Ricky Perez
Github root directory: https://github.com/RickyPerez79/da_submissions_2019

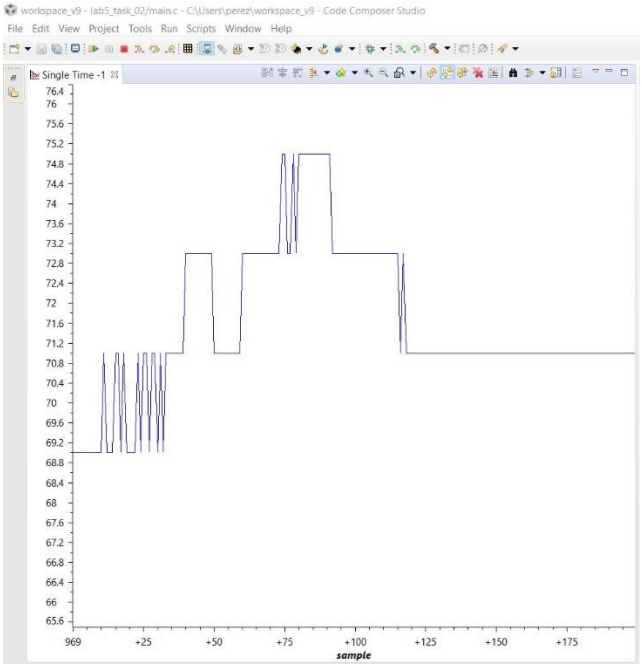
Date Submitted: **10/14/2019**

Task 00: Execute provided code

Youtube Link: **https://youtu.be/XwNNn6mg5a4**

Task 01:

Youtube Link: https://youtu.be/kcFvW51P-y0



Modified Code:

```
// Task01
#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"

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

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

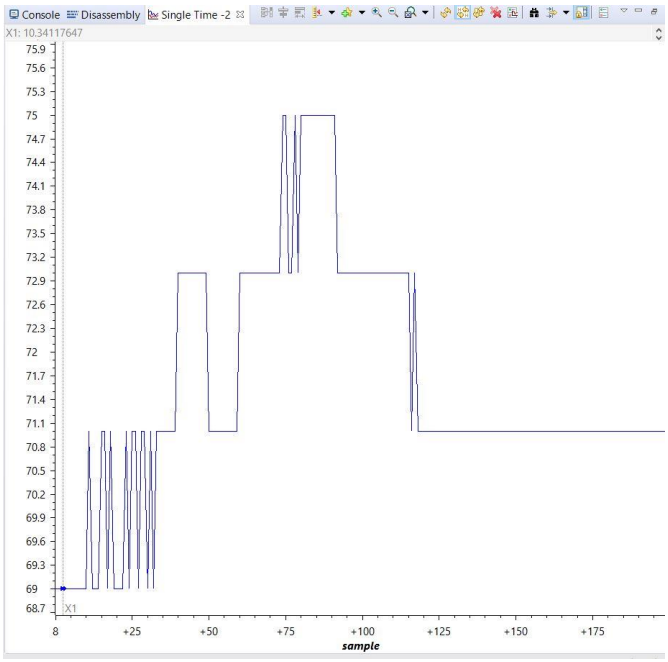
```
    volatile uint32_t ui32TempValueF;
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
// system clock run at 40MHz
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);// enable the ADC0 peripheral

   // ADCHardwareOversampleConfigure(ADC0_BASE, 64); // hardware averaging

    //configure the ADC sequencer.
    //We want to use ADC0
    //sample sequencer 1
    // we want the processor to trigger the sequence and we want to use the highest
priority.
    ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);

    // Configure all four steps in the ADC sequencer
    // Configure steps 0 - 2 on sequencer 1 to sample the temperature sensor
(ADC_CTL_TS)

    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);

    // The final sequencer step requires a couple of extra settings.
    // Sample the temperature sensor (ADC_CTL_TS) and configure the interrupt flag
(ADC_CTL_IE) to be set when the sample is done.
    // Tell the ADC logic that this is the last conversion on sequencer 1
(ADC_CTL_END).

    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

    while(1)
    {
        // We're going to read the value of the temperature sensor and calculate the
temperature endlessly.

        // The indication that the ADC conversion process is complete will be the ADC
interrupt status flag.
        ADCIntClear(ADC0_BASE, 1);
        // Trigger the ADC conversion with software
        ADCProcessorTrigger(ADC0_BASE, 1);
        // Wait for the conversion to complete.
        while(!ADCIntStatus(ADC0_BASE, 1, false))
        {
        }
        // When code execution exits the loop in the previous step,
        // we know that the conversion is complete and that we can read the ADC value
from the ADC Sample Sequencer 1 FIFO.
        // The function we'll be using copies data from the specified sample
sequencer output FIFO to a buffer in memory.
```

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

```
        // The number of samples available in the hardware FIFO are copied into the
buffer, which must be large enough to hold that many samples.
        // This will only return the samples that are presently available, which
might not be the entire sample sequence if you attempt to access the FIFO before the
conversion is complete.

        ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
        //  Calculate the average of the temperature sensor data.

        ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
ui32ADC0Value[3] + 2)/4;

        // Calculate the Celsius value of the temperature.
        // TEMP = 147.5 - ((75 * (VREFP - VREFN) * ADCVALUE) / 4096)
        ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;

        // The conversion from Celsius to Fahrenheit is F = ( C * 9)/5 +32.
        // Adjusting that a little gives: F = ((C * 9) + 160) / 5

        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/-tR0xtG9RE8



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

Modified Code:

```
// Task02
#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

    //configure the ADC sequencer.
    //We want to use ADC0
    //sample sequencer 1
    // we want the processor to trigger the sequence and we want to use the highest
priority.
    ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);

    // Configure all four steps in the ADC sequencer
    // Configure steps 0 - 2 on sequencer 1 to sample the temperature sensor
(ADC_CTL_TS)

    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);

    // The final sequencer step requires a couple of extra settings.
    // Sample the temperature sensor (ADC_CTL_TS) and configure the interrupt flag
(ADC_CTL_IE) to be set when the sample is done.
    // Tell the ADC logic that this is the last conversion on sequencer 1
(ADC_CTL_END).

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

    ADCSequenceEnable(ADC0_BASE, 1);// enable ADC sequencer 1
```

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

```
    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);
}

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

    // The indication that the ADC conversion process is complete will be the ADC
interrupt status flag.
    ADCIntClear(ADC0_BASE, 1);
    // Trigger the ADC conversion with software
    ADCProcessorTrigger(ADC0_BASE, 1);
    // Wait for the conversion to complete.
    while(!ADCIntStatus(ADC0_BASE, 1, false))
    {
    }
    // When code execution exits the loop in the previous step,
    // we know that the conversion is complete and that we can read the ADC value
from the ADC Sample Sequencer 1 FIFO.
    // The function we'll be using copies data from the specified sample sequencer
output FIFO to a buffer in memory.
    // The number of samples available in the hardware FIFO are copied into the
buffer, which must be large enough to hold that many samples.
    // This will only return the samples that are presently available, which might
not be the entire sample sequence if you attempt to access the FIFO before the
conversion is complete.

    ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
    //  Calculate the average of the temperature sensor data.

    ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
ui32ADC0Value[3] + 2)/4;

    // Calculate the Celsius value of the temperature.
```

```
    // TEMP = 147.5 – ((75 * (VREFP – VREFN) * ADCVALUE) / 4096)
    ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;

    // The conversion from Celsius to Fahrenheit is F = ( C * 9)/5 +32.
    // Adjusting that a little gives: F = ((C * 9) + 160) / 5

    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

}

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

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