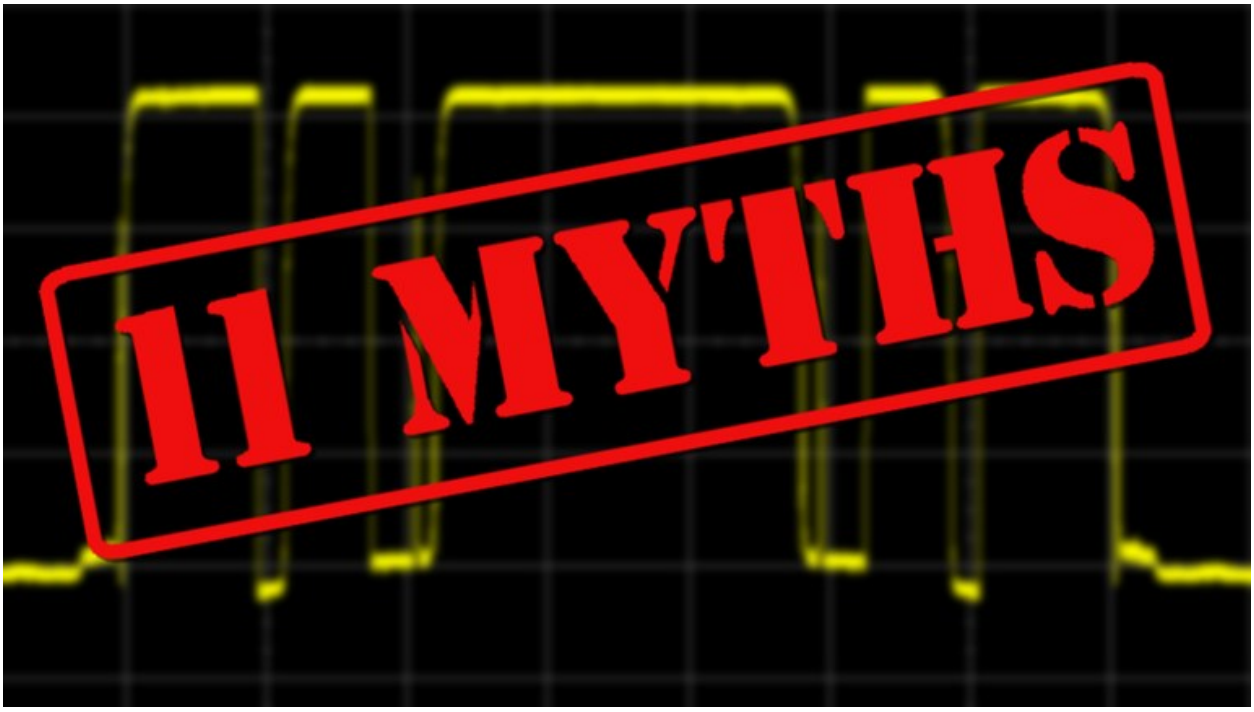
ElectronicDesign.
FEB 23, 2021
Link to Article



11 Myths About Switch Bounce/Debounce

An often-overlooked issue, “switch bounce” can become particularly problematic in safety- and mission-critical systems. Thus, the need for “switch debounce” tactics. This article debunks myths that surround both of them.

Clive "Max" Maxfield
Related To:
LogiSwitch

What you'll learn:

- What causes a switch to bounce, and how many times will a switch bounce?
- Methods of debouncing a switch.
- What is NoBounce technology?

Most people, especially non-engineers, tend not to think about switches beyond the act of actually using them. If they do think about them, they typically assume that the switch is an ideal device that's either Off (open, inactive) or On (closed, active). In reality, much like dropping a ping-pong ball on a wooden table, when a change is instigated, the switch will bounce On-Off-On-Off… multiple times before settling in its new state.

Not surprisingly, this effect is known as "switch bounce." This isn't an issue for things like light switches because it happens too quickly for people to notice it. The problem arises when the signal from a switch is fed into an electronic system that perceives each bounce as a separate switch event. In the case of a floor-mounted switch counting the number of people entering a room, for example, a single person might end up being counted as a multitude.

The term "switch debounce" refers to the process of mitigating the effects of a switch's bouncing. The problem is that a lot of confusing and contradictory information abounds about switch bounce and debounce, and even practicing engineers can mess things up if they're not careful. Below are 11 common myths associated with switch bounce/debounce.

1. Switch bounce is a problem only with toggle switches.

Even in the case of people who are aware of switch bounce, many of them are under the impression that this phenomenon affects only toggle switches. For some reason, they assume that pushbutton switches and limit switches (such as those used in robotics and industrial-automation applications) are immune to switch bounce.

Actually, though, just about every type of switch on the planet bounces. The only switches that don't bounce are specialist devices like mercury tilt switches, which the vast majority of people never use anyway.

2. Switch bounce isn't a problem with modern devices.

Some people are under the misapprehension that only "old switches" bounce, and that "new switches" fabricated using modern manufacturing

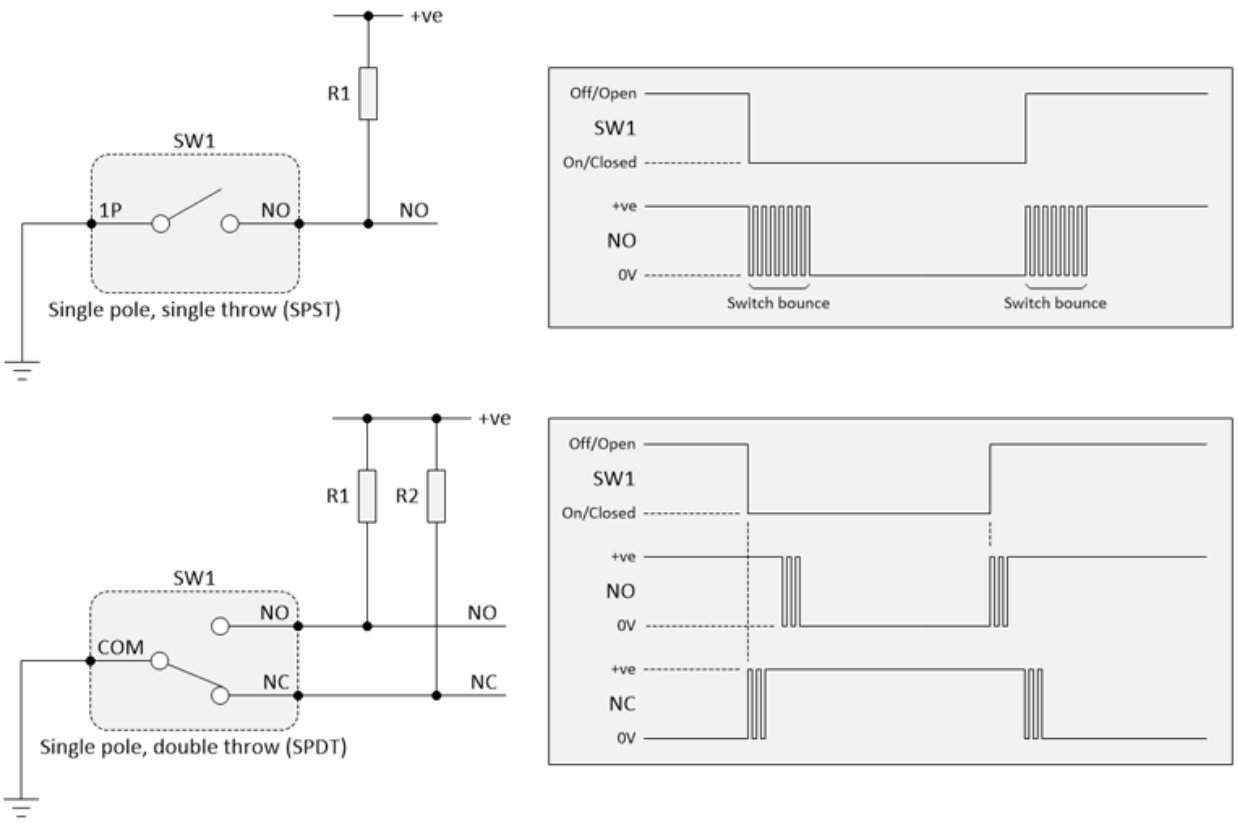
methods are unaffected by this problem. Sadly, this isn't the case—switches bounce; that's what they do.

Others believe that switches include special debouncing circuitry. Actually, this myth has a hint of truth because, although the overwhelming majority of switches are presented "as-is" (requiring users to debounce them using hardware or software techniques), LogiSwitch, for example, includes its unique NoBounce adaptive debounce technology in a series of pushbutton and limit switches.

3. Switches only bounce a couple of times.

According to *The Art of Electronics* by Horowitz & Hill (Page 506 of the Second Edition), "When the switch is closed, the two contacts actually separate and reconnect, typically 10 to 100 times [...]"

Furthermore, each switch is different. Switches of the same type may exhibit very different bounce characteristics and these characteristics may themselves vary as a function of environmental conditions such as temperature and humidity.



Examples of switch bounce. (Image source: Max Maxfield)

4. Switches only bounce when they are turned on.

Many users consider the act of turning a toggle switch to its "On" position to be more important that returning it to its "Off" position. Similarly, many users believe that pressing a pushbutton switch is intrinsically more significant than releasing it to its original position.

In fact, in an electronic system, transitioning a switch from any state to any other state can be used to initiate or terminate subsequent actions. This is important because that switch bounce can occur when a toggle switch is turned on or off, when a pushbutton switch is pressed or released, and when a limit switch is activated or deactivated.

5. Switch bounce doesn't last more than one millisecond.

The rumor that switch bounce doesn't last longer than 1 ms—that is, one thousandth of a second—is a common tale that's told to young,

inexperienced engineers. This value is often imparted by older, more experienced engineers who learned it from their own mentors deep in the mists of time. The problem is that it's simply not true.

In his *Guide to Debouncing*, embedded-systems legend Jack Ganssle gathered a selection of different switches, put them on a test bench, and activated/deactivated each switch 300 times, logging the minimum and maximum amount of bouncing for both the opening and closing of the contacts. Although some of the switches ceased bouncing in less than 1 ms, the average was 1.6 ms, and the maximum was 6.2 ms.

6. Debouncing using monostables is a good idea.

Using a monostable is actually a bad idea, since it produces a pulse rather than a clean transition when the switch is opened or closed. This isn't the behavior that most designs expect from a switch. More details are available in Part 4 of the *Ultimate Guide to Switch Debounce* series of articles.

7. Debouncing switches using hardware solutions is "old school" and no longer done.

It's certainly true that the original switch debouncing approaches involved hardware, such as resistor-capacitor (RC) delays driving buffers with Schmitt trigger inputs for single-pole single-throw (SPST) switches, or set/reset (SR) latches formed from back-to-back NAND or NOR gates for single-pole double-throw (SPDT) switches.

It's also true that many designers of modern systems prefer to use software techniques to debounce switches. However, there are occasions when mitigating switch bounce using a hardware solution offers the best (sometimes only) option.

8. Debouncing switches using software is the only way to go.

It's certainly true that debouncing switches in software can offer advantages, not the least of which is reducing the number of components, costs, and real estate on the printed circuit board (PCB). The problem is that many software developers are unfamiliar with the physical characteristics of switches—some software developers aren't even aware that switch bounce exists.

Switch bounce is very difficult to simulate in systems test and usually isn't implemented. Thus, switch-debounce routines may be changed or removed as the software evolves, and the problem not detected during system test.

Even if the software developer does understand debounce, some applications use tiny processors that are constrained in terms of memory and clock speed. Thus, the developer may be reluctant to devote code space and clock cycles to implement switch-debounce routines. Further, a variety of simple, non-processor-based systems also require switches to be debounced.

9. Using interrupts to implement debounce is a good idea.

Sometimes, systems have one or more switches connected to the interrupt pins on a microcontroller (MCU). This may be a valid implementation if the designers wish the MCU to respond to a transition on the switch as fast as possible. If this is the case, then the signal from the switch should be debounced in hardware before being presented to the interrupt input, thereby avoiding the possibility of multiple interrupts and leaving the MCU free to service the interrupt with all possible haste.

The problem arises if someone decides to use the interrupt service routine (ISR) to debounce the switch, thinking that this is a good idea. It isn't, because it either requires the ISR to be triggered on every bounce, or the ISR to wait for the switch to stop bouncing, thereby preventing anything else from happening.

10. All dedicated switch-debounce integrated circuits work the same way.

Several dedicated switch-debounce ICs are available on the market. Different families may support different numbers of switches. Some devices require the addition of external components, such as a resistor and capacitor, to fine-tune the timing; others require an external clock to be provided.

A relative newcomer to the party is LogiSwitch with its NoBounce technology mentioned above. Available in lead-through-hole (LTH) and surface-mount-technology (SMT) packages, The company's ICs support 3, 6, or 9 channels (switches). Unlike older solutions, these devices don't require any external components and they work with any supply voltage between 2.3 and 5.5 V.

11. Flags are required to keep track of the switch's state.

A software debounce routine requires looping to detect the switch status. If the switch starts in its Off/Inactive state, the code is essentially going to be saying: "Has the switch gone active? Has the switch gone active? Has the switch gone active? …"

When the switch does eventually transition to an active state, the software must address the switch's bouncing (this can be achieved with a variety of techniques) and perform whatever task is expected of it. At this point, the code has another problem, because it now has to wait for the switch to return to its inactive state ("Has the switch gone **in**active? Has the switch gone **in**active? Has the switch gone **in**active? …") before starting the whole cycle again.

As a result, the software developer must establish and maintain a flag for each switch to keep track of its current state—inactive waiting to go active, or active waiting to go inactive. A better approach is taken with the LogiSwitch LS1xx series of debouncer ICs, which involves a unique single-wire request/acknowledge-based handshake protocol. The programmer no longer needs to maintain a flag in software because the

IC essentially implements this function in hardware. All the program has to do is wait for the switch to go active, perform whatever task is required of it, use the handshake to “clear” the IC, and then return to waiting for the next time the switch goes active.

Conclusion

Considering the countless billions of toggle, pushbutton, and limit switches in the world, one would think there would be a standard way to debounce them by now. The sad truth, however, is that almost every hardware designer or software developer has his or her own ways of doing things. Sometimes they work as hoped. Other times they don’t.

Usually, switch bounce is merely annoying, although the way things are these days, anything that can minimize annoyances is a good thing to do. On the other hand, if a switch is connected to some safety- or mission-critical system, and if the system's designers haven’t adequately addressed its switch-bounce characteristics, then the chances are that someone, sometime, is going to have a very serious problem. Proper switch debounce is an absolute requirement.