CS466 Lab 2 -- Hardware, Development Tools and Blinking the LED via RTOS
Due by Midnight Thursday 2-9-2017.

!!! Must use provided lab format on Blackboard !!!

Note: This is an individual lab, you are free to collaborate but every student must perform the lab and hand in a lab report. It will be critical that each student is able to develop to the target platform..

Overview:

- This lab is an enhancement of lab 1 requirements using a multithreaded implementation.

Lab result requirements:

- Modify the starting code so that it meets the requirements in steps 6 and 8:

Lab Preperation:

- Review your Lab1 problem and solution.. Does it meet all the requirements in step 6?
- Read the lab document CS466 Lab 2 Information.pdf, it supply's some information required to get FreeRTOS up and running
- Look at the provided blinkyRtos.c file and familiarize yourself with all the code in the file.
- Locate the FreeRTOS API documentation at http://www.freertos.org/a00106.html and read about the xSemaphoreTake() and xSemaphoreGiveFromISR() FreeRTOS functions in detail.
- Take a hard look at the blinkyRtos.c program. There are a lot of new concepts in it and it will be confusing at first.
    - Several driver library functions are used to simplify GPIO setup
    - Several #define macros are new that make the program read a little better.
    - There is an interrupt handler in place to detect the SW1 press. Study
        - What the interrupt handler does
        - How it's enabled

Objective:

- To discover that the RTOS helps organize and make your code more independent and flexible.
- To get a first introduction with multithreaded programs and communications with an asynchronous ISR.

Lab Work

1. □ Perform a 'git pull' on the class repo to get the required lab02 directories.
2. □ Download FreeRTOS Version 9.0.0 from https://sourceforge.net/projects/freertos/ unzip so that the FreeRTOS directory is a peer to your TivaDriver directory.

```
$ cd ~/src/cs466_s17
```

```
        $ unzip ~/Downloads/FreeRTOSv90.0.0.zip
```

~~3.~~ □ This Lab also starts to use the TI library code for the Tiva board. Before you can use it you will need to compile the TivaDriver library.

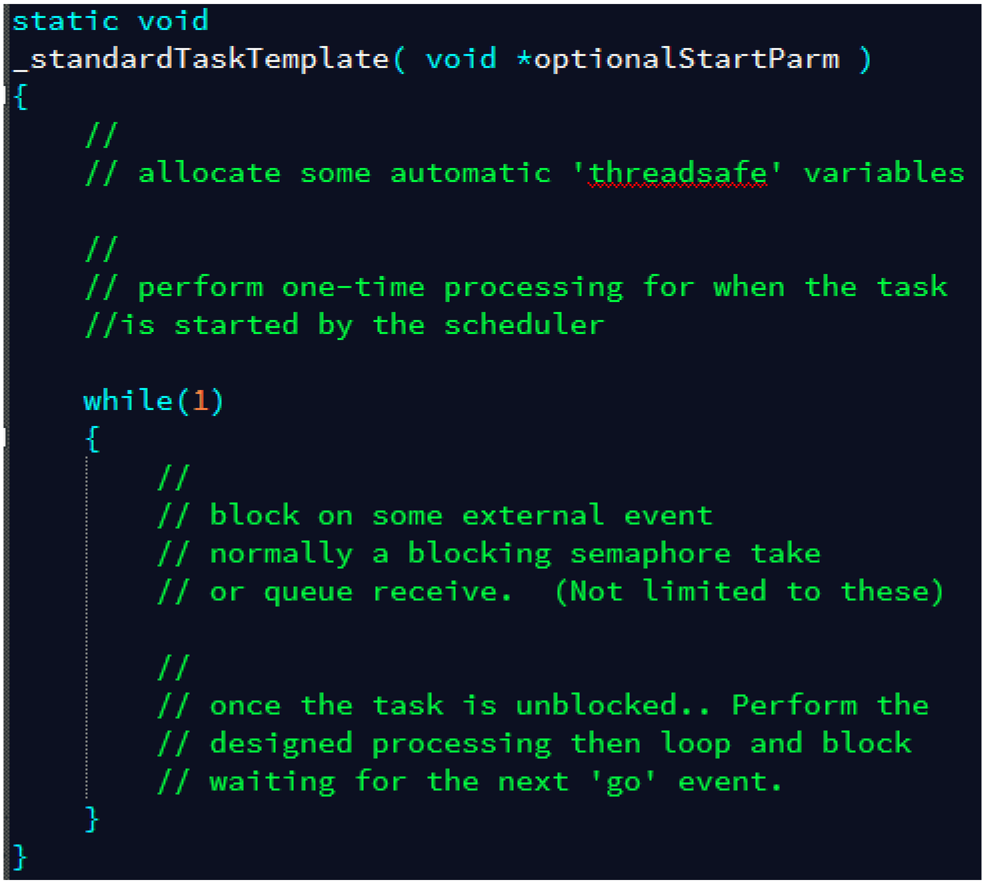
```
        $ cd ~/src/cs466_s17/TivaDriver/driverlib
        $ make clean
        $ make
```

4. □ Add code to read the status of `SW1` and `SW2`. You will need to enable new GPIO pins for input and internal pull-up resistors. Do both switches follow the same rules (hint, hint).

a) It should be a pretty simple change to the existing program to verify that `SW1` and `SW2` are sensed and actionable.

5. □ Add two tasks to the system for each of the other two LED behaviors.

6. □ Normally a simple RTOS task has the overall structure:



```
static void
_standardTaskTemplate( void *optionalStartParm )
{
    //
    // allocate some automatic 'threadsafe' variables

    //
    // perform one-time processing for when the task
    //is started by the scheduler

    while(1)
    {
        //
        // block on some external event
        // normally a blocking semaphore take
        // or queue receive.  (Not limited to these)

        //
        // once the task is unblocked.. Perform the
        // designed processing then loop and block
        // waiting for the next 'go' event.
    }
}
```

See if you can make the semaphore_take blocking on the new normally inactive tasks so the they will happily wait forever.

7. □ Modify the code so that:

a) The green LED always blinks at 1Hz

b) By default the red and blue LED, are off.

c) Pressing SW1 will cause the red led to flash 20 times at 15Hz

d) Pressing SW2 will cause the blue led to flash 10 times at 13Hz
e) This should work with any combination of SW1 and SW2
f) Use only the button press as a trigger, the led should blink their required number of times no matter how long the button is pressed.

8. □ Verify the frequency (or as close as you can get) with the scope. What can you do if the frequencies are off?

9. □ Add two behaviors
a) If you don't press a button in 60 seconds red and 90 seconds for blue that the LED's will flicker for 1 second to remind the user that they are available.
b) Because sometimes hardware fails, check that the buttons are not stuck by asserting if either is pressed for more than 3 minutes. Assert!.
c) (Note: as you are developing these behaviors.. If you test at the actual times you'll be sitting around for many many minutes waiting.. I suggest that you use a #define SPEED_TEST or some similar concept to reduce actual times during your implementation and all but final testing.. Imagine if the times were hours or weeks?)

10. □ Write up your lab using the lab format provided on Angel. Include your program as a fixed spaced (`I recommend Lucida Console`) addendum to your lab. I will cut points for proportionally spaced code pasted in the end of the lab.

!!! Must use provided lab format on Blackboard !!!