**Lab 5**

**Task 1**

**#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"

**#ifdef** DEBUG

**void\_\_error\_\_**(**char** \*pcFilename, uint32\_t ui32line)

{

}

**#endif**

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

{

uint32\_t ui32ADC0Value[4]; //uses the 4 deep ADC FIFO, hence the size of the array

**volatile** uint32\_t ui32TempAvg, ui32TempValueC, ui32TempValueF; //variables to hold temporary average values, Celsius, and Farenhiet

ROM\_SysCtlClockSet (SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ); //Sets the system clock to 40MHz

ROM\_SysCtlPeripheralEnable (SYSCTL\_PERIPH\_ADC0); //Enables ADC GPIO

ROM\_ADCHardwareOversampleConfigure (ADC0\_BASE, 64); //samples the adc based on the amount declared in the last argument (sample 64 times with 4 samples per time)

**ADCSequenceConfigure** (ADC0\_BASE, 1, ADC\_TRIGGER\_PROCESSOR, 0); //Configures the ADC to use sequencer 1

ROM\_ADCSequenceStepConfigure (ADC0\_BASE, 1, 0, ADC\_CTL\_TS); //Configures the sample sequencer 1's steps 0-2 to sample the on-chip temperature sensor

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

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

ROM\_ADCSequenceStepConfigure (ADC0\_BASE, 1, 3, ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END); //Configures the last step to sample the temperature sensor, and enable interrupt and end

ROM\_ADCSequenceEnable (ADC0\_BASE, 1); //Enables the ADC0 sequence sampler

**while** (1)

{

ROM\_ADCIntClear (ADC0\_BASE, 1); //clears the interrupt before working with it

ROM\_ADCProcessorTrigger (ADC0\_BASE, 1); //Triggers the ADC conversion via software

**while** (!ROM\_ADCIntStatus (ADC0\_BASE, 1, false)) //waits until the interrupt flag is set, wait for sampling to finish

{

}

ROM\_ADCSequenceDataGet (ADC0\_BASE, 1, ui32ADC0Value); //get the data from the FIFO and store in the variable

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4; //averages the FIFO data

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10; //calculates the celsius unit of the temperatures

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5; //calculates the farenheit version of the average temperature

}

}

**Task 2**

**#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/timer.h"

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

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

**#ifdef** DEBUG

**void\_\_error\_\_**(**char** \*pcFilename, uint32\_t ui32line)

{

}

**#endif**

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

{

uint32\_t ui32ADC0Value[4]; //uses the 4 deep ADC FIFO, hence the size of the array

**volatile** uint32\_t ui32TempAvg, ui32TempValueC, ui32TempValueF; //variables to hold temporary average values, Celsius, and Farenhiet

ROM\_SysCtlClockSet (SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ); //Sets the system clock to 40MHz

ROM\_SysCtlPeripheralEnable (SYSCTL\_PERIPH\_ADC0); //Enables ADC GPIO

ROM\_ADCHardwareOversampleConfigure (ADC0\_BASE, 64); //samples the adc based on the amount declared in the last argument (sample 16 times with 4 samples per time => 64)

//Configures the GPIO

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

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

**ADCSequenceConfigure** (ADC0\_BASE, 2, ADC\_TRIGGER\_PROCESSOR, 0); //Configures the ADC to use sequencer 1

ROM\_ADCSequenceStepConfigure (ADC0\_BASE, 2, 0, ADC\_CTL\_TS); //Configures the sample sequencer 1's steps 0-2 to sample the on-chip temperature sensor

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

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

ROM\_ADCSequenceStepConfigure (ADC0\_BASE, 2, 3, ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END); //Configures the last step to sample the temperature sensor, and enable interrupt and end

ROM\_ADCSequenceEnable (ADC0\_BASE, 2); //Enables the ADC0 sequence sampler

**while** (1)

{

ROM\_ADCIntClear (ADC0\_BASE, 2); //clears the interrupt before working with it

ROM\_ADCProcessorTrigger (ADC0\_BASE, 2); //Triggers the ADC conversion via software

**while** (!ROM\_ADCIntStatus (ADC0\_BASE, 2, false)) //waits until the interrupt flag is set, wait for sampling to finish

{

}

ROM\_ADCSequenceDataGet (ADC0\_BASE, 2, ui32ADC0Value); //get the data from the FIFO and store in the variable

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4; //averages the FIFO data

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10; //calculates the celsius unit of the temperatures

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5; //calculates the farenheit version of the average temperature

**if** (ui32TempValueF > 79)

//If the farenheit value is above 79 degrees, the led at PORTF.1 will turn on

//(Value of 68 was used in the video because I could not get the chip to get to >79 degrees)

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 2);

}

**else**

{

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

}

}

}

**Task 3**

**#include** <stdint.h>

**#include** <stdbool.h>

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

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

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

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

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

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

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

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

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

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

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

**#ifdef** DEBUG

**void\_\_error\_\_**(**char** \*pcFilename, uint32\_t ui32line)

{

}

**#endif**

uint32\_t ui32ADC0Value[4]; //uses the 4 deep ADC FIFO, hence the size of the array

**volatile** uint32\_t ui32TempAvg, ui32TempValueC, ui32TempValueF; //variables to hold temporary average values, Celsius, and Farenhiet

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

{

ROM\_SysCtlClockSet (SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ); //Sets the system clock to 40MHz

ROM\_SysCtlPeripheralEnable (SYSCTL\_PERIPH\_ADC0); //Enables ADC GPIO

ROM\_ADCHardwareOversampleConfigure (ADC0\_BASE, 16); //samples the adc based on the amount declared in the last argument (sample 16 times with 4 samples per time => 64)

//Configures the GPIO

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

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

**ADCSequenceConfigure** (ADC0\_BASE, 2, ADC\_TRIGGER\_PROCESSOR, 0); //Configures the ADC to use sequencer 1

ROM\_ADCSequenceStepConfigure (ADC0\_BASE, 2, 0, ADC\_CTL\_TS); //Configures the sample sequencer 1's steps 0-2 to sample the on-chip temperature sensor

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

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

ROM\_ADCSequenceStepConfigure (ADC0\_BASE, 2, 3, ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END); //Configures the last step to sample the temperature sensor, and enable interrupt and end

ROM\_ADCSequenceEnable (ADC0\_BASE, 2); //Enables the ADC0 sequence sampler

//Enables the interrupt configurations

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER0);

**TimerConfigure**(TIMER0\_BASE, TIMER\_CFG\_PERIODIC);

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, 13000000); //set the timer to overflow when 13000000 is reached (this is 0.333 sec)

//Enables the interrupt for TIMER0

**IntEnable**(INT\_TIMER0A);

**TimerIntEnable**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT);

**IntMasterEnable**();

//Enables TIMER

**TimerEnable**(TIMER0\_BASE, TIMER\_A);

**while** (1)

{

}

}

**void** **Timer0IntHandler**(**void**)

//This is the interrupt handler that will be called when the Timer reaches the value specified

{

ROM\_ADCIntClear (ADC0\_BASE, 2); //clears the interrupt before working with it

ROM\_ADCProcessorTrigger (ADC0\_BASE, 2); //Triggers the ADC conversion via software

**while** (!ROM\_ADCIntStatus (ADC0\_BASE, 2, false)) //waits until the interrupt flag is set, wait for sampling to finish

{

}

ROM\_ADCSequenceDataGet (ADC0\_BASE, 2, ui32ADC0Value); //get the data from the FIFO and store in the variable

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4; //averages the FIFO data

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10; //calculates the celsius unit of the temperatures

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5; //calculates the farenheit version of the average temperature

**if** (ui32TempValueF > 79)

//If the farenheit value is above 79 degrees, the led at PORTF.1 will turn on

//(Value of 68 was used in the video because I could not get the chip to get to >79 degrees)

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 2);

}

**else**

{

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

}

}