tom side of a breadboard. Wire-wrap sockets are still available for DIPs in standard sizes for two or three wraps per post. The actual wrapping is accomplished with a special tool—either manual or automatic.

A benefit of wire-wrapping is that dense wiring can be achieved without the risk of melting through insulation with a hot soldering iron. Changes in connectivity are made by unwrapping a wire. However, if the wire to be unwrapped is at the bottom of a stack of other wires, those others may need to be removed as well. A fully manual wire-wrap process requires that a wire be cut to length and stripped at each end to expose approximately 1 in of bare wire, and the wrapping tool is turned around the square post at each end. Automatic wire-wrap guns are available that automatically strip and wrap wire, making the process move fairly quickly. The popularity of wire-wrapping has diminished significantly over time as clock frequencies have increased and PCBs have become much less expensive in small prototyping quantities. It may still be an appropriate construction technique for those who are less comfortable with a soldering iron and who require very dense wiring.

19.4 MICROPROCESSOR RESET

Almost all digital systems hinge around some type of microprocessor that controls the basic operation of specific peripherals. Smaller embedded systems may use microcontrollers that contain integrated microprocessor and memory elements. Others include discrete microprocessors. Regardless of the form, a microprocessor requires a clean initialization sequence for it to begin executing the program that has been designed for it. It is advisable to design a simple and reliable scheme for the microprocessor to boot so that initial system bring-up can proceed smoothly. Once a microprocessor successfully boots, it can be used to run software that helps with the remainder of system bring-up. Functions such as memory debug, accessing control bits in peripheral logic, and setting up a debugging console that can be accessed from a terminal program. All require that the microprocessor be alive. When a microprocessor doesn't boot correctly, it is difficult to make further progress, because the microprocessor is usually the gateway to the rest of the system.

Reset is the first hardware element, subsequent to stable power supplies, that a microprocessor needs to boot. Some microcontrollers have built-in power-on-reset circuits that guarantee a valid reset pulse to the internal microprocessor. Other microprocessors require that an external reset pulse be applied. While not complicated, generating a reliable power-on-reset has eluded more than one engineer. Dedicated power-on-reset ICs have become available in recent years that all but guarantee clean reset behavior once the power supplies become stable. At their simplest, these devices have three terminals (power, ground, and reset), and reset is held active-low for several hundred milliseconds after power passes a predetermined threshold. More complex devices have multiple power inputs for multivoltage systems, and the deassertion of reset occurs only when all power inputs have exceeded certain thresholds for a minimum time. Power-on-reset ICs are available from companies including Linear Technology, Maxim, and National Semiconductor.

When a dedicated power-on-reset chip is unavailable, the function can be implemented using discrete components in many configurations. Two simple schemes involving an RC circuit along with a discharge diode are shown in Fig. 19.7. Both circuits hold the microprocessor in reset for approximately 10 ms after power is applied. The first circuit uses just three passive components and starts out with RESET* at logic 0. As the capacitor charges, it reaches the logic-1 voltage threshold of the microprocessor. A diode is present to rapidly discharge the capacitor when power is removed. It becomes forward biased as V_{CC} drops and a charge is present on the capacitor. This ensures that the reset circuit will behave properly if the system is quickly turned on again and also prevents a capacitor discharge path through the microprocessor. Incomplete discharge is more likely with a larger RC

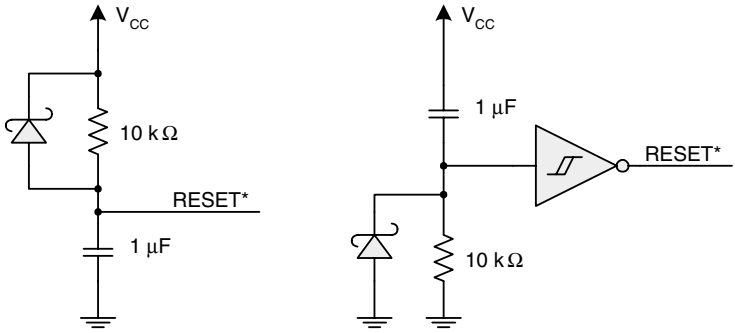

FIGURE 19.7 Discrete power-on-reset circuits.

time constant, as is required by some microprocessors. Better results are obtained using a Schottky diode, because its lower forward voltage discharges the capacitor to a lower voltage.

The second circuit is more robust, because it uses a Schmitt trigger to drive the microprocessor's input, guaranteeing a clean digital transition despite variations in the slope of the RC voltage curve. This is especially helpful when long RC time constants are required to generate long reset pulses as dictated by a microprocessor. A 74LS14 or similar Schmitt-trigger logic gate may be convenient to design into a system, and it can be used in places other than the power-on-reset circuit. Alternatively, a smaller voltage comparator can be used to implement the same function by designing in hysteresis. Before power-on, the inverter input node is at 0 V. At power-on, the voltage step of the power supply passes through the capacitor, because the voltage across the capacitor is initially 0 V and brings the input node to a logic-1 voltage, which in turn causes RESET* to be driven to logic 0. The resistor immediately begins pulling the voltage toward ground and eventually causes RESET* to be deasserted. The diode is present to clamp the inverter input node to ground during power-down. Clamping is desirable, because the resistor has already pulled the input node to ground and, without the diode, a negative V_{CC} step would force the input node to a negative voltage. When the diode is present, the input node remains near 0 V and is able to serve its intended purpose during an immediate power-on. The diode also prevents a large negative voltage from potentially damaging the inverter.

Power-on is not the only condition in which a microprocessor reset may be desired. Especially during the debugging process, it can be very useful to have a reset button that can quickly restart the system from a known initial state when software under development encounters fatal bugs. Many of the aforementioned power-on-reset ICs contain circuitry to *debounce* an external pushbutton. When a button is pushed, it may appear that a clean electrical connection is made and then broken when the button is released. In reality, the contact and release events of a button are noisy for brief periods of time as the internal metal contacts come into contact with each other. This noise or bounce may last only a few milliseconds, but it can cause a microprocessor to improperly exit its reset state. Debouncing is the process of converting the noisy edges of a pushbutton into a clean pulse. Filtering is a general solution for debouncing a noisy event and can be performed in an analog fashion or digitally by taking multiple samples of the event and forcing the bouncing samples to one state or the other.

19.5 DESIGN FOR DEBUG

"To err is human" is a truism that directly applies to engineering. The engineering process is a combination of design and debugging in which inevitable problems in the original implementation are