**Date Submitted: 10/22/19**

**Task 00:**

**#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]; //sequencer 1 with depth of 4

//volatile so that each variable cannot be optimized out by the compiler

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

//Have clock run at 40MHz

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

//Enable ADC0 peripheral

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0);

**ADCHardwareOversampleConfigure**(ADC0\_BASE, 64);

//ADC will run at default rate of 1Msps

//Configure ADC sequencer 1

//want the processor to trigger the sequence and use highest priority

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

//Configure all four steps of ADC sequencer

//temp 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);

//Configure interrupt flag = ADC\_CTL\_IE

//Tell 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

//read the value of the temperature sensor and calculate the temperature endlessly

**while**(1)

{

**ADCIntClear**(ADC0\_BASE, 1); //clear interrupt flag

**ADCProcessorTrigger**(ADC0\_BASE, 1); //trigger ADC conversion with software

//wait for conversion

**while**(!**ADCIntStatus**(ADC0\_BASE, 1, false))

{

}

//get data from a buffer in memory

**ADCSequenceDataGet**(ADC0\_BASE, 1, ui32ADC0Value);

//Calculate average of the temperature sensor

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

//Calculate Celsius

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10;

//Calculate Fahrenheit

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5;

}

}

**Youtube Link:**

<https://www.youtube.com/watch?v=qUxirbQrL-M>

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

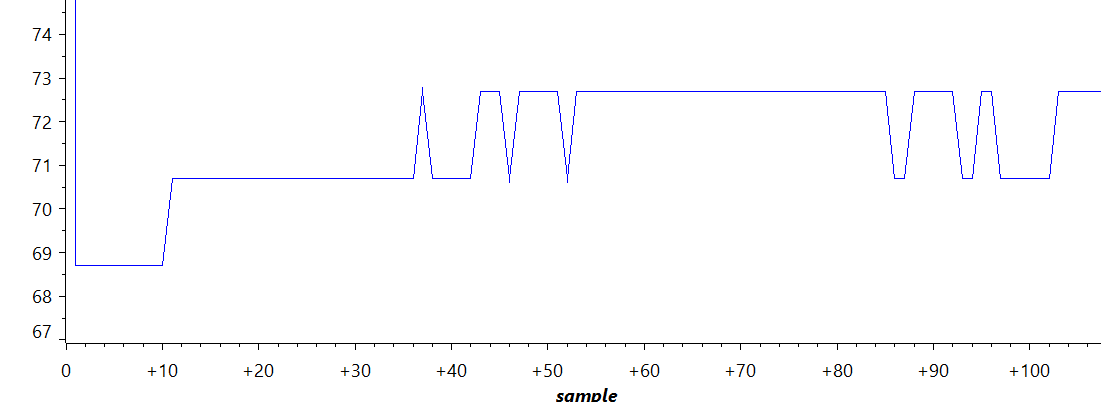
**Task 01:**

In this task, I will change the ADC sequencer to SS2(4 sequences). Once, the temperature is greater than 72 degreesF LED PF2 will turn on else PF1 is ON. I will be using the internal temperature sensor and displaying temperature on the built-in graph tool. In my code I had to do at 70 degrees due to internal temp. sensor.

Youtube Link:

<https://www.youtube.com/watch?v=O4vN-P4HWEs>

**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"

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

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

{

uint32\_t ui32ADC0Value[4]; //sequencer 2 with depth of 4

//volatile so that each variable cannot be optimized out by the compiler

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

//Have clock run at 40MHz

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

//Enable ADC0 peripheral

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0);

ADCHardwareOversampleConfigure(ADC0\_BASE, 64);

//ADC will run at default rate of 1Msps

//Configure ADC sequencer 2

//want the processor to trigger the sequence and use highest priority

**ADCSequenceConfigure**(ADC0\_BASE, 2, ADC\_TRIGGER\_PROCESSOR, 0);

//Configure all four steps of ADC sequencer

//temp sensor = ADC\_CTL\_TS

**ADCSequenceStepConfigure**(ADC0\_BASE, 2, 0, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 2, 1, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 2, 2, ADC\_CTL\_TS);

//Configure interrupt flag = ADC\_CTL\_IE

//Tell ADC logic that this is the last conversion on sequencer 2 = ADC\_CTL\_END

**ADCSequenceStepConfigure**(ADC0\_BASE,2,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

**ADCSequenceEnable**(ADC0\_BASE, 2); //Enable ADC sequencer 2

//Enable PortF and all 3 LEDs

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

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

//read the value of the temperature sensor and calculate the temperature endlessly

**while**(1)

{

**ADCIntClear**(ADC0\_BASE, 2); //clear interrupt flag

**ADCProcessorTrigger**(ADC0\_BASE, 2); //trigger ADC conversion with software

//wait for conversion

**while**(!**ADCIntStatus**(ADC0\_BASE, 2, false))

{

}

//get data from a buffer in memory

**ADCSequenceDataGet**(ADC0\_BASE, 2, ui32ADC0Value);

//Calculate average of the temperature sensor

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

//Calculate Celsius

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10;

//Calculate Fahrenheit

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5;

**if**(ui32TempValueF >= 70)

{

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

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

}

**else**

{

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

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 2);

}

}

}

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

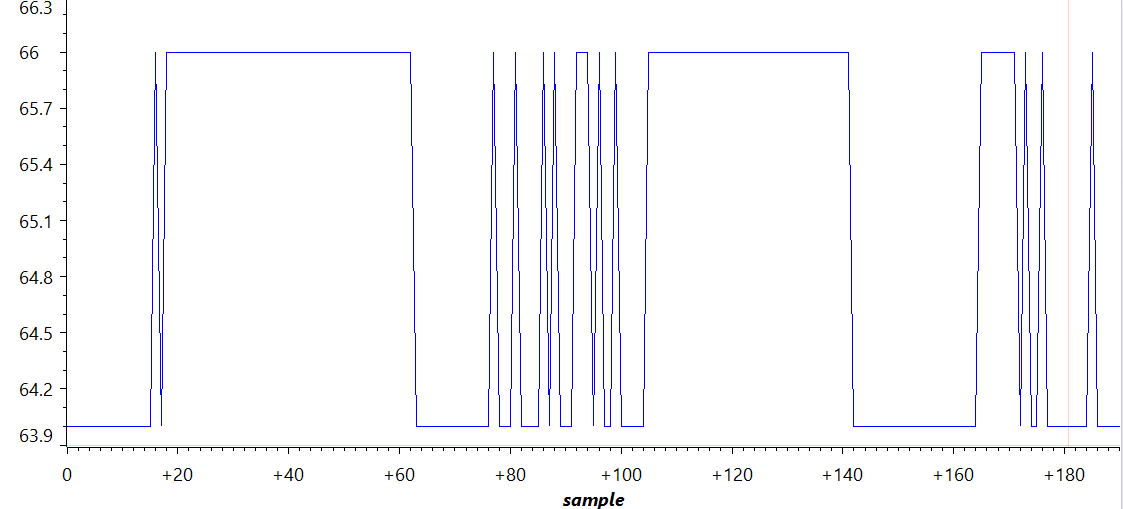
**Task 02:**

In this task I was to set the hardware averaging to 32 and have the interrupt to occur every 0.5 sec.

Youtube Link:

<https://www.youtube.com/watch?v=sQUfnzrPhIc>

**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"

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

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

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

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

uint32\_t ui32ADC0Value[4]; //sequencer 2 with depth of 4

//volatile so that each variable cannot be optimized out by the compiler

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

uint32\_t ui32Period;

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

{

//Have clock run at 40MHz

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

//Enable ADC0 peripheral

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0);

**ADCHardwareOversampleConfigure**(ADC0\_BASE, 32); //hardware average of 32

//ADC will run at default rate of 1Msps

//Configure ADC sequencer 2

//want the processor to trigger the sequence and use highest priority

**ADCSequenceConfigure**(ADC0\_BASE, 2, ADC\_TRIGGER\_PROCESSOR, 0);

//Configure all four steps of ADC sequencer

//temp sensor = ADC\_CTL\_TS

**ADCSequenceStepConfigure**(ADC0\_BASE, 2, 0, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 2, 1, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 2, 2, ADC\_CTL\_TS);

//Configure interrupt flag = ADC\_CTL\_IE

//Tell ADC logic that this is the last conversion on sequencer 2 = ADC\_CTL\_END

**ADCSequenceStepConfigure**(ADC0\_BASE,2,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

**ADCSequenceEnable**(ADC0\_BASE, 2); //Enable ADC sequencer 2

//Enable PortF and all 3 LEDs

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

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

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

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

ui32Period = (**SysCtlClockGet**() / 10) / 2;//10Hz and 50% DC

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

//Enabling Interrupt

**IntEnable**(INT\_TIMER1A);

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

**IntMasterEnable**();

//Will start the timer and interrupts will begin triggering on the timeouts

**TimerEnable**(TIMER1\_BASE, TIMER\_A);

//Enable ADC interrupt

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

//The toggling of the GPIO will happen in the interrupt service routine

**while**(1)

{

}

}

**void** **Timer1IntHandler**(**void**)

{

**TimerIntClear**(TIMER1\_BASE,TIMER\_A);

//read the value of the temperature sensor and calculate the temperature endlessly

**ADCIntClear**(ADC0\_BASE, 2); //clear interrupt flag

**ADCProcessorTrigger**(ADC0\_BASE, 2); //trigger ADC conversion with software

//wait for conversion

**while**(!**ADCIntStatus**(ADC0\_BASE, 2, false))

{

}

//get data from a buffer in memory

**ADCSequenceDataGet**(ADC0\_BASE, 2, ui32ADC0Value);

//Calculate average of the temperature sensor

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

//Calculate Celsius

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10;

//Calculate Fahrenheit

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5;

**if**(ui32TempValueF >= 66)

{

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

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

}

**else**

{

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

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 2);

}

}

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