**Date Submitted:**

**Task 00: Execute provided code**

**Youtube Link:**

**No Submission required for Task 00**

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

**Task 01:**

Youtube Link: <https://youtu.be/ROXDzclzz3g>

**Modified Schematic (if applicable): N/A**

**Modified Code:**

**// Insert code here**

**#include** <stdint.h> //include all necessary libraries

**#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"

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

**#ifdef** DEBUG

**void\_\_error\_\_**(**char** \*pcFilename, uint32\_t ui32Line)

{

}

**#endif**

uint8\_t ui8PinData;

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

{

uint32\_t ui32ADC0Value[4];

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

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

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0);

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

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE,GPIO\_PIN\_2); //set pins as outputs

//For every ADC0\_BASE , 1 change the 1 to a 2 to switch the ADC Sequencer to SS2

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

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

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

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

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

**ADCSequenceEnable**(ADC0\_BASE, 2);

**while**(1)

{

**ADCIntClear**(ADC0\_BASE, 2); //clear the ADC interrupt status flag

**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 > 65) //if temperature is over this number then the LED at PF2 will turn on, if not it will stay off

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE,GPIO\_PIN\_2, 0x0E); //turn LED on at PF2

}

**else**

**GPIOPinWrite**(GPIO\_PORTF\_BASE,GPIO\_PIN\_2, 0x00); //turn LED off at PF2

}

}

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

**Task 02:**

Youtube Link: <https://youtu.be/aFtx91sYaDY>

**Modified Schematic (if applicable): N/A**

**Modified Code:**

**#include** <stdint.h> //include all necessary libraries

**#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"

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

**#include** "inc/tm4c123gh6pm.h" //added libraries for task 2

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

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

**#ifdef** DEBUG

**void\_\_error\_\_**(**char** \*pcFilename, uint32\_t ui32Line)

{

}

**#endif**

uint8\_t ui8PinData;

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

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

{

uint32\_t ui32Period;

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

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0);

**ADCHardwareOversampleConfigure**(ADC0\_BASE, 32);

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

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE,GPIO\_PIN\_2); //set pins as outputs

//For every ADC0\_BASE , 1 change the 1 to a 2 to switch the ADC Sequencer to SS2

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

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

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

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

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

**ADCSequenceEnable**(ADC0\_BASE, 2);

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

**TimerConfigure**(TIMER1\_BASE, TIMER\_CFG\_PERIODIC); //configure timer1 to periodic mode

ui32Period = **SysCtlClockGet**()/2; //Timer overflow every 0.5 second

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

**IntEnable**(INT\_TIMER1A);//enables the specific vector associated with Timer0A

**TimerIntEnable**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);//enables a specific event within the timer to generate an interrupt

**IntMasterEnable**();//master interrupt enable API for all interrupts

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

**while**(1)

{

}

}

**void** **Timer1IntHandler**(**void**)

{

uint32\_t ui32ADC0Value[4];

// Clear the timer interrupt

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

// Read the current state of the GPIO pin and

// write back the opposite state

**ADCIntClear**(ADC0\_BASE, 2); //clear the ADC interrupt status flag

**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 > 65) //if temperature is over this number then the LED at PF2 will turn on, if not it will stay off

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE,GPIO\_PIN\_2, 0x0E); //turn LED on at PF2

}

**else**

**GPIOPinWrite**(GPIO\_PORTF\_BASE,GPIO\_PIN\_2, 0x00); //turn LED off at PF2

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, **SysCtlClockGet**()/2); //reset timer1

}

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