**Date Submitted: 10/9/2018 8:04 PM**

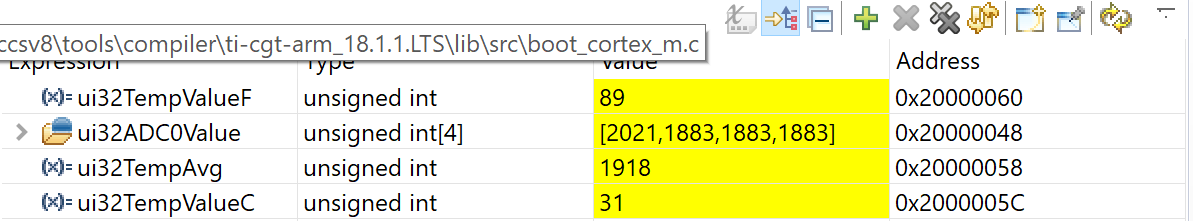
**Task 00: Execute provided code**

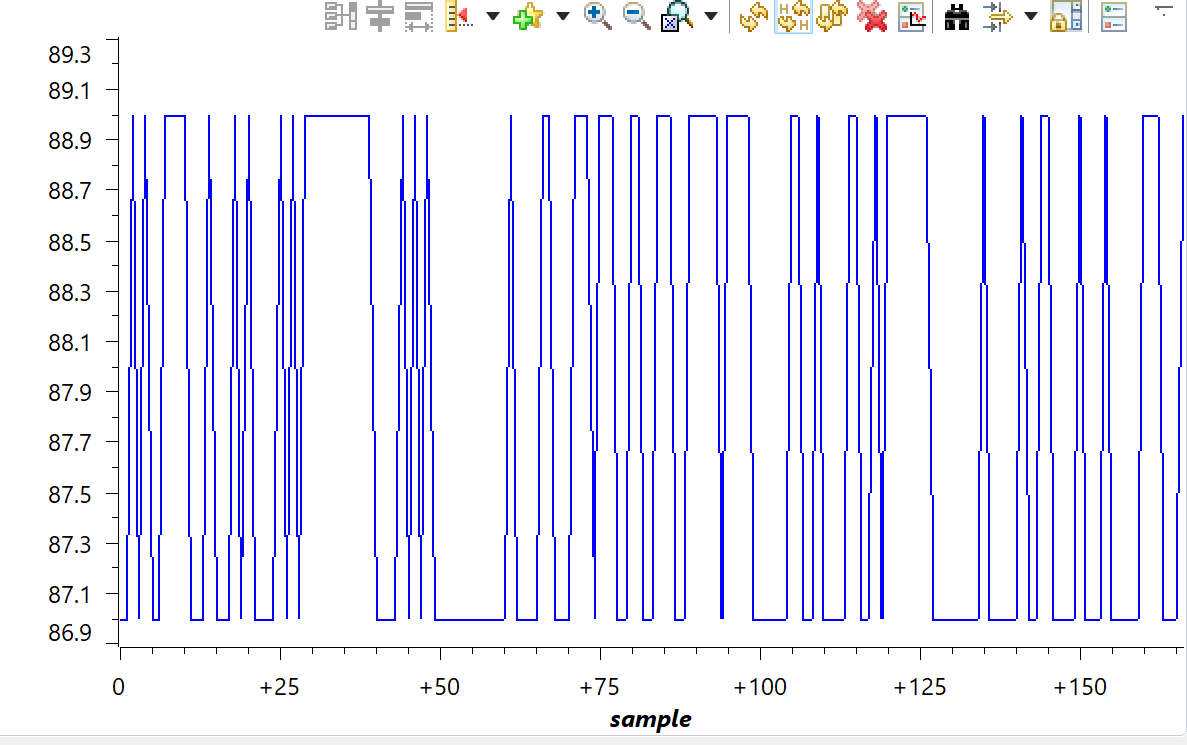
**Youtube Link: No Submission required for Task 00**

**------------------------------------------------------------------------------------**

**Task 01:** Change the ADC Sequencer to SS3. Turn on the LED at PF2 if the temperature is greater that 72 degF. Use internal temperature sensor for all SS2 sequence. Display the temperature in the built-in graph tool.

Youtube Link: <https://www.youtube.com/watch?v=dpeaDlF4u4o>





Above is the live graph used to measure the temperature value (ui32TempValueF). This program implements ADC conversion. The sample sequencer used was SS3. When the temperature is read, if the temperature is larger than 72 degrees an LED at PF2 (Blue) will turn on.

**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"

**#ifdef** DEBUG

**void\_\_error\_\_**(**char** \*pcFilename, uint32\_t ui32Line)

{

}

**#endif**

**int** **main**(**void**)

{

uint32\_t ui32ADC0Value[4]; //array for storing the data read

**volatile** uint32\_t ui32TempAvg; //avg Temp

**volatile** uint32\_t ui32TempValueC; //celsius

**volatile** uint32\_t ui32TempValueF; //F

**SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0); //enable ADC0

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF); //enable GPIOF

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3); //setoutput

**ADCSequenceConfigure**(ADC0\_BASE, 3, ADC\_TRIGGER\_PROCESSOR, 0); //SS3

**ADCSequenceStepConfigure**(ADC0\_BASE,3,0,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END); //configure step in ADC sequencer

**ADCSequenceEnable**(ADC0\_BASE, 3); //enable ADC sequencer 3

**while**(1)

{

**ADCIntClear**(ADC0\_BASE, 3);

**ADCProcessorTrigger**(ADC0\_BASE, 3);

**while**(!**ADCIntStatus**(ADC0\_BASE, 3, false))

{

}

**ADCSequenceDataGet**(ADC0\_BASE, 3, 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){ //if Farenheit is over 72 then turn on blue LED (PF2)

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 4);

}

**else**{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0);

}

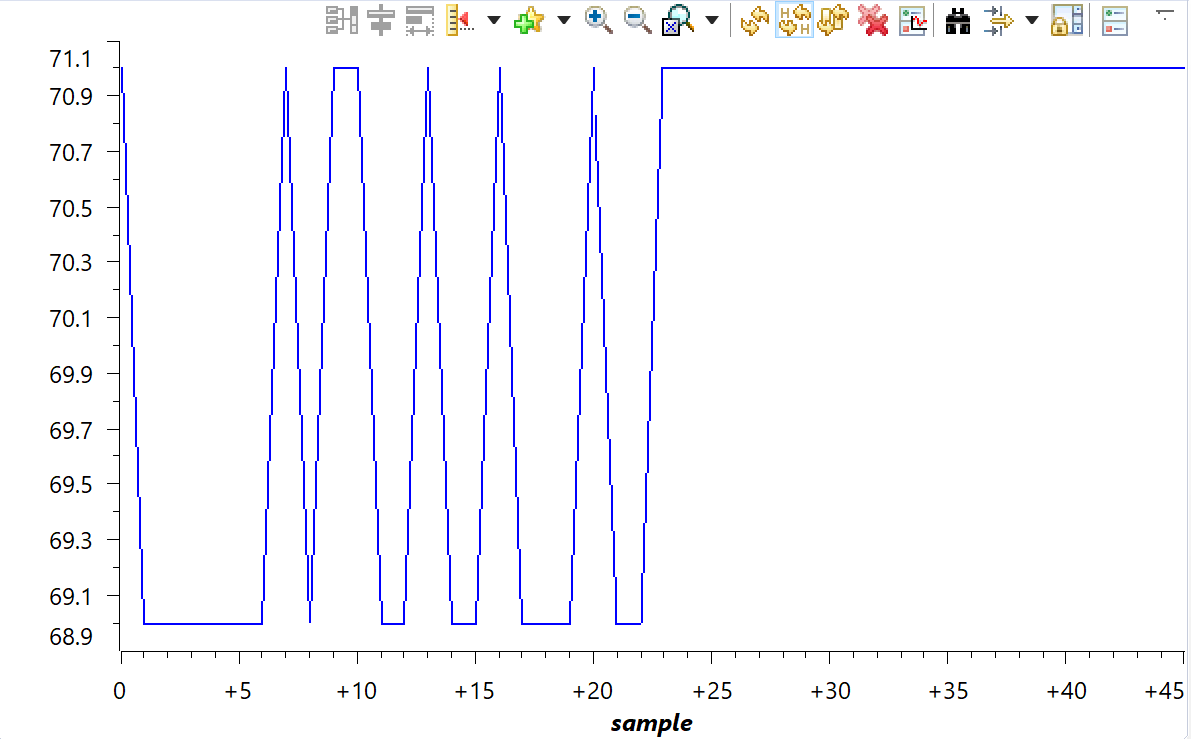
}

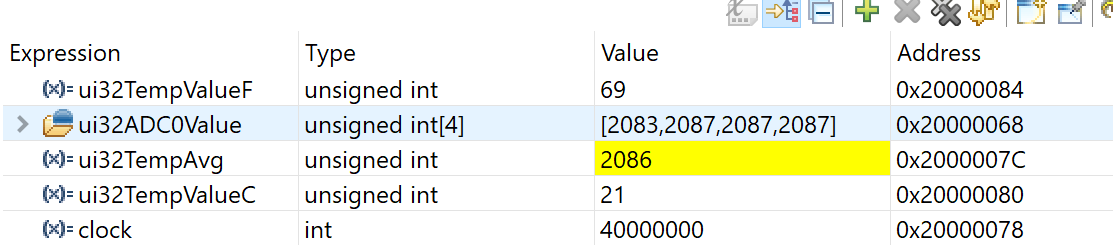
}

**------------------------------------------------------------------------------------**

**Task 02:** Introduce hardware averaging to 32. Using the timer TIMER1A conduct an ADC conversion on overflow every 0.5 sec. Use the Timer1A interrupt. Display the temperature in the built-in graph tool.

Youtube Link: <https://www.youtube.com/watch?v=4fRExM7t4Ww>





Above is the graph used to measure ui32TempValueF. This program is responsible for implementing hardware averaging to 32. This program will also use Timer1A and when 0.5s has passed, data will be read from the temperature sensor and will be inputted into the graph.

**Modified Code:**

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_types.h"

**#include** "driverlib/timer.h"

**#include** "driverlib/debug.h"

**#include** "driverlib/sysctl.h"

**#include** "driverlib/adc.h"

**#include** "driverlib/gpio.h"

**#include** "inc/tm4c123gh6pm.h"

**int** clock; //check clock freq

uint32\_t ui32ADC0Value[4];

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

**void** **Timer1IntHandler**(**void**) //Timer1A interrupt

{

clock = **SysCtlClockGet**();

SYSCTL\_RCGCTIMER\_R |= 2; //enable clock

TIMER1\_CTL\_R = 0; //disable Timer before initialization

TIMER1\_CFG\_R = 0x04; //16-bit option

TIMER1\_TAMR\_R = 0x02; //periodic mode and down-counter

TIMER1\_TAILR\_R = 80000-1; //TimerA interval load value reg

TIMER1\_TAPR\_R = 250 - 1; //TimerA prescaler 40MHz/250 = 160000Hz => 1/160000 = 6.25us => .5s/6.25us = 80000

TIMER1\_ICR\_R = 0x01; //clear the TimerA timeout flag

TIMER1\_CTL\_R |= 0x01; //enable Timer A after initialization

**while**((TIMER1\_RIS\_R & 0x1) == 0)

;

**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;

TIMER1\_ICR\_R = 0x1; //clear flag

}

**int** **main**(**void**)

{

uint32\_t ui32Period;

**SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER1);

**TimerConfigure**(TIMER1\_BASE, TIMER\_CFG\_PERIODIC);

ui32Period = (**SysCtlClockGet**() / 10) / 2;

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, ui32Period -1);

IntEnable(INT\_TIMER1A);

**TimerIntEnable**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

IntMasterEnable();

**TimerEnable**(TIMER1\_BASE, TIMER\_A); //enable Timer1A

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0); //enable ADC

**ADCHardwareOversampleConfigure**(ADC0\_BASE, 32);

**ADCSequenceConfigure**(ADC0\_BASE, 1, ADC\_TRIGGER\_PROCESSOR, 0); //SS1

**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)

{

}

}