Keyboard Matrix Help

Dave Dribin

v1.1 24 Jun, 2000

Revision History

Revision 1.1	24 Jun, 2000	Revised by: dd
Converted to DocBook SGML and updated diode information.		
Revision 1.0	4 Feb, 1999	Revised by: dd
Initial release.		

Attempts to explain how a keyboard matrix works, what "ghosting" and "masking" are, and how to prevent them.

---

1. Introduction

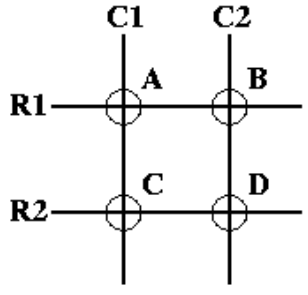
It took me a bit to figure this out, partly due to the fact of no really good explanation of it. So, I'm going to have a crack at it. Basically, I wanted to understand how keyboard matrices work. Specifically, I wanted to know why keyboard "ghosting" and "masking" happen, and how to prevent them.

---

2. The Matrix Circuit

Keyboards use a matrix with the rows and columns made up of wires. Each key acts like a switch. When a key is pressed, a column wire makes contact with a row wire and completes a circuit. The keyboard controller detects this closed circuit and registers it as a key press. Here is a simple keyboard matrix:

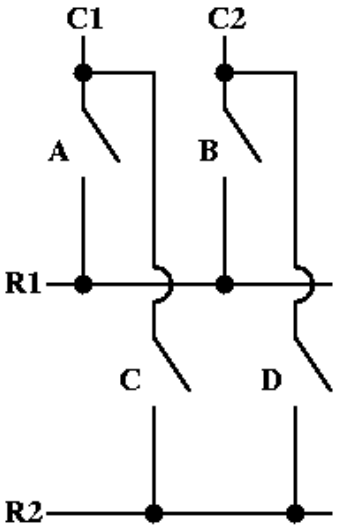
Figure 1. Conceptual Matrix Circuit



This keyboard only has 4 keys: *A*, *B*, *C*, and *D*. Each key has a unique grid location, much like points on a graph. Key *A* is at node C1R1, key *B* is at node C2R1, key *C* is at node C1R2, and key *D* is at node C2R2. In reality this is pretty useless which is why real keyboards use many more rows and columns. However, I want to keep it simple.

The electronic circuit for this matrix looks something (not exactly) like this:

Figure 2. Switch Open



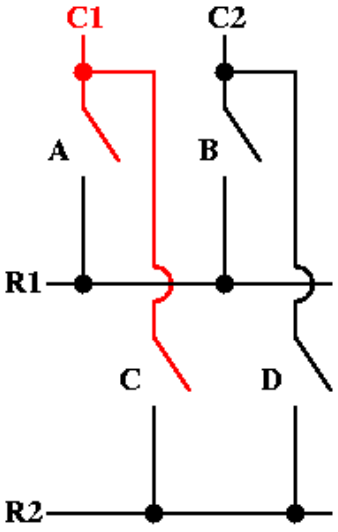
---

3. Scanning to Detect Key Presses

In order to detect key presses, the keyboard controller will scan all columns, activating each one by one. When a column is activated, the controller detects which rows are "activated".

To step through this procedure, the controller activates column C1 and checks rows R1 and R2:

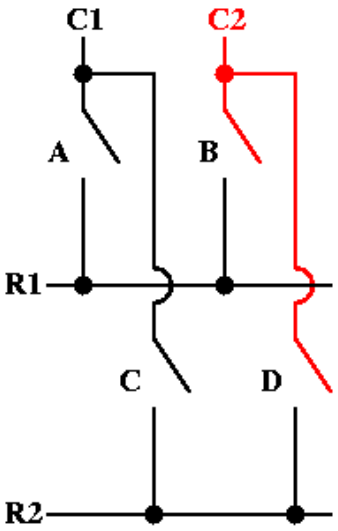
Figure 3. Scanning Column 1



Neither key *A* nor *C* are pressed, so neither row R1 nor R2 is activated. The controller now knows that both nodes C1R1 and C1R2 are deactivated.

Then it activates column C2 and checks rows R1 and R2 again:

Figure 4. Scanning Column 2



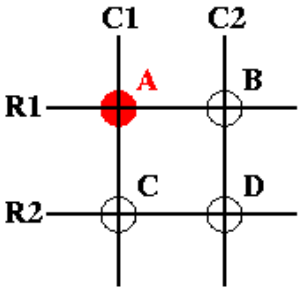
Neither key *B* nor *D* are pressed so neither row R1 nor R2 is activated. The controller now knows that both nodes C2R2 and C2R2 are deactivated.

---

4. Single Key Presses

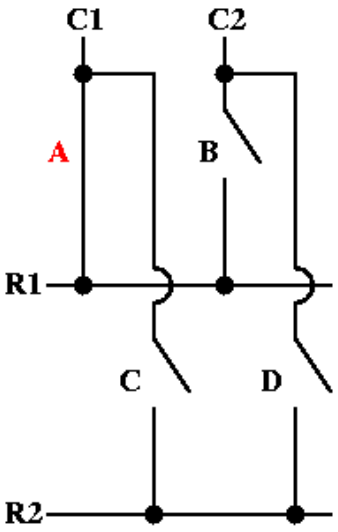
Now, let's say the the *A* key is pressed. The matrix will look like this:

Figure 5. The A Key is Pressed



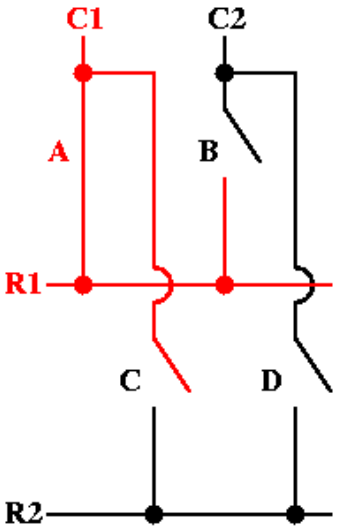
Key *A* corresponds to node C1R1. By going back to the circuit, we can see how C1R1 is detected. First, here is the circuit with switch A closed:

Figure 6. The A Switch is Closed



Walking through the scanning procedure again, the controller activates column C1 and detects which rows are activated:

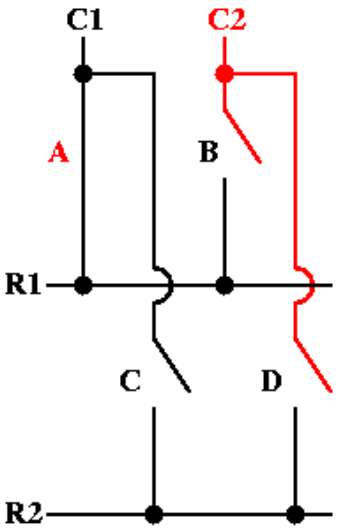
Figure 7. Row 1 is Activated



This time, row R1 is activated. So the controller now knows that node C1R1 is pressed. Since C1R1 corresponds to the *A* key, the controller knows that the *A* key is pressed.

When the controller activates column C2, neither row R1 nor R2 are activated. Both switches B and D are open:

Figure 8. Neither Row is Activated



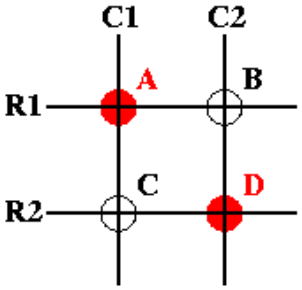
When the *A* key is released, the circuit goes back to the original, and the controller detects the node C1R1 is no longer activated.

---

5. Multiple Key Presses

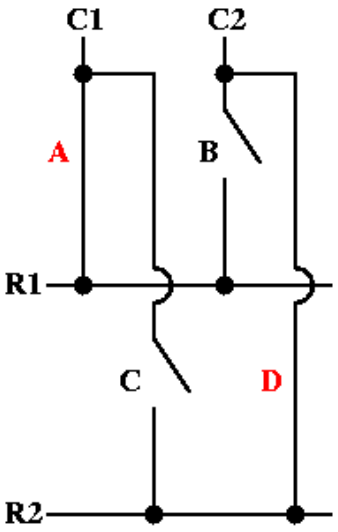
Now we will cover multiple key presses. Let's say that both keys *A and D* are pressed at the same time. The matrix will look like this:

Figure 9. The A and D Keys are Pressed



Both nodes C1R1 and C2R2 should be detected. Here is the circuit with both switches A and D closed:

Figure 10. The A and D Switches are Closed



Scanning column C1 then column C2 produces an outcome like this:

Figure 11. Row 1 is Activated

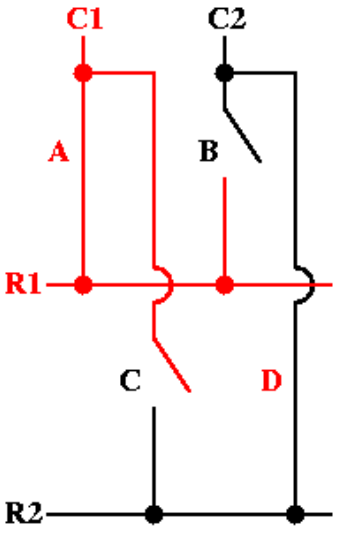
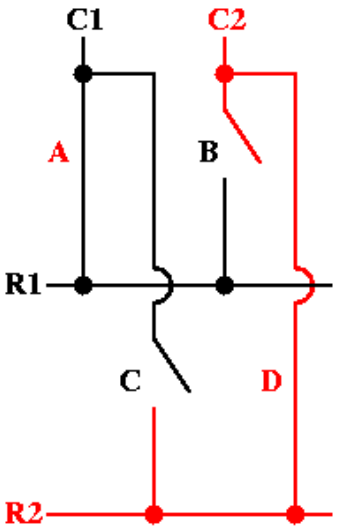


Figure 12. Row 2 is Activated



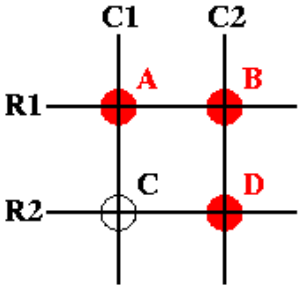
When column C1 is activated, row R1 is active, hence node C1R1 is activated. When column C2 is activated, row R2 is activated, hence node C2R2 is activated.

---

6. Three Simultaneous Key Presses and Ghosting

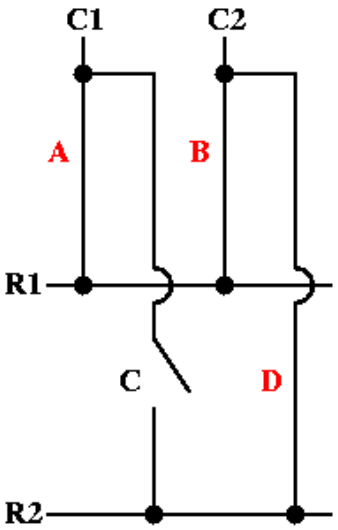
When three keys are pressed at the same time, ghosting may occur. In this example, keys *A*, *B*, and *D* are pressed. The matrix looks like this:

Figure 13. The A, B, and D Keys are Pressed



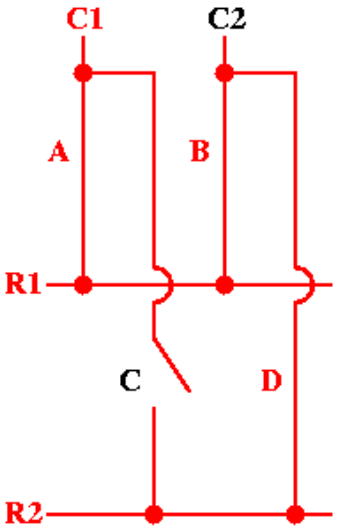
If everything goes well, nodes C1R1, C2R1, and C2R2 will be detected. Let's look at the circuit view again:

Figure 14. The A, B, and D Switches are Closed



When activating column C1, the circuit now looks like this:

Figure 15. Rows 1 and 2 are Activated

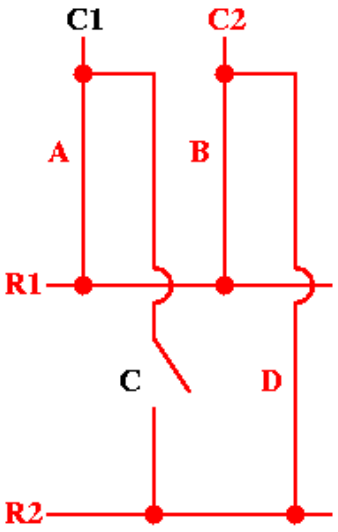


If you get the feeling something is not right, you are correct! This is where the ghosting problem comes to play. Row R1 is activated as well as row R2, so both nodes C1R1 and C1R2 are activated. Node C1R1 is expected as it corresponds to the *A* key that is pressed. However, node C1R2 corresponds to the *C* key. Switch C is open, so the key is not really pressed. The keyboard controller does not know this and incorrectly generates a *C* key press.

What happens is that closing switch *B* and switch *D* at the same time creates an electrical path from C1 to R2, bypassing the open switch *C*. The keyboard does not know that switch *C* is open and generates a "ghost" key press. Ghosting will show up when any 3 corners of a rectangle in the matrix are pressed at once. In my simple example, *any* 3 keys causes ghosting, but in a bigger matrix only 3 corners of a rectangle cause it.

Just for completion, here is the circuit when activating column C2:

Figure 16. Rows 1 and 2 are Activated



As expected, nodes C2R1 and C2R2 are activated corresponding to the *B* and *D* keys.

---

7. The Masking Problem

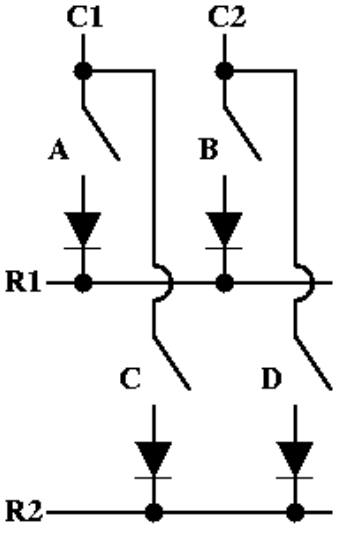
Take the scenario above where keys *A*, *B*, and *D* are all pressed. Now press the *C* key. Given the ghosting affect, nothing changes as the controller already thought that *C* was pressed. Release the *B* key. Aha! The same problem occurs. The key release is not detected because switch *B* is bypassed by the closed switches *A* and *C*.

---

8. Getting Rid Of Ghosting and Masking

By using diodes, both the ghosting and masking problems are eliminated. Just put diodes in series with each switch, like so:

Figure 17. Schematic with Diodes



Let's go back to the scenario where *A*, *B*, and *D* are pressed simultaneously. Activating columns C1 and C2 now look like this:

Figure 18. Only Row 1 is Activated

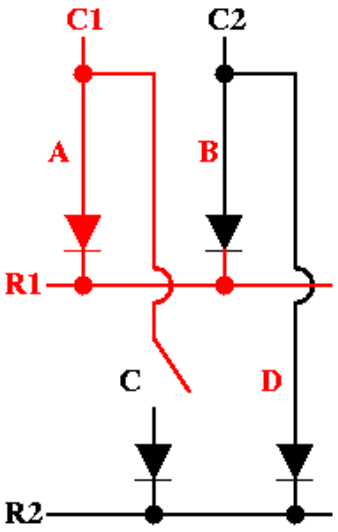
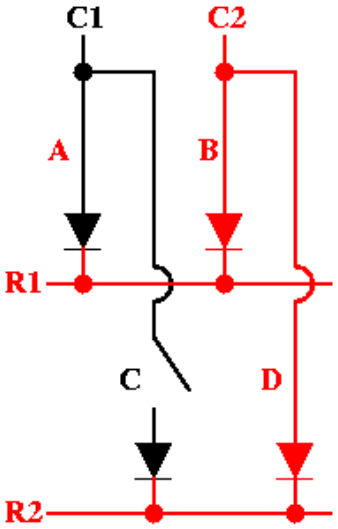


Figure 19. Row 1 and 2 are Activated



Voila! The diodes stop the current from flowing in the "wrong" direction back up switch *B*. When column C1 is activated, only node C1R1 is activated and C1R2 is not. Also, when column C2 is activated, both R1 and R2 as expected.

This fixes the masking problem for the same reason, the diode stops the current.

---

9. What Diode Parts to Use

Now that the theory has been explained, you need to know how to actually build a circuit with diodes to fix this problem. Unfortunately, this is where my knowledge fades. I know the theory but not the practice (that's what happens when you take a college class but never use that information outside the classroom. :-) Nonetheless, I will explain what I do know (or think I know). Thanks goes out to a reader who would like to remain anonymous for clearing up much this information for me.

There are many kinds of diodes for all different purposes. In this case, you need what is called a switching diode. Other common types of diodes are rectifier diodes to rectify AC current to DC, power diodes, which can handle more current without breaking down and/or melting, and everyone's favorite, light emitting diodes (LEDs).

If you go looking at an electronics store, you will probably stumble across a diode by the name of 1N4001. I was going to use these, but when I went to buy them, the clerk said that I should use the 1N4148 due to faster switching time. For over a year, I was unsure if the 1N4001s would actually work, until a reader clarified this issue.

The 1N4001s were designed to rectify the AC wall current. Since the wall current "switches" at 60 times per second, the 1N4001 must be within a 60Hz tolerance. This is plenty fast for a keyboard switch unless you can press a button faster than 60 times per second (doubtful :).

The 1N4148s are designed for fast switching and have a switching time of 4 nanoseconds. Since this is *much* faster than the 1N4001, this is what the clerk was talking about. This makes the 1N4148 the more "proper" and economical choice, even if it is overkill. Since the 1N4148 sell for $0.90 for a pack of 30, this is my recommendation.

---

10. Conclusion

It's pretty simple: Use diodes in series with your switches! What kind of diodes, you may ask? Use 1N4148 switching diodes and you'll have no problems!