Lab 3: ADC, UART, DMA, and LCD Display

Goals

1. Familiarity with the TM4C123 analog to digital converters
2. Integrating timers and interrupts into an Analog to Digital Converters (ADC) application
3. Understanding the relationship between the clock speed and power dissipation
4. Configuring a Universal Asynchronous Receiver/Transmitter (UART) interface
5. Understating DMA transfer modes, channels, triggers
6. Interfacing a Liquid Crystal Display (LCD) interface to display information for the system
7. Learning to write driver files
8. Emphasizing the ability to extract information from the datasheet to correctly setup registers

Objective of the lab

For section A, using the user switches to run the system at different clock speeds and mapping speed to the output of the internal temperature sensor of the TM4C123 LaunchPad. The different temperatures will be represented by certain color codes to be displayed on the board's LEDs.

For section B, configuring a UART interface.

For section C, using DMA to transfer data.

For section D, connecting a Liquid Crystal Display (EB-LM4F120-L35) and develop a sketch for it.

What you need for the lab

1. The EK-TM4C123 Launchpad (http://www.ti.com/tool/EK-TM4C123GXL)
2. TM4C123 data sheet (https://canvas.uw.edu/courses/1205180/files/folder/Ek-TM4C123GXL?preview=49165887)
3. Lecture 7 slides
4. IAR workbench or other IDE
5. ADC (https://en.wikipedia.org/wiki/Analog-to-digital_converter)
6. DMA (https://sites.google.com/site/luiselectronicprojects/tutorials/tiva-tutorials/tiva-dma/understanding-the-tiva-dma)
7. UART (https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter)
8. LCD (EB-LM4F120-L35)(http://www.kentecdisplay.com/uploads/soft/Products_spec/EB-LM4F120-L35_UserGuide_04.pdf)

Section A. ADC

The ADC Module

The TM4C123GH6PM ADC module features 12-bit conversion resolution and supports 12 input channels plus an internal temperature sensor. The ADC module uses a Successive Approximation Register (SAR) architecture to deliver a 12-bit, low-power, high-precision conversion value. The range of this conversion value is from 0x000 to 0xFFF (0 to 4095). It uses internal signals VREFP and VREFN as references to produce a conversion value from the selected analog input. VREFP is connected to VDDA and VREFN is connected to GNDA, as shown in Figure 1

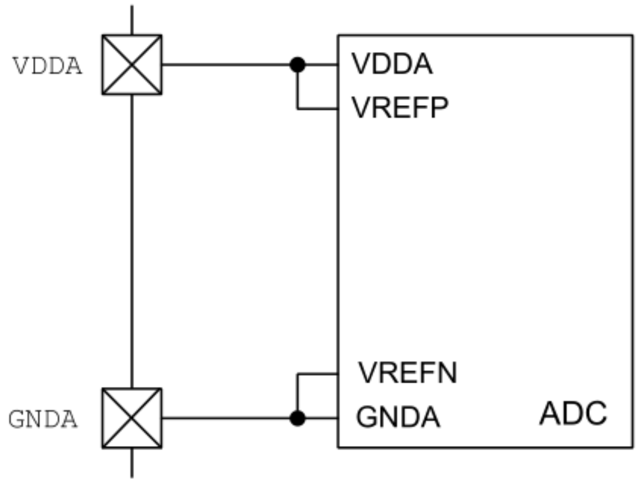

Figure 1 ADC voltage (figure 13-8 of the datasheet)

This configuration results in a resolution that can be calculated using the following equation:

mV per ADC code = (VREFP - VREFN) / 4096

Please note that to produce accurate results, the analog input voltages must be within the limits described in table 24-33 p.1389 of the data sheet.

The Internal Temperature Sensor

The built-in internal temperature sensor of the TM4C123 LaunchPad notifies the system when the internal temperature is too high or low for reliable operation. It converts a temperature measurement into a voltage V_{TSENS} according to Figure 2

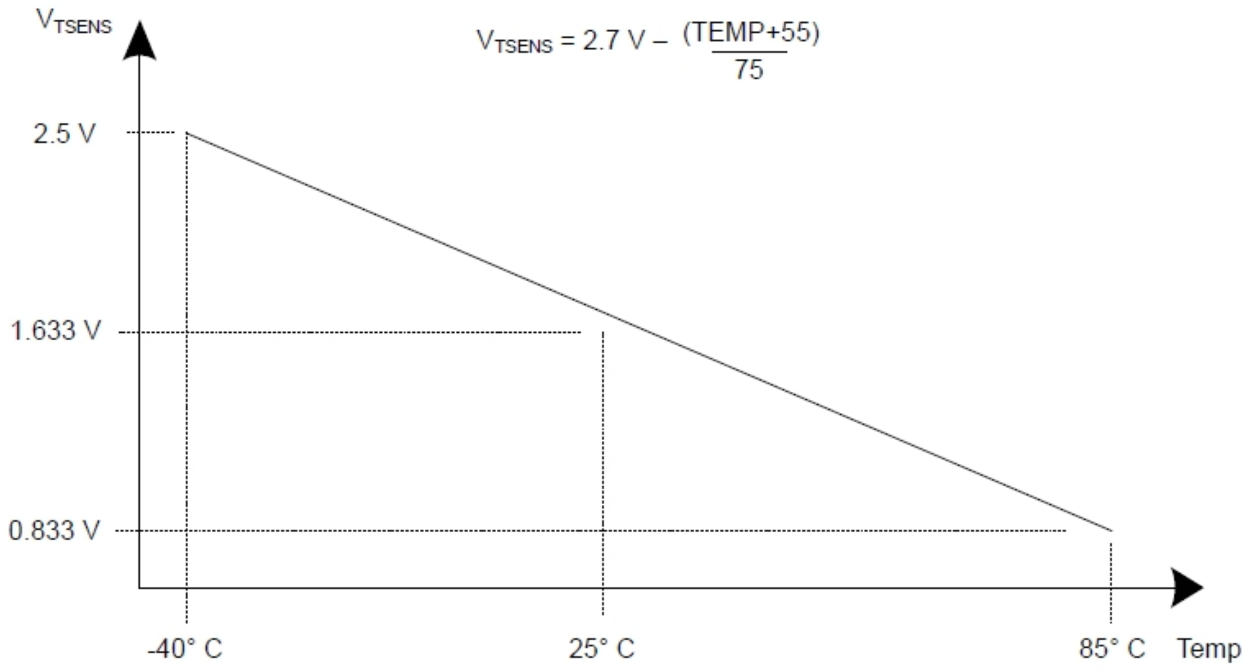

Figure 2 Internal Temperature Sensor Characteristic (figure 13-11 of the datasheet)

The temperature reading from the temperature sensor is a function of the ADC value. The following formula calculates temperature (TEMP in □) based on the ADC reading (ADC CODE, given as an unsigned decimal number from 0 to 4095) and the maximum ADC voltage range (VREFP - VREFN).

$$\text{TEMP} = 147.5 - ((75 * (\text{VREFP} - \text{VREFN}) \times \text{ADC CODE}) / 4096)$$

where the ADC CODE is the output of the ADC in the **ADCSSFIFOn** register.

Section A Required Task

Your task is to control the microcontroller's clock speed from the user switches and find the corresponding temperature generated by the internal temperature sensor. When switch PF0 is pressed, the system clock should operate at 4MHz and if the other switch is pressed, the system clock should operate at 80MHz. A timer should trigger the ADC to read a sample every 1 second and find the corresponding temperature of the CPU. The temperature obtained will be displayed by the LEDs based on the following temperature scale

Color	PF3 PF2 PF1 value	Temperature in Celsius
Red	001	0-17
Blue	010	17-19
Violet	011	19- 21
Green	100	21-23
Yellow	101	23-25

Light Blue	110	25-27
White	111	27-40

Your program could be structured to contain the following functions:

1. **Timer0_Init** function: initialize the timer per instructions are given in lab 2. The GPTMCTL register should be configured to enable the timer to trigger ADC. Since the clock will not be fixed in this lab, it's better to pass the maximum counter value to this function based on the clock used. For example, in lab 2 we set the counter value to 16,000,000 for a 16MHz clock so we fixed this value in the Timer0 module which is not the case in this lab.
2. **PortF_Init** function: initializes the GPIO Port F Pins to accept input from the switches and display output to the LEDs
3. **PLL_Init** function: initializes the Phase Lock Loop module of the TM4C123 microcontroller based on the steps explained in lecture 4 slides.
4. **ADC_Init** function: configures the ADC module based on the instructions explained in the datasheet and lecture 7 slides. Note that you may need to insert some delay after the RCGCADC register is configured as the ADC sometimes requires more than one clock cycle to work especially at high clock rates. Make sure to configure the appropriate registers to enable interrupts and allow the ADC to be triggered by the timer. You also need to update the vector table in the cstartup_M.c file accordingly
5. **ADC0_Handler**: an interrupt service routine for the ADC0 module. You will use sequencer 3 since you only need one input. In this handler the temperature will be calculated according to the internal temperature sensor equation and based on the value, the appropriate color should be displayed on the LEDs. Make sure to clear the ADC flag after each conversion.
6. **the main** function: calls initialization functions and checks on the switches all the time

Section B. UART

Universal Asynchronous Receiver/ Transmitter (UART) is a commonly used serial interface. You will be communicating with your board via the board's built-in UART communication tool in this class. For this to work, you will need a program on your computer that can serially communicate with the board.

We recommend using a program called PuTTY. Once you have PuTTY installed, we need to set up a communication link with your Tiva C series board. You will need to make sure that the settings on your board match the communications settings in PuTTY. To do this, find your Tiva board in Windows Device Manager. Make note of the COM port that your board is associated with; it should say next to the board. Right-click on your board and click on properties. Navigate to the "Port Settings" tab. Now, in PuTTY, set the "Serial Line" field to the COM port you found earlier; for example, mine is currently COM3. Set the Speed to match your board—the standard is usually 9600. Set the connection type to "Serial", since what you will be using, UART, is a serial communication protocol. Verify the rest of the settings on your board match the PuTTY settings under the "Serial" menu in the lower left. From here, we are leaving it up to you to figure out how to get the UART link working with your board. We recommend starting by investing PA0 and PA1. The datasheet has all the information you need for getting the communication up and going, so use the skills that you've gained this quarter!

Section B Required Task Deliverables

We want you to integrate UART communication with section A. Find a way to get your temperature result to print to the PuTTY console.

For an extra badge, implement a way to send a clock frequency to the board instead of having to press the buttons to change it.

Section C: DMA

Direct Memory Access (DMA) allows peripherals to access the memory without accessing through the processor. Commonly, you need to save all data into the memory for later usage. Alternatively, we could use DMA to access the data while the processor works on other operations. Please read this post (https://sites.google.com/site/luiselectronicprojects/tutorials/tiva-tutorials/tiva-dma/understanding-the-tiva-dma) for more details.
Please download and run DMATimerPortRead_4C123.zip from the Canvas (https://canvas.uw.edu/courses/1205180/files/folder/labs).

Section C Task

Modify the code and blink the three LEDs (one yellow, one green, and one red) in the kits every one second.

Section D: LCD

In this section, we use EK-TM4C123GXL to drive a touchscreen LCD (EB-LM4F120-L35). This 3.5'' LCD uses 8-bit parallel interface to communicate with the TIVA.

Hardware

To get started, plugging the LCD onto the top of the TIVA as the figure 3 shows. The datasheet of EB-LM4F120-L35 can be found here (http://www.kentecdisplay.com/uploads/soft/Products_spec/EB-LM4F120-L35_UserGuide_04.pdf). We do not need any jumper wires for the hardware setup.

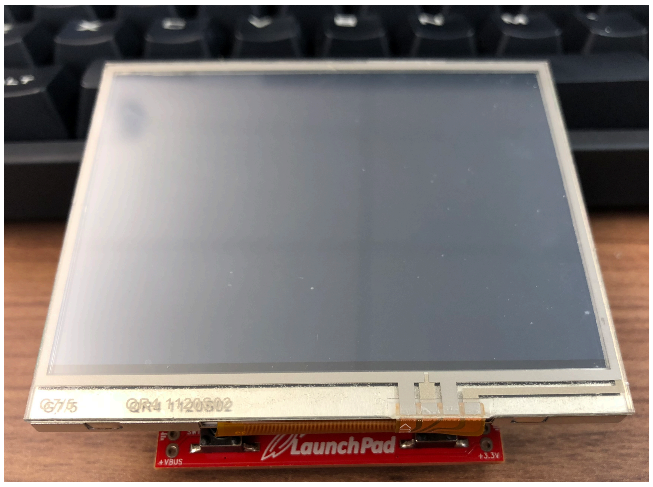

Figure 3 Connecting LCD with TIVA

Write LCD driver

Download and run the starter files from https://canvas.uw.edu/courses/1205180/files/folder/labs, the file name is lab3_starter.

```
// Runs on LM4F120/TM4C123

// Driver for the SSD2119 interface on a Kentec 320x240x16 BoosterPack

// - Uses all 8 bits on PortB for writing data to LCD

//   and bits 4-7 on Port A for control signals


// Data pin assignments:

// PB0-7   LCD parallel data input


// Control pin assignments:

// PA4    RD  Read control signal          -------------------------

// PA5    WR  Write control signal          | PA7 | PA6 | PA5 | PA4 |

// PA6    RS  Register/Data select signal    | CS  | RS  | WR  | RD  |

// PA7    CS  Chip select signal           -------------------------


// Touchpad pin assignments:

// PA2    Y-                    -------------  -------------

// PA3    X-                    | PA3 | PA2 |  | PE5 | PE4 |

// PE4    X+   AIN9                | X-  | Y-  |  | Y+  | X+  |

// PE5    Y+   AIN8                -------------  -------------
```

Section D Task

1) Filled the functions (that marked as TODO) in header file SSD2119.c , see comments for details.
   - Dimensions of the LCD in pixels
   - LCD_GPIOInit
   - LCD_WriteCommand
   - Touch_Init

2) Display the temperature data from section A on the LCD, which is the same information that displayed using URAT in section B.

3) Draw a cube in the center of the screen, the length, width, and depth of the cube are all 0.6 inches. Let the cube rotate, and use the touchscreen to start and stop the rotation of the cube. Print the coordinates of the touch point on the LCD.
4) Draw the same cube in the center of the screen as the sub-task 1. The length, will , and depth of the cube are all 0.6 inches. We fill the cube with the color white. Let the cube rotate. For this sub-task, whenever the touchscreen is pressed, the fill color will change once. You can use any colors for each press, but the colors need to change for each press. Print the RGB color code on the screen.
5) FSM
   i) Recall the traffic light controller in lab 2. Replace the physical al push buttons with two virtual buttons that displayed on the LCD. Let's call the start/stop button as virtual button 1, and the button for the passenger as virtual button 2 for the following sections. You can display the button either vertically or horizontally. When a virtual button is pressed, the system will respond only if the user holds down the button (on the LCD) at least 2 seconds.
   ii) If the user presses the virtual button 1 (start/stop button, hold for 2 seconds), but not the virtual button 2 (passenger button), the system will start with the stop stage (where the red LED is on, and other LEDs are off). After 5 seconds, the system will move to go stage. Then wait for another 5 seconds to change from go stage to stop stage. In other words, the go and stop stage will last for 5 seconds and switch to each other.
   iii) If the user presses the virtual button 2 (passenger button, hold for 2 seconds) to indicate a passenger tries to across the street, the system will stop the current stage and move the warn stage immediately. The warn stage will last 5 seconds, and move to stop stage.

Deliverables:

1. A lab report along
2. Demonstration of the circuit to your TA
3. Upload the report and source files for both sections to the canvas. One submission is expected per team.