Lab 4: DMA and LCD Display

Goals

1. Improve familiarity with the TM4C123 analog to digital converters
2. Understand DMA transfer modes, channels, triggers
3. Interface with a Liquid Crystal Display (LCD) interface to display information
4. Learn to write driver files
5. Learn to extract information from the datasheet to correctly setup registers

Objective of the lab

In Section A, transfer data by use of Direct Memory Access or DMA.

In Section B, Interface an LCD into your the system such that it can be used to display information. Furthermore, you'll learn to design, implement, and test a device driver.

What you need for the lab

1. The EK-TM4C123 Launchpad (http://www.ti.com/tool/EK-TM4C123GXL)
2. TM4C123 data sheet (In canvas)
3. Slides from in-class lecture 3 and 6. (In canvas)
4. IAR workbench or other IDE
5. DMA
6. LCD (EB-LM4F120-L35)(http://www.kentecdisplay.com/uploads/soft/Products_spec/EB-LM4F120-L35_UserGuide_04.pdf)

Section A: 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. You may refer to chapter 9 of the datasheet for more information on setting up the DMA on the Tiva board. For this lab, you are provided with the driver code for the DMA.

Download DMA_new.zip from the Canvas.

Section A Task

The DMA code provided flashes 8 colors in every 10ms. Your task is to modify the code to individually blink the three on-board LED colors (one yellow, one green, and one red) in one second intervals. Continuously cycle through the 3 LEDs forever.

Section B: LCD

In this section, we use the TIVA Launchpad EK-TM4C123GXL to drive a touchscreen LCD (EB-LM4F120-L35). This 3.5’’ LCD is a booster pack that plugs directly into the LaunchPad (see Figure 1). This interface employs an 8-bit data bus, and uses Port A pins 7-2, Port B pins 7-0, and Port E pins 4 and 5. The touchpad interface utilizes Port A pins 2 and 3 and also Port E pins 4 and 5. The display interface uses PA8-PA4 and all of Port B pins.

Hardware

To get started, plug the LCD female headers into the male headers on the TIVA board so that it looks like figure 1. The datasheet of EB-LM4F120-L35 can be found here (http://www.kentecdisplay.com/uploads/soft/Products_spec/EB-LM4F120-L35_UserGuide_04.pdf).

Note: You will NOT need any jumper wires for this section’s hardware setup.

Note 2: In order to use the LCD display, we needed to remove the R9 and R10 resistors on TIVA. These two resistors should have already been removed before the TIVA boards were distributed. Please double check that those resistors have been removed prior to connecting the LCD to the Launchpad. You can look at the TIVA schematic in the TI User Manual (on canvas) to find out which resistors these correspond to. If you are unsure, double check with your lab mates or TA. If the resistors have not been removed DO NOT attempt to remove them yourself, and inform your TA.

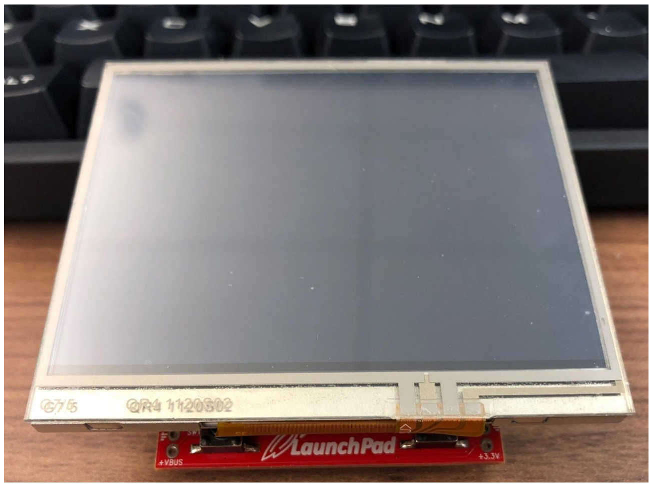

Figure 1 Connecting LCD with TIVA

Write to the LCD driver

The objective of this section of the lab is to develop a device driver for the LCD display. A device driver contains function definitions and appropriate I/O port mappings that are needed to ensure proper functionality of a device. A proper device driver creates an easy to use interface between different electronic systems. The idea is to create functions that can take an input from one electronic domain and translate it into some kind of meaningful data or action in the other electronic domain (in this lab the domains would be the TIVA board and the LCD).

Download and run the starter files from canvas, the file name is lab4_starter.

When looking through the provided code, *pay extra attention* to the comment block below.

```
// 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 B Task

1) Open the provided SSD2119.c and fill-in the missing parts of the functions (that are marked "TODO"). Specifically:

- Dimensions of the LCD in pixels
- LCD_GPIOInit

Note: The LCD_GPIOInit (initialization) will be similar to the previous port initializations: activate the clock, configure the direction register, turn off alternate functions, and enable the data pins. All four control signals, PA7-PA4, should be initially high. The code below should point you in the right direction for the LCD_GPIOInit function definition.

```
;           ************** LCD_GPIOInit ****************************

;           Initializes Ports A and B for Kentec EB-LM4F120-L35
;           1) Port A bits 4-7 are output to four control signals
;           2) Port B bits 0-7 are output data is the data bus
;           3) Initialize all control signals high (off)
;           PA4 RD Read control signal
;           PA5 WR Write control signal
;           PA6 RS Register/Data select signal
;           PA7 CS Chip select signal
;           |--------------------------------|
;           |PA7  | PA6 | PA5  | PA7  |
;           |CS    | RS    | WR  | RD    |
;           |--------------------------------|
;           4) wait 40 us
;           Invariables: This function must not permanently modify registers R4 to R11
```

Note 2: Your **LCD_GPIOInit** function will be called from the **LCD_Init** function. The **LCD_Init** function is given to you in the **SSD2119.c** file. This initialization sequence contains over 50 steps, and we do not expect students (or professors) to understand the details of this 50-step initialization. We copied these steps from the code provided in the Kentec starter application (we downloaded grlib_demo(K350).rar from http://www.kentecdisplay.com.)

Note 3: Regarding the **LCD_WriteCommand** in the **SSD2119.c** file, this function (we provided to you) will be used to output 8-bit commands to the LCD, and the **LCD_WriteData** function (we also provided for you) will be used to output 16-bit data to the LCD. Figures 2 illustrate the timing diagram for writing to the LCD if you are interested.

```
************** LCD_WriteCommand ************************

- Writes an 8-bit command to the LCD controller
- RS low during command write
8-bit command passed in R0
1) LCD_DATA = 0x00;  // Write 0 as MSB of command;
2) LCD_CTRL = 0x10;  // Set CS, WR, RS low;
3) LCD_CTRL = 0x70; // Set WR and RS high;
4) LCD_DATA = command; // Write 8-bit LSB command;
5) LCD_CTRL = 0x10; // Set WR and RS low;
6) wait 2 bus cycles // Set CS, WR, RS high;
7) LCD_CTRL = 0xF0;
Invariables: This function must not permanently modify registers R4 to R11
```

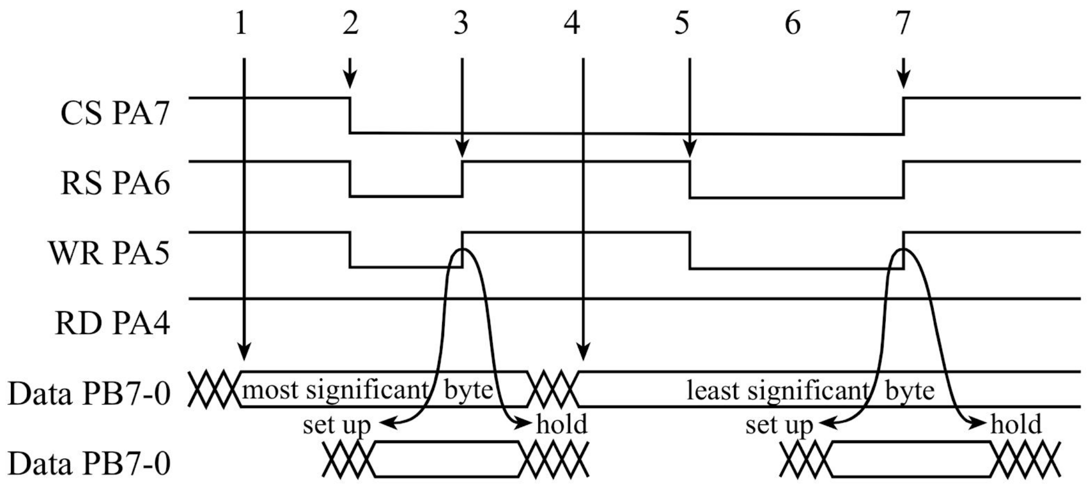

Figure 2. Timing diagram of a 16-bit write command to the LCD, ***LCD_WriteCommand*** *(RD=PA4 is always high). The most significant byte of the command will be 0, and the least significant byte contains the actual command. The first 5 lines describe what will be driven by the microcontroller, while the last line describes what is needed by the LCD.*

2) In lab 3 you displayed the TIVA board's temperature in PuTTy over UART. For this lab, **instead** of transmitting the temperature data over UART, display it on the LCD using the LCD driver functions. You can keep the internal clock at 16 MHz.

3) Next, when you touch a point on the LCD with your finger, print the coordinates somewhere on the LCD at the same time. Have this continuously reading your finger position and printing its coordinates.

4) Next, (in code) draw a cube (**not** square) on the LCD in the center of the screen with a length, width, and depth of 0.6 inches.

5) Take the same cube you made in the previous subtask, but now whenever you press the screen (press and release) the cube will fill with a color. Each time you press the screen, the cube will be filled with a new color and stay that color until there is another press. Cycle through R, G, and B indefinitely.

6) FSM - *you will need the traffic controller breadboard circuit from lab 2*

   i) Recall the traffic light controller in lab 2. Instead of physical buttons on a breadboard, we are going to create 'virtual' buttons on the LCD. These buttons should be displayed as a shape, and perform functions when they are pressed on the LCD. Virtual button 1 should now be the start/stop button for the traffic controller, virtual button 2 should be the pedestrian button. You can display these buttons either vertically or horizontally. In this section of the lab, a virtual button will only be considered pressed if the user holds down the virtual button (on the LCD) for at least 2 seconds then releases (use a timer).

   ii) If the user presses virtual button 1 (start/stop button, hold for at least 2 seconds then release), but not the virtual button 2 (pedestrian button), the traffic light will begin in the Stop State (where the red LED is on, and other 2 LEDs are off). After 5 seconds (timer), the system will move to Go State. Then wait for another 5 seconds to change from Go State to Stop State. (This functionality isn't anything new from lab two, so use that as a reference if you need a refresher). Remember that Stop, Go, and Warn all abide by the virtual button 1's off function.

   iii) If the user presses virtual button 2 (pedestrian button) and holds the button for 2 seconds then releases (while in the Go State). The system will immediately move the Warn State (Yellow led). The Warn State will last 5 seconds, and then the system will move back to the Stop State unless it was turned off with virtual button 1. (Again, this functionality isn't anything new from lab two, so use that as a reference if you need a refresher).

Extra Credit:

Add-on to part 5 (this part will help if you intend to do animations for your final project):
Using the filled in cube you made previously, add a function where you can start and stop the rotation of the cube depending on the location where you touch the LCD (make the speed slow enough and rotation angle step size small enough that the TAs can visually tell that it's rotating). You only need to rotate in 1 dimension (aka the x-plane or y-plane). The cube doesn't need to change colors for this demo. Touch the screen once in the LCD location that corresponds to rotating and the cube should rotate indefinitely, touch the stop location once and it will stop indefinitely. The start and stop locations should be different locations on the LCD. Don't hold your finger down, just press and release like a button.

Deliverables:

1. Demonstrate to the TAs the system operating according to the specifications. When demoing make sure the LCD touch interface and display is not glitchy. Also, ensure your refresh rate is at an appropriate speed so your lab looks professional (doesn't have to be the fastest possible rate but if it is noticeably slow you will have to adjust your code - ask yourself "would I invest in this product").
2. A Lab Report uploaded to Canvas; clearly written and including source files for Sections A and B.