Enrique Saldana

<https://github.com/enri10>

**Date Submitted: 10/26/18**

**Task 00: Execute the provided code, no submission is required.**

**LAB05 Task00 :** [**https://youtu.be/zQvXL27QY8g**](https://youtu.be/zQvXL27QY8g)

**Task 01: Change the ADC Sequencer to SS2. Turn on the LED at PF2 if the temperature is greater that 72 degF. Use internal temperature sensor for all SS2 sequence.**

**LAB05 Task01:** [**https://youtu.be/PQtWNvZ\_3sE**](https://youtu.be/PQtWNvZ_3sE)

**#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"

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

{

//Sequencer 2 FIFO depth of 4

uint32\_t ui32ADC0Value[4];

//Variable used to calculate average temperature

//Variables used to store the temperature values in Celsius and Fahrenheit

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

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

//Enable ADC0 Peripheral

//64 measurements being averaged

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 64);

//Configure ADC Sequencer as Sequencer 2, triggered by processor and highest priority

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

//Configure steps 0 -2 on sequencer 2 to sample the temperature sensor

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);

//Configure interrupt flag and tell ADC Logic that it is the last conversion on sequencer 2

ROM\_ADCSequenceStepConfigure(ADC0\_BASE,2,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

//Enable ADC sequencer 2

ROM\_ADCSequenceEnable(ADC0\_BASE, 2);

//Enable clock for peripheral

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

//Configure PF2 LED as output

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

**while**(1)

{

//Clear ADC interrupt status flag

//Trigger ADC conversion

ROM\_ADCIntClear(ADC0\_BASE, 2);

ROM\_ADCProcessorTrigger(ADC0\_BASE, 2);

//Wait for conversion to finish

**while**(!ROM\_ADCIntStatus(ADC0\_BASE, 2, false))

{

}

//Read ADC values

//Calculate average temperature

//Calculate Celsius value

//Calculate Fahrenheit value

ROM\_ADCSequenceDataGet(ADC0\_BASE, 2, ui32ADC0Value);

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

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

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

//Turn PF2 LED on if temp value is greater than 72

**if**(ui32TempValueF > 72)

{

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

ROM\_SysCtlDelay(2000000);

}

**else**

{

ROM\_GPIOPinWrite(GPIO\_PORTF\_BASE,GPIO\_PIN\_2, 0x00);

}

}

}

**Task 02: Introduce hardware averaging to 32. Using the timer TIMER1A conduct an ADC conversion on overflow every 0.5 sec. Use the Timer1A interrupt.**

**LAB05 Task02:** [**https://youtu.be/8m3gZNzUhyY**](https://youtu.be/8m3gZNzUhyY)

Setting the clock to 2Hz will give us a period of 0.5 sec

**#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"

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

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

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

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

{

//unsigned 32-bit variable

uint32\_t ui32Period;

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

//Enable ADC0 Peripheral

//32 measurements being averaged

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 32);

//Enable Timer1 peripheral

//Configure Timer1 as a 32-bit timer in periodic mode

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER1);

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

//Calculate number of clock cycles required for 2Hz

//Load 2Hz period into the Timer's Interval Load register

ui32Period = (ROM\_SysCtlClockGet() /2)/2;

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

//Enable specific vector associated with TIMER1A

//Enable interrupt to be generated on timeout of TIMER1A

ROM\_IntEnable(INT\_TIMER1A);

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

//Configure ADC Sequencer as Sequencer 2, triggered by processor and highest priority

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

//Configure steps 0 -2 on sequencer 2 to sample the temperature sensor

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);

//Configure interrupt flag and tell ADC Logic that it is the last conversion on sequencer 1

ROM\_ADCSequenceStepConfigure(ADC0\_BASE,2,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

//Enable GPIO peripheral

//Configure PF2 LED as output

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

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

//Master interrupt enable API for all interrupts

ROM\_IntMasterEnable();

//Enable ADC sequencer 2

ROM\_ADCSequenceEnable(ADC0\_BASE, 2);

//Start Timer1

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

**while**(1)

{

}

}

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

{

// Clear the timer interrupt

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

//Sequence 2 FIFO depth of 4

uint32\_t ui32ADC0Value[4];

//Variable used to calculate average temperature

//Variables used to store the temperature values in Celsius and Fahrenheit

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

//Clear ADC interrupt status flag

//Trigger ADC conversion

ROM\_ADCIntClear(ADC0\_BASE, 2);

ROM\_ADCProcessorTrigger(ADC0\_BASE, 2);

//Wait for conversion to finish

**while**(!ROM\_ADCIntStatus(ADC0\_BASE, 2, false))

{

}

//Read ADC values

//Calculate average temperature

//Calculate Celsius value

//Calculate Fahrenheit value

ROM\_ADCSequenceDataGet(ADC0\_BASE, 2, ui32ADC0Value);

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

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

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

//if temp value is greater than 72 turn blue LED on

**if**(ui32TempValueF > 72)

{

ROM\_GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 4);

ROM\_SysCtlDelay(2000000);

}

**else**

{

ROM\_GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0x00);

}

}