# ELEC 3040/3050 Lab 5
# Alternate Assignment

## The Directional Input Pad

The purpose of this lab is to use the microcontroller to control a peripheral device, interfaced through parallel I/O ports and accessed using interrupt-driven I/O. The peripheral device for this lab is a directional input pad (D-pad), as used on a variety of products (game consoles, cell phones, etc.). The D-Pad is pictured in Figure 1. Note that there are 5 pins on the side of the keypad. These are designed to be placed directly into a standard breadboard. The D-pad is designed to electrically mimic the keypad simulated in *CodeWarrior* that has been used in numerous ELEC 2200 projects. However, it should be noted that the D-pad only contains two columns and three rows, rather than four rows and four columns. Refer to the keypad scanning example in Chapter 18 of the Cady text book (Hardware and Software Engineering), or Chapter 14.8 of the Y. Zhu text book (Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C, 2nd Ed.), or Chapter 8.4 of the Valvano text book (Introduction to ARM Cortex-M Microcontrollers). One of these three books will have been used in ELEC 2220. Again, keep in mind: the same keypad scanning algorithm will work for both the 4x4 standard keypad and the 3x2 provided D-pad for this lab.

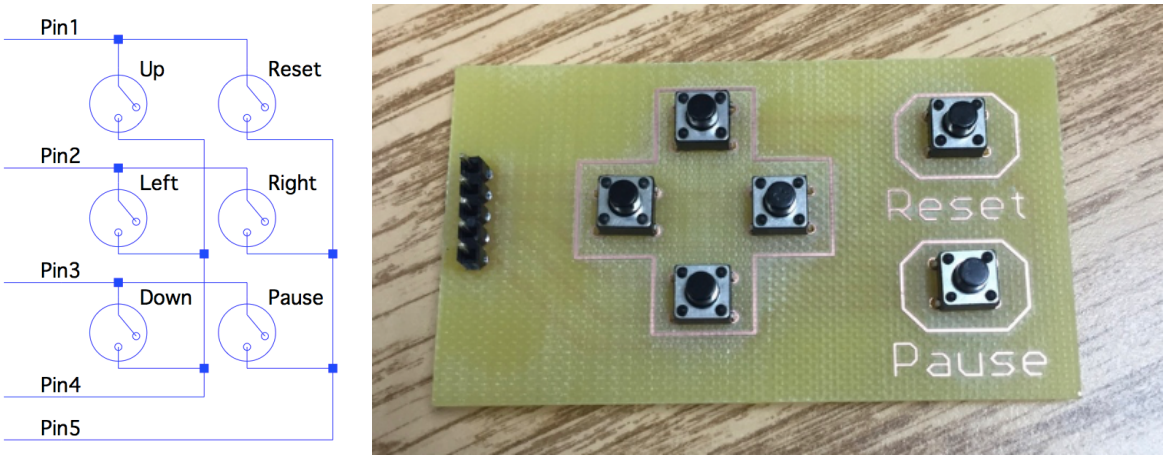


Figure 1: Directional Input Pad with Two Face Buttons

| D-pad Button | Opcode |
|---|---|
| **Up** | 0x01 |
| **Left** | 0x02 |
| **Right** | 0x04 |
| **Down** | 0x08 |
| **Pause** | 0x10 |
| **Reset** | 0x20 |

Table 1: D-pad Button Opcode Values

## Interfacing the D-Pad to the Microcontroller

Refer to the presentation slides for the Monday lab lecture for a description of keypad operation and interfacing to a microcontroller. Keypad scanning examples are provided in Chapter 18 of the Cady text book and Chapter 8.4 of the Valvano text book.

## Lab Objective

For this lab, the "main program" is to display a binary-coded decimal number on 4 LEDs, with the number incrementing approximately once per second in an endless loop, and rolling over from 9 to 0. If a D-pad button is pressed, as detected via an interrupt, the D-pad scanning routine should identify and display the button's opcode on the LEDs, instead of the count. This opcode number should remain on the LEDs for approximately 5 seconds, and then the program should resume displaying the incrementing count on the LEDs. The counter should not stop during these 5 seconds, so when counter display resumes, the count should be approximately the time of the interrupt plus 5. (Hint – if using interrupts, the interrupt service routine can set a global variable that can be tested by the main program to see if a D-pad opcode is being displayed.

## Pre-Lab Assignment

### Reading

Review the Lab 5 lecture slides and the previous labs on GPIO pins and interrupts. In addition, the textbook chapters listed in the first paragraph of this document include examples of matrix keypads which implement a similar algorithm.

### Hardware Design

The I/O port connections to the "peripheral devices" should be made as listed in Table 1. GPIO pins PC[3:0] are to drive the four LEDs displaying the incrementing count. GPIO pins PB[5:4] and PB[2:0] are to be used for the D-pad interface. To generate an interrupt signal on a button press, a 4-input AND gate (CMOS 4082B chip) will be needed to combine the row signals to drive the ***IRQ#*** line (GPIO pin PA1). You may use additional LEDs for debugging, if you wish. Note that internal pull-up resistors should be activated for the GPIO pins connected to the D-pad row lines.

| GPIO Pins | Connected Devices |
|---|---|
| **PB [2:0]** | D-Pad row lines 3-1 (inputs) |
| **PB [5:4]** | D-Pad column lines 2-1 (outputs) |
| **PC [3:0]** | LEDs (for the counter) |
| **PA1** | IRQ# |
| **Other pors** | LEDs for debug, as needed |

Table 2. Parallel Input/Output port connections

In your laboratory notebook sketch a diagram that corresponds to the connections described in Table 1. Show details of how the microcontroller, D-pad, 4-input AND gate, and EEBoard (test instruments) are to be connected. ***Do this prior to lab.***

***To aid in debugging, be prepared to use the logic analyzer and/or oscilloscope to observe the states of the various peripheral device lines.***

### Software Design

Review the earlier labs to recall how to initialize and access I/O ports, set/clear/test individual bits of a word, and set up interrupt-driven operation. Thoroughly comment your program to demonstrate your understanding of the keypad scanning operation.

The test program should comprise a main program and an interrupt service routnine. The main program should configure all GPIO ports used, configure the interrupt request pin and NVIC, initialize the column lines of the keypad to all 0's, enable interrupts, and then enter a continuous loop. Note that all four column lines should initially be driven low so that any pressed key will cause one of the four row lines to go low and trigger an interrupt. In the main program's continuous loop, the 4-bit counter should be incrementing once per second, and displayed on the 4 LEDs. The keypad scanning routine should be executed if the CPU is interrupted by a pressed key. If a pressed key is detected, the key number should be displayed on the LEDs, instead of the count, for approximately 5 seconds. Refer to the discussion above for a description of the D-pad scanning process.

In your laboratory notebook, record the following ***prior to the lab.***

1. Flowcharts for program and the interrupt service routine
2. Draft program and the interrupt service routine (or directions to content on H: drive)
3. A plan for testing the D-pad

***NOTE: It has been found that there is often a short delay between writing a pattern to an output port and observing that pattern on the output pins. Therefore, you should insert a few "dummy" instructions following a write to the output port driving the columns, before testing the input port driven by the rows.***

## Lab Procedure

1. Double-check the ground connection between the Discovery board and the EEBOARD.
2. Mount the D-pad and AND gate chip on your breadboard and connect them to the microcontroller as shown in Table 1. You should also connect the keypad signals and AND gate output to DIO pins, so that you can use the logic analyzer for debugging. Note that you should activate the internal pull-up resistors for the input port hardware.
3. Enter, compile, and download your D-pad scanning program. Execute your program, verifying the correct opcode is displayed for each of the 6 buttons. ***(Record your observations of the reaction to each button press in your notebook)***.
4. During debugging, verify that you can access the GPIO ports properly, using the test methods learned in previous labs. If you experience problems, you might consider testing the I/P ports using the test programs from Labs 2-3-4 and the logic analyzer.
5. Demonstrate your working programs to the lab instructor.
6. If time permits, investigate the button bouncing with the oscilloscope to determine if, and for how long, bouncing occurs following a button press.

## Possible Information to Include in Future Lab Reports

1. Briefly describe your hardware design. Include a schematic diagram
2. Include a printout of your C program, including well thought through comments.
3. Provide two logic analyzer screen captures, one showing where the LEDs changed from an incrementing count to a button's opcode and the other showing the IRQ pin and the D-pad row and column immediately following a button press. (This should show your scanning algorithm being executed.)
4. Provide an oscilloscope screen capture, showing the IRW pin or one of the row lines following a button press, to determine if there is any "bouncing" associated with the key press, and if so, how long the bouncing lasts.
5. Discuss your results.