**Task 0**

No submission required - ran the given code - code submitted on github

**Task 1**

#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"

#define TARGET\_IS\_BLIZZARD\_RB1

#include "driverlib/rom.h"

#include "driverlib/gpio.h"

/\*

#ifdef DEBUG

void\_\_error\_\_(char \*pcFilename, uint32\_t ui321Line){}

#endif

\*/

int main(void)

{

// my variables

uint32\_t ui32ADC0Value[4];

volatile uint32\_t ui32TempAvg;

volatile uint32\_t ui32TempValueC;

volatile uint32\_t ui32TempValueF;

// set the clock

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

// enable periphery

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

// GPIO output enable for LEDs

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

// enable ADC0, set to 64

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 64);

// configure ADC sequencer to 3 and assign the ADC value types

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

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

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

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 2, 2, ADC\_CTL\_TS);

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

ROM\_ADCSequenceEnable(ADC0\_BASE, 2);

//ROM\_ADCIntEnable(ADC0\_BASE, 3);

while(1)

{

// used SS2 for internal temperature sensor

// clear ADC interrupt and set processor trigger

ROM\_ADCIntClear(ADC0\_BASE, 2);

ROM\_ADCProcessorTrigger(ADC0\_BASE, 2);

// continue to loop while calculations are taking place

while(!ROM\_ADCIntStatus(ADC0\_BASE, 2, false)){}

// here we grab all the adc data and calculate the C and F

ROM\_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;

// i picked 62 because I could only get the temp up to 64 but

// still wanted to show LED turn on and it always initialized at 62

if(ui32TempValueF > 62){GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0x04);}

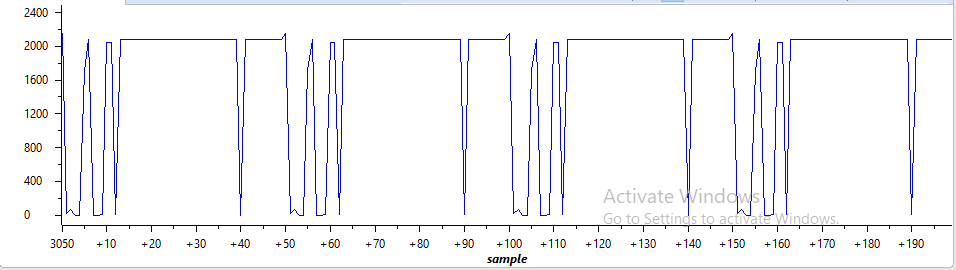
else{GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0x00);}

}

}



Average ADC Values



**Youtube Link:** <https://youtu.be/O9LgRkr2wvU>

**Task 2**

#include <stdint.h>

#include <stdbool.h>

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "inc/tm4c123gh6pm.h"

#include "driverlib/debug.h"

#include "driverlib/sysctl.h"

#include "driverlib/adc.h"

#define TARGET\_IS\_BLIZZARD\_RB1

#include "driverlib/rom.h"

#include "driverlib/gpio.h"

#include "driverlib/timer.h"

#include "driverlib/interrupt.h"

/\*

#ifdef DEBUG

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

#endif

\*/

//my variables

uint32\_t ui32ADC0Value[4];

volatile uint32\_t ui32TempAvg;

volatile uint32\_t ui32TempValueC;

volatile uint32\_t ui32TempValueF;

int main(void)

{

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

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

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

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 32);

//step up adc sequencer 2

ROM\_ADCSequenceConfigure(ADC0\_BASE, 1, ADC\_TRIGGER\_PROCESSOR, 0);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 0, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 1, ADC\_CTL\_TS);

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

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 3, ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

ROM\_ADCSequenceEnable(ADC0\_BASE, 1);

//enable timer1

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER1);

//config timer 1 for overflow every half second

ROM\_TimerConfigure(TIMER1\_BASE, TIMER\_CFG\_PERIODIC);

ROM\_TimerLoadSet(TIMER1\_BASE, TIMER\_A, (SysCtlClockGet()/2)-1);

//enable interrupt and timer 1

ROM\_TimerEnable(TIMER1\_BASE, TIMER\_A);

ROM\_IntEnable(INT\_TIMER1A);

ROM\_TimerIntEnable(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

//enable masters

IntMasterEnable();

while(1){}

}

void Timer1IntHandler(void)

{

//clear timer interrupt

ROM\_TimerIntClear(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

//clear adc interrupt and set processor trig to 2

ROM\_ADCIntClear(ADC0\_BASE, 1);

ROM\_ADCProcessorTrigger(ADC0\_BASE, 1);

// continue to loop while calculations are taking place

while(!ROM\_ADCIntStatus(ADC0\_BASE, 1, false)){}

// here we grab all the adc data and calculate the C and F

ROM\_ADCSequenceDataGet(ADC0\_BASE, 1, ui32ADC0Value);

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

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

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

// i picked 62 because I could only get the temp up to 64 but

// still wanted to show LED turn on and it always initialized at 62

if(ui32TempValueF > 64){GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 4);}

else{GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0);}

}

**Youtube Link:** <https://youtu.be/7NS4VPf-Lxc>