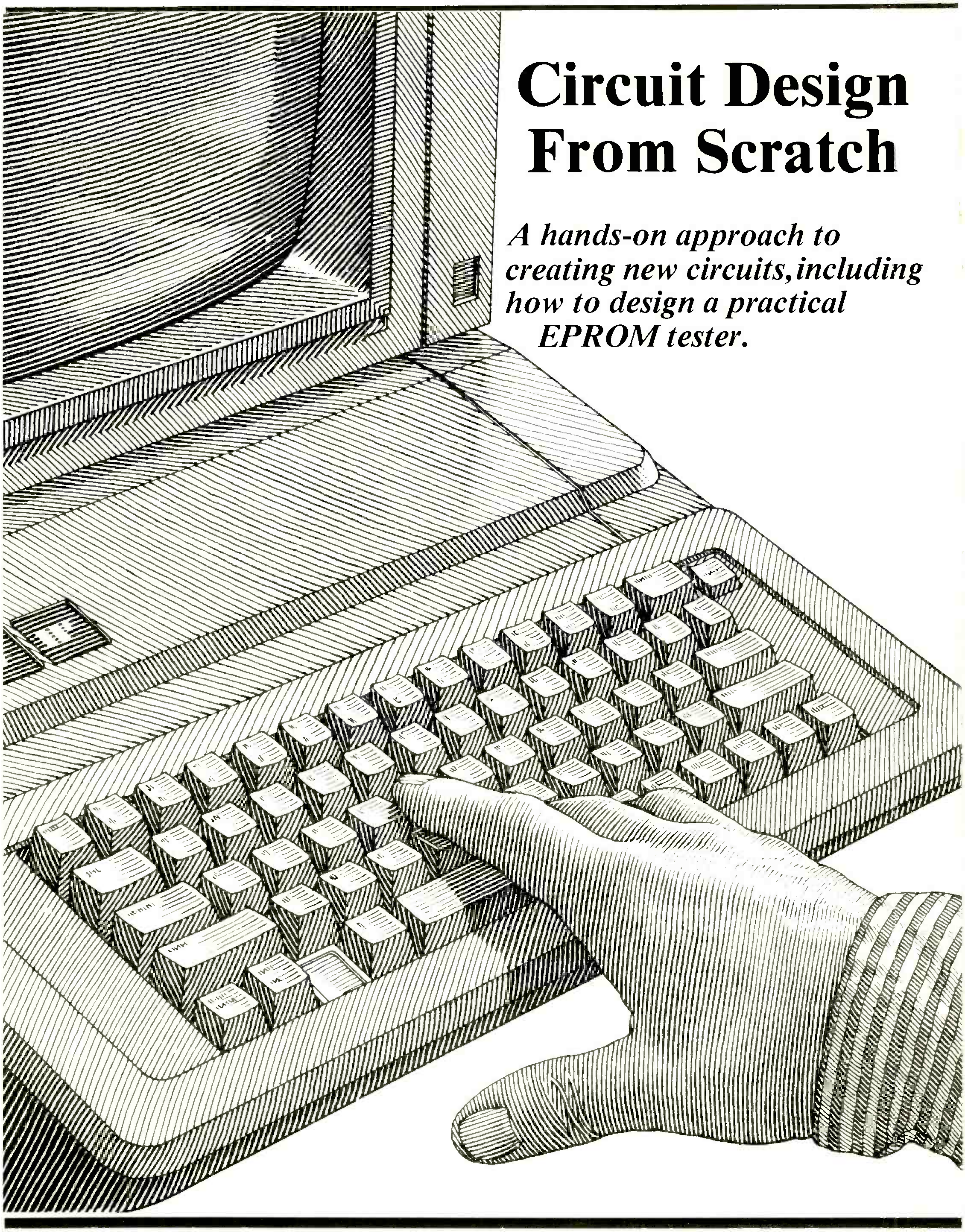
Project *No. 3 of a continuing series.*



Circuit Design From Scratch

A hands-on approach to creating new circuits, including how to design a practical EPROM tester.

By Jules H. Gilder

In the last two installments of this series, we dealt with projects for the home and car. This month, we're going to move into another area of growing interest to electronics enthusiasts. We're going to design a project that will be useful to computer buffs, as well as the advanced electronics hobbyist. The project is an EPROM tester that can be used to determine if an EPROM is completely erased prior to programming.

Most EPROM programmers, whether stand-alone or on boards that plug into computers, will check an EPROM to determine if it is blank before attempting to program it. This may be okay with you if you don't mind tying up your computer, but for practical purposes a portable, battery-powered tester is the better choice. Not only will if free up your computer to do other things, but a dedicated tester can work much faster than a computer can.

The specifications for this project are very simple. We want:

1) a battery-powered tester
2) simple pushbutton operation
3) an indicator to light when an EPROM being tested is not blank
4) the ability to test a device in about one second
5) a tester that will accommodate 2716 and 2732 EPROMS, which are the most popular types now in use.

The requirements don't appear to be too demanding in terms of design effort, but before we can actually design the tester, we must know something about how EPROMs work.

1) How EPROMs Work

Perhaps the best way to gain an understanding of how EPROMS work is to take a close look as the data sheets for the 2716 and 2732. (Data sheets are usually available for the asking from any store or company that sells the device.)

From the data sheet, the first thing we learn is that the chip is powered by a 5-volt source, which means our tester is going to have to contain a 5-volt power supply. Next, looking at the pin configurations (commonly called pinouts) for both the 2716 and 2732 (Fig. 1), we discover that the only difference between the two devices is in regard to pin 21. On the 2732, this pin is used to address the extra memory on the chip. So if we're going to accommodate both chips, we must design in a switch to toggle the connection to this pin.

Another bit of information we glean from the data sheets is that if we want to read data from the 2716, pin 21 must be at +5 volts. On the 2732, however, pin 21 is used as an address pin and, therefore, must be connected to whatever device we use to supply the address to the EPROM. Hence, a single-pole, double-throw (spdt) switch will be needed.

How To Read Memory. Before we go much further, it would be easier for us to understand how to design our tester if we understood how data is read from memory chips. Basically, a memory chip is an array of storage locations that can contain information for later use. In RAM (random-access memory), information in memory can be altered at will. (By the way, the term "random-access memory" is a misnomer, because even ROMs and EPROMs are random-access devices. Any byte in any of these devices can be directly accessed. Therefore, a more proper term for the RAM would be R/WM, for read/write memory).

In the ROM (read-only memory), once information is stored, it cannot be changed. In the EPROM (electrically programmable read-only memory), information can be stored in memory and when no longer needed be erased to ready it for entry of new information. The difference between RAM and EPROM memory is that a RAM can be erased and have new

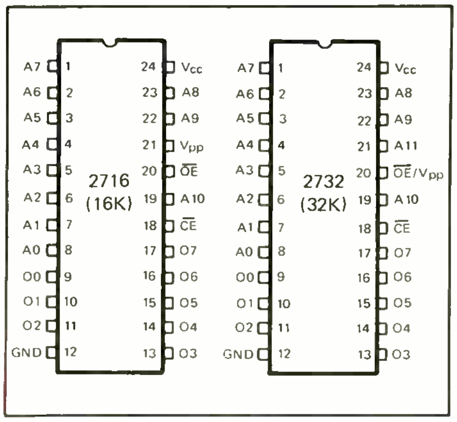


Fig. 1. Pin identification drawings for standard 16K and 32K EPROMs.

data entered into it immediately with electrical signals. RAMs also lose their data when power is removed from them and, consequently, are called *volatile* memory. EPROMs, on the other hand, must be erased by exposing them to intense ultraviolet light for about a half hour. Only then can they be rewritten and then only with a higher programming voltage. Also, any data programmed into EPROMs remains there even when power is removed, making these devices *nonvolatile* memory.

To be able to look at a particular cell or location in memory, you must tell the chip which memory location you wish to examine. Just as you must tell a friend your address if you want him to visit you, you must tell the memory chip the address of the location you wish to examine.

In the case of computer memory devices, the address of the cell being referenced is designated by a series of bits. There is a pin connection on the memory chip for each address bit (location). A *bit* represents the smallest piece of digital information and can have only one of two possible states —on or off, high or low, true or false, etc. Since a bit represents only two possible states, it's also referred to as a Binary digIT, from whence the contraction "bit" is derived. A bit can

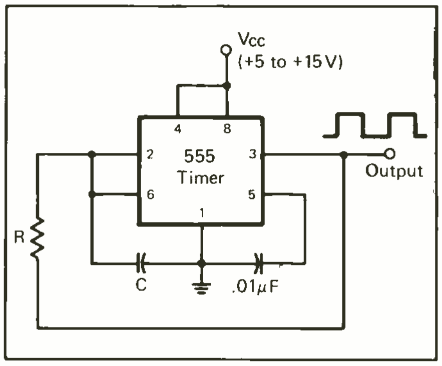


Fig. 2. Just three external components turn a 555 timer IC into an oscillator that services as a clock.

have a value of either a 0 or 1. Combining eight bits into a single entity yields a *byte*.

Numbers can be represented by a combination of bits. For example, two bits can represent any number between 0 and 3, while eight bits (one byte) can represent any number between 0 and 255 and 16 bits (two bytes) can represent any number between 0 and 65,535.

Memory devices can be organized in a variety of ways. For example, it's possible to have a 16K-bit device organized as 16,384 locations, each of which is one bit long, or as 2048 locations, each of which is eight bits (one-byte) long. The second approach is used to organize the memory in the 2716, while the 2732 is organized into 4096 by 8 bits.

Knowing the organization of the chips we're going to be testing, we can see that our tester will be required to access either 2K or 4K of memory. To access 2K, we'll have to know how many binary digits are required to represent 2K numercially. That number of digits is the same number of address lines we'll need. To represent 2048, the actual figure represented by the 2K shorthand commonly used in computing, we must have 11 address lines for the 2K by 8 EPROM. From Fig. 1, we see that the 2716 meets the requirement with the 11 address lines labeled A0 through A10. Similarly, the 2732 would need 12 address lines, which it has, labeled A0 through A11.

One more thing about memory devices and we should have their operation down pat. Whenever an address is placed on the address lines of a memory device, the contents of the cell at that address are automatically placed on the data (output) lines. Thus, as the address changes, the data on the output lines changes with it. Since the chip is organized as 2K by 8 bits, there must be eight data output lines. We can see in Fig. 1 that these lines are present in both the 2716 and 2732, labeled O0 through O7 (sometimes also represented by the labels D0 through D7).

To produce a device that tests EPROMs to determine whether or not they are blank, we must know what a blank EPROM looks like. This is easy: a blank EPROM has a 1 stored in each location on the device.

Now, if we can figure out a way to constantly change the address applied to the address lines and we check each of the eight output lines to make sure there are indeed 1s, we'll know if an EPROM being tested is blank or not. If even one bit in the EPROM is not a 1, then we don't have a blank device and it is either bad or must be erased.

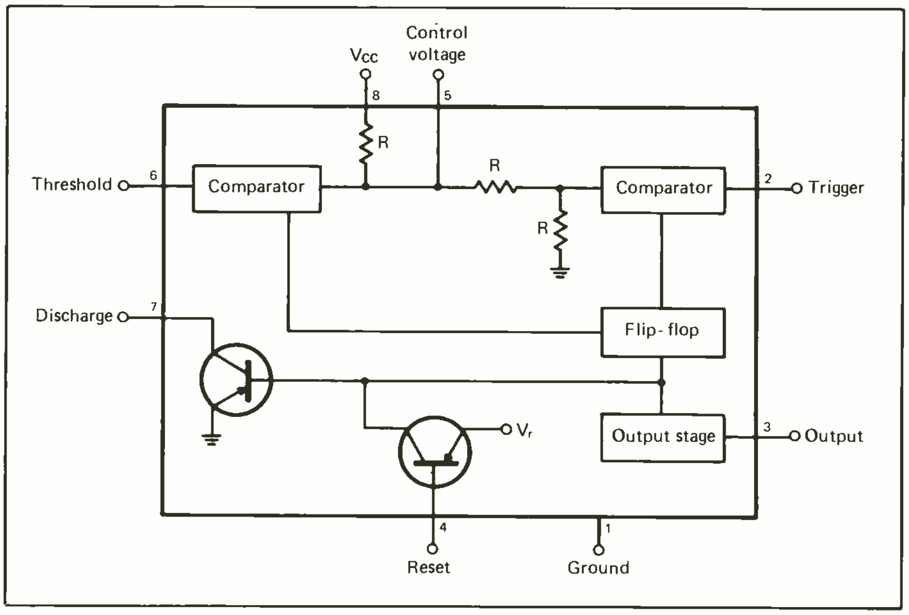
2) Design The Output Line Monitor

Since we're checking to determine if the output lines of an EPROM are all 1s and we want an indication if any of them is not, we should used a NAND gate. From earlier installments in this series, we know that if all inputs to a NAND gate are 1, the output is 0, and if any one or more inputs are 0, the output will be 1. Thus, we need an

Pinout Comparison Chart

	555	556	
	timer	timer 1	timer 2
Function	pin No.	pin No.	pin No.
ground	1	7	7
trigger	2	6	8
output	3	5	9
reset	4	4	10
control voltage	5	3	11
threshold	6	2	12
discharge	7	1	13
supply voltage	8	14	14

Fig. 3. This drawing illustrates the internal elements of the 555 timer.



8-input NAND gate, such as the 4068 CMOS device.

Connecting each of the data output pins of the EPROM under test to an input of the 4068, we can check the output of this gate to determine if and when it goes high. If the output does go high, we know that there's a bit in the EPROM that has been used.

To insure that the inputs to the gate stay high until a 0 is applied to one of the data lines, "pull-up" resistors are connected to each input and the positive side of the power supply. As long as no 0 is applied to a data line, these resistors will pull the data lines up to the power supply voltage. If a data line has a 0 on it, the side of the resistor connected to the NAND gate goes low, as does the input to the gate.

3) Address The EPROM

Now that we have a way of monitoring the output, we must have a means of applying addresses to the address lines and to change them. To do this, we'll use a *binary ripple counter,* which uses a series of "clock" pulses to advance the count for each pulse applied. Since this is a binary counter, its output is the binary equivalent of the number of pulses applied. If we connect the binary outputs to the address lines of the EPROM and then apply a clock signal that constantly increments the binary output of the ripple counter, we've found a way to change the address of the EPROMs as required.

Since we need 12 bits to address all memory in the 2732, we need a 12-stage binary ripple counter to supply the addresses for our EPROM. Going through a list of CMOS devices (or Don Lancaster's very handy *CMOS Cookbook* from Howard W. Sams), we find that the 4040 CMOS IC is exactly what we need.

The 4040 uses a clock input to produce a count, on 12 separate output lines, that represents the binary equivalent of the number of pulses applied. When the count exceeds 4096, the outputs return to 0 and counting starts over. Thus, with a continuous stream of pulses, the addresses in the EPROM will be continuously and cyclically addressed.

4) Design The Clock Oscillator

Using the 4068 and 4040, we've accomplished the bulk of the design work required for this project. Only a few things remain to be done. Firstly, we need a source of clock pulses to drive the binary counter. Secondly, we need some means of indicating when a bad or nonblank EPROM is encountered.

One of the easiest devices to use for a clock oscillator is the 555 timer IC. The circuit for this and many other useful projects can be found in my book *110 IC Timer Projects* (Hayden Book Co.). Using this IC and only one resistor and capacitor, it's possible to assemble an oscillator (Fig. 2) whose frequency is determined by the formula $F_0 = 0.772/RC$, where F_0 is in Hz if R is in megohms and C is in microfarads.

Figure 2 specifies a 0.01-μF capacitor connected to pin 5 of the IC. While this isn't really needed and the circuit will work without it, it does provide some noise immunity for the circuit, and good design practices dictate that it should be included.

In operation, the output of the Fig. 2 oscillator at pin 3 of the 555 is initially high and the timing capacitor starts charging exponentially at a time constant of RC. When the voltage across the capacitor reaches 0.67 of the value of the supply voltage (the upper threshold), a voltage comparator built into the 555 triggers an internal flip-flop (Fig. 3) and the potential on pin 3 drops close to 0 volt. The capacitor starts to discharge exponentially with a time constant of RC.

When the voltage across the capacitor reaches 0.33 of the value of the supply voltage (the lower threshold), a second internal voltage comparator triggers the 555's internal flip-flop and the output of the IC at pin 3 goes high once again.

The result of this back-and-forth charging and discharging is a series of pulses at pin 3 of the 555 timer IC.

The time in seconds it takes for the charge on the capacitor to reach the upper or lower threshold is 0.693RC (R is in megohms and C is in microfarads). A complete charge/discharge cycle is simply twice this time, or 1.386RC. The frequency of the pulses produced is the inverse of the time it takes for one complete charge/discharge cycle, or 0.722/RC.

With such a simple circuit and design formula, determining component values shouldn't be difficult. The first thing we must decide is at what frequency we want the clock to operate. We want to test an EPROM in less than a second. If a 2732 is used, we must check 4096 memory locations. To check this many locations in one second, we'd need a clock with a frequency of 4096 Hz. To check 4096 locations in 0.5 second, the clock frequency would have to be 8192 Hz. Though this would be perfectly acceptable, I preferred to use a 0.33-second test time, which requires a clock frequency of around 12,000 Hz (12 kHz).

Plugging 12 kHz into our formula and solving for RC, we obtain $0.722/12{,}000 = 60 \times 10^{-6}$. Any combination of resistance and capacitance that produces this time constant is acceptable, but a few details must be borne in mind. Resistor values should be kept to less than 1 megohm whenever possible so that a decent amount of current will flow, and capacitance should be kept to less than 1 μF if possible to avoid having to use high-leakage electrolytics.

Since 10,000-ohm resistors are used elsewhere in the project and there would be fewer components to keep track of, let's see if we can use a 10,000-ohm resistor here, too. The decision-making process in this case is as arbitrary as that. If $R = 10{,}000$ ohms, C would have to be 60.2 nF. Using a much more common 50-nF capacitor, however, we find that the frequency becomes 14.4 kHz, which is a little higher than the 12 kHz we wanted but still perfectly acceptable.

5) Design The Output Indicator

Although a high clock frequency will permit quick testing of EPROMs, if an EPROM has only one bad bit in it, you'll have only about 70 microseconds (1/14,000) to see it. This is obviously not going to be feasible. So, to overcome this, we're going to "stretch" the pulse to make it long enough in duration to light a LED and provide a visual indication.

Stretching of short-duration pulses is common in digital circuits, accomplished with a special circuit known as a *monostable multivibrator.* A pulse of any duration applied to the input of a monostable multivibrator generates an output pulse also of any duration. The only limitation is that the input pulse must be shorter in duration than the output pulse.

For our monostable multivibrator, we once again call on the versatile 555 timer. While the normal monostable operating mode of the 555 requires that the input pulse be negative (from a positive voltage momentarily to a negative or zero voltage and back), it's possible to use positive pulses, such as the one generated by the 4068 in our circuit, to trigger the monostable multivibrator (Fig. 4).

When the circuit is first activated, the trigger input to pin 6 (the output of the 4068) is low, timing capacitor *C* is discharged, and pin 2 is momentarily held low. This causes the internal flip-flop to switch state and forces the output at pin 3 to go high. With the output high, *C* charges to a value near the supply voltage and stays there.

When a trigger pulse of an amplitude greater than 0.33 of the supply voltage is applied to pin 6 (this is the pulse produced by the 4068 when a "bad" bit is encountered), the internal flip-flop is activated once again

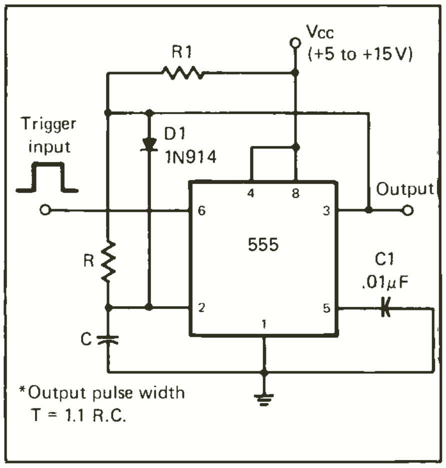


Fig. 4. Here the 555 timer is used as a monostable multivibrator.

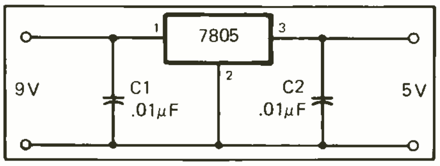


Fig. 5. A 7805 regulator outputs 5 volts from a 9-volt battery source.

and the output at pin 3 is forced low. This effectively grounds the point between *R1* and *R*, allowing *C* to start discharging through *R* to ground.

Discharge of *C* continues until the voltage on *C* and pin 2 of the IC reaches 0.33 of the supply voltage. When it does, the internal voltage comparator connected to pin 2 causes the flip-flop to switch again and *C* is quickly recharged via *R1* and the diode. The width of the pulse produced is 1.1RC.

We want to easily see even a single pulse out of the 4096 that will be used to test the EPROM. Therefore, if we stretch the pulse so that its width is as long as the time it takes to test the EPROM, even a single bad bit will cause the output to remain high all the time. Thus, we want a pulse width of roughly 0.33 second. You simply divide by the 1.1 in the formula to obtain RC = 0.3 second.

Arbitrarily selecting a 10-μF value for *C*, we find that *R* must have a value of 30,000 ohms. Although we could use 10 μF and 30,000 ohms, we'll opt for a standard value resistor as close to 30,000 ohms as we can get, which is 27,000 ohms. Using this value with the 10-μF capacitor, we obtain a pulse width of 0.297 second. This is close enough to what we originally wanted.

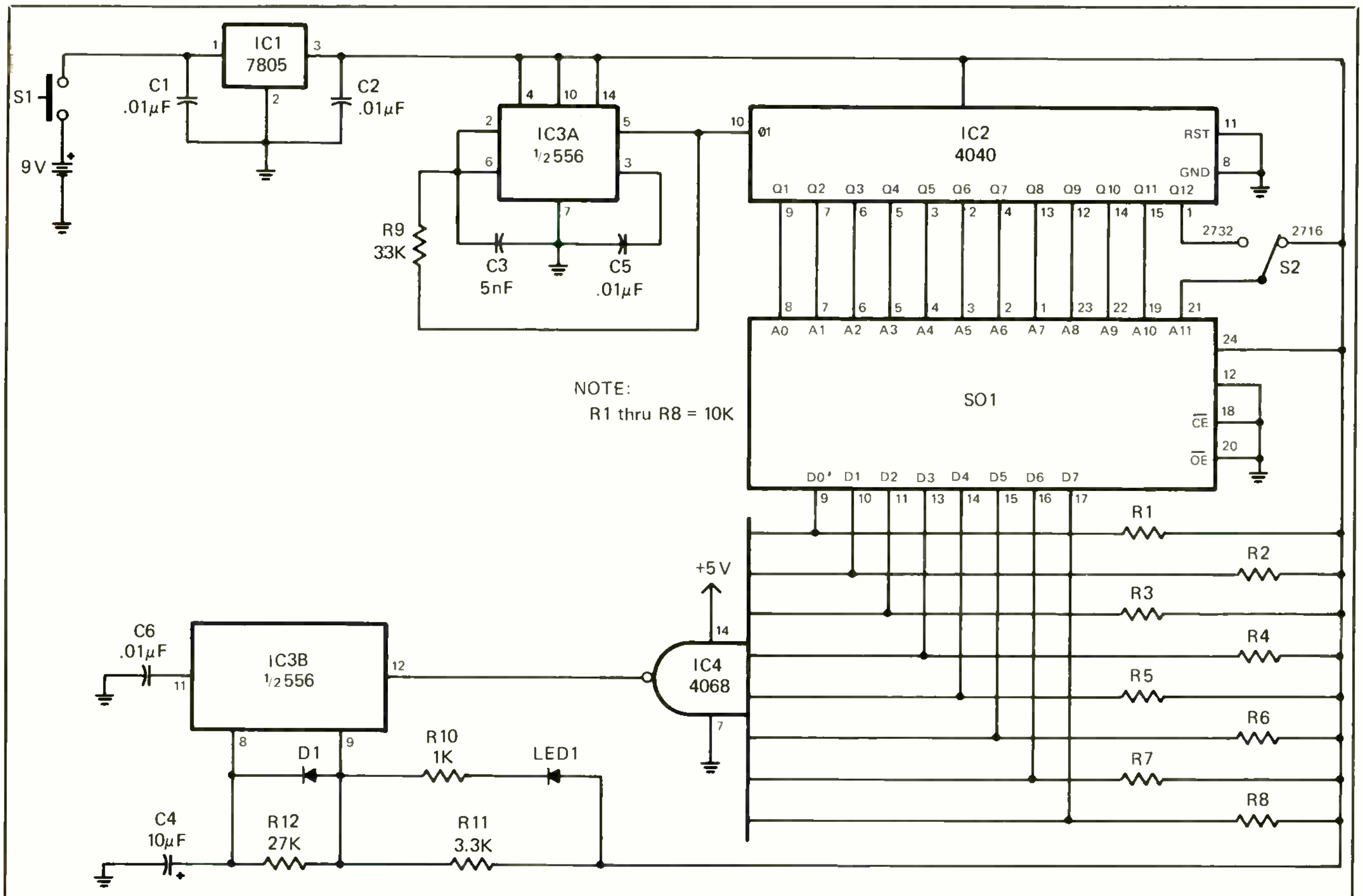
The positive-triggered monostable multivibrator produces a negative pulse. That is, its output is normally high and goes low only when the device is triggered but returns to a high state. The 555 timer can supply enough current to directly drive a LED. Therefore, if we connect a LED (and appropriate current-limiting resistor) between the positive-voltage power supply line and the output of the timer at pin 3, we'll have a visual indicator that tells us when an EPROM isn't blank.

When the monostable multivibrator isn't triggered, the voltage on both sides of the LED is essentially the same and no current flows. Triggering the monostable multivibrator causes the device's output to go low, essentially grounding the cathode side of the LED to complete the current path and turning on the LED. Once again, a capacitor is used to bypass pin 5 to ground to provide noise immunity for the circuit.

Since both the oscillator and the monostable multivibrator use 555 timers, we could build the circuit with two 555s. However, we can simplify the construction phase of the project by using a dual 555 timer device, known as the 556 dual timer. The timers in both devices are essentially identical, but since two timers are built into the 556's package, adjustments must be made in pin designations to assure a properly operating circuit. The pinouts for the two devices are enumerated in the Table.

6) Designing The Power Supply

Our design is nearly complete. All we have to do now is take care of one



PARTS LIST

Solid-state devices
D1—1N914 diode
IC1—7805 5-volt regulator
IC2—4040 CMOS 12-stage binary ripple counter
IC3—556 dual timer
IC4—4068 CMOS 8-input NAND gate
LED1—Light-emitting diode
Capacitors
C1,C2,C5,C6—0.01-μF disc
C3—5-nF disc
C4—10-μF, 15-volt electrolytic
Resistors (¼-watt, 10%)
R1 thru R9—10,000 ohms
R10—1000 ohms
R11—3300 ohms
R12—27,000 ohms
Miscellaneous
B1—9-volt transistor battery
S1—Normally-open, momentary-action spst pushbutton switch
S2—Spdt slide or toggle switch
SO1—24-pin ZIF (zero-insertion-force) socket for testing EPROMs

Sockets for ICs; suitable enclosure; printed-circuit board; terminal clip for B1; holder for B1; machine hardware; hookup wire; solder; etc.

Note: The following items are available from Redlig System, Inc., 2068 79 St., Brooklyn, NY 11214; printed-circuit board for $15; complete kit of parts (not including battery, case, EPROM, or ZIF socket) for $25.

Fig. 6. This is the overall schematic of the EPROM tester. EPROMs to be tested plug into SO1, which should be a zero-insertion-force socket. Tests are initiated by pressing S1; LED1 serves as a good/bad indicator.

minor detail—the power supply. While the 556 and CMOS ICs will operate over a broad range of voltages, the EPROMs themselves require a power supply capable of delivering a stable 5 volts. To make things simple, we'll power the whole thing from a 5-volt supply.

Since 5 volts isn't easily available from batteries, we'll use a common 9-volt battery. With a voltage-regulator IC, this is the simplest part of our project's design.

Our choice of regulator here is the 7805. This 5-volt regulator is supplied in a three-pin, transistor-like package. To use it, we merely connect the + terminal of the 9-volt battery to pin 1 and the − terminal to pin 2. The 9 volts going into the 7805 emerging as regulated 5 volts at pin 3. The common connection for the en-

(Continued on page 100)

Circuit Design From Scratch *(from page 73)*

tire circuit, including power supply is pin 2 of the 7805 (see Fig. 5).

Once again, good design practice dictates that bypass capacitors *C1* and *C2* be included in the circuit to minimize noise.

3) Putting It All Together

Now that we have all the elements needed to produce our EPROM tester, let's put them all together into one circuit, as shown in Fig. 6. Note in this schematic diagram that the circuit will be controlled by normally-open pushbutton switch *S1*. Pressing the button on *S1* supplies power to the whole circuit for as long as the switch is held closed.

Because *S1* controls power to all the circuits, instead of just the EPROM, when its button is first pressed, the monostable multivibrator may be triggered once or twice before all voltages stablize at the steady-state levels. Hence, you may see *LED1* flash once or twice when the button is first pressed, even if the EPROM being tested is good. After those first brief flashes, however, if the EPROM is erased and is in good condition, the LED should remain off. Of course, if even one bit in the EPROM is programmed with data, the LED will remain on for as long as *S1* is held closed.

To switch the tester from the 2716 to the 2732 mode, spdt switch *S2* is used. Since power for this tester is drawn only when an EPROM is being tested, battery life is very long. To test battery condition, just press *S1* with no EPROM in the circuit. If the LED is brightly lit, the battery is okay; when the LED starts to dim, change the battery.

To make construction of this project easy, a printed-circuit board is available from the source given in the Parts List.

In Closing

In the first two installments in this series, we led you through the various phases of the design procedure by providing all the information you needed to make your own project. This time around we've introduced you to data sheets and how they are used to help you in your endeavors. In upcoming installments, we'll be taking you deeper into design procedures and practices, building your confidence in designing your own projects from scratch. **ME**