DRAWING BOARD

Bank-switching

ROBERT GROSSBLATT,
CIRCUITS EDITOR

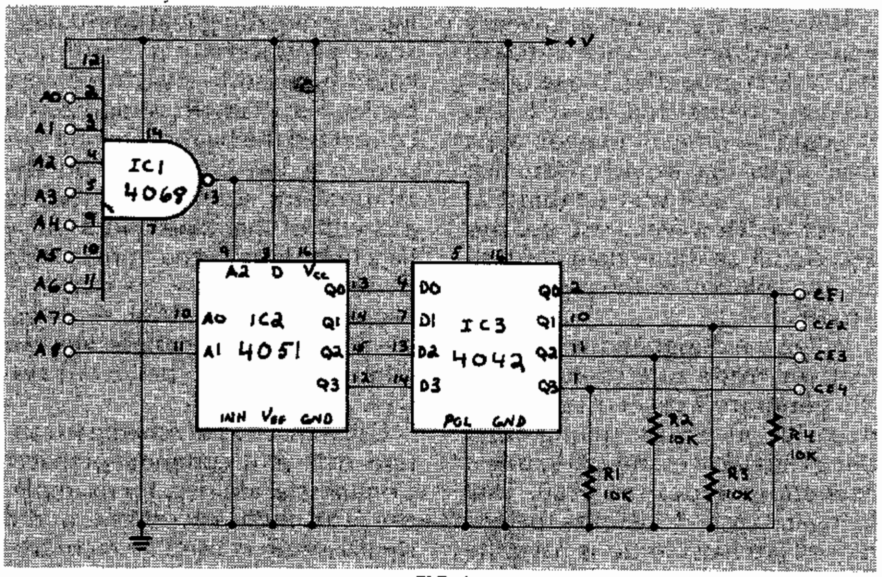

FIG. 1

WHEN WE LEFT OFF LAST TIME WE HAD just finished designing a basic memory-management circuit. If you breadboarded the circuit, you may have questions about how it works or how to use it. Before we discuss usage, let's discuss a few of the nitty-gritty details of circuit operation we couldn't get to last time.

Except for the addition of resistors R1–R4, the circuit shown in Fig. 1 here is quite similar to the circuit shown in Fig. 3 last time. We'll discuss why we need the resistors below; for now let's look at the circuit just in terms of logic.

Since the CMOS data selector we're using is a 4051, we can send either a high or a low to the selected output (Q1–Q4), depending on what the 4051's DATA input (pin 3) is connected to. In this case, it is connected to the positive supply rail.

When the low-order address line (AØ–A6) are high, IC1's output goes low. Then, depending on the state of A7 and A8, a high will be fed to one of the outputs of the 4051. That high will be clocked into four-bit latch IC2. That signal provides an active-high memory-enable signal.

Active-high or active-low?

Each of the four banks of our memory system is made up of four 5101's. The circuit was shown in Fig. 1 of the April 1986 issue of **Radio-Electronics,** and it contains one subtle error that could be confusing in light of the bank switching circuit presented here. The pins marked CE1 in that figure should be marked $\overline{CE1}$. The active-low *vs.* active-high issue wasn't important there, but adding additional banks as we're doing now requires careful attention to the level of the enabling signal.

That aside, if you examine that circuit, you'll see that we're doing memory banking there as well. Each bank is composed of two IC's, and the $\overline{CE1}$ pin of each IC is used for bank selection. Our circuit uses a nine-bit address bus; the high bit of that bus allows us to switch 256-byte banks transparently, yielding a 512-byte system.

What we want to do now is add three additional 512-byte banks to the system. To do that, we'll have to use the 5101's other enable pins. If the IC had only one enable pin, we'd be faced with a fairly complex gating problem, but, fortunately for us, the 5101 has *three* separate enable pins. We discussed how they work in the April column.

We use pin 19 to select a 256-byte page in a bank, so we'll use pin 17 to select the bank as a whole (512 bytes). If that sounds confusing, it may help you to think of each 512-byte page as a separate section.

If you bring pin 17 of the 5101 low, the IC "goes to sleep." Actually, it goes into a low-power mode in which all data are retained and power requirements are reduced to less than 10 microamps. So, by using pin 17 as a bank selector, we get reduced current drain as a freebie! The point is that the banking circuit must put a high on the enable pin of the bank we want to select and a low on all the others. That's why we tie the DATA input (pin 3) of the 4051 high.

The reason we need resistors R1–R4 is to ensure that the memory's enable pins are in a well-defined state. We wouldn't want those pins to float; data in the RAM might be garbled if the IC were accidentally accessed for even a brief period of time. Since we need active-*high* enable signals, we need to use pull-*down* resistors.

To use the bank-switching circuit shown here with the memory circuit shown in the April column (**Radio-Electronics**, April 1986), tie pin 17 from all 5101's in each bank together and connect that line to one of the CE outputs of the circuit shown here. That gives us four banks of memory, each of which is selectable by flipping a soft switch located at an address ranging from 1FC to 1FF.

How can we select a memory bank? And how can we initialize the system properly? Those questions are interrelated, as we'll see. It would be difficult to generate a reset signal that would ensure that only one bank was enabled at power up. However, if we use our Z80 circuit as the system controller, the solution to the problem is simple.

When we designed the Z80 system, we made sure there was an RC-generated reset pulse produced at power-up. Among other things, that forces the Z80 to begin execution at location ØØØØ. That being the case, to initialize the system all we have to do is have the Z80 flip one of the soft switches before it tries to access any RAM. To do that, the Z80 can execute any instruction that causes the address of the soft switch to be put on the address bus. The system's hardware will take over the task from that point.

Any of the Z80's "load" instructions is a good choice. For example, if the first instruction in your program is LD A,(Ø1FF), the Z80 will obligingly put Ø1FF on the address bus and cause our circuit to enable one of the memory banks. Believe it or not, that's all you have to do to flip a soft switch. And it doesn't matter which instruction is used as long as it results in the appearance of a soft switch's address on the bus.

Other uses

We mentioned last time that there are several ways to set up alternate banks of memory in a system. Soft switches are a neat way to organize memory, but they can be used for other things as well. And there are other ways of generating banking signals too. Let's look at some of those.

For example, if you're using only seven bits of data, you could use the eighth bit as a select line. Set the bit to talk to one bank of memory, and reset it to talk to the other. A more reasonable alternative would be to use an otherwise-unused Z80 control signal—an unused address line, perhaps, would serve well.

An externally generated signal may be a good choice in some systems. For example, if you're sensing and recording real-world data, some predetermined condition could be used to switch memory banks. For example, suppose you connected the bank switching system to a low-battery or power-failure alarm. A signal produced by a circuit of that kind could cause data to be transferred from volatile dynamic memory to battery-backed-up CMOS RAM. And yes, it's perfectly reasonable to mix two different memory types in one system.

Using that sort of scheme to capture data is a common practice. Commercial airlines, for example, use it in flight recorders, and a similar setup is used by recorders for unattended remote data gathering.

There are a few things to keep in mind when you set up a banked-memory system. The IC's you use in your switch circuitry must match or exceed your system's operating speed. The CMOS NAND gate we used is slow—about 150 nanoseconds. You could use a high-speed CMOS or TTL part to increase that speed by about a factor of 10.

If you're really interested in microprocessor system design, put the circuit together and connect it to the Z80 demo system we've been discussing. Of course, the Z80 has a 16-bit address bus, so switching 512-byte banks of memory is unnecessary, but you'll learn a great deal by making the system work. And that knowledge could be put to real-world use by designing a banking system that switches between 64K banks—and a memory system of that sort can be very useful. **R-E**