DRAWING BOARD

ROBERT GROSSBLATT

Automatic data sequencing

IN OUR LAST DISCUSSION OF MEMORIES, we mapped out the design criteria for our demonstration circuit. We've taken care of keyboard data entry with the binary keyboard encoder that we've already built. This time, we'll see what must be added to that circuit to make it do something useful. After all, what good is the encoder without having some way of storing and/or manipulating its output data.

Since one of the design criteria is automatic sequencing of the address and data, we'll need something in the circuit that automatically does one thing after another. The problem of data sequencing was addressed when we built the binary keyboard encoder, which was designed to continuously scan a series of switches in search of a depressed key.

Automatic sequencing scheme

Since we'll be sequencing both address and data, we also need some way to let the circuit know which is which. The easiest way to keep track of what's stored where is to store the low and high order halves of the address, plus the data separately. Given all that, let's take a look at Fig. 1, a tentative solution to the problem.

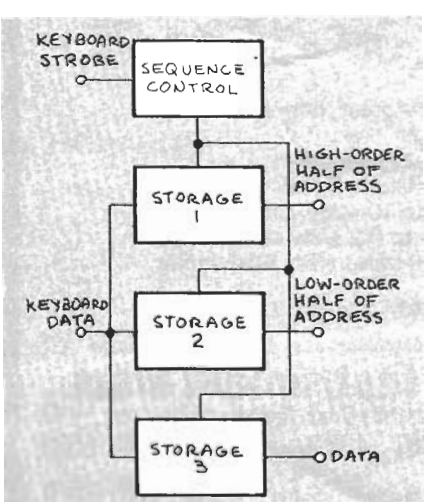

FIG. 1

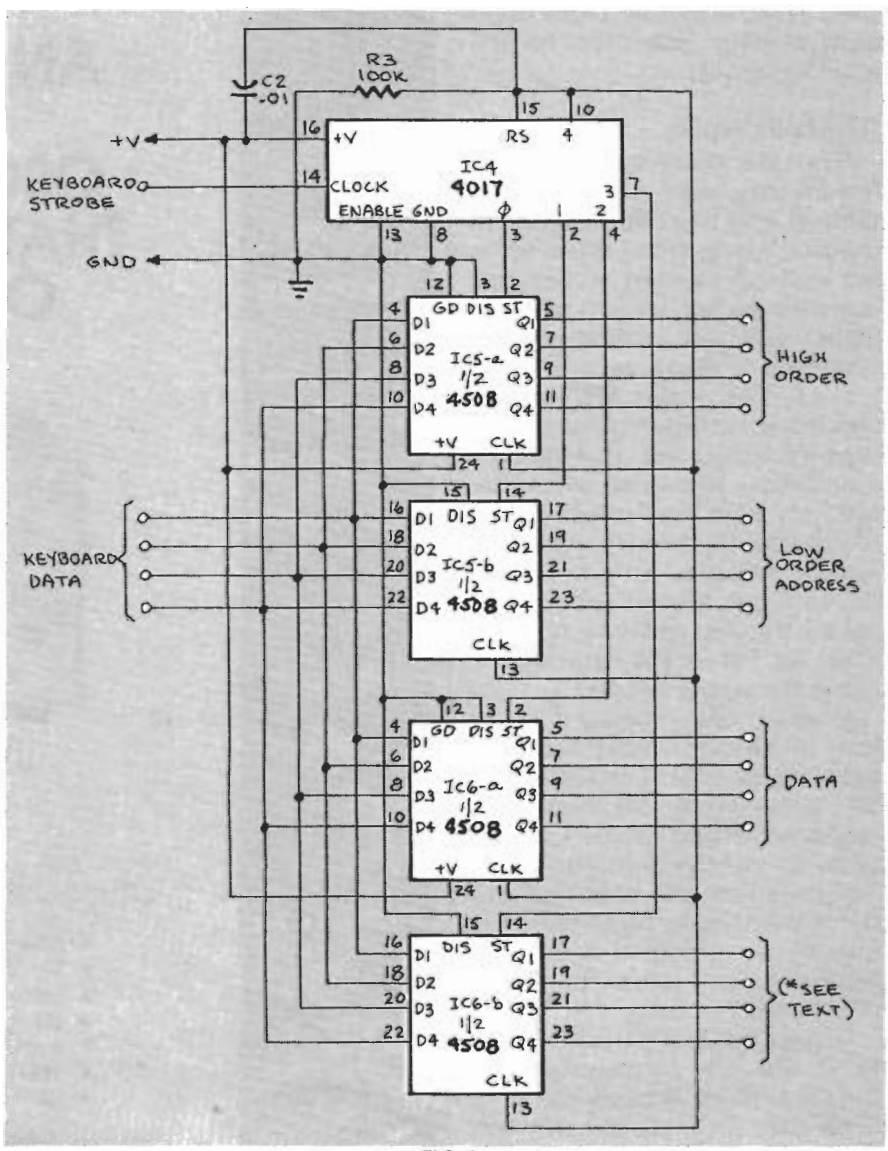

FIG. 2

The data coming out of last month's keyboard are fed to a series of latches, each of which is four bits wide. Since the 5101 (the

static RAM that we'll be using for storage) has eight address pins and four data inputs, we'll need three latches to handle the job. You can get IC's that are 4-bit latches, but a "neater" way is to use a 4508. It's a 24-pin IC that is really two 4-bit hold-and-follow latches in one package. By using that IC, we'll only need two 4508's and have one latch left over for any brainstorms we might come up with in the future.

The easiest way to sequence things is to use a 4017 binary counter, an IC you should be really familiar with by now. We spent some time discussing that IC in November and December, 1983. Now that we have the cast of characters for this portion of the circuit, let's put them together and see how they fit into our design.

How it works

Figure 2 is a schematic of the circuit we'll use to sequence the binary in information from the keyboard encoder that we built last month. It consists of a 4017 counter/driver and two 4508 dual 4-bit latches. (If you're wondering about the parts numbering, I'm keeping things in line with last month's circuit to avoid confusion.)

The action of the 4017 is (or should be) self explanatory. One thing that does deserve a bit of attention, however, is the way the clocking is being done. The 4017 sequences on the positive going (ground to +V) half of the incoming clock pulse.

Note that the four data lines (0–3) of the 4017 are connected to the STROBE inputs of the latches. That is done to sequentially enable each latch. Also notice that the DISABLE pins of each latch is tied to ground: That means that the outputs of the 4508's are permanently enabled. However, there is no data output unless the strobe input is high.

The 4508 can provide a three-state output, but there is no need for it because there's no common output-bus. Each latch will be used to control different parts of the memory and will, therefore, be connected to different pins. (But keep the three-state option in mind because many applications

require that feature, which is not found on all latches.)

When power is turned on, all four latches are cleared by the R-C pulse generated by R3 and C2. Also at turn-on, pin 3 (output "Ø") of the 4017 goes high and enables the inputs of the first latch. The circuit then sits there and waits patiently for you to press a key on the keyboard. In other words, any data presented to the input of IC5-a will now be transferred to its output.

When a key is pressed, encoded data enters IC5-a and is passed on to its outputs (which is connected to 4 address inputs on the 5101 CMOS static RAM). At the same time, the strobe line goes from high to low and remains in that state so long as the switch is held down.

After the key is released, the strobe line returns to the high state and the 4017 advances by one count. That disables the first latch (IC5-a) and enables IC5-b, the next latch in the chain. The same action is repeated for each successive latch.

When putting the circuit together (and you should), note that the 4017 counts on the release of the switch rather than on the pressing of the switch. You could make things happen when the key is first pressed if you want, but it would take a bit more hardware and, quite honestly, I can't think of one reason for doing things that way.

When data comes off the keyboard, they are sequentially stored in each latch automatically. Now, automation is a wonderful thing, but there are times when you want a little more control. In our case, automation means that there's no way to go back. Put another way, if you hit the wrong key, there's no way to correct it. That's the penalty you pay for not designing things to work with an "enter" key.

If you feel that you'd like to be able to go back, or you want to actually strobe the data into the latch separately, all you have to do is put a clear switch into the circuit, or clock the 4017 with a separate switch. How to go about doing such things is a good exercise to see if you really have a clear understanding of all the things we've done so far. Design it yourself and try it out. As for me, I'm all in favor of automation.

When you add extra features to the circuit, keep in mind all the design rules we've discussed. Write everything down, from the criteria to the actual hardware you're putting in the circuit. As I said before, bad habits are hard to break, and any circuits designed with bad habits have a way of going up in smoke!

You'll notice that we have an extra latch left over. Since all we're handling is four data-bits and eight address-bits, the last latch seems to be an unavoidable waste of hardware. Why not put in some brain burning time until next time; see if you can think of some use for it? I've already got something slick in mind—how about you?

Feedback

Before we end this month's discussion, there's a piece of important business that requires immediate attention. I thought that I'd taken care of it some time ago, but it seems that it needs a little bit of clearing up and now's as good a time as any.

I've received several letters lately that contain more or less the same comment. Several of you have said, "I like what you're doing but" (followed by): "I know a way to do it better;" "I know a way to do it easier;" "I can do it with fewer parts;" "My way uses a lot less power," and other such statements.

I guess there's some confusion about what you're supposed to get from this column. There are many schematics of workable circuits drawn and explained here, but the point of them *isn't* to compile a list of construction projects or, for that matter, anything like the sort of thing you'd see in the many books dealing with the 555 timer.

If you look over some of our past discussions, like the ones on the 4018, which began in January, 1984, you'll realize there's much more to it than just figuring out how to generate sinewaves. After all, our final circuit only had a handful of parts and was far short of what is needed for any really serious audio measurements. Not only that,