**Date Submitted: 10/28/19**

**Video Link:** [**https://youtu.be/XsasAnN3nII**](https://youtu.be/XsasAnN3nII)

**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: Took the code from Lab 6 PWM for potentiometer.**

// 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: Took the code from Lab 7 UART.**

// 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: Took the code from Lab 6 LED.**

// 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);

}

}

}

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

**Timer to Schedule 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;

}

}

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

**Hardware Initialization:**

**Modified Code:**

**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);

// 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**() / 1000); // period = CPU clk div 1000 (1 ms)

**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

// Setup ADC1 Channel 3

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC1);

**ADCSequenceConfigure**(ADC1\_BASE, 1, ADC\_TRIGGER\_PROCESSOR, 0);

**ADCSequenceStepConfigure**(ADC1\_BASE, 1, 0, ADC\_CTL\_CH3);

**ADCSequenceStepConfigure**(ADC1\_BASE, 1, 1, ADC\_CTL\_CH3);

**ADCSequenceStepConfigure**(ADC1\_BASE, 1, 2, ADC\_CTL\_CH3);

**ADCSequenceStepConfigure**(ADC1\_BASE, 1, 3, ADC\_CTL\_CH3 | ADC\_CTL\_IE | ADC\_CTL\_END);

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

// Setup PE0 as an ADC input to ADC1 Channel 3

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOE); // Enable GPIOE

**GPIOADCTriggerEnable**(GPIO\_PORTE\_BASE, GPIO\_PIN\_0); // PE0 is ADC input

// Setup UART0

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_UART0);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOA); // Enable GPIOA

**GPIOPinConfigure**(GPIO\_PA0\_U0RX); // Configure the PA0 and PA1 for UART

**GPIOPinConfigure**(GPIO\_PA1\_U0TX);

**GPIOPinTypeUART**(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

**UARTConfigSetExpClk**(UART0\_BASE, **SysCtlClockGet**(), 115200,

(UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE | UART\_CONFIG\_PAR\_NONE));

// Set the PWM clock

**SysCtlPWMClockSet**(SYSCTL\_PWMDIV\_64);

// Enables PWM1, GPIOD

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_PWM1);

// Configure PF1 as a PWM

**GPIOPinTypePWM**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1);

**GPIOPinConfigure**(GPIO\_PF1\_M1PWM5);

// Unlock PF0 for SW

HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_LOCK) = GPIO\_LOCK\_KEY;

HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_CR) |= 0x01;

HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_LOCK) = 0;

// Configure pull-up for PF0 and PF4

**GPIODirModeSet**(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0, GPIO\_DIR\_MODE\_IN);

**GPIOPadConfigSet**(GPIO\_PORTF\_BASE, GPIO\_PIN\_4|GPIO\_PIN\_0, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPU);

// Configure PWM clock

ui32PWMClock = **SysCtlClockGet**() / 64;

ui32Load = (ui32PWMClock / PWM\_FREQUENCY) - 1;

**PWMGenConfigure**(PWM1\_BASE, PWM\_GEN\_2, PWM\_GEN\_MODE\_DOWN);

**PWMGenPeriodSet**(PWM1\_BASE, PWM\_GEN\_2, ui32Load);

// Adjust width and enable PWM

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

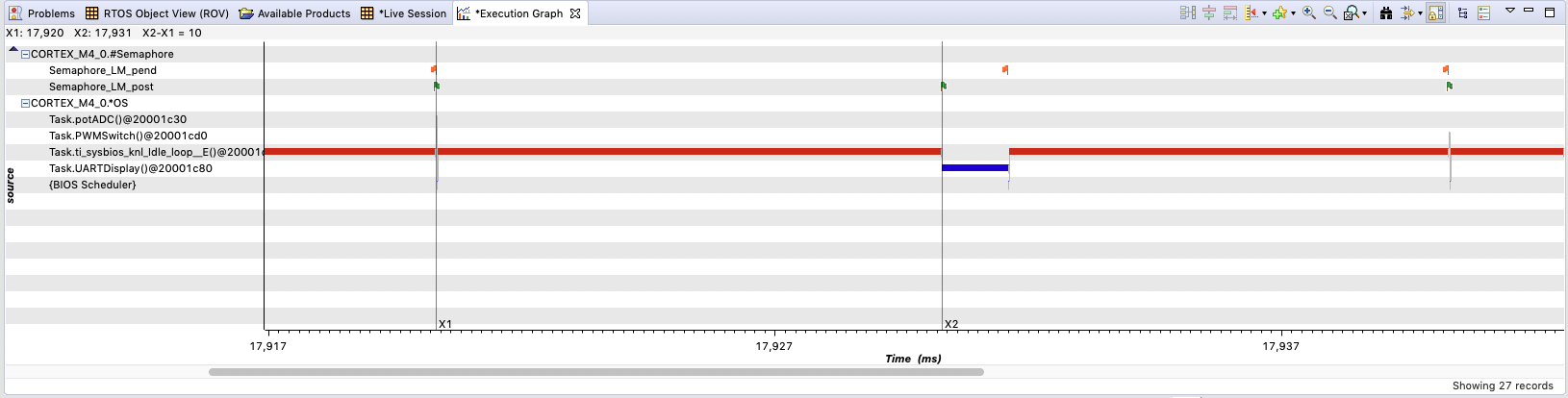
**PWMOutputState**(PWM1\_BASE, PWM\_OUT\_5\_BIT, true);

**PWMGenEnable**(PWM1\_BASE, PWM\_GEN\_2);

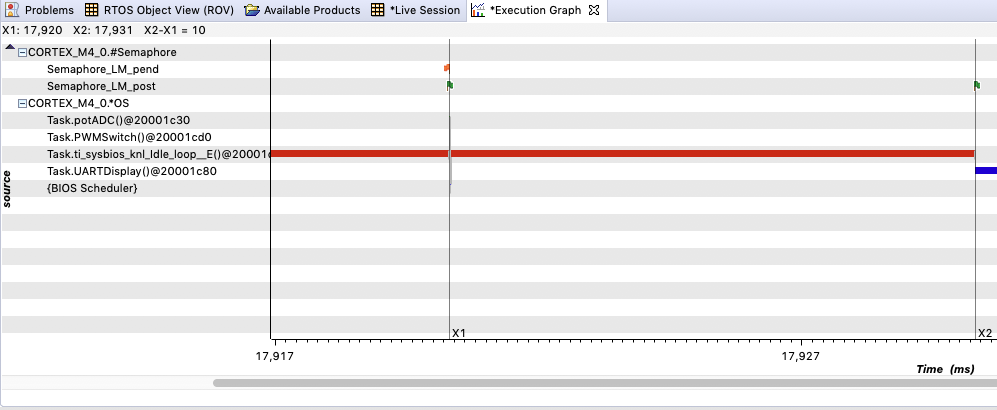
}

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

**Execution Graph:**

****

**Overall Graph**

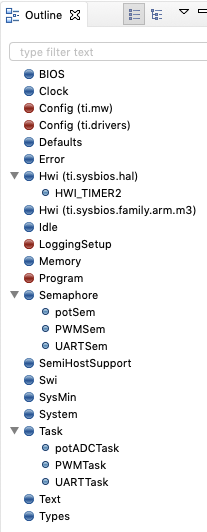
****

**Using the measuring tool, it’s clear that each task are 10 ms apart from each other.**

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

**Conclusion:**

**This assignment involves using the TI-RTOS GUI to program the TM4C to perform three tasks: read the ADC input, output to the terminal, and adjust the brightness of the LED. Each task is 10 ms apart from each other and a task repeats every 30 ms. When running the debugger, use the ‘Execution Graph’ tool to see if the scheduler is performing the three tasks on time. The tasks are timed using a hardware interrupt and semaphores.**

****

**Outline of the RTOS**