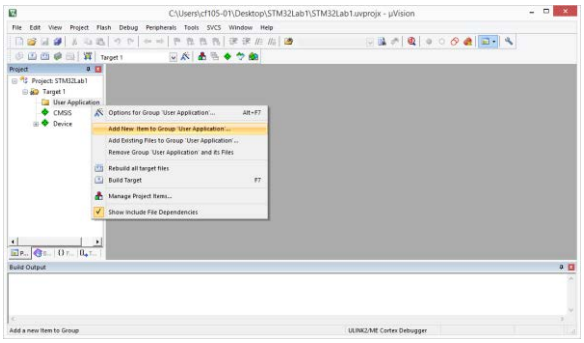
   Click **Resolve** at the lower left corner to include all the necessary libraries for the drivers you selected.

   Validation Output should be **empty** and click OK.

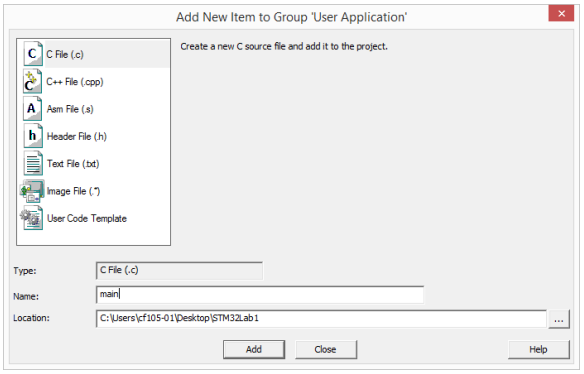


6. Organize project explorer Rename **Source Group 1** to **User Application**.

   Right click on **User Application** -> **Add New item to Group "User Application"**.

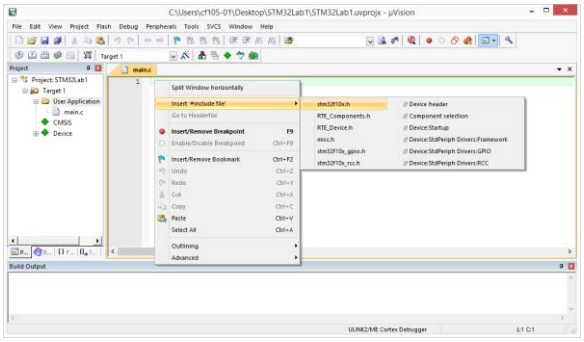


7. Choose **C File(.c)** and name it **main**. Then click **Add**.

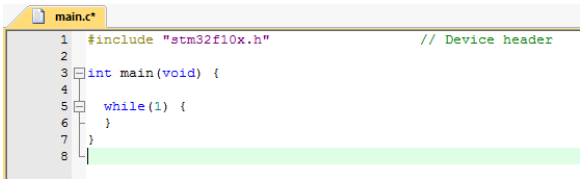


8. Open **main.c**.
Right click the first line -> **Insert "#include file"** -> **stm32f10x.h**

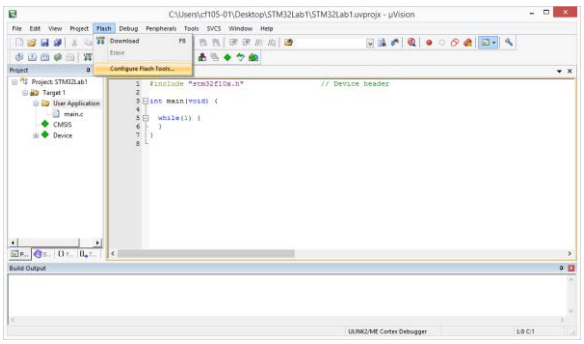
Stm32f10x.h defines all the registers and constants for the system.



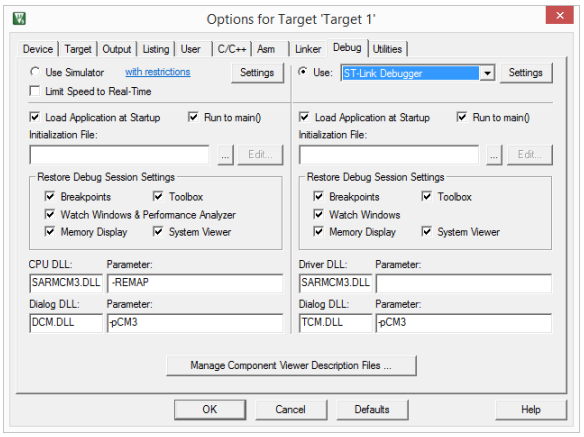
9. Create a main function



10. **Flash** -> **Configure Flash Tools**

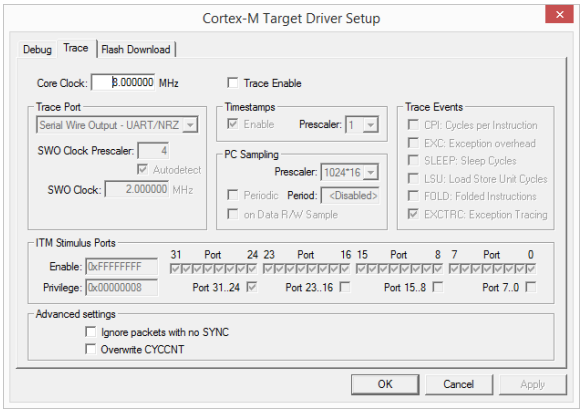


11. **Debug** -> **choose ST-Link Debugger**

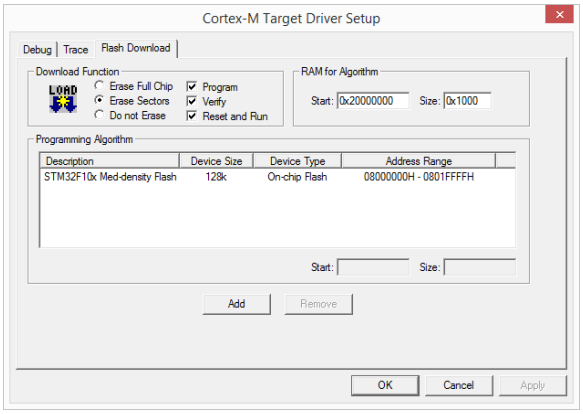


12. On the same **Debug** tab -> **Settings** -> **Trace**

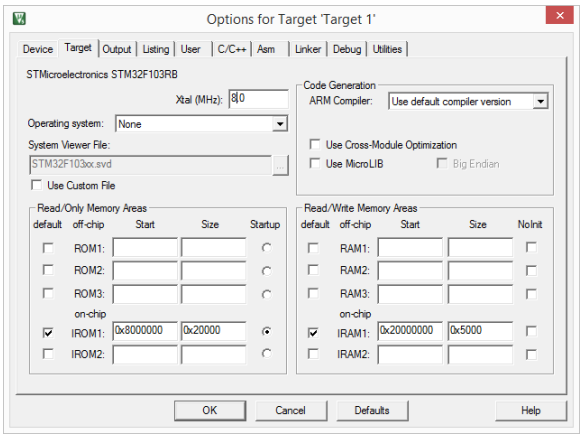
Change Core Clock to **8** MHz.



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



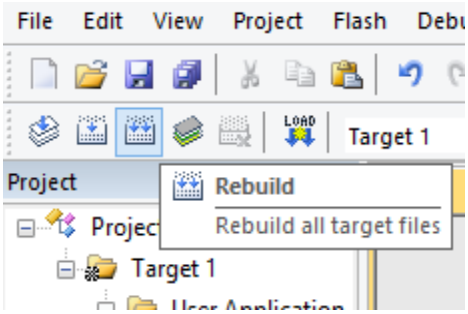
14. Go to **Target** tab. Change the crystal frequency to **8** MHz.
Then click **OK**.



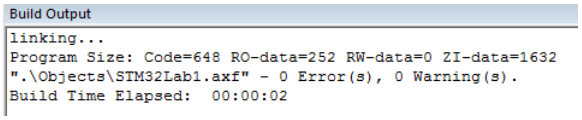
15. Type in the following statements in Line 4.

```
//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;
```

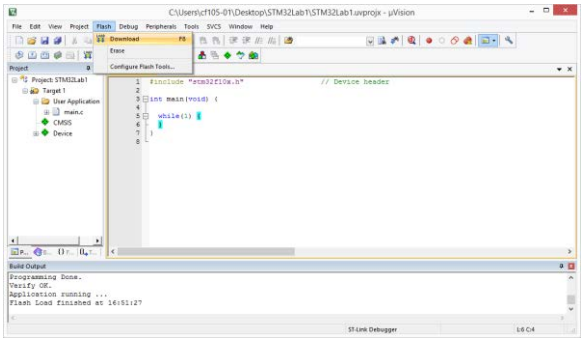
16. Click the **Rebuild** button to rebuild the project.
OR
Project -> **Rebuild all target files**



17. Fix any errors and warnings.

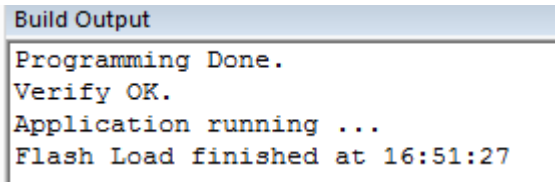


18. Download the program to the ARM microcontroller.
Flash -> **Download**



19. Build Output for successful download.

Of course, you will not see anything happen on the board for an empty main function.



Section B: Flash the on-board LED (PD2) by using a delay loop.

Flash the on-board LED (LD2) 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, j;
  for(i=0; i<t; i++)
    j++;
}
```

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.

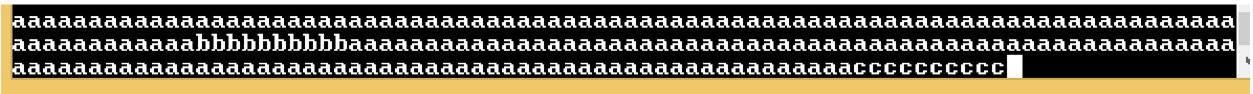
2. When you press a key (say 'b'), 10 characters of this key (i.e., 'b') are printed out and then stop.

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:



<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>**.

Ivan Lau
Lawrence Cheung
August 2018