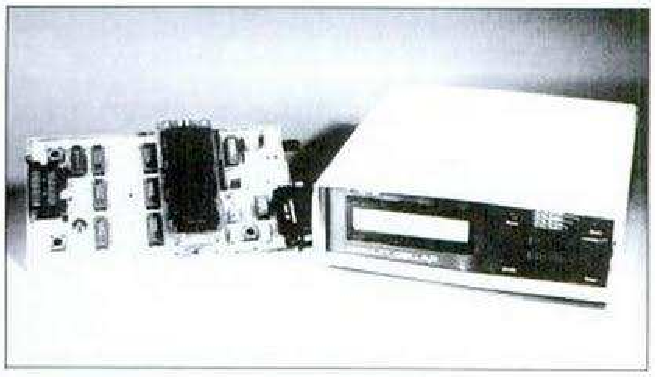
7

BUILD THE CIRCUIT CELLAR IC TESTER

PART 1: HARDWARE

This versatile tester can save you hours of troubleshooting when building and debugging electronic systems



Having designed and debugged many electronic systems, I have seen more than my share of defective ICs. I have also wasted more time than I care to remember discovering that my latest creation was not deficient, but that one of the factory-fresh ICs I put in it was in fact defective. You'd think they'd test them, wouldn't you?

An IC tester can provide both time savings and increased confidence when building and debugging electronic systems. In fact, finding defective ICs before manufacturing an electronic product can also save a considerable amount of money by minimizing the labor and board damage costs involved with reworking electronic boards.

For the most part though, IC testers are used for repairing failed electronic circuits. My latest example was my home: While I was preparing this project, lightning struck my house and practically everything got blitzed. If it were not for my IC tester's help in finding the 29 blown chips in my home-control and automatic-lighting system, I'd still be sitting in a dark, dead house (I thought I had added every preventive measure I could, but I can see we'll need another project on transient protection). I was especially thankful that it could successfully test open-collector driver chips—a problem for most economical testers.

Having an IC tester saved my day, and it may be something you have always needed, too. In this part of Chapter 7, I will describe the design and construction of a digital IC tester with tutorial emphasis on the thinking I had to go through in the process of building it. In Chapter 7, part 2, I will conclude with a discussion of the specific operation of this tester and its advanced software.

Design Considerations

The first step in designing any project is to carefully consider and define what the device is to do. For the IC tester, I first looked at units already on the market and noted their features, prices, deficiencies, and benefits.

I found a price range that varies from less than $200 to several thousand dollars. They also vary considerably in their operation and capability. The low-cost units are generally bus-specific—plugged into a computer slot (Apple II or Commodore 64)—and include operating software. Up the scale from those are the stand-alone—but relatively "dumb"—IC identifiers. With these, if you put a good chip into the socket, a two-, three-, or four-digit number indicating its identification appears on the seven-segment LED display.

The low-end (less than $1000) testers I found have fixed device libraries and perform only simple digital tests (i.e., no AC-parametric tests and no logic-threshold tests). Most, however, indicate that they do provide "periodic" library updates as new standard parts become available.

The high-end testers, costing several thousand dollars, allow some AC-parametric testing, threshold testing, and testing of analog ICs. While they are probably incapable of verifying complete compliance to manufacturers' data sheets, they certainly come close. They can help identify chips with marginal timing specifications. The cost of these devices (including the cost of maintenance, special adapters, and new device support) makes them prohibitive to ordinary users; such devices typically find their home in large corporations with special testing requirements (often those involved with military or aerospace applications).

Flexibility at an Economical Price

My goal in developing the IC tester was to provide as much capability and flexibility as possible in an affordable device that can be used by small businesses and electronic experimenters.

Certainly, economics played its part in requiring compromises in the design. I decided that AC-parametric testing and threshold-level testing would put the device into a higher-price category than I was targeting, so these features were the first to go. Then, I needed to determine what the user interface should be like.

One possibility was to design a card that plugged into an IBM PC slot, with an external test box connected by a cable. This approach would let me develop and include PC software permit-

Copyright © 1987 Steven A. Ciarcia. All rights reserved.

Copyrighted material

ting users to develop tests for their own devices. This would include standard devices not yet in the master library and custom devices, like programmable array logics. Unfortunately, this limited the use of the tester to owners of PCs or compatibles (with a free backplane slot and a long extension cord), and the tester would hardly be portable.

Another possibility was to configure the tester to connect to a dumb terminal, or to any computer with terminal-emulation capability, via RS-232C. While this would broaden the number of potential users of the tester, and would give the tester a little more flexibility, it would also take away the flexibility of user-generated device tests unless that extra (and I might add, very intensive) software capability was provided within the tester.

Finally, I could choose the pure stand-alone approach. Such a configuration would be a self-contained portable tester with its own display and some form of entry panel. Even though it's an easier concept, a stand-alone unit would be more expensive to build and would potentially have the same limitations as terminal-based testers unless it also contained the "smarts" of a larger computer.

Three Units in One

After considering the various circuit possibilities, I concluded that my IC tester should support all three modes of operation. With only a slight increase in hardware complexity, I could present a single design that operates in different ways depending upon which peripheral components and software you install (see photo 1). The operating configurations are called PC-host mode, terminal mode, and stand-alone LCD mode.

The PC-host and terminal modes simply require a serial port for operation. In terminal mode, the tester presents all statements regarding test functions and results on the video terminal's display. The PC-host mode is similar, with the exception that it has the added flexibility of letting you directly modify and extend the device library.

In the stand-alone LCD mode, the tester shows device parameters and data on a 2-line by 20-character LCD. (It should be noted that the LCD is optional; you can operate the tester in the other two modes without it.)

In essence, the stand-alone LCD mode provides a portable (i.e., battery-operated) IC tester suitable for testing any chips that are precoded within its extensive EPROM-resident device library. (The Revision 1.0 library currently contains about 600

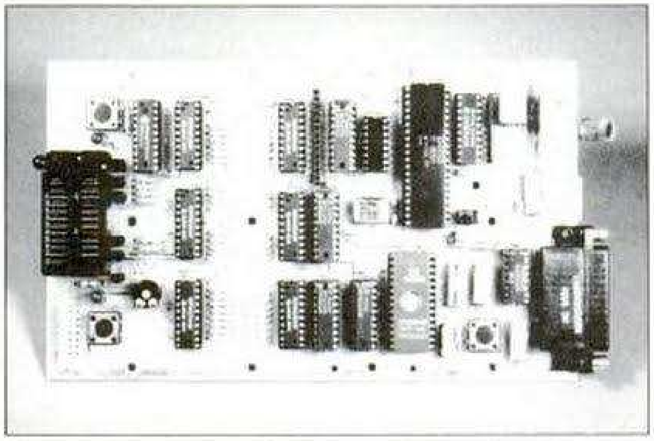


Photo 1: *The prototype IC tester printed circuit board configured for terminal operation. The IC under test is inserted into a special zero-insertion-force socket on the left side, and test information and menu selections are displayed on a terminal connected through the DB-25 connector on the right side.*

74xx00-series and CMOS 4000-series chips.)

The terminal mode provides a menu format intended to maximize the information displayed, while the PC-host mode converts this otherwise stand-alone piece of hardware into an interactive and configurable diagnostic tool with the intelligence of a full computer.

Testing Logic ICs

How do you go about testing ICs? Certainly, I had to answer this question before I could design the tester.

Testing 7400-series logic devices appears relatively straightforward (I didn't consider AC and voltage threshold checking for reasons of economics). To test a two-input NAND gate, for example, you merely set specific logic levels on the gate inputs and check that the outputs are what they are supposed to be.

The process involves a series of test vectors. A test vector is a pattern of bits (0s and 1s) applied to the inputs of the device under test (DUT), to which the DUT responds with a response vector (a pattern of bits on the DUT's outputs). You then compare the response vector from the DUT to the expected response vector, with bit differences indicating pin failures.

You can specify any number of test vectors for a device, allowing you to test the chip as completely as you desire. For each test vector specified for a device, you must also specify a corresponding expected response vector. Since there are cases when some outputs of a device may be in an unknown state, you must also provide a "don't care" mask for each expected response vector, indicating which bit comparisons the tester should ignore.

One significant difference between my IC tester and others in the same price range is that mine does a full-function logic test using as many vectors as necessary to exercise all logic possibilities on the test device. Most inexpensive testers don't do this.

So Many Logic Families

Unfortunately, real-world electronics doesn't quite follow theory. Specifying test vectors is only part of the job. Dealing with all the electrical parameters of the various IC logic families is the real problem.

Since its initial development and introduction by Texas Instruments, the 7400 series of ICs has become an industry standard—at least in terms of device functions and pin-outs. These chips are composed of a large variety of SSI-, MSI-, and LSI-logic building blocks, which designers put together to produce the desired functions.

The original 7400-series family consisted primarily of simple functions, like gates and flip-flops. These were adequate for many applications, but designers kept demanding devices with increasingly greater complexity and functionality.

IC technology did not stand still as designers needed more devices with higher speed and lower power. These requirements led to the introduction of the 74H00-series (high-speed) and 74L00-series (low-power) devices. For the most part, these new series maintained the device pin-outs established by the standard-TTL predecessors (the 7400 series). However, the 74H00-series devices consumed substantially more power than, and the 74L00-series devices were slower than, the standard 7400-series devices.

As the technology improved, even more families appeared. A faster family using Schottky technology was established, the 74S00 series, along with a popular low-power Schottky family, the 74LS00 series.

Eventually, the very-low-power CMOS devices that had been manufactured with 4000-series numbering shifted over to the more popular 74xx00-series pin-out and numbering scheme with the introduction of the 74C00-series family of devices. These devices were slow and had low-current-drive outputs, but

Copyrighted material

they filled a niche in designs requiring extremely low power consumption.

Other families include 74ALS (advanced low-power Schottky), 74AS (advanced Schottky), 74HC (high-speed CMOS), 74HCT (high-speed CMOS, TTL-compatible), 74AC (advanced CMOS), 74ACT/74AHCT (advanced CMOS, TTL-compatible), and 74F (Fairchild advanced Schottky).

Simple Concept, Tough Trade-offs

Digitally speaking, the logical parameters of a 74xx00 are the same regardless of its family, and you could easily be misled into thinking that we are designing a digital tester. However, each of these families has analog characteristics that differ from the other families. The IC tester is actually more an exercise in analog design. Let me explain.

Typical differences between logic families are power consumption, speed, output current drive, input current loading, input transition thresholds, and output voltage swings. Comparisons of some of these parameters for a 74x00 quad NAND gate from several families are shown in table 1. (While the parameters specified in table 1 for the 74x00 devices do not apply to all devices within the respective families, they are representative of the majority of the devices).

In effect, table 1 shows the wide variations of input and output parameters that the ideal IC tester must support. Low-level input currents range from 1 microampere to 2 milliamperes (and much higher on some device inputs), and low-level output currents range from 360 μA to 20 mA.

The tester's ability to identify a device presents an important consideration. If the tester is designed for 74ALS or 7400 "straight" TTL, you might smoke a 74C chip if you inserted it into the tester operated at the current levels of those devices.

Any truly general purpose (read usable) tester must accommodate the wide ranging voltage and current parameters of all the families. Since the tester may not know at the outset what device is installed in the ZIF (zero insertion force) socket (remember, one of the modes is to identify unmarked chips), it cannot make any assumptions as to which pins are inputs and which are outputs.

The tester requires a certain amount of trial and error to identify an unknown device, and it must employ current-limiting resistors between the DUT (in the ZIF socket) and the IC tester's vector-generation circuitry (for when a DUT and tester output are connected together).

Also, while most devices have totem-pole outputs, some have tristate, open-collector, or open-drain outputs. The tester must be able to pull tristate outputs high and low when they are in the high-impedance state to verify the state, and it must also be able to pull open-collector and open-drain device outputs high and low to verify proper operation.

The catch-22 is to determine a resistor value that will support the input and output current specifications for all the device families to be tested, yet not overstress the DUT. If you go strictly by the book, no single current-limiting resistor value works for both inputs and outputs in all families.

The device specifications provided in table 1 are the manufacturer's recommended operating conditions (ROCs). Looking further into the data sheets, however, we find more information regarding what the chips can do if they have to, such as limited-duration short-circuit output current.

In effect, if we take advantage of our regulated testing environment, we can stretch the ROC a little to choose a resistor that presents the best compromise for handling all the logic families. Think of it as the electronic equivalent of poetic license.

All things considered, I found that the resistor value should be in the 390- to 421-ohm range. Since 390 ohms is the nearest

Copyrighted material

standard resistor value (5 percent tolerance), I chose it for the tester. (After I built the tester, I substituted all standard resistor values between 300 and 430 ohms, inclusive, and verified that the 390-ohm choice provides the best overall performance.)

How It Works

After determining the above, I had one more hurdle. The tester needed to be able to apply virtually any number of test vectors to the DUT without losing the device's state from the previous vector—and without causing undo stress on the DUT (i.e., without keeping any of the DUT outputs in a high-current output mode for an extended period of time). I solved this with what I like to refer to as a combinatorial-latch circuit.

Each ZIF-socket pin typically has three circuit connections to the IC tester (see figure 1). One connection (connection A) is to

Table 1: *Comparison of specifications for various 74xx00 devices. (Subscript identifiers are IL—input low, IH—input high, OL—output low, and OH—output high.)*

Device name	I_{IL} max	I_{IH} max (μA)	V_{IL} max (V)	V_{IH} min (V)	V_{OL} max (V)	V_{OH} min (V)	I_{OL} max (mA)	I_{OH} max (mA)
74LS00	−0.4 mA	20	0.8	2.0	0.5	2.7	8.0	−0.4
74H00	−2.0 mA	50	0.8	2.0	0.4	2.4	20	−0.5
74L00	−0.18 mA	10	0.7	2.0	0.4	2.4	3.6	−0.2
74S00	−2.0 mA	50	0.8	2.0	0.5	2.7	20	−1.0
74AS00	−0.5 mA	20	0.8	2.0	0.5	2.5	20	−2.0
74ALS00	−0.1 mA	20	0.8	2.0	0.5	2.5	8.0	−0.4
74HC00	−1.0 μA	1.0	1.2	3.15	0.33	3.84	4.0	−4.0
74HCT00	−1.0 μA	1.0	0.8	2.0	0.33	3.84	4.0	−4.0
74F00	−0.6 mA	20	0.8	2.0	0.5	2.7	20	−0.36
74C00	−1.0 μA	1.0	1.5	3.5	0.4	2.4	0.36	−0.36
7400	−1.6 mA	40	0.8	2.0	0.4	2.4	16	−0.4

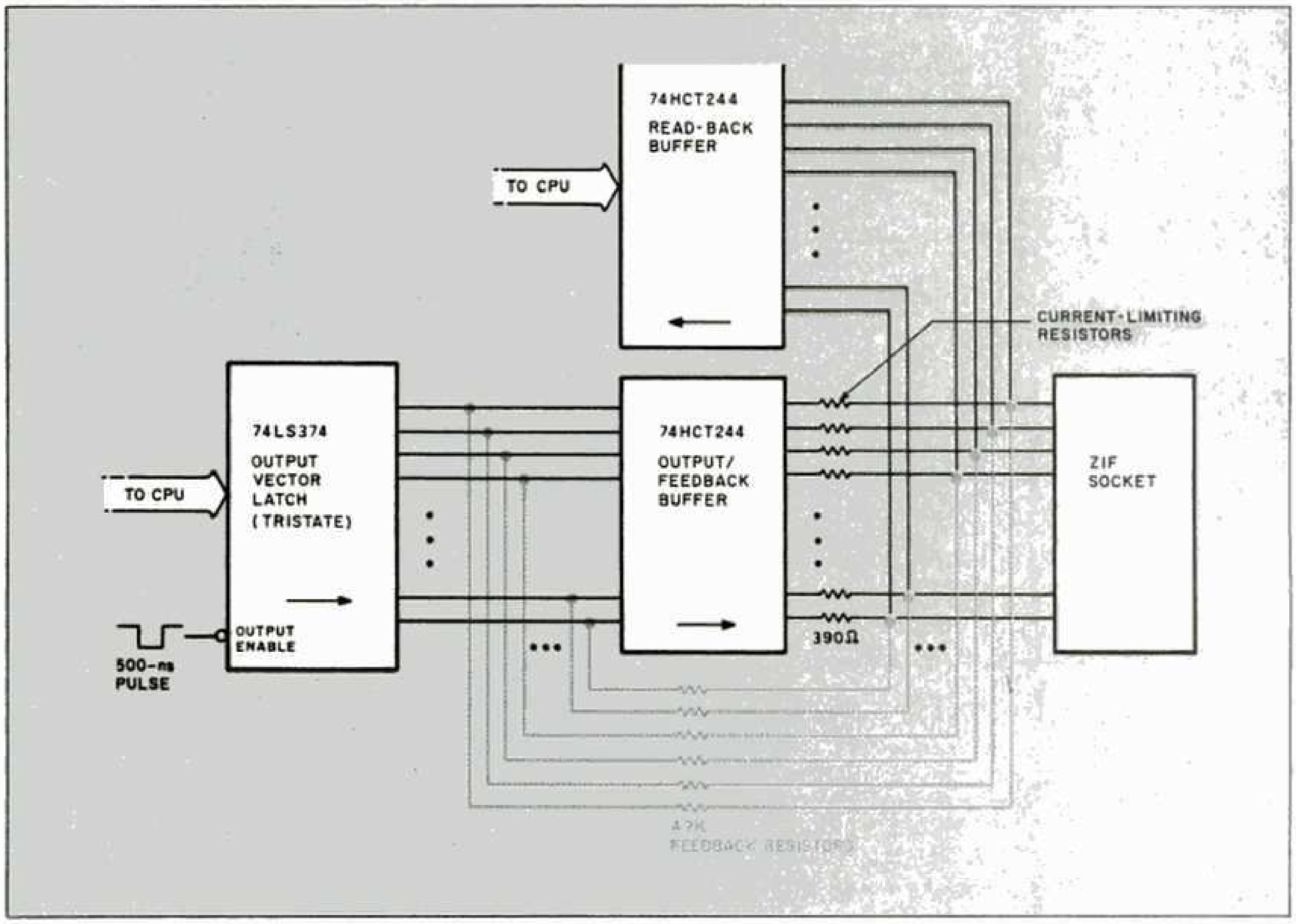


Figure 1: *Diagram of the IC tester's combinatorial-latch circuit. The zero-insertion-force socket holds the device under test.*

Copyrighted material

The DB-25S connector provides the RS-232C interface connection to an IBM PC or any dumb terminal.

an output of a 74HCT244 buffer—the feedback buffer—through a series 390-ohm current-limiting/load resistor. Another connection (connection B) is to the corresponding input of the same 74HCT244, through a 4.7-kilohm series feedback resistor. The 74HCT244 input is also connected to an output of a 74LS374 tristate latch.

The final ZIF-socket-pin connection (connection C) is directly to an input of another 74HCT244 tristate buffer—the read-back buffer. By reading the 74HCT244 read-back buffer, the processor can determine the logic levels of the DUT pins (the ZIF-socket pins).

The IC tester sends a test vector to the DUT by writing the desired bit pattern into the 74LS374 latch, while the latch's outputs remain in the high-impedance state. The system then enables the outputs of the 74LS374 (i.e., they are allowed to go active) for a period of 500 nanoseconds, applying the test-vector bit pattern to the inputs of the feedback 74HCT244 buffer.

During the 500-ns 74LS374-enable period, the relatively high value of the feedback resistors (4.7 kilohms) ensure that the 74HCT244 inputs will see the test-vector logic levels from the 74LS374, regardless of the logic levels present at the DUT pins.

Within a few nanoseconds (i.e., propagation time) of the time the feedback 74HCT244 first sees the new logic levels from the 74LS374, the same logic levels will appear on the outputs of the 74HCT244; these logic levels will remain on the 74HCT244 outputs for the duration of the 500-ns pulse.

If a DUT output in the ZIF socket is in the opposite logic state as the corresponding 74HCT244 output, the resistor between the 74HCT244 output and the DUT pin will present a load to the DUT output, possibly causing it to go into its "overdrive" mode in an attempt to retain its desired output logic level. The overdrive operation will continue until the end of the 500-ns pulse, when the 74LS374 outputs are finally disabled, returning to their high-impedance state.

When the 74LS374 outputs are disabled, the only inputs to the feedback 74HCT244 will be from the DUT feedback resistors. Since the feedback buffer is a 74HCT-series device, it presents negligible input current loading (about 1 μA), so the voltage levels reaching the 74HCT244 inputs through the feedback resistors will be nearly the same as those at the corresponding DUT pins.

If the voltage coming through a feedback resistor to the 74HCT244 is the same logic level as that presented previously by the enabled 74LS374 output (the case when the DUT pin is an output of the same logic level or when the DUT pin is an input), the 74HCT244 output will remain unchanged. Thus, the logic level is combinatorially latched by the 74HCT244.

If the voltage appearing at the 74HCT244 input from the feedback resistor is the opposite logic level of that presented previously by the 74LS374 (which is the case when the DUT pin is an output of the opposite logic level), the 74HCT244 will see the new logic level at its input and change its output to match. When this occurs, the 74HCT244 output then matches the output of the DUT pin, eliminating the loading that was present. Again, the new logic value will be combinatorially latched by the 74HCT244 using the feedback loop.

You can see that the loading duration on a DUT output will essentially be the duration of the enable pulse—only 500 ns. This keeps potential chip stress to a minimum, while verifying the ability of device outputs to operate properly under load conditions.

The IC Tester Hardware

The schematic for the IC tester is shown in figure 2. The 8031 single-chip microcontroller (IC1) is the brains of the tester. The firmware to run the tester is provided in an EPROM at IC6. The current standard device library (version 1.0) is supplied on a 27256, but IC6 can accommodate several EPROM types, including 2764, 27128, and 27512 devices. The type you would use is determined by the JP1's jumper configuration.

The ZIF socket (IC17) is an Aries universal socket. This specific socket supports devices up to 24 pins, having either 0.3- or 0.6-inch DIP-package widths. When you insert devices into the ZIF socket, you bottom-justify them.

Unfortunately, one problem with using a single ZIF socket on a tester is configuring the power pins for the DUT. Most ICs conform to the standard diagonally opposite corner-pin power/ground configuration: pins 24/12, 16/8, and 14/7. However, a number of devices have oddball power and ground pin-outs. These include 14-pin ICs with ground on pin 11 and power on pin 4, 16-pin ICs with ground on pin 12 and power on pin 5, and 16-pin ICs with ground on pin 13 and power on pin 5, among others (there are also devices with two power pins to support voltage-level conversion).

After reviewing the devices in each oddball pin-out category, I chose to support the two categories with the most devices: 14-pin devices with ground on pin 11 and power on pin 4 and 16-pin chips with ground on pin 12 and power on pin 5. This is, of course, in addition to supporting devices having corner power and ground pins. (In the stand-alone identify-unmarked-chip operating mode, the tester will successfully identify only corner-pin-powered chips.)

The DB-25S connector provides the RS-232C interface connection to an IBM PC or any dumb terminal. The connector is configured as a DCE (data communication equipment) device, allowing you to use a straight-through cable. You need only three pins on the connector (pins 2, 3, and 7—receive, transmit, and signal ground, respectively), but I've hard-wired the DTR (pin 6) handshaking line to a logic high for terminals that need it.

The IC tester has push buttons and some switch-selectable options. A four-position DIP switch (SW1) is used for several purposes, including data-transfer-rate selection, PC-host/terminal mode selection, and 74Cx mode selection (to be described next month). Push buttons PB1 and PB2 are for supporting stand-alone mode operation. PB1 is the identify button, and PB2 is the retest button.

J3 is the connector for the optional LCD, which uses the 8031's P1 connector as its data bus. I chose the P1 bus as the LCD's driver to meet the LCD's (relatively slow) timing requirements. The 74LS139 (IC7) is the address-decoding circuit for accessing several devices on the tester. It decodes the ZIF tristate latches (IC8 through IC10) and read-back buffers (IC14 through IC16), as well as the power/ground transistor latch (IC19).

The 74LS139 also provides a special signal that enables the outputs of the 74LS374 tristate latches for approximately 500 ns (the 8031 WR\ strobe duration), transferring the latched 74LS374 bits to the combinatorial latches formed by the 74HCT244s (IC11 through IC13) and their associated feedback resistors.

For the tester's buffers (IC11 through IC13), I chose 74HCT devices instead of 74LS (or other family) devices. Members of this family drive their outputs close to the power and ground rails, can source a lot of current, and provide negligible load on the resistor-feedback circuit. Similarly, the read-back buffers (IC14 through IC16) are 74HCT devices to keep loading to an

Copyrighted material

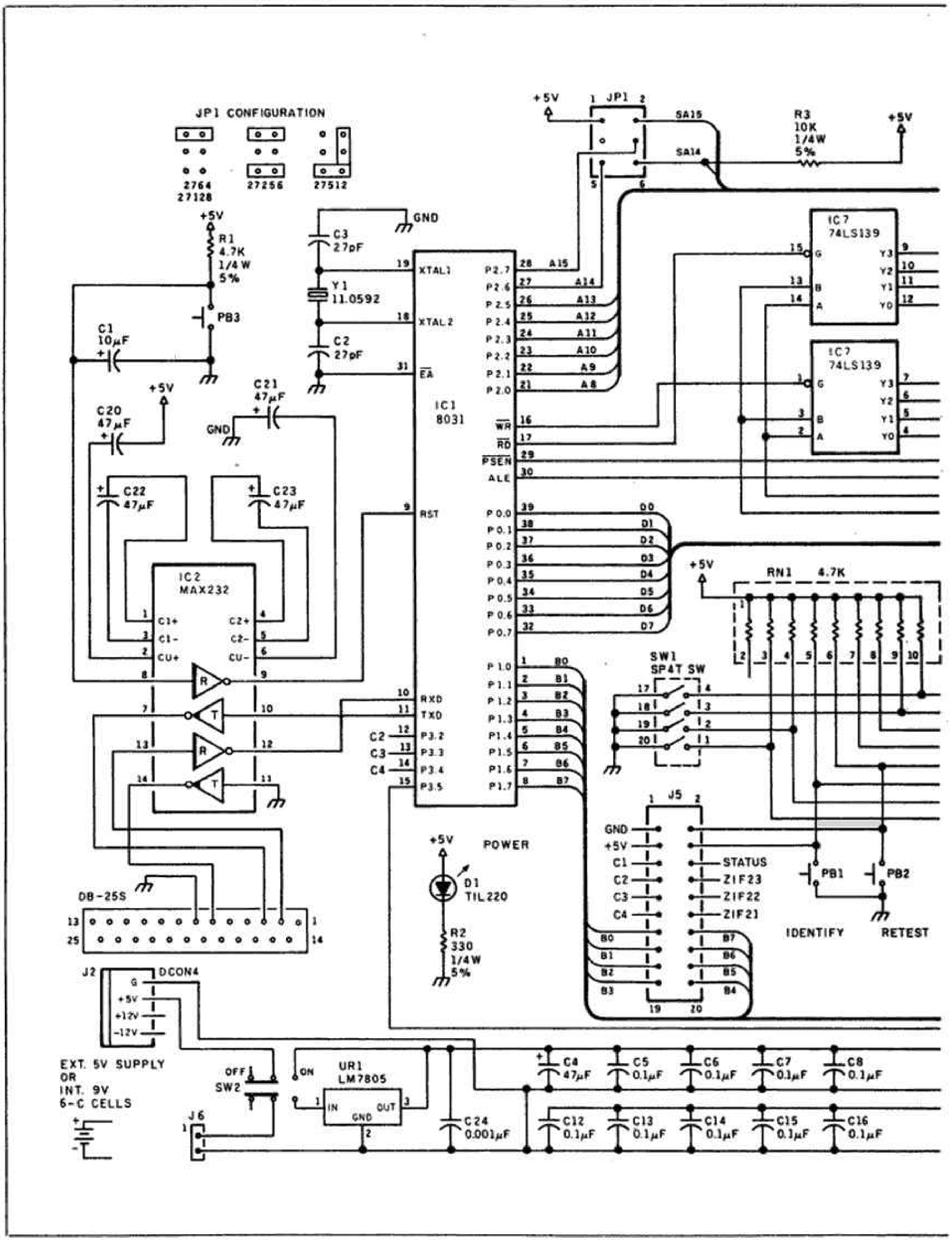
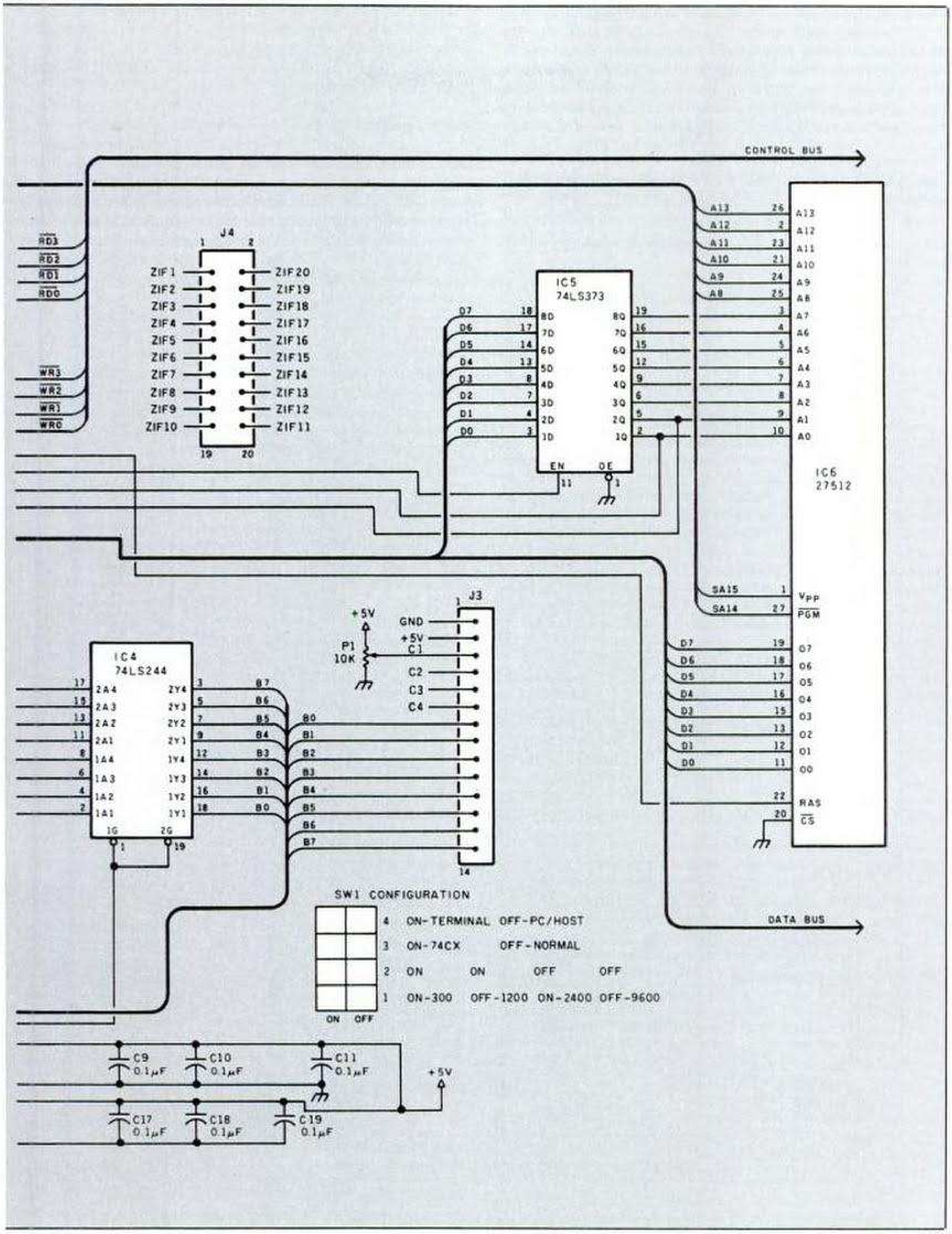


Figure 2: *Schematic for the Circuit Cellar IC tester.*

Copyrighted material



Copyrighted material

absolute minimum (do *not* substitute 74LS devices).

The discrete transistors (Q1 through Q6) provide the power and ground switching for the ZIF socket (IC17). Pin 24 of the ZIF socket is connected directly to +5 volts, eliminating the need for an additional transistor. The PN2907s (Q3 through Q6) are for turning on power (+5 V) to various ZIF-socket pins (9, 19, 20, and 22), while the PN2222s (Q1 and Q2) are for turning on ground to two of the ZIF-socket pins (12 and 16).

The 74HCT374 latch (IC19) controls the transistors. As mentioned earlier, 74HCT devices can source and sink current equally well. This fact made the 74HCT374 a good choice for driving the transistors, since it can handle the transistor base currents equally well for the ground switches (high 74LS374 outputs) and the +5-V switches (low 74LS374 outputs).

The tester has two LEDs. D1 is merely a power-on indicator that lights whenever power is applied. D2 is a software-controlled status LED used to indicate when the device is operating in an RS-232C mode (PC-host or terminal, LED on) or a stand-alone mode (LED off).

Experimenters

While you can order printed circuit boards and kits for the Circuit Cellar IC tester, I encourage you to build your own. If you don't mind doing a little work, I will again support your efforts. A hexadecimal file of the executable code for the 8031 Revision 1.0 system EPROM code, suitable for stand-alone or terminal operation, is available for downloading from my bulletin board at (203) 871-1988.

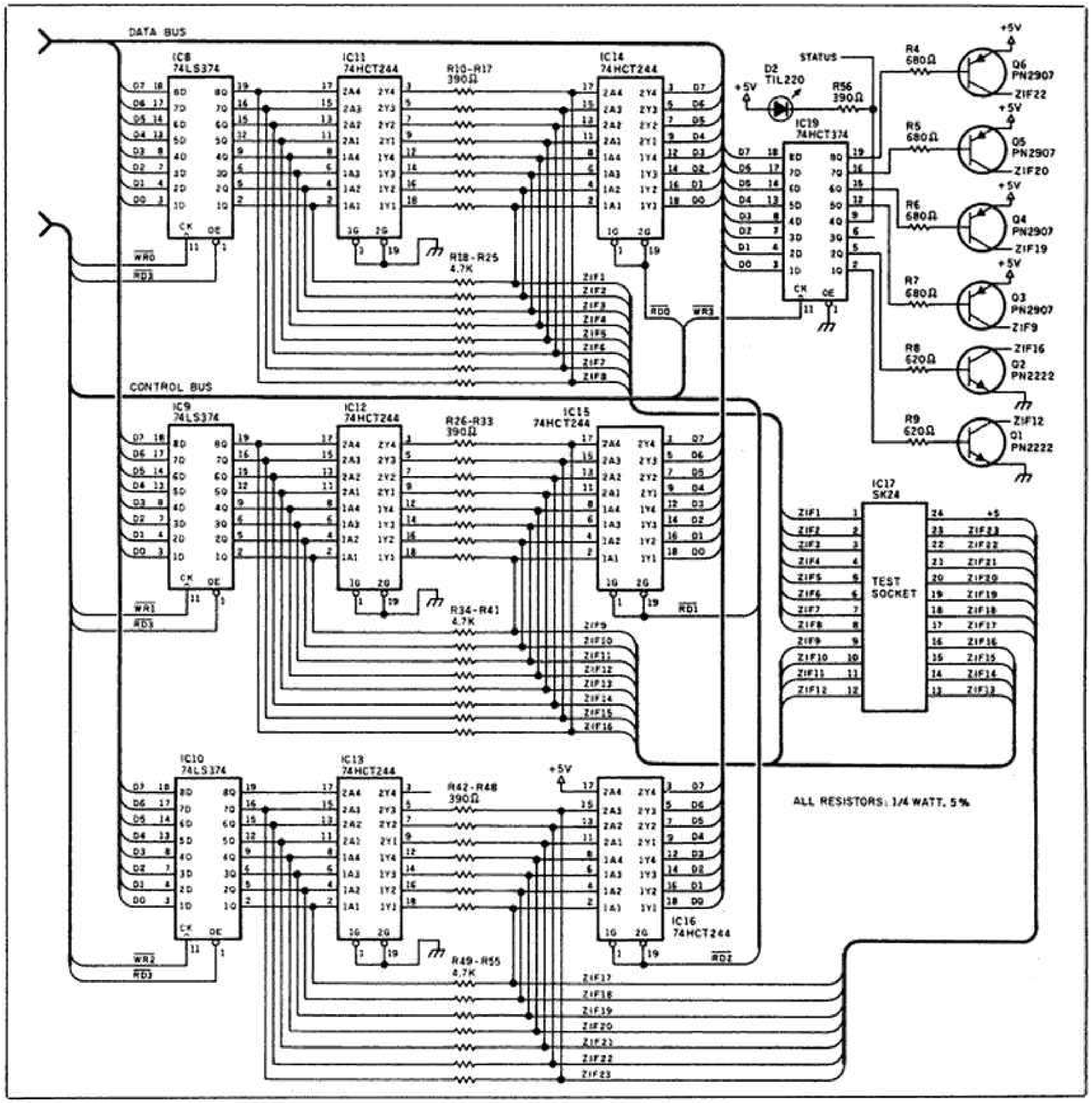


Figure 2: *Continued.*

Copyrighted material

Alternatively, you can send me a preformatted IBM PC 5¼-inch disk with return postage, and I'll put the file on it for you (the hexadecimal file could be used with my CCSEP serial EPROM programmer, for example). Of course, this free software is limited to noncommercial personal use.

In Conclusion

In Chapter 7, part 2, I will present the tester's software, which lets you develop and debug your own test vectors and device libraries.

Special thanks to Roger Alford, Jeff Bachiochi, and William Potter for their work on this project.

Editor's Note: Steve often refers to previous Circuit Cellar articles. Most of these past articles are available in book form from BYTE Books, McGraw-Hill Publishing Company, P.O. Box 400, Hightstown, NJ 08250, (1-800-2-MCGRAW).

Ciarcia's Circuit Cellar, Volume I covers articles in BYTE from September 1977 through November 1978. *Volume II* covers December 1978 through June 1980. *Volume III* covers July 1980 through December 1981. *Volume IV* covers January 1982 through June 1983. *Volume V* covers July 1983 through December 1984.

The following items are available from

Circuit Cellar, Inc.
4 Park St., Suite 12
Vernon, CT 06066
(203) 875-2751

1. Circuit Cellar IC tester experimenter's kit for stand-alone or terminal operation. Contains IC tester printed circuit board, 11.0592-megahertz crystal, programmed 27256 EPROM with Revision 1.0 device library, MAX232 level shifter, Aries 24-pin narrow-format ZIF socket, and manual with complete parts list.
ICT01-EXP $99
2. Circuit Cellar IC tester full printed circuit board kit for stand-alone, terminal, or PC-host operation. Contains IC tester printed circuit board, 8031 processor and crystal, programmed 27256 EPROM with Revision 1.0 device library, Aries 24-pin narrow-format ZIF socket, IC sockets, all board-mounted components and ICs, PC-host software on PC format disk, power supply, and manual.
ICT01-FULL $179
3. Complete Circuit Cellar IC tester kit with stylish enclosure. Full printed circuit board kit with all components, right-angle-mounted enclosure adapter board with ZIF socket and LCD, software on PC format disk, power supply, and manual.
ICT02 .... $349
4. Two-line by 20-character LCD and 14-pin Berg connector for either item 1 or 2.
2x20 LCD $32

All payments should be made in U.S. dollars by check, money order, MasterCard, Visa, or American Express. Surface delivery (U.S. and Canada only): add $5 for U.S., $8 for Canada. For delivery to Europe via U.S. airmail, add $14. Three-day air freight delivery: add $10 for U.S. (UPS Blue), $25 for Canada (Purolator overnight), $45 for Europe (Federal Express), or $60 for Asia and elsewhere in the world (Federal Express). Shipping costs are the same for one or two units.

There is an on-line Circuit Cellar bulletin board system that supports past and present projects. You are invited to call and exchange ideas and comments with other Circuit Cellar supporters. The 300/1200/2400-bps BBS is on-line 24 hours a day at (203) 871-1988.

Copyrighted material

7

BUILD THE CIRCUIT CELLAR IC TESTER

PART 2: SOFTWARE AND OPERATION

Steve guides us on a tour of the software that makes his inexpensive IC tester possible

In part 1 of this chapter, I talked about the design of my IC tester. In this part, I'll talk about its software and operation.

Three in One

To refresh your memory, the IC tester supports three modes of operation: PC-host mode, terminal mode, and stand-alone LCD mode.

PC-host mode requires that you connect the tester to a serial port on an IBM PC or compatible. In this mode, the PC handles all test-vector transfers and comparisons and provides the highest level of flexibility and power.

To operate the tester in terminal mode, you connect it to a dumb terminal or any microcomputer that emulates a terminal (see photo 1). The options are essentially the same as those offered in PC-host mode, although you can use only a fixed, ROM-resident device library.

The stand-alone mode of operation lets you operate the tester with only two push-button switches and a 2-line by 20-character LCD. As in terminal mode, this mode operates only with a fixed, ROM-resident device library. It lacks some features of the other two modes, but it permits device identification (using the Identify push button) and specified-device testing (using the Retest push button). The latter lets you determine specific pin failures on a bad IC and display this information on the LCD.

Much of the flexibility of the IC tester comes from its modifiable and expandable device library. While an IBM PC (or clone) is essential for PC-host mode operation, it is required if you're going to make any system software changes, like adding new chips to the library.

With the exception of a single assembly language serial-port driver, all the software was written in Turbo Pascal on an IBM PC. (While the programs do take advantage of some PC-specific features of Turbo Pascal, you shouldn't have much trouble converting them to other Pascal compilers.)

The Definition of a Test Vector

In order to define test vectors, it is important to develop a straightforward means of describing the vector information. What information do we need to define a device and its test vectors?

The device definition consists of the device name (e.g., 7400), the specific package size (e.g., 14 pins), the locations of the power and ground pins (e.g., 14 and 7), and which pins are inputs, outputs, or tri-state.

A test vector merely specifies the high (1) and low (0) logic levels to be written to the pins of the device under test (DUT). A test vector written to the DUT pins is referred to as an output vector.

To determine if the DUT responded properly to the output vector (i.e., to make sure outputs switched as expected and to verify that no inputs are shorted), the tester must read a corresponding read-back vector from the DUT and compare this to an expected read-back vector. Each complete test vector consists of an output vector and an expected read-back vector.

The format for specifying the vector-definition modules is shown in table 1. The order of the different line types is important, though you may freely intersperse comment lines. (Like many assemblers, all characters on a line following an asterisk are ignored by the test-vector compiler.)

The best way to understand the vector-definition module format is by example. Table 2 shows the vector-definition module for a 7400 quad two-input NAND gate. As its name implies, this device contains four two-input NAND gates (the pin-out is shown in figure 1).

Photo 1: *The Circuit Cellar IC tester shown here is operating in terminal mode, connected to a Tandy DT-100 terminal via the RS-232C port on the top of the tester.*

Copyright © 1987 Steven A. Ciarcia. All rights reserved.

Copyrighted material

In table 2, the first line is the device name. The name can appear anywhere on the line after the pound sign (preceding and following spaces are ignored). As a general rule, you should keep device names as generic as possible. Instead of using the name "74LS00" use "7400," and so on. Since the tester will logically identify both a 74LS00 and 74HC00 as the same chip, it is better to display "7400," or perhaps "74xx00." There are, of course, cases where you can make exceptions.

The second noncomment line of the vector-definition module is the setup line, which has an *S* in the first column. Three numbers with delimiting spaces must follow the *S*; the first number indicates the number of pins the device has, the second indicates the ground pin number, and the third indicates the power pin number. These numbers tell the compiler (and the tester) what the chip's device is. As I described in Chapter 7, part 1, the tester supports six device types (see table 3).

Following the next comment line is the pin-function line, which has an *F* in the first column. This line specifies a pin-function identifier for each pin, with the identifiers being separated by one or more spaces. Valid identifiers are *I* for input pins, *O* for output pins, and *T* for tri-state pins.

The pin-function line also determines the columnization for the remainder of the device vector definition. All 1s and 0s in the test vectors must be aligned under these columns, and the pin numbers in the pin-number line (the next line in the definition) must also be aligned under these columns.

The next line in the vector-definition module is the pin-number line. It has the letter *P* in the first column. This line specifies the device pin numbers used in testing. The numbers must correspond to the pin-function identifiers specified in the pin-function line and must fall in the columns defined by the function identifiers. If the pin number for a column has two digits (e.g., pin 14), either of the two digits can fall in the column.

The next several lines in the vector-definition module are the actual test vectors. The lines beginning with *I* are initial vectors (output vectors), and the lines beginning with *R* are the expected read-back vectors.

For *I* vectors, the acceptable identifiers are 1 and 0, corresponding to high and low digital values, respectively. For *R* vectors, acceptable identifiers are 1, 0, and *X*, with *X* indicating "don't care." (*X* indicates that the tester should ignore the specified pin when comparing the actual read-back vector to the expected read-back vector. If the 1 or 0 bit value of a column does not change from one line to the next, leaving the column blank in the subsequent line[s] implies that the value should be the same as the last value explicitly stated for that column.)

The last line in the vector-definition module is the end line,

Table 1: *Device test-vector definition-module format.*

```
# DeviceName                   * Device-name record
* Comment lines may be interspersed
* for documentation and clarification purposes.
S #Pins Gpin Ppin              * Device-setup record
F I  O  ...T  I  I   * Pin-function record
P p# p# ...p# p# p#  * Pin-number record
I 0  1  ...1  1  0   * Initial (output) vector record
R 1  0  ...1  X  0   * Expected read-back vector record
I...
R...
.
.
.
E                              * End-of-definition record
```

Notes: DeviceName = name of device
* = start of comment area
#Pins = number of pins on the IC
Gpin = ground pin number
Ppin = power pin number
p# = pin number

Table 2: *Device test-vector definition module for the 7400.*

```
# 7400          * Quad two-input NAND
S 14 7 14
*  NAND 1      NAND 2      NAND 3       NAND 4
F I  I  O      I  I  0     I  I  O      I  I  O
P 1  2  3      4  5  6     9  10 8      12 13 11
I 0  0  0      0  1  0     1  0  0      1  1  1
R       1            1           1            0
I 0  1  0      1  0  0     1  1  1      0  0  0
R       1            1           0            1
I 1  0  0      1  1  1     0  0  0      0  1  0
R       1            0           1            1
I 1  1  1      0  0  0     0  1  0      1  0  0
R       0            1           1            1
E  *  end of 7400
```

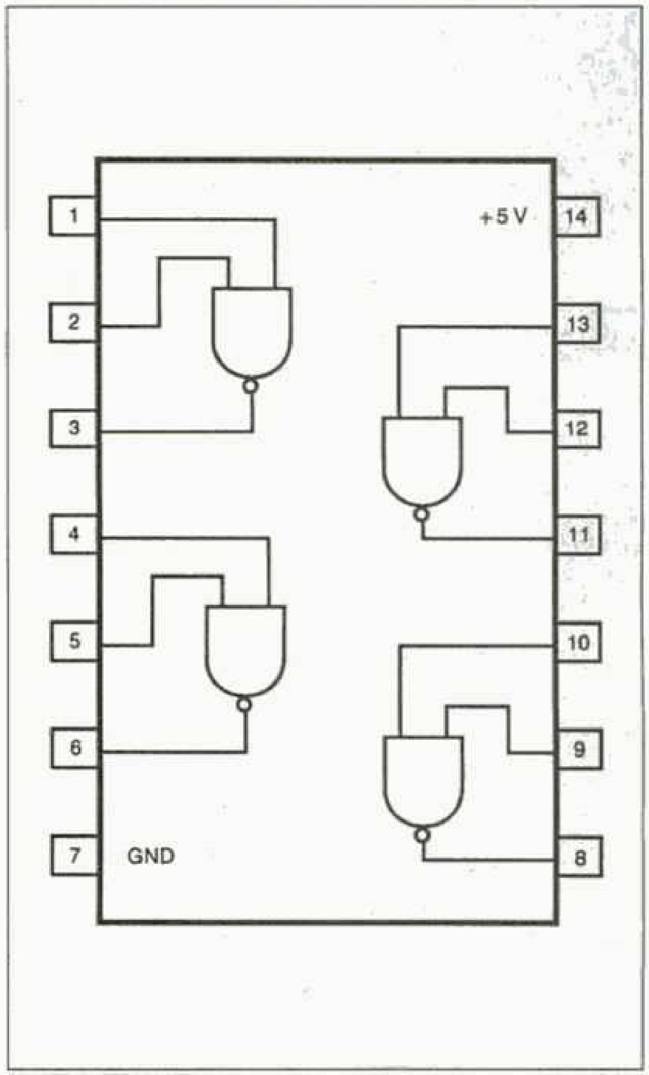


Figure 1: *Pin-out for a 7400 quad two-input NAND gate.*

Copyrighted material

which begins with an *E*. This is the only letter required to specify the end of the vector-definition module.

Finally, there is the issue of logically identical devices that have different part numbers (like 74LS04 and 74LS14). Differences typically lie in some of the special operational parameters, like Schmitt-trigger inputs or improved current drive capability, which cannot be detected by this IC tester.

Functionally identical devices (i.e., the same test vectors would pass on both devices) are declared to be clones of a specific device. An example of this is shown in table 4. The device-name and end lines are the same as the standard vector-definition module, but only the *C* line is found between them, indicating which device it is supposed to be cloned from.

Compiling Test Vectors for Use

The test-vector compaction compiler, VECCPT.PAS, is a Turbo Pascal program that accepts files conforming to the device test-vector definition format described above. It converts the device and vector information into a single compact module that the computer and tester use to test the devices.

VECCPT.PAS uses seven primary arrays to store the compacted vector information. The primary array, `VectorTable`, holds the actual test-vector information, including the device pin-function information (i.e., which pins are inputs, which are outputs, and which are tri-state), the output vector bytes, the input vector bytes, and the "don't care" mask bytes.

Because the ZIF (zero insertion force) socket has 24 pins, tester software uses 3 bytes for pin and vector information for every device, regardless of size. Consequently, the pin-function and test-vector information is stored as if a 24-pin device were being tested.

The "don't care" mask generated by VECCPT.PAS automatically masks the read-back vector pins not associated with the device being tested. For example, if a 20-pin device is being tested, the bits of the 3-byte read-back vector associated with ZIF-socket pins 1, 2, 23, and 24 will be masked by the "don't care" mask (power and ground pins are automatically masked).

Each of the six device types supported by the IC tester has its own associated array for storing device names and pointers into the `VectorTable` array. These arrays are called `DeviceType` arrays.

While the `VectorTable` array uses variable-length records, with each record being the information to support one device, the `DeviceType` arrays use fixed-length records, with each record containing a 9-byte field for the device name (8 bytes for the name and 1 byte for the string size) and an integer (2-byte) field for the `VectorTable` pointer.

Figure 2 illustrates the information stored in the various arrays and how the arrays interact. As shown, device names are stored in the appropriate `DeviceType` array, and the device pin-function and test-vector information is stored in the `VectorTable` array. A pointer in the `DeviceType` array indicates the start of the corresponding vector-information record in `VectorTable`.

The `VectorTable` device record begins with a 2-byte field indicating the number of bytes in the record. The next 3 bytes specify which pins are inputs and which are outputs (set bits are inputs, and cleared bits are outputs).

The following 3 bytes indicate which pins are tri-state (set bits are tri-state). If a pin is indicated as being tri-state, the 1/0 value in the corresponding bit position of the previous I/O definition bytes is irrelevant. By default, VECCPT.PAS specifies unused ZIF (zero insertion force)-socket pins as being tri-state.

Following the 2 record-size bytes and 6 device pin-function definition bytes, the actual test-vector information begins. Each complete test vector consists of 9 bytes in the record. The first 3 specify the output vector, the next 3 specify the expected read-back vector, and the last 3 specify the "don't care" mask.

As VECCPT.PAS executes, it stores device-name, pin-function, and test-vector information into the appropriate arrays. Notice that the program does not need to store device-type information, since a device's type is determined by which `DeviceType` array it is placed in.

Device clones are handled somewhat differently. When a device is specified as a clone of another device (the "original" device), the name of the clone is placed into the next available record of the appropriate `DeviceType` array. The record number of the original device (in the same array) is then determined, and the value 32,767 is subtracted from the record number; this value (always negative) is then stored in the pointer field of the clone record.

Thus, when the operating software finds a negative integer value in the pointer field of a device record, it will know the device is a clone of another device. It then adds 32,767 to the pointer value to get the record number of the original device.

I should point out that when VECCPT.PAS processes a clone, it looks through its arrays to find the named original device. If the specified original device is not found in any of the six `DeviceType` arrays, the software generates an error, and the clone device will not be stored in any array (the compiler would not even know which array should get the clone record). Thus, it is essential that you specify clone devices only after the corresponding original device.

When compaction of the test-vector files is complete, the compacted information is stored in a binary file. (The format of the data stored in the compacted file is shown in figure 3.)

Operating Software

Once the device test vectors have been developed and compiled into a compacted file, we are ready to use the tester for testing and identifying devices. This involves the cooperation of several programs.

First, there's a ROM-resident program on the IC tester. This program is written in 8031 assembly language and handles the three operating modes from the tester's vantage point. Then there's a Turbo Pascal program that executes on the IBM PC (or XT or AT) for operating the tester in PC-host mode.

Finally, another Turbo Pascal program converts the information in the output file produced by VECCPT.PAS into Intel hexadecimal ASCII format. This permits you to download to an EPROM burner. This lets you put new device vector information into the IC tester's ROM for operation in the terminal and stand-alone LCD modes.

Table 3: *The six device types supported by the tester.*

Device type	Number of pins	Gnd pin	+5-V pin
1	14	7	14
2	14	11	4
3	16	8	16
4	16	12	5
5	20	10	20
6	24	12	24

Table 4: *Definition module for the 7437, a "clone" of the 7400.*

```
#7437     * Quad two-input NAND buffers
C 7400    * Clone of 7400 (Dev. type=1)
E         * End of 7437 definition
```

Copyrighted material

Explaining all the software for the IC tester would involve considerably more space than I have available here (see the Circuit Cellar Ink applications publication for additional support materials). While my description here is tailored to the application and use of the IC tester, the user's manual and distribution software contain much source code and go into significant detail describing the process for creating a new device library and testing custom devices.

PC-Host Mode

The PC-host mode of operation is the most powerful of the three modes. I'll start with its description, because the basic testing technique is the same for all three modes.

The PC-host mode provides flexibility in letting you download and use different device libraries and offers test-vector debugging features not available in the other two modes. Functions like `Identify` and `Test Specified Device` differ only in the information displayed and are the same in all modes.

Once you give the PC-host mode operating program the name of the compacted test-vector file and the serial-port number (1 or 2), the software attempts to establish a communication link with the IC tester. If the tester does not respond, the PC will perform two retries (three tries total) before printing an error message and sounding a beep.

Once communication is established, the PC reads the specified compacted test-vector file, downloads it to the IC tester, and displays the version number and a formatted operation menu on the screen. The typical menu offers four device-testing options and two mode-selection options.

The display also shows three status/information lines. The first line, `Device:`, indicates the name of the current or most recent device being tested, or the name of an identified device. The second line, `Message:`, displays messages like `Device Passed` and `Device Not Found`. The third line, `Pin Failures:`,

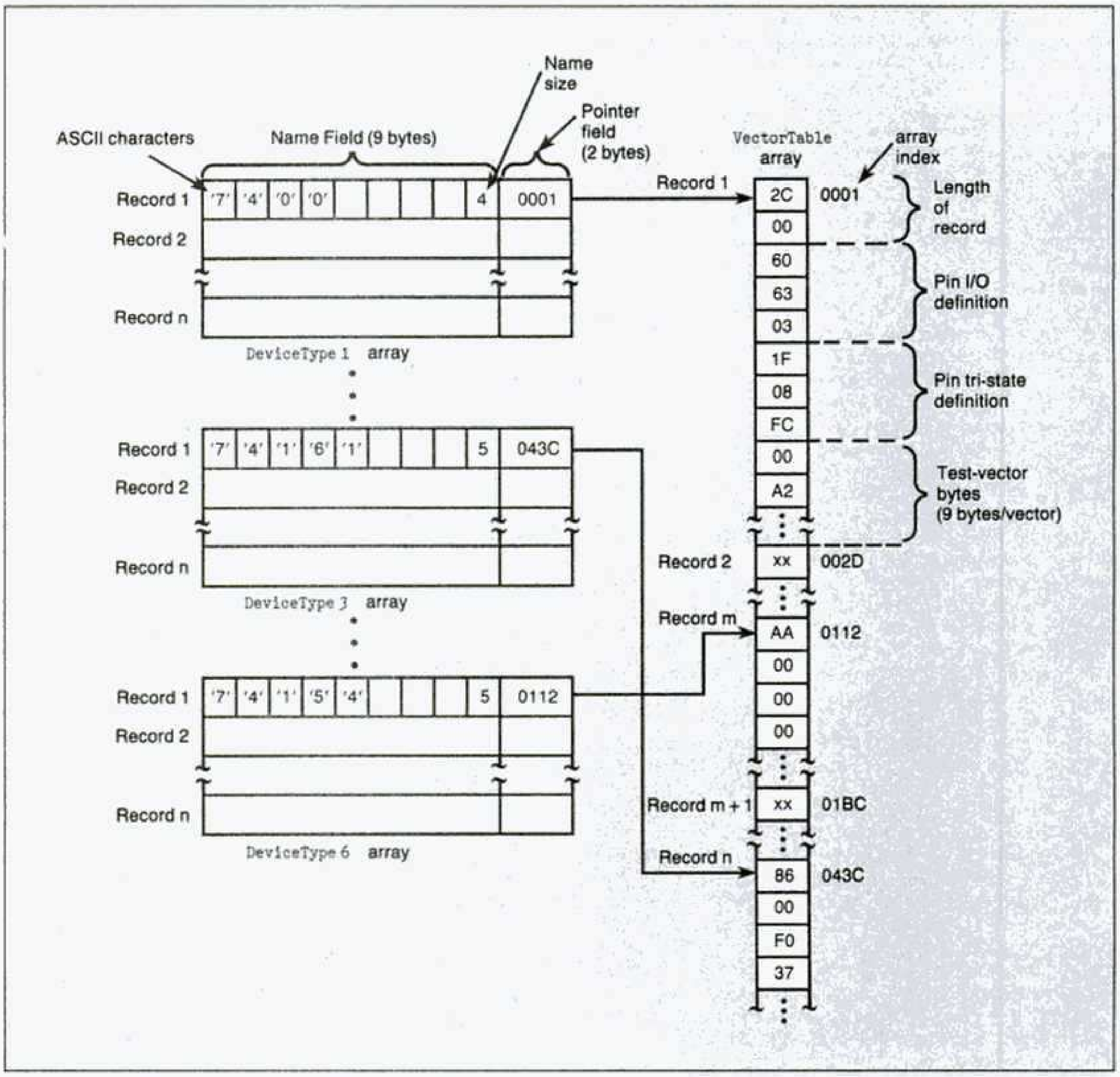


Figure 2: *Illustration of the information storage in VECCPT.PAS's primary arrays.*

Copyrighted material

displays pin numbers that failed vector tests when testing a specified device or an EPROM.

The first menu item, `Identify Device`, tells the tester to attempt to identify the device in the ZIF socket (the device-identification algorithm supports only devices having the corner power and ground pins). To identify a device, the system powers the ZIF socket for a 24-pin device and then applies the first 24-pin device test vector (if any) in the device library to the DUT.

If the read-back vector compares favorably to the expected read-back vector (along with the "don't care" mask), the next vector for the same device is applied, and so on. This continues until the DUT passes all the test vectors—indicating proper device identity—or until a vector failure occurs. If a vector failure occurs, a check is made to see which bits in the read-back vector, if any, are different from those sent out in the output vector. These bits represent pins that must be either output or tri-state pins, and the pin values are noted in an accuracy array.

If the DUT passes all the test vectors, the tester has identified the device; its name is displayed, and control returns to the menu. If the DUT fails a vector, the next device in the 24-pin library is checked.

Testing continues until the DUT is identified or no more 24-pin devices are left to test. If the program runs out of 24-pin devices, it clears the accuracy test array and repeats the same procedure with the 20-pin, then 16-pin, and finally 14-pin devices. Inability to finally identify the part is only the result of the device not being in the library, or because it is defective.

The second menu item, `Test Specified Device`, moves the cursor to the `Device` line. If any devices have already been tested or identified, the name of the last device tested is automatically displayed on the line. If you desire to retest the same part type, press Return (or Enter). If you wish to choose a different device, enter the new device name and press Return to test the DUT.

By telling the IC tester what type of chip is in the ZIF socket, all the test vectors for that device will be applied to the device and checked, regardless of whether they pass or fail. If vector failures do occur, you'll see the pin numbers on the `Pin Failures:` line.

The first two menu items represent the operations you will probably want to do 99 percent of the time and can be done in all three operating modes. Sometimes, however, you may have 2716 or 2732 EPROMs that you would like to verify are blank. Menu items 3 and 4 provide this capability.

In addition to performing a blank check on the EPROM, the EPROM tests also check for shorts on the EPROM input pins. If shorted pins are detected, an error message is displayed and the failed pins are displayed on the `Pin Failures:` line. Since the ZIF socket is only 24 pins, the tester cannot accommodate larger EPROMs.

The third menu selection deals with CMOS logic devices only. As I discussed in part 1 of this chapter, all the standard 74xx00-series logic families except the 74C00 series (and some specific devices within other families) are capable of sourcing and sinking enough current on their outputs for proper operation of the tester.

The 74C00-series devices (and the similar 4000-series CMOS devices) have a problem sinking enough current to switch logic states when an output is pulled up to +5 volts. Most of the tests for the 74xx00-series families attempt to load the device outputs in the direction opposite the expected state (if an output is expected to go low, it is loaded with a pull-up resistor), causing particular problems when testing the 74C00-series family devices when reading outputs that are expected low, but are being pulled up.

The remedy for the 4000-series devices is simple: Write all test vectors for these devices always using a pull-down load on all outputs. In order to keep the 74xx00-series tests the same for all families, however, I had to use a different approach. Menu item 5 lets you `Set 74Cx Mode`.

In this mode, regardless of the original output vector-bit levels, all output vector bits that correspond to device output (non-tri-state) pins are changed to low (pull-down). This allows the 74C00-series devices to pass the generic 74xx00-series tests. You can also select this mode for identifying 74C00-series devices.

The final menu option is `Set Diagnostic Mode`. This option is available only when operating the tester in PC-host mode. It adds an extra line to the bottom of the display, `Vector Failures:`, to indicate which test vectors failed when testing a device.

When testing a specified device (not when identifying a device) in diagnostic mode, the `Device:` line indicates the number of pins the device has, as well as the ground pin number and the power pin number. If the device is a clone of another device, this is also indicated, along with the device name of the original device.

If the device being tested fails, the `Message:` line indicates how many vectors failed (along with the normal failure message), and the `Vector Failures:` line indicates the vector numbers of the first 10 failed vectors (or all failed vector numbers, if fewer than 10 failed). The extra information can prove helpful when debugging new test vectors.

ROM-Resident Control Program

The IC tester's 8031 assembly language control program provides local support for all three modes. A software-readable, four-position DIP switch selects mode and data transfer rate, while a status LED indicates the tester's current operating disposition (a second LED acts as a power-on indicator).

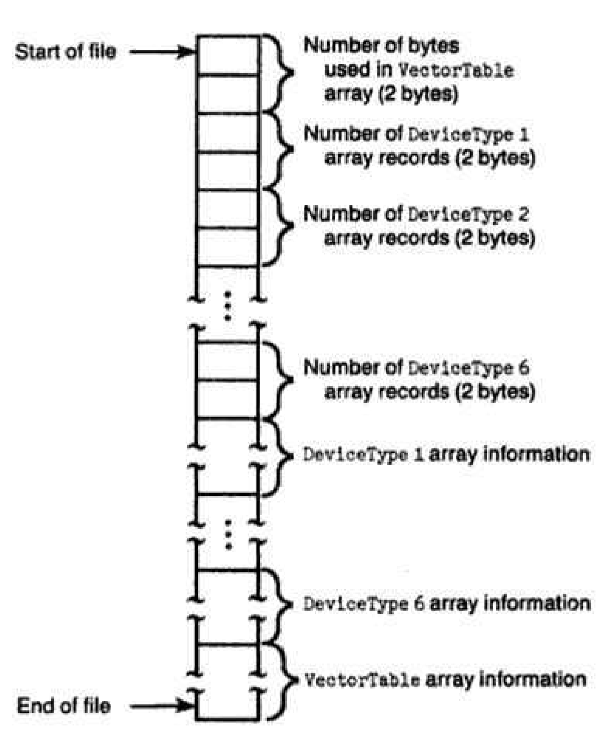

Start of file
Number of bytes used in VectorTable array (2 bytes)
Number of DeviceType 1 array records (2 bytes)
Number of DeviceType 2 array records (2 bytes)
Number of DeviceType 6 array records (2 bytes)
DeviceType 1 array information
DeviceType 6 array information
VectorTable array information
End of file

Figure 3: *Format of the vector-compaction file output by VECCPT.PAS.*

Copyrighted material

Upon power-up or reset (using the on-board reset button), the program initializes the 8031's on-chip ports to turn off all power and ground transistors to the ZIF socket and to place the LCD interface lines in their appropriate default states.

It then generates a brief delay (nominally, 1 second) to provide time for power to stabilize for all devices on the board. Software then checks two of the DIP switches to determine the desired data transfer rate and configures the 8031's on-chip UART to handle serial communications at the specified data transfer rate.

Once initialization is completed, the program checks another DIP switch to see if the user has selected PC-host mode or terminal mode. For terminal mode, the system turns on the status LED (to indicate that a serial operating mode, as opposed to stand-alone mode, is currently enabled) and sends a sign-on message and menu out the serial port to the attached terminal. For PC-host mode, no sign-on message is sent.

In either case, the tester also displays a sign-on message on the optional LCD, if present. In order to select the stand-alone mode, you merely press the "Identify" push button—which is constantly polled during both serial operating modes—and the system will turn off the status LED to indicate stand-alone mode operation. The only way to return to serial mode operation is by pressing Reset.

When operating in PC-host mode, the IC tester's ROM program merely responds to commands from the host. Various commands allow "reset" (power and ground transistors turned off), software version request, power and ground switch setup, and DUT output vector application and read-back vector reading. Terminal mode operation is similar to PC-host mode operation, with the exception that you are restricted to the device library stored in ROM, and the diagnostic mode described earlier is unavailable.

Stand-alone operation requires no connection to the serial port, but it does require that you have the LCD installed (see photo 2). All interaction is via the on-board "Identify" and

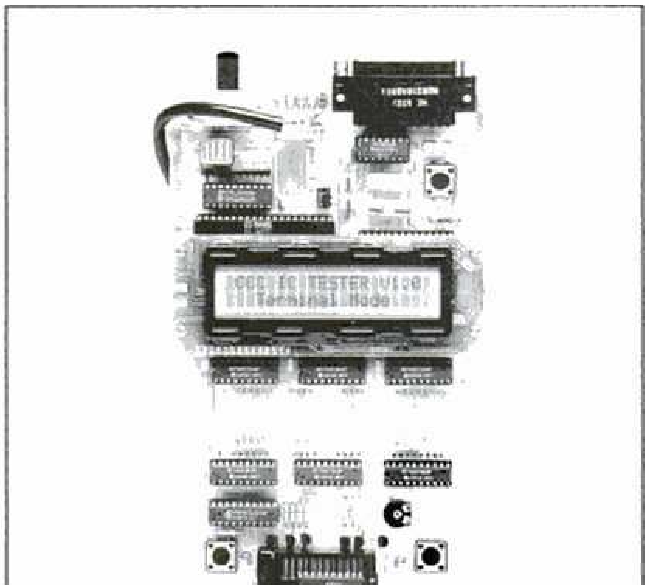


Photo 2: *You can use the IC tester in stand-alone mode, provided you have attached the tester's optional LCD. The push-button switches in the upper right, lower left, and lower right control the tester's operation.*

"Retest" push-button switches and the LCD. A DIP switch enables or disables "74Cx" mode.

Pressing the "Identify" push button causes the tester to attempt to identify the device in the ZIF socket. If the identification is successful, the device name is displayed on the LCD; otherwise, an identification failure message is displayed.

Once a device has been identified, you can test other devices of the same type using the "Retest" push button. The test vectors for the identified device are then applied to the DUT, and detected pin failures, if any, are displayed on the LCD.

Flexibility

While the Circuit Cellar IC tester represents hundreds of hours of hardware and software development, the end result is something that was designed to be simple to operate. It clearly offers a great deal of flexibility for testing common devices, but it is also useful for developing tests for custom or proprietary devices like programmable array logic.

In order to test a PAL, you must develop a series of test vectors that apply bit patterns to the device inputs and watch for expected output values just like those from any standard 74xx logic device. The PAL test vectors are based on the logic-transfer functions (the logic equations) of the device.

You compile and name the test vectors and then add them to the device library. To test PALs, you run the IC tester in the normal way: Just insert the PAL to be tested in the ZIF socket (bottom-justified) and specify either the `Identify Device` option (the easier choice) or the `Test Specified Device` option, giving the device's name, "PAL1," for example.

In Conclusion

The powerful, yet easy-to-use, Circuit Cellar IC tester can provide testing and identification for innumerable standard and custom IC devices, in packages ranging from 14 to 24 pins. It's a tool that can save you time and money by catching potential problems during production, helping debug problem boards, and by identifying and/or verifying unknown devices or devices with uncertain operation. The flexibility and capability offered by this tester were previously available only to those willing to spend thousands of dollars.

In all honesty, I have to admit that the hardware for this project was trivial compared to the enormous software task involved in creating the operating system and device library. The initial Revision 1.0 ROM-resident library contains more than 200 generic entries. Considering that a generic entry of "7400" can cover 10 clone entries, the library physically covers about 800 chips. I owe a special debt of gratitude to those who helped put this project together and saved me from having to deal with all this software.

Experimenters

While you can order printed circuit boards and kits for the Circuit Cellar IC tester, I encourage you to build your own. If you don't mind doing a little work, I will again support your efforts. A hexadecimal file of the executable code for the IC tester's 8031 EPROM (a 27256) is available free for downloading from my bulletin board at (203) 871-1988. It contains the complete Revision 1.0 ROM-resident device library and software for complete stand-alone and terminal mode operation.

Alternatively, you can send me a preformatted IBM PC 5¼-inch disk (2.0 or higher) with return postage, and I'll put the file on it for you. Please add $5 for a printed copy of the user's manual. Of course, as always, this free software is limited to non-commercial personal use.

Copyrighted material

I would like to personally thank Roger Alford and Bill Potter for their collaborative efforts on this project. Bill Potter's tireless dedication creating the test-vector library and Roger Alford's clever programming expertise served to make the Circuit Cellar IC tester a true performer.

Editor's Note: Steve often refers to previous Circuit Cellar articles. Most of these past articles are available in book form from BYTE Books, McGraw-Hill Publishing Company, P.O. Box 400, Hightstown, NJ 08250, (1-800-2-MCGRAW).

Ciarcia's Circuit Cellar, Volume I covers articles in BYTE from September 1977 through November 1978. *Volume II* covers December 1978 through through June 1980. *Volume III* covers July 1980 through December 1981. *Volume IV* covers January 1982 through June 1983. *Volume V* covers July 1983 through December 1894.

It's virtually impossible to provide all the pertinent details of a project or cover all the designs I'd like to in the pages of BYTE. For that reason, I have started a 24-page bimonthly supplemental publication (with no advertising) called Circuit Cellar Ink, which presents additional information on projects published in BYTE, new projects, and supplemental applications-oriented materials. For a one-year subscription, send $14.95 to Circuit Cellar Ink, or call(203) 875-2199.

The following items are available from

Circuit Cellar, Inc.
4 Park St., Suite 12
Vernon, CT 06066
(203) 875-2751

1. Circuit Cellar IC tester experimenter's kit for stand-alone or terminal operation. Contains IC tester printed circuit board, 11.0592-megahertz crystal, programmed 27256 EPROM with Revision 1.0 device library, MAX232 level shifter, Aries 24-pin narrow-format ZIF socket, and manual with complete parts list. ICT01-EXP..$99
2. Circuit Cellar IC tester full printed circuit board kit for stand-alone, terminal, or PC-host operation. Contains IC tester printed circuit board, 8031 processor and crystal, programmed 27256 EPROM with revision 1.0 device library, Aries 24-pin narrow-format ZIF socket, IC sockets, all board-mounted components and IC's, PC-host software on PC format disk, power supply, and manual. ICT01-FULL......$179
3. Complete Circuit Cellar IC tester kit with stylish enclosure. Full printed circuit board kit with all components, right-angle-mounted evclosure adapter board with ZIF socket and LCD, software on PC format disk, power supply, and manual. ICT02..............................$349
4. Two-line by 20-character LCD and 14-pin Berg connector for either item 1 or 2. 2x20 LCD..$32

All payments should be made in U.S. dollars by check, money order, MasterCard, Visa, or American Express. Surface delivery (U.S. and Canada only): add $5 for U.S., $8 for Canada. For delivery to Europe via U.S. airmail, add $14. Three-day air freight delivery: add $10 for U.S. (UPS Blue), $25 for Canada (Purolator overnight), $45 for Europe (Federal Express), or $60 for Asia and elsewhere in the world (Federal Express). Shipping costs are the same for one or two units.

There is an on-line Circuit Cellar bulletin board system that supports past and present projects. You are invited to call and exchange ideas and comments with other Circuit Cellar supporters. The 300/1200/2400-bps BBS is on-line 24 hours a day at (203) 871-1988.

Copyrighted material