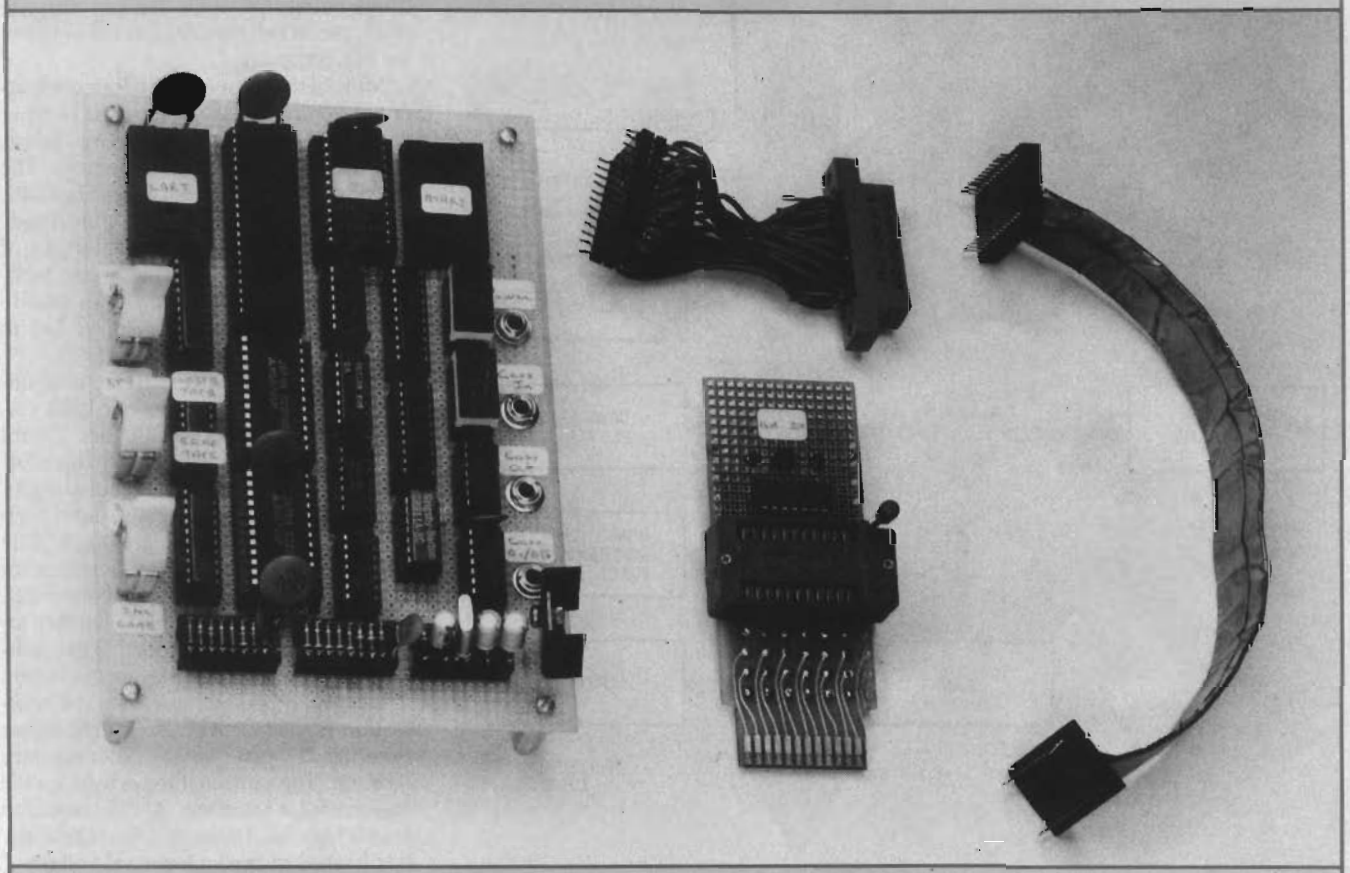
ATARI Game Recorder

GUY VACHON and DAVID A. CHAN



Store your library of Atari videogame cartridges on cassette tapes!

DID YOU EVER WISH THAT YOU COULD make copies of game cartridges for your Atari *2600*? Well, with the circuit we'll describe, you can! We'll show you how to record the contents of your cartridges on cassette tape—and how to load the game back into the *2600*. Before we get into the details of the circuit, let's review some basics.

As you probably know, a videogame is just a simplified home computer—one that has been dedicated to the specific task of playing games. Keep in mind, though, that the videogame operates much the same as any computer—the electronic circuits that make up the machine (the hardware) execute instructions that make up the game (the software).

The software is stored, of course, in the game cartridge, which consists simply of ROM (*R*ead-*O*nly *M*emory). As its name implies, you cannot change the contents of ROM—that type of memory can be read but not written to.

If a videogame is just a simple home computer, as we stated earlier, you might wonder why the programs stored in the ROM cartridges cannot be stored on magnetic tape or floppy disks like programs for other home computers. Well, they can! But videogames like the Atari *2600* lack the necessary hardware to record them. And that's what this article is all about.

The basic approach

Our approach will be to copy the contents of the ROM cartridge into RAM. (That's *R*andom-*A*ccess *M*emory, also known as read/write memory.) Once we have the game program in RAM, we can then copy it to cassette tape.

Figure 1 shows a very basic block diagram of what we need: the videogame, a game cartridge (ROM), a cassette player, RAM, and a control device, which allows us to correctly direct the flow of data. For example, the first step in copying a game cartridge is to load the contents into RAM. Then the contents of RAM is transferred to cassette tape much in the same way that many home computers save programs on cassette. (The Timex Sinclair *1000*, is one example.) When we want to play the game, we reload it from cassette to RAM. We get the *2600* to think that the RAM is just a game-cartridge ROM by setting the READ/WRITE input of the RAM to READ and connecting its other inputs and outputs to the *2600* just as if it were ROM.

We can also make tape-to-tape copies easily using that scheme. Once we load the program from cassette into RAM, we can simply dump the RAM contents to another tape.

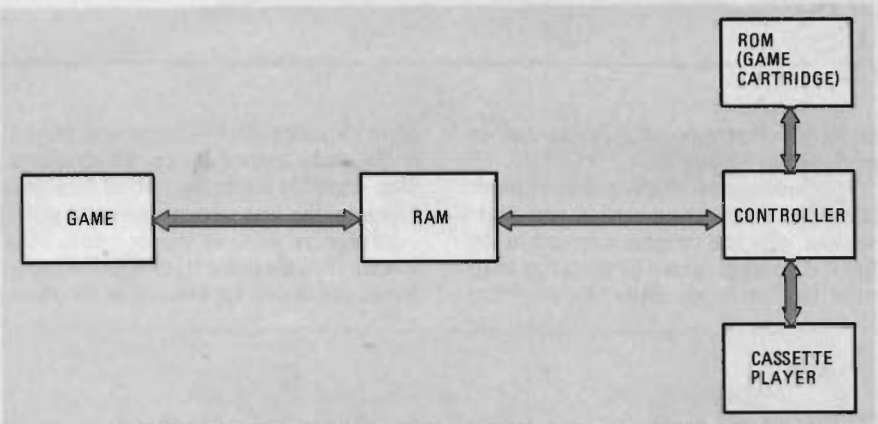


FIG. 1—A CONTROLLER IS NEEDED to properly direct the flow of data from the game cartridge ROM to the computer's RAM, from the cassette player to the RAM, etc.

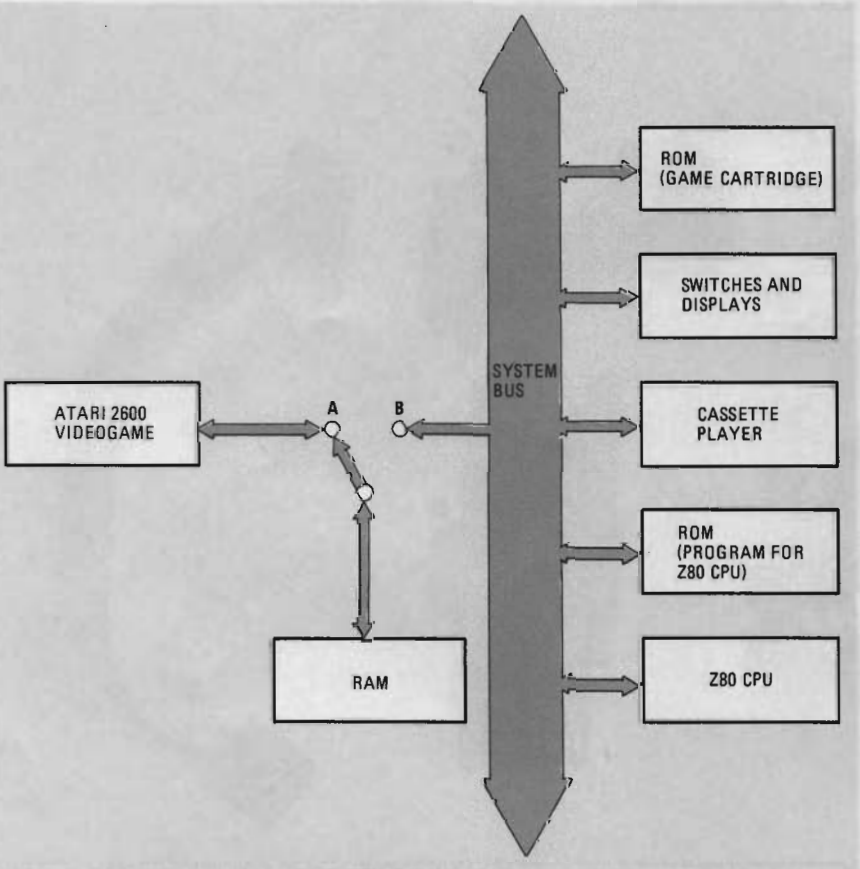


FIG. 2—A DEDICATED, SIMPLIFIED COMPUTER. This block diagram gives a basic idea of what we need to record the contents of Atari *2600* videogame cartridges.

A dedicated computer

If you're familiar with home computers—even the cheapest models that you can buy for under $50—you know that they have all the capabilities we need. We could approach the problem by modifying a computer to do exactly what we want. But that is not the way we will go. Instead, we will build our own dedicated, simplified computer.

A block diagram of the computer that we need is shown in Fig. 2. When the switch is in position "B," the *2600* is out of the picture and we're left with only our dedicated computer. It sees the game cartridge and RAM as part of its memory. It can transfer data from the ROM cartridge to the RAM, and it can store and retrieve programs from tape. The game cartridge is not the only ROM: Another block of ROM holds what can be thought of as the operating system of our computer. It contains the instructions that tell the Z80 CPU how to perform the appropriate data-transfer tasks.

Note that also "hanging from the bus" of our computer are switches and displays that are used for I/O. By setting the switches, we can give the computer certain commands. The displays let the computer tell us what it is doing.

When the switch shown in Fig. 2 is moved to position "A," the Atari *2600* videogame uses the RAM simply as if it were ROM. So, for example, after you loaded the RAM with a program contained from cassette, you would flip the switch to position "A" so that the *2600* could see it.

Game-recorder computer hardware

Let's look at the hardware that we'll use to help us record game cartridges. Figure 3 shows the schematic of the computer/recorder. As you can see, the computer is structured around the Z80 bus. Connected to the bus, directly or through buffers, are all the computer's components: the Z80 microprocessor (IC3), the RAM (IC11–IC13), and the I/O devices (S1–S5, DISP1, DISP2, cassette output, game cartridge connector, etc.). We can also see IC10, the ROM that contains the program for our computer.

Most of the components of our computer are used in the usual fashion. In other words, the ROM and RAM is used just as it is in any given home computer. The cartridge connects directly to the bus and, as far as our computer is concerned, seems to be another several kilobytes of addressable memory. That same technique of memory-mapped I/O is used to drive the seven-segment displays and to interface with the tape recorder.

Note that we do not use BCD-to-seven-segment decoders to drive our LED displays. Instead, the Z80 CPU has control over the segments and turns them on or off as needed to represent hexadecimal digits Ø–F and an error message of three horizontal bars. But we're getting a little ahead of ourselves. What is important here is that, as far as the Z80 is concerned, the displays are "write only" memory locations. Information encoded in the common seven-segment display format is sent to the display at their locations. The information is latched with the $\overline{\text{WRITE}}$ signal from the Z80 just like any other memory location. The same method is used for the tape-recorder interface. A 1 is latched to send a high-level voltage to the tape and a Ø is latched to send a low-level voltage.

The last major component of the computer is the interface to the Atari *2600*. That interface is essentially made up of IC4, IC5, and IC6—three 74LS244 octal buffers with three-state outputs. By looking at the direction of the buffers, you can see that the dedicated computer accepts addresses from the Atari game and outputs data to it. (Remember: That's just what ROM does!) The buffers are enabled by the $\overline{\text{BUS ACKNOWLEDGE}}$ signal (pin 23) from the Z80 (IC3).

When we want the *2600* to play a game, we simply close the SETUP/PLAY switch, S6, which brings the Z80's $\overline{\text{BUS REQUEST}}$ line (pin 25) low. Thus, the *2600* actually does DMA (*D*irect *M*emory *A*ccess) on the computer when requested by you through the SETUP/PLAY switch.

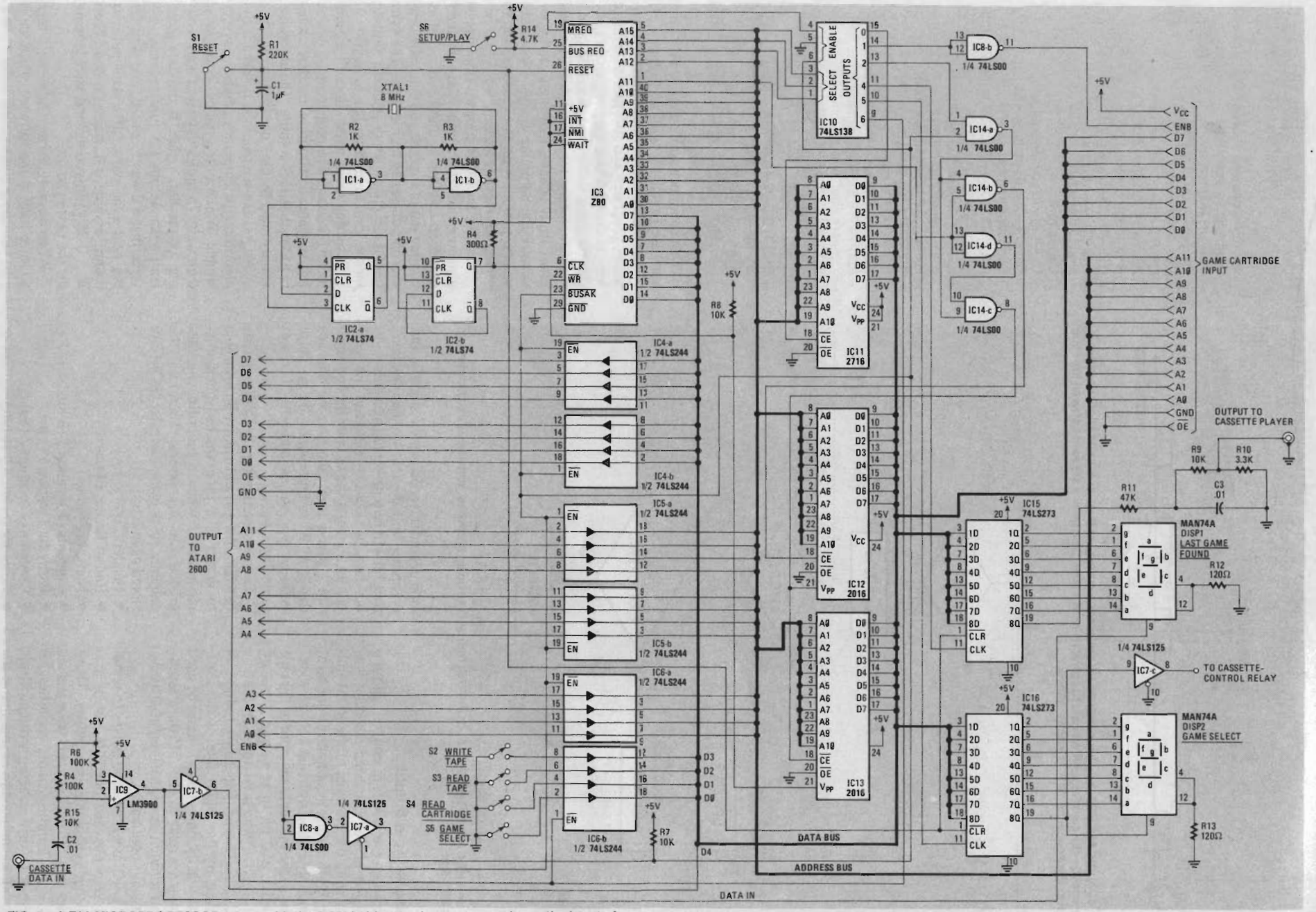


FIG. 3—A Z80 MICROPROCESSOR, along with the ROM-held operating system make up the heart of our computer/recorder. The game-cartridge ROM, LED displays, and cassette input/output are memory mapped.

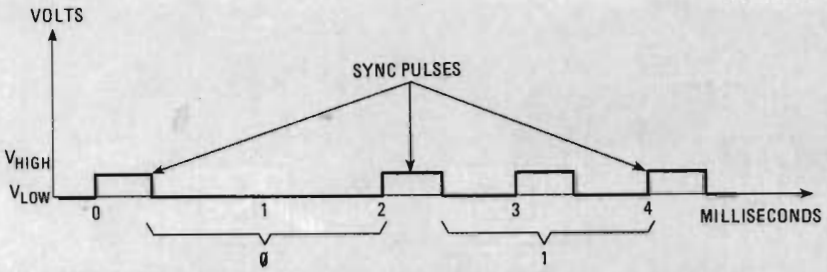


FIG. 4—SYNC PULSES, 2 milliseconds apart, are used to make sure that—even with slight changes in tape speed—the computer will be able to correctly read a tape. Data pulses are sent between the sync pulses: the sequence "Ø1" is shown above.

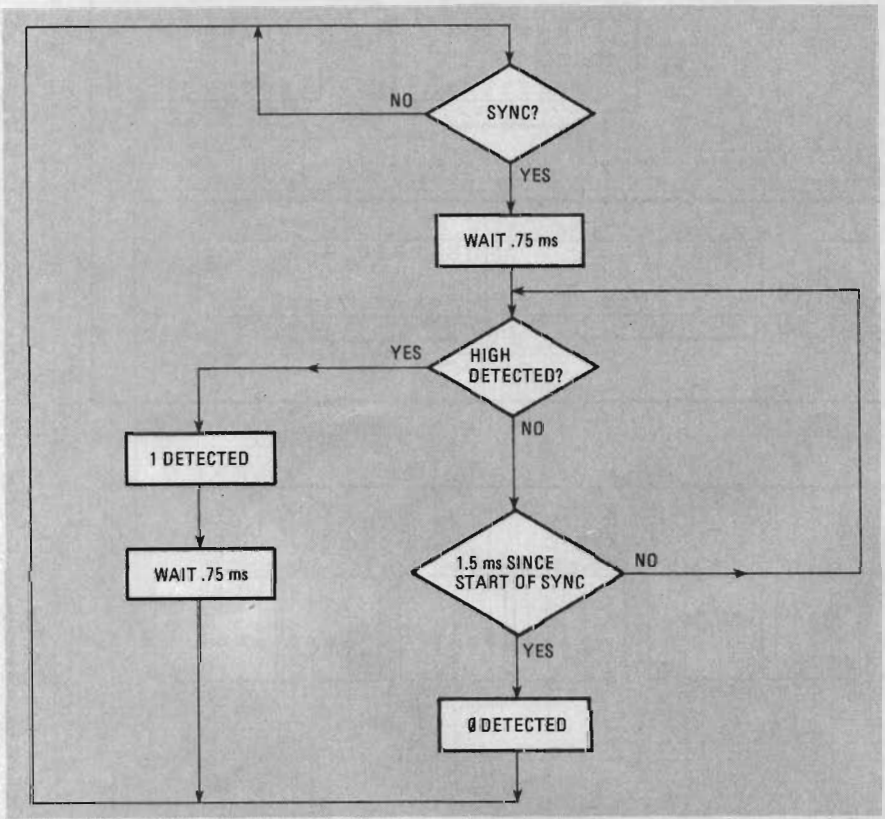


FIG. 5—THE CASSETTE-READ ALGORITHM. Rather than using hardware, software is used for timing operations.

Cassette input/output

Reading the ROM, of course, isn't the only job of our computer/recorder—we must write the contents of the ROM into tape. We'll do that by outputting one bit at a time by sending different voltage waveforms to the microphone input of the tape recorder. The bit to be output is put on data line D8, which is stored in IC16 (a 74LS273 D-type flip-flop), which is clocked by the $\overline{\text{MREQ}}$ line of the Z80. (Note that the output of IC7-c can be used to switch a relay to control your cassette player through its REMOTE input. That is, of course, optional.)

We will not only send pulses to represent zero bits and one bits—we'll also send out synchronization pulses. Those pulses are 0.25 milliseconds wide and are sent every 2 milliseconds, regardless of whether a 1 or 0 is being written. Those sync pulses are used to ensure that if the tape recorder speed varies slightly, our computer will be able to keep track. A data bit will be represented by a pulse—or the lack of a pulse—between the sync pulses. Figure 4 shows what the sequence "Ø1" would look like. As we can see there, a "Ø" is represented by no pulse between sync pulses, and a "1" is represented by a pulse.

Each instruction for the *2600* consists of 8 bits. We will add a parity bit to be able to detect if a program has been misrecorded or when a recording has degraded and has errors. Therefore, the contents of a game-cartridge ROM are stored as follows:

- Header (2000 zeros)
- End of header (4 ones)
- Name tag (4 bits)
- Contents of location 1 (8 bits)
- Parity for location 1 (1 bit)
- Contents of location 2 (8 bits)
- Parity for location 2 (1 bits)

PARTS LIST

All resistors are ¼-watt, 5%, unless otherwise specified.

R1—220,000 ohms
R2, R3—1000 ohms
R4—330 ohms
R5, R7–R9—10,000 ohms
R6, R15—100,000 ohms
R10—3300 ohms
R11—47,000 ohms
R12, R13—120 ohms
R14—4700 ohms

Capacitors

C1—1 µF, 10 volts, electrolytic
C2, C3—.01 µF, ceramic disc

Semiconductors

IC1, IC8, IC14—74LS00 quad 2-input NAND gate
IC2—74LS74 dual D-type flip-flop
IC3—Z80 microprocessor
IC4–IC6—74LS244 octal buffer
IC7—74LS125 quad bus buffer
IC9—LM3900 quad op-amp
IC10—74LS138 3-to-8 line decoder
IC11—2716 EPROM containing the computer's operating system
IC12, IC13—2016 2K × 8 static RAM
IC15, IC16—74LS273 octal D-type flip-flop
DISP1, DISP2—MAN74A
S1–S6—SPST switches
XTAL—8 MHz

That continues until all the ROM's contents are stored.

The header serves two purposes: It separates programs and provides an audible tone to detect where the program begins. (If you listen to the tape, you will hear a high pitch tone for the header. The program itself sounds like high- and low-noise.) The name tag allows you to save several programs on one cassette, and to search for those programs.

Reading the ROM contents from tape is also done one bit at a time. The computer/recorder constantly monitors what is coming out of the tape and can tell whenever the output is high or low. The sync (and data) pulses are detected by waiting for the level to go from low to high.

Detecting those pulses doesn't require much hardware. The proper timing can be implemented by counting machine cycles in a loop that does nothing. The algorithm is illustrated by the flowchart in Fig. 5.

As you can see from the flowchart, the algorithm for cassette operation is very simple: Wait for the sync pulse to appear then wait .75 millisecond and start looking for the data pulse. If the data pulse isn't seen within 1.5 milliseconds, assume that a Ø was recorded, and wait for the next sync pulse. If a data pulse *is* found, assume a 1 was recorded. Then wait 0.75 milliseconds and start looking for the next sync bit.

When we continue next time, we'll take a closer look at the software. Then we'll give you some construction hints. **R-E**