**Date Submitted: 10/2**

**Task 00: Execute provided code**

**Youtube Link:** [**https://youtu.be/CopjJSKdo8U**](https://youtu.be/CopjJSKdo8U)

**#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**

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

{

uint32\_t ui32ADC0Value[4];

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

// f = 40 MHz

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

// Enables the ADC

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 64);

// ADC0, sample sequencer 1, processor triggers, highest priority

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

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);

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

ROM\_ADCSequenceEnable(ADC0\_BASE, 1);

**while**(1)

{

ROM\_ADCIntClear(ADC0\_BASE, 1); // Clear ADC interrupt flag

ROM\_ADCProcessorTrigger(ADC0\_BASE, 1); // Trigger the ADC conversion

**while**(!ROM\_ADCIntStatus(ADC0\_BASE, 1, **false**)); // Poll until conversion completes

ROM\_ADCSequenceDataGet(ADC0\_BASE, 1, ui32ADC0Value); // Store the temperature value

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] // Average the sampled temperatures

+ ui32ADC0Value[2] + ui32ADC0Value[3]

+ 2) / 4;

// Convert to Celsius

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

// Convert to Fahrenheit

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

}

}

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

**Task 01:**

Youtube Link: [**https://youtu.be/Nh9oje9tkrI**](https://youtu.be/Nh9oje9tkrI)

**Modified Code:**

**First modification is to change all the SS1 to SS2 for the ADC. Next, interface the LED using PF2. This LED will turn on when the temperature is above 72 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"

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

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

**#ifdef** DEBUG

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

{

}

**#endif**

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

{

// Variables

uint32\_t ui32ADC0Value[4];

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

// f = 40 MHz

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

// Enables the ADC and GPIOF

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 64);

// ADC0, sample sequencer 2, processor triggers, highest priority

// Typo on Assignment PDF instruction for Task 01

// Task 00 already used SS1 whereas "Task01 should being using SS2"

ROM\_ADCSequenceConfigure(ADC0\_BASE, 2, ADC\_TRIGGER\_PROCESSOR, 0);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 2, 0, ADC\_CTL\_TS);

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);

ROM\_ADCSequenceEnable(ADC0\_BASE, 2);

ROM\_GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_2); // Enable PF2

**while**(1)

{

ROM\_ADCIntClear(ADC0\_BASE, 2); // Clear ADC interrupt flag

ROM\_ADCProcessorTrigger(ADC0\_BASE, 2); // Trigger the ADC conversion

**while**(!ROM\_ADCIntStatus(ADC0\_BASE, 2, **false**)); // Poll until conversion completes

ROM\_ADCSequenceDataGet(ADC0\_BASE, 2, ui32ADC0Value); // Store the temperature value

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] // Average the sampled temperatures

+ ui32ADC0Value[2] + ui32ADC0Value[3]

+ 2) / 4;

// Convert to Celsius

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

// Convert to Fahrenheit

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

// Turn on PF2 LED if temp > 72 F

**if**(ui32TempValueF > 72)

ROM\_GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 4);

**else**

ROM\_GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0);

}

}

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

**Task 02:**

Youtube Link: [**https://youtu.be/kWaHJLC10o8**](https://youtu.be/kWaHJLC10o8)

**Modified Code:**

**First, modify the startup file to include Timer1A interrupt. Next, set up Timer1A interrupt that triggers every 0.5 s. Finally, move the ADC conversion code from the while loop and into the Timer1A ISR.**

**#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/interrupt.h"

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

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

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

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

**#ifdef** DEBUG

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

{

}

**#endif**

// Variables

uint32\_t ui32ADC0Value[4];

uint32\_t ui32Period;

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

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

{

// f = 40 MHz

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

// Enables the ADC

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 32); // Average at 32 samples

// ADC0, sample sequencer 1, processor triggers, highest priority

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

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);

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

ROM\_ADCSequenceEnable(ADC0\_BASE, 1);

// Set up TIMER1

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER1);

ROM\_TimerConfigure(TIMER1\_BASE, TIMER\_CFG\_PERIODIC);

// (40 MHz / 0.5 Hz)

ui32Period = ROM\_SysCtlClockGet() \* 0.5;

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

// Enable Timer1A interrupt

ROM\_IntEnable(INT\_TIMER1A);

ROM\_TimerIntEnable(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

ROM\_IntMasterEnable();

// Start TIMER1A

ROM\_TimerEnable(TIMER1\_BASE, TIMER\_A);

**while**(1)

{

}

}

// Timer 1A ISR

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

{

// Reset the count on Timer1

ROM\_TimerDisable(TIMER1\_BASE, TIMER\_A); // Disable Timer1

ROM\_TimerLoadSet(TIMER1\_BASE, TIMER\_A, ui32Period); // (40 MHz / 0.5 Hz)

ROM\_TimerIntClear(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT); // Clear Timer1A flag

// Perform ADC conversion

ROM\_ADCIntClear(ADC0\_BASE, 1); // Clear ADC interrupt flag

ROM\_ADCProcessorTrigger(ADC0\_BASE, 1); // Trigger the ADC conversion

**while**(!ROM\_ADCIntStatus(ADC0\_BASE, 1, **false**)); // Poll until conversion completes

ROM\_ADCSequenceDataGet(ADC0\_BASE, 1, ui32ADC0Value); // Store the temperature value

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] // Average the sampled temperatures

+ ui32ADC0Value[2] + ui32ADC0Value[3]

+ 2) / 4;

// Convert to Celsius

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

// Convert to Fahrenheit

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

ROM\_TimerEnable(TIMER1\_BASE, TIMER\_A); // Re-enable Timer1

}

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