The dirt on switching

Switches can do some really odd things. Most engineers learn this soon after connecting a switch or relay to a digital system. Switches don't make and break cleanly on the time scales of digital systems. Instead, a typical switch makes multiple transitions during the tens of milliseconds required to open or close it. Commonly called switch bounce, this behaviour is an inescapable fact of life, as John Wettroth explains.

After connecting a standard switch to a digital counting circuit, you can observe several counts on opening and several counts on closing, **Figs 1-2**. This erratic action can wreak havoc on data, because the exact number of counts does not necessarily repeat in the long term.

Switch bounce is not consistent from unit to unit, lot to lot, or even over the life of an individual switch. Membrane switches and some other types appear not to bounce when new, but all mechanical switches bounce sometimes. Nothing can ensure that another switch of the same type will act the same way, or that a particular switch will remain bounce-free as it ages.

In addition to bounce, switches and digital systems have other annoying habits. Strange things happen, for example, when you run switch wiring in a noisy industrial environment. An open switch has high impedance by definition, so interfering signals have an easy load to work against. Any noise impulse that is capacitively or inductively coupled to the switch wiring can cause phantom switch closures.

Imagine a PLC, i.e. a programmable-logic controller, switching a motor through a hefty relay. A limit switch placed near the motor provides position feedback to a digital input on the PLC.

When the PLC tells the motor to start, a surge of current flowing to the relay and motor can couple to other conductors in the long wiring runs, causing ground bounce or a capacitively coupled spike in the digital input. If not properly designed, the PLC may interpret this spike as a premature switch closure and shut down the operation.

Similar things can happen when the PLC turns the load off, due to the effect of wiring capacitance, wiring inductance, and the inductive kick of the relay and motor. If the PLC and its digital inputs are not properly designed these spikes and transients can cause erroneous readings on the digital inputs.

The digital and analogue inputs on equipment used in the home, office, and industry are subject to the effects of overvoltage, voltage transients and ESD strikes. Improper wiring, miscellaneous fault conditions and power-supply sequencing – in which one box with power off is connected to another with power on, even temporarily – cause overvoltage.

Voltage transients are often associated with capacitively or inductively coupled spikes, as discussed above. ESD can strike a connector, an operator console, or a terminal strip during installation. Any of these transients can cause destruction if the system latches up. If not destructive, they can cause CPU resets, watchdog overflows, and other erratic operation.

System designers should be aware of these problems and the methods used to combat them. One solution for such interface problems is a new series of ICs. Available in low-cost, easy-to-use configurations, these devices offer foolproof, software-free debouncing along with protection against overvoltage and ESD.

This article highlights the application of IC switch debouncers while describing the classic methods for thwarting overvoltage, voltage/current spikes, switch bounce, and ESD.

Switch bounce

If asked, most engineers would say that switches are debounced in software, and that debouncing is no problem. Both assumptions are true if you pay proper attention to the details. Software debouncing takes care of the bounce, but does not address the problems of overvoltage, ESD, or other transients.

Debouncing with resistors and capacitors is also possible. In general, you need a pull-up resistor, a resistor and capacitor in series, a resistor to the input of a Schmitt-trigger buffer, and often a diode to ensure that the capacitor charge doesn't force lots of current through the buffer's input-protection network during power-down.

The resulting parts count can be unwieldy for multiple-input systems, **Fig. 3**, so this approach will not be covered in any detail.

Debouncing via software

Debouncing via software is the primary method in use today. A good debouncing routine is actually real-time software that acts like a simple low-pass digital filter. Non-switch digital inputs are often routed through debounce filters as well. That technique can eliminate short transients at the input by ensuring a stable state before reporting the input open or closed.

The pseudo code in **List 1** illustrates a software-debounce routine for one input. It accommodates multiple inputs if you generalise the routine and use pointer-based variables, etc. Though a mediocre approach at best, this type of routine is often used in spite of the problems and flaws outlined below.

The routine debounces switch closures, but it will accept 'open' as a legitimate state even when the switch is bouncing. Though unintentional, this asymmetrical operation might be acceptable in keypads and other systems that take action on closures but not on opens. For general-purpose inputs, you should debounce both edges.

Another shortcoming is that this routine assumes the switch is open if not closed, thereby ignoring a third state in which the switch is unstable – i.e. still bouncing. A better routine should therefore report the last non-bouncing state until the switch reaches a new debounced state. This action can also cause problems, however. In such cases, the software should recognise a third state of 'changing.'

Sampling wastes

Many debounce routines sample the input repeatedly, waiting for it to remain in the same state for a prearranged number of samples. If the switch changes state during that interval, the routine tests the new state for stability in the same way. This action can cause large delays that eat up a lot of CPU time.

As an extreme case, a programmable logic controller with high frequency applied to one of its general-purpose input ports – whether inadvertent, on purpose, or due to failure – would completely hang the processor. A watchdog timer might bring back the processor, but the problem would recur indefinitely; not a robust design.

Further, you need a lot of memory and code to debounce a large industrial system with lots of inputs, such as a PLC or general-purpose input board.

Each input requires a closed counter, an open counter, and two bits to define its state.

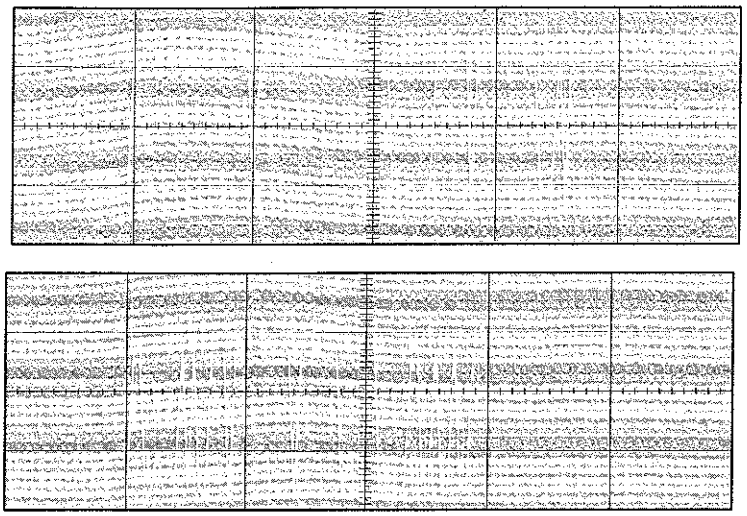

Fig. 1. This rising-edge switch bounce for a small push-button switch shows an approximate 5ms bounce interval that includes ten transitions. Like a bouncing ball, the switch-action frequency increases towards the right.

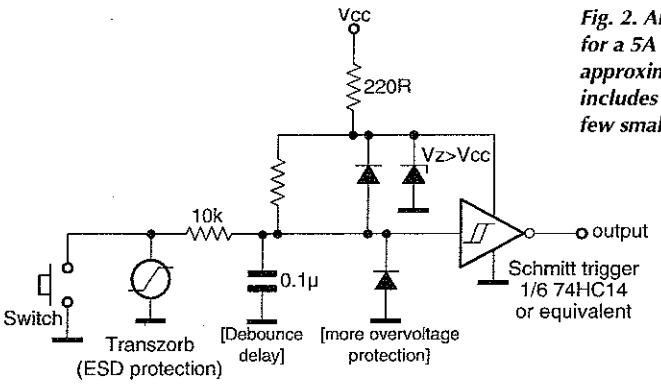

Fig. 2. Another rising-edge switch bounce for a 5A contact relay shows an approximate 5.5ms bounce interval that includes 20 full-amplitude transitions and a few smaller ones.

Fig. 3. Discrete components can provide debouncing with protection against ESD and overvoltage, but a properly designed discrete interface that accounts for all likely faults is unwieldy for more than a few inputs.

Transient and ESD suppression

The standard prevention for ESD is a transient suppressor or MOV device at each external input.

Quad and octal Tranzorbs, for example, are straightforward and relatively inexpensive devices that can reduce clutter and real estate requirements, but care must be taken to avoid cross coupling of fault currents. This approach is common in industrial and automotive systems, where engineers understand the peril of omitting such protection.

A good practice is to connect a 220Ω resistor in series with the V_{CC} line for port input devices. A common CMOS input device like the octal 74HC244 or 74HC573, for example, draws very little current. Should it latch up, the 220Ω resistor limits the current and power dissipation to a safe level.

Power cycling may still be necessary, though. In general, you should not connect the port pins of a microcontroller to the external inputs directly. Latchup is a problem, but radiated EMI is likely to be even worse.

Because a part cannot latch up unless sufficient current is injected into one of its pins, some designers believe that resistors in series with CMOS digital inputs prevent these problems. Indeed, the threshold for SCR latchup in modern CMOS ICs can exceed 50mA.

This high current threshold, covered in the next section, actually protects against overvoltage to some extent, but is not necessarily effective for ESD. A 15kV ESD strike can force significant currents through parasitic paths and around resistors, and it can force a large current even through 100kΩ.

Overvoltage protection

Overvoltage protection enables a system to withstand continuous and longer-term-transient inputs that extend beyond the rails.

As an example, an IC with no V_{cc} applied has 24V from an external source applied to the inputs. Such applied voltage often 'backdrives' the protection networks, forcing voltage onto the power rail inside a system.

One effective countermeasure is a

List 1. Action sequence for debouncing using software.

Action	Comments
1. Input timer: expired?	A timer bit is polled in the main routine
2. Return if no timer	Go do something more useful
3. Get input bit	The "bouncy" input
4. Count++ if high; clear else	Increment a counter if input is high
5. If count >4 state=1, else 0	Check counter and clamp it at 4
6. Return input state	State is debounced

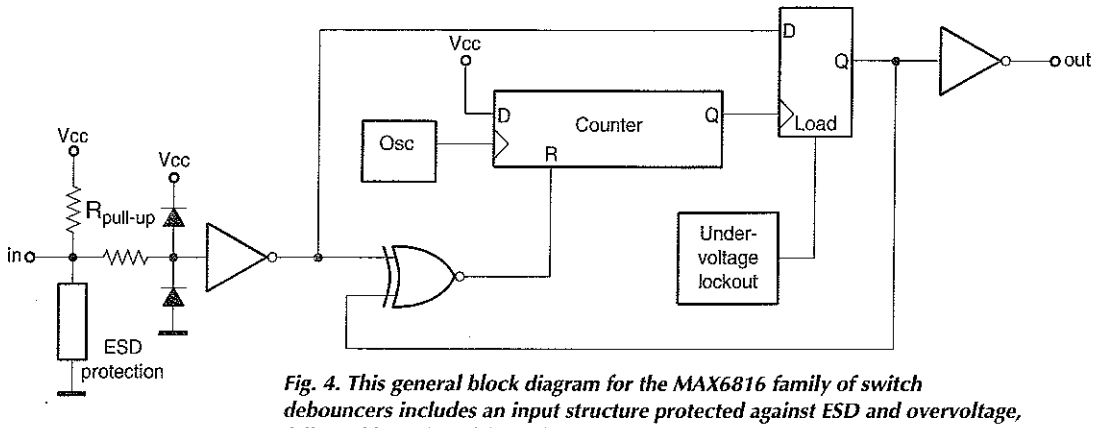

Fig. 4. This general block diagram for the MAX6816 family of switch debouncers includes an input structure protected against ESD and overvoltage, followed by a digital filter that debounces the input and applies undervoltage lockout.

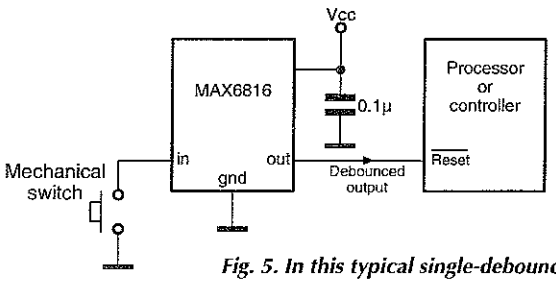

Fig. 5. In this typical single-debouncer application, the only components are a small bypass capacitor and the 4-pin SOT-23 package.

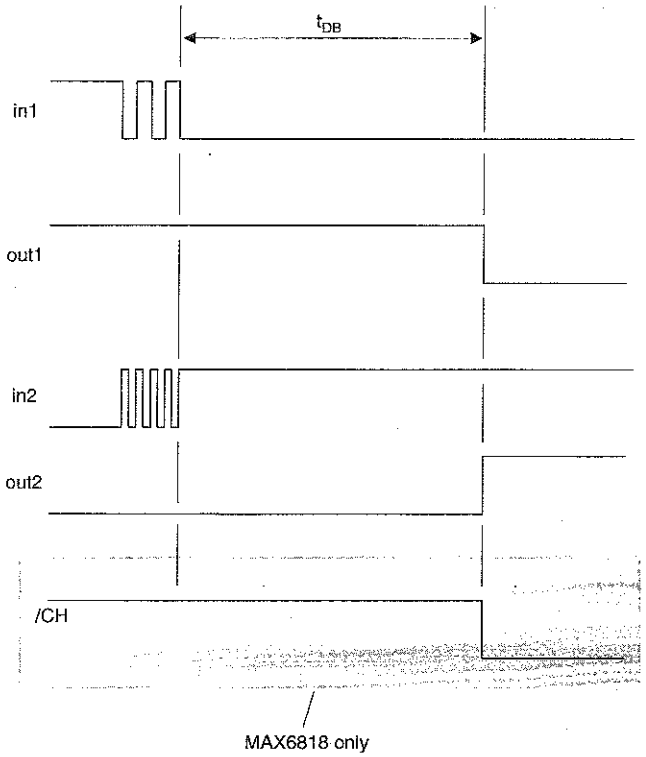

Fig. 6. Timing diagram for the MAX6816 switch-debouncer family shows that the outputs change state about 40ms after the inputs become stable. An additional MAX6818 output that indicates a change of state for any of the inputs. The $\overline{CH}$ output reduces polling overhead, especially in multiple-input systems.

resistor in series with the input, acting against protection diodes tied to the rails. A zener diode across the V_{cc} rails of the input port will also help.

To ensure that the protection circuits won't fail under worst-case conditions, you should calculate the maximum power dissipation of this zener and the series input resistors.

Integral debouncing ESD protection

Several years ago, Maxim engineers saw the need for a simple interface device capable of debouncing a switch while protecting it against ESD and overvoltage.

Some customers were using the manual-reset input of microcontroller-supervisory ICs like the *MAX811* just to obtain the single-channel debouncer function in a SOT-23 package. Others were using ESD-protected RS232 transceivers as general-purpose digital-input devices.

Customers were attracted to the RS232 ICs because they could handle low-voltage transitions while withstanding high voltage and ESD. Putting these facts together, Maxim produced a line of switch debouncers that incorporated ESD protection and robust input features, **Figs 4-5**.

The *MAX6816* is a single-switch debouncer in a four-pin SOT-23 package, while the *MAX6817* is a dual-switch debouncer in a six-pin SOT-23 package. They provide debounce logic and a digital filter, input overvoltage protection to ±25V, and ESD protection to ±15kV for harsh industrial environments.

Operating on single supply voltages in the range 2.7V to 5.5V, they draw typical supply currents of only 6µA. They also provide undervoltage-lockout circuitry that ensures correct output states on power-up.

Because the proprietary ESD-protection structure at each input includes an overvoltage clamping diode and 63kΩ pull-up resistor, these ICs provide a direct interface to the switch without external components. Their nominal debounce delay of 40ms ±20ms masks the bounce produced by even the ugliest of switches, **Fig. 6**.

An octal debouncer

The *MAX6818* octal-switch debouncer is designed for data-bus interfacing, **Fig. 7**. It monitors eight switches, providing a change-of-state output, $\overline{CH}$, and three-stated data-bus output in addition to the debounce and input-protection features of the single and dual parts. In particular, its $\overline{CH}$ output greatly simplifies the polling and interrupting of microprocessors.

Each time the system reads the data outputs by driving the enable line low, the IC resets high. Output $\overline{CH}$ then goes low when any input changes state. The *MAX6818* is pin-compatible with the 74HC573 and other standard, 20-pin octal logic devices. It handles multiple inputs with ease.

These new switch debouncing ICs solve multiple problems associated with connecting digital systems to noisy, transient-prone, 'bouncing' inputs. They make systems more robust and reliable by simplifying design, reducing CPU time and overhead, and replacing multiple passive components. ■

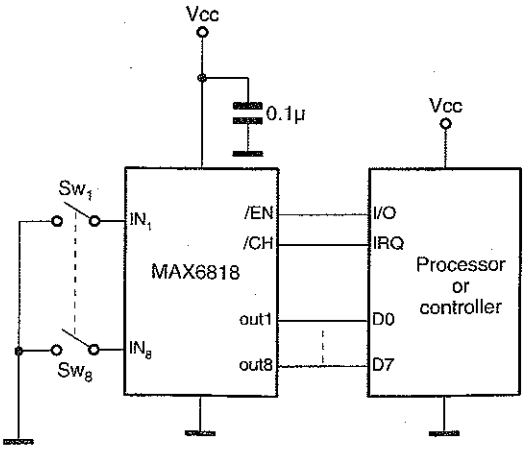

Fig. 7. In a typical application, the MAX6818 data outputs remain three-stated until $\overline{EN}$ is pulled low. The change output, $\overline{CH}$, is reset high following each read, and set low following a change of state at any input. It can either be polled by the system or tied to an interrupt as shown.