Ameera Essaqi
Github root directory: https://github.com/AmeeraE/microcontrollers/tree/master/TIVAC

Date Submitted: 11/30/19

Task 00: Execute provided code

Youtube Link: **https://youtu.be/Zyh6cONgi8Q**

```
#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"

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

          SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

          SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);

          ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
          ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_TS);
          ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_TS);
          ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_TS);
          ADCSequenceStepConfigure(ADC0_BASE,1,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
          ADCSequenceEnable(ADC0_BASE, 1);

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

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

             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;
          }
}
```

Task 01:

Youtube Link: https://youtu.be/2l7GTxZR5cs

Modified Code:

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
```

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

```
#include "inc/hw_types.h"
#include "driverlib/debug.h"
#include "driverlib/sysctl.h"
#include "driverlib/adc.h"
#include "driverlib/gpio.h"

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

    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);

    //ADCHardwareOversampleConfigure(ADC0_BASE, 64);

    ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);


    ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_TS);


    ADCSequenceStepConfigure(ADC0_BASE,1,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);

    ADCSequenceEnable(ADC0_BASE, 1);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_2);

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

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

       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;

       // Turn on the LED at PF2 if the temperature is greater than 72 degF.
       if(ui32TempValueF > 72) {GPIOPinWrite (GPIO_PORTF_BASE,GPIO_PIN_2,4); } // 4 = BLUE_LED
       else {GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_2,0);} // Keep LED off

    }
}
```

Task 02:

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

Modified Code:

Ameera Essaqi
Github root directory: https://github.com/AmeeraE/microcontrollers/tree/master/TIVAC

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/tm4c123gh6pm.h"
#include "driverlib/interrupt.h"
#include "driverlib/timer.h"
#include "driverlib/debug.h"
#include "driverlib/sysctl.h"
#include "driverlib/adc.h"
#include "driverlib/gpio.h"


uint32_t tPeriod;
uint32_t ui32ADC0Value[4];
volatile uint32_t ui32TempAvg;
volatile uint32_t ui32TempValueC;
volatile uint32_t ui32TempValueF;

int main(void)
{

    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ); //
system clock run at 40MHz
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);// enable the ADC0 peripheral

    ADCHardwareOversampleConfigure(ADC0_BASE, 32); // hardware averaging


    ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);

    ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_TS);

    ADCSequenceStepConfigure(ADC0_BASE,1,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);

    ADCSequenceEnable(ADC0_BASE, 1);// enable ADC sequencer 1
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_2); // Enable PF2

    tPeriod = SysCtlClockGet()/2;
    configTimer1A();
    IntMasterEnable();
    ADCIntEnable(ADC0_BASE,2);
    while(1)
    {
    }
}

void configTimer1A()
{

    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);
    TimerLoadSet(TIMER1_BASE, TIMER_A, tPeriod-1);  // counts up to sec_delay
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
    IntEnable(INT_TIMER1A);
    TimerEnable(TIMER1_BASE, TIMER_A);
}
```

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

```
Timer1IntHandler(void)
{
    TimerIntClear(TIMER1_BASE, TIMER_A);

    ADCIntClear(ADC0_BASE, 1);
    ADCProcessorTrigger(ADC0_BASE, 1);
    while(!ADCIntStatus(ADC0_BASE, 1, false))
    {
    }

    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); } // 4 = BLUE_LED
    else {GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_2,0);} // Keep LED off

}
```