Experimenter's Corner

By Forrest M. Mims

READ/WRITE MEMORIES (RAM's), PART 1

LAST MONTH, we built a read-only memory (ROM) with some diodes and a BCD-to-deimal decoder. Now, we're going to experiment with the 7489 IC, a factory-produced read/write memory that can store sixteen 4-bit words.

As you know, ROM's store information without the need for electrical power and are called *non-volatile* memories. Most RAM's, on the other hand, are *volatile* memories; turn off the power and they forget whatever information is stored in them. You've probably seen read/write memories labeled RAM's *and* R/WM's. RAM, *random access memory*, is a fancy way of saying that any bit or word stored in the memory can be addressed as fast as any other. This contrasts with a *serial* memory like magnetic tape where a time-consuming search may be required to find a particular bit or word.

Since both ROM's and RAM's are random access memories, R/WM is a better label for the read/write memory than RAM. But "RAM" is pronounceable and R/WM isn't, so most people use RAM.

The 7489 RAM. The storage capacity of the 7489 is a far cry from that of the 4k (4,096 8-bit bytes) RAM's used by hobby computer enthusiasts, but the 7489 does have some interesting applications. It will help you understand some basic microprocessor terminology and operations.

Fig. 1. The 7489 RAM pin outline.

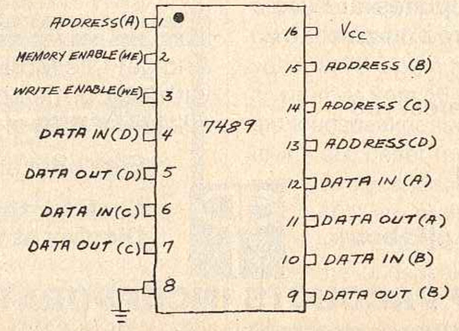

Figure 1 shows the pin diagram for the 7489. Here's a table that organizes the pins according to function:

Function	Pins			
	D	C	B	A
Address lines	13	14	15	1
Data in	4	6	10	12
Data out	5	7	9	11

The 7489 also has a couple of *enable* inputs. The Memory Enable (ME) input, pin 2, is connected to ground (logic 0) during read and write operations. The Write Enable (WE) input, pin 3, must be at logic 0 when data is written into the RAM. Data can be read from the RAM when WE is at logic 1.

The 7489 has 16 storage slots designated by the addresses 0000-1111. Thanks to a built-in address decoder, writing a word into a memory slot is a simple matter of applying the appropriate BCD number to the address lines, placing both ME and WE at logic 0, and presenting the bits to be stored at the data input lines.

Reading a word from the RAM is even simpler. First, the word's storage slot address bits are applied to the address lines. Then ME is placed at logic 0 and WE at logic 1. The *complement* of the word in the selected address will then appear at the output.

Complementing a word means changing its 0's to 1's and its 1's to 0's. Thus, the complement of 1010 is 0101. This means you have to complement a word you want to store before writing it into the memory if you want it to appear in uncomplemented form at the output. In other words, if you want to retrieve 1100, store 0011 instead.

There are two points to keep in mind in using a 7489. First, it is a volatile memory, so you must keep power applied as long as you want to save the data stored in it. It's also a *non-destructive* memory. That is, the selected word is not lost when it's read out; it stays in the memory until replaced by a new word.

RAM Demonstration Circuit. You can learn a lot about RAM's by plugging the 7489 into a solderless breadboard along with some LED's to indicate the output data and jumper wires to select addresses and apply input data. Use the arrangement shown in Fig. 2. Take a few minutes to label each jumper with a marked square of masking tape. This will save lots of time later. Connect jumpers you want to be at logic 0 to ground and those at logic 1 to +5 volts. Try loading each storage slot with its binary address for practice.

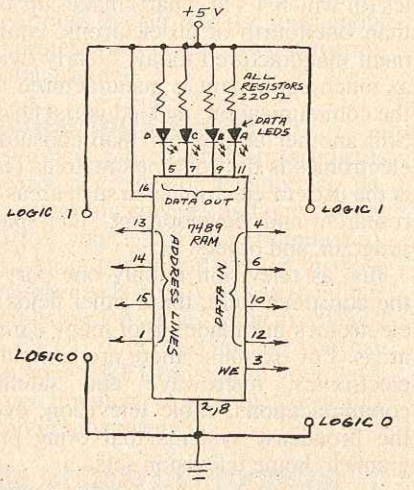

Fig. 2. A RAM demonstration circuit.

In this circuit a glowing LED indicates logic 0 and a dark LED, logic 1. You can automatically invert words stored in the memory by mentally assuming that a glowing LED signifies a logic 1. This means you don't have to load the complement of a word you want to save.

Automated RAM Demonstrator. A much better way to learn about RAM's as well as some microprocessor basics is to connect a binary counter to the address inputs of the 7489. Figure 3 is a block diagram that shows how everything goes together.

Here's how the circuit works using microprocessor terminology. Clock pulses enter the counter, and the 4-bit BCD count is applied to the address inputs of the RAM. The counter acts like a *pointer* as it sequentially selects first one address, then the next, and so forth.

In the READ mode (WE = 1; ME = 0), the data output LED's flash each word in succession as the pointer cycles through the memory. In the WRITE mode (WE and ME = 0), the LED's are extinguished, and the data on the input lines is written into the RAM. This means, of course, that the input data has to be changed between clock pulses or the pointer will load all the storage slots in the RAM with the same word.

The complete circuit for the automated demonstrator is shown in Fig. 4. It makes a great fully programmable light flasher so you'll want to build it just to watch it flash.

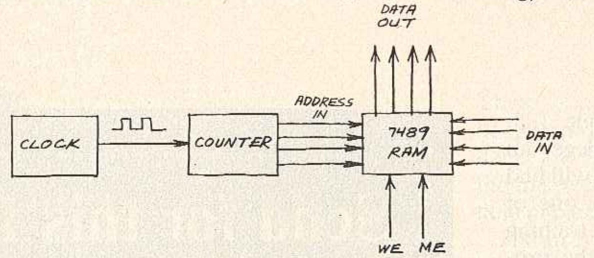

Fig. 3. Block diagram of automated RAM demonstrator.

The role of the clock is filled by a 555 timer, and a 7490 decade counter serves as the address pointer. The 7490 is a BCD counter so it recycles to 0000 after 1001 (decimal 9). This means it can address only ten of the 7489's storage slots. To address all 16 memory slots, you can use a full 4-bit counter (0000-1111) such as a 7493, 74161, or 74191. I've specified the 7490 because its operation has been covered previously here. It's also very inexpensive and readily available.

Notice the various switches and LED's in the circuit. Closing *S1* allows clock pulses to reach the counter. The CLOCK LED provides a handy visual indication that the clock is running and, below about 20 Hz, a rough idea of its rate.

The pointer LED indicates when the counter has recycled back to 0000. It's on when the count is 0000 through 0111 and off when the count is 1000 and 1001. This means the counter is pointing to address 0000 in the RAM the moment the pointer LED flashes on after being off for two pulses.

Toggle switches *S2* through *S5* let you form words to store in the RAM by simply flipping switches (much like a microcomputer designed for front-panel machine language program and data loading). Closing *S6* loads the word formed by the data input switches into the address slot selected by the pointer.

Of course, you don't have to use switches if you build the circuit on a solderless breadboard. Just remove and reinsert jumpers labeled with masking tape to simulate the on-off action of switches. Real switches, however, make the circuit much easier to use, particularly if you mount them on a small panel and identify them with labels.

Programming the automated RAM demonstrator is a straightforward procedure of loading words into the RAM one at a time until ten address slots are filled. Switch *S1* is turned off to disable the clock while a word is being loaded and turned on for one clock pulse to advance the pointer to the next address. It's easy when you slow the clock rate to about a pulse per second (by setting *R1* for maximum resistance) and keep an eye on the clock LED.

To Be Continued. Next month we'll discuss programming procedures in more detail. We'll also expand the demonstrator by adding an automatic pseudo-random data loader, and cover some ways to address all sixteen data storage slots in the 7489. ◇

Fig. 4. Automated Ram demonstrator.