Lab 10

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

// Project: Blink TM4C BIOS Using Task/Sem Dynamically (MEM) (STARTER)

// Author: Eric Wilbur

// Date: June 2014

//

// Note: The function call TimerIntClear(TIMER2\_BASE, TIMER\_TIMA\_TIMEOUT) HAS

// to be in the ISR. This fxn clears the TIMER's interrupt flag coming

// from the peripheral - it does NOT clear the CPU interrupt flag - that

// is done by hardware. The author struggled figuring this part out - hence

// the note. And, in the Swi lab, this fxn must be placed in the

// Timer\_ISR fxn because it will be the new ISR.

//

// Follow these steps to create this project in CCSv6.0:

// 1. Project -> New CCS Project

// 2. Select Template:

// - TI-RTOS for Tiva-C -> Driver Examples -> EK-TM4C123 LP -> Example Projects ->

// Empty Project

// - Empty Project contains full instrumentation (UIA, RTOS Analyzer) and

// paths set up for the TI-RTOS version of MSP430Ware

// 3. Delete the following files:

// - Board.h, empty.c, EK\_TM4C123GXL.c/h, empty\_readme.txt

// 4. Add main.c from TI-RTOS Workshop Solution file for this lab

// 5. Edit empty.cfg as needed (to add/subtract) BIOS services, delete given Task

// 6. Build, load, run...

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

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

// 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 taskOne;

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

// 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;

taskOne = 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\_MAIN);

// 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

}

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

// 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

}

}

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

// Timer ISR - called by BIOS Hwi (see app.cfg)

//

// Posts Swi (or later a Semaphore) to toggle the LED

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

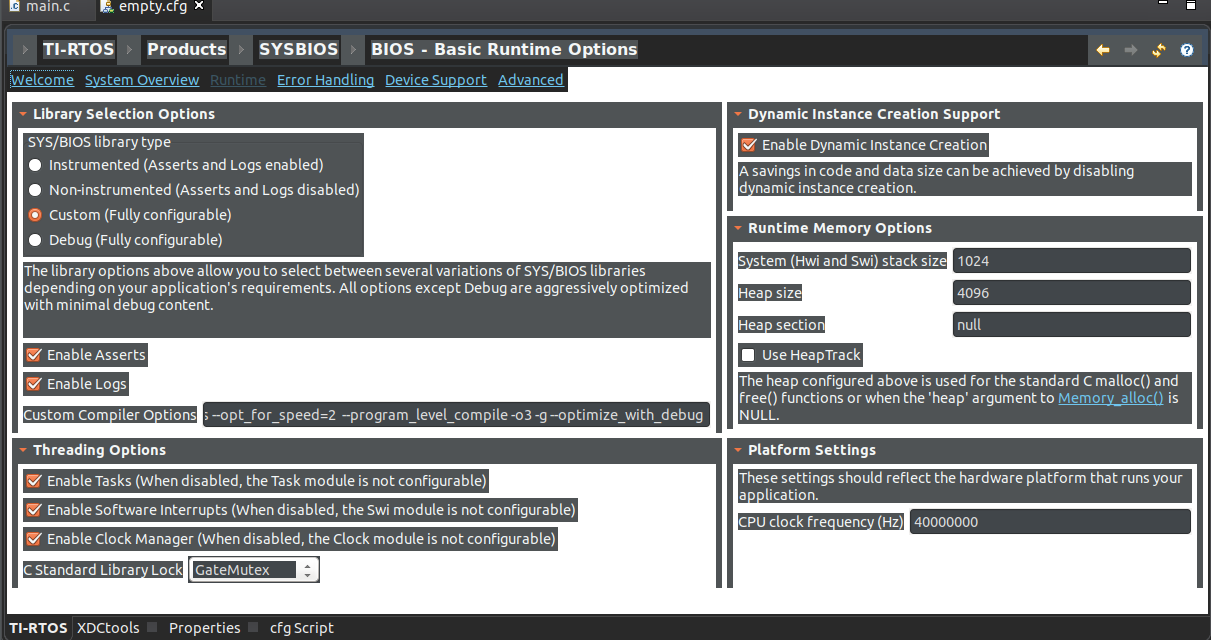
**void** **Timer\_ISR**(**void**)

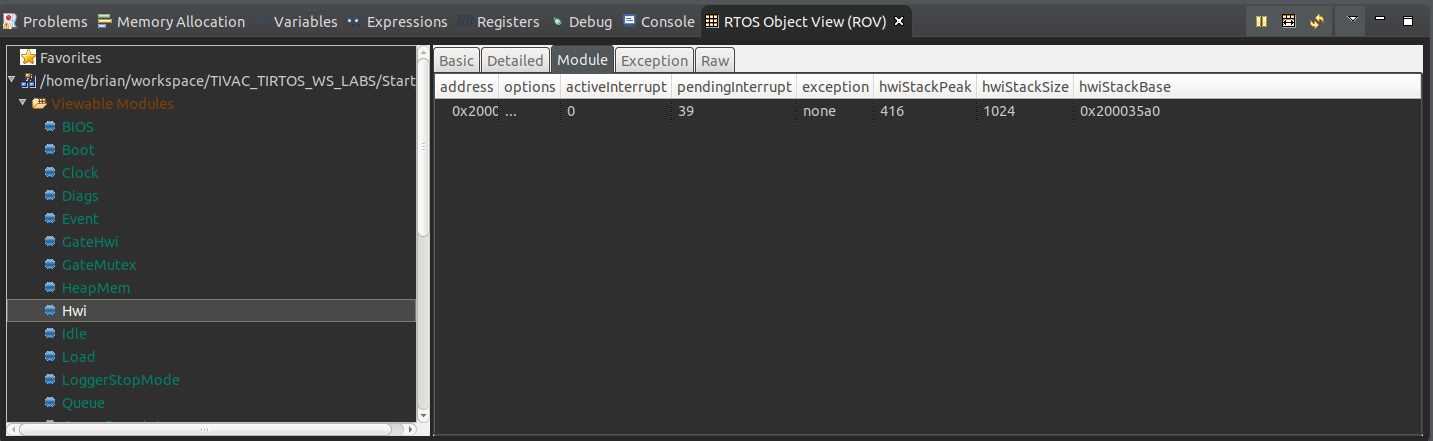
{

**TimerIntClear**(TIMER2\_BASE, TIMER\_TIMA\_TIMEOUT); // must clear timer flag FROM timer

**Semaphore\_post**(LEDSem); // post LEDSwi

}

Changes to .cfg file

Stack usage

