Microprocessor-Based Systems (E155)

Harris

Lab 4: Keyboard Scanner

Requirement

Design and construct a circuit to scan your 4 by 4 matrix keypad, and encode the output as a four bit number corresponding to the numerals on the keypad (it is required that the keys be decoded in the specified pattern[*]). With each key press, the four bit data corresponding to the depressed key should be latched into an output register, and the previous entry should be shifted into a second register (i.e., the pair of registers should always hold the data corresponding to the two most recent key presses). You should use your multiplexed display from Lab #3 to display the contents of the two registers, with the most recent entry appearing at the right.

Suggested Approach

- Understand what a matrix keyboard is and how it works.
- Understand how to scan (poll) a keypad.
- Construct a block diagram for your system, noting the required input-output relations for each block.
- Sketch a state transition diagram for your scanner block.
- Design, implement, and debug your blocks.
- Assemble and debug your overall system.

This is a thinking person's lab. If you thoroughly understand the problem and design a simple scanner FSM, you can complete the lab fairly efficiently. If you go by trial and error, you may find yourself in lab indefinitely.

[*] For those of you with unlabeled keypads, the required layout is described in the discussion section.

Discussion

Matrix Keypad

Your keypad is basically a four by four matrix of crossed wires, as shown in Figure 4.1. When a button is depressed, it creates an electrical connection (for the duration of the key press) between the row and column wire that intersect beneath that button. The pinout for the rows and columns is shown in Figure 4.2 for the two models of keypads that may be in the lab.

Keypad Scanning

The most common technique for encoding key presses on keypads and keyboards is called *scanning* or *polling*. In this technique, one column wire at a time is sent to a logic state that is distinct from the remaining column wires (e.g., one column at a time is sent LOW, while the others are held HIGH). With this scheme, the row wires should then be "pulled up" through fairly high value resistors (so they would appear HIGH if undisturbed). When a button is pressed, the corresponding row wire will be pulled LOW when the corresponding column is sent LOW by the polling circuitry. When a LOW is detected on any of the row wires, the polling is frozen, and the combination of which column is being driven LOW and on which row the output LOW is detected uniquely identifies which button has been pressed. This information is encoded as the desired output for the particular button location. When the button is released, polling is resumed.

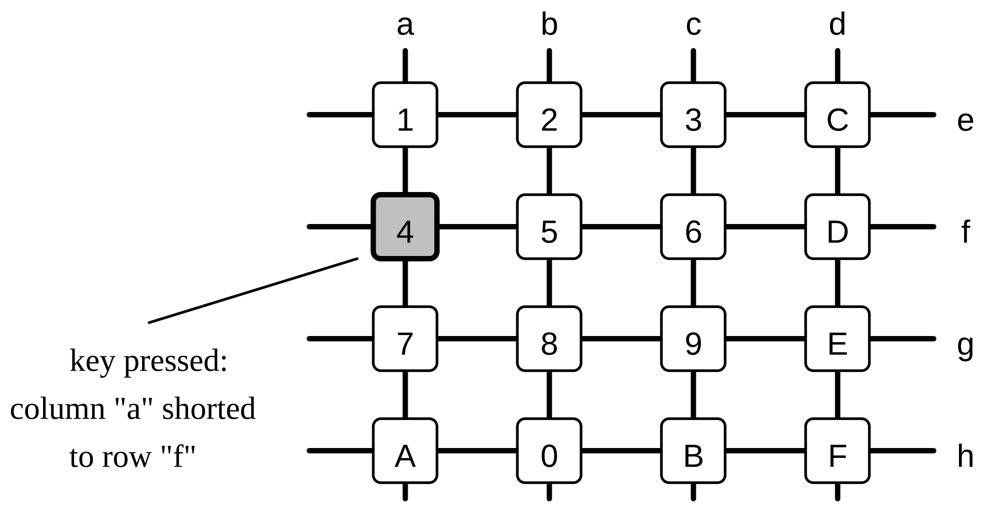

Figure 4.1: Keypad Matrix

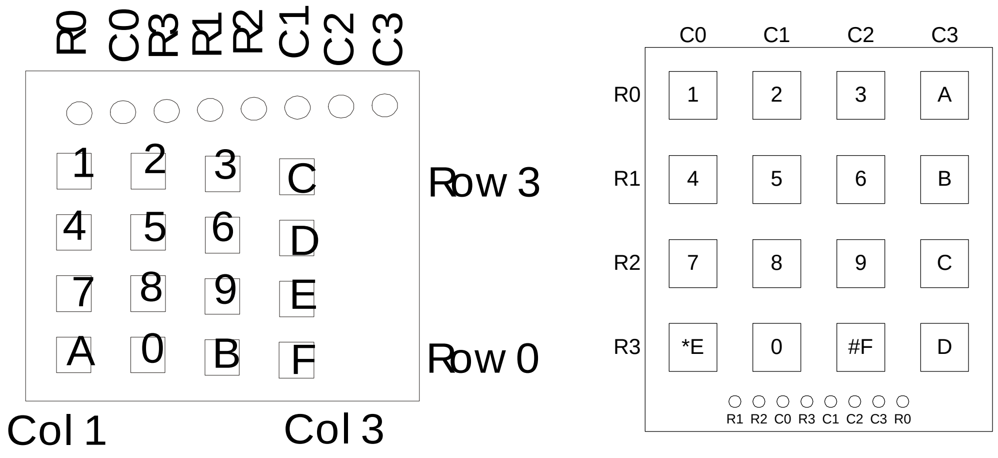

Figure 4.2: Keypad Pinout

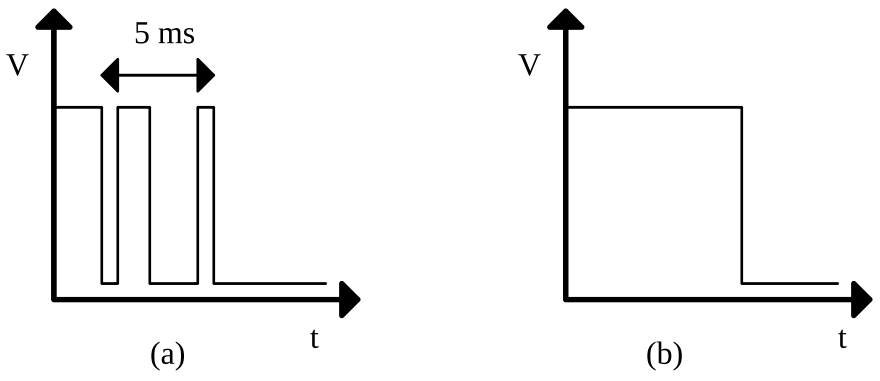

Figure 4.3 (a) Contact bounce of a switch. (b) Debounced signal.

Debouncing

Whenever a mechanical switch is actuated, the signal from the switch may "bounce" because the switch makes and breaks contact a number of times before it settles firmly into its new state. The duration of this bouncing is generally well under 5 milliseconds; look at it on the oscilliscope. If the signal from the switch is to be used to provide a digital clocking or latching signal, switch bounce can cause problems because each bounce will be interpreted as a separate pulse. Figure 4.3 (a) shows a signal with bounce, and Figure 4.3 (b) shows the desired form of the signal after debouncing.

If you sample the state of the keyboard too often, i.e. more than once every 5 ms, you might catch the switch bounce and interpret it as two very brief keypresses. You can entirely avoid the problem of switch bounce by using a sampling period that is longer than the potential bounce time yet shorter than the shortest press your fast-fingered friends can generate.

Xilinx Notes

You will need to generate rather complex stimulus to simulate your state machine. Many students have been tempted to skip simulating and debug on the real hardware. Almost all have regretted it.

What to Turn In

When you are done, have your lab checked off by the instructor. You should thoroughly understand how it works and what would happen if any changes were made. You should also be able to show the behavior of key signals on the logical analyzer. Turn in your lab writeup including the following information:

- Your design approach
- Your scanner FSM state transition diagram
- Your Verilog code and simulation results. Be sure to provide a simulation showing that you can successfully debounce a bouncy input signal.
- Schematics of the breadboarded circuit
- How many hours did you spend on the lab? This will not count toward your grade.