Tenniel Takenaka
Github root directory: https://github.com/TennielTakenaka/CLASS2/tree/master/lab5

**Date Submitted:** 10/1/19

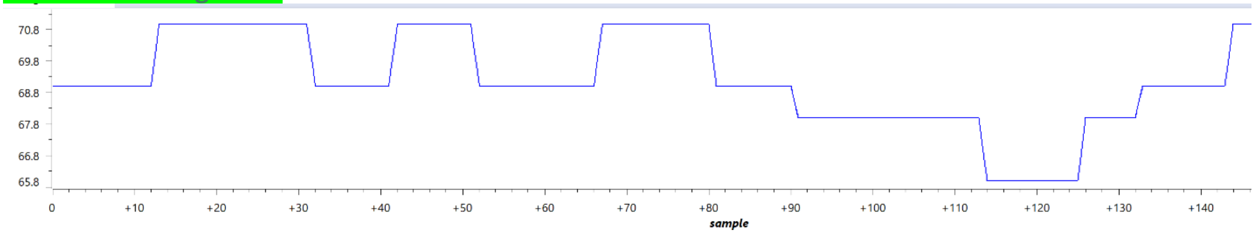
**Task 00: Execute provided code**

Youtube Link: https://youtu.be/JFLimvrrcP4

-----------------------------------------------------------------------------------

## Task 01:

Youtube Link: https://youtu.be/PBqi6_wUOPE
NOTE: I USE 70 DEGREES AS THE LED TURN ON THRESHOLD BC I HAD A HARDER TIME TRYING TO GET MY DEVICE TO BECOME & STAY HOT AT 75 DEGREES. The demonstration works just as well as 70 degrees.



Modified Code:

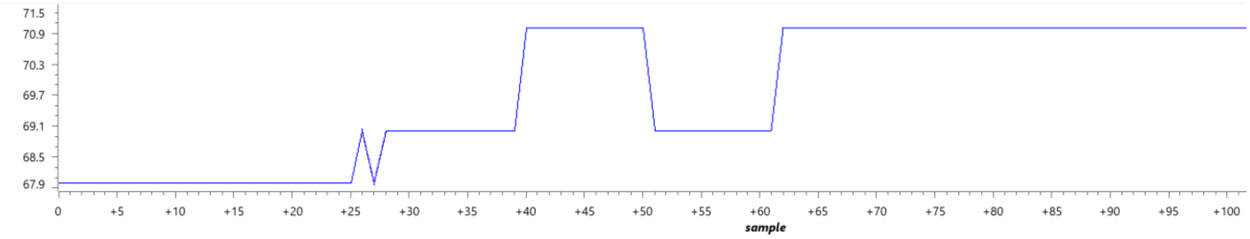
```
#include<stdint.h>
#include<stdbool.h>
#include"inc/hw_memmap.h"
#include"inc/hw_types.h"
#include"driverlib/debug.h"
#include"driverlib/sysctl.h"
#include"driverlib/adc.h"
#include "driverlib/gpio.h"//needed for the gpio led pins
#define TARGET_IS_BLIZZARD_RB1
#include "driverlib/rom.h"

#ifdef DEBUG
void__error__(char*pcFilename, uint32_t ui32Line)
{
}
#endif

int main()
{
    uint32_t ui32ADC0Value[4];
    volatile uint32_t ui32TempAvg;
    volatile uint32_t ui32TempValueC;
    volatile uint32_t ui32TempValueF;
    bool isOn;

    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); //peripherals for LEDs enabled
```

**Grading scheme:** 30% Coding, 30% Documentation, 40% Execution/Video.

```
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
//enable the LEDS for output

    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ADCHardwareOversampleConfigure(ADC0_BASE, 64); //64 measurements averaged for
sample. stops value from switching around too much

    ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0); //SAMPLE SEQUENCER
2 ENABLED
    ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS); //internal temperature
sensor, step determines the order the sample is collected when trigger occurs
    ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS); //step 1 configure for TS
    ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS); //step 2 configure for TS
    ADCSequenceStepConfigure(ADC0_BASE, 2, 3, ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
//step 3configure for TS, interrupt enable, or end flag
//tell adc logic there is the last conversion when these flags pop up
    ADCSequenceEnable(ADC0_BASE, 2);

    while(1) //read temp sensor and calculate the temp endlessly
    {
        ADCIntClear(ADC0_BASE, 2); //clear adc conversion done flag before writing
code that depends on it. change to sequence 2
        ADCProcessorTrigger(ADC0_BASE, 2); //config processor trigger for step 2

        while(!ADCIntStatus(ADC0_BASE, 2, false)) //wait for conversion to finish
        {
        } //if loop exited conversion is complete

        ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value); //gets samples from the
array
        ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
ui32ADC0Value[3] + 2)/4;
        ui32TempValueC = (1475 -((2475 * ui32TempAvg)) / 4096)/10;
        ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;

        if (ui32TempValueF < 70) { //if tempval < 70 turn on red LED
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 2); // Turn on the red LED
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0); // Turn off the LED
        }
        else if (ui32TempValueF > 70) { //if tempval > 70, turn on the blue LED
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4); // Turn on the blue LED
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0); // Turn off the LED
        }
    }

}
-----------------------------------------------------------------------------------
```

## Task 02:

Youtube Link: https://youtu.be/f-hfKR6rxZo



The youtube video can show that the speed of data updating is every 0.5 seconds.

```
Modified Code:
#include<stdint.h>
#include<stdbool.h>
#include "inc/tm4c123gh6pm.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"driverlib/interrupt.h" //needed for interrupt functions
#include "driverlib/timer.h" //needed for timer functions
#define TARGET_IS_BLIZZARD_RB1
#include"driverlib/rom.h"

#ifdef DEBUG
void__error__(char*pcFilename, uint32_t ui32Line)
{
}
#endif

uint32_t ui32ADC0Value[1]; //change to an array of 1 because sequence 0 only has 1
step
volatile uint32_t ui32TempAvg;
volatile uint32_t ui32TempValueC;
volatile uint32_t ui32TempValueF;

int main()
{
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); //peripherals for LEDs enabled
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ADCHardwareOversampleConfigure(ADC0_BASE, 32); //32 measurements averaged for
sample. stops value from switching around too much

    ADCSequenceConfigure(ADC0_BASE, 3, ADC_TRIGGER_PROCESSOR, 0); //SAMPLE SEQUENCER
3 ENABLED
    ADCSequenceStepConfigure(ADC0_BASE, 3, 0, ADC_CTL_TS | ADC_CTL_IE | ADC_CTL_END);
//enable the step 0 for sequence 3 to configure to sample either the temperature
sensor, cause an interrupt when step is complete, or when there is an end flag

    //Timer 1 Configure
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);
    TimerLoadSet(TIMER1_BASE, TIMER_A, (SysCtlClockGet() * .5));
    IntEnable(INT_TIMER1A);
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
    IntMasterEnable();
    TimerEnable(TIMER1_BASE,TIMER_A);

    ADCSequenceEnable(ADC0_BASE, 3); //enable sequence 3
    ADCIntEnable(ADC0_BASE, 3); //enable interrupt for sequence 3

    while(1) //read temp sensor and calculate the temp endlessly
    {
    }
}

void Timer1IntHandler(void) //created timer1 interrupt handler
{
    ADCIntClear(ADC0_BASE, 3); //clear adc conversion done flag before writing code
that depends on it. change to sequence 2
    TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT); //clear timer

    TimerLoadSet(TIMER1_BASE, TIMER_A, (SysCtlClockGet() * .5)); //timer loaded to .5
seconds
    ADCProcessorTrigger(ADC0_BASE, 3); //changed to sequence 3

    while(!ADCIntStatus(ADC0_BASE, 3, false)) //wait for conversion to finish
    {
    } //if loop exited conversion is complete

    ADCSequenceDataGet(ADC0_BASE, 3, ui32ADC0Value); //gets samples from the array
    ui32TempValueC = (1475 -((2475 * ui32ADC0Value[0])) / 4096)/10; //only one adcval
    ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;

    if (ui32TempValueF < 70) {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 2); // Turn on the red LED
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0); // Turn off the LED
    }
    else if (ui32TempValueF > 70) {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4); // Turn on the blue LED
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0); // Turn off the LED
    }
}

------------------------------------------------------------------------------------
```

**Grading scheme:** 30% Coding, 30% Documentation, 40% Execution/Video.