THE DRAWING BOARD

Designing a keyboard encoder

ROBERT GROSSBLATT

BEING ABLE TO TURN A CONCEPT FROM A vague idea into a working piece of hardware is one of the greatest joys of electronics. The increased complexity of modern electronics can give people second thoughts about deciding whether to build or to buy. The same dilemma had to be faced, though, when transistors replaced vacuum tubes—and you know what happened then.

This department is dedicated to those of you who like to get your hands dirty—the same people who can't resist the temptation to take the cover off a piece of new equipment to "see what's inside." We'll be discussing original design techniques, shortcuts, pitfalls, and breakthroughs. There will be an equal balance of theory and application so you'll learn not only what to do, but why you're doing it and how to adapt the ideas presented here for your own purposes.

And, if you know a better way to do something, or catch me in a mistake, let me know. I'm as fallible as any of you.

Design criteria

Complex circuits are made up of lots of discrete sections. That seemingly trivial statement is one of the *keys* to original design. You can bet your new pair of white tennis shoes that the 12-page schematic that came with the last kit you built started life as doodlings on someone's 8½ × 11 pad of paper. The first part of any original design is a list of design criteria—in other words, what you want the circuit to do and how you want to make it do it. Let's suppose we want to design a circuit that will allow us to enter data from a keypad. Our design criteria would be something like this:

1. Up to 10 digits will be entered from a keypad.
2. Data will be placed on a Tri-state bus.
3. Any data on the bus will be displayed on LED readouts.
4. Each keypad entry will generate an audible signal.
5. The keypad will have two-key rollover (two keys may be pressed "simultaneously," but the state of the second will not be read until the first has been released).
6. The circuit will have power-on reset.
7. The data bus will be able to be reset from the keyboard.
8. Operation will be glitch-free.
9. The entire circuit will use standard parts that are inexpensive and easily available in single-unit quantities.

Once the criteria have been set, the next thing we need is a block diagram that breaks the circuit into separate sections that can be designed individually. Figure 1 is a block diagram of the circuit that was described by our criteria. In order to meet criterion number 8, we'll use CMOS technology. Not only will its noise-immunity make the circuit as glitch free as possible, but the low power-requirements of that logic family will help a lot if you decide to run the circuit from batteries.

A keypad encoder

Since the BCD (*B*inary *C*oded *D*ecimal) encoder is obviously the heart of the

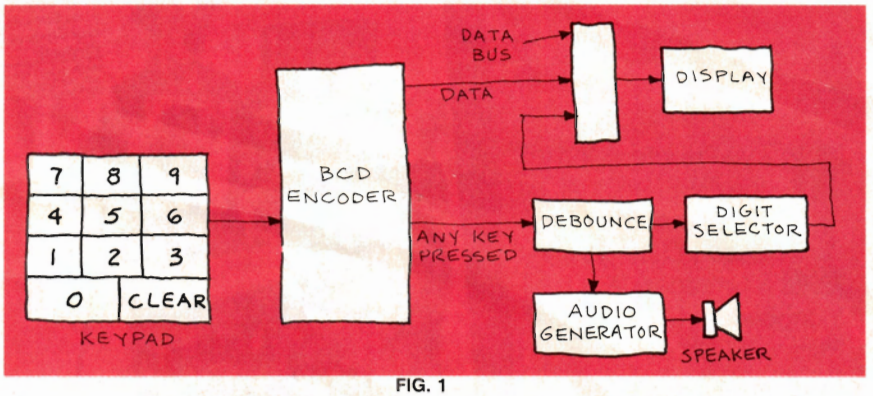

FIG. 1

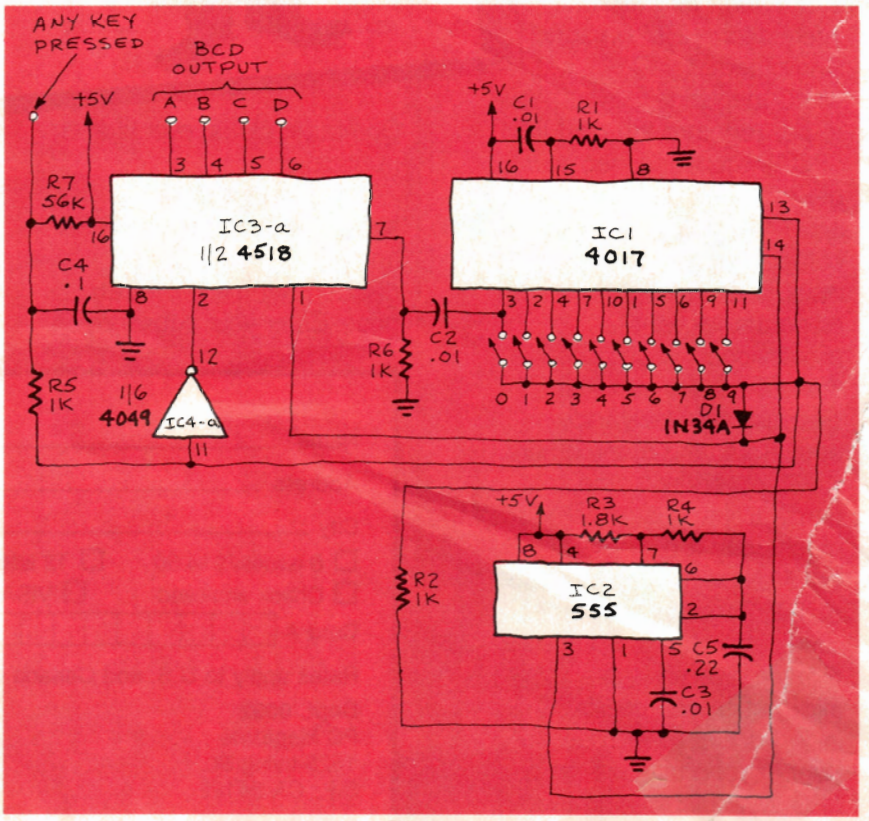

FIG. 2

circuit, we'll start with that. There are lots of different schemes that can be used to design such an encoder, but they all usually decode the output of a running counter. The approach we'll take is shown in Fig. 2. A 555 is set up as a signal source running at about 2 kHz. That clock is used to drive both a 4017 decade counter and a 4518 binary counter. The 4017 gives us decoded one-of-ten outputs at the same time that the 4518 counts up in BCD. The trick to getting the circuit to operate properly is to make sure that both IC's run in sync. We'll do that by using the reset pins on each IC.

The reset pin on the 4017 has to be held low for normal counting but we want to make sure that the count starts with zero when the circuit is first turned on. Therefore, resistor R1 holds the reset pin low so the IC can count, while C1 provides a positive pulse at power-up to reset the counter to zero. We can handle the 4518 in a similar fashion. The reset pin (pin 7) is held low by R6 and the count is started at zero by C2. When the "zero" output on the 4017 goes high it sends a positive pulse through C2 and on to the reset pin of the 4518. That does two things for us. Not only does it make sure that the two IC's are in sync at power-up, but it keeps them in sync; it does that by resetting the 4518 to zero every time the 4017's count hits zero.

Resistor R2 keeps the 4017 enabled as long as none of the keypad switches are closed. When a switch is closed, the IC is disabled and the count stops. The high signal generated by the key closure is also inverted by IC4-a and disables the 4518. At the same time, diode D1 conducts, making sure that the clock stops. The result of all that is that when you close a numbered switch on the keypad, the corresponding BCD code appears at the outputs of the 4518.

The circuit has two-key rollover because scanning starts up again at the instant that the keypad switch is released. When the second switch-closure is reached, scanning stops once again and the 4518 outputs the BCD equivalent of the second number. We also get an "any key pressed" output that is debounced by C4 and R7.

The circuit uses the 4017 as a one-of-ten decoder, and that means that all the keypad switches have one side tied together. It's just as easy to build a BCD encoder that uses switches in a row-and-column matrix, and there are even keyboard-encoder IC's such as the 74C922 that have the clock, counter, and all the rest of the circuitry on a single chip to take care of all the housekeeping chores that we performed with resistors and capacitors.

The reason for our design is given by criterion number 9. The IC's we're using are common ones that are available from almost any supplier (see the ads at the back of this magazine). They cost about a dollar apiece. Keyboard-encoder ICs aren't as readily available, and cost at least five times as much. If none of that impresses you, there's an even more basic reason.

There isn't very much you can learn by plugging a black box into the wall and turning it on so it can do whatever it was designed to do. And the more specialized an IC, the more it begins to resemble a black box. Thus, one of the best reasons I can think of for designing a circuit from the ground up is to understand how it works and to be able to correct any problems that crop up. Also, the more basic the components in the circuit, the more flexibility you have in design. Although that isn't readily apparent yet in our circuit, as we add the other sections shown in the block diagram, we'll see how really flexible the circuit is. And unless you have an understanding of how basic circuits work, it's impossible to understand more complicated ones. You can't learn to program a computer by buying game cartridges.

Now that we have our keyboard encoder operating and putting out a four-bit code, we can concentrate on the rest of the circuit. Next time we'll continue our project by adding the digit selector and display so we can encode up to ten digits sequentially. **R-E**