**Lab05**

Task01:

Sequencer 3 has a FIFO depth of 1, so the manual temp averaging done in task 0 is irrelevant for this part of the lab. Also, I believe that the temperature sensor on my TIVAC is faulty because the temperature only ranges from 62-69 degrees Fahrenheit, so an offset of 5 was added to the Fahrenheit calculation in the code to be able to show the LED turning on once the read temp surpasses 72 degrees.

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"

#define TARGET\_IS\_BLIZZARD\_RB1

#include "driverlib/rom.h"

#include "driverlib/gpio.h"

#ifdef DEBUG

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

{

}

#endif

int main(void)

{

// Define variable to store data from the ADC FIFO. SS3 has a FIFO depth of 1

uint32\_t ui32ADC0Value[1];

// Define variables for use in later calculations

volatile uint32\_t ui32TempValueC;

volatile uint32\_t ui32TempValueF;

// Clock run at 40MHz

ROM\_SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);

// Enable PortF

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

// Set blue LED to output

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

// Enable ADC Peripheral

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

// Average 64 values before putting in the ADC FIFO

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 64);

// Configure ADC sequencer to use ADC0 and SS3

ROM\_ADCSequenceConfigure(ADC0\_BASE, 3, ADC\_TRIGGER\_PROCESSOR, 0);

// Sample temp and enable the interrupt flag to say that calculation has been completed

ROM\_ADCSequenceStepConfigure(ADC0\_BASE,3,0,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

// Enable ADC sequence

ROM\_ADCSequenceEnable(ADC0\_BASE, 3);

while(1)

{

// Clear ADC calculation complete interrupt

ROM\_ADCIntClear(ADC0\_BASE, 3);

// Start ADC conversion

ROM\_ADCProcessorTrigger(ADC0\_BASE, 3);

// Wait for conversion to complete

while(!ROM\_ADCIntStatus(ADC0\_BASE, 3, false))

{

}

// Copy ADC value

ROM\_ADCSequenceDataGet(ADC0\_BASE, 3, ui32ADC0Value);

// Convert temp to Celsius

ui32TempValueC = (1475 - ((2475 \* ui32ADC0Value[0])) / 4096)/10;

// Convert temp to Fahrenheit

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

// If temp is above 72, turn on LED

if(ui32TempValueF > 72)

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

// If temp is not above 72, turn LED off

else

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

}

}

Task02:

Timer1A was used at 2Hz (0.5sec), so that every time the timer overflows, the ADC conversion sequence begins. The Timer1A overflow interrupt function was also added to the startup file for the lab, which is why the lab 5 submission includes two different startup files: one for task 1 and one for task 2.

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"

#define TARGET\_IS\_BLIZZARD\_RB1

#include "driverlib/rom.h"

#include "driverlib/timer.h"

#include "driverlib/interrupt.h"

#include "inc/tm4c123gh6pm.h"

#ifdef DEBUG

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

{

}

#endif

int main(void)

{

// Use to define the period of ADC conversion

uint32\_t ui32Period;

// Clock run at 40MHz

ROM\_SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);

// Enable ADC Peripheral

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

// Average 32 values before putting in the ADC FIFO

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 32);

// Configure ADC sequencer to use ADC0 and SS1

ROM\_ADCSequenceConfigure(ADC0\_BASE, 1, ADC\_TRIGGER\_PROCESSOR, 0);

// Setup steps for ADC sequencer. The temp value will show the average of 4 temp values

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 0, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 1, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 2, ADC\_CTL\_TS);

// Sample temp again and enable the interrupt flag to say that calculation has been completed

ROM\_ADCSequenceStepConfigure(ADC0\_BASE,1,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

// Enable ADC sequence

ROM\_ADCSequenceEnable(ADC0\_BASE, 1);

// Enable timer1

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER1);

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

// Clock period of 2Hz

ui32Period = SysCtlClockGet() / 2;

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

// Enable timer1 overload as interrupts

IntEnable(INT\_TIMER1A);

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

IntMasterEnable();

// Start timer

TimerEnable(TIMER1\_BASE, TIMER\_A);

while(1)

{

}

}

void Timer1IntHandler(void)

{

// Define variable to store data from the ADC FIFO

uint32\_t ui32ADC0Value[4];

// Define variables for use in later calculations

volatile uint32\_t ui32TempAvg;

volatile uint32\_t ui32TempValueC;

volatile uint32\_t ui32TempValueF;

// Clear ADC calculation complete interrupt

ROM\_ADCIntClear(ADC0\_BASE, 1);

// Start ADC conversion

ROM\_ADCProcessorTrigger(ADC0\_BASE, 1);

// Wait for conversion to complete

while(!ROM\_ADCIntStatus(ADC0\_BASE, 1, false))

{

}

// Copy ADC value

ROM\_ADCSequenceDataGet(ADC0\_BASE, 1, ui32ADC0Value);

// Calculate average temperature reading

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

// Convert temp to Celsius

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

// Convert temp to Fahrenheit

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

}