Serak Gebremedhin
Github root directory: https://github.com/Ber-geb/solid-octo-tribble.git

**Date Submitted:**

**Task 00: Execute provided code**

**Youtube Link:**

https://youtu.be/gKwfyXiwNq4

---

## Task 01:

Youtube Link:

https://youtu.be/gKwfyXiwNq4

**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"
#define TARGET_IS_BLIZZARD_RB1
#include "driverlib/rom.h"

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

int main(void)
{
    uint32_t ui32ADC0Value[4];

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    volatile uint32_t ui32TempAvg;
    volatile uint32_t ui32TempValueC;
    volatile uint32_t ui32TempValueF;


ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ROM_ADCHardwareOversampleConfigure(ADC0_BASE, 64);
    ROM_ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
```

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

```
    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);

    while(1)
    {
        ROM_ADCIntClear(ADC0_BASE, 1);
        ROM_ADCProcessorTrigger(ADC0_BASE, 1);

        while(!ROM_ADCIntStatus(ADC0_BASE, 1, false))
        {
        }

        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;

        if (ui32TempValueF > 72){
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
        }
        else{
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
        }
    }
}
```

---

## Task 02:

Youtube Link:

https://youtu.be/gKwfyXiwNq4

**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"
#define TARGET_IS_BLIZZARD_RB1
#include "driverlib/rom.h"
#include "driverlib/timer.h"
```

```
#include "driverlib/interrupt.h"
#include "inc/tm4c123gh6pm.h"

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


void configureTimer1A()
{
    uint32_t ui32Period;
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1); //Enable Timer 1 Clock
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC); //configure timer operation as
periodic
    //Configure timer frequency
    //Frequency is given by MasterClock / CustomValue
    ui32Period = (SysCtlClockGet() / 1) * 0.5; // 1Hz with 50% duty cycle
    TimerLoadSet(TIMER1_BASE, TIMER_A, ui32Period -1);

    IntEnable(INT_TIMER1A); //Enable timer 1a interrupt
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT); //timer 1a interrupt when
timeout
    IntMasterEnable(); //Enable Interrupts
    TimerEnable(TIMER1_BASE, TIMER_A); //Start Timer 1a
}


int main(void)
{
   // uint32_t ui32ADC0Value[4];

    configureTimer1A();

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
/*
    volatile uint32_t ui32TempAvg;
    volatile uint32_t ui32TempValueC;
    volatile uint32_t ui32TempValueF;
*/

ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ROM_ADCHardwareOversampleConfigure(ADC0_BASE, 32); //number of samples averaged
set to 32
    ROM_ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);

    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_TS);//Sequencer Step 0:
Samples Channel PE3, SS1 (sample sequencer 1)
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_TS);//Sequencer Step 1:
Samples Channel PE2 SS1
```

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

```
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_TS);//Sequencer Step 3:
Samples Channel PE1 SS1

    ROM_ADCSequenceStepConfigure(ADC0_BASE,1,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
//Sequencer Step 4: Samples Channel PE0 SS1

    ROM_ADCSequenceEnable(ADC0_BASE, 1);

    while(1)
    {
        /*
        ROM_ADCIntClear(ADC0_BASE, 1);
        ROM_ADCProcessorTrigger(ADC0_BASE, 1);

        while(!ROM_ADCIntStatus(ADC0_BASE, 1, false))
        {
        }

        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;

        if (ui32TempValueF > 72){
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
        }
        else{
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
        }
        */
    }
}


void Timer1AIntHandler(void){
    uint32_t ui32ADC0Value[4];

    volatile uint32_t ui32TempAvg;
    volatile uint32_t ui32TempValueC;
    volatile uint32_t ui32TempValueF;

    //Required to launch next interrupt

    TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
    //TimerIntClear(TIMER1_BASE, TIMER_A);

    // Read the current state of the GPIO pin and
    // write back the opposite state
    ROM_ADCIntClear(ADC0_BASE, 1);
    ROM_ADCProcessorTrigger(ADC0_BASE, 1);

    while(!ROM_ADCIntStatus(ADC0_BASE, 1, false))
    {
    }
```

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

```
    ROM_ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
    ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
ui32ADC0Value[3] + 2)/4; //temp avg value
    ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10; //temp value C
    ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5; //temp value F

    if (ui32TempValueF > 72){ //if temperature passes 72 F, led will turn on,
otherwise will stay off
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
    }
    else{
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
    }
}
```

---

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