Lab 4: Keypad and I/O Port Expansions

OBJECTIVES

- To explore and understand the implementation of memory-mapped I/O. Add an 8-bit input port and an 8-bit output port.
- To understand how a keypad functions as a raster scan input device and to learn how to interface a keypad to a microprocessor.

REQUIRED MATERIALS

- *Wire wrap tool, multi-meter, DSC board*
- *1 - 74HC573, 1- 74HC574, 1 keypad, 2 - 0.1μF bypass capacitors, wire wrap*
- Read/save the following document:
  - *Timing diagram on SPRU949D (pg 33-36)*
  - *GPIO Multiplexing and PCLKCR3 description, in SPRUFB0D*
  - *Spec sheets for 74HC573 and 74HC574*

Helpful Materials:

- 2 - 20-pin wire wrap sockets for 74HC573 and 74HC574
  - Digi-Key part # ED4320-ND ($4.21 each)
- 1 – 10-pin socket for keypad (NOT wire wrap) IN YOUR KIT!!!
  - Digi-Key part # A24816-ND ($1.40 each) **[See Appendix for more info]**

PRELAB REQUIREMENTS

It is required that you make a flowchart or write pseudo-code **before** writing any program in this course. This will help you formulate a plan of attack for the code. Submit these flowcharts/pseudo-code as part of your prelab requirements.

Comments in your programs are also required. The Pre-lab Report Guidelines state that "All code should be commented to a level that someone how can readily understand the purpose of each section if not each line of code."

Part A: Input/Output Ports, General Description

The DSC and other many other microprocessors have several input/output (I/O) pins, often with multiple pins grouped together and called input/output ports. Each of the I/O pins typically has additional non-I/O features that can be utilized as an alternative to general I/O. Therefore, it would be useful if we can "free up" these pins for their special use by adding other memory-mapped I/O ports.

1. Generate the necessary chip-enable equations to implement one 8-bit input port and one 8-bit output port. Both ports should both be mapped to address 0x4000 and mirrored at addresses 0x4001 through 0x47FF. (For this lab **ONLY**, ignore address pins A21 through A16. For subsequent labs, we will add other external components that will overlap the addresses specified for the I/O ports **unless** we include the address pins A21 through A16. Failure to utilize these pins in this case will cause damage to your devices.) Use partial address decoding (also known as reduced-address decoding). Tri-state buffers **MUST** be used for input ports and flip-flops should be used for output ports (to save output values). Two chips included in your kit, the 74HC573 and 74HC574, contain the tri-state buffers and flip-flops.

2. Don't forget to enable the XINTFENCLK bit in PCLKCR3.

3. Use Quartus to design a circuit to add both the input and output port chip-enable equations to your CPLD. Use either schematic entry or VHDL to design your CPLD circuits. You will have to wire wrap from the ADDRESS BUS of your board to the CPLD HEADER. In addition, use jumpers in the CPLD JUMPERS header to connect specific signals to the CPLD. Submit this circuit design as part of your prelab requirements.

   NOTE: Some of the CPLD pins on the CPLD JUMPERS header are duplicated on the CPLD HEADER header. Be cautious when wiring. It is strongly recommended that you set all unused pins as tri-stated in Quartus: select *Assignments* | *Device* | *Device and Pin Options* | *Unused Pins* and then set as *input tri-stated*.

4. Simulate your CPLD circuit design to verify its correct operation. Submit this simulation as part of your prelab requirements.

Part B: Input Port:

1. Use the *74HC573 Octal 3-State Noninverting Transparent Latch* datasheet to create a wiring schematic (with Quartus or drawing software) for the input port using the 74HC573 acting as an 8-bit 3-state device. (You will submit your schematic by email and on paper.) Use pin labels instead of wires to show connections. The schematic should show connections to the DATA BUS header and to the CPLD HEADER. Include all pin numbers. [Note: The 74573 is not available in Quartus, but the operation of the 74373 is identical and can be used. The 74373 can be found in the parts library under *others* | *maxplus2*. The pinouts (not shown in Quartus) of the '373 and the '573 are different.]

2. Wire the 74HC573 chip for your input port using your wiring schematic on your UF F28335 Development Board. I recommend using the first DIP Breakout Islands on the right side of your board below the CPLD HEADER header. Begin by placing the chip on your board and soldering two corners. If the chip is firmly in place, complete the soldering of this chip. A bypass capacitor should **always** be soldered between Vcc and ground on all ICs. Next solder on two headers so that you can wire-wrap to

the chip. Finally, wire-wrap all the signals as shown on your circuit diagram. (If you can procure a 20-pin wirewrap socket, you can simplify this process Digi-Key part number: ED4320-ND. You still will need to solder the bypass capacitor.)

3. Remove the wire wrap from the switch bank from a previous lab and wire-wrap it to the new input port. Now you can use the switches to test the functionality of the input port.

4. Write a quick program to read from the input port store the values into AH. Have your switches set to 0xAA for one read and then 0x55 for the second. You will take two screenshots of your debug window showing both your program and that the correct values were loaded into AH.

   NOTE: Make sure you set the correct values in the GPIO multiplexer registers on the DSC so you can output the necessary signals from the DSC (address lines and data lines), and be sure enable the XINTF clock. The XRDn signal is always available as an output. . Don't forget to turn off the Watchdog timer and set EALLOW in each of your programs.

Part C: Output Port

1. Once the input port functions properly, use the *74HC574: Octal 3-State Noninverting D Flip-Flop* datasheet to create a Quartus design for the output port using the 74HC574 acting as an 8-bit register of D flip-flop. Add your schematic circuit diagram to the one with the input port. (As the semester continues, we will add to this schematic many times.) The schematic should show connections to the DATA BUS header and to the CPLD HEADER. Include all pin numbers. You can add the pin numbers by hand on your printouts or with Quartus. Note: The 74574 is not available in Quartus, but the operation of the 74374 is identical and can be used. The 74374 can be found in the parts library under *others* | *maxplus2*. The pinouts (not shown in Quartus) of the '374 and the '574 are different.
2. Wire the output port using your wiring schematic on your UF F28335 Development Board. I recommend using the DIP Breakout Islands on the right side of your board below the one used for the input port. Begin by placing the 74HC574 chip on your board and soldering two corners. If the chip is firmly in place, complete the soldering of this chip. A bypass capacitor should **always** be soldered between Vcc and ground on all ICs. Next solder on two headers so that you can wire-wrap to the chip. Finally, wire-wrap all the signals as shown on your circuit diagram. (If you can procure a 20-pin wirewrap socket [Digi-Key part number: ED4320-ND], you can simplify this process. You still will need to solder the bypass capacitor.)

   NOTE: Power and Ground signals are usually distributed using a bus for each to every chip. Wire-wrap is usually too thin and therefore provides too much resistance. In our case, two wire-wrap wires are probably sufficient. In future courses, like senior design, you should solder power bus lines with thicker wire (or use wide traces for a PCB).
3. Now test your output port by adapting the code you generated previously for the input port. Be sure to output any other additional signals that you might need for your output port to function (XWEn for instance), and add additional code to write the values 0xAA and 0x55 to the new output port. You can determine the values on your output port pins by wirewrapping to your LED circuit (after first disconnecting the GPIO outputs to the LED port) or, if you are less enthusiastic about wire wrapping, you could just use your multimeter to measure the output voltages.
4. Write a simple test program to continuously read your input port and write the data to the output port. Verify that this program works on your board. Email this file with to the class gmail account, as usual.

Part D. Keypad

In this section you will add a keypad to your board. You will use your newly functioning input and output port to communicate with the keypad. You will need 4 input pins and 4 output pins.



Figure 1: Classic telephone.

The keypad included in your kit is similar to a classic touchtone telephone keypad shown in Figure 1. See Figure 2 for a ***possible*** description of the keypad wiring. (Note that there is no guarantee that the keypad you receive has this pinout. You should verify by experiment the actual pinout of the keypad. The pinouts can be easily determined with a multimeter set on resistance. You should verify the pinout **before** any soldering is started.) 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 3 shows one way that the keypad is interfaced with a microprocessor's I/O's. A pull-up or pull-down resistor is used on either each row wire or each column wire.

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 so that when a key is pressed, Vcc

will never be connected to ground. With pull-up resistors, the input port pins will all be at "1" (Vcc = +3.3V) when no key is pressed; similarly, when a key is pressed in a column that has an output of "1" (Vcc), a "Vcc" (1) will appear on the input of that row (column); but, when a key is pressed in a column (row) that has an output of "0" (GND), a "0" (GND) will appear on the input of that row (column). (I suggest that you use the largest SIP resistor pack you have available.) Note that in the below description, the words row and column can be reversed (with a corresponding modification to the circuit).

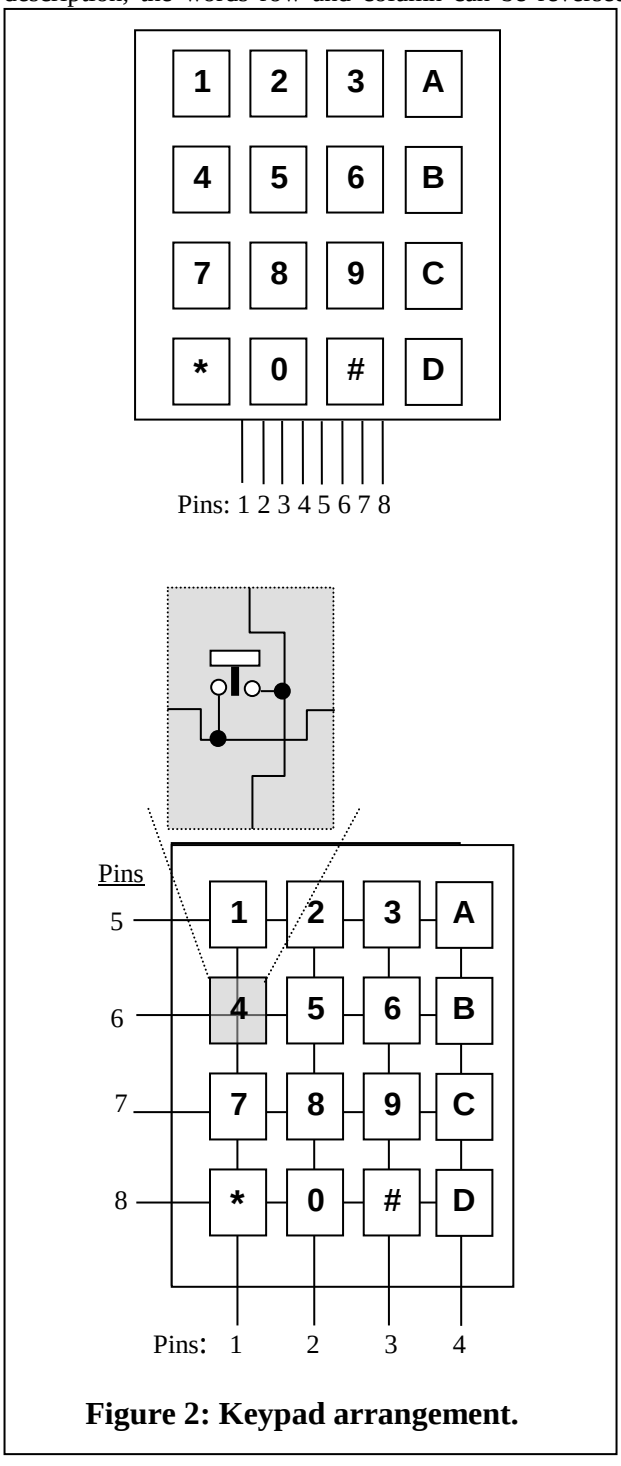


Figure 2: Keypad arrangement.

The keypad is read by scanning it one row (column) at a time (Figure 3). First, write a 0111_2 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 row 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. (This process will repeat continuously.)

Warning: Do **NOT** push two keys at once. Pressing two keys in the same row will connect output pins!

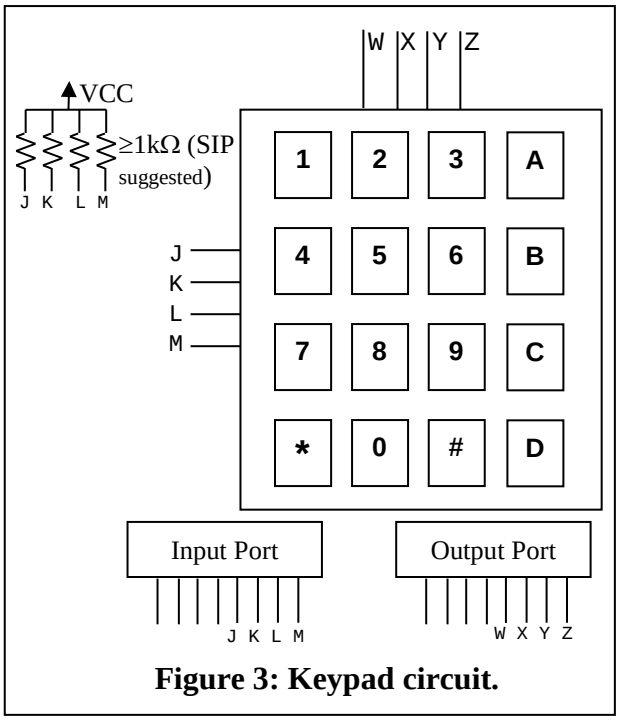


Figure 3: Keypad circuit.

1. **Before wiring your keypad** onto your PCB, solder a 14-pin header (cut one to size) in the location immediately under the UA1 location (which will hold your SRAM chip for lab 5). This would go in the top right of Figure 4, but is not shown in the figure. Insert this header from the bottom of the board and solder on top.

2. To implement the keypad interface, you need 4 input pins and 4 output pins. You will use 4 pins from the input port and 4 pins from the output port. Undo the wiring of 4 pins of the switch bank. The other four pins of the input port can be left in place. (You can leave the LED wiring as is.) Now wire up your keypad as shown in Figure 3. Add the keypad

connections to the schematic that you previously during this lab.

Since there are no pins in your keypad, you will need to install a 10-pin single row header (cut from one of your header strips) into the bottom of the keypad's pin holes. (The header's plastic should be on the bottom of the keypad PCB.) Solder the pins on to the keypad PCB.
Cut a **second** 10-pin header. (You will insert this 10-pin header into your PCB **on second row from the bottom**.) Your keypad should be placed immediately to the left of the 74HC573 and 74HC574 ICs previously installed on the right DIP islands, but mounted in the bottom row of pins, as shown in Figure 4. Once you find the correct keypad location (hold it in place but do **NOT** solder it), insert the other 10-pin header from the bottom of your PCB in the row above the keypad header so that the plastic is on the bottom of the PCB and the short pins go through the PCB. Now remove the keypad and solder the 10-pin header. Next re-insert the keypad immediately below the just solder 10-pin header (into the bottom row of pins) and solder the keypad into place. You can also buy a socket or female header row to make it easier to remove your keypad in the future (Digi-Key part number: A209-ND). This would be soldered in place of the header connected to the keypad. The keypad would then plug into it.

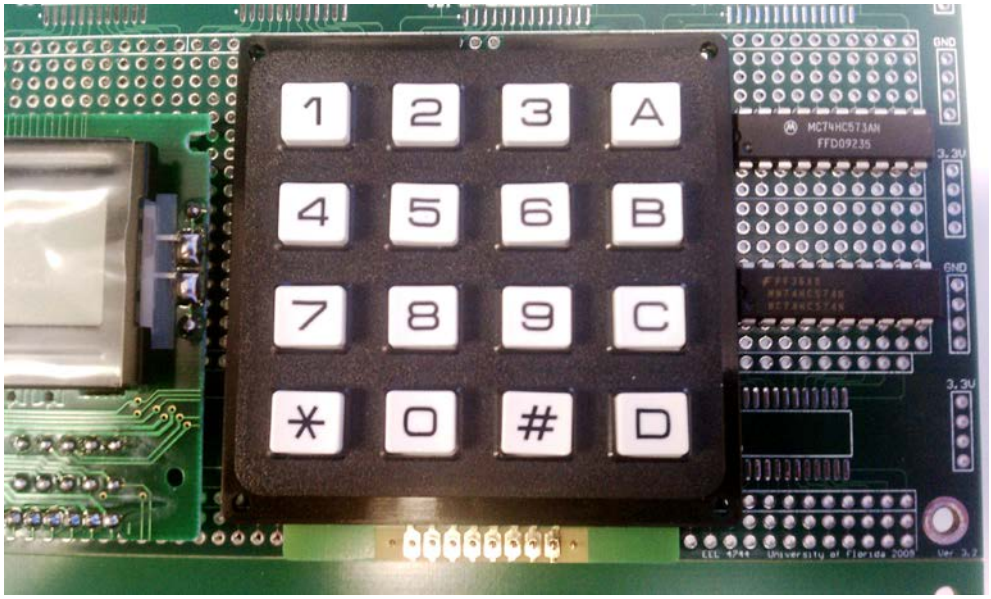


Figure 4: Keypad location on UF DSC Board.

3. Write a subroutine to scan the keypad and determine the key pressed. The subroutine should return a single value. Some possible return values are the ASCII values of the key pressed or the actual number itself with something else for '*' and '#'. Show that the keypad subroutine works with the LEDs you added in Lab 3. If you push the 9 button, the LEDS should display 1001. If you push A, the LEDS should display 1010 and so on. Use 0xE and 0xF for the '*' and '#' respectively. The values should display as long as the key is pressed.

   NOTE: The DSC has an instruction pipeline that can, and most likely will, cause problems for your keypad program. To test your keypad code, use the debugger and single step through the scanning while keeping a key pressed down. Once you verify that the code works, try running the code at full speed. If the code does not act the same, try adding multiple (3 to 5) NOP instructions in between your column scans. This is a dummy instruction that performs no operation. It is used in our case to "flush" the pipeline.

4. All mechanical switches bounce, which for the keypad would result in seeing multiple values of the same key, for example 444477744444444 might result, instead of 4744. To demonstrate that this is in fact occurring, modify your code in the previous step to count the number of times a single key is (apparently) pressed in a row. Display this number in binary on the available 4 LEDs.

5. To prevent the result of bouncing to appear as multiple key presses, after a key press is detected, you should delay at least 200ms before continuing the keypad scanning. (The keypad might only need 10ms; you can experiment with the delay. If you do, please email Dr. Schwartz with the smallest delay that always worked for this keypad.) Once this time has elapsed, wait for the key to be released. Once released, another similar time delay should be started.

Modify the code of the previous step to demonstrate that each key only registers once when pressed.

6. Now that you know how to debounce your keypad, rewrite the subroutine above to again output the key pressed to your available 4 LEDs.

PRELAB PROCEDURE

1. Answer all prelab questions.
2. Bring a printed summary text document and the wiring schematic to turn in to your TA: The summary text document and your wiring schematic must be submitted on paper and by email with **EVERY** pre-lab.
3. Email the following to the class gmail account: your Quartus archive file, screen shots, flowcharts/pseudo-code, asm and list files, your complete schematic of I/O ports and keypad, answers to the prelab questions, and the summary text document.

Note: Prelab requirements **MUST** be accomplished **PRIOR** to coming to your lab.

PRELAB QUESTIONS

1. When you add your input and output ports at 0x4000 to 0x47FF, what other images are being created? Hint: We have 22 bits of address. What are the total number of images of your input/output ports? **NOTE: In subsequent labs, you will need to remove these high address images.**
2. What is the benefit of doing partial address decoding compared to full address decoding?
3. What is a benefit of adding I/O ports to a design?
4. What will happen if you press two row keys at the same time on the keypad?
5. Why should you not press two column keys at the same time on the keypad?

LAB PROCEDURE

Demonstrate that your final keypad program works to the TA. If your final program does not work, demonstrate previous parts of the lab that does work. Be prepared to answer questions about your program.

Get your SRAM (for lab 5) from your TA. Solder the RAM into the location labeled UA1. Verify that you soldered it correctly by viewing it under the microscope.

APPENDIX

If you purchase the female header suggested in the "Helpful Materials" part of this lab, then read the following.

1. Place the pins of the female header through the bottom of your keypad holes and solder them on the top (side with the keys).
2. Solder male header pins as shown in the below figure, with the long pins inserted through the top of your board and soldered from the bottom.

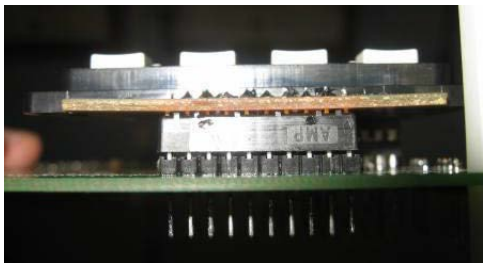


Figure A.1: Side view showing keypad, female header, and male header.

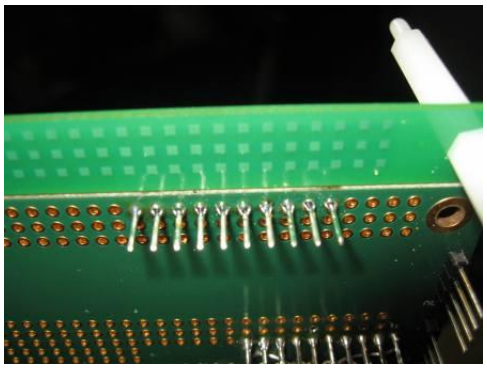


Figure A.2: Bottom view showing soldered male header with long pins soldered on the bottom of the board. Wire wrap **directly** to these pins.



Figure A.3: Top view showing keypad in place.