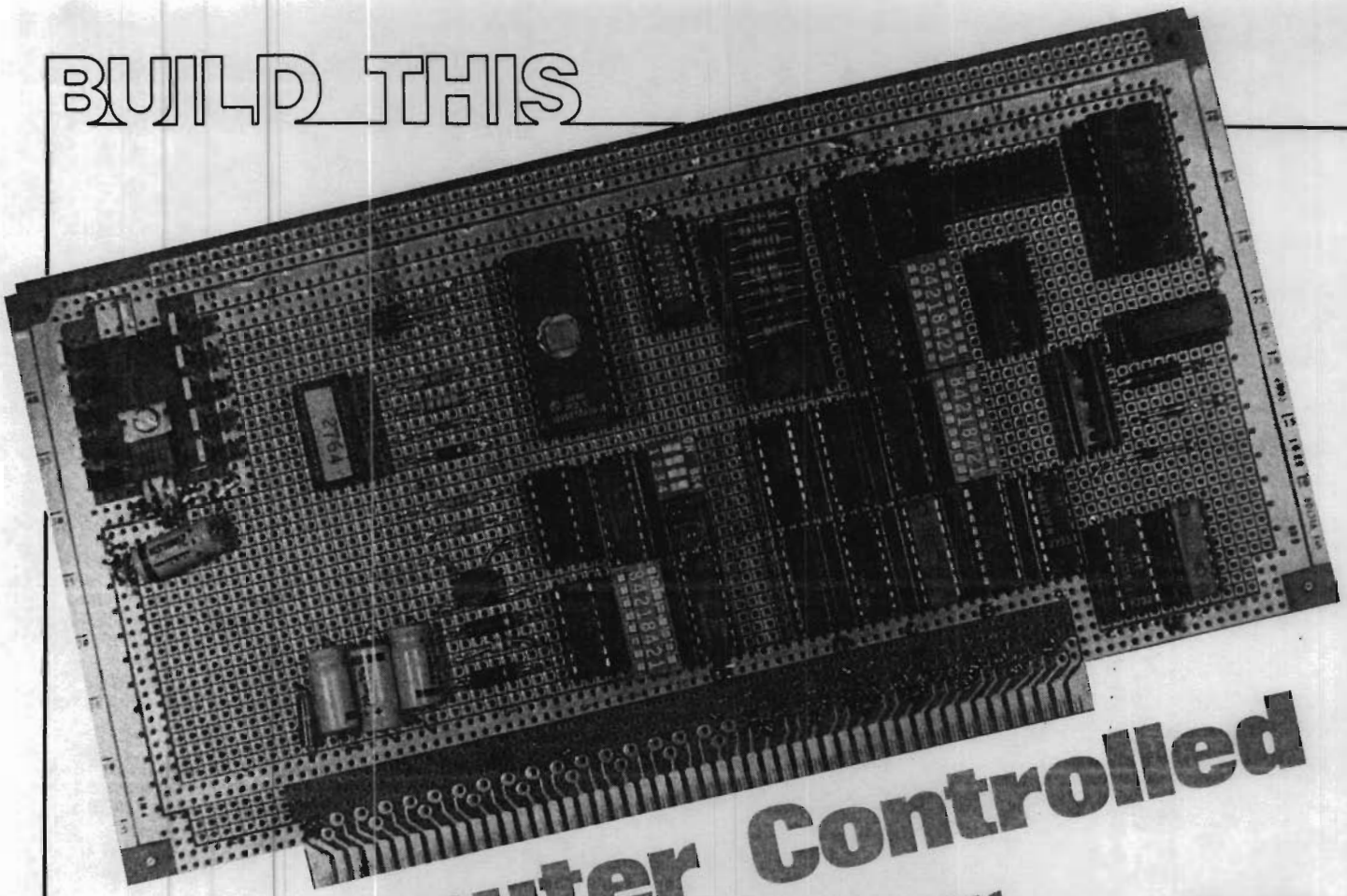
BUILD THIS



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

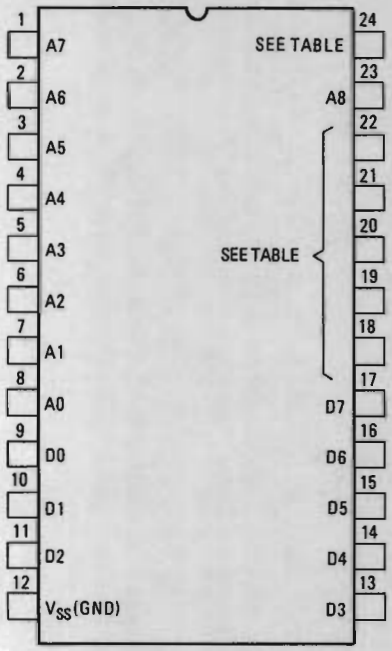
The digital IC tester—along with your computer—can be used to program and verify virtually any 24-pin member of the 2700 EPROM family. That includes everything from the original 2704/2708 through and including the 2732. Although the circuit can be modified to program 28-pin devices such as the 2764, 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-



TYPE \ PIN	18	19	20	21	22	24
2704	PROGRAM	V_{DD}	$\overline{CS}$/WE	V_{BB}	V_{SS}	V_{CC}
2708	PROGRAM	V_{DD}	$\overline{CS}$/WE	V_{BB}	A9	V_{CC}
2758	$\overline{CE}$/(PGM)	A_R	$\overline{OE}$	V_{PP}	A9	V_{CC}
2716	$\overline{CE}$/(PGM)	A10	$\overline{OE}$	V_{PP}	A9	V_{CC}
2732	$\overline{CE}$/(PGM)	A10	$\overline{OE}/V_{PP}$	A11	A9	V_{CC}

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 ($\overline{OE}$, pin 18) is brought to +5 volts, and the programming-voltage pin (V_{PP}, 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 $\overline{CE}$/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 A5 being high, the EPROM upper address (A8–A12) is loaded into IC15. In the second phase, the eight least-significant address bits (AØ–A7) along with eight data bits (DØ–D8) 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 defined 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 (A0–A4) are fed to IC15. The A5 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 A5 is high ($\overline{A5}$ 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 TTLP flip-flop for *TTL P*ulse.)

The second phase starts with the next stimulus load pulse, which will now reset the LOAD flip-flop and set the second (TTLP) flip-flop to initiate the burn interval (see Fig. 12). TTLP 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 TTLP 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 TTLP 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 TTLP signal controls the base of Q1 through IC17-c and IC17-d. Transistor Q1 controls V_{PP}, 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_{PP} high without regard to the state of TTLP.

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

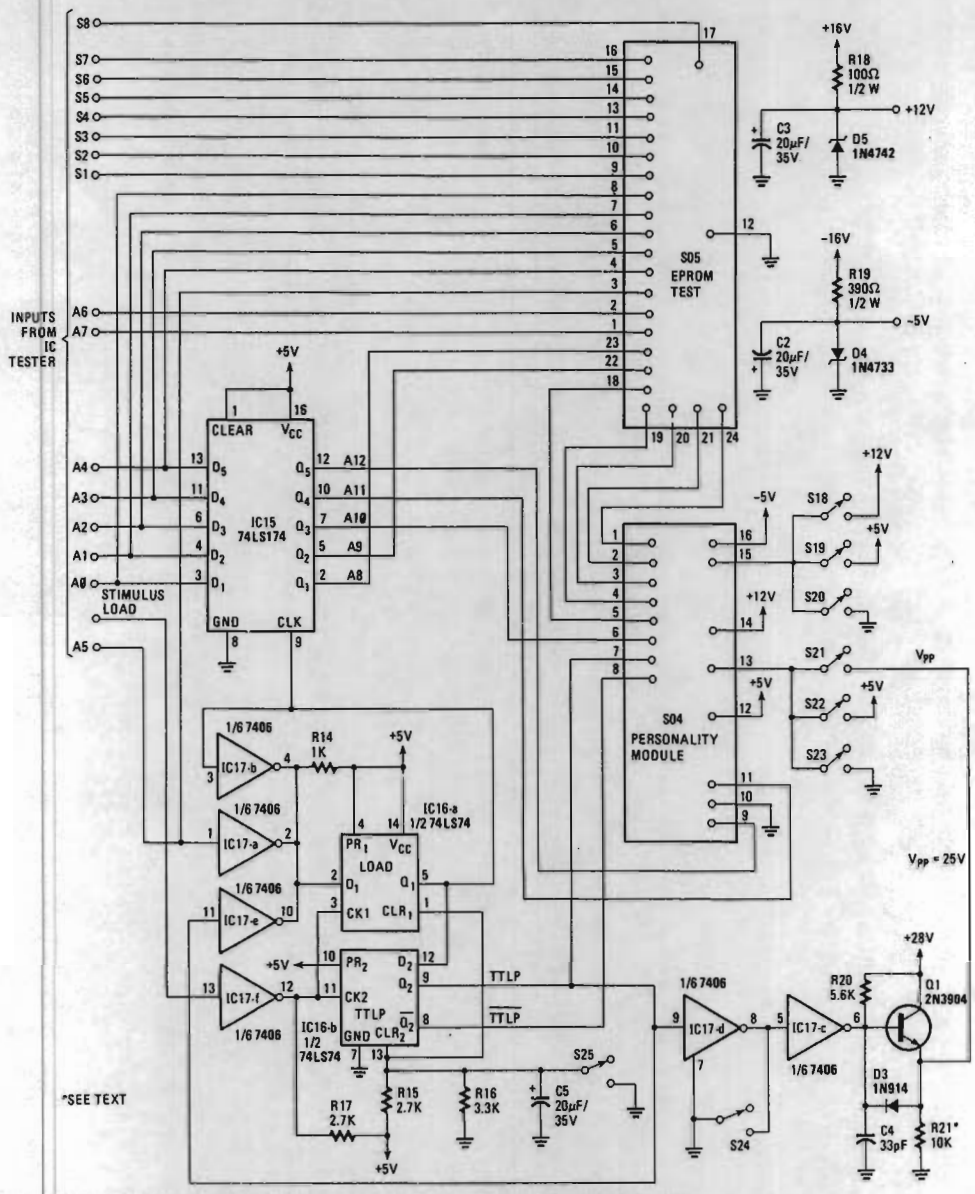


FIG. 11—EPROM PROGRAMMER SCHEMATIC. Note that some EPROM's require a 21-volt (instead of 25–26 volt) programming voltage. When programming such devices, you can either use a variable power-supply, or you can substitute a voltage divider (or trimmer potentiometer) for R21.

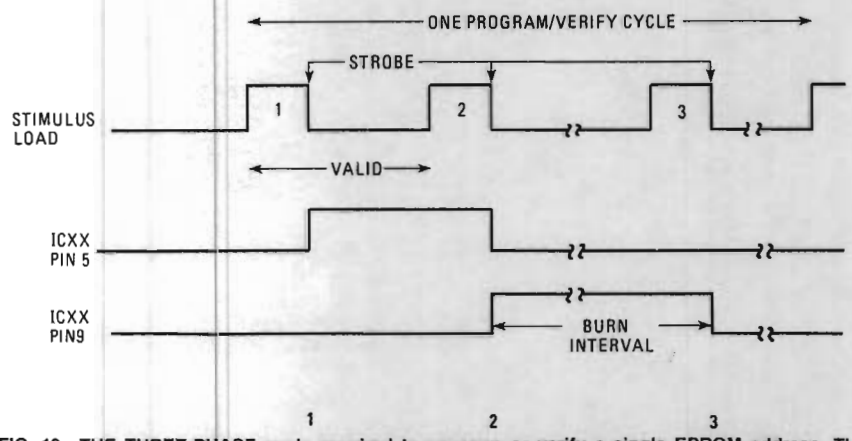


FIG. 12—THE THREE-PHASE cycle required to program or verify a single EPROM address. The program or verify function is selected by the programmer's switches. (See Fig. 13.)

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 either 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 ¼-watt, 5% unless stated otherwise
R14—1000 ohms
R15, R17—2700 ohms
R16—3300 ohms
R18—100 ohms, ½ watt
R19—390 ohms, ½ watt
R20—5600 ohms
R21—10,000 ohms
Capacitors
C2, C3, C5—20 μF, 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, A0 through A7, are connected to the most-significant byte of the stimulus latches and five of those (A0–A4), 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

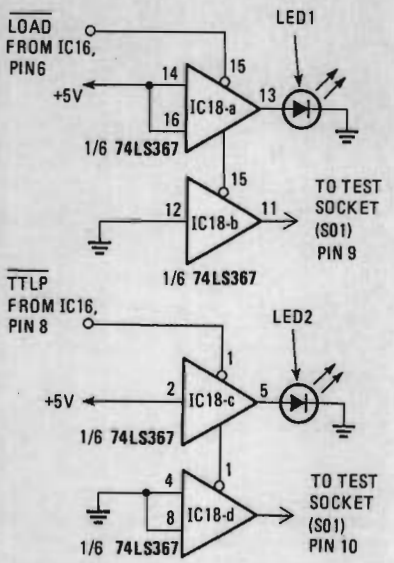


FIG. 14—FOR A VISUAL INDICATION of the programmer's current phase, you can add this diagnostic logic.

program/verify cycle will contain the upper EPROM address on the lines assigned to test-socket pins 9–13. The A5 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 A5 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 A5 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 A5 low will guarantee that the programmer goes to the idle state. The software can use that principle to implement a "restore" function.

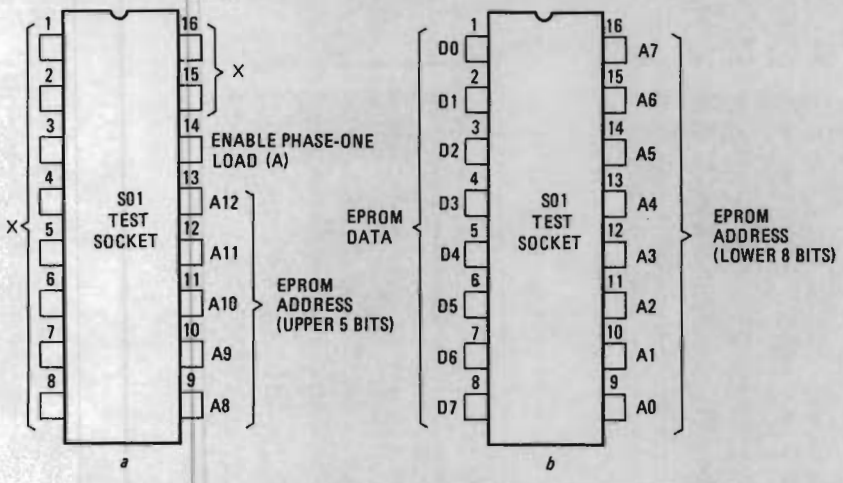
For the remaining tests, you will need to insert a personality module for a 4K EPROM and set up the DIP switches to program. Make sure the switches agree with the personality-module style (see Fig. 13) and leave the EPROM socket empty. Send stimuli which will load various bit patterns into the address 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 5 – 10*	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).

p69 11/84 2:87



FUNCTION	STIMULUS PATTERNS
WRITE 56 INTO ADDRESS 398	2300 – PHASE ONE 9856 – PHASE TWO 9856 – PHASE THREE (398 = 56)
WRITE 74 INTO ADDRESS 39A	2300 – PHASE ONE 9A74 – PHASE TWO 9A74 – PHASE THREE (39A = 74)

c

FIG. 15—THE SOFWARE 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 ½-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 TTLP.

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, ± 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 TTLP 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, ±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**