THE DRAWING BOARD

Finishing up the keyboard encoder

ROBERT GROSSBLATT

FOR THE LAST TWO MONTHS WE'VE BEEN designing a keyboard encoder. As we left off last month, the encoder was neither easy-to-use nor useful. This month we're going to change all that. When a piece of equipment is designed and put on the market, one of the phrases that gets bandied about is "human engineering." In actual fact, that is more of an advertising phrase than a technical one and the more competitive the market, the more often you hear it. All it means is that the product was designed to make it easy to use. That, I suppose, is to set it apart from all those products that were designed to be *hard* to use. Advertisements for technical equipment with an esoteric market will stress accuracy and reliability; consumer-product advertising usually deals with color, shape, and sex appeal. Anyone who has ever tried to get detailed technical information about a consumer product knows exactly what I'm talking about.

As we finish up the design of our keyboard encoder, we're going to throw in a bit of "human engineering" in the form of—are you ready for this?—audio keyboard-feedback. How's that for a bit of psuedo-technical jargon? In any event, it's a nice convenience feature and since we get it for practically nothing we may as well throw it in. Our scan oscillator, IC2 (a 555), was set to run at a rate of about two kHz or so. That frequency was chosen for two reasons. First of all, it's highly unlikely that anyone would be able to enter data fast enough to "beat the clock" so to speak, and secondly, two kHz is a nice frequency for an audible "key-pressed" indicator. The trick is to use the oscillator without interfering with the operation of the encoder. It's clear that we will need two things: a switch to connect the 555 to the speaker, and some way of throwing the switch every time a keypad switch is closed. Let's automatically rule out double-pole double-throw switches at the keyboard and see if we can take care of business in a more elegant fashion. For us, that means using inexpensive components, and whatever unused silicon there is left in the circuit as it is so far.

Designing the beeper

Figure 1 shows a possible solution to the problem. It's not the only way to get the job done and, if you can dream up something slicker, so much the better.

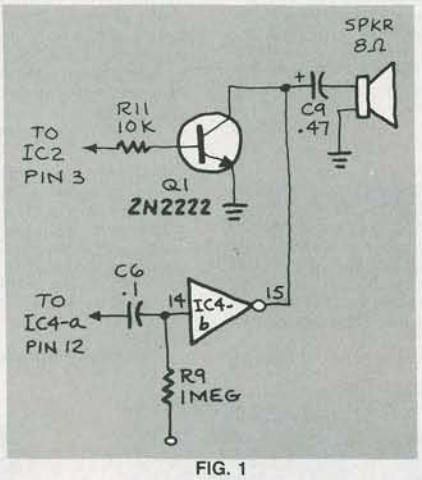

FIG. 1

Transistor Q1 is used as a switch between the 555 and the speaker, and the action of that switch is controlled indirectly by the "any-key-pressed" output of the encoder. I say "indirectly" because there's some silicon on the line. As long as we hold the collector of Q1 low the transistor is in cutoff and there's no audio. When a keypad switch is closed, the "any-key-pressed" line goes high, and that turns on the transistor and sounds the beep. As long as the keypad switch remains closed the beep will sound. That works perfectly, but we can make it slicker and use up some spare silicon at the same time.

We still have an unused inverter in IC4, so we'll build ourselves a half monostable with a period of about 100 milliseconds (0.1 second). We want it to output a positive-going pulse when a keypad entry is made; that means that using the "any-key-pressed" signal isn't possible since it's an active-high. As you now know, if you use an inverter to build a half monostable, a positive pulse at the output is produced by a positive-to-negative transition at the input. As it happens, we have such a signal available to us because we inverted the "any-key-pressed" signal to disable the 4518 BCD counter. All we have to do is pick up the trigger from the output of IC4-a, route it through our new edge-detector, and we'll get a clean beep lasting a tenth of a second whenever we enter a digit. Capacitor C9 keeps the eight-ohm impedance of the speaker from loading down the output of IC4-b, and R11 isolates that section from the rest of the circuit.

A data bus

Let's now turn our attention to the data bus—or rather the lack of one. So far, our encoder displays entered digits, but the data isn't available to us for use. That's a serious problem—since we want more out of this exercise in circuit design than an elaborate do-nothing box, a real data-bus is a must. At the moment, data from the keyboard goes directly to the display circuitry. What we have to do is change that so that the display shows whatever is on the data bus. That means we have to put some storage space between the encoder and the display. The easiest way to

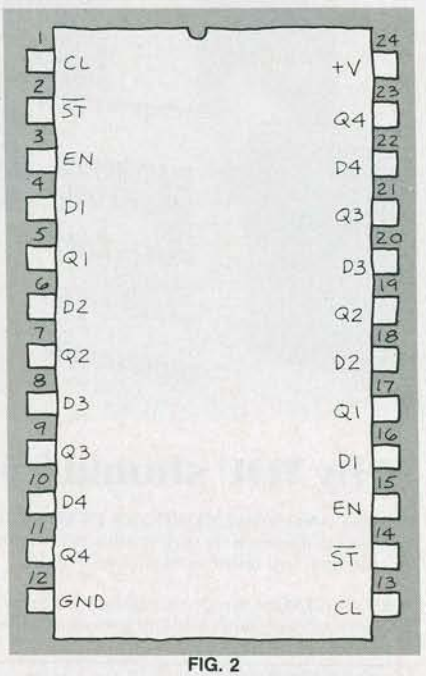

FIG. 2

do that is to put some hold-and-follow latches there and use them to remember the entered numbers.

Figure 2 shows the pinout of exactly what we need—a 4508 hold-and-follow latch that has the added bonus of Tri-State outputs. It's dual 4-bit latch, and each half of the IC can be used independently of the other. The EN (enable) pin controls the outputs, and bringing it high puts them into a high-impedance state without having any affect on the operation of the latch itself. The ST (store) pin controls the loading of the latch. If it's held high, the latch is transparent, but taking it low will

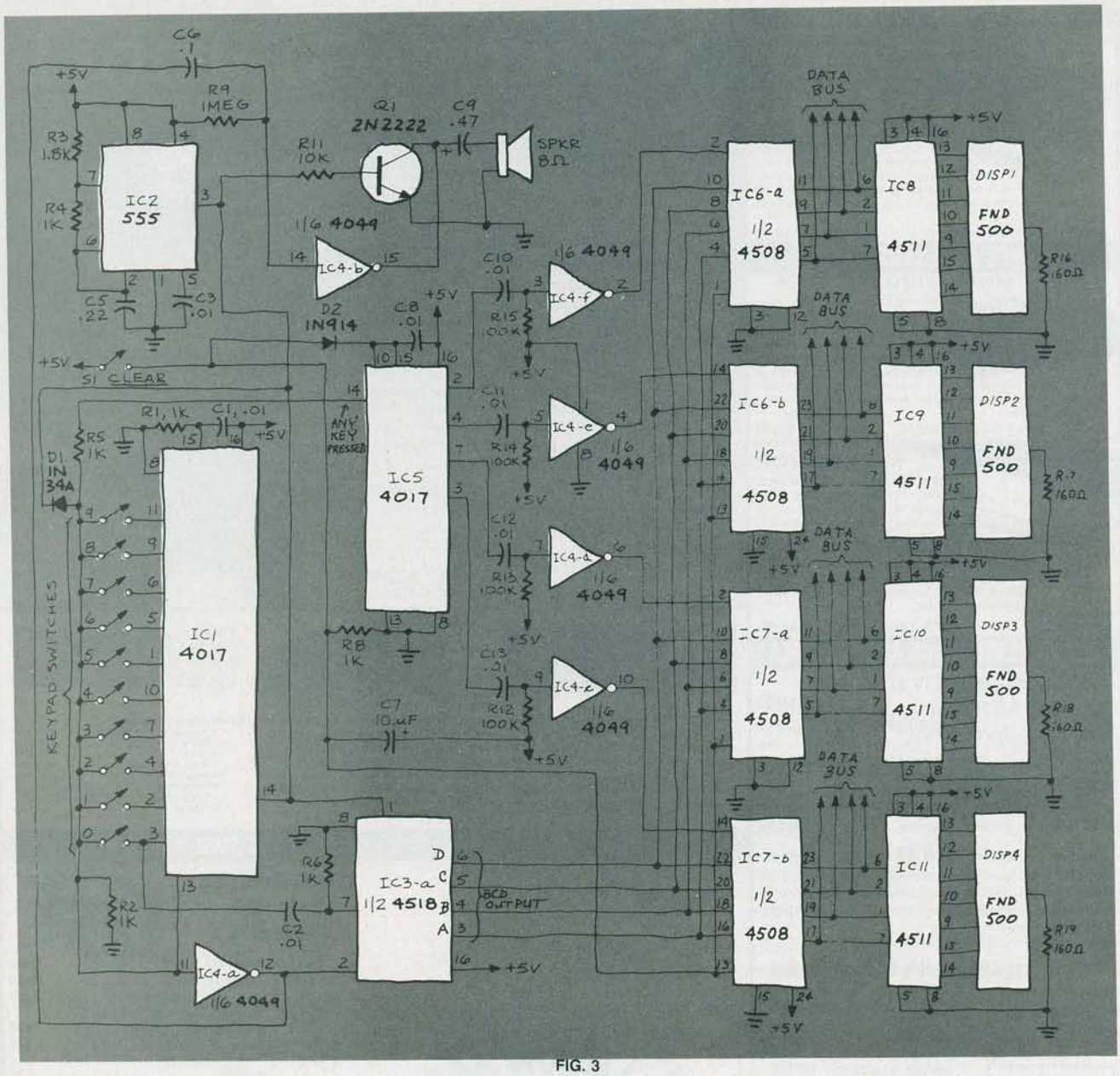

FIG. 3

close the latch and store whatever was at the inputs at that time.

On the face of it, there would seem to be a major problem in using a 4508 with the circuit we've designed so far, because the latch in the 4511 is enabled high and the 4508's latch is enabled low. A little thinking can solve that problem when we remember that the half monostables we're using to enable the latches are really edge—not level—detectors. As the circuit stands now, we're using the leading edge of the 4017 outputs to trigger the edge detectors. Well, what goes up must come down, so we can use the trailing edge instead and reconfigure our edge detectors to provide a pulse of the correct polarity. What that means in terms of circuit design is that instead of latching the first thing on the bus when a digit selector output goes high, we're grabbing the last thing on it before if goes low. Since the 4518 BCD counter is disabled while the keypad switch is thrown, the data remains unchanged.

Figure 3 is the complete schematic of our keyboard encoder. We've kept it down to four digits because the principle is the same for any amount of digits. Note that the 4508's have been put between the encoder and the display, and a true Tri-State bus now exists to display whatever data is put on it. The half monostables have been reconfigured and the digit selector, IC5, has been reconnected so that the first digit is now selected by pin 3, the zero output.

A "clear" function

The last piece of business we have to take care of to satisfy our original design criteria is to create a keyboard "clear." Now that the latches are in place, that becomes a simple problem. The CL pins of the 4508's are tied together and held at ground by R8. Throwing the CLEAR switch (S1) on the keyboard clears the latches and they output zeros. The same "clear" signal is sent on to IC5, the digit selector, to reset that to zero as well. Diode D1 prevents the normal resetting of the digit selector from clearing the latches. Capacitor C7 should be familiar by now. It provides a power-on reset pulse for the latches so they always power up with zeros. Resistor R9 forces a high on the collector of Q1 so that hitting the CLEAR switch will cause the beep to sound.

The EN pins of the 4508's have been tied to ground. If you're going to use the Tri-State feature of the data bus, some sort of selector logic will be needed to control access to the bus. It's a "one-and-only-one" sort of situation, and you should be able to do it with a few simple gates. Which ones you use will depend on the particular application, but the basic

continued on page 160

continued from page 144

problem is the same. A good exercise in design is to work out a keyboard-switchable selector without using any mechanical rotary switches. Bear in mind the fact that there is still some unused silicon in the circuit—namely, the other half of the 4518. If you work out a neat way to solve the problem, let me know and I'll pass it along.

If your particular application for the encoder requires more than four digits, it would be a good idea to consider multiplexing the display. Ten digits and ten drivers mean more than seventy connections and that's a bit unreasonable, to say nothing of power hungry. Using only one driver and multiplexing the display saves a lot of trouble when you get beyond four digits. If all you're interested in is cutting back on power consumption (and don't mind adding a bit more circuitry), you have another option—you can just multiplex the LED's themselves. That's a much simpler problem, and only calls for a clock and a multiline selector that sequentially scans the digits. You can use the cathode connections of the digits but a much slicker way is to use the blanking pins of the drivers. There are lots of ways of multiplexing and I'd be interested in seeing what you can come up with. If you're thinking of trying it, remember that the 555 can deliver up to 200 milliamps and is running at a speed that's perfectly suitable for multiplexing.

Our keyboard encoder meets the criteria we set for it, and although it's more complex than using an IC that's designed especially for the purpose, it's cheaper and much more flexible. But it's certainly not the last word in keyboard encoders. There are two important rules to remember when you're doing any circuit design. The first is not to be afraid to experiment. If you find a way to do the job that looks weird on the surface and uses components in ways they were never used before—great. It's not called "weird," it's called "innovation." If your method works, don't be afraid to use it. Fresh minds bring fresh approaches, and something that seems perfectly obvious to you may never have occurred to anyone else.

Don't be afraid of blowing up your circuit. With prices as low as fifteen cents a gate, you can't do much financial damage. And when you've worked out a really slick way around a problem and figure that everything's taken care of, remember the second rule: There's always a better way. If you can do it with five packages, somebody else can do it with four.

As a matter of fact, if you think of a better way to build or improve our keyboard encoder, let me know and we'll let everybody else know. **R-E**