

# BUILD THIS



## Computer Controlled IC Tester

FLOYD L. OATS

*Use your computer to test digital IC's and eliminate your troubleshooting guesswork!*

HOW DO YOU TEST IC'S? IF YOU'RE A DIGITAL enthusiast on a tight budget, then you probably go about it the low-cost way: You set up your breadboard, a handful of jumpers, a power supply, and a measuring instrument such as a logic probe or oscilloscope. Manual operation of such a crude setup can quickly become tedious—especially when the circuit being tested has multiple edge-sensitive inputs.

There's a bigger problem with such a setup, however: Since you must perform the tests one gate at a time or one latch at a time, you can expect to find only the most obvious failures. The more subtle types of failures—such as interaction between separate circuits within the same package—are best found by some kind of automated tester.

If you have a computer, you can easily build an automated tester for digital IC's. The tester we'll describe permits automatic testing of digital integrated circuits with up to sixteen pins. The host may be an eight-bit or a sixteen-bit microcomputer

and can have either separate input and output data busses or a bi-directional data bus. While the author's prototype was built as a tester for TTL IC's, the principles that we'll discuss apply equally well to other popular logic families.

Although the tester we'll describe was designed for use with an S-100 system, you should be able to adapt the circuit for virtually any computer. We should note here that using the tester will require some programming proficiency. While the article will describe what the software has to do, actual program instructions are not included.

Figure 1 shows the basic idea behind the IC testing scheme. A host computer is used as a stimulus generator: It sends data patterns, called stimuli, to the circuit under test. Each stimulus is sixteen bits wide—one bit is assigned to each pin of the test circuit. After the stimulus is sent, the host will accept a response (also sixteen bits wide) from the circuit under test and will perform an analysis of that re-

sponse. Response analysis begins with a comparison between the actual response and some known expected response. If they are equal, the host continues with the next stimulus. What happens if they're not equal depends on the software that is used by the host system.

### Stimulus generation

How do we determine the set of stimuli that we want the host system to generate? We must ensure that the set of stimuli for a given type of integrated circuit is both valid and complete. At first glance, you might think that a set of stimuli that presents every possible bit combination to the input pins of the circuit under test would meet those requirements. Those stimuli could be created very easily simply by causing the host to "count through" the input pins. Consider the simple hex inverter with six input pins and six output pins. Using bit-masking techniques, the host could "count through" those six input bits from zero to sixty-three and thereby



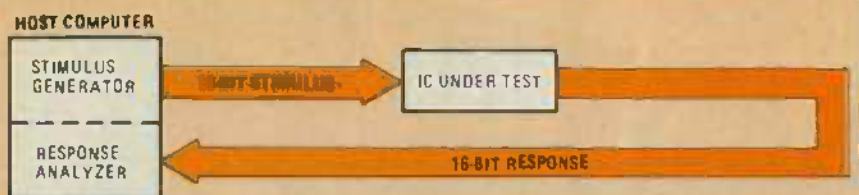


FIG. 1—THE IC TESTER SENDS a 16-bit stimulus to the IC to be tested, and then examines the response.

present sixty-four stimuli for the hex inverter.

That counting scheme does generate a complete and valid set of stimuli for IC's that contain nothing but raw gating. But it falls short with IC's that have edge-sensitive inputs. For example, counting through the input pins of an ordinary flip-flop allows the clock pin to change simultaneously with the data pin(s) at some counts. Those counts lead to an indeterminate response of the flip-flop, rendering the set of counting stimuli invalid.

Let's assume further that the clock pin is assigned to, say, the fourth significant bit of the stimulus generator and a data pin is assigned to the second or third significant bit. Due to the nature of the up-count function, the clock pin will change only when the data pin is high—the stimuli set is incomplete as well as invalid. In that particular case, the problem is partially solved by counting downward. But with multiple edge-sensitive inputs, the problem quickly becomes unmanageable. As we'll see, we'll need to use methods other than the counting stimuli to overcome those difficulties.

### Response analysis

Once we present the stimulus to an IC, we have to look at the response. Response analysis includes all 16 pins of the test circuit. You might wonder why we want to look at all 16 pins—after all, we don't

have to look at the input pins to determine the output response. There are two reasons why we do look at all pins. First, it is simpler from the software standpoint—it eliminates the extra steps required to exclude certain pins from analysis. The second and better reason is that if we examine the input pins of the test circuit, we can detect failures associated with input loading problems. Also, we can verify that the stimulus was actually presented to the test circuit. In other words, we can give the tester self-diagnostic capability.

If the host is going to analyze the response of an IC, it has to know what type of response to expect. The only way it can know what to expect is if we teach it. A set of good responses can be generated by presenting stimuli to an IC that is known to function properly. The responses along with the stimuli used to produce them can then be stored on disk or tape for future use.

### Functional description

Figure 2 is a block diagram of the tester; it should help to make our description of the schematic (Fig. 3) clearer. As shown in both figures, data or stimuli from the host system comes into a set of latches made up of IC1, IC2, IC4, and IC5. (In eight-bit systems, eight data lines and the lower eight address lines are used to feed the stimulus latches.) As shown in Fig. 3, the STIMULUS LOAD signal (which enables

the latches) is connected to all four latch IC's. Therefore, it's essential that all sixteen stimulus bits be presented to the latches simultaneously. The use of eight address lines to carry stimulus information causes the tester to occupy 256 memory locations. (In 8-bit systems, that corresponds to 256 bytes. In 16-bit system, those 256 locations correspond to 512 bytes.)

The outputs of the four 4-bit latches are fed to sixteen isolation switches, S1-S16. Only the switches that feed the input pins of the IC to be tested will be closed. Switches connected to the output pins of the test circuit will be open to prevent interference between stimulus latches and output signals. Switches connected to the power supply pins will be open to prevent possible damage to stimulus latches.

The lines from the isolation switches go to test socket SO1. As you can see from the schematic, the isolation-switch numbers correspond to the test-circuit pin that they feed. For example, switch S11 feeds pin 11 of test socket SO1. That makes determining the switch positions a little easier.

The power supply consists of a five-volt regulator (IC11), filter capacitor C1, and a power-supply-source socket, SO3. (The regulator can be omitted from many systems that contain central five-volt power supplies.) Notice that the voltage is wired into SO3 according to standard convention:  $V_{CC}$  on pin 14, and ground on pin 7.

Power is supplied to the IC under test by a pair of movable jumpers connected from the power-supply-source socket, SO3, to the power-supply-select socket, SO2 (which is wired in parallel with the test socket). If, for example, you wanted to test a 7475, you would connect a jumper from pin 14 of SO3 to pin 5 of SO2. You would also jumper pin 7 of SO3 to pin 12 of SO2.

Response data are accepted by the host

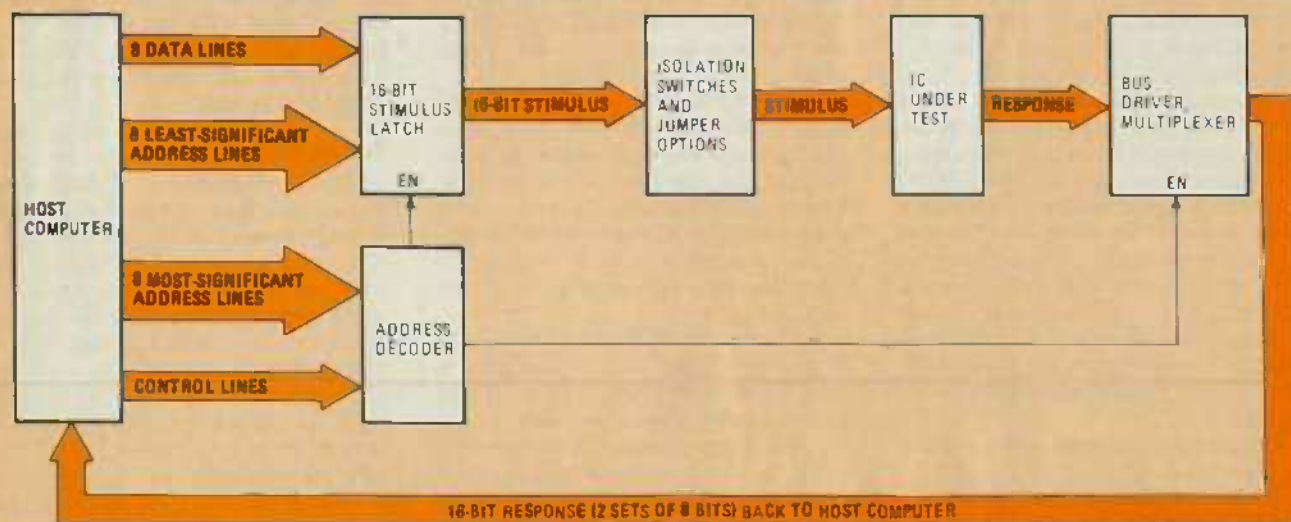
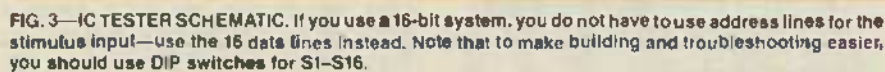


FIG. 2—BLOCK DIAGRAM OF THE IC TESTER shows how the host computer's data, address, and control lines are used to control the tester.



Control logic, made up of IC8, IC9, and IC10 develops signals that control the loading of the stimulus latches and the gating of response data to the host. The

SEPTEMBER 1984



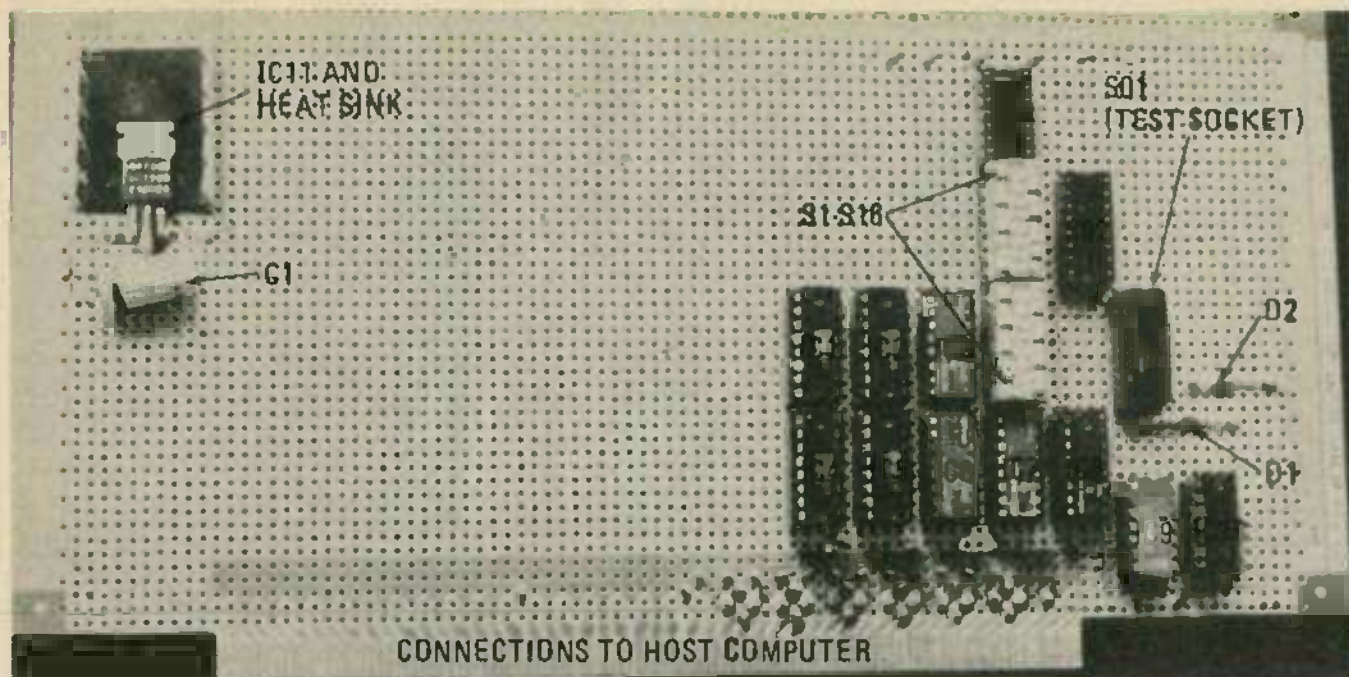


FIG. 4—THE AUTHOR'S PROTOTYPE was wire-wrapped on an S-100 prototyping board. Parts placement and construction technique are not critical.

the tester. In this case, the address range chosen is C000H through CFFFH. (Note that an "H" indicates that a number is written in hexadecimal form.) While there are two inverters shown in the schematic, any number of inverters may be used. They are placed so that, when the address lines are designating the memory area assigned to the tester, all inputs to IC9 are high. That, presents a high to IC8 pin 6, partially enabling it. Pins 4 and 5, fed by inverted I/O-status lines, must be low to complete the enabling of IC8. Those lines will be low when the address lines carry a memory address as opposed to an input/output port address (the latter being indicated by a high on IC6 pin 12 or 14). In systems where all input/output areas are memory-mapped and there are no separate input/output functions, pins 4 and 5 of IC8 should be directly grounded.

When IC8 is enabled, it will decode the inputs on pins 1, 2, and 3 into an active-low signal on one of eight output pins. When SMEMR, the memory-read status signal, is high (and MWRT, the memory-write enable signal, is low), pin 12 or 13 will be low, depending on the state of A0. If A0 is low, IC8 pin 13 will gate pins 1 through 8 of the test circuit to the response lines. When A0 is high, IC8 pin 12 will gate pins 9 through 16 of the test circuit to the response lines.

To write to the stimulus latches, MWRT is brought high while SMEMR is low. That causes either pin 10 or 11 of IC8 to go low. Since A0 is part of the stimulus information, the STIMULUS LOAD signal must be insensitive to the state of A0. This is accomplished by the diodes D1 and D2 connected to IC8 pins 10 and 11—they permit the STIMULUS LOAD signal to be generated

by MWRT without regard to the condition of A0.

#### Construction

You can build the tester using either printed-circuit or wire-wrap techniques. Component layout is not critical and there is no need to be concerned with special considerations such as lead lengths and extensive decoupling. The author's prototype was wire-wrapped on an S-100 plugboard. It is recommended that you use a board that is configured for your computer system.

We recommend that you use DIP switches for the isolation switches: It keeps the board much neater and, thus, easier to troubleshoot. Be sure to label the switch numbers. Mount the DIP switch packages and all of the IC's in wire-wrap sockets. The power-supply jumper sockets SO2 and SO3 should also be empty wire-wrap sockets. For the test socket, SO1, you might want to use a "zero insertion force" type socket. Do not substitute

LS-type IC's for IC3, IC6, and IC7.

If your 8-bit system has separate input and output data busses, the data-output lines should be fed to the stimulus latches while the data-input lines should be fed from the response output of the tester. In sixteen-bit systems, the stimulus latches may be loaded from the sixteen data lines rather than using the lower eight address lines as part of the stimulus. And, of course, the response drivers may be tied to the sixteen data lines instead of being multiplexed into two sets of eight.

If your host computer system uses a bidirectional data bus instead of separate input and output busses, then the data lines (but, of course, not the address lines) into the stimulus latches and the data lines from the response drivers may be connected to the unified bus.

Using a host computer with sixteen-bit organization simplifies the control circuit by permitting the A0 input (IC8 pin 1) to be grounded (since its only purpose is to split the sixteen response lines into two groups.) Pins 12 and 10 of IC8 could be left open, and the output from pin 13 would gate all sixteen response drivers.

The control circuit shown was designed for an eight bit host with sixteen address bits. In hosts which have fewer than sixteen address bits, there will be fewer than eight lines feeding into IC9. Consider a system with only fourteen address bits. The eight least-significant address bits will be assigned as stimulus bits, leaving only six bits for the address recognition function. The two unused pins of IC9 may be connected together and tied to +5 volts through a one-kilohm resistor.

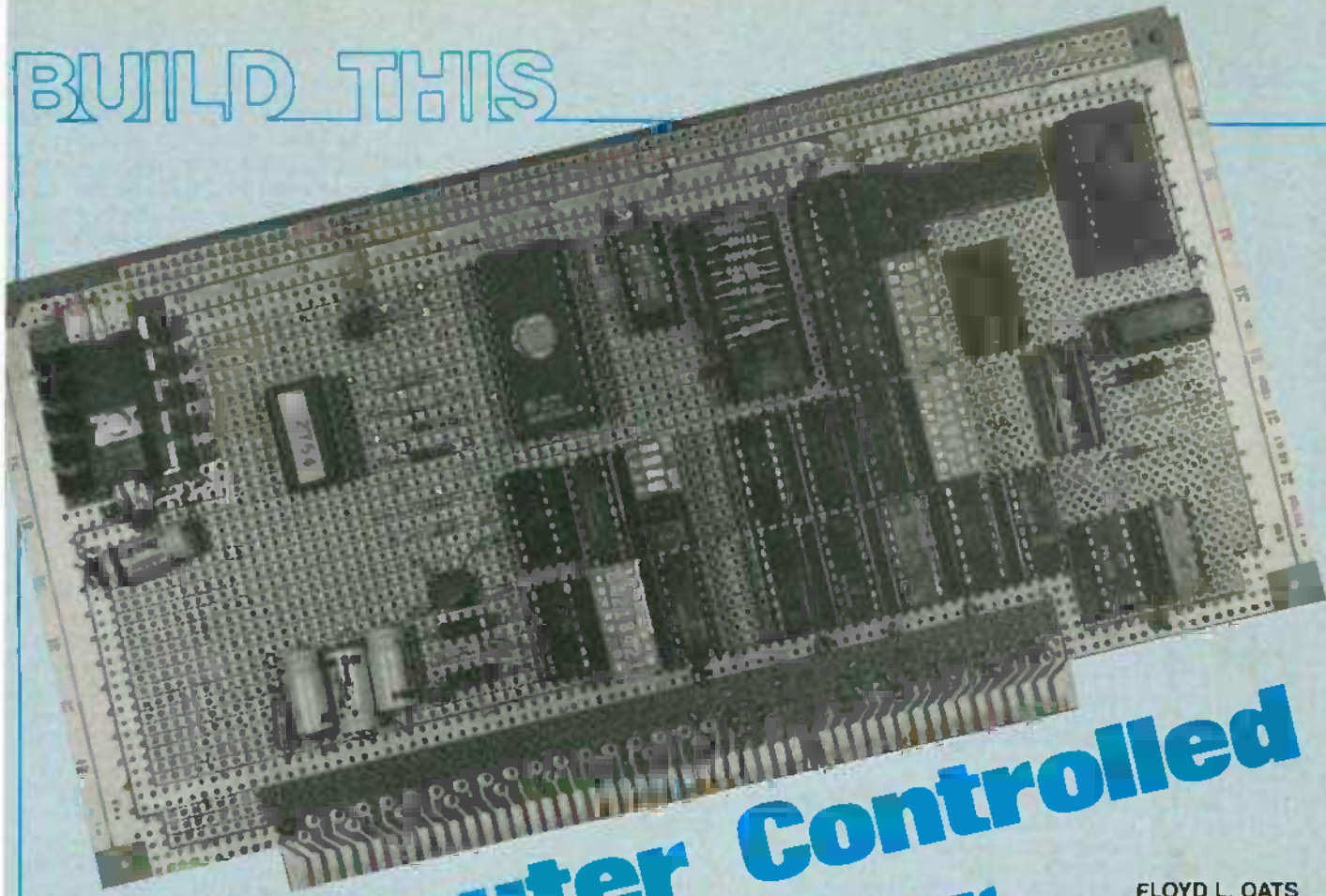
When we continue, we'll begin by testing the tester. R-E

#### PARTS LIST

IC1, IC2, IC4, IC5—74LS75 4-bit bistable latch  
IC3, IC6, IC7—74367 hex bus driver  
IC8—74LS138 3-to-8 line decoder/multiplexer  
IC9—74LS30 8-input positive NAND gate  
IC10—74LS04 hex inverter  
D1, D2—1N914  
R1, R2—2000 ohms  
S1-S16 SPST switch (DIP switches are recommended)  
C1—20μF, 10 volts, electrolytic  
Miscellaneous: IC sockets, plugboard, wire-wrap wire, hardware, power-supply jumper wires, +5-volt DC power source, etc.



# BUILD THIS



## Computer Controlled IC Tester

FLOYD L. OATS

*This month we'll test the IC tester and then we'll look at some options that you can add.*

**Part 2** LAST MONTH, WE DESCRIBED the IC-tester circuit and the theory behind how it works. Now it's time to make sure that it *does* work. After we do that, we'll look at some options that you can add to your tester. For example, we'll build a low-budget logic analyzer—it works by expanding your oscilloscope display to 16 traces.

### Testing the tester

Most of the components and wiring are located in the data paths so the inherent self-diagnostic feature of the tester can be utilized as a debugging aid for the finished project. After the device is built and connected to the host computer, preliminary testing can begin.

For the purpose of discussion, we will assume that the device has been mapped into memory addresses CF00H through CFFFH (as it is shown in the schematic). **Note:** An "H" indicates that a number is written in hexadecimal form.

Open all the isolation switches (SI-S16) and make sure that the test socket is empty and that there are no power-supply jumpers between SO3 and SO2. Read the sixteen response bits by performing memory reads on addresses CF00H and CF01H. Both of those addresses should return FFH, indicating that the response is equal to sixteen "1" bits. Next, close all the isolation switches and write sixteen zeros in the stimulus latches by writing 00 into address CF00H. Read the response as before and look for sixteen zero bits. Now write all ones in the stimulus latches by writing FFH into memory address CFFFH and check for a response of all ones, as before. At this point, the tester is in a configuration where all stimulus information should be exactly duplicated on the response lines.

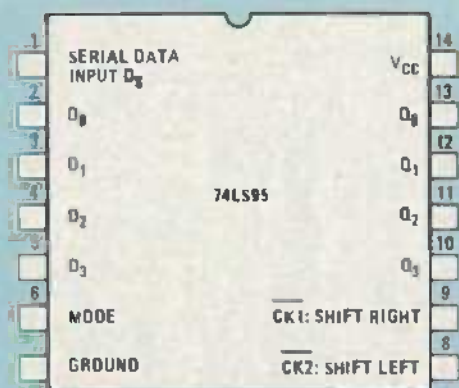
The next test will require a short program to send counting stimuli to the tester. After each stimulus is sent to the latches, the response is accepted and compared with the stimulus for equality. If

single-bit failures are observed during the test, the components and wiring associated with that particular bit should be checked carefully. If the bit patterns don't change or if they don't even resemble the correct patterns, the control circuitry might be at fault. Any discrepancies noted up to this point must be repaired before proceeding with further tests.

The final series of tests will verify the wiring of the isolation switches. Starting with all switches closed, open one switch and send a stimulus of all zeros. The response should be all zeros except for a single "1" bit, which should correspond to the open switch. Now close the switch just tested and open the next switch, then perform a similar stimulus/response test on this switch. Continue in this fashion until all sixteen switches have been tested. The final test is started with all switches open and is similar to the previous test in that one switch at a time is tested. This time the switches will be closed one at a time. Send test patterns of all zeros and

OCTOBER 1984





OPERATION	MODE	INPUTS				OUTPUTS			
		CK1	CK2	D <sub>3</sub>	D <sub>0</sub>	Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
PARALLEL LOAD	H	X	↓	X	1	L	L	L	L
	H	X	↓	X	H	H	H	H	H
SHIFT RIGHT	L	↓	X	1	X	L	q0	q1	q2
	L	↓	X	H	X	H	q0	q1	q2
MODE CHANGE	↑	L	X	X	X	NO CHANGE			
	↑	H	X	X	X	UNDETERMINED			
	↓	X	L	X	X	NO CHANGE			
	↓	X	H	X	X	UNDETERMINED			

↓: LOW VOLTAGE ONE SETUP TIME PRIOR TO HIGH-TO-LOW CLOCK TRANSITION  
 LOWER CASE LETTERS REPRESENT STATE OF REFERENCED OUTPUT ONE SETUP TIME PRIOR TO  
 HIGH-TO-LOW CLOCK TRANSITION  
 ↑: LOW-TO-HIGH TRANSITION  
 ↓: HIGH-TO-LOW TRANSITION

FIG. 5—THE 74LS95. Before you test any IC, you have to prepare a function table. That's not only because you have to determine a valid and complete set of stimuli, but you also have to determine what kind of response you should expect.

look for a response of all ones, except for the single closed switch.

### Using the tester

In order to test an IC, the isolation switches and the power-supply jumpers must be configured for the specific circuit to be tested. The switches assigned to input pins of the test circuit must be closed and those assigned to output pins and power supply pins must be open. The actual test is done by comparing the set of test responses with a set of known good responses.

The best way to obtain a set of good responses is to issue stimuli to an IC (of the type you want to test) that you know to be good. You can save the responses for future comparison. Actually you should save both stimuli and responses since the patterns issued during the test must exactly match those used to create the initial patterns. A less attractive way of obtaining good response data is to issue patterns to a circuit thought to be good and then manually examine the response data to see if it is correct.

Let's consider the steps involved in testing the SN74LS95, a four-bit shift register with parallel-load capability. Figure 5 shows the pinout for that 14-pin IC along with a table that describes the circuit's functions. The MODE input (pin 6) deter-

mines the mode in which the circuit is operating (parallel load or shift), and also determines which of the two clocks is permitted to change the register contents. The function table reveals that there are edge transitions in the mode-change area that will cause indeterminate results. Those transitions should be avoided during the testing process because they represent invalid stimuli.

Fourteen-pin IC's should be mounted in the 16-pin socket such that pins 8 and 9 of the 16-pin socket are empty. (Consequently, pin 14 of the IC is connected to pin 16 of the test socket and to S16.) Figure 6 shows that and also indicates the required position of each isolation switch for that particular IC. The +5-volt power-supply jumper must be connected from SO3 pin 14 to SO2 pin 16. The ground jumper goes from SO3 pin 7 to SO2 pin 7. Set up the switches and power-supply jumpers, insert the IC into the test socket, and testing can begin.

The 74LS95 has multiple edge-sensitive inputs. As we mentioned previously, counting through the inputs does not generate suitable (complete and valid) stimuli for that IC. We shall use what amounts to a smaller and somewhat simpler set of test patterns as shown in Table 1.

Table 1 indicates that several patterns

are issued to the 74LS95 before each response is read. For example, consider line 12 of the table. A stimulus of 0228H is sent, bringing CK2, MODE, and D<sub>3</sub> high. That is followed by 0068H on line 13, which brings CK2 low. A response is then accepted; it and should equal 9028H on line 14.

How does the host know when to accept a response? The software driver can take advantage of the fact that the isolation switches for the two power-supply pins are known to be open and that, consequently, those two pins are not sensitive to stimuli. Bits 7 and 16 can be used to imbed control-flag bits into the test data. Those can be used to tell the host whether to generate a stimulus, expect a response, call a subroutine, etc.

For example, when the ground line (bit 7) is made high, as on line 16, it indicates (to the author's test software) that a response is to be taken after the pattern is sent. When the +5-volt line is made high as on line 18, the software driver will call a subroutine to clear the 74LS95 before sending the test pattern on line 18. Notice that line 15 brings bits 16 and 7 of the test circuit low. The software driver writes that pattern and then proceeds to the next pattern on line 16. The pattern on line 16 has bit seven high (ground pin), causing the driver to accept a response (line 17) after sending the 0070H pattern. When both supply lines are high as on line thirty, it signifies "end of test." That use of power supply pins is one way, but not the only way, of simplify the passage of control parameters to the software.

Because the SN74LS95 does not have a separate CLEAR pin, the clear subroutine (Table 1, lines 1-4) must use the parallel-load capability of the circuit to load all zeros into the internal register. The response should be tested after the clear subroutine since a failure here will cause subsequent failures in the main body of the test. It is good general practice to flag the earliest possible failure in a series of tests, especially if the software is going to perform a complete and exhaustive failure analysis.

### Options

If you build the tester on a prototyping board designed for your host system, you will almost certainly find that there is plenty of space left on the board for possible expansion. For example, an 18-pin socket may be added for the purpose of testing specific 18-pin circuits such as the popular 2114 RAM series. (The power supply pins would be permanently wired to the power-supply lines, while the other sixteen pins would be wired parallel to the sixteen test socket pins.) Pin correspondence between the test socket and the 18-pin socket can be assigned in any convenient order.

The addition of a PROM or EPROM



SN7495 PIN	14	13	12	11	10	9	8			7	6	5	4	3	2	1
FUNCTION	+5V	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>2</sub>	Q <sub>3</sub>	CK1	CK2	NC	NC	QD	MODE	Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	Q <sub>s</sub>
SWITCH NUMBER	S16	S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1
AND POSITION																

FIG. 6—SWITCH SETUP for testing the 7495 4-bit shift register.

programmer is practical because there are already sixteen latches on the board. If you add another SN74LS75, you'll have enough latches for the data and address lines of a 4K PROM. A square- or rectangular-wave generator may be added by selecting a bit and placing it under software control. Parameters describing the desired wave could be passed to software which would then compute the loop variables required to create the wave on the chosen bit line. Squarewave monitoring could be done by bringing the external wave into a bit line and letting the software sample the line. Those are just a few of the many ways to expand the IC tester.

#### Oscilloscope adapter

The logic analyzer is a very useful tool. But because it's also rather expensive, most hobbyists do not have access to one. However, we'll offer an alternative to the logic analyzer: an adapter for your oscilloscope that allows it to display 16 sig-

nals. Of course, its linearity and general quality of presentation won't match that of the logic analyzer. And it won't imitate the varied and complex functions of a logic analyzer. But it *can* be built from inexpensive, common logic components. So for a very small expense, you can add a valuable tool for testing digital IC's to your testbench. Perhaps more important, the oscilloscope-adapter logic analyzer can be an excellent tutorial aid.

The author's display-adapter prototype was developed as an expansion of the digital IC tester that was presented last month, and it uses TTL IC's. However, the oscilloscope adapter can be built as a stand-alone unit, and the ideas can be applied to other logic families as well as TTL.

#### A look at the circuit

The oscilloscope adapter's circuit (its schematic is shown in Fig. 7) uses a counter/multiplexer/converter scheme to time-share sixteen digital signals onto a single

oscilloscope channel.

The display counter, IC12 (a 74LS191 synchronous up/down counter), selects one of the sixteen channels for display. As it counts, each channel is selected in proper sequence—channel 1 is displayed on top, and channel 16 on the bottom.

A 74150 1-of-16 data selector/multiplexer, IC11 accepts the sixteen digital inputs from the tester. It selects one of them to pass to its output (pin 10). The selected input depends on the contents of the display counter.

The inverting buffer/drivers, IC13, along with its associated resistors, form a 5-bit 32-state digital-to-analog (D/A) converter. It combines the four display-counter bits and the single, selected channel bit into one of thirty-two discrete voltage steps—two voltage levels for each displayed channel. The output signal to the oscilloscope swings through more than four volts of the available 5-volt power-supply range.

TABLE 1

		SIGNAL TYPE	HEX VALUE	BIT VALUES				ACTION
				16..13	12..9	8..5	4..1	
CLEAR	01	Stimulus	0220	0000	0010	0010	0000	- Bring $\overline{CK2}$ and MODE high
	02	Stimulus	0020	0000	0000	0010	0000	- Drop $\overline{CK2}$ (load zeros)
	03	Stimulus	0040	0000	0000	0100	0000	- Drop MODE, then test
	04	Response	8000	1000	0000	0000	0000	- All pins low except $V_{cc}$
PARALLEL LOAD TEST	05	Stimulus	8020	1000	0000	0010	0000	- Clear, then bring MODE high
	06	Stimulus	0222	0000	0010	0010	0010	- Bring $\overline{CK2}$ , MODE, $Q_0$ high
	07	Stimulus	0062	0000	0000	0110	0010	- Drop $\overline{CK2}$ , then test
	08	Response	C022	1100	0000	0010	0010	- $V_{cc}$ , $Q_0$ , MODE, $Q_0$ high
	09	Stimulus	0224	0000	0010	0010	0100	- Bring $\overline{CK2}$ , MODE, $Q_1$ high
	10	Stimulus	0064	0000	0000	0110	0100	- Drop $\overline{CK2}$ , then test
	11	Response	A024	1010	0000	0010	0100	- $V_{cc}$ , $Q_1$ , MODE, $Q_1$ high
	12	Stimulus	0228	0000	0010	0010	1000	- Bring $\overline{CK2}$ , MODE, $Q_2$ high
	13	Stimulus	0068	0000	0000	0110	1000	- Drop $\overline{CK2}$ , then test
	14	Response	9028	1001	0000	0010	1000	- $V_{cc}$ , $Q_2$ , MODE, $Q_2$ high
	15	Stimulus	0230	0000	0010	0011	0000	- Bring $\overline{CK2}$ , MODE, $Q_3$ high
	16	Stimulus	0070	0000	0000	0111	0000	- Drop $\overline{CK2}$ , then test
	17	Response	8830	1000	1000	0011	0000	- $V_{cc}$ , $Q_3$ , MODE, $Q_3$ high
SHIFT TEST	18	Stimulus	8401	1000	0100	0000	0001	- Clear, then bring $\overline{CK1}$ , $D_8$ high
	19	Stimulus	0041	0000	0000	0100	0001	- Drop $\overline{CK1}$ , then test
	20	Response	C001	1100	0000	0000	0001	- $V_{cc}$ , $Q_0$ , $D_8$ high
	21	Stimulus	0400	0000	0100	0000	0000	- Bring $\overline{CK1}$ high $D_8$ low
	22	Stimulus	0040	0000	0000	0100	0000	- Drop $\overline{CK1}$ , then test
	23	Response	A000	1010	0000	0000	0000	- $V_{cc}$ , $Q_1$ high
	24	Stimulus	0401	0000	0100	0000	0001	- Bring $\overline{CK1}$ , $D_8$ high
	25	Stimulus	0041	0000	0000	0100	0001	- Drop $\overline{CK1}$ , then test
	26	Response	D001	1101	0000	0000	0001	- $V_{cc}$ , $Q_0$ , $Q_2$ , $D_8$ high
	27	Stimulus	0400	0000	0100	0000	0000	- $CK_1$ high and $D_8$ low
	28	Stimulus	0040	0000	0000	0100	0000	- Drop $\overline{CK1}$ , then test
	29	Response	A800	1010	1000	0000	0000	- $V_{cc}$ , $Q_1$ , $Q_3$ high
	30	Stimulus	8040	1000	0000	0100	0000	- END OF TEST



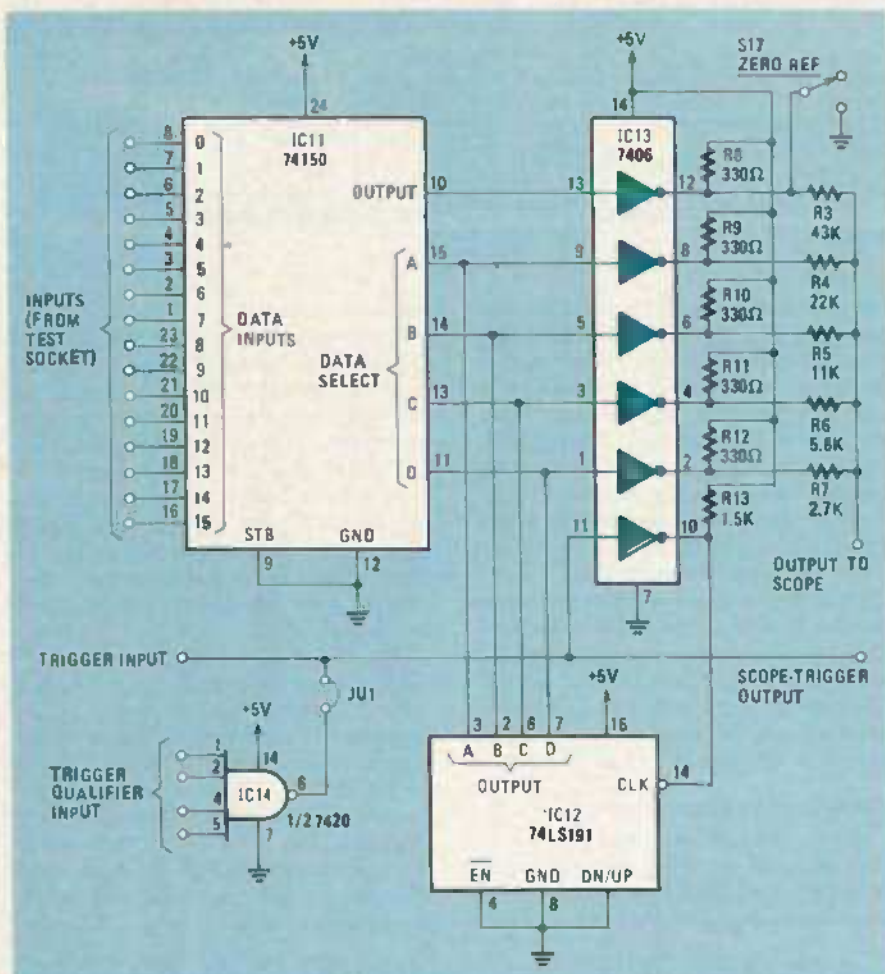


FIG. 7—OSCILLOSCOPE ADAPTER SCHEMATIC. IC12 selects which of IC11's inputs gets sent to the output buffer. The resistor network at the output of the buffer establishes 32 discrete voltage steps—two for each channel. IC14 is a trigger qualifier; it's optional, but suggested. Jumper JU1 is used to take the qualifier in and out of the circuit.

A trigger must be supplied by the logic system or device being observed. The trigger is needed both to act as a clock for the display counter and to initiate a horizontal oscilloscope sweep. The trigger signal is applied to pin 11 of IC13; sixteen of those trigger pulses are required to generate a complete 16-trace image on the scope. In complex digital systems, a single signal that can act as a suitable trigger is not always available—you may need an optional trigger qualifier so that you can create an acceptable trigger. A qualifier gate, such as what is shown for IC14, can be used. We recommend that you add such a qualifier gate along with a provision to enable it simply by the insertion of a jumper.

Switch S17 is an optional zero-reference switch. It is used to place a low logic-level on the least-significant input leg of the converter. When the switch is closed, the scope will display the 16 base lines. By momentarily closing the switch, you can quickly identify a steady-state channel as steady high or steady low. The switch is also useful in linearity tests.

#### Construction

You probably have enough room left on

your original IC tester board to build the display adapter. Since the wiring or component layout is not critical, practically any construction technique may be used with good results.

There are many possible component substitutions that can be made in the TTL design. For example, the counter does not have to be a 74LS191: It can be another kind of synchronous counter like the 74LS163 or the 74LS193. Or it may be a standard (as opposed to low-power Schottky) TTL counter like the 74191. If substitutions are made, be sure to check the pinout of the new IC and document the changes where necessary. The use of ripple counters such as the 7493 is not recommended in this circuit—they tend to introduce excessive channel-switching transients. The 7406 may be substituted by other open-collector hex inverters including the 7416, and the 7405.

One final construction note: Assuming that you are building the circuit as an expansion of the IC tester, each display channel should be connected to its corresponding pin on the test socket. In other words, channel 1 should display the signal on pin 1 of test socket SO1 and so on. That makes using the analyzer easier.

#### Using the display adapter

For illustration purposes, let's set up the IC tester so that we can examine the waveshapes associated with the 74LS138 one-of-eight decoder. Table 2 lists the proper stimulus patterns. The trigger should be jumpered from the stimulus latch side of the isolation switch for pin 8 to pin 11 of IC13 in Fig. 7, and the scope should be set to trigger from the positive edge of an external signal. Isolation switches S1–S6 should be closed, and switches S7–S16 should be open.

Notice that the first two stimulus patterns will send a clock pulse to the display counter and trigger the oscilloscope, respectively. As shown in Fig. 8, the counter counts on the leading edge of the trigger pulse and the display begins on the trailing edge. That eliminates the channel-switching lines and lets us see a clean display.

Program your host computer to generate the looping stimulus patterns in Table 2 and run the test on a 74LS138. Connect the oscilloscope test leads to the adapter and, with the test running, we are ready to observe digital waveforms.

Set the vertical-sensitivity control to 1

TABLE 2

STIMULUS (HEX)	COMMENT
0020	Trigger low, count counter
00A0	Trigger high, trigger scope
00A1	
00A2	
00A3	Test patterns
00A4	
00A5	
00A6	
00A7	
	Loop back to first stimulus

volt/division and set the timebase to the fastest sweep. The exact sweep speed required for observation will depend on the speed at which the host system emits stimuli. Decrease the sweep speed and carefully observe the counting stimuli on channels one, two, and three, to determine when the proper speed is found. (It will probably be necessary to decalibrate the timebase to display the entire interval between triggers.) If the display shows four or eight traces with connecting steps, then the sweep is too slow but is close to the correct speed. If the display resembles multiple downward staircase waveforms, then the sweep is far too slow. Once the timebase is adjusted, the vertical sensitivity can be adjusted (and decalibrated) so that the sixteen channels cover the entire face of the scope, giving maximum channel separation. The display should resemble that shown in Fig. 9.

(continued on page 111)



## DIGITAL IC TESTER

continued from page 86

You can use the display adapter to view signals that originate outside of the tester. With the test socket empty and the power supply jumpers removed, open the isolation switches and jumper the system sig-

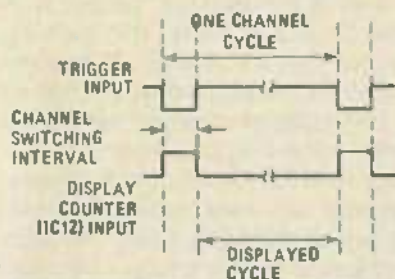


FIG. 8—THE COUNTER COUNTS on the leading edge of a logic trigger, but the scope triggers on the trailing edge. That eliminates the channel-switching lines from the display.

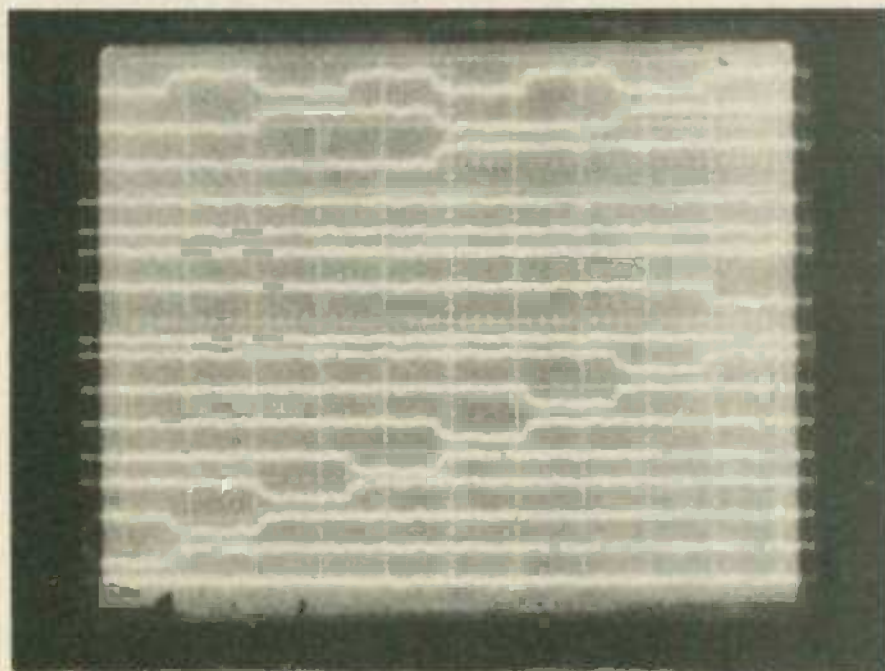


FIG. 9—THE OSCILLOSCOPE DISPLAY showing the waveforms generated by the 74LS138 test circuit. While it may not be the most eloquent logic analyzer, it can be an excellent teaching or learning tool.

nal(s) to the test socket pin(s). Bring in a suitable trigger, crank up the system, and you are ready to observe the chosen signals and their relationship to each other. Be aware that the displayed signals are now driving the additional load presented by the circuitry connected to the test socket—that includes the output drivers, the multi-channel adapter, and, perhaps, additional expansion hardware. System signals that are heavily loaded might not be able to drive that additional load.

The selection or generation of a trigger is most crucial in creating a stable and meaningful display. The trigger pulses

should be evenly spaced, and all displayed signals should repeat themselves exactly between triggers. The qualifier can be useful in dealing with complex systems, but proper qualification hinges on your

## PARTS LIST

All resistors are 1/4-watt, 5%

R3—43,000 ohms

R4—22,000 ohms

R5—11,000 ohms

R6—5600 ohms

R7—2700 ohms

R8—R12—330 ohms

R13—1500 ohms

Semiconductors

IC11—74150 1-of-16 selector/multiplexer

IC12—74LS191 binary synchronous up/down counter

IC13—7406 hex inverting buffer

IC14—7420 dual 4-input NAND gate

Other components

S17—SPST switch

## 9 reasons why the real pros prefer Endeco desoldering irons



1. Operates at 120v, 40w. Idles at 20w for longer tip life
2. Flexible, burn resistant Neoprene cord set
3. Cool, unbreakable polycarbonate handle
4. Exclusive bracket insures alignment, prevents damage
5. Safety light in handle tells when it's on
6. Stainless steel construction
7. Temperature control, Low, high or off.
8. Eight tip sizes. Comes with .063 I.D.
9. Converts to soldering iron with 1/4" shank type tip

See your distributor or write . . .

**Enterprise Development Corp.**

5127 E. 65th St. • Indianapolis IN 46220  
PHONE (317) 251-1231

For Home or Business: WHAT...WHEN...HOW...

## LEARN HOW TO USE A COMPUTER



Now at Home in Spare Time, you can learn everything you always wanted to know about personal computers. How to program in BASIC. How to understand and use more than 80 BASIC commands and functions. How to write and run your own programs...for both personal and business applications. How to use pre-packaged software and change it to meet your special needs. How to make sense of the overwhelming maze of books, information and advice available at your local computer store.

### More Than Just A Computer Manual

This is more than just another programming manual...it's an entire comprehensive course written by experts. Yet, because it was especially developed for home study, you learn everything right in your own home, without changing your job or lifestyle, without attending a single class.

### Plus You Get Your Own Computer

To give you practical hands-on experience, this course includes your own personal computer—plus a cassette recorder that lets you save your programs on tape. Get all the facts. MAIL COUPON TODAY!



COMPUTER TRAINING, Dept. DE894  
Scranton, PA 18515

Rush free facts how I can learn computer applications, programming and operation at home in spare time

Name \_\_\_\_\_ Age \_\_\_\_\_

Address \_\_\_\_\_

City/State/Zip \_\_\_\_\_

Phone-including area code (\_\_\_\_\_) \_\_\_\_\_

OCTOBER 1984



# BUILD THIS



## Computer Controlled IC Tester

FLOYD L. OATS

*We've already seen that the IC tester we've been building can be used for more than just testing digital IC's. This month we'll add an EPROM programmer.*

**Part 3** IN THE FIRST TWO parts of this article, we looked at a circuit that could be used—along with your computer—to test digital IC's. We then added a low-budget logic analyzer to the tester. It worked by allowing you to view 16 digital signals on a single-trace oscilloscope. This month, we'll expand the circuit even further by adding EPROM-programming capability.

Before we go on, we should add some cautions: The circuit was designed to be used with an S-100 bus computer. With modifications, it can be used with essentially any computer. In either case, some programming experience is essential: While the software that is required is described, the actual, complete programs are not presented.

### EPROM programmer

The digital IC tester—along with your computer—can be used to program and verify virtually any 24-pin member of the

2700 EPROM family. That includes everything from the original 2704/2708 through and including the 2732. Although the circuit can be modified to program 28-pin devices such as the 2764, we will not cover that here.

Figure 10 shows the pinouts for some of the various styles of 24-pin EPROM's. You'll want to refer to it as we talk about the different types of EPROM's available. We should add one more caution, though: Before you try to program any EPROM, make sure you have the manufacturer's data sheets. Pinouts, voltage requirements, and programming sequences can vary from one manufacturer to another, and from older devices to newer ones. Double checking is the safest and surest way to avoid problems.

Before we can discuss the EPROM programmer circuit, we have to know what we want it to do. So let's first consider how the 2708—a typical three-supply EPROM—is programmed. (Note that

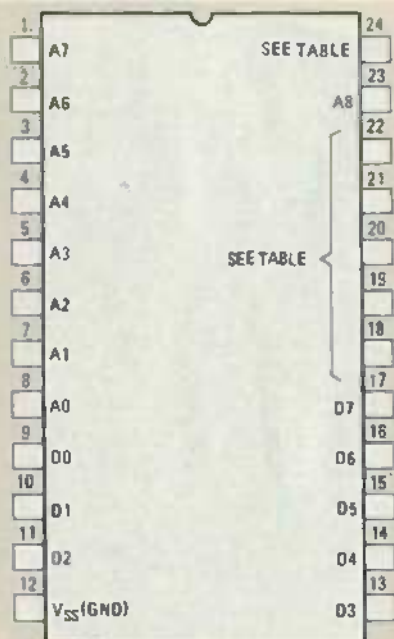
when we say "three-supply" or "single-supply," we are referring to the number of power supplies required for READ operation—we do not include the programming-voltage supply.)

To program the 2708, its  $\overline{CS}/WE$  pin is brought to +12 volts and is held at that voltage throughout the programming operation. (That causes the eight data lines to become input-data lines). An EPROM address, and the data to be stored in that address, are then presented to the EPROM. Both the address and data lines must be held in their valid state while a program pulse is applied to the program pin.

That +25-volt program pulse has a pulse width between 100 microseconds and 1 millisecond. After the pulse is terminated, the next (sequential) address, along with the contents to be stored there, are sent to the EPROM and another program pulse is issued. After all addresses have been pulsed, we say we have com-

NOVEMBER 1984





TYPE	PIN	18	19	20	21	22	24
2704		PROGRAM	V <sub>DD</sub>	CS/AWE	V <sub>BB</sub>	V <sub>SS</sub>	V <sub>CC</sub>
2708		PROGRAM	V <sub>DD</sub>	CS/AWE	V <sub>BB</sub>	A <sub>9</sub>	V <sub>CC</sub>
2758		CE/(PGM)	A <sub>8</sub>	OE	V <sub>pp</sub>	A <sub>9</sub>	V <sub>CC</sub>
2716		CE/(PGM)	A <sub>10</sub>	OE	V <sub>pp</sub>	A <sub>9</sub>	V <sub>CC</sub>
2732		CE/(PGM)	A <sub>10</sub>	OE/V <sub>pp</sub>	A <sub>11</sub>	A <sub>9</sub>	V <sub>CC</sub>

FIG. 10—PIN FUNCTIONS VARY from one type of EPROM to another. While you can use this as a guide, make sure you have the manufacturer's data sheet for any EPROM you want to program.

pleted a single pass.

It is necessary to make enough passes to bring the total program-pulse time to at least 100 milliseconds per address. That means that if your pulse width is, say, 500 microseconds, then at least two hundred passes must be made.

The 2708 is only one kind of EPROM—a three-supply type. Now we'll look at the programming procedure for a typical single-supply EPROM: the 2716.

To enter the 2716's programming mode, pin 20, the output-enable pin (OE, pin 18) is brought to +5 volts, and the programming-voltage pin (V<sub>pp</sub>, pin 21) is brought to +25 volts. The address and corresponding data are presented to the IC and held valid while a fifty-millisecond high-level TTL pulse is sent to the CE/PGM pin. (Note that, as opposed to the 2708, we program with a TTL-level pulse—the programming voltage applied to pin 21 remains constant at 25 volts. We should note, however, that the programming voltage can be switched, if desired.)

Only a single pulse is required to completely program a location. Unlike the 2708, sequential addressing is not necessary and only the addresses being written need be pulsed. Therefore, since all of the address locations in new (or erased) EPROM's are set to 1, to program a 2716, only the addresses that need to be changed to zero bits are "burned in." Those addresses that need to be set to one are

programmed by simply passing over the location.

### The EPROM-programmer circuit

Because the EPROM programmer (whose schematic is shown in Fig. 11) is being built as an add-on to the digital IC tester, we have to map up to four kilobytes of EPROM address into 256 memory-mapped locations. (You will recall that the IC tester uses eight address lines to carry stimulus information and thus occupies 256 memory locations.)

We overcome that problem by using a D-type flip-flop (IC15) to store the most-significant bits of the EPROM address, and by implementing a three-phase program/verify cycle (which we'll describe in more detail shortly). When we combine IC15 with the sixteen stimulus latches (IC1-IC4), we obtain thirteen address lines and eight data lines. That's enough to accommodate up to 8K of EPROM space.

The three-phase cycle is shown in Fig. 12. In the first phase, enabled by A<sub>3</sub> being high, the EPROM upper address (A<sub>8</sub>-A<sub>12</sub>) is loaded into IC15. In the second phase, the eight least-significant address bits (A<sub>0</sub>-A<sub>7</sub>) along with eight data bits (D<sub>0</sub>-D<sub>7</sub>) are loaded into the stimulus latches (IC1-IC4, Fig. 2)) and a "burn" interval is initiated. In the third phase, the burn interval is terminated and the programmer is restored to its idle state (which is de-

fined as both of the flip-flops in IC16 being reset).

Referring back to Fig. 11 (and to Fig. 3 in Part 1), you'll notice that five of the upper stimulus latches (A<sub>0</sub>-A<sub>4</sub>) are fed to IC15. The A<sub>3</sub> latch is sent to pin 1 of IC17. The STIMULUS LOAD signal is inverted to an active-low pulse by IC17-f and then applied to both flip-flops of IC16 as a clock.

Assume the programmer to be in the idle state, awaiting a phase-one load. If A<sub>3</sub> is high (A<sub>3</sub> low), pin 2 of IC16 is conditioned to set one of its flip-flops on the trailing edge of the stimulus-load pulse. (We'll call that flip-flop, IC16-a, the LOAD flip-flop.) When the LOAD flip-flop sets, it will clock pin nine of IC15, thereby capturing the EPROM upper address. The LOAD flip-flop primes the second flip-flop of IC16 to set and, through IC17-b, conditions its own reset. (We'll call the second flip-flop, IC16-b, the TTL flip-flop for TTL Pulse.)

The second phase starts with the next stimulus load pulse, which will now reset the LOAD flip-flop and set the second (TTL) flip-flop to initiate the burn interval (see Fig. 12). TTL activates the program/verify controls and also maintains a low into the LOAD flip-flop through IC17-e.

The LOAD flip-flop being reset causes the TTL flip-flop to reset on the next stimulus load (phase three). The burn interval is terminated and the programmer is returned to its idle state. We should note that IC15 still retains the bits captured during phase one.

During a program cycle, the TTL flip-flop supplies a TTL pulse for programming those EPROM's that require only a TTL-level pulse. The signal also controls the high-voltage pulse needed by many EPROM's. For example, the TTL signal controls the base of Q1 through IC17-c and IC17-d. Transistor Q1 controls V<sub>pp</sub>, the programming voltage. When Q1 is turned on, it will drop about two volts and pass the remaining 26 volts to the EPROM. When pin 6 of IC3 is low, diode D3 returns the program pin to ground. Capacitor C4 limits the overshoot on the emitter of Q1 to acceptable levels.

Switch S24 is useful in testing and troubleshooting Q1 and associated circuitry—when closed, it will hold V<sub>pp</sub> high without regard to the state of TTL.

Closing switch S25 will unconditionally hold the programmer in the idle state thereby disabling the entire circuit. A quick-arming power-on reset function is provided by C1, R15, and R16.

### Personality module

As shown in Fig. 11, the signals and voltages developed are sent to the EPROM socket (SO5) through a switching and distribution network (S18-S24, and S04). Socket SO4, the personality module socket, accepts a personality



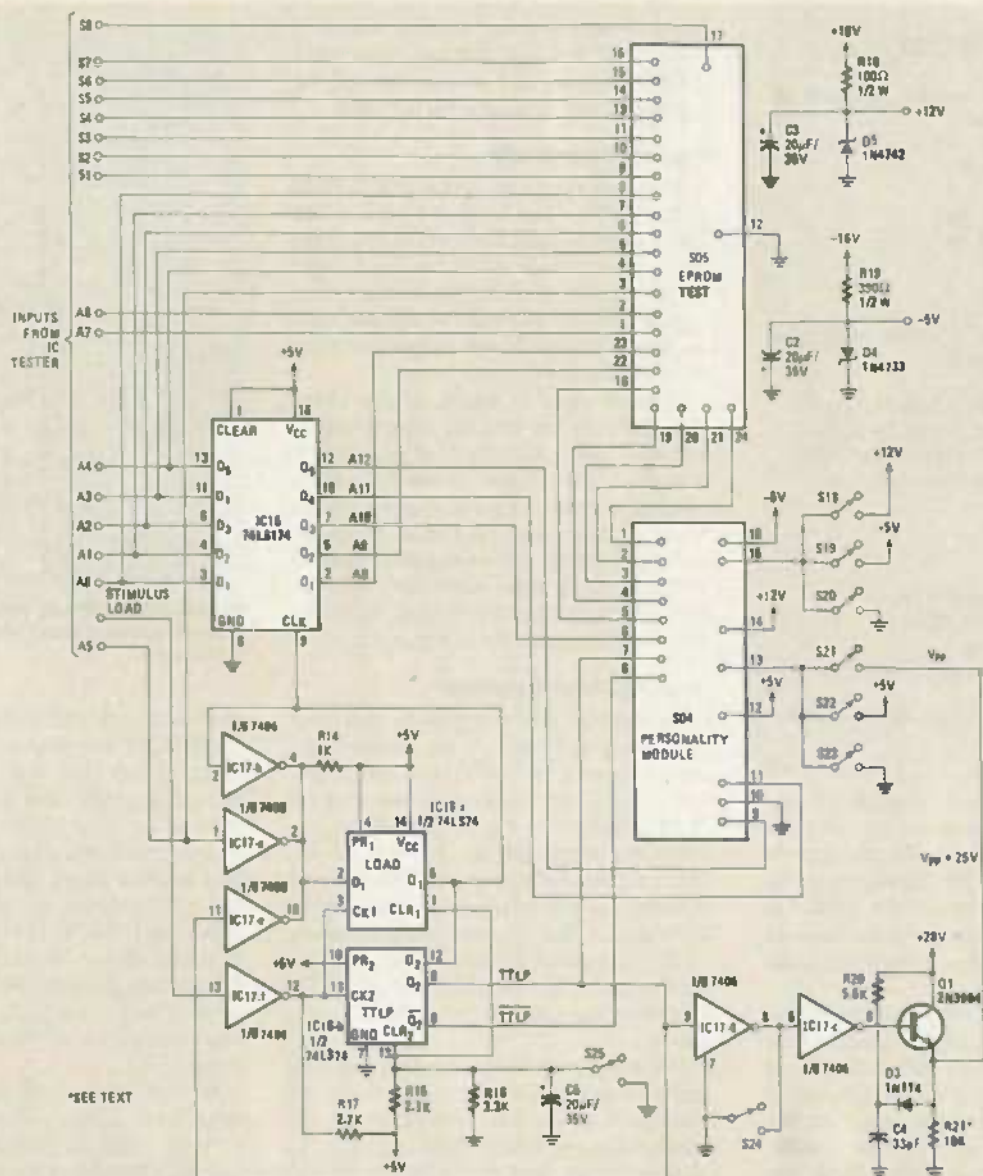


FIG. 11—EPROM PROGRAMMER SCHEMATIC. Note that some EPROM's require a 21-volt (instead of 25–26 volt) programming voltage. When programming such devices, you can either use a variable power supply, or you can substitute a voltage divider (or trimmer potentiometer) for R21.

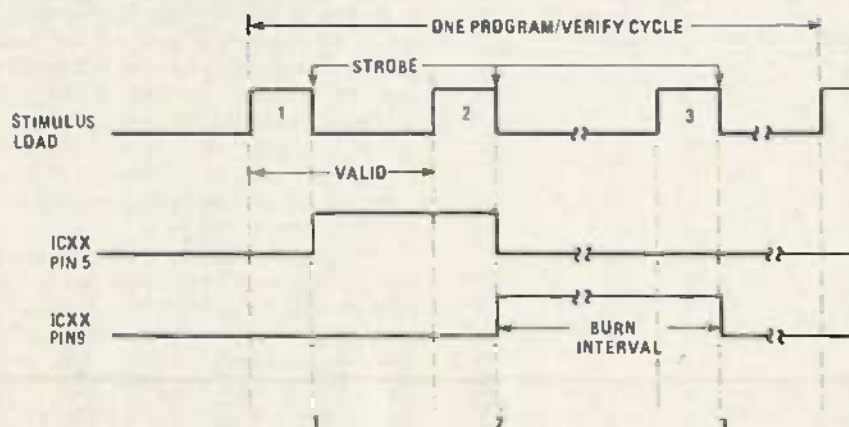


FIG. 12—THE THREE-PHASE cycle required to program or verify a single EPROM address. The Program or verify function is selected by the programmer's switches. (See Fig. 13.)

module that is custom made for each style of EPROM. (The personality module can be made on a DIP header, and consists of

appropriate jumpers. We'll get to the specifics of the jumpers shortly.) The switches (S18–S23) are configured to select ei-

ther a program or verify function.

The personality-module outputs (pins 1–5) feed what may be termed the "variable" EPROM pins. Figure 13 defines the personality module and the program/verify configuration needed for each EPROM. We'll discuss those configurations in more detail later.

### Construction

It is recommended—but not essential—that the programmer be built on the same board with the IC tester. Component placement, general signal routing, and construction method are not critical and are therefore left to your discretion. Don't forget that the EPROM programmer needs an external source of +28 volts at about 100 mA. (A variable power supply is preferred.)

Most of the signal connections to the IC tester can be seen in Fig. 11. The EPROM socket data-lines are connected to the isolation side (test-socket side) of the isolation switches so that they may be



## PARTS LIST

All resistors are 1/4-watt, 5% unless stated otherwise

R14—1000 ohms

R15, R17—2700 ohms

R16—3300 ohms

R18—100 ohms, 1/2 watt

R19—390 ohms, 1/2 watt

R20—5600 ohms

R21—10,000 ohms

### Capacitors

C2, C3, C5—20  $\mu$ F, 35 volts, electrolytic

C4—33 pF, ceramic disc

### Semiconductors

IC15—74LS174 hex D-type flip-flop

IC16—74LS74 dual D-type flip-flop

IC17—7406 hex inverting buffer/driver

IC18—74LS367 hex bus driver

Q1—2N3904

D3—1N914

D4—1N4733

D5—1N4742

LED1—standard green LED

LED2—standard red LED

### Other components

S18—25—SPST Switch (DIP switches preferred)

Miscellaneous: IC sockets, jumper header, etc.

connected to the stimulus latches for programming or disconnected for program verifying. The least-significant address bits, A<sub>0</sub> through A<sub>7</sub>, are connected to the most-significant byte of the stimulus latches and five of those (A<sub>0</sub>–A<sub>4</sub>), also go to IC15. The address lines are connected directly to the latches.

The personality-module socket (SO4) will accept a 16-pin header plug, which must be pre-wired for a particular type of EPROM (according to the data shown in Fig. 13.) As an example, consider the 2732 entry in that figure: Pin one is wired to pin twelve, pin two to pin eleven, pin three to pin thirteen, etc. Once you install

the jumpers on the header, label it "2732." If you can manage, you might want to squeeze the switch settings for program/verify selection on the label.

### Diagnostic indicator

Figure 14 shows an option you can add to the programmer to give both a visual and a software indication of the programmer's current phase. It can be very helpful for debugging and troubleshooting, and can be used by software to continuously monitor the state of the programmer for self-diagnostic purposes.

When a signal is active, its line to the test socket is low and the corresponding LED is illuminated. The host system may determine the current phase by simply reading the test socket and examining pins 9 and 10. (If you use this option, the isolation switches for test socket pins 9 and 10 will have to be open while the programmer is in use. Switch S25 must be closed when the programmer is not in use.)

### Testing the programmer

Preliminary tests designed to check out the wiring and logic of the programmer are made with the EPROM socket empty and the 28-volt supply removed. Set up the IC tester with the test socket vacant, the power-supply jumpers removed, isolation switches for pins 9 and 10 open, and all other isolation switches closed. Open all eight of the programmer's switches (S18–S25) and, if practical, power the device up and down several times and verify that the power-on reset forces the idle condition.

All stimulus patterns can be viewed in terms of test-socket pins since, from the standpoint of the host computer, the expanded tester still appears as merely a 16-pin test socket. Referring to figure 15, the first stimulus pattern of the three-phase

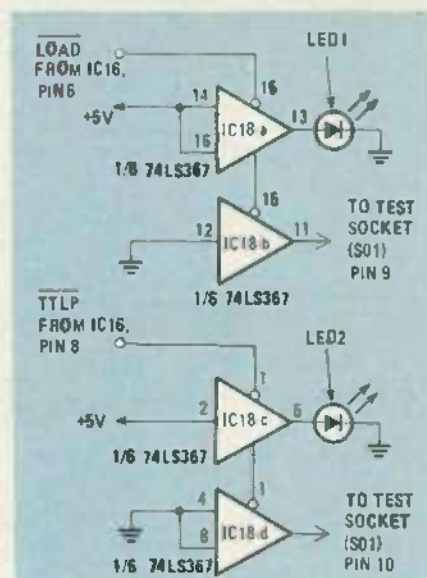


FIG. 14—FOR A VISUAL INDICATION of the programmer's current phase, you can add this diagnostic logic.

program/verify cycle will contain the upper EPROM address on the lines assigned to test-socket pins 9–13. The A<sub>5</sub> bit must be high to enable this phase-one load, so the line assigned to pin 14 must also be high to enable the address-register (IC15) load and the phase advance. The second and third patterns are identical and will direct the EPROM lower address to pins 9–16 and data to be programmed to pins 1–8. Although many of the patterns may never actually reach the test socket, the host computer will "think" that an IC is being tested.

We can begin by testing the phase-advance logic. Using a stimulus with the A<sub>5</sub> bit high, send the pattern repeatedly and check for the phases to progress according to Fig. 15. Starting in the idle condition, send the same stimulus once more and then change the pattern so that A<sub>5</sub> is low. Send this new pattern ten or twelve times and observe that the phase is properly advanced to the idle state but the new pattern is not able to cause any further activity. It should be apparent that the issuance of two or more consecutive stimuli with A<sub>5</sub> low will guarantee that the programmer goes to the idle state. The software can use that principle to implement a "restore" function.

For the remaining tests, you will need to insert a personality module for a 4K EPROM and set up the DIP switches to program. Make sure the switches agree with the personality-module style (see Fig. 13) and leave the EPROM socket empty. Send stimuli which will load various bit patterns into the address register and check the appropriate bits on the EPROM socket itself. Make the phase-two and -three bit patterns different from the first to be certain that the upper address bits are captured only during phase

EPROM TYPE	PERSONALITY MODULE JUMPERS	PROGRAM	VERIFY	VERIFY PROCEDURE
2704, 2708	1 – 12 2 – 16 3 – 15 4 – 14 5 – 13	S18 S21	S20 S23	V1 OR V2
2758	1 – 12 2 – 13 3 – 15 4 – 10 5 – 7	S19 S21 S24	S20 S21 S24	V1
2716	1 – 12 2 – 13 3 – 15 5 – 7	S19 S21 S24	S20 S21 S23	V1 OR V2
2732 *2732A	1 – 12 3 – 13 5 – 10	S21 S24	S23	V2

\*YOU MAY WANT TO ADD A SWITCH TO AVOID CHANGING JUMPERS

FIG. 13—PERSONALITY MODULE AND SWITCH configurations for various EPROM's. See the text for more information on verify procedures V1 (load, load, load, read) and V2 (load, load, read, load).



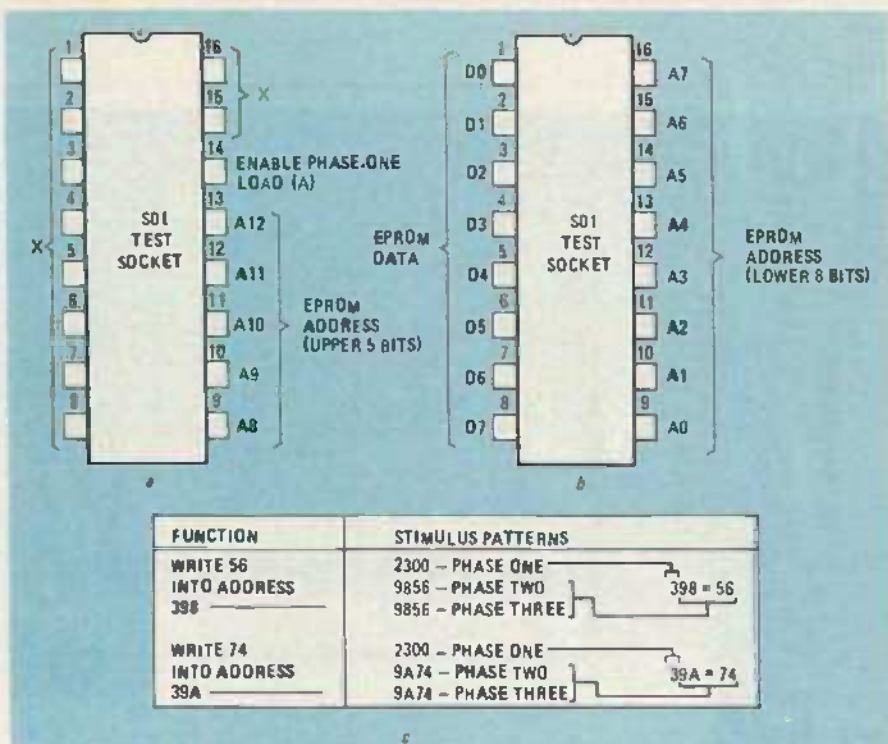


FIG. 15—THE SOFTWARE views the programmer as an IC under test. The first pattern sent is assigned to the EPROM lines as shown in a, while the second and third patterns are assigned to the lines shown in b. Sample bit patterns (in hexadecimal form) used to program two EPROM address locations are shown in c.

one. Next, vary the phase-two and -three patterns and check the data and lower-address bits, again on the EPROM socket. By testing the bits on the socket, the wiring is checked out as well as the source logic.

For the next test, an 820-ohm, 2-watt resistor is needed. (If you don't have that unusual item you can combine four 3300-ohm 1/2-watt resistors in parallel.) In either case, take notice that the resistor(s) will become very warm to the touch.

With the programmer in the idle condition, apply +28 volts to the collector of Q1 and observe Q1's emitter for a voltage near ground potential. Close switch S24, and the emitter should go to about 27 volts. Now connect the resistor, which constitutes a typical load, from the emitter to ground and look for a voltage of 25 or 26 volts. If Q1 has been substituted for, and is dropping four volts or more, the substitution is not acceptable.

Leave S24 closed and the load connected for ten minutes or longer and then open S7. The emitter should return to near ground potential. With switch S7 open, cycle the device through the phases and confirm that transistor Q1's emitter is controlled by TTL.P.

For the final hardware test, close switch S8 and make sure that the programmer is locked into the idle condition regardless of attempts to cycle it from the host computer. Now, with S8 closed and the EPROM socket empty, make sure the IC tester and any other expansion devices

still function properly.

### Using the programmer

Before you can use the programmer, you'll have to write some software to control it. A relatively simple (and popular) method is to store the intended EPROM contents in a main memory area and use software to transfer that data to the programmer (while keeping the address and data bits properly distributed and maintaining good program-pulse width).

The program will have to send the proper set of bits to the device during each phase and must control the program pulse width by means of a delay between phase two and phase three. Programming is more complex for the three-supply EPROM's because the source data is transferred to the EPROM a number of times as multiple passes are made. Don't forget that only a single pulse per address is permitted with single-supply EPROM types and the pulse width must be 50 milliseconds,  $\pm 10\%$ .

Figure 15 may be helpful in determining the address- and data-bit distribution among the three phases. Examples of exact stimuli for programming two locations are shown. Using that figure, you can develop software that will create and issue the appropriate patterns. You can leave the delay between phases two and three at a very low value for now. Check that the programmer is in the idle state when the software is initiated and that the software leaves the programmer in the idle state at

completion. If your program doesn't do that, it is not observing the three-phase cycle requirements.

Once the raw program is developed to your satisfaction, you can work on adjusting the pulse width. You can do that easily with the aid of an oscilloscope by simply measuring TTL.P and changing the software delay accordingly.

If you don't have an oscilloscope, use a clock or watch to measure the time it takes to transfer one or two kilobytes. Run the program with the EPROM socket empty, and adjust the software delay until the run time is 100 seconds,  $\pm 3$  seconds.

The EPROM verify software is easier to create because there are no delays or multiple passes involved. The three-phase cycle is used to load the addresses, but the data patterns have no significance since the isolation switches to the EPROM data lines will be open during the verify operation.

There are two verify procedures, v1 and v2 specified in Fig. 13: In the v1 process, the host reads pins 1-8 of the test socket during the idle state after having executed the three-phase cycle to load the EPROM address. In the v2 process, the read is performed during phase three, and one more load is done to complete the cycle. Normally, the contents of the EPROM is either read into a main memory area for examination or compared with a memory area to verify the EPROM contents.

When you have the hardware and software debugged, you are ready to test the programmer with an EPROM in the socket and the  $V_{pp}$  supply connected. The first thing you'll want to check is that the EPROM is erased. Power the programmer down, insert an EPROM and the associated personality module, set up the DIP switches for verify (using Fig. 13), and open the isolation switches for test socket pins one through eight. Power the programmer up and run the verify software, checking for FFH patterns in all addresses.

Power may be removed from the programmer while the DIP switches are changed between program and verify, but be careful that  $V_{pp}$  is not applied to the EPROM while the five volt supply is removed. Set up the programmer switches for the program function and close the isolation switches for pins one through eight of the test socket. With the desired EPROM data loaded into the main memory source-area and the programmer in the idle condition, bring up the 28-volt supply and execute the EPROM program software. After the program runs, remove the 28-volt supply and set up the DIP and isolation switches for verify. Run the verify software and if the EPROM contents prove to be good, you have now successfully programmed an EPROM and the project has passed its final test. R-E