**Date Submitted: 10/09/2018**

**Task 01:**

Youtube Link: <https://www.youtube.com/watch?v=VkdOh1EFsvk>

**// 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" // definitions for using the ADC driver

**#define** TARGET\_IS\_BLIZZARD\_RB1 // Gives the libraries access to the proper API’s in ROM.

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

**#include** "driverlib/gpio.h" // Include GPIO apis

**#ifdef** DEBUG

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

{

}

**#endif**

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

{

uint32\_t ui32ADC0Value[4]; // Stores the data read from the ADC FIFO

**volatile** uint32\_t ui32TempAvg; // Stores the avg of temperature

**volatile** uint32\_t ui32TempValueC; // Temperature in Celsius

**volatile** uint32\_t ui32TempValueF; // Temperature in Fahrenheit

// Set the system clock to run at 40MHz **SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF); // Enable GPIOF to use LED @ PF2

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0); // Enable ADC0

**ADCHardwareOversampleConfigure**(ADC0\_BASE, 64); // Hardware Averaging

// each sample in the ADC FIFO will be the result of 64 measurements being averaged together.

// Use ADC0, sample sequencer 3, let processor trigger sequence and use highest priority

**ADCSequenceConfigure**(ADC0\_BASE, 3, ADC\_TRIGGER\_PROCESSOR, 0);

// Gets four samples of the temperature sensor to average out

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

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

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

// Sample the temperaure sensor and

// configure the interrupt flag to be set when the sample is done.

**ADCSequenceStepConfigure**(ADC0\_BASE,1,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

**ADCSequenceEnable**(ADC0\_BASE, 1); // Enable ADC sequencer 1.

**while**(1)

{

**ADCIntClear**(ADC0\_BASE, 1); // Clear flag

**ADCProcessorTrigger**(ADC0\_BASE, 1); // Trigger ADC conversion

// note, assignment asked for threshold to be 72 but that was unreasonable in my circumstance.

**if** (ui32TempValueF < 64 )

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

**else**

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2,4);

**while**(!**ADCIntStatus**(ADC0\_BASE, 1, false)) // Wait for conversion

{

}

// Returns the samples that are presently available.

**ADCSequenceDataGet**(ADC0\_BASE, 1, ui32ADC0Value);

// Calculates the average of the temperature sensor data

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=Hw0JB9gPym8>

**// 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" // definitions for using the ADC driver

**#define** TARGET\_IS\_BLIZZARD\_RB1 // Gives the libraries access to the proper API’s in ROM.

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

**#include** "driverlib/gpio.h" // Include GPIO apis

**#include** "inc/tm4c123gh6pm.h" // Included for INT\_TIMER1A

**#include** "driverlib/timer.h" // Include Timer apis

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

**#ifdef** DEBUG

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

{

}

**#endif**

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

{

uint32\_t ui32Period;

// Set the system clock to run at 40MHz

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

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF); // Enable GPIOF to use LED @ PF2

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0); // Enable ADC0

**ADCHardwareOversampleConfigure**(ADC0\_BASE, 32); // Hardware Averaging

// each sample in the ADC FIFO will be the result of 64 measurements being averaged together.

// Use ADC0, sample sequencer 3, let processor trigger sequence and use highest priority

**ADCSequenceConfigure**(ADC0\_BASE, 3, ADC\_TRIGGER\_PROCESSOR, 0);

// Gets four samples of the temperature sensor to average out

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

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

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

// Sample the temperaure sensor and

// configure the interrupt flag to be set when the sample is done.

**ADCSequenceStepConfigure**(ADC0\_BASE,1,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

**ADCSequenceEnable**(ADC0\_BASE, 1); // Enable ADC sequencer 1.

// Enable timer1 peripheral, configure as periodic

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER1);

// configure Timer1A as periodic timer

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

//Find value to set to 2Hz or 0.5 second period till overflow and load timer

ui32Period = **SysCtlClockGet**()/2;

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

// Enable interrupts

**IntEnable**(INT\_TIMER1A); // Enable timer 1 interrupt

**TimerIntEnable**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT); // Enable event to trigger interrupt

**IntMasterEnable**(); // Enable master interrupt

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

**while**(1)

{

}

}

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

{

uint32\_t ui32ADC0Value[4]; // Stores the data read from the ADC FIFO

**volatile** uint32\_t ui32TempAvg; // Stores the avg of temperature

**volatile** uint32\_t ui32TempValueC; // Temperature in Celsius

**volatile** uint32\_t ui32TempValueF; // Temperature in Fahrenheit

**TimerIntClear**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT); // Clear timer interrupt

**ADCIntClear**(ADC0\_BASE, 1); // Clear ADC flag

**ADCProcessorTrigger**(ADC0\_BASE, 1); // Trigger ADC conversion

// note, assignment asked for threshold to be 72 but that was unreasonable in my circumstance.

**if** (ui32TempValueF < 67 )

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

**else**

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2,4);

**while**(!**ADCIntStatus**(ADC0\_BASE, 1, false)) // Wait for conversion

{

}

// Returns the samples that are presently available.

**ADCSequenceDataGet**(ADC0\_BASE, 1, ui32ADC0Value);

// Calculates the average of the temperature sensor data

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

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

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

}

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