**Task 00: Execute supplied code**

//

/\*

\* Melissa Cordova

\* CPE 403 - LAB 5

\*

\* main.c

\*/

**#include** <stdint.h> //variable deifinitions for the C99 standard

**#include** <stdbool.h> //boolean definitions for the C99 standard

**#include** "inc/hw\_memmap.h" //macros defining the memory map of the TivaC

**#include** "inc/hw\_types.h" //define common types and macros

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

**#include** "driverlib/sysctl.h" //defines and macros for System Control API

**#include** "driverlib/adc.h" //definitions for using the ADC driver

**#define** TARGET\_IS\_BLIZZARD\_RB1

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

**#ifdef** DEBUG

**void\_\_error\_\_**(**char** \*pcFilename, uint32\_t u132lLine)

{

}

**#endif**

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

{

uint32\_t ui32ADC0Value[4]; //create an array to store data from ADC FIFO

//with depth of 4

**volatile** uint32\_t ui32TempAvg; //sorts the average of the temp

**volatile** uint32\_t ui32TempValueC; //store the Celsius temp

**volatile** uint32\_t ui32TempValueF; //store the Fahrenheit temp

//system clock to run at 40MHz

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

//enable the ADC0 peripheral

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 64);

//ADC sequencer

**ADCSequenceConfigure**(ADC0\_BASE, 1, ADC\_TRIGGER\_PROCESSOR, 0);

//Configuring steps 0-2 on sequencer 1 to sample temp sensor

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);

//Sample temp sensor and configure interrupt flag to set

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

//enable ADC

ROM\_ADCSequenceEnable(ADC0\_BASE, 1);

**while**(1)

{

//ADC conversion

ROM\_ADCIntClear(ADC0\_BASE, 1);

//ADC conversion with software

ROM\_ADCProcessorTrigger(ADC0\_BASE, 1);

**while**(!**ADCIntStatus**(ADC0\_BASE, 1, false))

{

}

//ADC value from ADC Sampler Sequence

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

//Average of temp sensor data

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

//calculate celsius value of temp

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

//calculate Fahrenheit temp

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

}

}

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

**#include** <stdint.h> //variable deifinitions for the C99 standard

**#include** <stdbool.h> //boolean definitions for the C99 standard

**#include** "inc/hw\_memmap.h" //macros defining the memory map of the TivaC

**#include** "inc/hw\_types.h" //define common types and macros

**#include** "driverlib/sysctl.h" //defines and macros for System Control API

**#include** "driverlib/adc.h" //definitions for using the ADC driver

**#include** "driverlib/gpio.h" //defines macros for GPIO

**#include** "driverlib/interrupt.h" //defines and macros for NVIC

**#include** "driverlib/timer.h" //defines and macros for Timer API

**#define** TARGET\_IS\_BLIZZARD\_RB1

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

**.**

**.**

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

//enable the ADC0 peripheral

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

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

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 64);

//ADC sequencer

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

//Configuring steps 0-2 on sequencer 2 to sample temp sensor

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);

//Sample temp sensor and configure interrupt flag to set

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

//enable ADC

ROM\_ADCSequenceEnable(ADC0\_BASE, 2);

**while**(1)

{

//ADC conversion

ROM\_ADCIntClear(ADC0\_BASE, 2);

//ADC conversion with software

ROM\_ADCProcessorTrigger(ADC0\_BASE, 2);

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

{

}

//ADC value from ADC Sampler Sequence

ROM\_ADCSequenceDataGet(ADC0\_BASE, 2, ui32ADC0Value);

.

.

.

**if**(ui32TempValueF > 75) //if temp is > 75 degrees F

{

//Turn on LED at PF1 because could not get it to turn on PF3

ROM\_GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 2);

}

**else**

{

//Turn off LED

ROM\_GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 0);

}

}

}

**Task 02: Introduce hardware averaging to 32. Using the timer TIMER1A conduct an ADC conversion on overflow every 0.5sex. Use the TIMER1A interrupt.**

**#include** <stdint.h> //variable deifinitions for the C99 standard

**#include** <stdbool.h> //boolean definitions for the C99 standard

**#include** "inc/hw\_memmap.h" //macros defining the memory map of the TivaC

**#include** "inc/tm4c123gh6pm.h"

**#include** "inc/hw\_types.h" //define common types and macros

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

**#include** "driverlib/sysctl.h" //defines and macros for System Control API

**#include** "driverlib/adc.h" //definitions for using the ADC driver

**#include** "driverlib/gpio.h" //defines macros for GPIO

**#include** "driverlib/interrupt.h" //defines and macros for NVIC

**#include** "driverlib/timer.h" //defines and macros for Timer API

**#define** TARGET\_IS\_BLIZZARD\_RB1

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

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

{

uint32\_t ui32Period;

.

.

.

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

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

//calculate and set delay

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

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

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

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

.

.

.

//enable interrupts

**IntEnable**(INT\_TIMER1A);

//set timer 1 to interrupt

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

**IntMasterEnable**();

//enable timer

**TimerEnable**(TIMER1\_BASE, TIMER\_A);

**while**(1)

{

}

}

**void** **IntTimer1Handler**(**void**)

{

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

//ADC conversion

**.**

**.**

**.**