THE DRAWING BOARD

Adding a digit select to a BCD encoder

ROBERT GROSSBLATT

IF YOU BREADBOARDED THE BCD ENCODER we designed last month you found (we hope) that it was a trouble free, reliable circuit. However, its use was somewhat limited because the encoded data wouldn't latch and only one digit at a time could be placed on the bus. This month we're going to add additional logic to the circuit so that we can display and latch up to 10 digits at a time. We'll stick to the set of design criteria we listed last month and we'll use the same sort of step-by-step approach to add the new sections to our design. The choice of components will still be weighted in favor of those that are easily available and reliable, and that put the smallest possible dent in your wallet.

The digit select

We want the digit select to sequentially address one thing after another. You could use some sort of shift-register approach for that, but the clocking can be a problem and the package count can get pretty heavy. There's a neater way to solve the problem that also happens to work out better in the long run. Not only can we solve the addressing problem with only two IC's, but expanding the circuit to handle ten digits will only call for one additional IC.

Instead of the shift-register approach, we'll create an input data bus and design circuitry that will enable one digit at a time. We take the "any key pressed" output of our BCD encoder and use that to clock a 4017 one-of-ten decoder. That means that each time we close one of the keyboard switches, we put a corresponding nybble (4 bits) on the data bus and the 4017 puts a high on one of its output pins. A new digit entry will result in a new nybble on the bus and a new high from the 4017. That continues for up to ten entries (sequentially). Figure 1 shows how you would connect the 4017 to handle four digits with the encoder circuit we started last month. Although the circuit will handle ten digits we'll limit our illustration to four. (The principle is the same and it makes the circuit easier to understand.)

Capacitor C8 serves the same purpose that C1 did in last month's circuit. It gives us a reset to zero at power-up and makes sure that everything starts out at the beginning. With the 4017 set up as shown in Fig. 1, it will reset after four low-to-high transitions of the "any key pressed" output. If you want it to handle more than four digits all you have to do is connect the RESET pin, (pin 15), to the numbered output that is one past the number of digits you want to deal with. If you want to go all the way and encode ten digits, ground pin 15 through a 1K resistor.

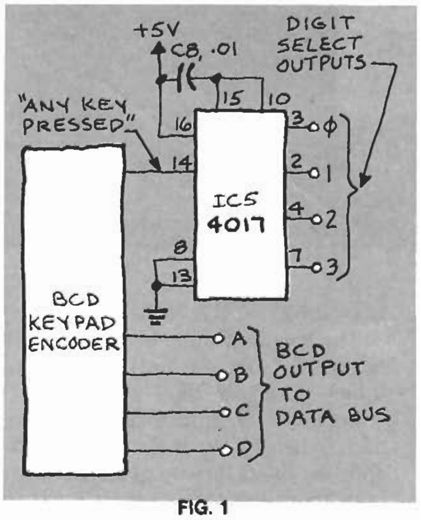

FIG. 1

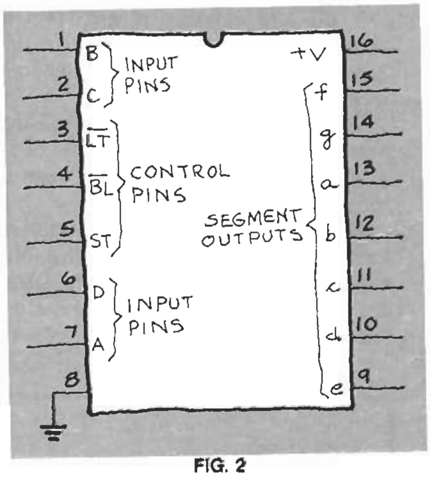

FIG. 2

Figure 2 shows the pinouts for a 4511—the decoder we will use to drive our display. The LAMP TEST and BLANKING control pins (pins 3 and 4) are active low and should be kept high for normal operation. The STORE input controls the internal latch and is active high. If it's held low, the 4511 will decode whatever BCD data is presented to its inputs. If it's made high it will latch and display the last nybble on the bus at the moment it went high. Any invalid BCD code will blank the display.

The obvious step in creating our data bus is to connect the A, B, C, and D inputs of the 4511's together and tie them to the appropriate BCD outputs of the encoder. Our four digit-select outputs would be connected to the STORE pins of the respective 4511's and we would be in business. Unfortunately that would fail miserably and a moment's reflection will show you why. The outputs of the encoder are constantly scanning from zero to nine at the clock rate, so the 4511's that weren't selected would display constant eights—and not even real eights at that. The selected digit would display the keyed number but would go to eights as soon as the digit selector shifted to the next digit.

What we need is a way of delivering a brief pulse to the STORE pin to open the latch just long enough to enter the nybble at the selected 4511. Now, pulse generators are a dime a dozen, and perfectly workable ones can be built with 555's and other IC's. In real down-and-dirty situations, you can get by with just a capacitor and a resistor, but the discharge time of the capacitor creates a very sloppy slope at the trailing edge of the waveform. The easiest way to get the job done and still be true to our design criteria is to use a half monostable.

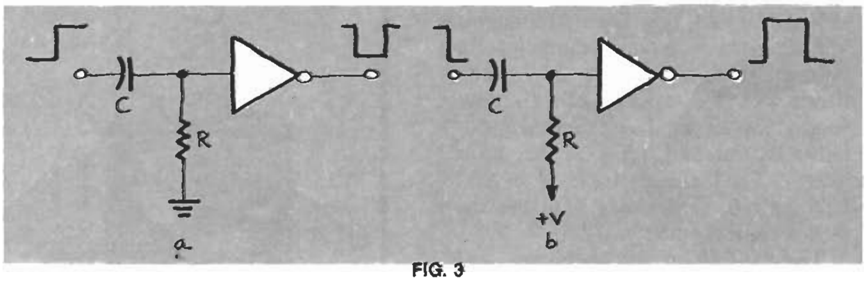

FIG. 3

Fig. 3 shows the basic configuration of half monostables. In actual fact they

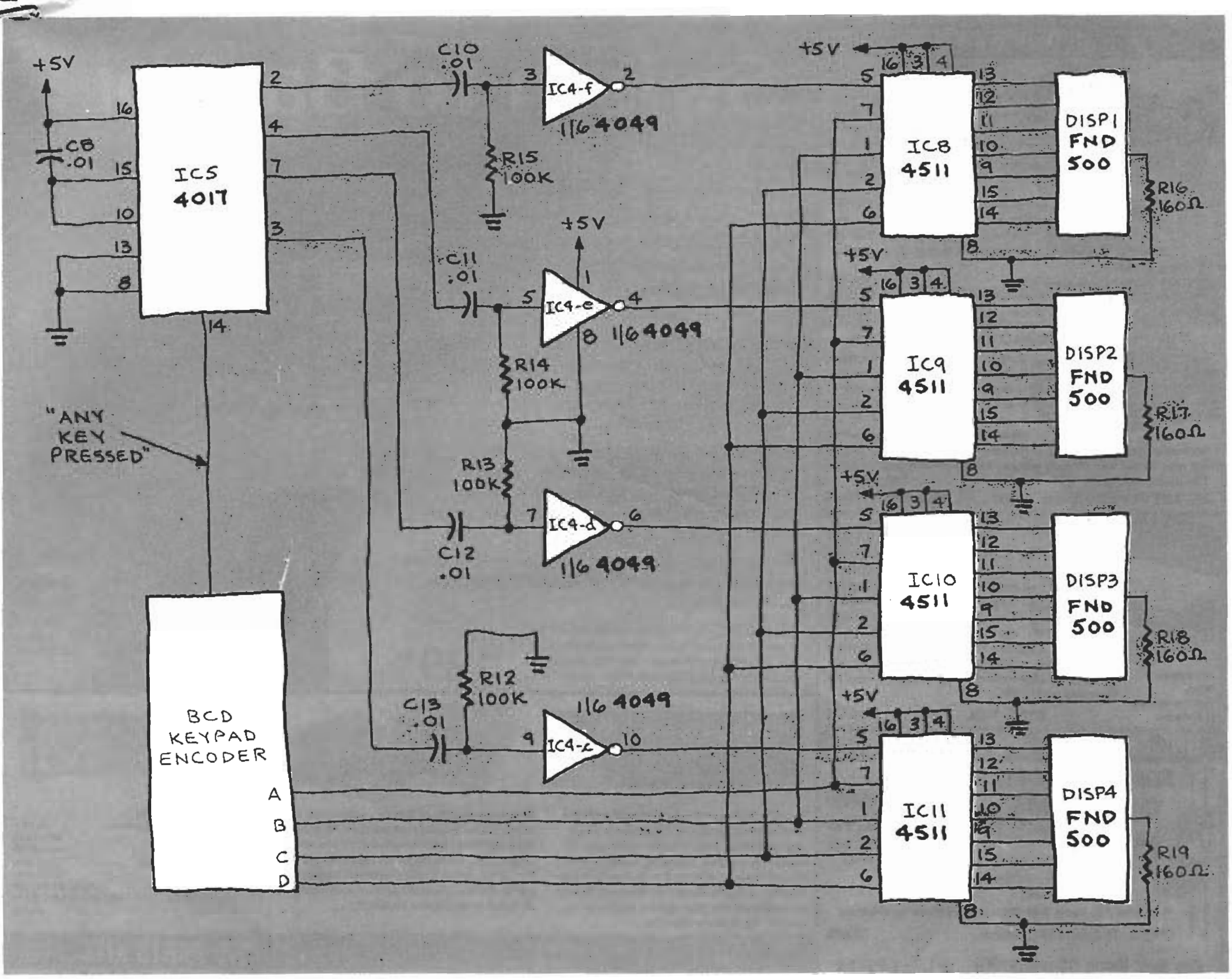

FIG. 4

should really be called edge detectors because they respond to either the leading or trailing edge of a logic level transition. With resistor R connected to V+, the gate input is held high. When the input goes low it forces the gate to change state for a period of time determined by the values of the resistor and the capacitor. The duration of the output pulse depends on the R–C value and the slope of the waveform depends on the transition time of the gate. A 4049 is a good choice here because it has enough internal gain all by itself to clean up the sloppy edge of the input waveform. The inherent hysteresis of a Schmitt trigger also makes it a good candidate for a half monostable. If you build those circuits with non-inverting gates, the same analysis applies but, of course, the output pulses will be in the opposite direction. The "big if" with these half monostables is that the input pulse has to be longer than the desired output pulse.

That is really self evident—a moment's thought will tell you that you have to give the capacitor enough time to charge up. If that condition isn't met the circuit won't blow up, but the output pulse will be the same width as the input pulse. In our case that's not a problem because the outputs of the 4017 latch high when they're decoded. All we have to do is make sure the output pulse-width of the half monostable is less than the fastest speed we can enter data from the keyboard. One millisecond should be fast enough for anybody—even for the world's fastest supermarket cashier.

In Fig. 4 we've completed the digit selector and display and connected it to the encoder we built last month. When we turn the power on, the 4017 is reset to zero and pin 3 goes high. Since the negative-to-positive transition is what triggers the half monostable, the first digit we enter will be on the negative-to-positive transition of output No. 1 (pin 2) of the 4017. That's why the schematic shows the zero output (pin 3) of the 4017 connected to the last digit.

In any event, as soon as power is applied, the circuit prepares itself to enter the first digit. When we close one of the keyboard switches, a BCD nybble is held on the data bus and the 4017 goes high on output No. 1. That triggers the half monostable and opens the 4511's latch just long enough to enter the nybble and then closes it again. The result is that the selected number appears in the display and stays there. When a second keyboard switch is closed, the 4017 enables the latch in the second 4511 and the number appears in the second display. That whole procedure continues until the fourth digit is entered and the 4017 resets. From that point on, the entered digits will write over the previously entered ones. The 4511 is designed to be used with common-cathode displays; we used Fairchild FND-500's. Only one current-limiting resistor was used for each numeral because I don't mind the slight differences in brightness that shows up when different numbers are displayed. If you want the numbers to be all of equal intensity, connect the cathodes of the display directly to ground and get yourself a huge supply of low-value resistors because you've got to put one on the line between each 4511 output and LED anode. Keep in mind that the 4511 can only supply about 25 milliamps per segment, so choose the resistor value accordingly. You can play with this circuit for a while but it will soon be painfully obvious that it leaves a bit to be desired.

Since we don't have any access to the nybble in the internal latch of the 4511 and decoding the segment outputs is, to put it mildly, a strange way to go about

continued on page 99

DRAWING BOARD

continued from page 84

things, it's clear our circuit is far from being complete. What we need is an output bus as well as the input bus used by the keyboard encoder. Another shortcoming is that we don't have any easy way to clear an entry other than entering zeros. We can enter numbers from a keyboard and have them show up in a display and even though we can expand to ten digits, more circuitry is needed before the encoder can be put to any practical use.

Next month we'll add all the bells and whistles to our encoder. We'll add a Tristate data bus, an audio indication of keyboard entry, and the ability to clear the display from the keyboard. **R-E**