

---

**ECE 3803: Microprocessor System Design**  
**Term D, 2007**  
**Laboratory 2 – LCD and Keypad**

Notes: Plan for at least 15 hours of laboratory time. You should work in groups of two. Submit one lab report for your group.
---

Posted: Monday, 3/26/2007  
Signoff: By 4:50pm Thursday, 4/12/2007. Early signoff earns extra credit.  
Report: In class on Monday 4/16/2007. **Do not forget to attach the signoff sheet.**

**Reading/reference material**

- Textbooks: Furber 11.9, Wolf 3.2, 4.4
- Component datasheets (see course website)

**Concepts**

- I/O peripherals
- Address decoding
- Bus timing analysis
- Initialization code
- Device driver software

**Downloads**

- LCD text/graphics library (see course website)

At this point, you should have a functional ARM-based system capable of running software from non-volatile memory. In part 2 of the ECE 3803 labs, you will add a basic user interface consisting of an LCD display and a keypad to your system. In addition, you will write device driver type software to allow application software to interact with your new peripherals without worrying about their implementation details. An optional (extra credit) part of this lab invites you to implement an application of your choice using the lab hardware.

## Problem statement

Your TA will supply you with additional parts:

Quantity	Description
1	LCD module graphic 122x32 w/LED backlight
1	Keypad 3x4
1	74HC139 dual decoder 2:4
1	74AHC574 octal D flip-flop
1	74AHC573 octal transparent latch
1	74AHC05 hex open-drain inverters
1	74HC14 hex Schmitt trigger inverter
1	Potentiometer 20k
1	Resistor 1.6k $\Omega$
4	Resistor 1.1k $\Omega$
2	Capacitor, 0.1 $\mu$ F
1	p-channel power MOSFET
1	Resistor 6.2 $\Omega$
1	Resistor 100k $\Omega$
1	Pushbutton

You are to add graphical display and keypad input capabilities to your Lab 1 system. You may use components other than the ones provided at your own expense, but within reason. You are expected to perform (and document) timing analysis and set wait states to ensure proper bus operation with each attached device under worst-case conditions.

You are to write device driver type software that provides a high-level interface to the LCD screen and the keypad.

You are already provided with a set of routines that implement text output and simple graphics (points/lines/circles). This software outputs pixels to a frame buffer in RAM. This is simply an array that associates a bit with each pixel, specifying whether the pixel is black or white. You will need to write a function that outputs the contents of this RAM frame buffer to the LCD screen. Ideally, this transfer should occur at regular time intervals transparently to the main program. Using a timer interrupt for this purpose is an extra credit assignment.

The keypad interface should consist of a function to determine if a key has recently been pressed, and a function that returns the ASCII code corresponding to that key. Another function that returns the status of each key independently (one bit per key) may be useful. Ideally, scanning the keypad for new key presses should occur at regular time intervals transparently to the main program. Again, this is part of an extra credit assignment. A good keypad driver is aware of the possibility of button contacts bouncing when pressed or released. Timing can be used to prevent contact bounces from registering as multiple consecutive key presses.

You will need to implement a minimal application that outputs keypad key presses as text on the LCD screen.

In addition, you are to add a button to control the backlight of the LCD. When the button is pressed, the LCD backlight should toggle from OFF to ON, or vice versa. Make sure that the button is de-bounced (only one press counted at a time). Keep in mind that the output of a typical logic IC **cannot** supply enough current for the LCD backlight. For this reason, you are supplied with a power MOSFET.

An additional extra credit assignment is implementing an application of your choice on this system. Your application will be judged based on difficulty and creativity, which are unfortunately mostly subjective criteria (but the maximum number of bonus points is small; see the Signoff Sheet).

The details of the implementation of this lab are left up to you. Here are some tips to help you along:

1. You will need an address decoder to accommodate the I/O ports and the LCD. It is suggested that the EEPROM use NCS0, and the peripherals use NCS1. This allows separate wait state settings for EEPROM and peripherals.
2. The LCD module supports two bus interface modes: the “80-series” compatible mode that can use the NRD and NWR signals from our CPU, and the “68-series” compatible mode that uses an R/W (read/write) select signal and an E (enable) strobe. To configure the LCD controller in the “80-series” compatible mode, the LCD reset signal should be treated as *active high* (see pages 2-6 and 2-8 of the LCD driver data sheet). The LCD controller actually resets on any edge of the reset signal, and the subsequent level of reset tells it what bus mode to use. The CPU, if you recall, uses an active low reset signal. Additionally, Lab 1 created a slow-rising reset signal. To allow the CPU and the LCD to reset simultaneously, as well as allow JTAG to reset the system, a new reset circuit is needed. Try to design such a circuit that satisfies the following criteria:
  - a. The CPU sees an active low reset.
  - b. The LCD sees an active high reset.
  - c. The reset signal transitions should be rapid, and happen at the same time for the CPU and LCD (within a small logic delay).
  - d. The system should stay in reset for about 1.5 sec after power-up.
  - e. Reset can be triggered by a reset button.
  - f. Reset can be triggered by an open-drain reset driver in the JTAG interface. This driver is directly connected to the NRST line of the CPU. It is only capable of pulling NRST low. When inactive, the driver output becomes high impedance instead of driving NRST high.
3. The LCD controller requires a clock signal to time the refresh of the pixels on the screen. The datasheet specifies a clock of 2 kHz, but we found this produced too much flicker. Implement a simple clock generator using a Schmitt trigger inverter, capacitor and resistor. Find a good clock frequency (up to 5 kHz) that results in the best looking image with no flicker on the LCD.

4. Your keypad scanning software should correctly handle cases where multiple buttons are pressed at the same time. An ASCII character should be returned whenever a new key press is detected, even if other keys are still pressed.

### **Laboratory Assignment**

1. Draw a detailed schematic of your ARM-based system, including all pin numbers.
2. Develop an assembly plan and troubleshooting strategies for your hardware/software. Include these plans in your report and explain how effective they were.
3. Wire, debug, and test each component of your hardware system.
4. Write the required lab software that demonstrates all the required functionality of your system.
5. Demonstrate the proper operation of your system to the TA, who will fill in and sign the signoff sheet. Try to sign off each step as soon as you get it working. Include the signoff sheet with your report.
6. Document your complete hardware and software design, bus timing analysis, assembly/troubleshooting plan, and testing results in your lab report. Include all source code.

---

Developed by Ghaith Hammouri (hammouri@wpi.edu) and Gene Bogdanov (gene@wpi.edu),  
March 2007.

**ECE 3803 D2007 Lab 2**  
**Signoff Sheet**  
**Due date: 04/12/2007**

**Names**

**Student 1:** \_\_\_\_\_ **ECE mailbox:** \_\_\_\_\_

**Student 2:** \_\_\_\_\_ **ECE mailbox:** \_\_\_\_\_

Requirement	Max points	TA's assessment	TA's initials
LCD operational: display a welcome message that occupies both halves of the screen.	20		
8-bit digital output port operational. Demonstrate using the oscilloscope.	10		
Keypad operational. Demonstrate using the debugger.	10		
Button controlled LCD backlight on/off feature	5		
Key presses show up as characters on the LCD	5		
All required device driver functions implemented.	10		
Wiring neatness	10		
<b>Extra credit:</b> LCD and keypad drivers use timer interrupts to perform their refresh and scan functions.	3		
<b>Extra credit:</b> An application of your choice running on your hardware.	3		
Report	30		
Total points	106		

Notes:

1. To pass this lab, you need have a minimally functional LCD, and a minimum signoff score of 35 (excluding report).
2. Maximum score: 111.3 when taking advantage of maximum early signoff bonus.

**TA's signature:** \_\_\_\_\_ **Date:** \_\_\_\_\_