**Date Submitted:**

**Goal:**

**Using TI-RTOS, program the TM4C to perform three tasks:**

1. **Task 01: Read the potentiometer value using an ADC on PE0.**
2. **Task 02: Print the read potentiometer value to the terminal using UART.**
3. **Task 03: Using the two switches, adjust the PWM of the LED.**

**Each task executes every 30 ms: Task 01 at 10 ms; Task 02 at 20 ms; Task 03 at 30 ms.**

**------------------------------------------------------------------------------------**

**Task 01:**

**Modified Code:**

// TASK #1

// Read the ADC value at PE0

// Value ranges from 0 < x < 4096

**void** **potADC**(**void**)

{

**while**(1)

{

Semaphore\_pend(potSem, BIOS\_WAIT\_FOREVER);

**ADCIntClear**(ADC1\_BASE, 1);

**ADCProcessorTrigger**(ADC1\_BASE, 1);

// Poll for the ADC flag

**while**(!**ADCIntStatus**(ADC1\_BASE, 1, false));

**ADCSequenceDataGet**(ADC1\_BASE, 1, ui32ADC1Value);

// Average out the sampled analog signal

ui32PotValue = (ui32ADC1Value[0] + ui32ADC1Value[1] + ui32ADC1Value[2] + ui32ADC1Value[3] + 2)/4;

}

}

**------------------------------------------------------------------------------------**

**Task 02:**

**Modified Code:**

// Function to print a string to the terminal

**void** **printString**(**char** \*string)

{

**while**(\*string)

{

**UARTCharPut**(UART0\_BASE, \*string);

string++;

}

}

// Task #2

// Display the ADC value on the terminal

**void** **UARTDisplay**(**void**)

{

**char** buffer[4];

**while**(1)

{

Semaphore\_pend(UARTSem, BIOS\_WAIT\_FOREVER);

**ltoa**(ui32PotValue, buffer); // Convert the potentiometer value into a string

printString("Read Potentiometer Value: ");

printString(buffer);

**UARTCharPut**(UART0\_BASE, '\n');

**UARTCharPut**(UART0\_BASE, '\r');

}

}

**-----------------------------------------------------------------------------------**

**Task 03:**

**Modified Code:**

// Task #3

// Adjust the DC of the PWM using the two switches.

// Changing the DC will adjust the red LED's brightness.

**void** **PWMSwitch**(**void**)

{

**while**(1)

{

Semaphore\_pend(PWMSem, BIOS\_WAIT\_FOREVER);

// Decrease DC

**if**(**GPIOPinRead**(GPIO\_PORTF\_BASE,GPIO\_PIN\_4)==0x00)

{

ui16Adjust = ui16Adjust - 100;

// Min DC = 10%

**if** (ui16Adjust < 1136)

{

ui16Adjust = 1136;

}

**PWMPulseWidthSet**(PWM1\_BASE, PWM\_OUT\_5, ui16Adjust);

}

// Increase DC

**if**(**GPIOPinRead**(GPIO\_PORTF\_BASE,GPIO\_PIN\_0)==0x00)

{

ui16Adjust = ui16Adjust + 100;

// Max DC = 90%

**if** (ui16Adjust > 10225)

{

ui16Adjust = 10225;

}

**PWMPulseWidthSet**(PWM1\_BASE, PWM\_OUT\_5, ui16Adjust);

}

}

}

**Servicing Each Task:**

**Modified Code:**

// Function to service the 1 ms HWI\_TIMER2

// Timeline:

// 10 ms - Task #1

// 20 ms - Task #2

// 30 ms - Task #3

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

{

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

i8TaskTime++; // Update counter

**switch**(i8TaskTime)

{

// At 10 ms, post the potentiometer [task #1]

**case** 10:

Semaphore\_post(potSem);

**break**;

// At 20 ms, post the UART [task #2]

**case** 20:

Semaphore\_post(UARTSem);

**break**;

// At 30 ms, post the PWM [task #3]

**case** 30:

Semaphore\_post(PWMSem);

// Reset the counter

i8TaskTime = 0;

}

}