(Highlighting: Task 01(no highlighting), Task 02, Task 03)

Task 01: Submit a comprehensive commented file of the original code

```
/*Shabrya Lott
 * Tiva_c Lab05
 * Usage:This is a simple program that calculates the avg temp of the on board sensor
 */
#include <stdint.h>                              //variable definitions for the C99 standard
#include <stdbool.h>                        //Boolean definitions for the C99 standard
#include "inc/hw_memmap.h"                  //macros defining the memory map of Tiva C Series
#include "inc/hw_types.h"             //defines common types and macros
#include "driverlib/debug.h"          //Macros for assisting debug of the driver library
#include "driverlib/sysctl.h"         //defines macros for System Control API of Driverlib
#include "driverlib/adc.h"            //definitions for using the ADC driver
#define TARGET_IS_BLIZZARD_RB1
#include "driverlib/rom.h"            //Macros to facilitate calling functions in the ROM

#ifdef DEBUG
void__error__(char *pcFilename, uint32_t u132Line)
{
}
#endif

int main(void)
{
        //will store 4 values from FIFO when using the sequencer 1
        uint32_t ui32ADC0Value[4];
        //stores average of 4 sampled values
        volatile uint32_t ui32TempAvg;
        //stores temperture in Celsius
        volatile uint32_t ui32TempValueC;
        //stores temperture in Fahrenheit
        volatile uint32_t ui32TempValueF;

        //set cock to 40MHz
        ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTA
L_16MHZ);

        //configure step 0 from temp sensor
        ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);

        //each sample in the ADC FIFO will be the result of 64 measurements being averaged together
        ROM_ADCHardwareOversampleConfigure(ADC0_BASE, 64);

        //Enable ADC0, sample sequencer 1, trigger with processor
        ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
        //configure ADC0, sequencer 1, step 0
        ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_TS);
        //configure ADC0, sequencer 1, step 1
        ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_TS);
        //configure ADC0, sequencer 1, step 2
        ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_TS);
        //configure ADC0, sequencer 1, step 3 and tell sequencer to finish
        ROM_ADCSequenceStepConfigure(ADC0_BASE,1,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
        //enable ADC0, sequencer 1
        ROM_ADCSequenceEnable(ADC0_BASE, 1);

        while(1) //infinite loop
        {
                //clear interrupt flag on ADC0, sequencer 1
                ROM_ADCIntClear(ADC0_BASE, 1);
                //Trigger ADC0 sequencer 1
                ROM_ADCProcessorTrigger(ADC0_BASE, 1);

                //wait for ADC conversion to finish
         while(!ADCIntStatus(ADC0_BASE, 1, false))
         {
         }
         //get data from FIFO and out into array
         ROM_ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
         //calculate avg temp (+2/4 used for rounding)
         ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3]
+ 2)/4;
         //calculate temp in Celsius
         ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
         //calculate temp in Fahrenheit
         ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;
        }
}
```

Task 02: Change the ADC sequencer to SS2. Turn on the LED at PF1 if the temperature than 79degF

.
.
.

```
int main(void)
{       .
        .
        .

        while(1) //infinity loop
        {       .
                .
                .

         if(ui32TempValueF > 79) //if temp is > 79 degrees F
                 //turn on PF1 (Red LED)
                 ROM_GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 2);
         else
                 //turn off PF1 (Red LED)
                 ROM_GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0);
        }
}
```

Task 03: Introduce hardware averaging to 64 by using TIMER0A to conduct an ADC conversion on overflow every 0.333 sec. Use TIMER0A interrupt.

```
.
.
.
int main(void)
{
	uint32_t ui32Period; //will be used for time delay
	//set cock to 40MHz
	SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

	//enable port F
	SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
	//configure in F1 as output
	GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1);
	//configure step 0 from temp sensor
	SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
	//enable timer 0 for interrupts
	SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);

	//configure timer 0 periodic mode
	TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
	//get clock and divide by 3 for 33% DC (0.333 sec)
	ui32Period = SysCtlClockGet()/ 3;
	//set period for Timer0A (delay)
	TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period - 1);

	//each sample in the ADC FIFO will be the result of 64 measurements being averaged together
	ADCHardwareOversampleConfigure(ADC0_BASE, 64);
	//configure PF1 as outpu
	GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1);
	//Enable ADC0, sample sequencer 2, trigger with processor
	ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);
	//configure ADC0, sequencer 2, step 0
	ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
	//configure ADC0, sequencer 2, step 1
	ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
	//configure ADC0, sequencer 2, step 2
	ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);
	//configure ADC0, sequencer 2, step 3 and tell sequencer to finish
	ADCSequenceStepConfigure(ADC0_BASE,2,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
	//enable ADC0, sequencer 2
	ADCSequenceEnable(ADC0_BASE, 2);

	//Enable interrupts on timer 0
	IntEnable(INT_TIMER0A);
	//set timer 0 to interrupt at timeout
	TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
	//enable master interrupt
	IntMasterEnable();
	//start the timer
	TimerEnable(TIMER0_BASE, TIMER_A);

	while(1) //infinite loop
	{
        }
}
void Timer0IntHandler(void)
{
        //clear interrupt flag on timer 0
        TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
        //clear interrupt flag on ADC0, sequencer 2
        ADCIntClear(ADC0_BASE, 2);
        //Trigger ADC0, sequencer 2
        ADCProcessorTrigger(ADC0_BASE, 2);

        //wait for ADC converion to finish
        while(!ADCIntStatus(ADC0_BASE, 2, false))
        {
        }
        //get data from FIFO and out into array
        ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value);
        //calculate avg temp (+2/4 used for rounding)
        ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] +
2)/4;
        //calculate temp in Celsius
        ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
        //calculate temp in Fahrenheit
        ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;

        if(ui32TempValueF > 79) //if temp is > 79 degrees F
                //turn on PF1 (Red LED)
                GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 2);
        else
                //turn off PF1 (Red LED)
                GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0);
}
```