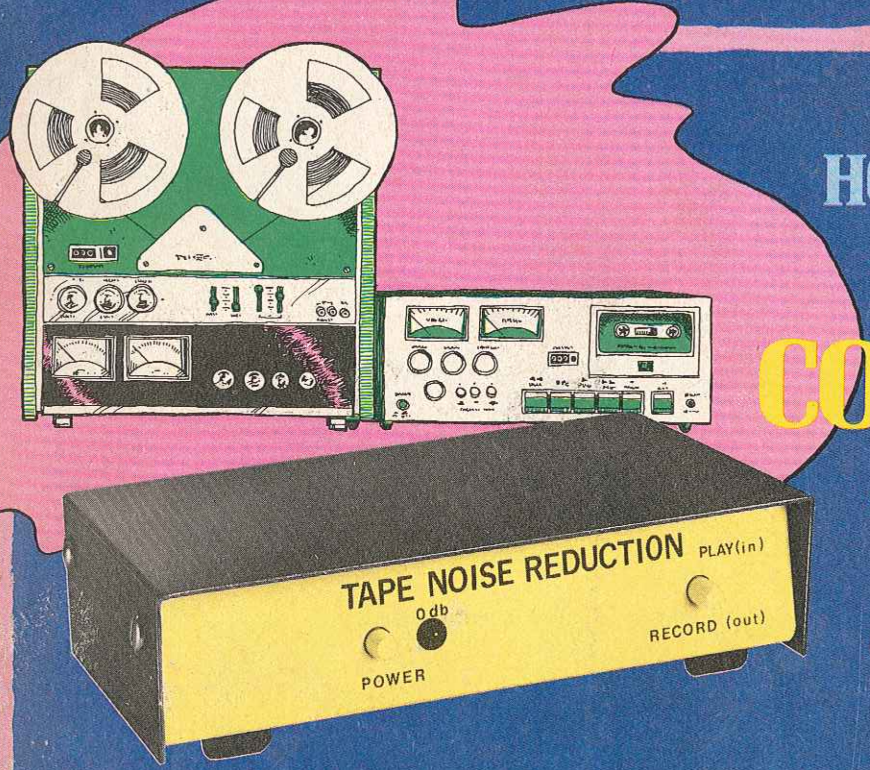
PE RATES THE NEW CASSETTE TAPES

Popular Electronics®

WORLD'S LARGEST-SELLING ELECTRONICS MAGAZINE NOVEMBER 1977/ $1.25



HOW TO BUILD AN AUDIO COMPANDER

- Improves dynamic range of tapes
- Reduces noise level

How To Dress Up Your Projects

- Tips on making projects attractive

A Field-Disturbance Sensor for Security Systems

Add Voltage Regulation to Color Enlargers

Piracy On The Airwaves

TESTED THIS ISSUE: Dual 1245 Automatic Turntable • Burwen DNF 1201A Noise Reduction System • Mitsu[illegible]shi DA-A15 Audio Power A[illegible] DA-P10 Preamp • [illegible]49 AM/SSB CB Mobile Transceiver • North Star Floppy Disk

Experimenter's Corner

By Forrest M. Mims

PROGRAMMABLE READ-ONLY MEMORIES

SEMICONDUCTOR memories are among the most important electronic circuits. They are found in almost all digital devices, ranging from pocket calculators to computers. Besides their obvious application in the storage of information, these memories can be used in the synthesis of unusual waveforms, music, and even human speech.

There are two basic types of semiconductor memories. *Read-only memories* (ROM's) are those from which data is normally only retrieved. *Read/write memories* (R/WM's) or *Random-access memories* (RAM's) are those into which data can be loaded or from which information can be retrieved, each with equal facility. ROM's are factory programmed with fixed data which cannot be changed.

Some ROM's, called PROM's, can be permanently programmed by the user. Others, called EPROM's, can be programmed by the user and then erased by exposure to ultraviolet light. After erasure, EPROM's can be reprogrammed. RAM's can be loaded with information, read, or reloaded electronically, depending on the logic states of the memory cell's READ and WRITE control lines.

Both types of semiconductor memories store information in the form of binary digits, abbreviated as bits, which have two possible states—logic 0 or logic 1. The stored data can be arranged as hundreds or even thousands of bits or combinations of bits called words. Words comprising four bits (*nibbles*) or eight bits (*bytes*) are the most common, but many other word lengths are also used.

Programmable Diode ROM's. An excellent way for the novice to learn more about ROM's is to assemble a programmable ROM or PROM that uses diode memory elements. A PROM of this type consists of a grid or array of input and output wires called *lines*. A logic 1 is loaded into the ROM by bridging the intersection of an input and output line with a diode. The absence of a diode at an intersection yields a logic 0.

You can use a simple diode PROM to simulate logic gates and combinational logic networks. The first step in designing a PROM for this purpose is to write the truth table for the gate you want to simulate. The truth table for a two-input NAND gate, for example, is

	Inputs		Output
	A	B	
0	0	0	1
1	0	1	1
2	1	0	1
3	1	1	0

This truth table has four possible input combinations and only one output for each set of inputs. Therefore, our PROM will be a 4 by 1 grid of lines as shown in Fig. 1. The truth table is loaded (programmed) into the PROM by placing a diode at the intersection of the output line and the line for each input, which results in a logic-1 output. The programmed array is shown in Figure 2. Figure 3 shows how to demonstrate the operation of the PROM with the help of a

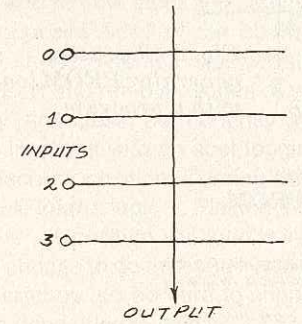


Fig. 1. PROM grid that can be used to simulate a NAND gate.

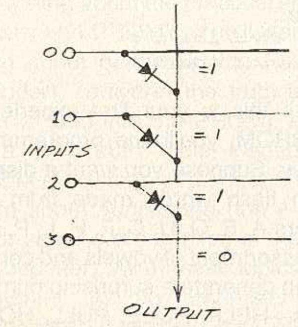


Fig. 2. 4X1 PROM programmed for 2-input NAND gate.

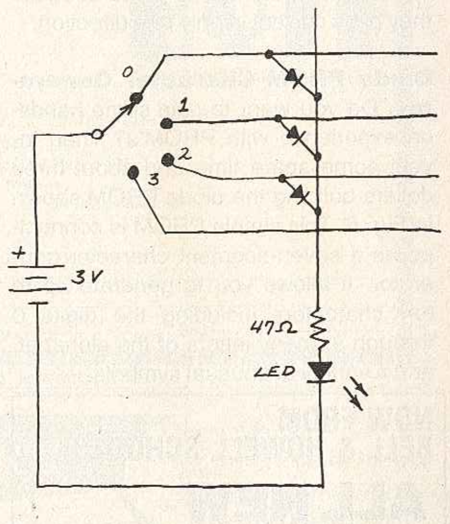
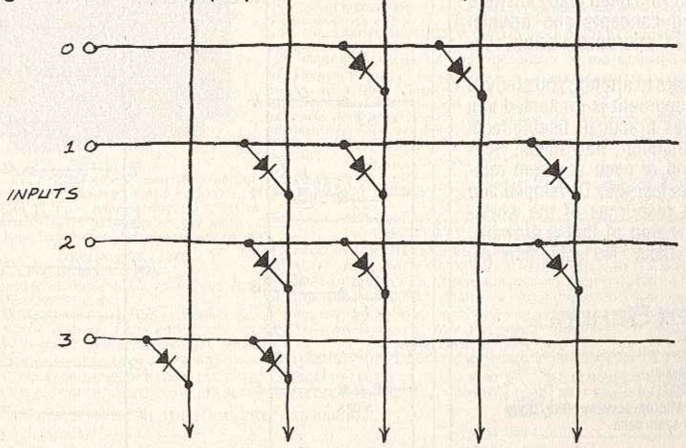


Fig. 3. Circuit to demonstrate the PROM in Fig. 2.



INPUTS	AND	OR	NAND	NOR	XOR
0-- 0 0	0	0	1	1	0
1-- 0 1	0	1	1	0	1
2-- 1 0	0	1	1	0	1
3-- 1 1	1	1	0	0	0

Fig. 4. Multiple function diode PROM.

battery, a LED, and a four-position switch.

Of course, our simple PROM version of the NAND gate is a trivial example of a read-only memory application. This is particularly true because the diodes aren't even necessary! Logic 1's can be represented simply by connecting the appropriate input lines to the output line. However, diodes are essential when the PROM becomes more sophisticated. For example, Fig. 4 shows a diode PROM that simulates the AND, OR, NAND, NOR, and EXCLUSIVE OR gates. Mass confusion would result without diodes because electrical current would thread its way through the wrong sections of the PROM via *sneak paths*. Diodes eliminate sneak paths because they pass current in only one direction.

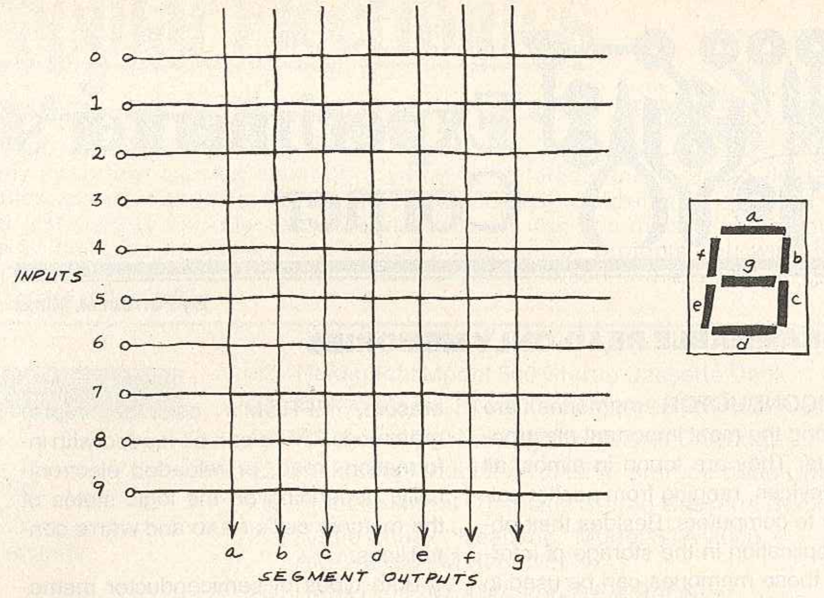


Fig. 5. Seven-segment display character generator diode PROM.

Diode PROM Character Generator. Do you want to gain some hands-on experience with PROM's? Then invest some spare time and about three dollars building the diode PROM shown in Fig. 5. This simple PROM is connected as a seven-segment character generator. It allows you to generate up to ten characters including the digits 0 through 9, many letters of the alphabet, and a variety of unusual symbols.

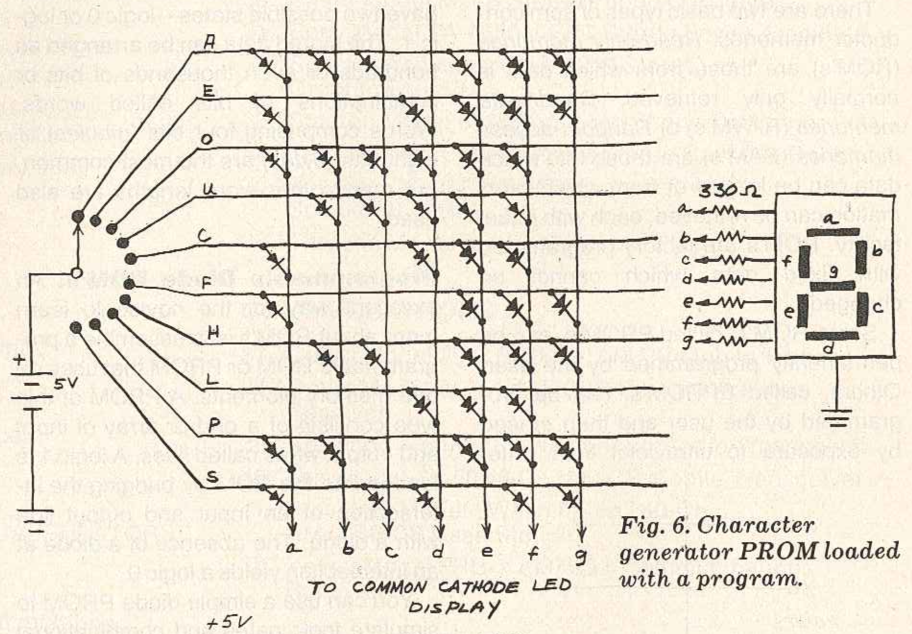


Fig. 6. Character generator PROM loaded with a program.

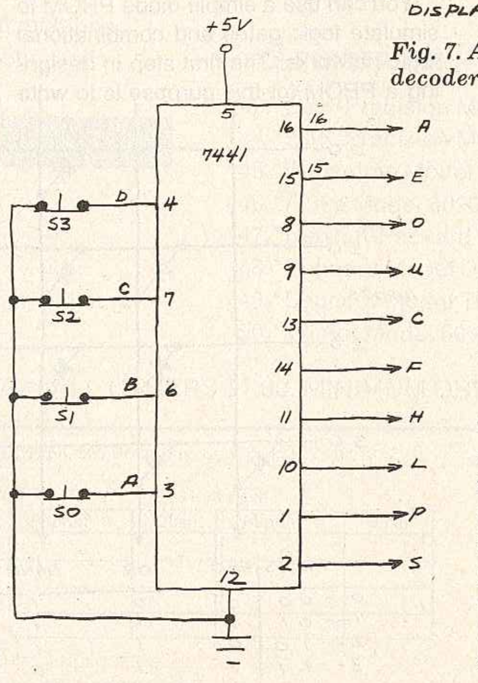


Fig. 7. Adding a decoder to the PROM.

Even if this is your first experience with a PROM, you'll find programming very easy. Suppose you want a display that can flash words made from the characters A, E, O, U, C, F, H, L, P, and S. This assortment of vowels and consonants can generate a surprising number of words—HELP, POLE, PULL, HOLE, COOL, SELL, CAP, etc. First, you write a truth table that shows the segments of the display that must be illuminated to produce each character.

Character	Segments						
	a	b	c	d	e	f	g
A	1	1	1	0	1	1	1
E	1	0	0	1	1	1	1
O	1	1	1	1	1	1	0
U	0	1	1	1	1	1	0
C	1	0	0	1	1	1	0
F	1	0	0	0	1	1	1
H	0	1	1	0	1	1	1
L	0	0	0	1	1	1	0
P	1	1	0	0	1	1	1
S	1	0	1	1	0	1	1

This prom has ten input and seven output lines, so it requires a grid of 10 by 7 wires. The PROM is programmed by inserting a diode at the intersection of each character and segment line where a logic 1 exists in the truth table (Fig. 6). Simple, isn't it? If you're not convinced, try designing the same character generator with logic gates!

We can improve the character generator PROM by replacing the manual selector switch with a BCD-to-decimal decoder. True, this adds an IC, but reduces the number of input lines from ten to four and makes it possible to interface the PROM with other circuits. Figure 7 shows how the decoder is connected to the PROM.

You can assemble a working version of this diode PROM on a perforated board. Insert flea clips at each bit position and run the input and segment lines on opposite sides of the board to prevent shorts. The flea clips will allow you to insert and remove diodes. If you want to go first class, permanently wire a diode in series with an spst toggle switch at each bit position. Turning the switch on will load a logic 1. Placing the switch in the off position will load a logic 0. If you choose to do this, you'll need seventy switches, so be sure to shop around for a good price.

In any event, I hope you'll build a working diode PROM if you're interested in learning about the practical aspects of ROM's and PROM's. You'll learn something about *hardware* (the PROM and decoder), *software* (the truth table you plan to load into the PROM), and *firmware* (the truth table loaded in the PROM in the form of diodes). You'll also learn about *addressing* (the 4-bit character select word applied to the input of the decoder). All of these topics are fundamental to an understanding of advanced digital logic devices like microprocessors, calculators, and hobby computers. ◇