Radio-Electronics

SPECIAL SECTION YOUR OWN COMPUTER

BUILD A DELUXE VIDEO TEST GENERATOR

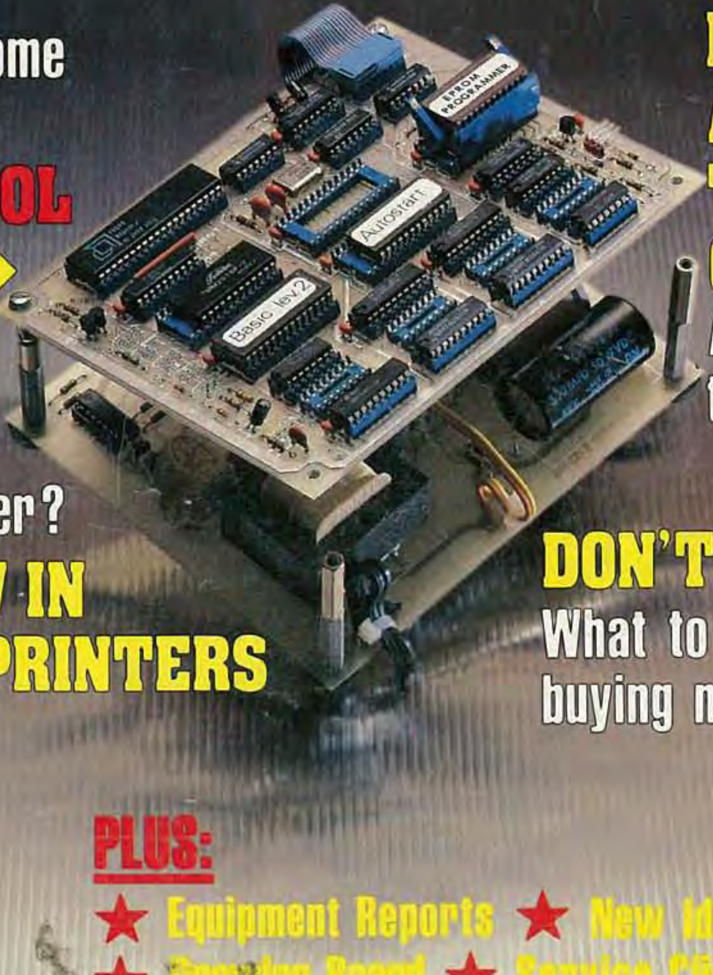
$1.75 APR. 1984
U.K. 85p
DGS

COMPUTERS - VIDEO - STEREO - TECHNOLOGY - SERVICE

Stop those annoying wrong telephone numbers!!! Build our add-on and
STOP WRONG NUMBERS

New back-to-school series starts this issue
DESIGNING WITH LINEAR IC'S

Automate your home
Build our
HOME CONTROL COMPUTER



PORTABLE AND TOTABLE COMPUTERS
Are they more than just toys?

Should I buy that new ink-jet printer?
WHAT'S NEW IN COMPUTER PRINTERS

DON'T GET STUCK
What to look for when buying new software

PLUS:
★ Equipment Reports ★ New Idea
★ Drawing Board ★ Service Clinic
★ State-Of-Solid-State

A GERNSBACK PUBLICATION

BUILD THIS

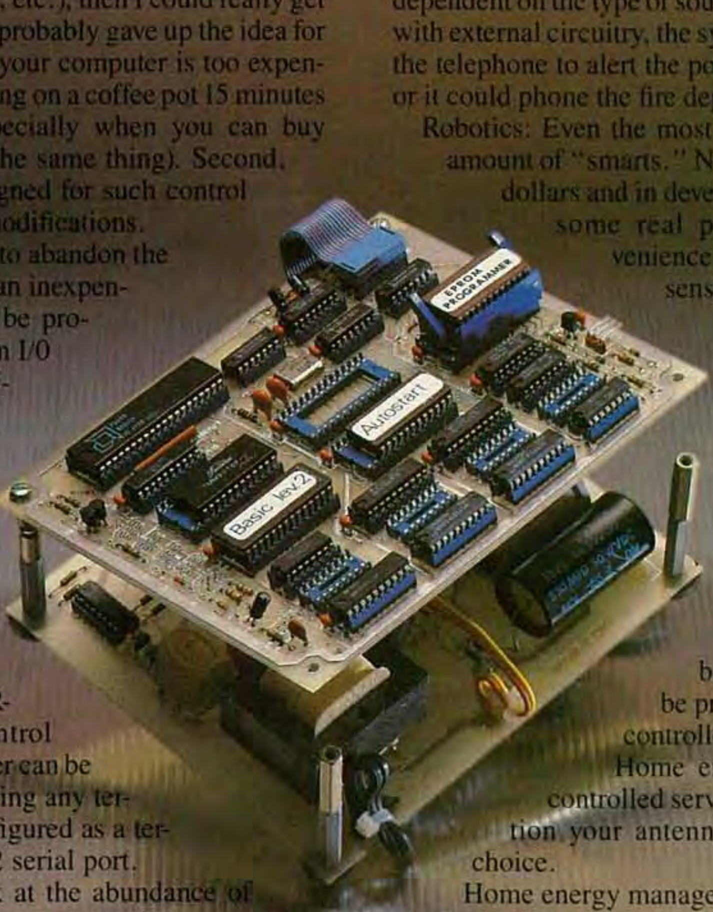
HOME CONTROL COMPUTER

If you've ever wanted to use your computer to control external appliances or systems—but didn't want to tie up your computer solely for that purpose—then this controller is for you!

STEVEN E. SARNS

HOW OFTEN HAVE YOU THOUGHT: "IF ONLY I COULD hook my computer up to that furnace (or model train, coffee pot, security system, etc.), then I could really get it to do what I want"? You probably gave up the idea for one of two reasons. First, your computer is too expensive to be relegated to turning on a coffee pot 15 minutes before you wake up (especially when you can buy inexpensive timers to do the same thing). Second, your computer is not designed for such control tasks and requires some modifications.

Those are good reasons to abandon the idea. But what if you had an inexpensive computer that could be programmed easily and had an I/O structure designed *specifically* for control applications? Then you could put some of your ideas into action. This control computer that you can build has—along with its many other I/O cababilities—the ability to control BSR-type wireless remote-control modules. And the computer can be programmed in BASIC using any terminal (or a computer configured as a terminal) that has an RS-232 serial port.



Let's take a quick look at the abundance of applications for a control computer that surround you—some of which you've probably considered and, perhaps, some you've never even thought of. We won't go into detail on how to use the controller for the following applications. Keep in mind that your programming capabilities may limit what you want to do. (You need to know at least BASIC programming to use the controller.) But we will explain how to use the control computer in enough detail so that you'll be able to tailor it to your own applications.

Security systems: A "smart" security system could arm itself in your absence. And the alarm could be dependent on the type or source of breach. For example, with external circuitry, the system could be interfaced to the telephone to alert the police if it sensed a break-in, or it could phone the fire department if it sensed a fire.

Robotics: Even the most drole robot requires some amount of "smarts." Now you can afford—both in dollars and in development time—to give him some real power. Imagine the convenience of independent drive- and sensory-systems. Imagine how much faster your development would be with a complete computer system for each function.

Model control: Imagine the complexity that you could build into a computer-controlled model-train layout, or the acrobatic maneuvers that could be programmed into your radio-controlled model airplane.

Home entertainment: A computer-controlled servo tracking system can position your antenna on the satellite of your choice.

Home energy management: (This is what the author's prototype was designed for.) The living-room thermostat could be the first target—it could automatically set back when you are at work and warm up before you return (but not on weekends). If you have electric heat, and if your electric company has peak and off-peak rates, your electric bills could be cut in half by averaging your power requirements instead of turning everything on at once. Remember—any equipment you purchase for energy management or control might be a tax credit for you. You'll have to check your own state's laws. If you're lucky, you might effectively cut the cost of your project in half!

Features of the controller

The underlying design goal for this controller was a small, inexpensive, yet powerful control computer with its own development system. That design had to achieve a successful balance: We wanted the board size and the cost to be kept down, yet we still wanted to include as many features as we could. However, if we tried to give the board too many bells and whistles, the resulting high cost would limit its practical (economical) applications. On the other hand, a cost-conservative approach would result in a board with only limited applications. Of course, we searched for a happy medium.

The result was a rather small board (about 5½ × 6½ inches) that has enough power to do its job at a competitive cost—take a look at the controller's features:

•A total of 46 inputs and outputs: Seven high-current, individually addressable outputs; 7 individually addressable inputs; 2 eight-bit input ports, and 2 eight-bit output ports.

•An RS-232 serial port: a terminal can be used for program entry.

•An eight-channel, 8-bit analog-to-digital converter with provisions for digital-to-analog conversion.

•A choice of two operating systems: BASIC or Forth. The high-level language will cut your programming time by 90%.

•An on board EPROM programmer makes a permanent copy of your RAM-based program.

•A real-time clock for time-of-day functions.

•Auto start of ROM-based programs.

•BSR-type remote controller is on a companion power-supply board. That system communicates to readily available receiver modules that you simply plug into the AC lines. All of the hassles of stringing wires from the controller to the control point are eliminated.

A closer look

Now that we have an idea of the basic features of the controller, let's discuss some of the theory behind it. Unfortunately, we won't be able to cover all of the points that we just mentioned this month—they will be discussed in upcoming installments of this article.

We'll start by describing the microprocessor and the support circuitry required to test, debug and program the basic system. The design will be discussed in enough detail so that even those who are not familiar with microprocessor design techniques will be able to get an overview of the process. The control computer's schematic is shown in Fig. 1.

The microprocessor selected for this project is the Intel 8088—the 16-bit microprocessor that forms the heart of IBM's personal computer. (Anyone who owns an IBM PC has a complete set of development tools for this board.) The 8088 can be thought of as being made up of two units. The first is the BIU or *B*us *I*nterface *U*nit, which prefetches instructions while the rest of microprocessor (the EU—*E*xecution *U*nit) is working on the current instruction. Besides speeding execution, that has an even more fortunate (economical) effect: Memory IC's with access times as slow as 400 ns will work on the board.

The bus structure

The microprocessor is connected to the memory and I/0 through the *data bus*, the *address bus* and the *control bus*. Those busses are shown in the computer's schematic in Fig. 1.

The data bus is the group of eight lines (D0–D7) over which data can be transferred between the microprocessor and any memory or I/O (*I*nput/*O*utput) device.

The address bus is made up of 20 lines, some of which are time multiplexed. Don't worry about now—we'll get to it shortly. You should know, however, that the microprocessor uses the address bus to select the desired memory address or I/O device to send data to or receive data from. That address is represented by the unique combination of address-line states.

We will be concerned with three lines of the control bus. The $\overline{\text{READ}}$ ($\overline{\text{RD}}$) and $\overline{\text{WRITE}}$ ($\overline{\text{WR}}$) lines determine whether the data is to be transferred to (read by) or from (written by) the microprocessor on the bi-directional data lines. The third line, IO/$\overline{\text{M}}$, is used to distinguish between a memory access or an I/O access.

The 8088 can address 1 megabyte (2^{20}) of memory with its 20 address lines. If we use only the lower 16 lines, we can address 64K. The 8088 combines the data bus and the lower eight address lines into what is called a time-multiplexed bus. That was done so that the 8088's package could be kept to 40 pins. The first design question is to decide whether to demultiplex the data and address bus or use it as is. Intel (and others) supports the multiplexed bus with an extensive range of products. Leaving the bus multiplexed will result in a smaller board that is easy to lay out. (That is one of our design goals.) However, because of the popularity of the non-multiplexed bus, peripheral IC's designed for it are more available and are less expensive. Because another of our design goals is to design a low-cost board, we must stick to popular components—or at least ones that we expect to become popular. Fortunately, demultiplexing the bus is an easy matter; it requires only a set of latches. A 74LS373 octal latch (IC15) is used. It is enabled by the ALE (*A*ddress *L*atch *E*nable) pin of the microprocessor (pin 25). Now we have 8 high-order address lines, A15–A8, (IC18 pins 39, 2–8), and 8 low-order address lines coming from the latch outputs, A7–A0, (IC15 pins 7, 12, 6, 15, 5, 16, 2, 19). We also have 8 bi-directional data lines, AD7–AD0) from the input side of the latch (IC18 pins 9–16). Note that the data lines still contain the multiplexed address-information. They will contain data at the time the appropriate control line ($\overline{\text{RD}}$ or $\overline{\text{WR}}$) is active.

The memory field must be divided into appropriate banks (or peripheral IC's). We must make sure that only a single peripheral IC can be active at any one time. If more than one device attempts to place data on the bi-directional data bus simultaneously, a condition known as *bus contention* rises. The result of bus contention is an undefined bus state and, consequently, undefined operation. Thus the output of our memory decoder will be a one-of-N type—only one output will be active at any one time. Each of those outputs will be connected to the chip-enable ($\overline{\text{CE}}$) input of a peripheral IC.

The selection of the size and type of memory is heavily influenced by our need to convert our finished program into ROM. If we can simply remove a RAM IC and replace it with a pin-compatible ROM, we will have a compact yet flexible board. The 2016 2K×8 RAM and the 2716 2K × 8 EPROM are pin-compatible, so they will be used. We also need memory space for ROM-ed development tools that can be used during the program testing, and an empty socket for the blank EPROM to be programmed.

Throughout the remaining description of the board, the highest order address lines (A19–A16) will be ignored. The address lines A15–A8 will be called the high-order address lines. The most significant high order address line (A15) can be used to chip-select the system ROM. Address lines A11 and A12 are used as the inputs to a 74LS139 one-of-4 decoder (IC16), which will be used to chip-select the other memory IC's. We have mapped our system ROM and 4 memory sockets uniquely into the 64K address space. Table 1 shows a memory map of our system. Note that A15 is inverted by IC19 to select the system ROM and that the one-of-4 decoder (IC16) is qualified by IC17-d: when A15 and the IO/$\overline{\text{M}}$ lines are low, a memory-field operation is indicated.

TABLE 1—MEMORY MAP

Address (hex)	IC/Function
0000–07FF	1C12/RAM
0800–0FFF	IC10/RAM
1000–17FF	IC13/ROM
1800–1FFF	IC14/ROM and EPROM programming socket
8000–FFFF	IC9/System ROM

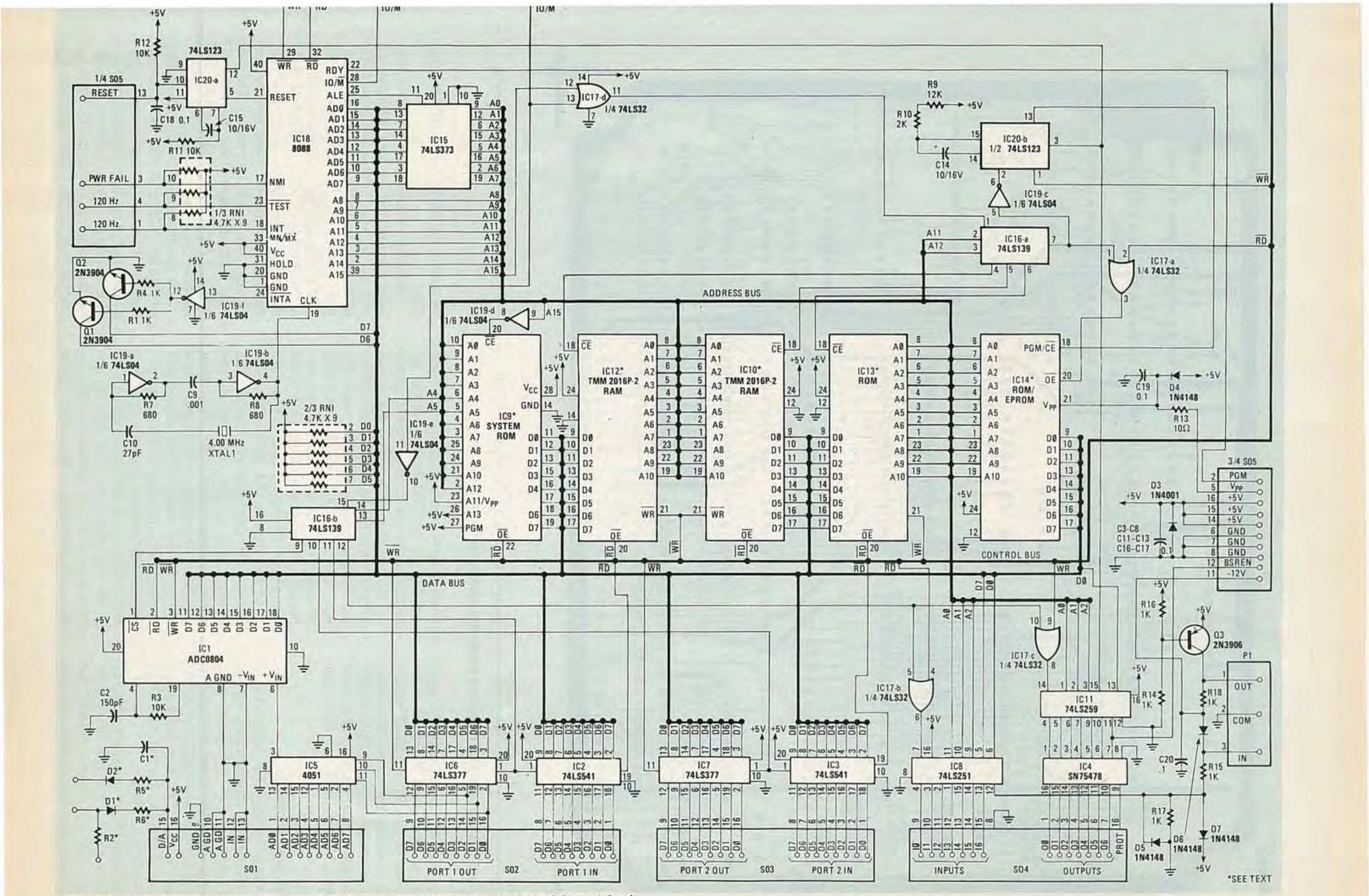


FIG. 1—THE CONTROL COMPUTER SCHEMATIC. Note that IC9 is the system memory. It is contained in ROM and is available from the supplier indicated in the Parts List. Then pin numbers shown for IC9 are *socket* pins. If a 2732 or 2764 is used, then pin 23 should be connected to A11 and not 5 volts. Also, pin 26 should be connected to A13, and not 5 volts. In both cases, you have to cut a trace and add a wire jumper.

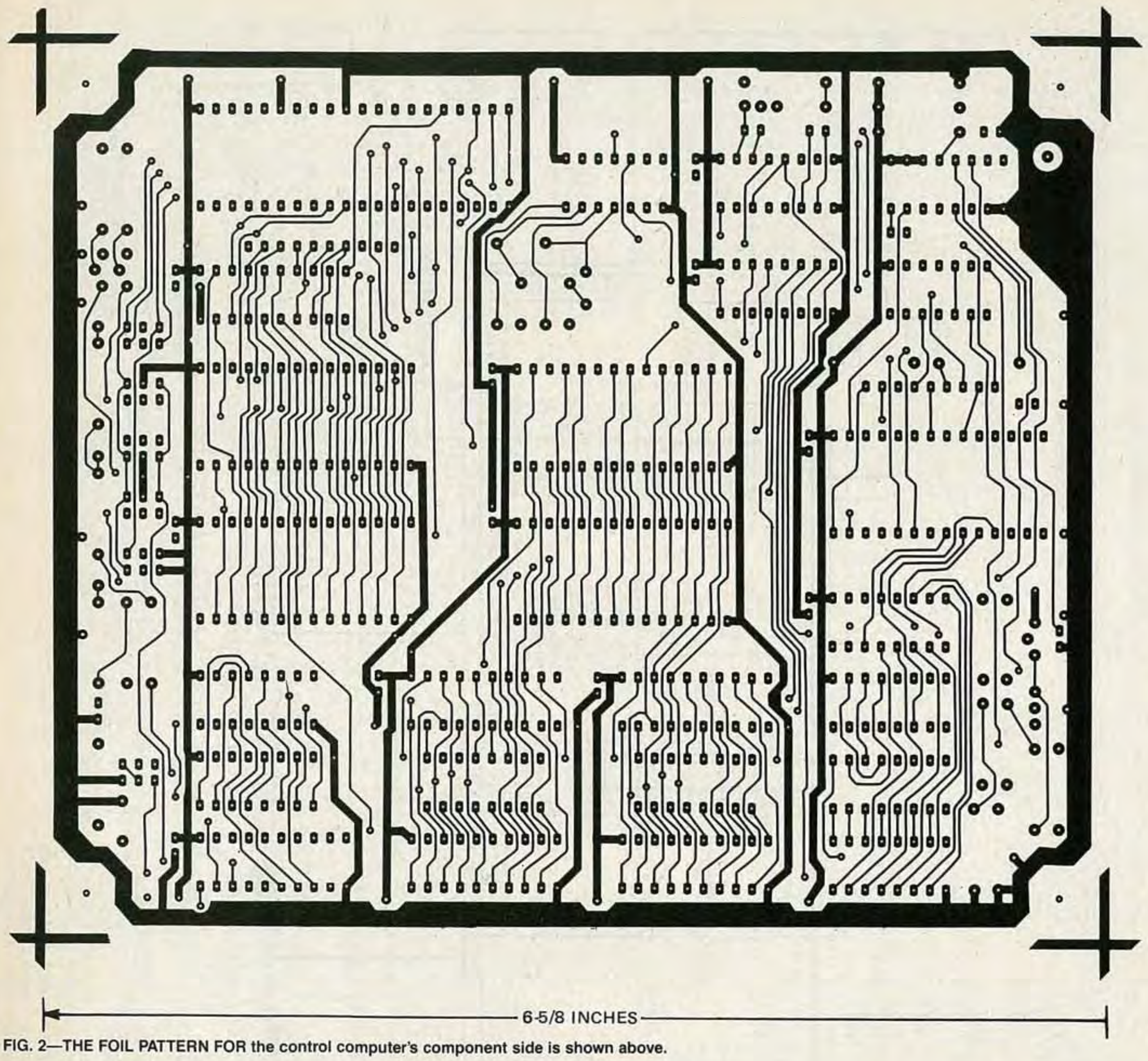


FIG. 2—THE FOIL PATTERN FOR the control computer's component side is shown above.

When the 8088 is reset, it will begin execution at address FFFFØH. That means our system ROM should occupy the highest memory position. (Remember that we're not decoding the highest-order address lines, so that when the 8088 looks at address FFFFØ, it will see the system ROM.

EPROM programming

Programming the 2716 EPROM is a simple matter. The programming voltage may be applied as a DC voltage to pin 21, the V_{PP} pin. The address and data must be stabilized for 50 milliseconds, and during that time, a single TTL-level pulse is applied to the $\overline{CE}$/PGM pin (pin 18). (It is pulsed from low to high.)

Rather than the more usual method of surrounding the EPROM being programmed with a bi-directional latch, the address and data information will be taken directly from stabilized address and data busses. The disadvantage of our approach is that the microprocessor cannot be doing anything else during that 50-ms interval when the busses are stabilized—including timing the 50 milliseconds. We will have to use a hardware timer. The advantages of our method are fewer components, software simplicity, and a small board size. (Figures 2 and 3 show full-sized foil patterns for the double-sided printed-circuit computer board. The power supply for the computer is contained on a second board. We'll talk about that board in a future installment of this article.) In fact, the EPROM programmer is completely invisible to the software—the EPROM appears to the operator as a very slow-to-write RAM-like device.

We will operate the 2716 from the microprocessor bus by externally qualifying $\overline{RD}$ and $\overline{CS}$ with IC17-a. The output of that OR gate is applied to the 2716 $\overline{OE}$ pin (pin 20).

The 2716's $\overline{CE}$/PGM line (pin 18) will be normally low and go high whenever $\overline{CS}$ and $\overline{WR}$ are true. Those two signals (at IC20, pins 1 and 2) are the trigger conditions for the 74LS123 50-ms one shot whose output (IC20, pin 4) is connected to the 8088 RDY input and the 2716 $\overline{CS}$/PGM input. Whenever the RDY line is low, the 8088 inserts wait states into the current microprocessor instruction. The wait state holds the current bus status (for 50 ms, as determined by C14 and R10) until the RDY line is returned high.

The specifications for the programming voltage are +25 ±.5 volts DC at 30 milliamps. We have found that reliable programming can be achieved with a programming voltage as low as 22 volts. (A programming voltage of 24 volts has worked well for us.) The maximum V_{PP}

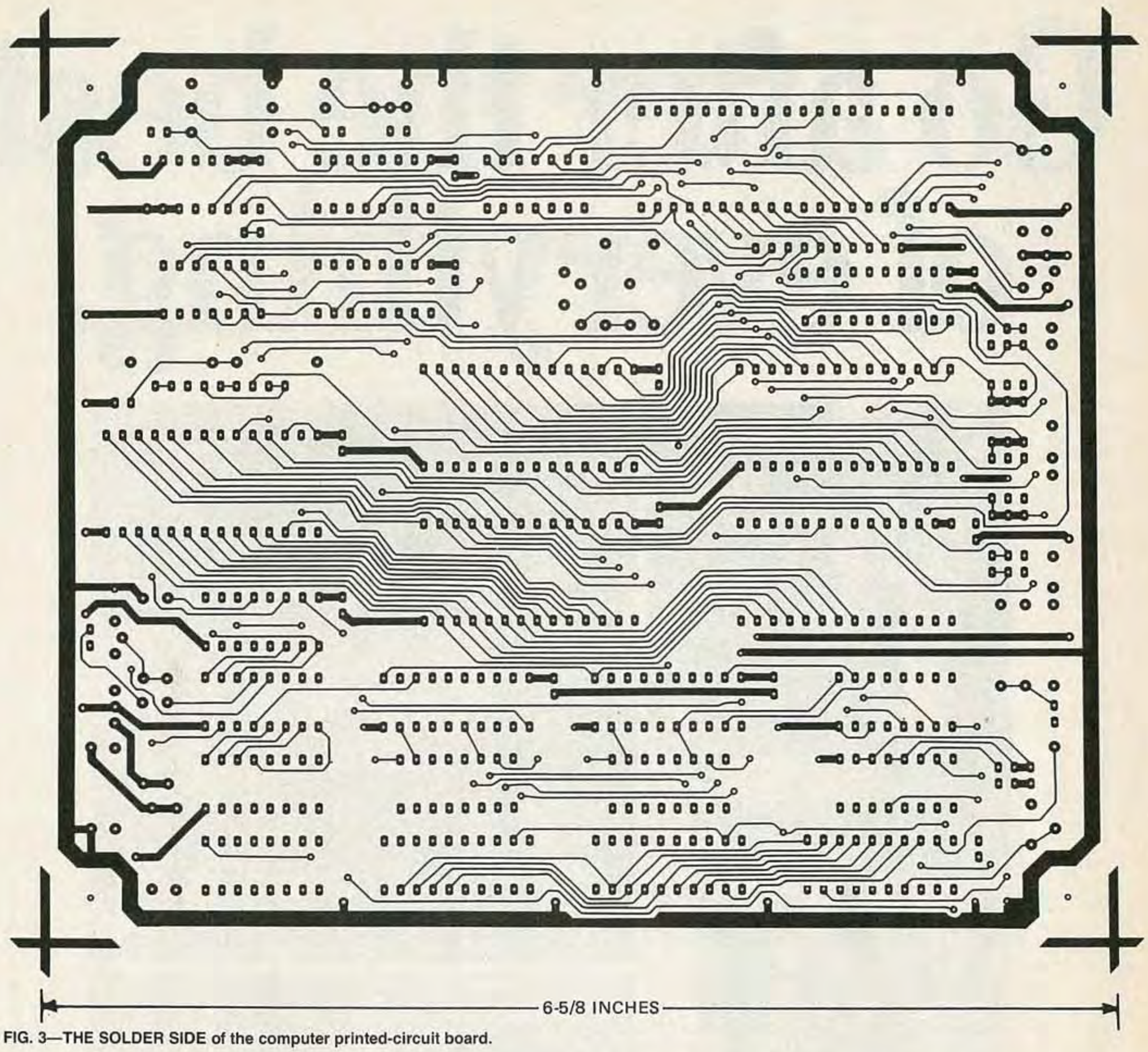


FIG. 3—THE SOLDER SIDE of the computer printed-circuit board.

voltage specification is an extremely important one to follow. We met up with disaster with a programming voltage of 26 volts. If the V_{pp} voltage is exceeded *even for a few nanoseconds,* the EPROM will fry! That means that the V_{pp} supply must not overshoot during turn-on or turn-off. Because we need to control the ramp-on and -off characteristics we will use a power supply design that always switches the supply on and off for each byte programmed. (The switching supply makes the control computer more versatile because it allows you to use other EPROM's—for example the 2732—that require a switched V_{pp} supply. The 2716 doesn't require a switched supply.) That switching programming supply is located on the separate power supply board. We'll discuss that circuit and its construction in a future part of this article.

Programming an EPROM involves only setting appropriate locations to "ø." A fully erased EPROM has all of its memory locations filled with 1's.

The time will come—either because of a programming mistake or because you no longer need a particular program—that you'll want to erase your programmed EPROM. You can erase the EPROM's by exposing them to ultraviolet light. Direct sunlight will erase an EPROM in about a week. Room-level fluorescent light will erase an EPROM in about 3 years. (Although that's not an efficient erasure method, it is still a good idea to cover the window with a label to block out room/sun light.) A commercial EPROM eraser is simply a source of ultraviolet (UV) light that irradiates the EPROM. You can make one yourself with a General Electric G15TB 18-inch germicidal bulb in a conventional fluorescent-lamp holder. **Do not look into the bulb when it is on.** The light is much more intense than it appears and quickly damages the eyes. Place the EPROMS to be erased within one inch of the bulb and leave it on for 10–15 minutes. That should change all the bits in the EPROM to 1's.

The microprocessor requires a system clock signal, which we obtain with a conventional TTL-type crystal oscillator (IC19, XTAL1, and other associated components). The frequency of the clock is 4.00 MHz, even though the 8088 could run at 5 MHz. The 33% duty-cycle constraint on the clock signal would require either a special clock generator or additional TTL chips.

A reset pulse is also required by the 8088, which we generate by using one-half of a 74LS123 (IC20-a) during power-up or whenever the reset line is grounded and released.

continued on page 94

CONTROL COMPUTER

continued from page 51

The I/O

The RS-232 port is really part of I/O system, which we'll be discussing in a future installment of this article. However, we must introduce it now because it is required for any operator I/O. It also must be used to confirm proper operation of the basic system.

The RS-232 port can be implemented in two distinctly different manners. The first and easiest is to use a UART (*U*niversal *A*synchronous *R*eceiver *T*ransmitter) IC. The UART constantly waits for an incoming character, receives it according to whatever protocol has been programmed, and stores the character in a holding register until the microprocessor requires it. Transmission is simply a matter of writing the character to the UART's transmit register.

The other alternative—the one that we'll use—is to use one input and one output of the board to send and receive the serial data. That method requires the microprocessor to assume control of the entire process of data reception and transmission. However, it eliminates the requirement for a large and expensive component. The principal limitation of this approach is that the microprocessor must know when to expect a character and it must be executing its input routine before the character is sent to the board. That's not a problem for most applications. In the case of BASIC, it means that all console commands and program INPUT statements are handled easily. However, there can be no INKEY$ statement.

The eighth output and eighth input of the bit-addressable I/O port will be used to implement the RS-232 port. The requirement for a negative voltage supply for the RS-232 link can be avoided with a bit of foxy circuit design. The incoming RS-232 signal is normally at the "mark" or negative level. Every character transmission ends with a stop bit designed to return the line to the negative level. We can generate our level from the incoming RS-232 line. That approach works fine for cables of less than 10 feet long. If longer cables are used, a separate negative supply between −5 and −12 volts DC should be used. Such a supply is available from the power-supply board, which—as we mentioned previously— we'll discuss in a future installment.

When examining serial asynchronous waveforms, remember that a logic one is less than −3 volts DC and a logic zero is greater than +3 volts DC. The line is held in the mark or one state when not active. The transmission always begins with a start bit = 0 and ends with a stop bit = 1. The stop bit returns the line to the mark state.

Full duplex serial operation that means separate wires carry data to and from the control computer (and terminal). Pin 2 of the RS-232 connector is transmitted data *to* the computer. Pin 3 is received data *from* the computer. At the terminal, those two pins are reversed so that pin 2 is an output and pin 3 an input. If you are using a personal computer as a terminal, you must determine if the RS-232 port of your computer is configured as a terminal or as a computer. Whatever the case may be, connect the board's input to the terminal's output and vice versa.

PARTS LIST—COMPUTER BOARD

All resistors ¼-watt, 5% unless otherwise noted
R1,R4,R14–R18—1000 ohms
R2,R5,R6—user-determined. To be discussed next month
R3,R11,R12—10,000 ohms
R7—680 ohms
R8—390 ohms
R9—12,000 ohms
R10—2,000 ohms
RN1—4.7K × 9 resistor network
Capacitors
C1—user-determined. To be discussed next month
C2—150 pF, ceramic disc
C3–C8,C11–C13,C16–C20—0.1 μF, ceramic disc
C9—0.001 μF
C10—27 pF
C14,C15—10μF, 16 volts, electrolytic
Semiconductors
IC1—ADC0804 A/D converter (National)
IC2,IC3—74LS541 octal buffer and line driver
IC4—SN75478 seven high-current darlington drivers (TI, also Sprague ULN-2003, Motorola MC1413)
IC5—4051 8-input analog multiplexer
IC6,IC7—74LS377 octal latch
IC8—74LS251 8-input digital multiplexer
IC9—System ROM. 2716, 2732, or 2764. 450 ns maximum access time.
IC10,IC12—TMM 2016P-2 (Toshiba or similar) 2K × 8 static RAM, 450 ns.
IC13—Programmed EPROM (2716)
IC14—EPROM to be programmed
IC11—74LS259 8-bit addressable latch
IC15—74LS373 octal latch
IC16—74LS139 dual 2-to-4 line decoder/multiplexer
IC17—74LS32 quad OR gate
IC18—8088 microprocessor
IC19—74LS04 hex inverter
IC20—74LS123 dual one-shot
Q1,Q2—2N3904
Q3—2N3906
D1,D2, D4–D8—1N4148
D3—1N4001
Miscellaneous: IC sockets, PC board, mounting hardware, etc.

The following are available from Vesta Technology, Inc., 2849 W. 35th Ave., Denver, CO 80211: KIT 1—Kit of all parts needed to control 7 LS-TTL outputs, monitor 7 inputs, program EPROM's, RS-232 serial port, and 2K RAM (does not include operating system—see below), $99.95; KIT 2—Kit of all parts for full-capacity I/O and 4K RAM (does not include operating system—see below), 169.95; Operating systems contained in ROM: BASIC I operating system, $12.95; BASIC II operating system, $29.95; Forth operating system, $79.95; Assembled, tested, and burned-in control computer with BASIC II operating system, $279; RS-232 cable, $24.95; 2716 EPROM, $6.95. Add $6 for shipping, handling, and insurance.

PARTS LIST—POWER-SUPPLY/ BSR LINK BOARD

All resistors ¼ watt, 5% unless otherwise noted
R1,R19—200 ohms
R2,R7,R8,R20—100 ohms
R3,R4,R11,R12,R15—1000 ohms
R6,R10—4700 ohms
R9—1 ohm (a jumper works fine)
R13—10,000 ohms trimmer potentiometer
R14—15,000 ohms
R16—220 ohms
R17—10,000 ohms
R18—470 ohms
Capacitors
C1,C2—0.1 μF, ceramic disc
C3,C5—0.01 μF, ceramic disc
C4,C6—0.047 μF, ceramic disc
C7—10 μF, 25 volts, tantalum
C8—150 pF, ceramic disc
C9—1 μF, 16 volts, electrolytic
C10—0.14 to 0.47 μF, 200 volts, electrolytic
C11—3300 μF, 16 volts, electrolytic
C12—500 μF, 50 volts, electrolytic
Semiconductors
IC1—ULN2003 darlington array (Sprague)
IC2—TL497 switching regulator
IC3—74LS00 quad NAND gate
IC4—LM340-5 +5-volt regulator
IC5—LM320-5 −5-volt regulator
Q1—2N3904
Q2—2N3906
D1,D3,D5,D6,D10—1N4001
D2,D4,D7–D9—1N4148
T1—11Z2100 1:1:1 pulse transformer (Sprague)
T2—16 volts, center tapped, 0.4 amps. (Signal ST-4-16 or similar)
S1—normally open momentary pushbutton switch
Miscellaneous:line cord, printed-circuit board, IC sockets, heat sink for regulator, mounting hardware, etc.

The following are available from Vesta Technology, Inc., 2849 W. 35th Ave., Denver, CO 80211: Power-supply/BSR-link kit, including all components, $59.95; Assembled, tested, and burned in power supply, $109. Add $6 for shipping, handling and insurance.

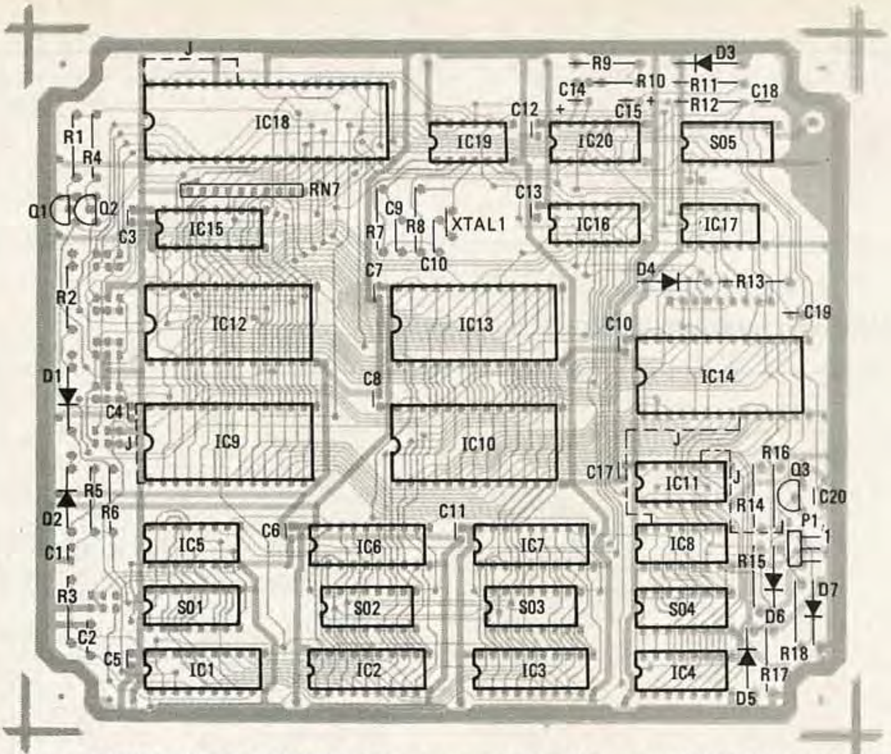


FIG. 4—PARTS-PLACEMENT DIAGRAM for the computer board. The dashed lines represent foil-side jumpers. There are many "unused" pads; they are for future expansion and experimentation. For example, note that a 28-pin socket is provided for the 24-pin 2716 (IC14). That allows larger EPROM's to be programmed. Simply plug the 2716 (or other 24-pin device) into the socket leaving pins 1 and 2 of the socket unoccupied.

Building the control computer

The actual assembly of the assembly of the controller requires no special techniques. You can use a printed-circuit board (we showed the foil patterns for the double-sided board in Figs. 2 and 3) or you can wire-wrap the project. You are probably better off with the PC board. Troubleshooting will be easier, and assembly will be quicker—about an hour, assuming you already have the board. (If you are not able to make your own board, see the Parts List for a supplier). Wire-wrapping the project will take about 16 hours. One note on wire-wrapping: Do not give into the temptation of wire-wrapping directly to the leads of the discrete components. The integrity of a wire-wrap joint depends on the square corners of the posts.

If you have the printed-circuit board, simply follow the parts-placement diagram in Fig. 4. Be careful not to create solder bridges, and when you are finished, wash the board in flux solvent. That will remove any flux residues left on the board. Don't forget that we've showed the foil and parts-placement diagrams for only the computer board. The power supply and BSR-type controller will be located on a separate board of the same size. We'll discuss that single-sided board next time, and we'll give you some more hints on the control computer's construction.

Troubleshooting

After you install all the parts, carefully inspect your board. You should check that all IC's are in the correct sockets and that they are correctly oriented. Transistors and diodes should also be checked. Then measure the power-supply voltages. Although we have not yet discussed the board for the power supply and BSR-type controller, you can test the computer board if you have a +5-volt DC regulated supply.

Regardless of what supply you use, ensure that it is the correct voltage, and be sure to orient it correctly. Reversing the power-supply polarity is like reversing an IC, only it's more efficient—it will burn out *all* of the IC's. The BASIC operating system should be inserted into the system ROM socket (IC9). The first memory socket (IC12) should contain a 2016 RAM IC. Do not insert additional memory yet. Also, unneeded I/O should be removed. Get your terminal to the proper protocol (4800 baud, no parity, 8 data bits, full duplex, caps lock on) and connect it to the board. Apply +5 volts DC. If the BASIC prompt (>) appears, we can assume that the basic microprocessor circuits are correct. (If you use the BASIC II operating system, then you do not have to set your terminal to 4800 baud. Simply hit the space bar within 7 seconds after the board is turned on or reset.)

If the BASIC prompt does not appear, then we have some troubleshooting to do. Unfortunately, though, we've run out of room to talk about it this month. But it's the first thing we'll deal with next time.

Along with the troubleshooting hints, we'll look at the power supply, remote controller, and also the I/O capabilities of the compuier **R-E**