```
/*
* CMPE244 Lab Assignment 1
* Hardware LED Switch
*
* > Create 2 RTOS tasks to read internal and external switch inputs.
* > Create 2 RTOS tasks to control internal and external LEDs.
* > LED state should toggle upon switch release.
*
* File     : main.cpp
* Author   : Harmeen Joshi
* Date     : 09/04/2018
*/
#include <tasks.hpp>
#include "LPC17xx.h"
#include "LAB_GPIO1.hpp"

volatile bool LED_Flag;
volatile bool Switch_Flag = true;

struct pin_info internal_switch;
struct pin_info internal_LED;
struct pin_info external_switch;
struct pin_info external_LED;

/* Class constructor */
LabGPIO_1::LabGPIO_1(struct pin_info *ptr)
{
   pin_selected = ptr;
}
```

```
void LabGPIO_1::setAsInput(void)
{
   pin_selected->LPC_GPIO->FIODIR &= ~(1 << (pin_selected->pin_number));
}

void LabGPIO_1::setAsOutput(void)
{
   pin_selected->LPC_GPIO->FIODIR |= (1 << (pin_selected->pin_number));
}

void LabGPIO_1::setDirection(bool output)
{
   output ? setAsOutput() : setAsInput();
}

void LabGPIO_1::setHigh(void)
{
   pin_selected->LPC_GPIO->FIOPIN |= 1 << (pin_selected->pin_number);
}

void LabGPIO_1::setLow(void)
{
   pin_selected->LPC_GPIO->FIOPIN &= ~(1 << (pin_selected->pin_number));
}

void LabGPIO_1::set(bool high)
{
   high ? setHigh() : setLow();
```

```
}

void LabGPIO_1::toggle(void)
{
   pin_selected->LPC_GPIO->FIOPIN ^= (1 << (pin_selected->pin_number));
}

bool LabGPIO_1::getLevel(void)
{
   uint32_t masked_pin = (pin_selected->LPC_GPIO->FIOPIN
                        & (1 << (pin_selected->pin_number)));
   return (0 != masked_pin);
}

void vReadSwitch(void *pin_info_ptr)
{
   /* Create an object of the given GPIO pin */
   LabGPIO_1 my_gpio((struct pin_info*)pin_info_ptr);

   my_gpio.setAsInput();

   while(1) {
      while(!Switch_Flag);

      LED_Flag = false;   // Reset LED_Flag indicating "Do not set LED".
      vTaskDelay(10);

      if (!my_gpio.getLevel()){          // Active-HIGH switch.
         while (!my_gpio.getLevel());    // Wait till switch not pressed.
```

```
      } else {
         while (my_gpio.getLevel());     // Wait till switch not released.
         LED_Flag = true;              // Set flag to toggle LED
         vTaskDelay(10);
      }
   }
}

void vControlLED(void *pin_info_ptr)
{
   /* Create an object of the given GPIO pin */
   LabGPIO_1 my_gpio((struct pin_info*)pin_info_ptr);

   my_gpio.setAsOutput();

   while(1) {
      while(!LED_Flag);

      Switch_Flag = false;    /* Reset Switch_Flag */
      vTaskDelay(10);

      my_gpio.toggle();      /* Toggle LED */

      Switch_Flag = true;     /* Set Switch_Flag */
      vTaskDelay(10);
   }
}

int main(void)
```

```
{
    /* Set internal_switch to PORT1-PIN9 */
    internal_switch.LPC_GPIO = LPC_GPIO1;
    internal_switch.pin_number = 9;

    /* Set internal_LED to PORT1-PIN0 */
    internal_LED.LPC_GPIO = LPC_GPIO1;
    internal_LED.pin_number = 0;

    /* Set external_switch to PORT0-PIN1 */
    external_switch.LPC_GPIO = LPC_GPIO0;
    external_switch.pin_number = 1;

    /* Set external_LED to PORT0-PIN0 */
    external_LED.LPC_GPIO = LPC_GPIO0;
    external_LED.pin_number = 0;

    /* Create RTOS task to read internal switch */
    xTaskCreate(vReadSwitch,
            "read_internal",
            STACK_SIZE,
            (void *) &internal_switch,
            PRIORITY_HIGH,
            NULL);

    /* Create RTOS task to read external switch */
    xTaskCreate(vReadSwitch,
            "read_external",
            STACK_SIZE,
```

```
            (void *) &external_switch,
            PRIORITY_HIGH,
            NULL);


    /* Create RTOS task to control internal LED */
    xTaskCreate(vControlLED,
            "control_internal",
            STACK_SIZE,
            (void *) &internal_LED,
            PRIORITY_HIGH,
            NULL);

    /* Create RTOS task to control external LED */
    xTaskCreate(vControlLED,
            "control_external",
            STACK_SIZE,
            (void *) &external_LED,
            PRIORITY_HIGH,
            NULL);

    vTaskStartScheduler();

    return 0;
}
```