Task 01: Change the ADC Sequencer to SS2. Turn on the LED at PF3 if the temperature is greater that 75 degF. Use internal temperature sensor for all SS2 sequence.

The task asked for 75 degrees fahrenheit but it was switched to 70 for the video for easier use. The code still reflects 75 degrees. Below is the code.

…

#include "driverlib/rom.h"

#include "driverlib/gpio.h"

//to use the LEDs

...

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3);

//enable and specify LEDs as outputs

ADCHardwareOversampleConfigure(ADC0\_BASE, 64);

ADCSequenceConfigure(ADC0\_BASE, 2, ADC\_TRIGGER\_PROCESSOR, 0);

//Configure ADC sequencer. ADC0, sample sequencer 2, highest priority

ADCSequenceStepConfigure(ADC0\_BASE, 2, 0, ADC\_CTL\_TS);

ADCSequenceStepConfigure(ADC0\_BASE, 2, 1, ADC\_CTL\_TS);

ADCSequenceStepConfigure(ADC0\_BASE, 2, 2, ADC\_CTL\_TS);

//Sequencing steps 0-2

ADCSequenceStepConfigure(ADC0\_BASE, 2, 3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

ADCSequenceEnable(ADC0\_BASE, 2);

//Enable ADC sequencer 2

while(1)

{

ADCIntClear(ADC0\_BASE, 2);

ADCProcessorTrigger(ADC0\_BASE, 2);

while(!ADCIntStatus(ADC0\_BASE, 2, false))

{

}

ADCSequenceDataGet(ADC0\_BASE, 2, ui32ADC0Value);

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] +2) / 4;

ui32TempValueC = (1475 - ((2475\*ui32TempAvg)) / 4096) / 10;

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5;

if(ui32TempValueF > 75)//to check if temp is higher than 75. IF so, turn on red LED

{

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 2);

}

else

{

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 0);

}

}

}

Task 02: Introduce hardware averaging to 32. Using the timer TIMER1A conduct an ADC con-

version on overflow every 0.5 sec. Use the Timer1A interrupt.

To get an interrupt at every half second. Timer1A was set to a period of .5s using the equation in the code below.

...

#include "driverlib/rom.h"

#include "driverlib/gpio.h" //task 01

#include "driverlib/timer.h" //task 02, to use timer and interrupts

#include "driverlib/interrupt.h"

#include "inc/tm4c123gh6pm.h"

…

int main(void)

{

uint32\_t ui32Period;

SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER1);//task 02

TimerConfigure(TIMER1\_BASE, TIMER\_CFG\_PERIODIC);

ui32Period = (SysCtlClockGet()/2);

//Set period to .5seconds full duty cycle

TimerLoadSet(TIMER1\_BASE, TIMER\_A, ui32Period-1);

TimerEnable(TIMER1\_BASE, TIMER\_A);

TimerIntEnable(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

IntEnable(INT\_TIMER1A);

IntMasterEnable();

ADCHardwareOversampleConfigure(ADC0\_BASE, 32);//64);

ADCSequenceConfigure(ADC0\_BASE, 2, ADC\_TRIGGER\_PROCESSOR, 0);

//Configure ADC sequencer. ADC0, sample sequencer 2, highest priority

ADCSequenceStepConfigure(ADC0\_BASE, 2, 0, ADC\_CTL\_TS);

ADCSequenceStepConfigure(ADC0\_BASE, 2, 1, ADC\_CTL\_TS);

ADCSequenceStepConfigure(ADC0\_BASE, 2, 2, ADC\_CTL\_TS);

//Sequencing steps 0-2

ADCSequenceStepConfigure(ADC0\_BASE, 2, 3, ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

ADCSequenceEnable(ADC0\_BASE, 2);

//Enable ADC sequencer 1

while(1)

{//Loop here while waiting for interrupt

}

}

void Timer1IntHandler(void)

{

//Interrupt handler. Calculations have been moved here. Done whenever .5 seconds elapse in

//Timer 1A

volatile uint32\_t ui32TempAvg;

volatile uint32\_t ui32TempValueC;

volatile uint32\_t ui32TempValueF;

uint32\_t ui32ADC0Value[4];

//array for storing data read from ADC FIFO, Sequencer 1

//Clear the timer interrupt

TimerIntClear(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

ADCIntClear(ADC0\_BASE, 2);

ADCProcessorTrigger(ADC0\_BASE, 2);

while(!ADCIntStatus(ADC0\_BASE, 2, false))

{

}

ADCSequenceDataGet(ADC0\_BASE, 2, ui32ADC0Value);

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] +2) / 4;

ui32TempValueC = (1475 - ((2475\*ui32TempAvg)) / 4096) / 10;

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5;

}