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


//------------------------------------------
// 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;

//-------------------------
// for Queue - Part B
//-------------------------
typedef struct MsgObj {
	Queue_Elem   elem;
	Int    val;                   // message value
} MsgObj, *Msg;                   // Use Msg as pointer to MsgObj
```

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

```
//---------------------------------------------------------------------------
// main()
//---------------------------------------------------------------------------
void main(void)
{

   hardware_init();                    // init hardware via Xware

   BIOS_start();                       // start BIOS Scheduler

}


//---------------------------------------------------------------------------
// 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
	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

}



//---------------------------------------------------------------------------
```

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

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

```
// mailbox_queue Task() - Run by BIOS_Start(), then unblocked by Timer ISR
//
// Places state of LED (msg.val) into a mailbox for ledToggle() to use
//---------------------------------------------------------------------------
void mailbox_queue(void)
{

//---------------------------------
// msg used for Mailbox and Queue
//---------------------------------
      MsgObj msg;
            // create an instance of MsgObj named msg

//---------------------------------
// msgp used for Queue only
//---------------------------------
      Msg msgp;
            // Queues pass POINTERS, so we need a pointer of type Msg
      msgp = &msg;
      // init message pointer to address of msg


      msg.val = 1;
      // set initial value of msg.val (LED state)

      while(1){

            msg.val ^= 1;
      // toggle msg.val (LED state)

            Semaphore_pend(mailbox_queue_Sem, BIOS_WAIT_FOREVER);
            // wait on semaphore from Timer ISR

//------------------------------
// MAILBOX CODE follows...
//------------------------------
            //Mailbox_post (LED_Mbx, &msg, BIOS_WAIT_FOREVER);               //
post msg containing LED state into the MAILBOX


//------------------------------
// QUEUE CODE follows...
//------------------------------
            Queue_put(LED_Queue, (Queue_Elem*)msgp);
      // pass pointer to Message object via LED_Queue
            Semaphore_post (QueSem);
      // unblock Queue_get to get msg

      }

}



//---------------------------------------------------------------------------
```

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

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

```
// ledToggle()  - called by BIOS_Start(), then unblocked by mailbox_queue()
//
// toggles LED on Tiva-C LaunchPad
//-----------------------------------------------------------------------------
void ledToggle(void)
{

//---------------------------------
// msg used for Mailbox and Queue
//---------------------------------
      MsgObj msg;
                                              //define msg using MsgObj struct
created earlier

//---------------------------------
// msgp used for Queue only
//---------------------------------
      Msg msgp;
                                              //define pointer to MsgObj to use with
queue put/get
      msgp = &msg;
                                        //init msgp to point to address of msg (used
for put/get)


      while(1)
      {


//------------------------------
// MAILBOX CODE follows...
//------------------------------
//            Mailbox_pend(LED_Mbx, &msg, BIOS_WAIT_FOREVER);
                                  // wait/block until post of msg, get msg.val


//------------------------------
// QUEUE CODE follows...
//------------------------------
             Semaphore_pend(QueSem, BIOS_WAIT_FOREVER);
                                 // unblocked by mailbox_queue() when Queue has msg
             msgp = Queue_get(LED_Queue);
                                              // read contents of queue to get value
of LED state


             // LED values - 0=OFF, 2=RED, 4=BLUE, 8=GREEN

             //if (msg.val)
                                              // MAILBOX "if" - msg.val contains LED
state

             if(msgp->val)
                                        // QUEUE "if" - mspg->val contains LED state
for QUEUE's the use pointers
```

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

```
			{
			GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 8);
	// turn LED on
			}
			else
			{
			GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
	// turn LED off
			}

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

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

	}
}



//---------------------------------------------------------------------------
// Timer_ISR()
//
// Called by Hwi when timer hits zero
//
// TimerIntClear is needed here because THIS fxn is the ISR now
//---------------------------------------------------------------------------
void Timer_ISR(void)
{
    TimerIntClear(TIMER2_BASE, TIMER_TIMA_TIMEOUT);              // must clear timer
flag FROM timer

	Semaphore_post(mailbox_queue_Sem);
		// post Sem to unblock mailbox-queue-task

}
----------------------------------------------------------------------------------
```

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