Nathan Hanuscin
Github root directory: https://github.com/nhanuscin/HappyFunStuff

Date Submitted: **11/11/2019**

Task 01:

Youtube Link: N/A

Modified Schematic (if applicable): N/A

Modified Code:

```
//---------------------------------------
// BIOS header files
//---------------------------------------
#include <xdc/std.h>             //mandatory - have to include first, for BIOS types
#include <ti/sysbios/BIOS.h>     //mandatory - if you call APIs like BIOS_start()
#include <xdc/runtime/Log.h>     //needed for any Log_info() call
#include <xdc/cfg/global.h>      //header file for statically defined objects/handles

#include <ti/sysbios/knl/Semaphore.h>         //when using Semaphores (dynamically)
#include <ti/sysbios/knl/Task.h>              //when using Tasks (dynamically)


//---------------------------------------
// TivaWare Header Files
//---------------------------------------
#include <stdint.h>
#include <stdbool.h>

#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "inc/hw_ints.h"
#include "driverlib/interrupt.h"
#include "driverlib/timer.h"


//---------------------------------------
// Prototypes
//---------------------------------------
void hardware_init(void);
void ledToggle(void);
void Timer_ISR(void);


//---------------------------------------
// Globals
//---------------------------------------
volatile int16_t i16ToggleCount = 0;

Semaphore_Handle LEDSem;
Task_Handle ledToggleTask;
```

```
//*****************************************************************************
// main()
//*****************************************************************************
void main(void)
{

	//---------------------------------------------------
	// [START] - DYNAMIC CREATION OF TASK AND SEMAPHORE
	//---------------------------------------------------

		Task_Params taskParams;

		LEDSem = Semaphore_create(0, NULL, NULL);
	// create ledToggleSem Semaphore

		Task_Params_init(&taskParams);
	// create ledToggleTask Task
		taskParams.priority = 2;
		ledToggleTask = Task_create((Task_FuncPtr)ledToggle, &taskParams, NULL);

	//---------------------------------------------------
	// [END] - DYNAMIC CREATION OF TASK AND SEMAPHORE
	//---------------------------------------------------

	//previous main() contents follow...


   hardware_init();                                    // init hardware via Xware

   BIOS_start();

}


//*****************************************************************************
// hardware_init()
//
// inits GPIO pins for toggling the LED
//*****************************************************************************
void hardware_init(void)
{
	uint32_t ui32Period;

	//Set CPU Clock to 40MHz. 400MHz PLL/2 = 200 DIV 5 = 40MHz
	SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAI
N);

	// ADD Tiva-C GPIO setup - enables port, sets pins 1-3 (RGB) pins for output
	SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
	GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

	// Turn on the LED
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```
	GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 4);

	// Timer 2 setup code
	SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER2);		// enable Timer 2
periph clks
	TimerConfigure(TIMER2_BASE, TIMER_CFG_PERIODIC);	// cfg Timer 2 mode
- periodic

	ui32Period = (SysCtlClockGet() /2);					//
period = CPU clk div 2 (500ms)
	TimerLoadSet(TIMER2_BASE, TIMER_A, ui32Period);		// set Timer
2 period

	TimerIntEnable(TIMER2_BASE, TIMER_TIMA_TIMEOUT);	// enables Timer 2
to interrupt CPU

	TimerEnable(TIMER2_BASE, TIMER_A);					//
enable Timer 2

}


//*****************************************************************************
// ledToggle()
//
// toggles LED on Tiva-C LaunchPad
//*****************************************************************************
void ledToggle(void)
{

	while(1)
	{
		Semaphore_pend(LEDSem, BIOS_WAIT_FOREVER);		//
wait for Sem from ISR

		// LED values - 2=RED, 4=BLUE, 8=GREEN
		if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
		{
			GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,
0);
		}
		else
		{
			GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
		}

		i16ToggleCount += 1;
	// keep track of #toggles

		Log_info1("LED TOGGLED [%u] TIMES",i16ToggleCount);	// send
toggle count to UIA

	}

}
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```
//---------------------------------------------------------------------------
// Timer ISR - called by BIOS Hwi (see app.cfg)
//
// Posts Swi (or later a Semaphore) to toggle the LED
//---------------------------------------------------------------------------
void Timer_ISR(void)
{
    TimerIntClear(TIMER2_BASE, TIMER_TIMA_TIMEOUT);              // must clear timer
flag FROM timer

      Semaphore_post(LEDSem);
      // post LEDSwi
}
-----------------------------------------------------------------------------------
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.