DRAWING BOARD

Designing with memory IC's

ROBERT GROSSBLATT

EVERYONE KNOWS THAT THERE'S A world of difference between theory and practice in electronics. As we've seen time and time again, what works perfectly well on paper tends to blow up perfectly well on the breadboard. I can't tell you how many times I've helplessly sat back and watched acres of silicon "real estate" go up in smoke at the speed of light!

One way to avoid blowing up expensive or even inexpensive components is to be really familiar with the eccentricities of the device. That applies to everything in your design and not only IC's. Switches, relays, batteries, and even lowly resistors have operating peculiarities that can screw things up under what would seem to be the most ordinary of circumstances. Therefore, it is best to know a little something about a component before you begin using it.

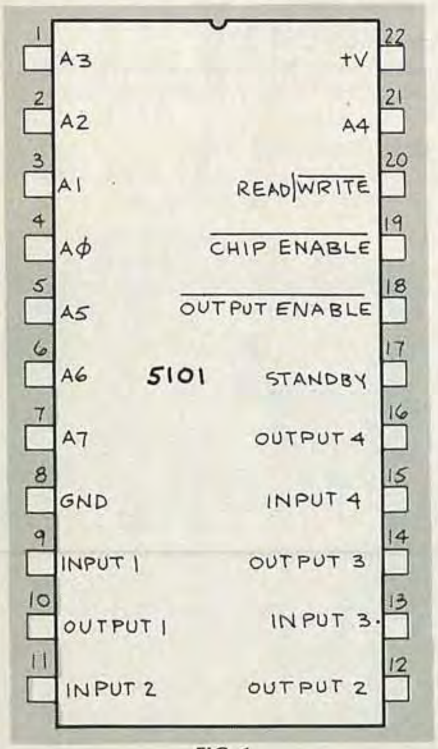
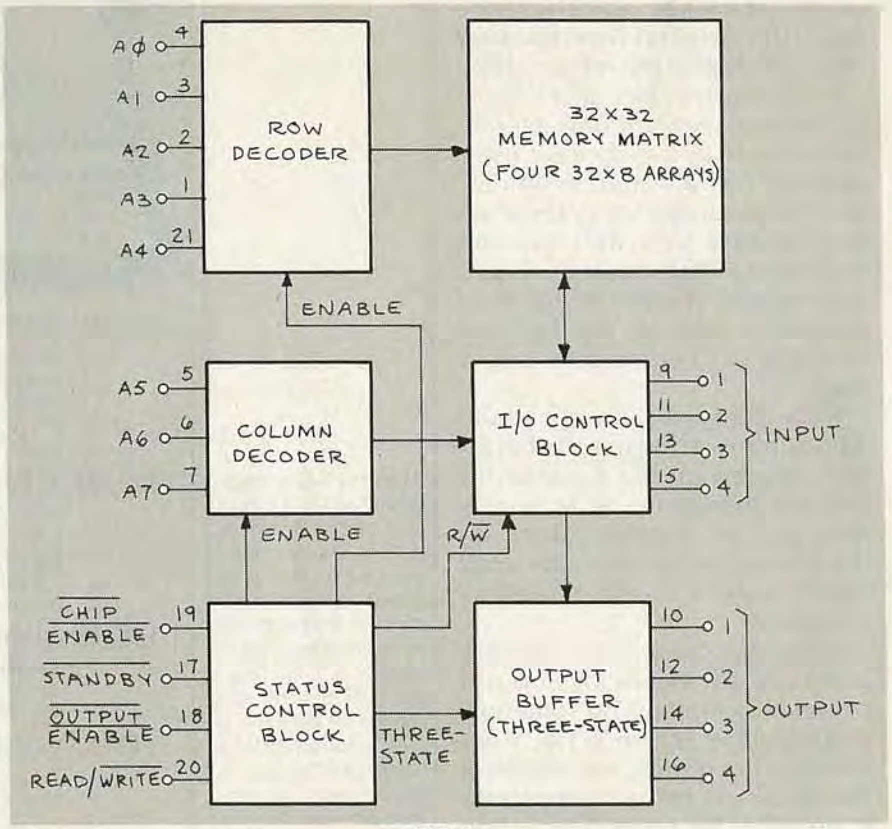


FIG. 1



Aϕ 4
A1 3
A2 2
A3 1
A4 21
ROW DECODER
32 X 32 MEMORY MATRIX (FOUR 32 X 8 ARRAYS)
ENABLE
A5 5
A6 6
A7 7
COLUMN DECODER
I/O CONTROL BLOCK
9 1
11 2
13 3
15 4
INPUT
ENABLE
R/$\overline{\text{W}}$
$\overline{\text{CHIP ENABLE}}$ 19
$\overline{\text{STANDBY}}$ 17
$\overline{\text{OUTPUT ENABLE}}$ 18
READ/$\overline{\text{WRITE}}$ 20
STATUS CONTROL BLOCK
THREE-STATE
OUTPUT BUFFER (THREE-STATE)
10 1
12 2
14 3
16 4
OUTPUT

FIG. 2

The best way to learn about any electronic component is to pick up a few and do a little experimenting, or build a demonstration circuit. Nowhere is that more true than when designing memory-based circuits. Using a demonstration circuit lets you learn to safely use a particular memory, and see what requirements have to be kept in mind for its use in general.

Now, there's no single circuit you can design that will teach you everything about all types of memory. And even if we limit our discussion to RAM, we'll find that looking at one type won't teach us everything we need to know. (We've already seen that there's a big difference between the *static* and *dynamic* types.) So that you may become familiar with the fundamentals, let's start off with *static* RAM. When we're done, we'll see that only a few additions and changes have to be made to accommodate *dynamic* RAM.

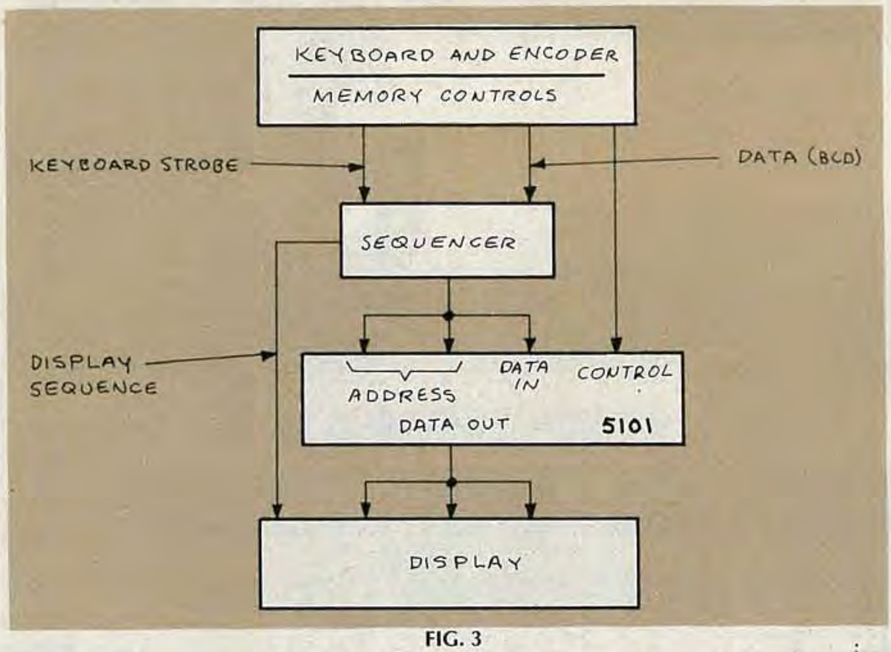


FIG. 3

Static RAM

For our discussion we'll be using the *5101* 256×4 RAM. There are several advantages to using that IC: It's cheap, (under $3 mail-order), widely available, CMOS, and features a low-power data-retention mode so a battery can be used to back up stored data.

Several manufacturers make the 5101 and although there are minor differences between them, any one you can get your hands on will be fine for our purposes. Table 1 is a listing of several pin-for-pin equivalents of the 5101. The variations in the IC usually have to do with things like maximum operating-voltage, access time, and the like. If we keep the supply at 5 volts and are willing to live with a 450-nanosecond access time, we can forget about the differences altogether.

Figure 1 shows the pinout of the 5101. A block diagram of the IC's innards is shown in Figure 2, but it's no substitute for a data sheet. The timing diagrams and such that are found on data sheets are *absolutely invaluable* when you're using memory IC's. You can build a demonstration circuit without them, but you'll learn a lot more if you have them in front of you while you work. (Think of it as a poor man's substitute for an oscilloscope.)

TABLE 1

AMI	**S5101 (any suffix)**
HARRIS	**6561**
HITACHI	**435101**
INTEL	**5101**
NATIONAL	**74C920 or NMC6551**
MOTOROLA	**145101**
NEC	**5101**
RCA	**MWS5101 or CDP1822 (any suffix)**
SSS	**5101**
SYNERTEX	**5101**
TOSHIBA	**5101**

The first step in designing the demonstration circuit or any other circuit, for that matter, is to have a perfectly clear idea in your mind of exactly what you want the circuit to do. That means we first must list all design criteria, and then draw a block diagram of the circuit. Once that's done, we can actually begin the breadboarding. The design criteria for our circuit are:

- Keyboard entry of data and address
- Switch control of read and write
- Random read and write operations
- Display of address, data in, and data out
- Automatic keyboard sequencing of address and data
- Keyboard control of all memory functions and modes

A block diagram of a circuit that meets those requirements is shown in Fig. 3.

The first thing we need is a way to generate a binary code from a keyboard. That's exactly what we'll take care of next time. **R-E**