TivaC Lab 5 - ADC

CPE 403

**Checklist for Lab 5**

* A text/word document of the initial code with comments
* In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also include the comments.
* Provide a permanent link to all main and dependent source code files only (name them as LabXX-TYY, XX-Lab# and YY-task#)Screenshots of debugging process along with pictures of actual circuit
* Video link of demonstration.

**Code for Experiment**

**Task 1:**

**#include** <stdint.h>

**#include** <stdbool.h>

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

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

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

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

**#define** TARGET\_IS\_BLIZZARD\_RB1 // Symbol for the API's in ROM.

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

**#ifdef** DEBUG

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

}

**#endif**

**int** **main**() {

uint32\_t ui32ADC0Value[4]; // ADC FIFO

**volatile** uint32\_t ui32TempAvg; // Store average

**volatile** uint32\_t ui32TempValueC; // Temperature in C

**volatile** uint32\_t ui32TempValueF; // Temperature in F

// Use a 40 MHz

ROM\_SysCtlClockSet(

SYSCTL\_SYSDIV\_5 | SYSCTL\_USE\_PLL | SYSCTL\_OSC\_MAIN

| SYSCTL\_XTAL\_16MHZ);

ROM\_SysCtlPeripheralEnable( SYSCTL\_PERIPH\_ADC0); // enable the ADC0

ROM\_ADCHardwareOversampleConfigure( ADC0\_BASE, 64); // configure hardware averaging

// Configure ADC0 sequencer to use sample sequencer 1, and have the processor trigger the sequence.

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

// Configure each step.

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 temperature

ROM\_ADCSequenceStepConfigure( ADC0\_BASE, 1, 3,

ADC\_CTL\_TS | ADC\_CTL\_IE | ADC\_CTL\_END);

ROM\_ADCSequenceEnable( ADC0\_BASE, 1); // Enable ADC sequencer 1

**while** (1) {

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

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

**while** (!**ADCIntStatus**(ADC0\_BASE, 1, false))

; // wait for conversion to complete.

ROM\_ADCSequenceDataGet(ADC0\_BASE, 1, ui32ADC0Value); // store data into ui32ADC0Value

// Average read values, and round.

// Each Value in the array is the result of the mean of 64 samples.

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2]

+ ui32ADC0Value[3] + 2) / 4;

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096) / 10; // calc temp in C

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5; // convert from C to F

}

}

**Task 2:**

**#include** <stdint.h>

**#include** <stdbool.h>

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

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

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

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

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

**#define** TARGET\_IS\_BLIZZARD\_RB1 // Symbol for the API's in ROM.

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

**#ifdef** DEBUG

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

}

**#endif**

**int** **main**() {

uint32\_t ui32ADC0Value[4]; // ADC FIFO

**volatile** uint32\_t ui32TempAvg; // Store average

**volatile** uint32\_t ui32TempValueC; // Temperature in C

**volatile** uint32\_t ui32TempValueF; // Temperature in F

// Use a 40 MHz Clock

ROM\_SysCtlClockSet(

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

// GPIO Enable on port F

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); // enable port F

// Set LEDs as outputs

ROM\_GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE,

GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3);

ROM\_SysCtlPeripheralEnable( SYSCTL\_PERIPH\_ADC0); // enable the ADC0 peripheral

ROM\_ADCHardwareOversampleConfigure( ADC0\_BASE, 64); // hardware averaging (64 samples)

// Configure ADC0 sequencer to use sample sequencer 2

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);

// Sample temperature sensor

ROM\_ADCSequenceStepConfigure( ADC0\_BASE, 2, 3,

ADC\_CTL\_TS | ADC\_CTL\_IE | ADC\_CTL\_END);

ROM\_ADCSequenceEnable( ADC0\_BASE, 2); // Enable ADC sequencer 2

**while** (1) {

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

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

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

; // wait for conversion to complete.

**ADCSequenceDataGet**(ADC0\_BASE, 2, ui32ADC0Value); // store converted data.

// Average read values, and round.

// Each Value in the array is the result of the mean of 64 samples.

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2]

+ ui32ADC0Value[3] + 2) / 4;

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096) / 10; // calc temp in C

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

// light LED 1 if temp > 80 deg-F

**if** (ui32TempValueF > 80) {

// Turn on LED at PF1

ROM\_GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 2);

} **else** {

// Turn off all LEDs

ROM\_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** "driverlib/debug.h"

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

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

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

**#define** TARGET\_IS\_BLIZZARD\_RB1 // Symbol for the API's in ROM.

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

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

**#include** "driverlib/interrupt.h" // interrupt APIs and macros

**#include** "inc/tm4c123gh6pm.h" // Define interrupt macros for device

**#ifdef** DEBUG

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

}

**#endif**

**void** **IntTimer0Handler**(**void**); // timer handler prototype

**int** **main**() {

uint32\_t ui32Period; // Period of timer

// Use 40 MHz clock

ROM\_SysCtlClockSet(

SYSCTL\_SYSDIV\_5 | SYSCTL\_USE\_PLL | SYSCTL\_OSC\_MAIN

| SYSCTL\_XTAL\_16MHZ);

// GPIO configuration

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); // enable port F

// Set LEDs as outputs

ROM\_GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE,

GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3);

ROM\_SysCtlPeripheralEnable( SYSCTL\_PERIPH\_ADC0); // enable the ADC0 peripheral

ROM\_ADCHardwareOversampleConfigure( ADC0\_BASE, 64); // hardware averaging (64 samples)

// Configure ADC0 sequencer to use sample sequencer 2, and have the processor trigger the sequence.

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

// Configure each step.

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);

// Sample temperature sensor.

ROM\_ADCSequenceStepConfigure( ADC0\_BASE, 2, 3,

ADC\_CTL\_TS | ADC\_CTL\_IE | ADC\_CTL\_END);

ROM\_ADCSequenceEnable( ADC0\_BASE, 2); // Enable ADC sequencer 2

// Timer configuration

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER0); // enable clock to TIMER0

**TimerConfigure**(TIMER0\_BASE, TIMER\_CFG\_PERIODIC); // Configure TIMER0 as 32 bit timer

// Calculate and set delay

ui32Period = **SysCtlClockGet**() / 2; // set frequency of interrupt to 2 Hz.

**TimerLoadSet**(TIMER0\_BASE, TIMER\_A, ui32Period - 1);

// Enable interrupt

**IntEnable**(INT\_TIMER0A); // enable vector associated with TIMER0A

**TimerIntEnable**(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT); // Enable event to generate interrupt

**IntMasterEnable**(); // Master int enable for all interrupts

// Enable the timer

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

**while** (1)

;

}

**void** **IntTimer0Handler**(**void**) {

uint32\_t ui32ADC0Value[4]; // ADC FIFO

**volatile** uint32\_t ui32TempAvg; // Store average

**volatile** uint32\_t ui32TempValueC; // Temperature in C

**volatile** uint32\_t ui32TempValueF; // Temperature in F

// Clear the timer interrupt.

ROM\_TimerIntClear(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT);

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

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

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

; // wait for conversion to complete.

**ADCSequenceDataGet**(ADC0\_BASE, 2, ui32ADC0Value); // store converted d/ Average read values, and round.

// Each Value in the array is the result of the average of 64 samples

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2]

+ ui32ADC0Value[3] + 2) / 4;

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096) / 10; // calc temp in C

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

// Read the current temperature. Light LED 1 if temp > 80 deg-F

**if** (ui32TempValueF > 80) {

ROM\_GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 2);

} **else** {

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

0);

}

ROM\_TimerIntClear(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT);

}

**Video Link to Demo**

[**Task**](https://www.youtube.com/watch?v=6Nrp7cJtleM) **2:** [**https://www.youtube.com/watch?v=hlk4ggL-7LQ**](https://www.youtube.com/watch?v=hlk4ggL-7LQ)

**Task 3:** [**https://www.youtube.com/watch?v=qmEybHfJPR4**](https://www.youtube.com/watch?v=qmEybHfJPR4)