# Contact Bounce and De-Bouncing

**The Definition**

Push-button switches, toggle switches, and electro-mechanical relays all have one thing in common: contacts. It's the metal contacts that make and break the circuit and carry the current in switches and relays. Because they are metal, contacts have mass. And since at least one of the contacts is on a movable strip of metal, it has springiness. Since contacts are designed to open and close quickly, there is little resistance (damping) to their movement.

Because the moving contacts have mass and springiness with low damping they will be "bouncy" as they make and break. That is, when a normally open (N.O.) pair of contacts is closed, the contacts will come together and bounce off each other several times before finally coming to rest in a closed position. The effect is called "contact bounce" or, in a switch, "switch bounce" ***See Figure 1.*** Note that contacts can bounce on opening as well as on closing.

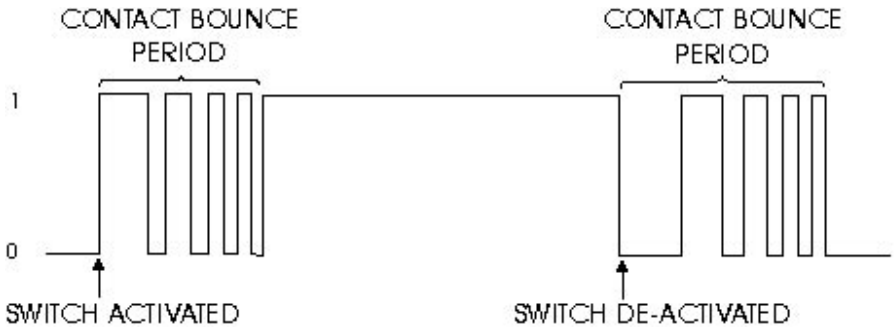


*Figure 1*

**The Problem**

If all you want your switch or relay to do is turn on a lamp or start a fan motor, then contact bounce is not a problem. But if you are using a switch or relay as input to a digital counter, a personal computer, or a micro-processor based piece of equipment, then you must consider contact bounce. The reason for concern is that the time it takes for contacts to stop bouncing is measured in milliseconds. Digital circuits can respond in microseconds.

As an example, suppose you want to count widgets as they go by on a conveyor belt. You could set up a sensitive switch and a digital counter so that as the widgets go by they activate the switch and increment the counter. But what you might see is that the first widget produces a count of 47, the second widget causes a count of 113, and so forth. What's going on? The answer is you're not counting widgets, you're counting how many times the contacts bounced each time the switch is activated!

**The Solution**

There are several ways to solve the problem of contact bounce (that is, to "de-bounce" the input signal). Often the easiest way is to simply get a piece of equipment that is designed to accept "bouncy" input. In the widget example above, you can buy special digital counters that are designed to accept switch input signals. They do the de-bouncing internally. If that is not an option, then you will have to do the debouncing yourself using either hardware or software.

**Using Hardware**

A simple hardware debounce circuit for a momentary N.O. push-button switch is show in ***Figure 2***. As you can see, it uses an RC time constant to swamp out the bounce. If you multiply the resistance value by the capacitance value you get the RC time constant. You pick R and C so that RC is longer than the expected bounce time. An RC value of about 0.1 seconds is typical. Note the use of a buffer after the switch to produce a sharp high-to-low transition. And remember that the time delay also means that you have to wait before you push the switch again. If you press it again too soon it will not generate another signal.

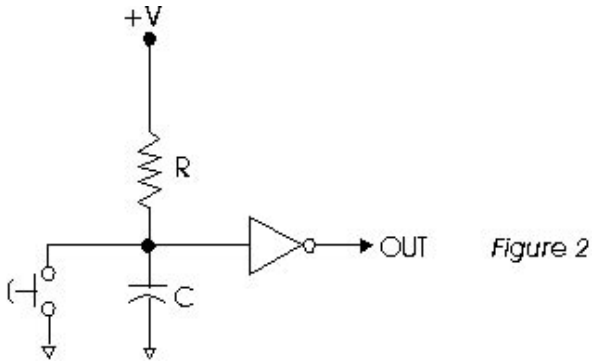

Figure 2

Another hardware approach is shown in ***Figure 3.*** It uses a cross-coupled latch made from a pair of nand gates. You can also use an SR (sometimes called an SC) flip flop. The advantage of using a latch is that you get a clean debounce without a delay limitation. it will respond as fast as the contacts can open and close. Note that the circuit requires both normally open and normally closed contacts. In a switch, that arrangement is called "double throw". In a relay, that arrangement is called "Form C"

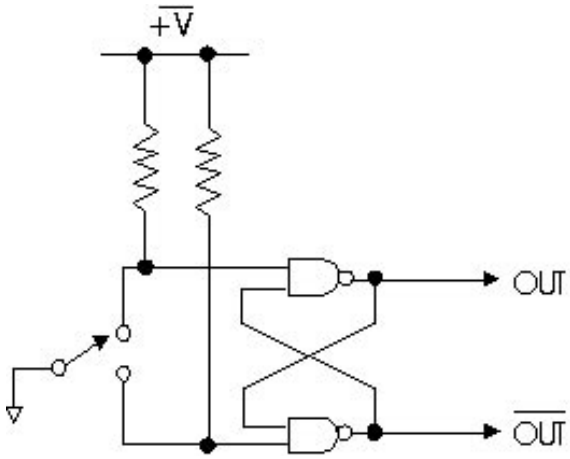

Figure 3

**Using Software**

If you're the one developing the digital "box", then you can debounce in software. Usually, the switch or relay connected to the computer will generate an interrupt when the contacts are activated. The interrupt will cause a subroutine (interrupt service routine) to be called. A typical debounce routine is given below in a sort of generic assembly language.

```
DR:       PUSH     PSW      ;  SAVE PROGRAM STATUS WORD
LOOP:     CALL     DELAY    ;  WAIT A FIXED TIME PERIOD
          IN       SWITCH   ;  READ SWITCH
          CMP      ACTIVE   ;  IS IT STILL ACTIVATED?
          JT       LOOP     ;  IF TRUE, JUMP BACK

          CALL     DELAY    ;
          POP      PSW      ;  RESTORE PROGRAM STATUS
          EI                ;  RE-ENABLE INTERRUPTS
          RETI              ;  RETURN BACK TO MAIN PROGRAM
```

The idea is that as soon as the switch is activated the Debounce Routine (DR) is called. The DR calls another subroutine called DELAY which just kills time long enough to allow the contacts to stop bouncing. At that point the DR checks to see if the contacts are still activated (maybe the user kept a finger on the switch). If so, the DR waits for the contacts to clear. If the contacts are clear, DR calls DELAY one more time to allow for bounce on contact-release before finishing.

A debounce routine must be tuned to your application; the one above may not work for everything. Also, the programmer should be aware that switches and relays can lose some of their springiness as they age. That can cause the time it takes for contacts to stop bouncing to increase with time. So the debounce code that worked fine when the keyboard was new might not work a year or two later. Consult the switch manufacturer for data on worst-case bounce times..