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);
	}
}
```