LAB 5: MATRIX KEYPAD INTERFACE USING PARALLEL I/O

THE VELLEMAN 16-KEY MATRIX KEYPAD

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 matrix keypad, as used on a variety of products (phones, keyless entry systems, appliances, etc.) The keypad is pictured in Figure 1. Note that there are 8 pins on the bottom of the keypad. A ribbon cable has been soldered to these, with the other end connected to an 8-pin header that can be inserted into a standard breadboard. This keypad closely resembles the simulated keypad in *CodeWarrior* that has been used in some ELEC 2200 projects. 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.

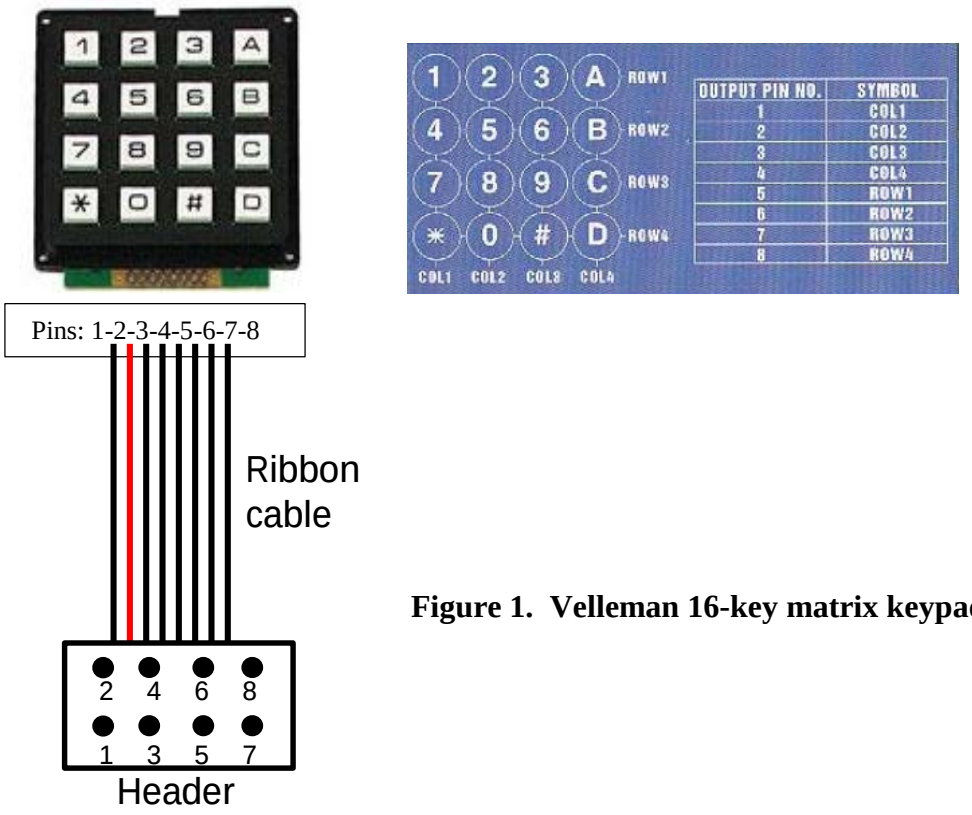

Figure 1. Velleman 16-key matrix keypad

INTERFACING THE MATRIX KEYPAD 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 also 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 key is pressed, as detected via an interrupt, the keypad scanning routine should identify and display the key number on the LEDs, instead of the count. This key 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 key number 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 similar matrix keypads.

Hardware Design

The I/O port connections to the "peripheral devices" should be made as listed in Table 1. GPIO pins PC3-PC0 are to drive the four LEDs displaying the incrementing count. GPIO pins PB7-PB0 are to be used for the keypad interface. To generate an interrupt signal on a key 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 keypad row lines.

GPIO Pins	Connected Devices
PB3-PB0	Keypad row lines 4-1 (inputs)
PB7-PB4	Keypad column lines 4-1 (outputs)
PC3-PC0	LEDs (for the counter)
PA1	IRQ#
Other ports	LEDs for debug, as needed

Table 1. 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, keypad, 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 routine. 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 count 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 keypad scanning process.

In your laboratory notebook, record the following ***prior to 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 keypad

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 keypad 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 in software.
3. Enter, compile and download your keypad scanning program. Execute your program, verifying that correct key number is displayed for each of the 16 keys. ***(Record your observations of the reaction to each key 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/O 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 key bouncing with the oscilloscope to determine if, and for how long, bouncing occurs following a key 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 key number and the other showing the IRQ pin and the keypad row and column immediately following a key press. (This should show your scanning algorithm being executed.)

4. Provide an oscilloscope screen capture, showing the IRQ pin or one of the row lines following a key 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.