**Date Submitted: 10/1/2019**

**Task 00: Execute provided code**

**Youtube Link:** <https://www.youtube.com/watch?v=iFQrkolzrCs>

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

**Task 01:**

Youtube Link: <https://www.youtube.com/watch?v=BR_pk3PW3Es>

**Modified Schematic (if applicable):**

**Modified Code:**

**// Insert code here**

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_types.h"

**#include** "driverlib/debug.h"

**#include** "driverlib/sysctl.h"

**#include** "driverlib/adc.h" // includes adc functions

**#include** "driverlib/gpio.h" // includes GPIO functions

**int** **main**(**void**)

{

uint32\_t ui32ADC0Value[4]; // array to store samples of ADC with 4 steps

**volatile** uint32\_t ui32TempAvg; // stores avg temp

**volatile** uint32\_t ui32TempValueC; // stores temp in Celsius

**volatile** uint32\_t ui32TempValueF; // stores temp in Fahrenheit

**SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ); // using 40MHz clock source

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0); //enable ADC0 peripheral

//Configure ADC

**ADCSequenceConfigure**(ADC0\_BASE, 1, ADC\_TRIGGER\_PROCESSOR, 0); // using ADC sample sequencer 1 (SS1), set as the highest priority, and processor will trigger ADC

**ADCSequenceStepConfigure**(ADC0\_BASE, 1, 0, ADC\_CTL\_TS); // ADC sample step 0

**ADCSequenceStepConfigure**(ADC0\_BASE, 1, 1, ADC\_CTL\_TS); // ADC sample step 1

**ADCSequenceStepConfigure**(ADC0\_BASE, 1, 2, ADC\_CTL\_TS); // ADC sample step 2

**ADCSequenceStepConfigure**(ADC0\_BASE,1,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END); //ADC sample step 3, set ADC interrupt flag, end sampling

**ADCSequenceEnable**(ADC0\_BASE, 1); // enable ADC0

//configure and enable GPIO PORT F

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF); // enable GPIO port F peripheral

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3); // set PORTF as output

**while**(1)

{

**if** (ui32TempValueF >= 75) { // if temp value is greater than or equal to 75 degrees Fahrenheit

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0);

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 4); // turn on blue LED

}

**else** { // if temp value is less than 75 degrees Fahrenheit

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0);

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 2); // turn on red LED

}

**ADCIntClear**(ADC0\_BASE, 1); // clear ADC interrupt

**ADCProcessorTrigger**(ADC0\_BASE, 1); // processor begins to trigger ADC

**while**(!**ADCIntStatus**(ADC0\_BASE, 1, **false**)) // wait for ADC conversion..

{

}

**ADCSequenceDataGet**(ADC0\_BASE, 1, ui32ADC0Value); // get ADC value from samples

//calculations of temperatures

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

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

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

}

}**------------------------------------------------------------------------------------**

**Task 02:**

Youtube Link: <https://www.youtube.com/watch?v=3hGuJrnBst4>

**Modified Schematic (if applicable):**

**Modified Code:**

**// Insert code here**

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "inc/tm4c123gh6pm.h" // includes interrupt assignments for the Tiva C

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_types.h"

**#include** "driverlib/debug.h"

**#include** "driverlib/sysctl.h"

**#include** "driverlib/interrupt.h" // includes interrupt functions

**#include** "driverlib/adc.h" // includes adc functions

**#include** "driverlib/timer.h" // includes timer functions

//GLOBAL VARIABLES

uint32\_t ui32ADC0Value[4]; // array to store samples of ADC with 4 steps

**volatile** uint32\_t ui32TempAvg; // stores avg temp

**volatile** uint32\_t ui32TempValueC; // stores temp in Celsius

**volatile** uint32\_t ui32TempValueF; // stores temp in Fahrenheit

//Timer 1 Delay Function

**void** **timer1A\_delaySec**(**int** ttime) {

**int** i;

SYSCTL\_RCGCTIMER\_R |= 2;

TIMER1\_CTL\_R = 0;

TIMER1\_CFG\_R = 0x04;

TIMER1\_TAMR\_R = 0x02;

TIMER1\_TAILR\_R = 64000 - 1;

TIMER1\_TAPR\_R = 250 - 1;

TIMER1\_ICR\_R = 0x1;

TIMER1\_CTL\_R |= 0x01;

**for** (i=0; i < ttime; i++) {

**while** ((TIMER1\_RIS\_R & 0x1) == 0);

TIMER1\_ICR\_R = 0x1;

}

}

**int** **main**(**void**)

{

**SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ); // using 40MHz clock source

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0); //enable ADC0 peripheral

**ADCHardwareOversampleConfigure**(ADC0\_BASE, 32); // introducing hardware averaging with 32 samples to be averaged

//Configure ADC

**ADCSequenceConfigure**(ADC0\_BASE, 1, ADC\_TRIGGER\_PROCESSOR, 0); // using ADC sample sequencer 1 (SS1), set as the highest priority, and processor will trigger ADC

**ADCSequenceStepConfigure**(ADC0\_BASE, 1, 0, ADC\_CTL\_TS); // ADC sample step 0

**ADCSequenceStepConfigure**(ADC0\_BASE, 1, 1, ADC\_CTL\_TS); // ADC sample step 1

**ADCSequenceStepConfigure**(ADC0\_BASE, 1, 2, ADC\_CTL\_TS); // ADC sample step 2

**ADCSequenceStepConfigure**(ADC0\_BASE,1,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END); //ADC sample step 3, set ADC interrupt flag, end sampling

**ADCSequenceEnable**(ADC0\_BASE, 1); // enable ADC0

//configure and enable Timer1 and Interrupt for Timer1

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER1); // enable timer1 peripheral

**IntMasterEnable**(); // enable all interrupts

**TimerConfigure**(TIMER1\_BASE, TIMER\_CFG\_PERIODIC); // config timer 1 to periodic mode

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, 120000000); // initial delay for timer

**IntEnable**(INT\_TIMER1A); // enable timer1A interrupt on vector table

**TimerIntEnable**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT); // trigger timer 1 int at TIMEOUT

**TimerEnable**(TIMER1\_BASE, TIMER\_A); // enable timer1A

**while**(1)

{

// wait for timer 1 interrupt to occur...

}

}

**void** **timer1A\_ISR** (**void**) {

**TimerIntClear**(TIMER1\_BASE, TIMER\_A); // clear int for timer 1

// call timer1a delay function

timer1A\_delaySec(0.5); // delay for 0.5s

// perform ADC conversion

**ADCIntClear**(ADC0\_BASE, 1); // clear ADC interrupt

**ADCProcessorTrigger**(ADC0\_BASE, 1); // processor begins to trigger ADC

**while**(!**ADCIntStatus**(ADC0\_BASE, 1, **false**)) // wait for ADC conversion..

{

}

**ADCSequenceDataGet**(ADC0\_BASE, 1, ui32ADC0Value); // get ADC value from samples

//calculations of temperatures

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

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

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

}

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