**Date Submitted: 10/09/2018**

**Assignment Youtube Playlist:** **https://www.youtube.com/playlist?list=PL4oTyvRrubXe3x1bETaZWiDH8p9KR-ccO**

**Task 00: Execute the supplied code, no submission required.**

**Youtube Link:** **https://youtu.be/kwca8qs7iGg**

**Original Code (added comments):**

**#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 all required files and drivers for the lab.

**#define** TARGET\_IS\_BLIZZARD\_RB1

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

//Enable calling the Peripheral Driver Library from ROM, saving flash space.

**#ifdef** DEBUG //Record file name and line number of any errors from a library API with incorrect parameters.

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

{

}

**#endif**

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

{

uint32\_t ui32ADC0Value[4];

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

//Set up an array to store our ADC output, must be greater or equal to our FIFO size to capture everything.

//Set up variables to store the calculated average, degrees C, and degrees F of our temperature.

**SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ); //40MHz clock total.

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0); //Enable clock to ADC0.

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

//ADC0, Sample Sequencer 1, triggered by processor, highest priority (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);

//Configure the four steps of our sequencer. All using ADC0 and sequencer 1, measuring the internal temperature

//sensor, then triggering the interrupt flag when finished and signal the last conversion on sequencer 1.

//Enable ADC after all four steps are set.

**while**(1)

{

**ADCIntClear**(ADC0\_BASE, 1); //Clear ADC interrupt flag.

**ADCProcessorTrigger**(ADC0\_BASE, 1); //Trigger ADC conversion with the processor.

**while**(!**ADCIntStatus**(ADC0\_BASE, 1, false)) //Wait for ADC conversion to complete.

{

}

**ADCSequenceDataGet**(ADC0\_BASE, 1, ui32ADC0Value); //After conversion is complete, pull the data into the array.

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

//Average the four array temperatures, adding +2 to compensate for rounding due to integer math.

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10; //Calculate degrees C.

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5; //Calculate degrees F.

}

}

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

**Task 01:** Change the ADC Sequencer to SS0 (Corrected per Dr. Venki’s E-mail). Turn on the LED at PF2 if the temperature is greater that 72 degF. Use internal temperature sensor for all SS0 sequence. Display the temperature in the built-in graph tool.

**Youtube Link:** **https://youtu.be/o3mf1iupLlY**

For Task 1 I adjusted the ADC sequencer to SS0 per Dr. Venki’s E-mail (Depth of 8). We use integer math to average the eight samples and then calculate the temperature in degrees Fahrenheit and degrees Celsius. This causes our LED at PF2 (blue) to light up if our calculated temperature is at or above 72 degrees Fahrenheit.

**#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 all required files and drivers for the lab.

**#define** TARGET\_IS\_BLIZZARD\_RB1

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

//Enable calling the Peripheral Driver Library from ROM, saving flash space.

**#ifdef** DEBUG //Record file name and line number of any errors from a library API with incorrect parameters.

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

{

}

**#endif**

uint32\_t ui32ADC0Value[8];

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

//Set up an array to store our ADC output, must be greater or equal to our FIFO size to capture everything (SS0 size 8).

//Set up variables to store the calculated average, degrees C, and degrees F of our temperature.

**const** uint8\_t ui8LED = 4; //Store value to turn on LED at Port F pin 2 (Blue).

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

{

**const** uint8\_t ui8LED = 4; //Store value to turn on LED at Port F pin 2 (Blue).

**SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ); //40MHz clock total.

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0); //Enable clock to ADC0.

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

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

//Enable GPIO Port F and set Pin 2 as output.

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

//ADC0, Sample Sequencer 0, triggered by processor, highest priority (0).

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

**ADCSequenceStepConfigure**(ADC0\_BASE, 0, 1, ADC\_CTL\_TS);

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

**ADCSequenceStepConfigure**(ADC0\_BASE, 0, 3, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 0, 4, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 0, 5, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 0, 6, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 0, 7, ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

**ADCSequenceEnable**(ADC0\_BASE, 0);

//Configure the four steps of our sequencer. All using ADC0 and sequencer 0, measuring the internal temperature

//sensor, then triggering the interrupt flag when finished and signal the last conversion on sequencer 0.

//Enable ADC after all eight steps are set.

**while**(1)

{

**ADCIntClear**(ADC0\_BASE, 0); //Clear ADC interrupt flag.

**ADCProcessorTrigger**(ADC0\_BASE, 0); //Trigger ADC conversion with the processor.

**while**(!**ADCIntStatus**(ADC0\_BASE, 0, false)) //Wait for ADC conversion to complete.

{

}

**ADCSequenceDataGet**(ADC0\_BASE, 0, ui32ADC0Value); //After conversion is complete, pull the data into the array.

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + ui32ADC0Value[4] + ui32ADC0Value[5] + ui32ADC0Value[6] + ui32ADC0Value[7] + 4) /8;

//Average the eight array temperatures, adding +4 to compensate for rounding due to integer math.

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10; //Calculate degrees C.

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5; //Calculate degrees F.

**if**(ui32TempValueF > 72)

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, ui8LED); //If Temperature is greater than 72F, turn on LED.

}

**else**

{

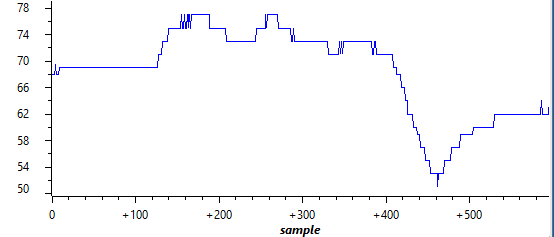
**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0); //If temperature is not greater than 72F, turn off LED.

}

}

}

Temperatures observed in internal graphing tool while heating/cooling the device.



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

**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://youtu.be/-3HpKFUo1aQ**

For Task 2 I introduced hardware averaging for 32 samples. This combines with our 8 averaged samples from SS0 for a total of 256 samples. This is triggered by timer 1A every 0.5s. The graph can be seen to react much more slowly, but is much less erratic.

**#include** <stdint.h>

**#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"

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

//Include all required files and drivers for the lab.

**#define** TARGET\_IS\_BLIZZARD\_RB1

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

//Enable calling the Peripheral Driver Library from ROM, saving flash space.

**#ifdef** DEBUG //Record file name and line number of any errors from a library API with incorrect parameters.

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

{

}

**#endif**

uint8\_t ui8LED = 4; //Store value to turn on LED at Port F pin 2 (Blue).

uint32\_t ui32ADC0Value[8];

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

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

{

//uint32\_t ui32ADC0Value[8];

//Set up an array to store our ADC output, must be greater or equal to our FIFO size to capture everything (SS0 size 8).

//volatile uint32\_t ui32TempAvg;

//volatile uint16\_t ui16TempAvg1;

//volatile uint16\_t ui16TempAvg2;

//volatile uint32\_t ui32TempValueC;

//volatile uint32\_t ui32TempValueF;

//Set up variables to store the calculated average, degrees C, and degrees F of our temperature.

//Two averaging variables required, total numbers being operated on exceeds 32-bit capacity.

uint32\_t ui32TimerDelay; //Set timer delay for use in ADC triggering.

**SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ); //40MHz clock total.

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0); //Enable clock to ADC0.

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER1); //Enable Timer 1 peripheral.

**TimerConfigure**(TIMER1\_BASE, TIMER\_CFG\_PERIODIC); //Configure timer 1 to be periodic.

uint32\_t ui32Clock = **SysCtlClockGet**(); //Retrieve system clock and store in variable.

ui32TimerDelay = (ui32Clock / 2); //Trigger at 2Hz, 0.5 second period.

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, ui32TimerDelay -1); //Load button on-period into timer 1.

**IntEnable**(INT\_TIMER1A); //Enable vector associated with Timer 1A.

**TimerIntEnable**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT); //Enable interrupt from Timer 1A.

**IntMasterEnable**();

**TimerEnable**(TIMER1\_BASE, TIMER\_A); //Enable and start timer 1A.

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

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

//Enable GPIO Port F and set Pin 2 as output.

**ADCHardwareOversampleConfigure**(ADC0\_BASE, 32); //Configure hardware averaging of ADC0, 32 samples.

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

//ADC0, Sample Sequencer 0, triggered by processor, highest priority (0).

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

**ADCSequenceStepConfigure**(ADC0\_BASE, 0, 1, ADC\_CTL\_TS);

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

**ADCSequenceStepConfigure**(ADC0\_BASE, 0, 3, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 0, 4, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 0, 5, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 0, 6, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 0, 7, ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

**ADCSequenceEnable**(ADC0\_BASE, 0);

//Configure the eight steps of our sequencer. All using ADC0 and sequencer 0, measuring the internal temperature

//sensor, then triggering the interrupt flag when finished and signal the last conversion on sequencer 0.

//Enable ADC after all eight steps are set.

//TimerControlTrigger(TIMER1\_BASE, TIMER\_A, 1); //Enables the ADC trigger output, support to be added at a later time.

**while**(1)

{

**if**(ui32TempValueF > 72)

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, ui8LED); //If Temperature is greater than 72F, turn on LED.

}

**else**

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0); //If temperature is not greater than 72F, turn off LED.

}

}

}

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

{

// Clear the timer interrupt

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

**ADCIntClear**(ADC0\_BASE, 0); //Clear ADC interrupt flag.

**ADCProcessorTrigger**(ADC0\_BASE, 0); //Trigger ADC conversion with the processor.

**while**(!**ADCIntStatus**(ADC0\_BASE, 0, false)) //Wait for ADC conversion to complete.

{

}

**ADCSequenceDataGet**(ADC0\_BASE, 0, ui32ADC0Value); //After conversion is complete, pull the data into the array.

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + ui32ADC0Value[4] + ui32ADC0Value[5] + ui32ADC0Value[6] + ui32ADC0Value[7] + 4) /8;

//ui16TempAvg1 = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2) /4;

//ui16TempAvg2 = (ui32ADC0Value[4] + ui32ADC0Value[5] + ui32ADC0Value[6] + ui32ADC0Value[7] + 2) /4;

//Average the eight array temperatures, adding +4 to compensate for rounding due to integer math.

//Two operations required due to 32-bit limitations. May result in slight rounding errors.

//ui32TempAvg = (ui16TempAvg1 + ui16TempAvg2 + 1) /2;

//Calculating total average.

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10; //Calculate degrees C.

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5; //Calculate degrees F.

**if**(ui32TempValueF > 72)

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, ui8LED); //If Temperature is greater than 72F, turn on LED.

}

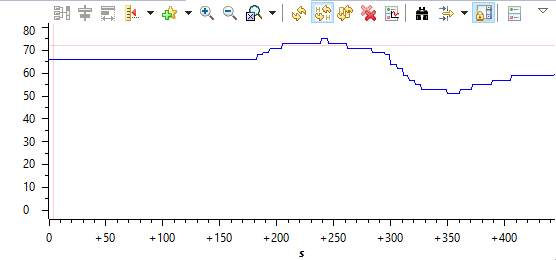
**else**

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0); //If temperature is not greater than 72F, turn off LED.

}

}



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