Geovanni Portillo
Github root directory: https://github.com/portig1/submissions_E2

Date Submitted: **September 27, 2019**

Task 00: Execute provided code

Youtube Link: https://youtu.be/pBW98WdJ8ZE

Task 01:

Youtube Link: https://youtu.be/1hP1qyyGREA
Modified Schematic (if applicable):

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

//#ifdef DEBUG
//void__error__(char *pcFilename, uint32_t ui32lLine)
//{
//}
//#endif

int main(void)
{
    uint32_t ui32ADC0Value[4]; //ADC Sequencer SS2 has FIFO depth of 4 so
we need an array of the same size
    volatile uint32_t ui32TempAvg;
    volatile uint32_t ui32TempValueC;
    volatile uint32_t ui32TempValueF;

    //LED Config
    uint8_t ui8LED1 = 2;
    uint8_t ui8LED2 = 4;
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    SysCtlDelay(3);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,
GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_
16MHZ);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ADCHardwareOversampleConfigure(ADC0_BASE, 64);

    ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0); //We
want to use ADC0, sampler sequencer 2, we want the processor to trigger
the sequence and we want the highest priority
    ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 3,
ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
    ADCSequenceEnable(ADC0_BASE, 2);

    while(1)
    {
        ADCIntClear(ADC0_BASE, 2);
        ADCProcessorTrigger(ADC0_BASE, 2);

       while(!ADCIntStatus(ADC0_BASE, 2, false))
       {
       }

       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;
       if(ui32TempValueF > 71) {
           GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,
ui8LED2);
       }
       else {
           GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,
ui8LED1);
       }
    }
}
```

Task 02:

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

Modified Schematic (if applicable):

Modified Code:

```
#include <stdint.h>
```

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

```
#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"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
#include "driverlib/interrupt.h"


int main(void)
{
    uint32_t ui32Period;




SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_
16MHZ);

    //LED Config
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    SysCtlDelay(3);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,
GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

     //ADC Config
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ADCHardwareOversampleConfigure(ADC0_BASE, 32); //Hardware averages 32

    ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0); //We
want to use ADC0, sampler sequencer 2, we want the processor to trigger
the sequence and we want the highest priority
    ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 3,
ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
    ADCSequenceEnable(ADC0_BASE, 2);

     //Timer1A Config
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);

    ui32Period = (SysCtlClockGet() / 1) / 2; //To get period of 0.5s,
divide system clock for 1s period and divide by 2 for 0.5s
    TimerLoadSet(TIMER1_BASE, TIMER_A, ui32Period -1);

    IntEnable(INT_TIMER1A);
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
    IntMasterEnable();
    TimerEnable(TIMER1_BASE, TIMER_A);

    while(1)
    {
    }
}

void Timer1IntHandler(void)
{
    uint32_t ui32ADC0Value[4]; //ADC Sequencer SS2 has FIFO depth of 4 so 
we need an array of the same size
    volatile uint32_t ui32TempAvg;
    volatile uint32_t ui32TempValueC;
    volatile uint32_t ui32TempValueF;

    // Clear the timer interrupt
    TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);

    ADCIntClear(ADC0_BASE, 2);
    ADCProcessorTrigger(ADC0_BASE, 2);

   while(!ADCIntStatus(ADC0_BASE, 2, false))
   {
   }

   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;

   if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
   {
       GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
   }
   else
   {
       GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
   }
}


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

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