ECS502U Lab: Week 9

The purpose of this lab is combine the results of the previous two lab sheets into a working keypad/7-segment display application, on the 8051 development board. On the QMPlus module page you can find 8051 assembler instruction documentation, including a brief "cheat sheet." We suggest you create a directory for all your code for ECS502U, and then a sub-directory for this week.

The answers to the questions on this lab will be part of your submission next week, so do not lose them!

Review: Keypad and Display Interfacing

To employ keypads for data entry with a limited number of I/O lines a multiplexed arrangement, such as shown in Figure 1, is common. With reference to this schematic, consider the sequence of steps that a microcontroller must take in order to determine if a particular key is pressed. In this case assume the microcontroller is directly connected to the "R" row outputs and "C" column inputs. Also assume that the C inputs are weakly pulled low by resistors that are not shown.

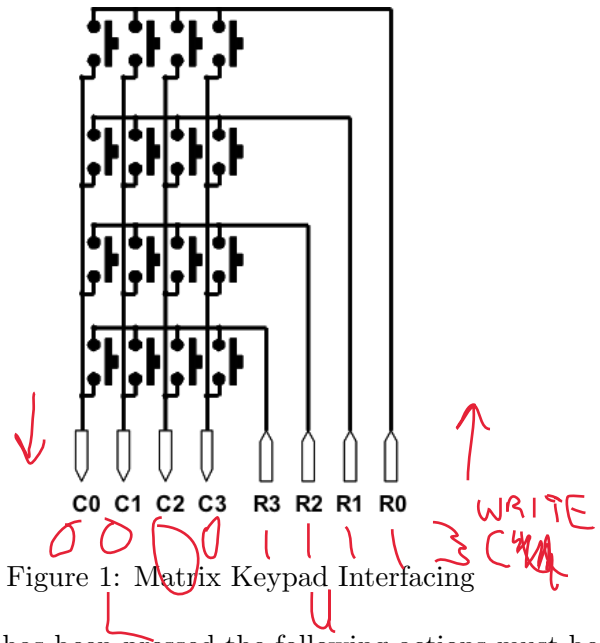

Figure 1: Matrix Keypad Interfacing

To detect whether a key has been pressed the following actions must be taken in sequence:

1. The row outputs are driven high, i.e., "1111"bin, and column values are read
2. If the columns read = "0000"bin then no keys are being pressed as the inputs are pulled low by the resistors. However, if any column reads = "1" then a key has been pressed, overriding the weak pull-down resistor. The column containing the "1" also indicates that the key concerned is one of the four keys along that particular column. The next step is to determine which of these four keys is being pressed as follows:
3. The microcontroller drives *only* one row at a time to the "1" state whilst reading the column values; that is the row outputs will be set to "0001"bin then to "0010"bin, "0100"bin and "1000"bin
4. When row containing the pressed key is set to "1" the column outputs will contain a value other than "0000"bin.

Controlling the Keypad via the 8255

As the 8051 has only a limited number of port pins, when external memory addressing is used, an 8255 device is employed to expand the I/O. The keypad is connected to Port C of the 8255 with the upper nibble of Port C connected to the row lines and the lower nibble of Port C connected to the columns.

The 8255 is automatically reset when the supply power is applied. In addition, its Chip Select (CS) line is tied low so that the device is permanently enabled. To appropriately configure Port C of the 8255, you must first setup the 8255 Control Register with the necessary bit values. The Control Register (CR) is accessible when the device's a1 and a0 are set to "11"bin. CR must be programmed so that the upper nibble of Port C acts as an output, whilst the lower nibble pins of Port C are inputs. Port C is accessible when a1 and a0 are set to "10", respectively.

Controlling the Display via the 8255

The display comprises of four seven-segment displays. Port A is wired in parallel to the a, b, ..., h segment pins of each of these display devices. The lower nibble of Port B is then used to select which of the four displays is active at a given time. The displays have no integral memory and so changing the active display will simply cause the same segment information to be displayed on the currently selected device(s).

To allow the four display elements to display dissimilar information "at the same time" we can make use of the human eye's persistence of vision. The concept operates as follows:

Four registers within the 8051 are used to hold the numerical value/segment code for their corresponding display segment. Each of the four display devices is selected in turn. Whilst a given display is active, its unique segment code is written to the segment lines causing the appropriate value to be displayed. When the next display device is selected, its display code is used instead. Once all four displays have been illuminated in this fashion, the cycle repeats.

If this operation takes place fast enough, it appears that all four displays are continuously illuminated, each with their appropriate numerical value.

QUESTION 1 Combine your work of the labs in weeks 6 and 8, by merging your implementation of the keypad loop of week 6 into the code of Question 2 in week 8. (If you have not completed one of these, you will want to start a fresh implementation based on `q2.asm` of week 8.) Specifically, where you could just directly read/write to the connected virtual hardware in week 6, use the read and write/write_C procedures of week 8 instead. Furthermore, you may have to adjust the bit patterns used for driving and reading the keypad to align with the instructions provided above (you have have driven the columns high on the virtual keypad, whereas now the rows need to be driven high).

Include your final code in the submission and ensure the following requirements are met, plus add describing text as stated below. The describing text preferably refers to source lines or procedures (or otherwise labelled instructions). Unless stated otherwise below, such text may be as simple as "This is achieved by lines 7 through to 42."

1. Periodically write the contents of the registers R3 (left most digit), R2, R1 and R0 (of Bank 0) to the display. The contents of these registers hold the segment code for the corresponding decimal character such that if the value of the right-most digit is 3 decimal, the R0 register holds the segment code to display a "3" character. Add a brief statement to describe how your code achieves this.

2. Scan the keypad for any single key press. If pressed the key that is pressed must be identified and its numerical value used to ensure the left-most digit displays the correct value of this key. Again, add a brief statement to describe how your code achieves this.

3. Upon each key press the value of each displayed digit must be shifted to the left. The most significant value is discarded when it is updated with the value shifted in from the adjacent display digit to its right. As the value of the digits are held in the registers R0, R1, R2 and R3, the contents of these registers must be adjusted according to corresponding sequence of actions:

   New R3 ← Old R2; New R2 ← Old R1; New R1 ← Old R0; New R0 ← Segment code for new key pressed.

4. The keypad should include a debounce mechanism as necessary, and the display must be programmed so as to provide flicker-free illumination of the characters held in the R0, R1, R2 and R3 registers. Again, add a brief statement to describe how your code achieves this. Furthermore explain how you concluded that your results are satisfactory.

5. Describe how your code responds to concurrent key presses. Which of them is being picked up? Why is this the case?

6. Optional: How would the code need to be modified in order to use interrupts instead of polling? Which changes to the development board would be necessary?

QUESTION 2

(a) Explain the difference between *port mapped* Input/Output and *memory mapped* Input/Output. Which of them apply to the 8051, and which assembly instructions are used?

(b) With regard to memory mapped Input/Output describe what is meant by partial and full address decoding. Give a simple example of each.