What's a RAM?

The vocabulary of engineers or experimenters working with computers, synthesizers, electronic calculators and similar digital devices is replete with acronyms you should know. RAM is one, read on to find out what it is and how it's used.

by DON LANCASTER

ANY MEMORY IS A STORAGE DEVICE THAT is given some information at some time and hopefully will return that identical information at a later date for reuse at least once. The most elemental unit of a memory storage system is the *cell* which can store one *bit* consisting of a "1-0" or "Yes-No" simple decision. Memory cells are often grouped into *words* of several bits each. These words can represent the number in a calculator, an instruction command in a computer, a tone and its duration in an electronic music composer, an alphanumeric character in a TV Typewriter and so on.

Memories can range from one bit to many billions of bits. The equivalent of the human memory is sometimes suggested as 10 billion bits while the longest memory you can buy in a single off-the-shelf integrated circuit is 4096 bits.

There are several different types of memories. You usually classify them by when, how and how often you put information in them. A *Read Only Memory* (See "What is a Read Only Memory?", **Radio-Electronics**, February 1974) has information put into it only *once*. It keeps the information inside it more or less permanently. Read only memories are often used for such things as square root, log and trig instruction *microprograms* in a calculator, for time-zone conversion in a digital clock and for many other situations where you always want the same response to your system. Some read-only-memory systems are called "table lookup" systems, for they provide an "answer" in the same way you would get it from a math handbook. A few read-only-memory systems *can* be altered, but not rapidly. This is done by erasing them with intense ultraviolet light and reprogramming them; others are altered with special voltage or current pulses. Sometimes these are called *Read Mostly Memories*.

We could also theoretically have a *write only memory* that would accept information but never return it. Contrary to some misguided and uninformed industry jokes about WOM's, these DO exist and have a very specialized use in computer programming, particularly in data stripping and formating. Still, nobody really manufactures WOM's. When you want a WOM function, you use one location of a read-write memory over and over again instead, never bothering to read it.

The most versatile memory is one that you can *write* (put information into) and *read* (receive information from) rapidly and in any sequence. Magnetic cores are typically this type of memory, although by eliminating or not using the write current generators, we can also obtain a read only function. Most cores are *destructively* read out, meaning that the information is lost the first time you use it. You then have to perform a *rewrite after read* operation and then put the information *back* into the memory cells if you are going to use the information again.

Most semiconductor memories are *non-destructively* read out in that you can accept information without physically altering the memory contents.

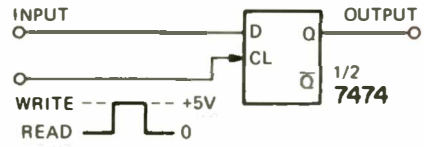
If you *must* put the information in and get it back in one specified sequence, you have a *sequential* memory. Long MOS shift registers can make a sequential memory. These have traditionally been lower in cost than true read-write memories, but have disadvantages of being noisier and having to wait a long time for the information you need to come out.

The more versatile read-write memory is one that you can read or write in any location at any time. This is called a *Random Access Memory*, or RAM for short. RAM's can be made sequential simply by deciding that you want to access or *address* things in order.

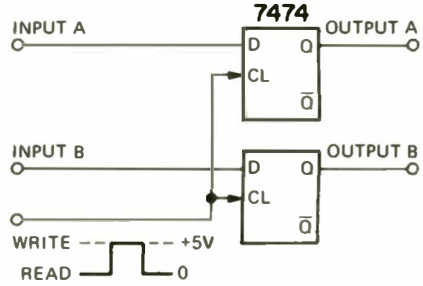
A memory is *non-volatile* if you can remove the supply power or stop moving the data around inside the memory and still hold the information. Magnetic core is usually non-volatile. Semiconductor read only memories are, of course, non-volatile. Most reasonable or available semiconductor RAM's are volatile and you must keep the supply power up or you will lose information. Many RAM's offer a reduced power mode where you can keep information for long times on battery power. In a few years, we can expect true non-volatile semiconductor RAM's, but for now, you have to design your memory application in such a way that the information is either no longer needed or stored somewhere else than in a semiconductor RAM during power down times. This really isn't nearly as bad as it seems for usually you can easily get around the problem one way or another. Often a mixture of ROM's and RAM's in a single system does the job.

There are two basic types of semiconductor RAM's. These are the *static* RAM and the *dynamic* RAM. *Both* of these are volatile and will lose information during power down times. The difference is that static RAM's will keep their information so long as power is applied without reshuffling or *refreshing* the data while a dynamic RAM has to have its internal storage moved around occasionally, often at a 500-hertz rate or faster. Static RAM's usually have a flip-flop cell for data storage. Once set or reset, it will stay in that state until power is removed or it is relr tten. Dynamic RAM's usually use a capacitor for data storage. The capacitor will eventually discharge and thus the data must be moved or *refreshed* before it is lost. Dynamic RAM's are normally far cheaper as you can pack a lot more bits onto a given size chip, but they add to the external circuit complexity and may take some elaborate timing to reliably get them to work. Thus, dynamic RAM's are more suited for very large memory systems, those over 50,000 bits or so.

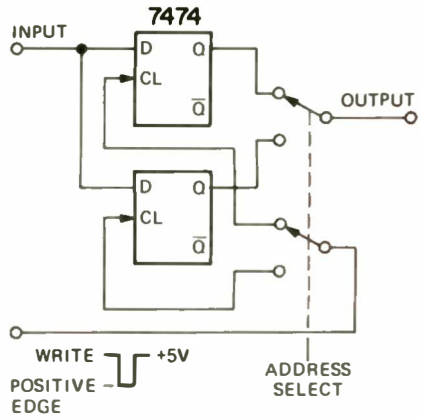
Today, you can buy a 256-bit surplus static RAM for $2.56 and get the same thing new for under $6.00. A 1024-bit dynamic



a ONE-BIT MEMORY

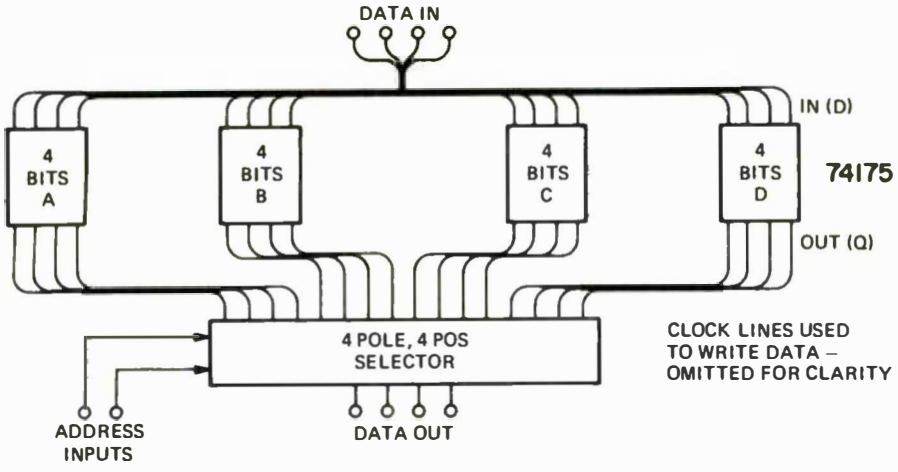


b TWO-BIT MEMORY, ONE WORD OF TWO BITS (1 X 2)



c TWO-BIT MEMORY, TWO WORDS OF ONE BIT EACH (2 X 1)

FIG. 1—SOME VERY SIMPLE RAM circuits using the TTL 7474 dual-D flip-flop.



a 4-WORD, 4-BIT-PER-WORD MEMORY USING FOUR 74175's

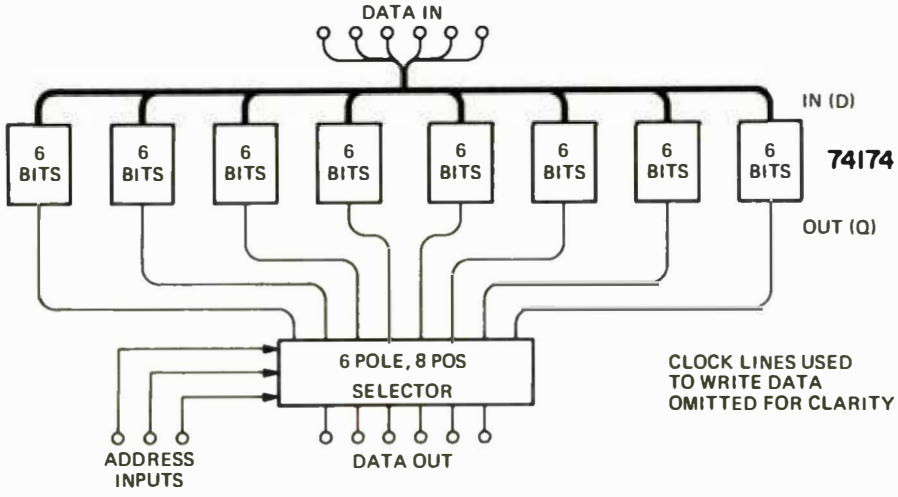


FIG. 2—TTL RAM CIRCUITS using quad and hex latches as memory IC's.

RAM runs around $5.00 surplus and under $12.00 new. Thus, we are talking prices *right now* of a penny per bit and under and projected pricing runs as low as *one-tenth* of a penny per bit. At this projected pricing, a minicomputer computer memory big enough to speak Basic or Fortran could be built for a memory component cost of $64.00 for 64,000 bits, perhaps arranged as 4000 words of 16 bits each. Simpler memory systems for things like terminals, electronic locks, music composers and a whole bunch of things nobody has thought up yet today should cost well under $20.00 and eventually should come down to $2.00. So, now is the time to start becoming familiar with these exciting new devices.

A simple semiconductor RAM

Let's start with a rather small RAM and see what we can do with it. We'll use the 7474 TTL dual type D flip-flop as shown in Fig. 1. We'll start with a one-bit memory and then double it to two bits by using both halves of the package.

In Fig. 1-a, we use half the 7474. This stage can store a "1" (often a high state around 3.3 volts) or a "0" (usually a low state around 0.5 volt). The stored value appears at the "Q" output. The opposite or *compliment* of the stored value appears at the Q output. We have a data or D input and a clock or CL input. Information present on the D line gets loaded into our memory at the time the clock goes from ground to a positive value. To enter information into our memory, we put the information on the D line. At that time, it does NOT go into the cell. At the instant we bring the clock line from ground to a positive level or from a TTL positive logic 0 to a positive logic 1, we actually load or write the information into our flip-flop cell. Whatever was on D at the instant of positive edge clocking gets loaded into the memory and appears at the Q output.

This is a random access memory as we always can get to the memory cell (trivial, as we only have one cell) anytime we want. It is static as it will keep the loaded information for as long as we apply power. It is volatile as the information will go away if we ever shut off the +5-volt power supply. And our simple memory is *organized* as "one word of one bit each."

We can watch or *read* our memory any time we like, but since a change may be produced during clocking, we shouldn't be using or reading at that particular instant. We call the clocking interval the *write cycle*. Time spent looking at this particular cell's output is called the *read cycle*. Normally, you don't read and write simultaneously. You either *execute* a read cycle where you monitor and use the output of the memory cell or you execute a write cycle where you place new information into the cell. The 7474 will do a write or a read cycle in under 50 ns. Since nothing physically changes internal to the 7474 during reading, the readout is non-destructive and we can reuse the stored information hundreds or even millions of times if we like.

A one-bit, one-word memory by itself isn't too useful, although you can think of an alarm system as a one-bit memory and there are numerous other trivial applications. To do any really useful function, we most often need quite a few more bits of storage.

Figure 1-b shows how you can use both halves of a 7474 to build a memory of one word of two bits. This is done by simultaneously clocking each half of the package and using both outputs at once. Thus we have two data input lines, two output lines and one write line. This organization is *one word of two bits*. In 1-c, we have the opposite, a memory of *two words of one bit each*. Now something new has been added. We have to *combine* or *select* which of the two memory bits is going to appear as an output. We also have to decide which of the two cells is going to have data written into it at any given time. This decision is called *addressing*. We now have to address cell A (a 0 on the address line) or cell B (a 1 on the address line). By controlling the address line, we select which memory cell is to be acted upon or read.

The more cells we have, the more complicated the addressing will become. Note that we needn't alternate memory cells if you don't want to. You can address either cell in any sequence you want. Hence the name *random access*.

Adding more bits

We could use as many 7474's as we like to build up any memory, but even at surplus prices, the 25¢ or so per bit and the large supply power and size will eventually get to us. The next step up is to use packages with more than two D flip-flops. Quad and hex latches, the 74175 and 74174 are a good choice. Figure 2 shows some memory circuits using these components.

In Fig. 2-a, we have a 16-cell memory arranged as four words of four bits each. We have four data lines, four output lines and *two* address lines. These two address lines are binarily decoded (00, 01, 10 and 11) to get at the *four* possible memory cell locations. We might use this memory to store four BCD numbers as part of a computer or calculator.

In Fig. 2-b, we use eight 74174's to build a 48-bit memory organized as eight words of six bits each. This time, we have six data input lines, six output lines, and *three* address lines. The three address lines are decoded (000, 001, 010, 011, 100, 101, 110 and 111) to get at the *eight* possible locations of six cell groups. Since we can represent a letter, number, space or punctuation with six bits of the standard ASCII code, this memory could be used to store an eight character message.

Which organization?

Suppose we had a 64-cell memory. How could we group the cells to obtain different combinations of bits-per-word and numbers of words? Figure 3 shows some possibilities. While each of these memories is 64 bits total capacity, the *organization* of each is different.

In Fig. 3-a, we have one word of 64 bits each. We need zero address lines since we are always looking at the same word, but we

need 64 input lines and 64 output lines. In Fig. 3-b, we have two words of 32-bits each. We now need *one* address line to select which half of the memory is to be written into or read from. There are 32 input leads and 32 output leads. The next combination of Fig. 3-c would be four words of 16 bits each. Here we need *two* address lines binarily decoded to select which quarter of the memory is to be active and there would be 16 input leads and 16 output leads.

You can rapidly run down the other organizations. Figure 3-d gives us eight words of eight bits each. There are three address lines needed that are decoded one-of-eight to pick one-eighth of the memory for use and we have eight input lines and eight output lines. In Fig. 3-e, we have four words of 16 bits each. Four address lines decode one of sixteen and there are four input and output lines. Two words of 32 bits each take two input lines, two output lines and five address lines, the latter decoded one- of -32 as shown in Fig. 3-f. Finally, in Fig. 3-g, we have 64 words of one bit each. There is one input line, one output line and six address lines which are binarily decoded one-of-64 to pick which of the individual memory cells is to be interrogated.

So, we have a wide choice of organizations to any memory. The more the bits, the more the choices. Which do we use?

This depends on you if you are working with a large system and depends on the integrated circuit manufacturer if you are trying to get the job done with only one or two stock integrated circuits. Obviously, you organize the memory to suit the information you are trying to put into it. Four-bit words are common for BCD (binary coded decimal) number storage in calculators. Six-bit words are often used to store ASCII characters. If the full ASCII code, including transparent control commands and lower case and error detection is to be used, we have to up to eight bits per word. Or, we might like to use the remaining two bits to select a color on a color display. We could get one of four with two bits. Minicomputers tend to use 8-, 9-, 12-, 13-, 16-, 17-, 18-, 24- or 25-bit words depending on the manufacturer and the task the computer is aimed at. So, for system's use, you pick the number of bits needed to do the job.

On the other hand, if you are a integrated circuit manufacturer, you want to have the most reasonable package in your system. The majority of semiconductor memories only have ONE input line and ONE output line and address lines for one-of-N decoding, giving you organizations such as 256 one-bit words 1024 one-bit words, 4096 one-bit words, and so on. Occasionally a smaller memory may have four bits per word, to make working with BCD numbers easier. Other arrangements are rarely used and you usually add packages to pick up the total number of bits you want.

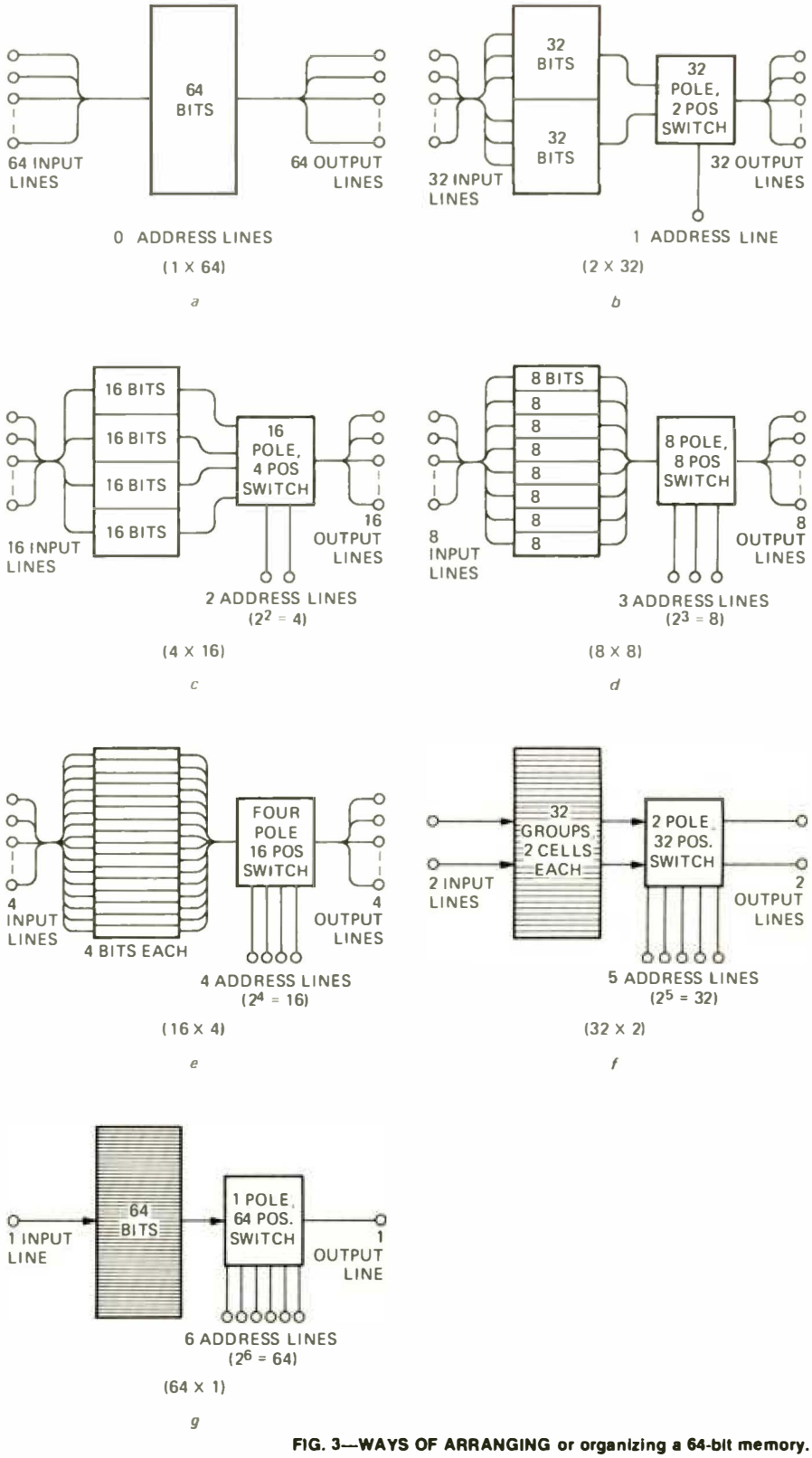


FIG. 3—WAYS OF ARRANGING or organizing a 64-bit memory.

Decoding

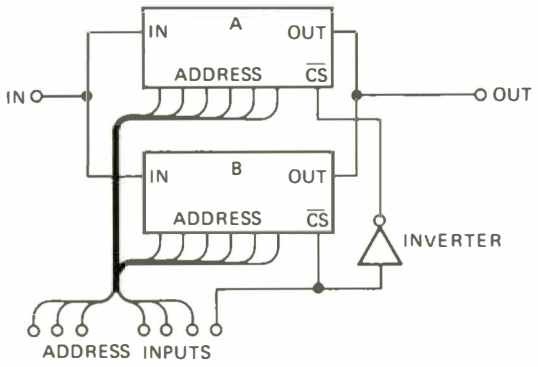
All organizations in Fig. 3 have binary to one-of-N decoders on the address lines. If this decoder is internally provided in the integrated circuit as it almost always is, we have an *internally decoded* memory. If we must provide external address decoding as is common with magnetic cores, we need *external decoding*. External decoding is also needed when you have several memory packages that you are combining for a total storage. In this case, you use *output enable* or *chip select* lines to pick which package is to be used. Once selected, each individual package then goes on to provide internal decoding. For instance, with two IC's we could simply tie their inputs and outputs together and drive the first memory's chip select as an address line and drive the second memory's chip select from the *compliment* of that line. Thus, we pick one-of-two memory IC's and the chip selects give us a new form of addressing. If we tie four memories together, we use two new address lines, one-of-four decode them and then chip select only one memory at a time. Figure 4 shows how you can expand memories using the chip select system.

Unlike magnetic cores and many older memory systems, the data input and output lines are completely separate with most new semiconductor RAM's. This eliminates amplifier recovery problems, steering networks, "single port" problems and things like this.

Who makes what?

Figure 5 is a list of my choice of the best

HERE WE USE A NEW ADDRESS LINE AND AN INVERTER TO SELECT ONE OF TWO MEMORY IC'S. IF A AND B ARE ORGANIZED AS 64 x 1, NEW MEMORY IS 128 x 1.



WITH TWO NEW ADDRESS LINES AND A ONE-OF FOUR DECODER, WE CAN QUADRUPLE A MEMORY BY USING FOUR IC'S. ONLY ONE MEMORY IS ENABLED AT A TIME. IF A, B, C, AND D ARE 64 x 1 MEMORIES, SYSTEM IS A 256 x 1 MEMORY.

ANY NUMBER OF IC'S CAN BE USED BY ADDING NEW ADDRESS INPUTS AND 1-OF-N DECODING THEM.

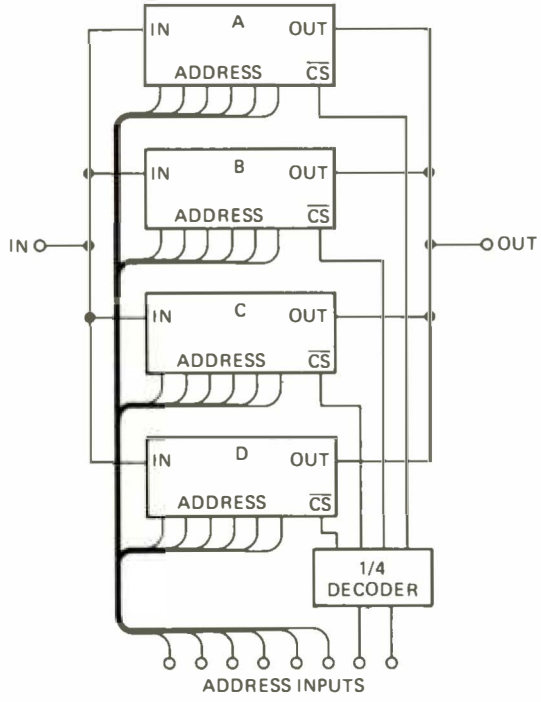


FIG. 4—MORE THAN ONE IC can be used in a memory by using the chip-select as a new address input. Everything is connected in parallel, but only one IC is enabled at any time.

NUMBER	BITS	TECHNOLOGY	ORGANIZATION	SUPPLY	MANUFACTURER*
7474	2	TTL	2x1 OR 1x2	+5,0	TEXAS INSTS.
74175	4	TTL	1x4	+5,0	TEXAS INSTS.
74174	6	TTL	1x6	+5,0	TEXAS INSTS.
7489	64	TTL	64x4	+5,0	TEXAS INSTS.
25L01	256	P MOS	256x1	+5,-12	SIGNETICS
2102	1024	N MOS	1024x1	+5,0	INTEL

*PARENT COMPANY—MANY SECOND SOURCES
FIRST NUMBER IN ORGAINZATION IS NUMBER OF WORDS; SECOND IS NUMBER OF BITS PER WORD.

FIG. 5—SIX IC's and their basic specifications.

ADVANCED MEMORY SYSTEMS
1276 Hammerwood Avenue
Sunnyvale, California, 94086

AMERICAN MICROSYSTEMS INC.
3800 Homestead Road
Santa Clara, California, 95051

ELECTRONIC ARRAYS
501 Ellis Street
Mountain View, California, 94040

INTEL CORPORATION
3065 Bowers Avenue
Santa Clara, California, 95051

INTERSIL MEMORY CORPORATION
10900 North Tantau Avenue
Cupertino, California, 95014

MICROSYSTEMS INTERNATIONAL
Box 3529-C
Ottowa, Canada, K1Y4J1

MOSTEK
1215 West Crosby Road
Carrolton, Texas, 75006

MOTOROLA SEMICONDUCTOR
Box 20912
Phoenix, Arizona, 85036

NATIONAL SEMICONDUCTOR
2900 Semiconductor Drive
Santa Clara, California, 95051

SIGNETICS
811 East Arques Avenue
Sunnyvale, California, 94086

TEXAS INSTRUMENTS
Box 5012
Dallas, Texas, 75222

FIG. 6—IC MEMORY MAKERS. Write for information you need.

current bets for experimental memory use. These are easy to use, widely available and low in cost. Leading memory manufacturers are shown in Fig. 6. As with any semiconductor work, always have the exact data sheet and as much as you can get in the way of applications information before you begin any work with semiconductor RAM's.

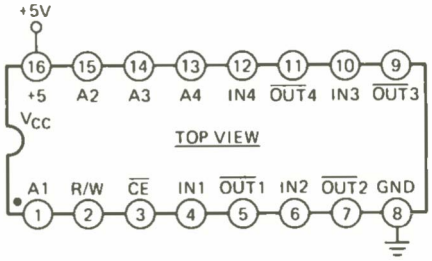
There are lots of different ways to classify semiconductor RAM's. One grouping is based on the process used. *Bipolar* RAM's include TTL and ECL logic. MOS versions include P-channel, (metal and silicon gate), N-channel, and CMOS types. In the past, MOS devices have almost always been slower and much cheaper. Some MOS memories are now as fast as TTL and most MOS devices will continue to be cheaper than bipolar for some time to come.

MOS memories are further broken down into *static* and *dynamic* versions. Dynamic versions are much cheaper and much harder to use, particularly in experimental or very small system applications.

Let's take a closer look at some specific IC's:

7489

The 7489 is a good choice for initial experiments with RAM's. It is TTL and works off a single 5-volt supply. Organization is 16 words of four bits each as shown in Fig. 7.



CHIP ENABLE LOW ENABLES MEMORY
CHIP ENABLE HIGH INHIBITS MEMORY
R/W LOW WRITES
R/W HIGH READS

FIG. 7—THE 7489 IS A 64-BIT TTL memory organized as 16 x 4 or 16 words of 4 bits each (16 4-bit words). It needs a single +5-volt supply. Output information is a compliment of the input.

There are four data inputs and four data outputs along with four address lines. The address lines are four-line-to-one-of-sixteen decoded internally. Internal circuitry is arranged so that you store and read out the *compliment* of the input information.

To read this memory, you apply a four-bit address to pick the slot you want to look at and then bring the memory enable line low. For instance, address 0101 selects the *fifth* group of four cells. Data appears at the output shortly after the address is stable.

To write into the 7489, pick an address, input the compliment of the data you want to store and then briefly bring the write enable low. This loads the memory.

One thing you have to watch very carefully in any semiconductor memory is that the address cannot be changed immediately before, during or immediately after a write command. (The definition of "immediately" varies with the IC—carefully consult the data sheets!) As a memory address changes, certain locations are "flashed" by in the decoding process. It is possible to write, erase or physically move data around if you aren't careful. ALWAYS *PULSE* THE WRITE COMMAND ON ANY SEMICONDUCTOR MEMORY. *NEVER* CHANGE ADDRESSES DURING WRITE PULSING! Put another way, always leave the memory in a disable or a read mode. Don't put into write mode until after the address is stable.

This particular memory cycles in under 50

(continued on page 78)

(continued from page 53)

ns. If you are running any memory very fast, times will occur when the old information or wrong information will be put out until the answers settle down. If this "garbage" time is too great for your application, you can add a latch to the output (perhaps a 74174) to sample the output only during instants when you know the data is good. A very few new memory IC's include internal latches and eliminate this problem.

By the same token, if you are running fast, the ripple and gate times on address changes may cut into your cycle time significantly. Again, if you are running fast, it pays to either use fully synchronous timing or else latch the addresses to get them all changing at once. Some semiconductor mainframes get around the problem by using emitter coupled logic (ECL) and its very high speeds for addressing.

The 7489 has a few obvious and apparently untapped electronic music applications. For instance, you can use sixteen four-bit words to completely specify one cycle of a music waveform, the attack-sustain-decay envelope of a note or a melodic sequence. These run around $3.50 surplus and under $11.00 new.

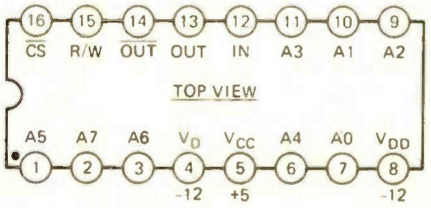
Other TTL memories

There's quite a few other TTL memories available, some as long as 1024 bits. The 7481 is a very old design arranged as sixteen words of one bit each. The 74170 is called a 4 × 4 file, meaning it is a 16-bit memory arranged as four words of four bits each. The 74200 and faster 74S200 are a 256 × 1 memory or 256 words of one bit each. There's also a bunch of "non-7400" TTL memories. The *Signetics* 8225 is a pin-for-pin replacement for the 7489.

1101

The 1101 is a MOS static memory arranged as 256 words on one bit each. It's shown in Fig. 8. MOS memories are generally much cheaper and often much slower than TTL ones. The 1101 works on +5, −9 supplies and runs quite hot. There is TTL compatibility on inputs, addresses and outputs. There are seven address lines, internally decoded to pick one of the 256 bits. There is one input line and two output lines, a normal one and its compliment.

To read, you make the chip select low and the read/write low after applying the selected address. The output data will be valid within a microsecond or so afterward.



CHIP ENABLE LOW ENABLES OUTPUT
CHIP ENABLE HIGH DISABLES OUTPUT
R/W HIGH WRITES
R/W LOW READS

FIG. 8—256-BIT MOS MEMORY takes +5 and −12-volt supplies. Typical devices are Signetics 25L01 and Mostek MK4007-4P. Older 1101 devices use same pinout but run hotter and take −9-volt supplies.

To write, you select your address, wait 300 ns, bring the read/write line high for at least 400 ns and then wait at least 100 ns after the write line goes low before changing addresses. As usual, NEVER change the address during, before or immediately after writing.

The 1101 is widely available and costs as little as $2.56 for probably good surplus units and as little as 50¢ for questionable surplus units. New cost is under $6.00. One possible application would to be using six of them in a data terminal or programmable calculator to store a 256-word message using the ASC11 code.

Improved 1101's

The original 1101's were rather slow and could take as long as 1.5 ms to read. They are very hot running and the −9 supply is usually a rather wierd thing to have to provide. Improved devices are now available. An 1101A1 cycles in one microsecond maximum. The *Signetics* 25L01 and the *Mostek* MK4007-4P are second-generation, pin-identical, versions that cycle in under a microsecond, consume much less supply power and work on standard +5, −12 supplies.

CMOS RAM's

One new type of 1101 replacement is the CD4061, a CMOS device made by RCA. This is a pin-for-pin replacement, but being CMOS, it takes only one supply and draws utterly negligible supply power if you aren't writing or changing the address. Thus, you can use this with a very small battery for power down storage and still hold the information. You can also run on incredibly lower currents than the 1101 style devices and much faster as well—several hundred nanoseconds. This makes the 4061 ideal for hand-held data equipment and calculators, as well as meter readers and things like this. The only hitch—it's a new device and still costs $40.00. Maybe next year.

Other CMOS memories include the *Motorola* 14505 (64 × 1), the *Solid State Scientific* SCL5554 (256 × 1) and the *Inselek* A5503 (256 × 1).

1103

The main reason we include the 1103 here is as a warning NOT to try and use it—unless you have lots of fancy equipment and considerable experience. This is especially true of surplus 1103's.

The 1103 is a 1024-word x 1-bit dynamic shift register. It is very low in cost. It ranks as the all time most successful single integrated circuit and it toppled "king core" from the computer world. The device trades a very simple and very dense internal circuit for quite a bit in the way of outside support circuitry. This IC needs critically controlled clocks, usually needs an output sense amplifier, and has a complex timing sequence so elaborate that a 30-ns overlap error in the wrong place will cause information dropout. The 1103 is eminently suited for large memories of at least 50,000 bits (this is tiny by mainframe computer standards) or so, where all the critical support circuitry is easily worked with and may be offset by the savings you get by cramming 1000 bits in each package.

The 1103 uses capacitors for internal data storage. The data must be moved around or *refreshed* at least 500 times per second.

(continued on page 98)

WHAT'S A RAM?

(continued from page 79)

The 1103 is obsolete today. There are some significantly improved devices available today that are much easier to use, but they still are a rather tough design problem if you do not have elaborate equipment and considerable digital know-how. Improved versions include the *Intel* 1103-A, the *Mostek* MK4006 and MK4008, the *Electronic Arrays* EA1500 1501 and 1502 and the *American Microsystems* S3103.

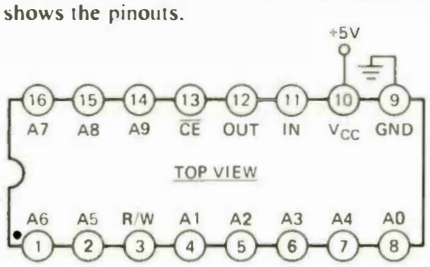
4096 bits

The big race today is to build an improved 1103-style integrated circuit with 4096 bits. At least one has been announced by TI at a 100-lot price of $26.00, or around 0.6¢c per bit. The other manufacturers aren't taking this sitting down and the race is on. Pinouts have pretty much been standardized and some should be available as you read this. Competitive products include the *Electronic Arrays* 1504, the *Intel* 2107A, *Standard Microsystems* 4412, *Texas Instruments* 4030, *Microsystems International* 7112, *Mostek* 4096, *Motorola* 6605 and probably a bunch more. Only eight of these integrated circuits are needed to build a decent minicomputer main memory.

+5-volt, single supply MOS memories

A number of new, very easy to use and interchangeable MOS memories are now available that use an n-channel static technology. They have no clocks and are entirely and absolutely TTL compatible. There are no clocks or sense amplifiers needed. These include the *Signetics* 2602, the *Intel* 2102, *Intersil* IM7552, *Motorola* MCM6602 and the *Microsystems International* MF 2102. Cost in single quantities is around 2¢ per bit as of this writing. They are far too new to crop up surplus. Organization is 1024 × 1, or 1024 words of one bit each.

These integrated circuits cycle in a microsecond and screened 0.5-ms devices are also available. There are ten input address lines, a data in and a data out. Figure 9 shows the pinouts.



CHIP ENABLE LOW ENABLES OUTPUT
CHIP ENABLE HIGH DISABLES OUTPUT
R/W HIGH READS
R/W LOW WRITES

FIG. 9—1024-BIT N-CHANNEL memory works off a single +5-volt supply and is very easy to use. Typical devices include Intel 2102, Signetics 2602, Motorola MCM6612, Intersil IM7552 and Microsystems International MM2102.

To read, you pick your address with the chip select low and the read-write high. The output data appears within a microsecond or so of an address change. To write, apply your input data, select your address, wait 400 ns, bring the write line low for at least 500 ns, send it back high again and wait at least 100 ns before changing the address. Once again, don't change the address immediately before, during or immediately after the write line is active low. The chip select can be used to expand the memory by several IC's. Six of these in parallel are ideal for a data terminal or TV typewriter memory. Prices should drop well under a penny per bit by next year.

What good are semiconductor memories? Calculators, programmable computers, teaching machines, terminals, TV typewriters, electronic games, minicomputers, full-blown computers, electronic music and hundreds of other applications exist now. What can *you* do with them? Let us know. **R-E**