VOLUME 13 NO. 5

Popular Electronics®

WORLD'S LARGEST-SELLING ELECTRONICS MAGAZINE

MAY 1978/$1

Preview of New Stereo/Hi-Fi Equipment

How to Add Triggered Sweep to Oscilloscopes

Microprocessor Microcourse, Part III

Investigating UFO's and Other Magnetic Phenomena

Kelvin

0 14024 14278
Popular Electronics

In This Issue

Micro-PROCESSOR MICROCOURSE

BY FORREST M. MIMS

PART 3.
MEMORIES, BUS ORIENTED LOGIC, AND MICROPROCESSOR ORGANIZATION.

IN PRECEDING parts of this course, we learned about binary, octal, and hexadecimal number systems. We also covered basic logic gates and combinational and sequential logic circuits.

This installment will describe semiconductor memories and show how three-state logic allows a logic circuit to transfer data to one or more other circuits over a common array of conductors called a *bus*. It will conclude with a look at the basics of microprocessor (or MPU) organization.

Memories. A microprocessor alone is merely a collection of logic circuits on a silicon chip, and must be provided with a detailed list of instructions called a *program* before it can perform useful work. The program, along with input data and even output data from the microprocessor, is stored in a memory.

Memories store information as individual bits (0's and 1's) or bit patterns (words). As we learned in Part 1, a binary word can indicate a numerical value (data), a memory address, or a computer instruction. This makes a memory device an exceptionally versatile component and an indispensable partner to the microprocessor.

Microprocessors can be used in conjunction with any kinds of memories ranging from magnetic bubbles and cores to cassette tapes and floppy disks. The two most important microprocessor memories, however, are semiconductor ROM's and RAM's.

ROM's are *r*ead-*o*nly *m*emories, since they contain information that can only be read out, and not modified or erased. RAM's are *r*ead/*w*rite *m*emories, and they generally store temporary data. The data they store can be easily modified or erased.

Both ROM's and RAM's are available as integrated circuits with dozens to thousands of individual binary storage cells printed, etched, diffused, and interconnected with an aluminum metalization pattern on a silicon chip. Thanks to the ingenious use of on-chip combinational logic decoders, it's possible to access all the storage cells in even a very large memory with relatively few input lines. A simplified view of how an *address decoder* accomplishes this is shown in Fig. 1. As you can see, simply applying the appropriate address to the memory's address lines causes the designated bit or word to appear on the output lines.

ROM's and RAM's store data as individual bits or multiple bit words (usually 4-bit nibbles or 8-bit bytes). Either way, the address decoder insures a fixed and very rapid time to access *any* bit or word in the memory. This feature is called *random access*. *Serial access* memories, like magnetic tape and high-capacity shift registers, are slower since their contents must be searched bit-by-bit to find a specified address.

ROM. The typical ROM is an array of intersecting conductors as shown in Fig. 2. A diode connecting two intersecting conductors represents a logic 1. The absence of a diode at an intersection is a logic 0. Information is stored in a ROM when it is manufactured and is therefore permanent. The information can be read out, but new information can never be written into the ROM.

ROM's can store binary data, addresses, or instructions. They can even simulate a logic circuit. Figure 3, for example, shows a diode ROM programmed with the truth table of the Exclusive-OR circuit. This circuit normally requires at least four logic gates, each containing several transistors. As you can see, the ROM version is considerably simpler.

It's easy to program a ROM to simulate virtually any combinational logic circuit, and to illustrate this, Fig. 4 shows a ROM programmed as an octal-to-binary encoder, a circuit usually designed with a network of OR gates. Designing this encoder using logic gates is both tedious and time consuming, but anyone can design a ROM encoder. All that's needed is the appropriate truth table, such as that shown below:

Octal Input								Binary Output		
0	1	2	3	4	5	6	7	2^2	2^1	2^0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Looking at Fig. 4, notice how the placement of the diodes in the ROM corresponds to the 1's in the output half of the truth table? The ROM is now programmed as an octal-to-binary encoder.

Using a ROM is as easy as programming it. Just activate the appropriate input line and the designated bit pattern appears on its output lines.

ROM's are available as reasonably priced standard parts programmed for such roles as encoders, decoders, and look-up tables of trigonometric and other mathematical functions. Semiconductor companies will also make custom ROM's upon request, a rather costly procedure unless thousands of identical ROM's are ordered. But how about a few one-of-a-kind custom ROM's for prototype or hobbyist applications?

The best solution here is the *programmable* ROM or PROM. A PROM is a ROM with diodes at *all* its storage locations. A truth table is loaded into the PROM by applying brief, high-current pulses to the inputs connected to the diodes that are *not* wanted. This vaporizes a thin layer of metalization called a *fusible link* that connects the diode to the PROM's conductors.

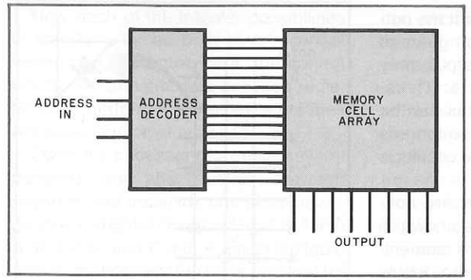


Fig. 1. How an address decoder simplifies access to a word stored in a ROM or R/WM.

PROM's, like ROM's, cannot be erased once they are programmed (though additional fusible links can be blown). A special *erasable* PROM, however, is available. It's programmed electrically and erased with ultra-violet radiation beamed through a quartz window that covers the silicon chip.

Various kinds of ROM's and PROM's can store from hundreds to thousands of bits. Since ROM's with storage capacities of from 2^8 (256) to 2^{16} (65,536) bits are the most common, ROM's (and RAM's) are often designated with a "k" factor that gives an approximation of their storage capacity—k comes from *kilo* and means 1,000. Thus a 1k memory stores 1,024 (2^{10}) bits, and a 4k memory stores 4,096 (2^{12}) bits.

Some memories store data as single bits. Therefore, a 1 × 256-bit ROM stores 256 bits of data, and an 8 × 256-bit ROM stores 256 8-bit bytes.

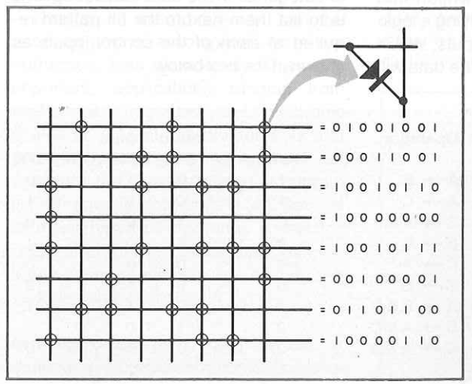


Fig. 2. A ROM is an array of intersecting conductors. When a diode connects the conductors, a logic 1 is represented.

RAM. A RAM, like a ROM, consists of an intersecting grid of conductors on a silicon chip. However flip-flops, not diodes, are placed at the intersections in the grid. Since flip flops can be made to change states, this means the data stored in a RAM can be electrically altered or erased. It also means RAM's are more complicated and therefore more expensive than ROM's.

ROM's are classified as nonvolatile memories since they store information without the presence of electrical power. RAM's are volatile. Remove the operating power from a RAM (even momentarily) and its stored information is lost since the internal flip-flop can assume either state at random.

RAM's are sometimes used to store the kind of information stored in ROM's. More frequently, however, they're used for microprocessor data and program storage, temporary data storage, and for any application that requires a quickly alterable truth table.

A simple 4-bit register can be thought of as a RAM that can store a single 4-bit word (a 1 × 4-bit RAM). But practical RAM's have substantially more data storage capacity. Today, RAM's capable of storing 16k (16,384) bits are available and 64k (65,536) bit RAM's will soon be along. Large-capacity RAM's like these can be operated in parallel to provide storage for multiple bit words.

Other Memories. Semiconductor RAM's AND ROM's are by far the most important microprocessor memories in use today. Other kinds of memories are also available, and since memories play such a vital role in the operation of a microprocessor you should at least be aware of them.

An important new semiconductor memory is the charge-coupled device (CCD). This device stores data as an electrical charge that can be moved from one memory cell to the next like a pail of water moving down a bucket brigade. The presence of a charge is logic 1 while no charge is logic 0. Since they must be accessed serially, CCD's are slower than ROM's and RAM's. However, CCD's can store more data on a silicon chip than a similar size ROM or RAM because the elaborate address decoders needed for random access aren't used. CCD's are read/write devices.

Magnetic bubble memories provide high-capacity read/write, nonvolatile data storage. Bits are stored as the presence (1) or absence (0) of microscopic magnetic cylinders called *domains* in a thin film of magnetic garnet or orthoferrite. The cylinders, which resemble bubbles when viewed on end through a microscope, can be rapidly moved along a path defined by a pattern of metalized bars, chevrons, or other shapes. The metal shapes are magnetized in different directions by a rotating magnetic field, and this causes the bubbles to move from one bar to the next.

Magnetic tape and floppy disk memories are commonly used with sophisticated microprocessor systems such as computers. There are several ways to store bits of data on magnetic tape, one of which is to encode logic 0 and 1 as two different audio frequency tones. Cassette recorders are inexpensive, readily available, and ideal for loading programs into the RAM of a microprocessor-based computer.

The floppy disk is a record-like disk of flexible plastic coated with the same magnetic substance used to make recording tape. Bits are stored as the presence or absence of magnetized spots on as many as a hundred or more concentric data tracks around the surface of the disk. The disk is spun at high speed, and a read/write head on a movable track permits access to any data track. Floppy disks provide very high capacity storage

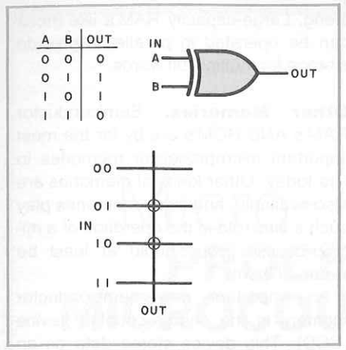


Fig. 3. At upper right is the logic symbol for Exclusive-OR, with its truth table at left. Below is the diode ROM for the same function.

with considerably faster access times than magnetic tape. But floppy disk systems are expensive; they often cost more than the computer.

Three-State Logic. Thus far we've learned something about basic logic gates, combinational and sequential logic circuits, and memories. We're almost ready to begin using these various devices as electronic building blocks to design a microprocessor. All that remains is to introduce a new kind of logic circuit called the *three-state gate*.

As you will recall from Part 1, the output of *all* the logic gates and circuits we've covered so far is various combinations of 0's and 1's. This is known as two-state logic.

A third output called the high-impedance (high-Z) state is available in three-state logic. In the high-Z state the output of a three-state gate is electronically disconnected from the gate. It's as if an on-off switch between the gate and its output line were turned off. In conventional operation, when the switch is on, 0's and 1's appear at the output.

A simple three-state buffer is shown in Fig. 5. When the *control* (or enable) input is logic 1, the buffer transmits the logic state (0 or 1) at its input to its output. When the control input is logic 0, the output enters the high-Z state.

All the basic gates are available in three-state versions. And many kinds of more advanced circuits such as flip-flops, counters, registers, and combinational networks are available with three-state outputs.

Three-state logic makes it possible to connect the output of two or more logic circuits to a common conductor called a *bus*. It's not possible to connect the outputs of two or more two-state gates to the same bus since some outputs may be logic 0 and others logic 1. Three-state logic means many gates can be connected to the same bus so long as the output of all but one of the circuits is in the high-Z state.

Several three-state buffers are connected to a common bus as shown in Fig. 6. Look at this circuit for a moment. Notice how the array of control inputs (C) allows data to be guided from any of the three inputs to either or both of the outputs. This operation is similar to that of a multiplexer, and three-state logic is sometimes used to simulate a multiplexer. More importantly, the circuit allows data to travel along the bus in either direction. That's why a three-state bus is often called *bidirectional*.

Register-to-Register Data Transfers. A typical microprocessor contains several data storage registers. Three-state logic provides an efficient way to transfer data from one of these registers to another.

Figure 7 shows three 4-bit registers connected to a common 4-conductor bus. The output of each register is connected to the bus through a 4-bit three-state buffer. This is why both the input *and* output lines from a register can be connected to the same bus.

Each register in Fig. 7 has three control inputs: Read, Write, and Clock. A logic 0 at its Read input places a register's output in the high-Z state and isolates the data stored in it from the bus—and therefore the other registers. A logic 1 at the Read input enables the three-state buffer and places the data in the register on the bus. Note that only *one* register can be in the Read mode at any one time; otherwise two or more registers will conflict with one another.

Data on the bus can be written into one or more registers by applying a logic 1 to the appropriate Write inputs. When the next clock pulse arrives, the data will

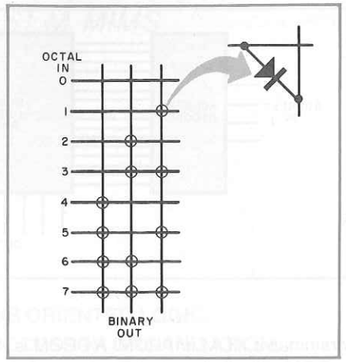


Fig. 4. A ROM programmed as an octal-to-binary encoder, a circuit usually designed with OR gates.

be written into the selected register (s).

Let's try a data transfer from register A to register C in Fig. 7. First, place A's Read input at logic 1. Then place C's Write input at logic 1. When the next clock pulse arrives, the contents of A will be copied into C. Register A will continue to retain its data, but the data in C will be lost.

You can use this simple procedure to transfer the contents of register A, B, or C to either or both of the remaining registers. What control inputs would you place at logic 1 to transfer the data word in register C to registers A and B?

The Concept of Control. We're almost ready to see how a microprocessor is put together. First, let's think about the control inputs to the three registers in Fig. 7.

There are nine authorized ways to transfer data among these registers: A into B; A into C; B into A; B into C; C into A; C into B; A into B and C; B into A and C; and C into A and B. A convenient way to categorize these data transfer options is to list them next to the bit pattern required at each of the control inputs as shown in the box below.

Operation	Control Inputs					
	A/R	A/W	B/R	B/W	C/R	C/W
A → B	1	0	0	1	0	0
A → C	1	0	0	0	0	1
B → A	0	1	1	0	0	0
B → C	0	0	1	0	0	1
C → A	0	1	0	0	1	0
C → B	0	0	0	1	1	0
A → B & C	1	0	0	1	0	1
B → A & C	0	1	1	0	0	1
C → A & B	0	1	0	1	1	0

Now each of the transfer possibilities is identified by its own binary control word. In a microprocessor, the control words that transfer data between registers and perform many other operations are called microinstructions.

Often it's necessary to make several transfers, one after another, between registers. For instance, one possible sequence using the circuit in Fig. 7 is A into B; B into C; and C into A. From the table above, the microinstructions required for these operations are:

100100
001001
010010

In a microprocessor, a sequence of microinstructions that carries out a specific series of operations like this is called a microroutine.

Microprocessors have a special control section that automatically generates the microroutines necessary to transfer data inside the microprocessor and perform many other operations as well.

The Microprocessor. The function of a conventional digital logic circuit cannot be significantly changed without extensive rewiring. The microprocessor is radically different. It can be made to perform many different functions simply by changing a sequence of binary words or instructions stored in one or more memory chips which are external to the microprocessor.

Its programmable nature makes the microprocessor essentially identical to the central processing unit of a digital computer. Add an external memory chip to store instructions and data, and a microprocessor becomes a microcomputer. Some recent microprocessors include on-chip memory for instructions and data and are called single-chip microcomputers.

Though a microprocessor can be used as part of a computer, there are numerous, less glamorous but equally important, applications ranging from traffic-light controllers and electronic scales to "smart" test instruments and pocket calculators. In many of these applications the microprocessor's program is permanently stored in a ROM. Several of these types of memories containing different programs can be used with the same microprocessor to accomplish various applications.

Microprocessor Organization. A minimal microprocessor contains a control section, a program counter that steps through the instructions and data stored in an external memory, several data and instruction registers, and an arithmetic logic unit (ALU). One way these circuits can be organized with respect to one another and to both a control and an address/data bus to form an ultra-simple microprocessor is shown in Fig. 8.

There's nothing remarkable or unusual about any of these circuits. What's important is the way they're connected to the two buses. Let's look at some of the operations performed by each of the sections in our basic microprocessor.

Control Section. The control section is the nerve center of a microprocessor. A typical microprocessor can execute perhaps fifty or more different instructions in almost any combination or sequence. (We'll look at some representative instructions later.) It's the role of the control section to fetch instructions one at a time from the ROM or RAM program memory connected to the microprocessor's address/data bus, decode and then execute them with a sequence of microinstructions; after which, it fetches the next instruction.

Program Counter. The program counter keeps track of a program that's being executed. It's simply a counter whose outputs are used as address inputs to the external memory containing the program and data being processed by the microprocessor.

The program counter and the address/data bus control how many words of external memory can be accessed by a microprocessor. Thus a 4-bit program counter can access a 16-word (2^4) memory. An 8-bit program counter can access 256 (2^8) words, and a 16-bit program counter can access 65,536 (2^{16}) words.

Normally the program counter sequences through a program one step at a time in ascending numerical order. Certain instructions, however, can load the program counter with a new data word which it will then use as the next external memory address. This allows the microprocessor to *branch* or *jump* to different parts of a program or loop through a specified section of program more than once.

Branching and looping can be unconditional or conditional. In the latter case, the program counter will receive a new address *only* if a specified condition is

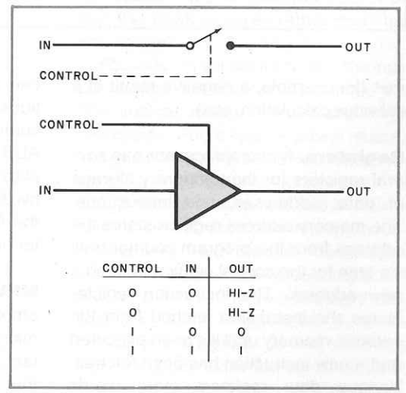
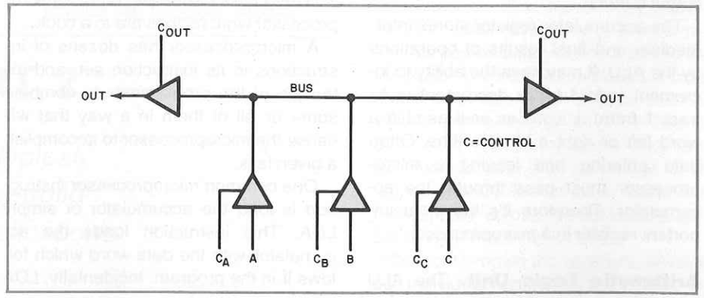


Fig. 5. Simple three-state buffer can be represented by switch as shown at top. Below is its logic symbol and truth table.

Fig. 6. Three-state buffers connected to a common bus. Since data can travel either way, the bus is called bidirectional.



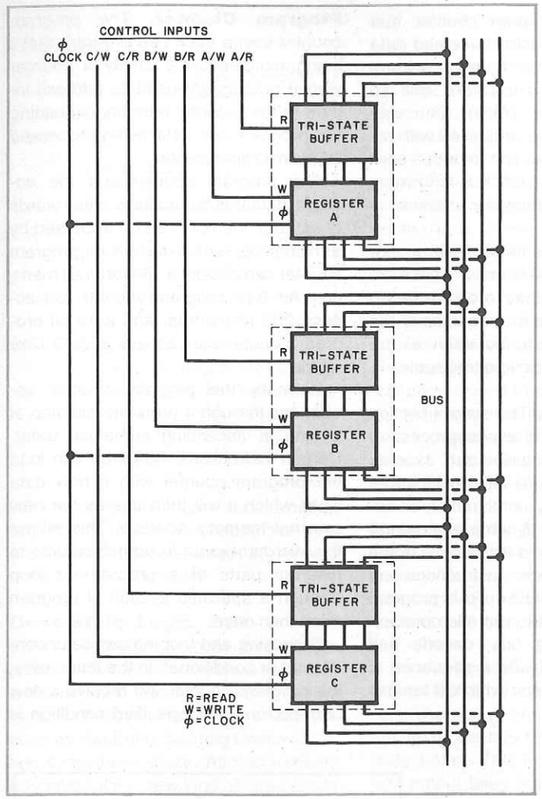


Fig. 7. Three 4-bit registers connected to a common 4-conductor bus through 4-bit three-state buffer. The three control inputs are: Read, Write, and Clock.

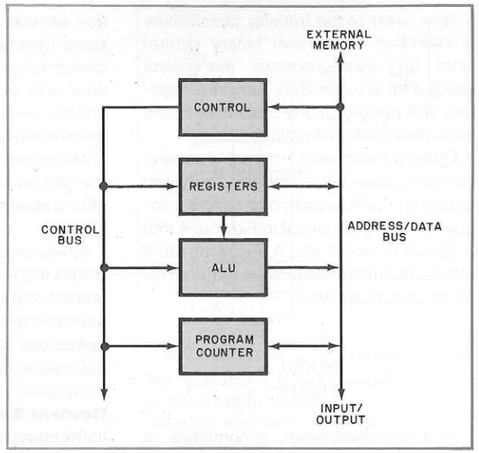


Fig. 8. Organization of a basic microprocessor. The control section is the nerve center. Counter sequences through a program one step at a time. Register is for storage, and ALU performs arithmetic or logic operations.

met (for example, a negative result of a previous calculation, etc.).

Registers. A microprocessor has several registers for the temporary storage of data, addresses, and instructions. The memory address register stores the address from the program counter until it's time for the control section to fetch a new address. The instruction register stores the instruction fetched from the external memory until it's been executed and a new instruction has been fetched. Various data registers store words awaiting further processing and act as output buffers.

The accumulator register stores intermediate and final results of operations by the ALU. It may have the ability to increment (add 1 to) or decrement (subtract 1 from) a word as well as shift a word left or right a bit at a time. Often data entering and leaving a microprocessor must pass through the accumulator. Therefore it's the most important register in a microprocessor.

Arithmetic Logic Unit. The ALU can perform arithmetic or logic operations on one or two data words. The accumulator is closely associated with the ALU. Typically, the accumulator supplies one of the words to be processed by the ALU. The result is then fed from the ALU's output back to the accumulator over the address/data bus.

MPU Programming. So far we've emphasized the *hardware* aspects of microprocessors. Hardware is important; but without software, the programs that tell a microprocessor what to do, a microprocessor is of no practical use. You might say software is to a microprocessor what recipes are to a cook.

A microprocessor has dozens of instructions in its instruction set, and it's the job of the programmer to combine some or all of them in a way that will cause the microprocessor to accomplish a given task.

One common microprocessor instruction is load the accumulator or simply LDA. This instruction loads the accumulator with the data word which follows it in the program. Incidentally, LDA is an abbreviated form of the instruction called a *mnemonic* by programmers.

Other common microprocessor instructions are JMP (jump unconditionally to the specified address); JZ (jump *only* if a zero is loaded in a special flip-flop); JP (jump *only* if the result of an operation is postive); CLA (clear the accumulator to 0); ADD (add contents of accumulator and data register and place sum in accumulator); MOV (move data from one specified register to another); RAL or RAR (rotate the bits in accumulator left or right); and HLT (halt the microprocessor).

Of course these instructions are only representative of those available with real microprocessors. Nevertheless, they provide an excellent illustration of the computer-like power of the microprocessor.

Next Month. We'll introduce PIP-2, a simple *P*rogrammable *I*nstruction *P*rocessor that demonstrates many fundamentals of microprocessor operation. We'll study PIP-2's operation in detail and learn how to program it. ◇

BY JOSEPH CARR

HOW TO DESIGN & BUILD POWER SUPPLIES

PART 2

Some typical, easy-to-build circuit applications.

LAST MONTH, we discussed the elements of power supplies, especially the low-voltage, high-current types used in microcomputers and other large-scale digital electronic projects. This month, we will give some advice on the overall circuit design of such supplies and discuss several construction projects.

Some Basics. The transformer for the supply should have a current rating higher than that required by the electronics system it is powering. Many transformers will operate excessively hot when operated at their rated current, so a safety margin is a good investment. Also, keep in mind that, when a transformer is specified, a bridge-rectified supply can safely draw only one-half of the transformer's rated secondary current without exceeding the transformer's primary VA (volts times amperes) rating. In some cases the secondary rating can be exceeded, but it is risky.

A filtered power supply will produce an output that is close to the peak voltage appearing across the transformer secondary, but the transformer ratings are likely to be in terms of the rms voltage, which is defined as 0.707E *peak*. The voltage that appears across the filter capacitor will be between 0.9E *peak* and the peak voltage, rather than the rms voltage.

You may conclude then, that the output voltage will approach 1.4 times the rms voltage rating of the transformer secondary. For a typical power supply designed for the Altair (S-100) bus systems, a 6.3-volt transformer with a bridge rectifier will generate the "nominal 8 volts" required with this approach. Alternatively, a 12.6-volt transformer and a conventional full-wave rectifier will also do the trick. These secondary voltages are popular in high-power transmitter filament supplies, so it is relatively easy to locate both 6.3- and 12.6-volt transformers having high current ratings at electronics surplus dealers, hamfests, and auctions. One transformer manufacturer, Triad, makes three transformers that power-supply builders should investigate. The F-22U is rated at 6.3 volts at 20 amperes; the F-24U at 6.3/7.5 volts at eight amperes; and the F-28U at 6.3/7.5 volts at 25 amperes. The last two models offer the advantage of a tapped primary so that either 6.3 or 7.5 volts appear across the secondary, depending on which tap is used.

Do not skimp on the rectifiers. Always use individual rectifier diodes, or molded