**ADC Lab**

**Links to videos:**

**Task 1:** http://screencast.com/t/ifILqx96WXx

**Task 2:** http://screencast.com/t/gsgpNASp3Zkr

**Task 3:** http://screencast.com/t/x39BEd5RGc34

**Task 1: Adding comments to original code**

//http://screencast.com/t/ifILqx96WXx

**#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]; //will store 4 values from FIFO when using sequencer 1

**volatile** uint32\_t ui32TempAvg; //stores average of 4 sampled values

**volatile** uint32\_t ui32TempValueC; //stores temperature in Celsius

**volatile** uint32\_t ui32TempValueF; //stores temperature in Fahrenheit

SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ); //set clock to 40MHz

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0); //Enable ADC0

ADCSequenceConfigure(ADC0\_BASE, 1, ADC\_TRIGGER\_PROCESSOR, 0); //set sequencer 1 to trigger with CPU

ADCSequenceStepConfigure(ADC0\_BASE, 1, 0, ADC\_CTL\_TS); //configure step 0 from temp sensor

ADCSequenceStepConfigure(ADC0\_BASE, 1, 1, ADC\_CTL\_TS); //configure sequencer step 1 from temp sensor

ADCSequenceStepConfigure(ADC0\_BASE, 1, 2, ADC\_CTL\_TS); //configure sequencer step 2 from temp sensor

ADCSequenceStepConfigure(ADC0\_BASE,1,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END); //configure sequencer step 3 from temp sensor and tell sequencer to finish

ADCSequenceEnable(ADC0\_BASE, 1); //enable sequencer

**while**(1) //loop forever

{

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

ADCProcessorTrigger(ADC0\_BASE, 1); //Trigger ADC0 sequencer 1

**while**(!ADCIntStatus(ADC0\_BASE, 1, **false**)) //wait for ADC conversion to finish

{

}

ADCSequenceDataGet(ADC0\_BASE, 1, ui32ADC0Value); //get data from FIFO and put into array

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4; //calculate average temperature (+2/4 used for rounding)

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10; //calculate temp in Celsius

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5; //calculate temp in Fahrenheit

}

}

**Task 2: Change the ADC Sequencer to SS2. Turn on the LED at PF1 if the temperature is greater than 80 deg-F.**

//http://screencast.com/t/gsgpNASp3Zkr

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

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

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

{

uint32\_t ui32ADC0Value[4]; //will store 4 values from FIFO when using sequencer 1

**volatile** uint32\_t ui32TempAvg; //stores average of 4 sampled values

**volatile** uint32\_t ui32TempValueC; //stores temperature in Celsius

**volatile** uint32\_t ui32TempValueF; //stores temperature in Fahrenheit

SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ); //set clock to 40MHz

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

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); //enable port F

GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_1); //Configure pin F1 as output

ADCSequenceConfigure(ADC0\_BASE, 2, ADC\_TRIGGER\_PROCESSOR, 0); //configure ADC0, sequencer 2, trigger with processor

ADCSequenceStepConfigure(ADC0\_BASE, 2, 0, ADC\_CTL\_TS); //configure ADC0, sequencer 2, step 0

ADCSequenceStepConfigure(ADC0\_BASE, 2, 1, ADC\_CTL\_TS); //configure ADC0, sequencer 2, step 1

ADCSequenceStepConfigure(ADC0\_BASE, 2, 2, ADC\_CTL\_TS); //configure ADC0, sequencer 2, step 2

ADCSequenceStepConfigure(ADC0\_BASE,2,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END); //configure ADC0, sequencer 2, step 3, and finish

ADCSequenceEnable(ADC0\_BASE, 2); //enable ADC0, sequencer 2

**while**(1) //infinite loop

{

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

ADCProcessorTrigger(ADC0\_BASE, 2); //trigger ADC sequencer 2

**while**(!ADCIntStatus(ADC0\_BASE, 2, **false**)) //wait for ADC conversion to finish

{

}

ADCSequenceDataGet(ADC0\_BASE, 2, ui32ADC0Value); //get 4 values from sequencer 2 FIFO to buffer

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4; //take average of 4 values

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10; //convert temperature to Celsius

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5; //convert temperature to Fahrenheit

ui32TempValueF += 5; //calibrate Fahrenheit temperature to be displayed

**if**(ui32TempValueF > 80) //turn on LED on PF2 if temp is over 80 deg F.

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

**else**

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 0); //else turn off LED on PF2

}

}

**Task 3: Introduce hardware averaging to 64. Using the timer TIMER0A conduct an ADC conversion on overflow every 0.5 sec. Use the Timer0A interrupt.**

//http://screencast.com/t/x39BEd5RGc34

**#include** <stdint.h>

**#include** <stdbool.h>

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

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

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

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

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

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

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

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

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

uint32\_t ui32ADC0Value[4]; //will store 4 samples of temperature

**volatile** uint32\_t ui32TempAvg; //stores average temperature based on certain number of samples

**volatile** uint32\_t ui32TempValueC; //stores temperature in celsius

**volatile** uint32\_t ui32TempValueF; //stores temperature in fahrenheit

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

{

uint32\_t ui32Period; //will be used for time delay

SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ); //set clock to 40 MHz

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0); //enable adc0 for temperature sensor

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); //enable port f for LED

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER0); //enable timer 0 for interrupts

TimerConfigure(TIMER0\_BASE, TIMER\_CFG\_PERIODIC); //configure timer 0 periodic mode

ui32Period = SysCtlClockGet()/ 2; //get clock and divide by 2 for 50% DC

TimerLoadSet(TIMER0\_BASE, TIMER\_A, ui32Period - 1); //load value into timer 0

ADCHardwareOversampleConfigure(ADC0\_BASE, 64); //configure hardware averaging of 64

GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_1); //configure PF1 as output

ADCSequenceConfigure(ADC0\_BASE, 2, ADC\_TRIGGER\_PROCESSOR, 0); //configure ADC0 sequencer 2 to trigger with processor

ADCSequenceStepConfigure(ADC0\_BASE, 2, 0, ADC\_CTL\_TS); //configure ADC0 sequencer 2 step 0

ADCSequenceStepConfigure(ADC0\_BASE, 2, 1, ADC\_CTL\_TS); //configure ADC0 sequencer 2 step 1

ADCSequenceStepConfigure(ADC0\_BASE, 2, 2, ADC\_CTL\_TS); //configure ADC0 sequencer 2 step 2

ADCSequenceStepConfigure(ADC0\_BASE,2,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END); //configure ADC0 sequencer 2 step 3

ADCSequenceEnable(ADC0\_BASE, 2); //enable ADC0 sequencer 2

IntEnable(INT\_TIMER0A); //Enable interrupts on timer 0

TimerIntEnable(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); //set timer 0 to interrupt at timeout

IntMasterEnable(); //enable master interrupt

TimerEnable(TIMER0\_BASE, TIMER\_A); //start the timer

**while**(1) //infinite loop

{

}

}

**void** Timer0IntHandler(**void**) //interrupt handler for timer 0

{

TimerIntClear(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); //clear interrupt flag on timer 0

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

ADCProcessorTrigger(ADC0\_BASE, 2); //trigger ADC0 sequencer 2

**while**(!ADCIntStatus(ADC0\_BASE, 2, **false**)) //wait for ADC conversion to finish

{

}

ADCSequenceDataGet(ADC0\_BASE, 2, ui32ADC0Value); //get 4 values from sequencer 2 FIFO into buffer

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4; //take average of 4 values

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10; //convert temperature to Celsius

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5; //conver temperature to Fahrenheit

ui32TempValueF += 5; //calibrate Fahrenheit temperature to be displayed

**if**(ui32TempValueF > 80) //turn on LED on PF2 if temp is over 80 deg F.

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 2); //turn on LED if temp > 80 deg F

**else**

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 0); //turn off LED if temp < 80 deg F

}