Lab 3 – Programming GPIOs - Simulation

In this exercise you will write a program to read and control GPIO ports and pins using the Keil simulator (based on Nucleo-F103RB board). The objective is to understand how the GPIO device connects with the microprocessor and how to manipulate GPIO pins to serve our purpose.

Useful sources (QMplus)

- GPIOonKeil project
- Week 3 Tutorial exercise
- Week 3 Lectures on GPIO and Timers

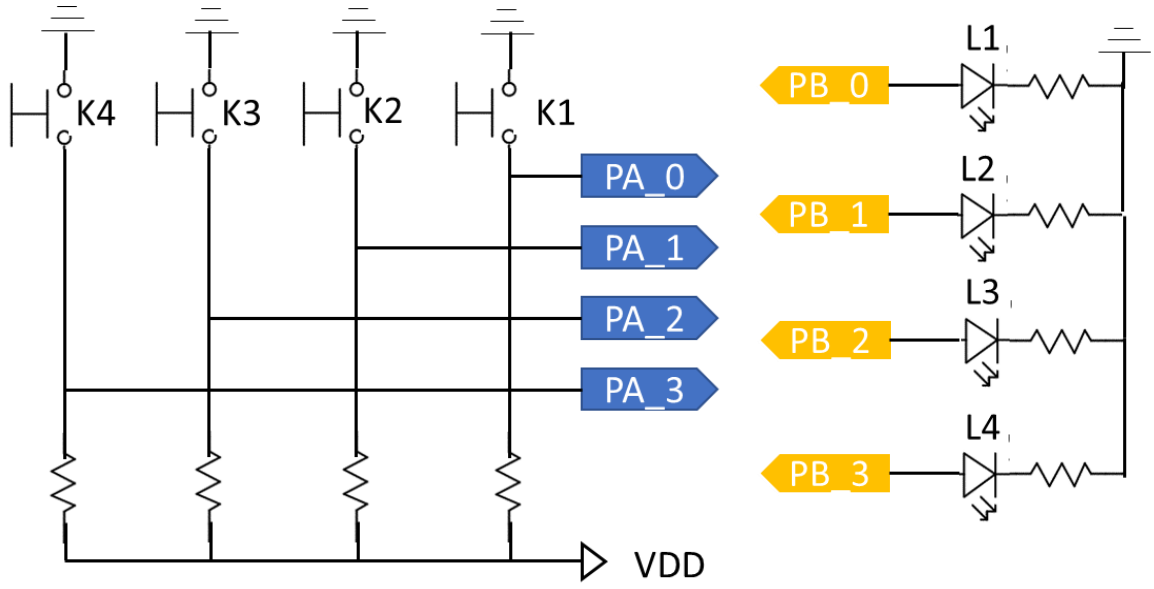
Circuit: Buttons and LEDs

GPIO pins Input/Output

In this project you are supposed to have FOUR input pins, each connected to a button and FOUR output pins, each connected to a LED, as in the table below:

Buttons		LEDs	
K1	PA_0	L1	PB_0
K2	PA_1	L2	PB_1
K3	PA_2	L3	PB_2
K4	PA_3	L4	PB_3

Please note that button shorts the pin to ground when pressed while LEDs are ON when port pin reads logic 1. Always double check the circuit schematic for actual electronic connections before you start writing your program:



Tasks

Your task is to write an ARM C program to read the buttons and control the LEDs as follows:

- L1 is turned ON when K1 is pressed; OFF otherwise;
- L2 is toggled (ON to OFF or OFF to ON) when K2 is pressed and then released;
- L3 is toggled X times when K3 is pressed;
- L4 is toggled Y times when K4 is pressed;

where X and Y are your last two digits of your QMUL student ID incremented by 1. For instance, if your student ID is 141054182, X = 9 and Y = 3.

Please note that you may ignore K3 when L3 is toggling and K4 when L4 is toggling. However, your program still has to work with K1 and K2 when some other LEDs is/are toggling.

Since it is not currently possible to access an actual ARM-based board, you can simulate the circuit and monitor the GPIO peripheral to test the functionality.

Procedure

Follow the steps below and answer the questions according.

Write the C code to configure the GPIO pins and related functionalities. Note: the reset value of the switches and corresponding pins should reflect none-pressed switches.

Test the behaviour of the buttons (running the code at full speed) and make sure the requirements are satisfied by following these steps in task 1 below.

Completed Tasks

Task1- Functionality tests

Follow the test procedure below and note down the values requested in the lab sheet.

1. Compile the code.
2. Start a debugger session.
3. Add a breakpoint (BP1) at the first instruction in the infinite loop.
4. Run the program until the opening brace in the main function is highlighted. Open the visualisation windows for both GPIOA And GPIOB from the menu Peripherals-> General Purpose I/O.
5. Place the cursor at BP1 and run to cursor line (Ctrl+F10). Note down the value of GPIOA_IDR (pins 3 to 0) in Table 1 Test 1. (if pin_0=1, pin_1=1, pin_2=0, pin_3=1, write 1011).

- GPIOA_IDR pins 3 to 0:

6. Run to cursor line Ctrl+F10. Note down the value of GPIOB_ODR_0:

- GPIOB_ODR_0:

7. Uncheck GPIOA_pin_0 and run to cursor line Ctrl+F10. Note down the value of GPIOB_ODR_0:

- GPIOB_ODR_0:

8. Uncheck GPIOA_pin_1 (button pressed) and run to cursor line Ctrl+F10. Note down the value

of GPIOB_ODR_1:

- GPIOB_ODR_1:

9. Check GPIOA_pin_1 (button released) and run to cursor line Ctrl+F10. Note down the value of GPIOB_ODR_1:

- GPIOB_ODR_1:

10. Repeat steps 8 and 9 and note down the values of GPIOB_ODR_1:

- First value of GPIOB_ODR_1:
- Second value of GPIOB_ODR_1:

11. Repeat steps 8 and 9 and note down the values of GPIOB_ODR_1:

- First value of GPIOB_ODR_1:
- Second value of GPIOB_ODR_1:

12. Check GPIOA_pin_3, GPIOA_pin_1, and GPIOA_0 and keep GPIOA_pin_2 unchecked then run to cursor line Ctrl+F10. Then, check GPIOA_pin_2 and run to cursor line Ctrl+F10 X times. Note down the value of X and all (X+1) values of GPIOB_ODR_2 separated by ";":

- X:
- GPIOB_ODR_2:

PLEASE MAKE SURE YOU INITIALISE THE GPIO_ODR TO ALL LEDS OFF BEFORE EACH OF THE FOLLOWING STEPS.

13. Check GPIOA_pin_2, GPIOA_pin_1, and GPIOA_0 and keep GPIOA_pin_3 unchecked then run to cursor line Ctrl+F10. Then, uncheck GPIOA_pin_0 and run to cursor line Ctrl+F10 Y times. Note down the value of Y and all (Y+1) values of GPIOB_ODR pins 3 to 0 separated by ";". (if pin_0=1, pin_1=1, pin_2=0, pin_3=1, write 1011).

- Y:
- GPIOB_ODR pins 3 to 0:

14. Check GPIOA_pin_2, GPIOA_pin_1, and GPIOA_0 and keep GPIOA_pin_3 unchecked then run to cursor line Ctrl+F10. Then, uncheck GPIOA_pin_1 and run to cursor line Ctrl+F10. Then, check GPIOA_pin_1 and run to cursor line Ctrl+F10 Y-1 times. Note down the value of Y and all (Y+1) values of GPIOB_ODR pins 3 to 0 separated by ";". (if pin_0=1, pin_1=1, pin_2=0, pin_3=1, write 1011).

- Y:
- GPIOB_ODR pins 3 to 0:

15. Check GPIOA_pin_2, GPIOA_pin_1, and GPIOA_0 and keep GPIOA_pin_3 unchecked then run to cursor line Ctrl+F10. Then, uncheck GPIOA_pin_2 and run to cursor line Ctrl+F10 max[(Y+1),(X+1)] times. Note down the value of X, Y and all values of GPIOB_ODR pins 3 to 0 separated by ";". (if pin_0=1, pin_1=1, pin_2=0, pin_3=1, write 1011).

- X:
- Y:
- GPIOB_ODR pins 3 to 0:

Task2- Main C Code

Your main code with annotations.

Task3- Research

Have you heard of the problem of button bouncing? It is not possible to demonstrate it without hardware, but it is a real problem related to the non-ideal functionality of switches and buttons. Describe the problem and TWO possible solutions based on your research.

- End of lab 2 -