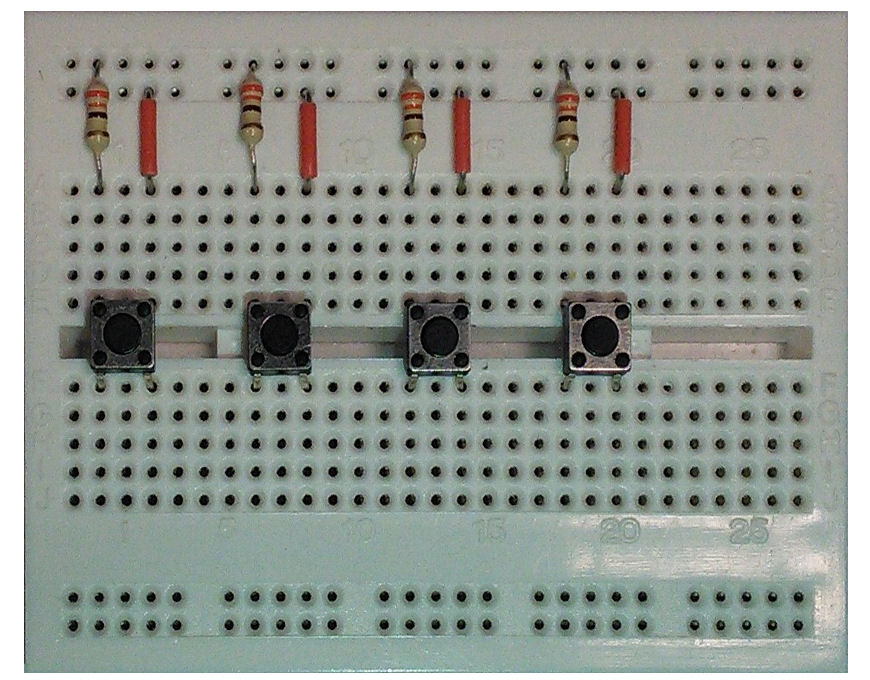
Lab Exercise:   
programming using mbed api

# Overview

In this lab, you will learn how to use the mbed SDK library to program two applications, the first replicates the previous module’s lab exercise using the mbed API, this time, and the second writes an interrupt handler with the mbed API. Wire up your solderless breadboard as shown:



Vcc

Buttons 1-4

PA\_10

PB\_3

PB\_5

PB\_4

**To this drawing also add an LED with cathode to ground and anode to port PA9 through an appropriate resistor or combination of resistors. Note that the ports can be connected to the board via the Arduino connector**

* Exercise 1: Interface the GPIOs using digital IO functions in the mbed API
* Exercise 2: Implement the interrupt handler using the interrupt libraries in the mbed API

# implementation details

## Software functions

### Digital in/out functions

The DigitalIn interface is used to read the value of a digital input pin.

For hardware setup, refer to the previous lab notes (“Interrupt and Low Power Features” or “Digital Input/Output and GPIO”).

For example, to read the value of a pin:

DigitalIn button\_press(*Input Pin*);

int main(){

if (button\_press)

Led\_out = ~Led\_out;

}

The DigitalOut interface is used to configure and control a digital output pin, for example

DigitalOut Led\_out(*Output Pin*);

int main() {

Led\_out = 1;

}

### bus in/out functions

The BusIn interface is used to create a number of DigitalIn pins that can be read as one value.

Any of the numbered mbed pins can be used as a DigitalIn in the BusIn. For example:

BusIn My\_Bus\_In(*Input Pin 1*, *Input Pin 2*, *Input Pin 3*);

int main(){

if(My\_Bus\_In == 0x01)

Led\_out = 1;

}

The BusOut interface is used to create a number of DigitalOut pins that can be written as one value, for example:

BusOut My\_Bus\_Out(*Output Pin 1*, *Output Pin 2*, *Output Pin 3*);

int main(){

My\_Bus\_Out = (1<<2) | (1<<1);

}

### low power consumption

Use the sleep mode to reduce the power consumption of your application, for example, to use the Sleep-on-Exit feature you can use the wait-for-interrupt operation:

\_\_wfi() ; //go to sleep

### interrupt in/out functions

The InterruptIn interface is used to trigger an event when a digital input pin changes. For example:

InterruptIn button\_press(*Input Pin*);

void button\_ISR(){

Led\_out = !Led\_out;

}

int main(){

button\_press.rise(&button\_ISR);

while(1); // waiting for interrupts

}

Other functions are listed below:

|  |  |
| --- | --- |
| **Function name** | **Description** |
| InterruptIn (PinName pin) | Create an InterruptIn connected to the specified pin |
| void rise (void(\*fptr)(void)) | Attach a function to call when a rising edge occurs on the input |
| template<typename T>  void rise (T \*tptr, void(T::\*mptr)(void)) | Attach a member function to call when a rising edge occurs on the input |
| void fall (void(\*fptr)(void)) | Attach a function to call when a falling edge occurs on the input |
| template<typename T >  void fall (T \*tptr, void(T::\*mptr)(void)) | Attach a member function to call when a falling edge occurs on the input |
| void mode (PinMode pull) | Set the input pin mode |
| void enable\_irq () | Enable IRQ |
| void disable\_irq () | Disable IRQ |

## your Application Code

### exercise-1

In the first exercise, you need to use mbed API functions to:

* Define BusIn, BusOut interfaces for inputs and outputs
* ~~The RGB LED is controlled by buttons~~
  + ~~Button 1: light up RED~~
  + ~~Button 2: light up GREEN~~
  + ~~Button 3: light up BLUE~~
  + ~~Button 4: light up WHITE (RED, GREEN and BLUE)~~

NOTE: There is no RGB LED on this board. Instead of what is listed above, let Button 1 turn LED LD2 on, Button 2 turns it off, Button 3 turns on an external LED in you wireless breadboard, and Button 4 turns the external LED off.

### exercise-2

In the second exercise, you need to use the mbed API functions to:

* Define InterruptIn and ISR for each bit from input
* In the interrupt service routine, the ~~RGB~~ LD2 and External LEDs are used to indicate when a button is pressed.
  + ~~Button 1: light up RED~~
  + ~~Button 2: light up GREEN~~
  + ~~Button 3: light up BLUE~~
  + ~~Button 4: light up WHITE (RED, GREEN and BLUE)~~
* NOTE: There is no RGB LED on this board. Instead of what is listed above, let Button 1 turn LED LD2 on, Button 2 turns it off, Button 3 turns on an external LED in you wireless breadboard, and Button 4 turns the external LED off.
* Put the processor into sleep mode upon exiting from the ISR

Try to issue an interrupt on different signal edges (rising edge or falling edge). What changes?

Draw a schematic of your final circuit (nothing fancy, can be done by hand and scanned).