1. Introduction
There was a lot of confusion on scanning a keypad. Instead of coming up with their own scan routine, the handout scheme was used without examining the underlying hardware. This paper is a supplement to the original Lab 6 handout.

2. Basic Hardware
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 wire are connected. There is no power supplied to the keypad except through an output port or with a power supply (through a pull-up or pull-down resistor). I will briefly discuss:
1) Scanning schemes,
2) Generating an interrupt,
3) Pressing multiple keys.

3. Basic Approach for Scanning
When a key is pressed, a connection is established between port C and port D. A value written to one port can be read from the other port. But what will be read on the unconnected (key not pressed) pins of the input ports? Pull-up or pull-down resistors are used so that the input port pins don’t have floating values. A large enough resistance should be used so that the current drawn through the resistor can be handled by the port. Without additional information on the ports of a specific board, the following values (Figure 2) should be assumed.

<table>
<thead>
<tr>
<th>Key</th>
<th>No Resistor</th>
<th>Pull-up on Input</th>
<th>Pull-down on Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Pressed</td>
<td>FLOAT</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Pressed, outport bit=0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pressed, outport bit=1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 2: Values at the Input Port Pins
- With pull-down resistors the input ports will be at “0” when no key is pressed and a “1” written on the output port can be detected.
- With pull-up resistors the input ports will be at “1” when no key is pressed and a “0” written on the output port can be detected.

4. Two Sets of Resistors
At the design stage, it does not matter whether the resistors are pull-up or pull-down, or which port is input and which is output. With one set of resistors you can use one port as an input port and know that you will either read “0” (GND) or “1” (Vcc).

Resistors on both the columns and rows allow both ports to be used as input ports. The resistors do not interfere in any way with the ports when they operate as output ports. Although this example uses 2 sets of pull-up resistors, 2 sets of pull-downs or a pull-up/pull-down set can be used just as easily. Here is a possible sequence:

1) Find Columns
   - Set port C (rows) as input ($00 to DDRC).
   - Set port D (columns) as output ($FF to DDRD) and write zeros ($00 to PORTD).
   - Read Port C and the pin with a “0” is the column of the pressed key.

2) Find Rows
   - Set port D (columns) as input ($00 to DDRD).
   - Set port C (rows) as output ($FF to DDRD) and write zeros ($00 to PORTC).
   - Read Port D and the bit with a “0” is the row of the pressed key.

This is really simple to code. This is also the type of scheme which could give you wrong results if multiple keys are pressed.
4.1 Keypad Scanning
If hardware minimization is important, only one set of resistors is needed. The remainder of this paper will assume only one set of pull-up resistors.

With only one bank of resistors, only one port can be used as an input port. The code is only slightly more complicated. Place pull-up resistors on port D (Figure 3) to use port D as an input port. Port C is an output port. Since the directions on the ports won’t change, write $00 to DDRD and $FF to DDRC in the initialization. The row and column of the pressed key are found at the same time,

1) Send %XXXX1110 to port C (Check row 1).
2) Read Port D
   • The bit with a “0” is the column of the pressed key.
   • If no bit was “0”, go to next row
3) Check subsequent rows by writing to port C,
   • %XXXX1101 to check row 2.
   • %XXXX1011 to check row 3.
   • %XXXX0111 to check row 4.
4) Loop back to step 2.

This technique requires more loops, but is easier to update the index into the lookup table. However, if two keys on the same column are pressed, an output port pin at GND can be shorted to an output port pin at Vcc. The question of damage aside, I would not know the actual voltage at the inport when simultaneously attached to two output pins at different voltages.

4.2 Generating an Interrupt

Again, pull-up resistors are on the input port D, a “0” written on port C can be detected if the appropriate key is pressed. If all the pins on port C are “0”, then one of the pins of input port D will also be “0” when any key is pressed. Connect port D pins 2-4 (column wires) to the inputs of a 3-input NAND gate. When any key is pressed, the output of the NAND gate goes high. Therefore, choose a rising edge to trigger the STAF. A read of port D also tells you the column of the pressed key, so the scan routine is simplified.

4.3 Short between Pins of the Output Port
We don’t have to worry about shorts while waiting for the interrupt since all the output pins are at the same voltage anyway. When scanning, two output pins cannot be shorted if there is only one output pin.

1) Send %XXXX0001 to DDRC.
2) Write %XXXX0000 to PORTC (Check row 1).
   • Read Port D
   • The bit with a “0” is the column of the pressed key.
   • If no bit was “0”, go to next row
3) Check subsequent rows by writing to DDRC,
   • %XXXX0010 to check row 2.
   • %XXXX0100 to check row 3.
   • %XXXX1000 to check row 4.
4) Loop back to step 2.

5. Some Trade-Offs

5.1 Hardware versus Code
The bottom line is that some limitations in hardware can be overcome with good code, and that extra hardware can make coding a lot easier.

5.2 Safety
The scan cycle is short enough that damage should not be a problem under normal operation. However, debugging is not normal operation, and why take chances on damage when there is an easy alternative?

5.3 Polling versus Interrupt
Interrupt service routines usually are slightly more complicated than polling routines and more appropriate for devices that don’t require frequent servicing. That is, ISR’s are better if the time freed by not polling is more that that incurred by extra code complexity (and you have something else to do with the freed time).