

## Lab 5: Keypad, LCD and Interrupts

### OBJECTIVES

- To understand how a keypad functions as a raster scan input device and to learn how to interface a keypad to a microprocessor.
- To understand how to control an LCD panel and to interface it to a microprocessor. (Normally, an LCD can be used as a memory-mapped I/O device, but due to current limitations of the 68HC12, a buffer will be used between the 68HC12's data bus and the LCD panel.)
- To understand how interrupts work on a particular microprocessor and how to use interrupts on our processor.
- To study the main timer sub-system and to become familiar with the real-time interrupt system on the 68HC12. (Read *M68HC12B Family Technical Data*, Section 10, for more info.)

### REQUIRED MATERIALS

- 1 - Keypad
- 1 - 2.7k or 1k SIP R-pack
- 1 - Socket for the SIP R-pack
- 1 - LCD (4-bit data)

Printout of the document: *LCD Panel Notes (4-bit data)*

### PART 1 – KEYPAD

To use the keypad, you will need 4 input pins and 4 output pins. You could use a single bi-directional port (e.g., Port P or Port T), but is slightly easier to use 4 pins of two dedicated ports (one input and one output). The choice is yours. (If you choose to use a single port, I suggest that you use Port P, since you will need parts of Port T in labs 6 and 7.)

The keypad included in your kit is similar to a phone keypad. See Figure 1 for a description of the keypad. Each key has two contacts, one attached to a “row” wire and the other attached to a “column” wire. When a key is pressed the column wire and row wires are connected (with a small resistance). There is no power supplied to the keypad except through inputs to the row or column wires. Figure 2 shows how the keypad is interfaced with a microprocessor’s port(s). A pull-up or pull-down resistor is used on either each row wire or each column wire. Please note that if you use a keypad different from the one supplied in this semester’s kit, the pinouts may change. (The pinouts can be easily determined with a multimeter set on resistance.)

When the keypad is connected to the microprocessor and a key is pressed, a connection is established between an input port pin and an output port pin. A value written to the output port can be read from the input port. But what will be read on the unconnected (key not pressed) pins of the input port? Pull-up or pull-down resistors are used so that the input port pins don’t have floating values. The resistance is necessary to limit the current drawn by the

port. With pull-up resistors, the input ports will be at “1” (+5V) when no key is pressed; when a key is pressed, a “0” (GND) will appear on the input in the when an appropriate microprocessor output is “0” and the key at the row/column in question is pressed.

Note that in the below description, the words row and column can be reversed (with a corresponding modification to the circuit).

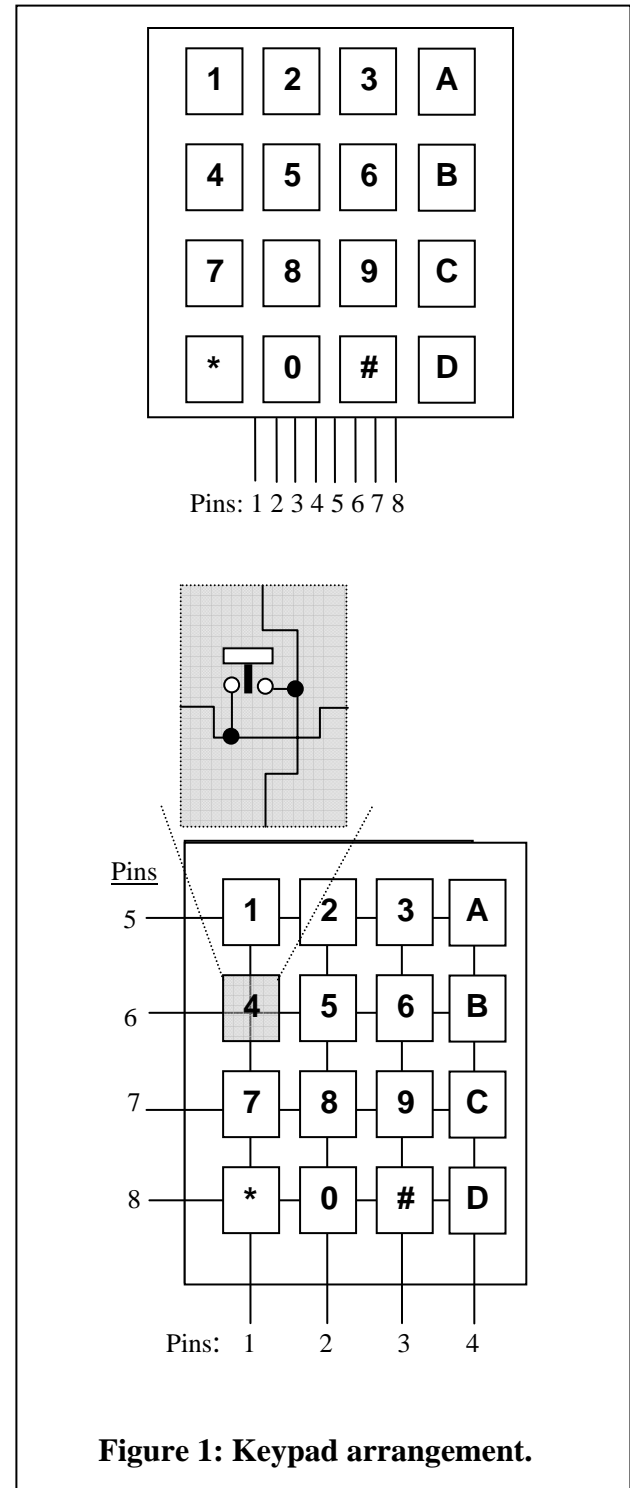


Figure 1: Keypad arrangement.

## Lab 5: Keypad, LCD and Interrupts

The keypad is read by scanning it one column at a time (Figure 2). First, write a %0111 to the appropriate output port pins (columns), and then read the input port (rows). This puts a zero on only the first column of the keypad. If any of the keys in column 1 are pressed, the corresponding input bits will be zero. If a key is not pressed, that row bit will be pulled high by the pull-up resistor. The input port will only see the keys that have a zero voltage on their column pin. Next, write a %1011 to the output port pins (columns) to scan the second column. Continue by writing a %1101 then a %1110 to the output port pins to scan the third and fourth columns, respectively.

- **Warning:** Don't push two keys at once. Pressing two keys in the same row will connect output pins!

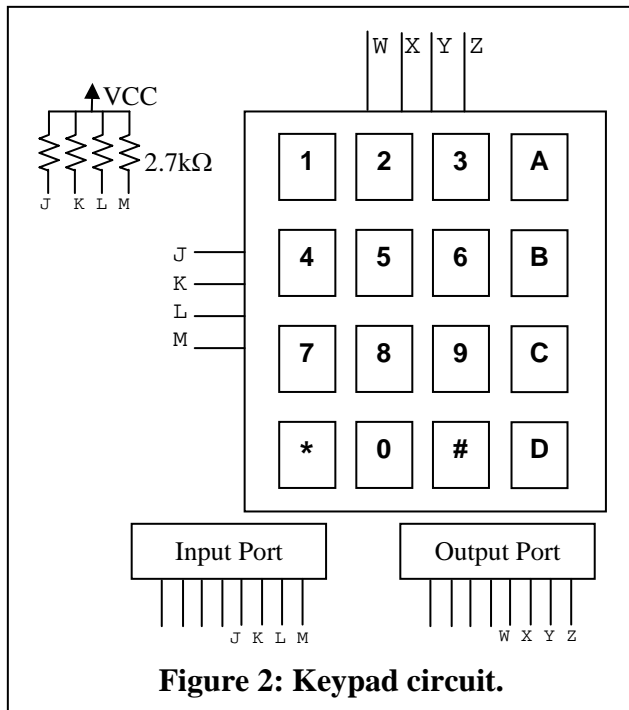


Figure 2: Keypad circuit.

1. To implement the keypad interface, you need 4 input pins and 4 output pins. You can either use an 8-bit bi-directional port for the inputs and outputs or use only four bits of two separate ports. Do not use Port T, since this port will be needed for the next two labs along with the keypad. Wire up your keypad as described in Figure 2. Add the keypad connections to the schematics done in previous labs.
2. Make sure that all of your external hardware still works. This includes the SRAM, input port, and output port. (The external UART is not needed for this lab.)
3. Write a subroutine to scan the keypad and determine the key pressed. The subroutine should return a single value. Some possible values are the ASCII values of the key pressed or the actual number itself,

with some made up numbers for \* and #. Test the keypad scanning with the OutChar subroutine you wrote in a previous lab. For example, if you hit a key on the keypad, the test program will display that character on the screen.

### PART 2 – LCD DISPLAY

In this section you will add a LCD display to your board. This particular LCD display takes input through a 4-bit data bus. Refer to the *LCD Panel Notes (4-bit data)* document for pin-out and command information.

We have found that when the LCD is connected as a memory-mapped device to the 68HC12's data bus, the 68HC12 has a problem providing the necessary current to the LCD. In order to allow the 68HC12 to use the LCD as a memory-mapped device, you must use a buffer chip between the 68HC12's data bus and the LCD data pins. A tri-state buffer chip will do the job nicely. (See the Figure 3 below.) The major drawback to this solution is that this makes it impossible to read from the LCD device without additional circuitry. (The good news is that few designs actually read from LCD devices.)

The LCD module included in your kit defaults to 1x16 mode (1 line containing 16 characters) and 8-bit data. The LCD has 2 registers: command and data. The *LCD Panel Notes (4-bit data)* document includes a flowchart showing how to use the LCD display module in the required mode.

1. Use Quartus to generate the appropriate chip-enable schematic or VHDL for the LCD (seen as an output port to the 68HC12). The command register will be at any available address \$X000 and the data register will be at \$X001, where X must be chosen so that it does not conflict with any of your other expanded devices (or internal resources on the 68HC12).. Add this schematic or VHDL to your existing CPLD contents.
2. Verify that your board still boots and that all other external devices still function, i.e., the SRAM, input port, output port, keypad, and UART.
3. Wire the LCD as shown in the *LCD Panel Notes (4-bit data)* handout with the modification described in the next step.
4. Due to current limitations on the 68HC12, we will need to add a tri-state buffer between the data bus and the LCD panel (as shown in the figure below).

**IMPORTANT Note:** If you don't have a potentiometer, you can try various resistors between GND and the  $V_O$  input.  $V_O$  is the LCD contrast adjustment. A higher resistance between  $V_O$  and GND dims the LCD display. I have used ~1kΩ with some LCD displays.

5. Once the LCD is wired, verify that your board still boots. Test all external components: SRAM, input

## Lab 5: Keypad, LCD and Interrupts

port, output port, keypad, and UART. Debug your hardware if there are problems.

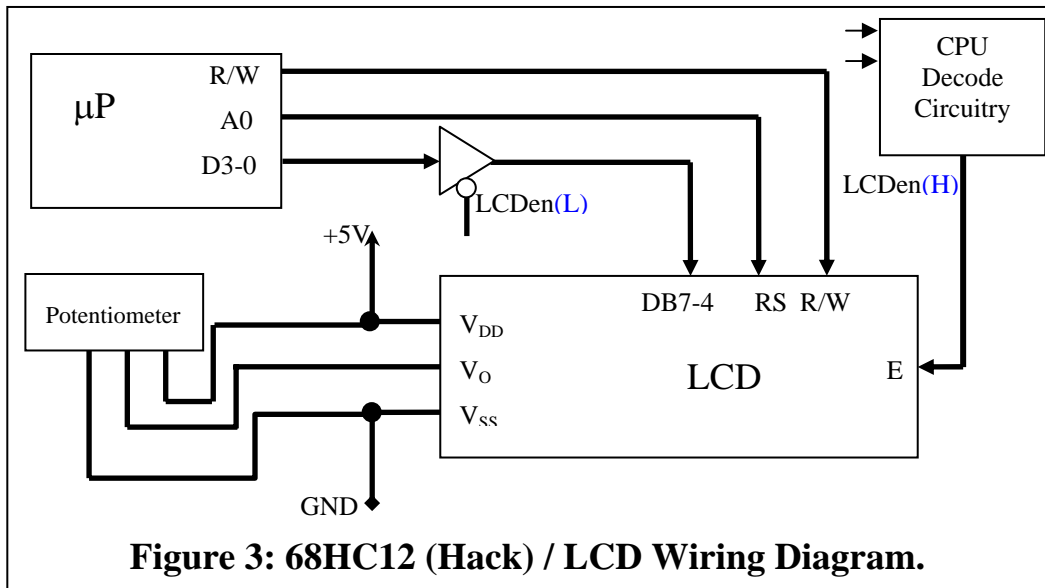
- Test your LCD by typing the following commands at the D-Bug4744 prompt.

• mm X000 03	• mm X000 00
• mm X000 03	• mm X000 01
• mm X000 03	• mm X001 03
• mm X000 02	• mm X001 00
• mm X000 02	• mm X001 03
• mm X000 08	• mm X001 01
• mm X000 00	• mm X001 03
• mm X000 0F	• mm X001 02

Your LCD should now show the numbers 0, 1, and 2. If your LCD does not show anything, double-check your connections using a multi-meter. Also, look over your CPLD chip-enable equation for the LCD.

- Write a program that displays your first name on one line and your last name on the next line.

Your LCD should already be in 2 row mode. To access the second line you must move the cursor. This is done by storing \$C0 in the command register.



**Figure 3: 68HC12 (Hack) / LCD Wiring Diagram.**

### PART 3 – Using Interrupts

The real-time interrupt (RTI) system on the 68HC12 can be used for repetitive processes at a fixed time rate. The RTI system is programmable with up to 7 different times. In this section of the lab you will use the RTI system to make a 2 second delay **without** using a timing loop (polling), i.e., using an interrupt (namely the RTI interrupt).

Use the RTI system at a rate of your choosing. In the RTI interrupt service routine you must have a variable that is decremented so that after approximately 2 seconds, it will execute a program. The variable must be initialized with a calculated value (2 seconds divided by the RTI period). This variable is decremented every time and RTI occurs. When it gets to zero, write a “0” to the LCD display and then reset the variable to the calculated value. The next time 2 seconds expires; write a “1” to the LCD display. This process should continue forever, displaying 0, 1, ..., 8, 9, 0, 1, ...

Now modify the above program to do the following. Every 2 seconds output (to the LCD display) the last keypad key that was pressed.

### QUESTIONS

- What changes would be needed if the keypad was 8x8 instead of 4x4.
- Consult the *LCD Panel Notes (4-bit data)* handout. Explain what the commands \$28 and \$C0 do. You must indicate what each bit does to get credit.
- What is the minimum and maximum periods you can get with the technique described in Part 3 using RTI if (a) the variable is a single byte; (b) the variable is a double-byte.
- If a bi-directional tri-state buffer chip (74HC245) was available (to replace the tri-state buffer), could the LCD device be both read and written to? If so, what equations would go to the direction and enable inputs of this device. Give a circuit diagram showing 68HC12 data bus and necessary signals, the bi-directional buffer and the LCD device.

## Lab 5: Keypad, LCD and Interrupts

---

5. If an 8-bit LCD panel was available, writing to it is a simple as writing to the SCI data register, i.e., a write could occur after checking a flag bit (the BUSY bit on the LCD device). Why do you think 8-bit LCD devices are now so hard to find or cost so much more than the 4-bit LCD devices?

### PRELAB REQUIREMENTS

1. Read the **ENTIRE** lab handout.
2. The printout of the Quartus BDF or VHDL for the LCD chip-enable equation in Part 2 step 1.
3. The printout of the *LCD Panel Notes (4-bit data)* handout.
4. Draw (by hand or by program) the keypad circuit expansion and the LCD circuit expansion.
5. The ASM and error-free LST files for the keypad scanning program of Part 1.
6. The ASM and error-free LST files for LCD program that displays your name (in Part 2).
7. The ASM and error-free LST files for the two programs in Part 3, the first using the RTI and LCD panel and the second using RTI, the LCD panel and the keypad.
8. Answers to all of the pre-lab questions.

### IN-LAB REQUIREMENTS

1. Show your TA that you can read the keypad correctly by demonstrating your program from part 1.
2. Show your TA that you can use your LCD correctly by demonstrating your program from part 2.
3. Show your TA that you can use your LCD correctly by demonstrating your two programs from part 3.

## JUST FOR FUN

### IRQ FOR KEYBOARD (not required)

This section is **just for fun, i.e., it is not required.**

Since in this section we will be using the IRQ pin, remove the UART equation using IRQ from your CPLD. Use the IRQ pin to help you detect when a key is pressed using the following algorithm.

Approximately every 4ms (conveniently, this is an RTI rate), change the row/column on the keypad to which you are outputting a zero (while outputting a one to the other rows/columns). An IRQ interrupt should be triggered when any of the keypad inputs (column/row) pins go low. Write the appropriate equation to do this and place it in your CPLD. You will need the 4 CPLD inputs (for the 4 columns/rows) and a single output (connected to the IRQ pin).

In the IRQ interrupt service routine, determine which key is pressed. Wire the IRQ pin to an additional input pin on your board. I'll call this pin IRQ\_Copy. Also in the IRQ interrupt service routine, setup RTI to delay an appropriate time for bouncing to finish once a key press is detected. When this time has passed, check the value of the IRQ\_Copy pin. Wait until it goes high (indicating the key has been released) and then wait again (using RTI) until the bouncing for the key release is completed.

If you implement the above, you have created an interrupt driven keypad. If you do this extra part of the lab, feel free to use it in subsequent labs.