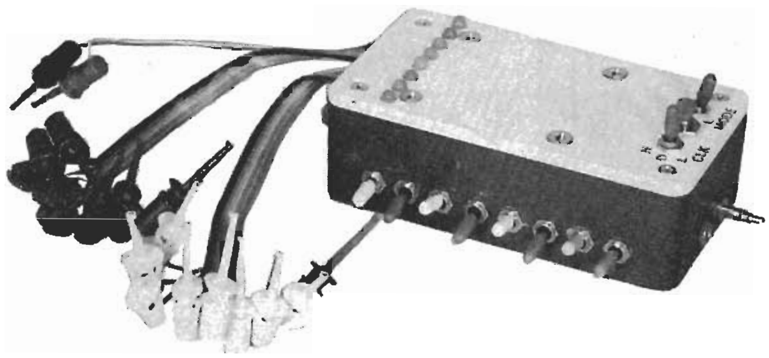
Build the Digilyzer: It performs some digital analyzer functions, but it costs a lot less—and it fits in the palm of your hand.



BUILD THE DIGILYZER

JOHN YACONO AND MARC SPIWAK

A DIGITAL ANALYZER IS AN EXtremely useful instrument for troubleshooting digital circuitry. Unfortunately, digital analyzers are usually priced beyond most hobbyists' budgets. However, the Digilizer, the subject of this article, can perform some of the functions of a digital analyzer. And the best thing about the Digilyzer is that you can build it for less than $50—a lot less than the purchase price of a factory-made analyzer.

The Digilyzer monitors the logic levels at eight of its inputs (called the *test inputs*), and when they match a user-set bit pattern, it latches the binary data present at its other eight inputs (called the *data inputs*). Once data has been latched, it is displayed on the unit by eight tri-colored LEDs. Digilyzers are end-stackable so that multiple units can monitor 16, 24, or 32-bit wide buses—this makes them quite versatile.

One of the most popular applications for the Digilyzer is monitoring the input and output from an integrated circuit to verify its operation. However, the Digilyzer performs many other complex tasks such as monitoring any memory location on a bus.

To use the Digilyzer you start by connecting the test inputs to the address bus, connect the data inputs to the data bus, and set the bit pattern for the address you want to observe. When the Digilyzer next encounters that address, it latches onto the data.

The Digilyzer also offers other options that make it especially suitable for troubleshooting. For example, you don't have to define all the bits that make up the bit pattern. Some can be left in a "don't care" state. This is useful for checking computers with faulty buses that have an intermittent line. It also permits you to observe what is occurring over a wide range of addresses.

The Digilyzer has two modes of operation: *latch* and *free run*. In the latch mode, the unit latches onto data when the test inputs match the user-set bit pattern, and it ignores subsequent matches. In the free-run mode, the latched data is updated each time there is a match.

The Digilizer is fitted with a BNC output that can trigger an oscilloscope when it detects a match. That feature allows the oscilloscope to display the serial data produced by a device such as an RS-232C port when its control and data lines are at user-specified values. This feature is particularly useful for testing parallel-to-serial converters, checking the protocol of serial ports (you'll be able to "see" the stop, parity, and data bits), and determining the handshaking lines being used by a device.

The match detector

The schematic shown in Fig. 1 shows three main sections: a match detector, a match-signal processor, and a data buffer that performs double duty as a display driver. An 8-bit identity comparator, IC1, accepts two 8-bit words (denoted A and B) and compares them.

If each bit of the two words match, the output pin 19 goes low; if any corresponding bits in the words don't match, the output remains high. Moreover, when a match is found, the output will go low only if the enable input (EN pin 1) is low. When the enable input is high, the output remains high.

The bits that form the A word input to IC1 are supplied by TEST inputs TP1 to TP8. Each of the eight bits for the B word can be user set by switches S1 to S8. Each of the SPDT center-off switches can be set in one of three positions:

- the *low* position that ties a B input to ground
- the *high* position that allows a B input to float high through a pull-up resistor
- the "don't care" position that ties a B input to its corresponding A input, ensuring a match regardless of the A bit's value.

The ENABLE input functions as part of an optional *clock* input (TP9). If used, it gives IC1 the ability to sense the clock of the device-under-test (DUT). That feature can prevent false

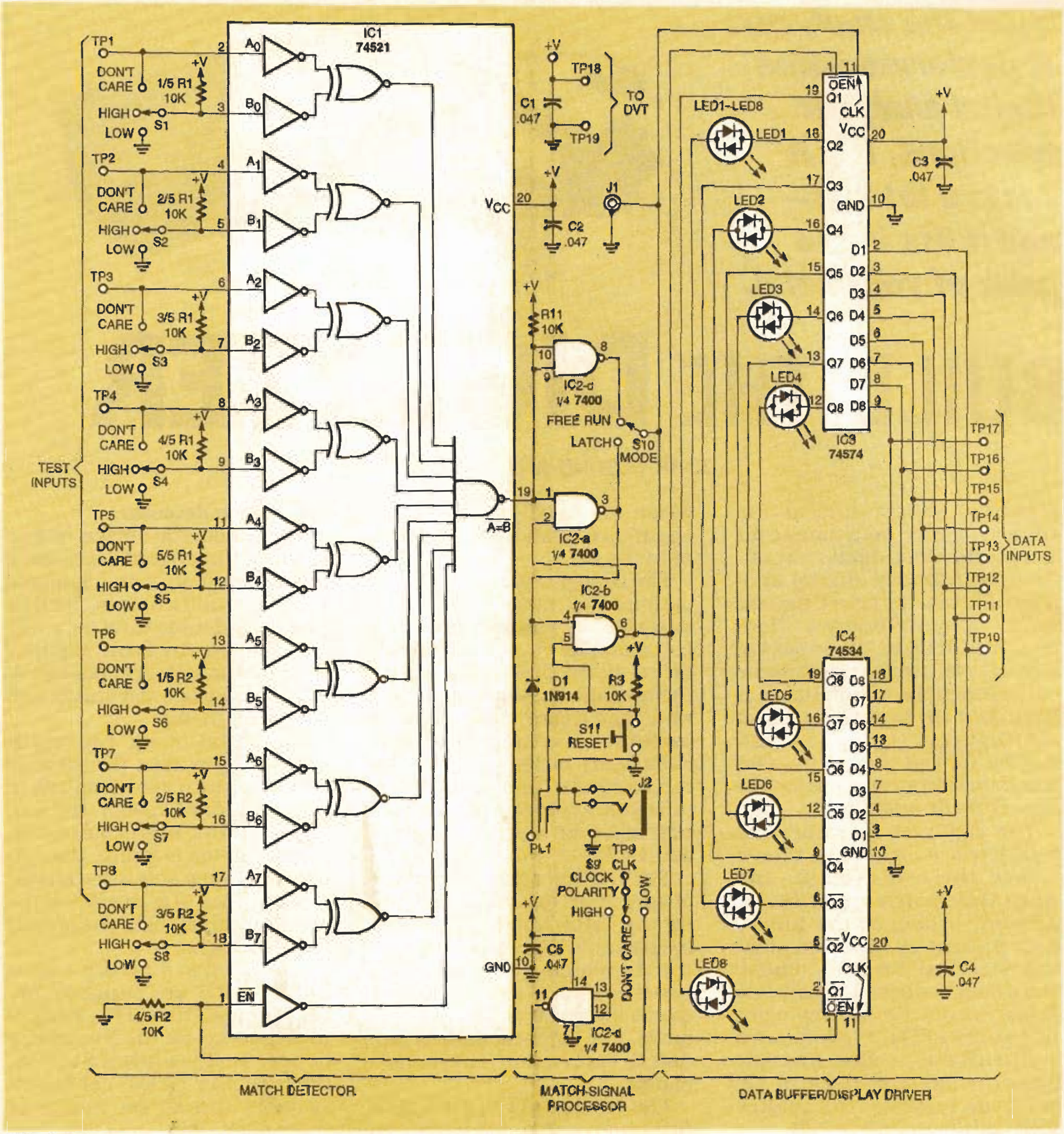


FIG. 1—THE DIGILYZER HAS THREE FUNCTIONAL SECTIONS: a match detector, a match-signal processor, and a data buffer that performs double-duty as a display driver.

triggering of the match detector because of the presence of *unsettled* test inputs, such as might occur when a parallel-printer port or a multiple-bit bus is being tested.

The clock pulses from TP9 enter SPDT center-off switch S9, which allows you to determine how the clock pulses will be treated. If a valid clock pulse is low-going, it can be passed directly to the enable pin by setting S9 to the low position.

If you want to ignore the clock input, placing S9 in the "don't care" position allows R9 to pull the enable pin low, causing IC1 to test data continuously. Putting S9 in the high position allows IC2-d to invert the incoming clock pulses so that a high clock signal will enable IC1.

The match-signal output from IC1 is passed to the match-signal processor. It maintains an internal clock line (CLK) and an output-enable line ($\overline{\text{OEN}}$) used by the data buffer. (The clock line generated by the match-signal processor should not to be confused with the external clock pulses from the device under test or DUT.)

An explanation of the data buffer/display driver will be helpful in explaining the overall function of the clock and output-enable lines before the discussion of the match-signal processor is completed.

The data buffer

The data buffer/display driver is composed of two complementary octal D-type flip-flops, IC3 and IC4. Test points TP10 to TP17, collectively called the DATA inputs, are connected to the inputs of both flip-flop ICs. As a result, for each flip-flop in IC3, a complementary (inverting) flip-flop receives the same data in IC4. Input data from the DATA inputs is clocked into both flip-flop ICs when there is a positive transition of the match-signal processor's CLK line.

Data contained in the flip-flops (whether inverted or not) is presented to the outputs of the two ICs only when the output-enable ($\overline{\text{OEN}}$) lines are low. If these lines are high, the flip-flop outputs go into high-impedance mode, neither sinking or sourcing current. When enabled, both ICs are capable of sourcing and sinking enough current to drive the LEDs.

Each complementary pair of flip-flop outputs is connected across a tri-color LED, which contains a red-emitting die and a green-emitting die. The dice in each LED are connected anode-to-cathode so that when the LED is biased in one direction, it emits red, and when biased in the opposite direction it emits green. However if it is powered by alternating current, yellow light is emitted.

It will be helpful if you understand the operation of one pair of complementary flip-flops and their associated LED. For example, if TP17 is low, that low is presented to D8 of IC3 and to D8 of IC4. When the flip-flops are clocked and the $\overline{\text{OEN}}$ lines are low, Q8 of IC3 presents a low to LED8, while Q8 of IC4 inverts the data and presents a high to LED8. The LED is oriented so that its green element is forward-biased (emits) under those conditions.

Had TP17 been high when the flip-flops were clocked, the LED would have been biased in the opposite direction, thus turning it red. So the color of an LED will indicate the logic level presented to its corresponding data input: red for high, green for low. The LEDs can also emit yellow, but only under conditions that won't be apparent to you until more of the circuit's operation has been explained.

The entire display is disabled when the flip-flop's OEN lines are held high. Moreover, the match-signal processor holds that line high until it receives a low match signal from the match detector. That keeps the display inactive until relevant data has been latched by the flip-flops.

Match-signal processor

The match-signal processor is responsible for clocking the flip-flops and enabling the display on receipt of a low from the match detector. Furthermore, it allows multiple Digilyzers to work in unison for 16-, 24-, and 32-bit wide data analysis. It also sets the unit for either a free-running mode or latched mode, which will be described later. The clock signal it generates is available as an input to the oscilloscope through J1, which will also be explained later.

Despite its many functions, the match-signal is composed of only three three NAND gates and a few support components. Two of those gates (IC2-a and IC2-b) form an R-S latch. One input of the latch receives the output of the match detector, and the other latch input is held high through R10. To simplify this, consider the latch input connected to the match detector, the S input, and the other latch input, the R input. That makes the output of IC2-a the Q latch output and the output of IC2-b the $\overline{\text{Q}}$ latch output.

Consider that the mode switch S10 is in the latched position. That puts the output of IC2-a in control of the clock line. The output of IC2-b is always in control of the output-enable line, regardless of the mode that is selected.

Follow the operation of the latch with Table 1 as a guide. To begin, assume that the match detector is high, indicating that there is no match between the switch settings and the incoming test data. Now press reset button S11, which forces the latch into the reset state: Q is low, $\overline{\text{Q}}$ is high. Now the clock line is low (ready to make a positive transition), and the output-enable line is high (turning off the display). When S11 is released, the R input goes high, but the CLK and $\overline{\text{OEN}}$ lines remain the same.

When IC1 detects a match between the switch settings and the test-data bits, and the external-clock input enable is in the right state, the S input to the latch goes low, and the latch sets; clock line Q goes high, and the output-enable line ($\overline{\text{Q}}$) goes low. That causes the positive-edge triggered flip-flops to take the data bits at the data inputs and display them. Because the latch is set, any further transitions of the match detector are ignored. Pressing S11 will reset the latch, again turning off the display and allowing the process to repeat.

If S10 is in the free-run position and the unit is reset, the display is initially off because the latch still controls the $\overline{\text{OEN}}$ line. However, a NAND-implemented inverter controls the clock line. Because the inverter is not part of the latch, it is free to make a positive transition (a flip-flop clock pulse) upon receipt of each match signal. Thus in the free-run mode, the display is initially off; it turns on with the first match, and re-

TABLE 1—LATCH OPERATION

Input			Output		
Activity	Match Detector (S)	Logic At Pin (R)	CLK-Line Logic Level (Q)	OEN-Line Logic Level ($\overline{\text{Q}}$)	Response
Depressing SII	High	Low	Low	High	Reset
Releasing SII	High	High	Low	High	No change
Data Match	Low	High	High	Low	Latch
Data Mismatch	High	High	High	Low	Still Latched

mains on to be updated by each match that follows.

If matches occur frequently while the Digilyzer is in the free-run mode, one or more of the LEDs might emit yellow. That means its corresponding data input(s) is changing rapidly from high to low and back again. This can't be achieved in the latched mode because the flip-flops are latched and cannot change state.

Whether or not the unit is in free-run or latch mode, the CLK line is available for external applications through BNC connector J1. The connector permits the Digilyzer to be connected to an oscilloscope's trigger input so that the oscilloscope can display serial data when user-set conditions have occurred.

Ganged operation

As explained earlier, the latch also allows two or more Digilyzers to be connected together and operated in unison. This feature is desirable when you want to monitor 16-, 24-, and 32- bit data/address lines. The units must be interconnected to prevent them from latching until all of them have found the right data at their respective test inputs. The Digilyzers are connected by means of plug PL1 and jack J2.

When connecting only two Digilyzers together, plug PL1 of one unit into J2 of the second, and vice versa. When connecting three units, mate the plug of the first unit with the jack of the second, and mate the plug on the second unit with the jack of the third, and connect the plug of the third unit with a short cable length back to the jack on the first Digilyzer.

Notice that the units are connected together in what amounts to a ring formation. You can insert a fourth unit into the ring 32-bit analysis.

Figure 2 shows the electrical connections made between the latches of two interconnected units. There can be more units in the series, but describing the operation of two should be sufficient to give you an understanding of what occurs when there

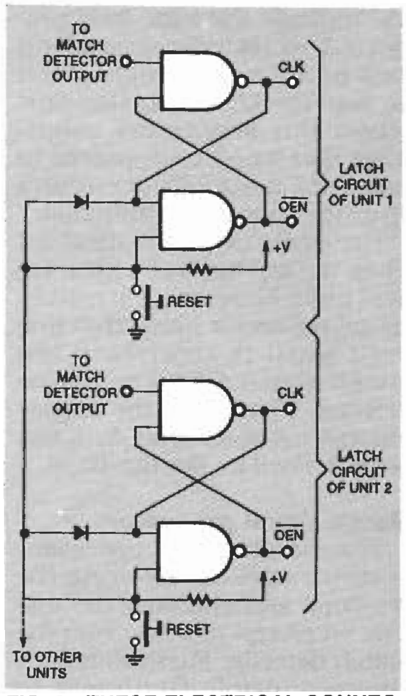


FIG. 2—THESE ELECTRICAL CONNECTIONS are made between the latches of two interconnected Digilyzers.

are three or four. The plugs and jacks were deliberately omitted from the drawing for clarity. Notice that the simplified wiring looks like a bus connecting one unit to another, but is actually not "ring-like" at all. That effect is achieved with clever wiring in the jacks.

Assume that one unit detects a match, but the other unit does not. It would not be desirable for either unit to latch because only one unit has detected a match; the latches must be inhibited in some way. For example, if unit 1 doesn't detect a match, the match-detector signal in that unit is high. The resulting CLK signal must be low, regardless of the reset input value (examine Table 1 to verify this).

That action pulls the reset inputs of both latches low through the diode in unit 1, which puts both latches in their *metastable* state, effectively inhibiting them. As soon as both (all) units detect a match, the diodes will be reverse-biased, and all the latch-reset inputs will be pulled high via their 10K resistors. All the units will latch data, and their displays will be activated.

Some might find that technique objectionable because conventional wisdom suggests that applying two lows to a NAND-implemented R-S latch is forbidden or disallowed. In rigorous mathematical terms, those adjectives are correct because the state cannot be defined with the rigid rules of logic. In short, if you try to determine the output of a metastable latch, you will be unable to arrive at a definitive answer.

However, latches are not mathematical constructs; they are practical components. Therefore, they must produce an output, and they do. Moreover, that output is definitive and consistent. In this situation, both outputs are high. The high on the clock line latches new data into the flip-flops, but the high on the $\overline{\text{OEN}}$ line prevents the display of the irrelevant data.

However, the output changes as soon as the latch is removed from the metastable state. This instability of the state justifies the term *metastable*. Thus the

LISTING 1

```
10 CLS
20 PRINT "If you want this procedure to auto-repeat press Y:"
30 AUT$=INPUT$(1)
40 INPUT "How many test values are there";N
50 DIM TEST(N-1)
60 FOR I= 0 TO N-1
70 PRINT "What's the #";I+1;"value";
80 INPUT TEST(I)
90 NEXT I
100 PRINT "Downloading dummy value. Press the reset button on the
Digilyzer"
110 PRINT "to begin test proceedure."
120 LPRINT CHR$(0)
130 FOR I= 0 TO N-1
140 LPRINT CHR$(TEST(I))
150 NEXT I
160 IF AUT$="Y" OR AUT$="y" GOTO 100
170 PRINT "The proceedure has ended. To repeat the procedure press Y:"
180 ANS$=INPUT$(1)
190 IF ANS$="Y" OR ANS$="y" GOTO 100
200 END
```

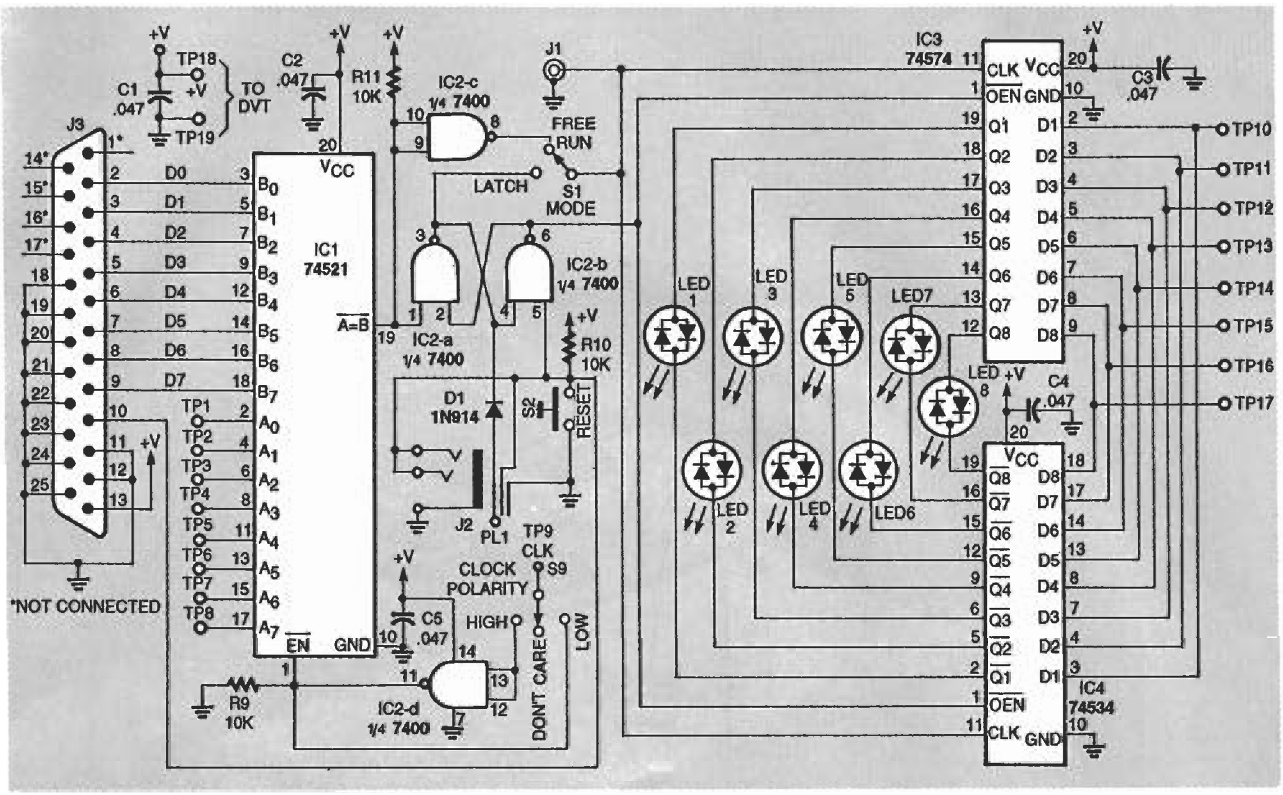


FIG. 3—AUTOMATED TESTING CAN BE PERFORMED: The Digilyzer can be controlled by a computer's parallel printer port. Here the computer downloads successive values of user-set test data.

instability allows the device to avoid premature latching so it can latch valid data.

Computer interface

If you plan to do automated testing, it will be easy to control the Digilyzer from a personal computer's parallel printer port. Figure 3 is a schematic diagram for computerized operation of the Digilyzer. In this application, the computer performs the often laborious task of downloading successive values of the user-set test data to the data-match detector. That permits you to avoid the need for setting the switches to one value after another.

The user-set (in this situation user-programmed) bits are sent to IC1 through pins 2 to 9 on the DB-25 connector shown. The 8-bit word contained on those lines provides the Digilyzer with the information that would have been provided by switches S1 to S8 in the manually-operated unit shown back in Fig. 1. Of course, none of those bits can be set to a "don't care state," but that restriction can be overcome by judicious use of the program, as will be described.

Notice that the clock-polarity setting is still switch- operated. There is no reason for automating that feature. You will probably never want to use more than one clock-polarity setting on a given Digilyzer. It will usually be a "set and forget" switch.

Each time the computer provides the unit with the eight user-programmed switch values, it waits to see an acknowledge signal (a low-going pulse) on pin 10. Of course, the analyzer will only respond to it if you key the reset button.

That allows you to read the LED display and reset the unit before allowing the computer to download the next value. This, and some special features of the program, allow you to forego the computer keyboard after all the test values have been entered, and control the pace of the test procedure from the Digilyzer's console.

The program (shown as Listing 1) is a specialized data-entry procedure. While the program is running, it will ask you to enter the decimal equivalent of the binary number that will be sent to the parallel port. Of course, each bit of that binary number will replace a switch, with D0 (the least-significant digit) as S1, and D7 (the most-significant digit) as S8. This data is called the *test data*.

Initially, the program asks if you want the list of test values you'll enter to be run automatically and repeatedly. That is a useful option for testing many identical Digilyzers because it frees you from having to return to the computer after you test each one.

Next, you will be asked how many test values you will be entering. That allows the program to allocate enough memory for all the values and set the size of two for/next loops: one for input and one for output. During the input loop, you will be asked to provide each of the test values in decimal form. That means that you will enter the decimal equivalent of the binary number formed by the eight user-set switches described earlier. If you are an adventurous programmer, you might want to create a subroutine that accepts binary values.

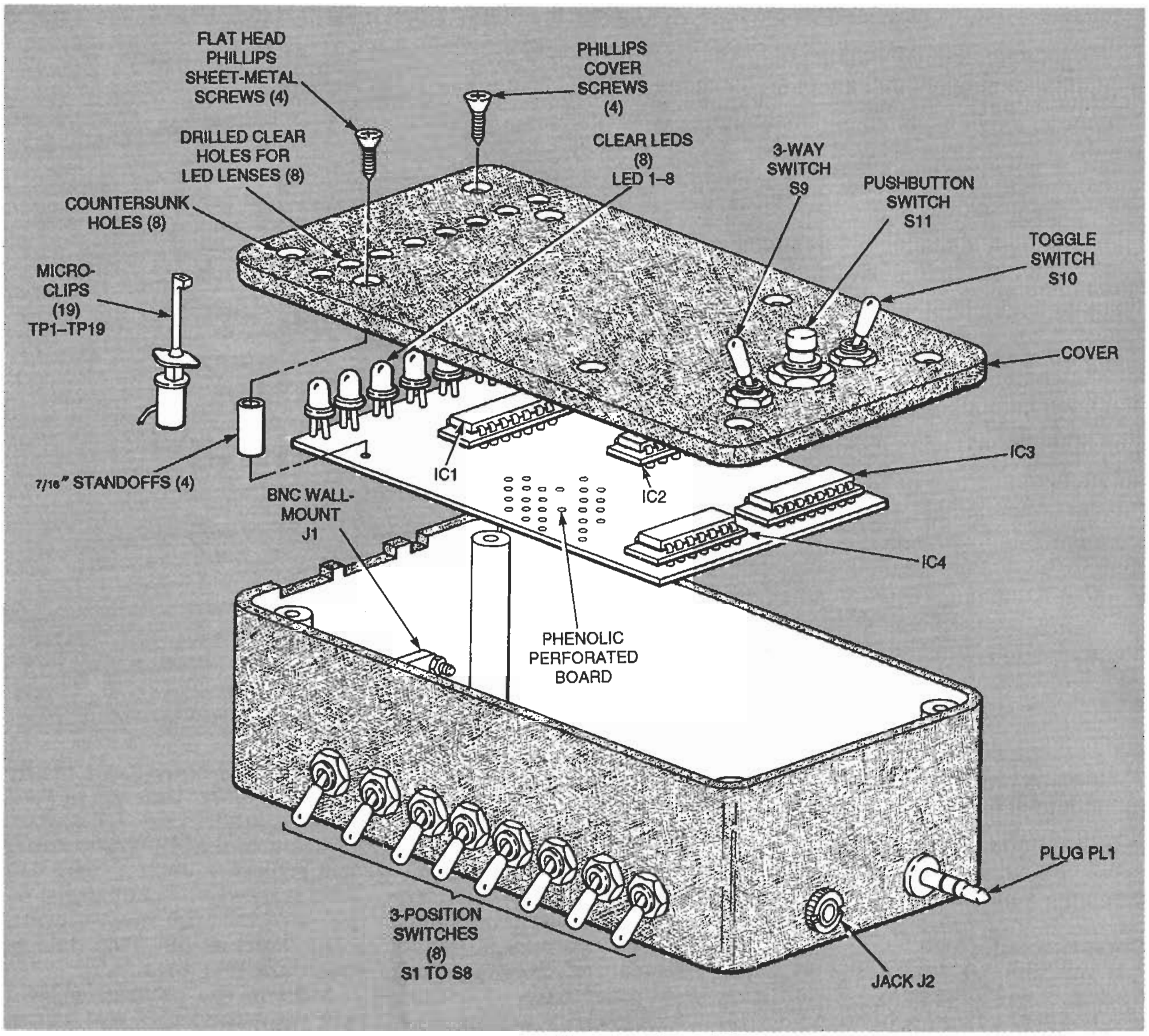

FLAT HEAD PHILLIPS SHEET-METAL SCREWS (4)
PHILLIPS COVER SCREWS (4)
DRILLED CLEAR HOLES FOR LED LENSES (8)
CLEAR LEDS (8) LED 1–8
3-WAY SWITCH S9
PUSHBUTTON SWITCH S11
COUNTERSUNK HOLES (8)
TOGGLE SWITCH S10
MICRO-CLIPS (19) TP1–TP19
COVER
IC3
IC1
IC2
7/16" STANDOFFS (4)
BNC WALL-MOUNT J1
IC4
PHENOLIC PERFORATED BOARD
PLUG PL1
3-POSITION SWITCHES (8) S1 TO S8
JACK J2

FIG. 4—EXPLODED VIEW OF DIGILYZER: The four ICs are positioned as shown. The circuit board is fastened under the cover with four sheet-metal screws and spacers. The LED lenses project through matching holes in the cover.

After all the test values have been entered, the program sends the NUL ASCII character to the Digilyzer. That action locks up the computer until you indicate that you want it to proceed by pressing the reset button. Once the button is pressed, the first test value is downloaded. When the LED display lights up, you can examine the latched data and get the next test value by depressing the reset switch.

If, for some reason, the DUT fails to generate a match to the programmed user setting in a reasonable length of time, the test value can be skipped by depressing the reset button. Thus, if the DUT fails, you can still continue your diagnosis without returning to the computer.

When all the test values have been run, the program checks to see if you chose the automatic mode of operation. If you did, the program produces the NUL character again. That locks up the computer and gives you a chance to connect another device that you want to test. Once you depress the reset button, it proceeds to run through the test values again, as before, and you don't have to return to the computer.

If you did not choose automatic mode, you are asked if you'd like to run through the list of test values again. It is a useful feature if you believe the results of the first test were unclear. If you don't want to rerun the procedure, the program will terminate. Although the computer might be locked, the program can be terminated at any time by pressing CTRL-BREAK on the PC keyboard.

As was previously explained, you can test devices as if you programmed in a "don't-care state." Enter two test values for each bit in the "don't-care state." One test value should have the ambivalent bit low and the other should have it high.

While multiple don't-care bits can make data entry a chore because you must consider all the combinations, the program can be modified to handle don't-care states and arrive at suitable test values on its own.

PARTS LIST

All resistors are 1/4-watt, 5%, unless otherwise specified.
R1, R2—10,000-ohm five-resistor, single-in-line network (see text)
R3—10,000 ohms
Capacitors
C1–C5—0.047 µF, polyester
Semiconductors
D1—1N914 small-signal silicon diode, 75 PIV
IC1—74521 8-bit identity comparator circuit
IC2—7400 quad NAND gate
IC3—74574 flip-flop, D-type, octal, positive-edge triggered, three state
IC4—74534 flip-flop, D-type, octal, positive-edge triggered, inverting, three-state
LED1–LED8—tri-color light-emitting diode, T-¼
Other components
S1–S9—SPDT toggle switch w/ center-off position, miniature panel mount, Radio Shack No. 275-375 or equivalent
S10—SPDT toggle switch, miniature, panel mount, Radio Shack No. 275-635 or equivalent
S11—momentary pushbutton switch, normally open, miniature panel mount, Radio Shack No. 275-1547 or equivalent
J1—male BNC connector, chassis mount, type 1094, Radio Shack No. 278-105 or equivalent
J2—stereo jack, miniature
PL1—stereo plug, miniature
TP1–TP19—test clips, miniature (see text)
Miscellaneous: Project case (see text), perforated phenolic circuit board, 0.1×0.1 grid (see text), three 20-pin, one 14-pin wire-wrap IC sockets, ribbon cable (see text), wire-wrap wire, insulated wire 28AWG, four Phillips-flat-head sheet-metal screws, solder

Some other useful additions to the program that you might want to add include subroutines to write the test data to a file or the printer. Similarly, some means of test-data retrieval and editing might also be valuable.

Construction

Building the Digilyzer is relatively simple because, aside from the four ICs specified, the only other circuit components are two single-in-line (SIP) resistor networks, one discrete resistor, and one diode. The SIP networks simplify wiring. With the exception of a single 10K resistor that has one grounded lead, all other 10K resistors function as pull-up resistors, making it a straightforward network application.

The prototype circuit was built with point-to-point wiring on a perforated circuit board measuring 2½×3½-inches with 0.42-inch holes in a 0.1×0.1-inch grid. The dimensions of the circuit board were determined by the inside dimensions of the construction case: 4¾×2½×1½-inches. The case is large enough to contain the circuitry and internal wiring without crowding, yet the package is small and convenient to handle.

Refer to schematic Fig. 1 for wiring and exploded view Fig. 4 for a general layout of the integrated circuits IC1 to IC4. Start by wiring the circuit on the perforated board. Sockets are recommended for all four ICs. Leave the insertion and soldering of the eight LEDs as the last step.

When the circuit-board assembly is complete except for the LEDs, select four spacers to separate of the top surface of the perforated board from the underside of the case cover, as shown in Fig. 4. (The spacers in the prototype are ⅜-inch high, slightly higher than the upper surfaces of the ICs mounted in sockets.)

Mount and solder the eight LEDs at one end of the board at a height that will allow their lenses to project through holes drilled in the cover of the case with the spacers in place, as shown in Fig. 4.

After the circuit is complete, add labeled lengths of insulated wire to all points necessary for connecting the switches, jacks, and test leads. After all wiring is in place, solder the other ends to the correct terminals on the switches and jacks as shown in Fig. 1.

Next, solder approximately 7-inch lengths of ribbon cable to all test points. The prototype was wired with a 9-conductor multicolor ribbon cable for TP1 to TP9, and an 8-conductor multicolor ribbon cable for TP10 to TP17. Because the ribbon cable had 10 conductors, the remaining two-conductors removed were used for the V_{CC} and ground leads.

Because of the correspondence between the standard resistor color code and the colors of the wires bonded to the flat cable, the black wire in the nine-conductor cable, was assigned to test clip 1 (TP1) and the black wire in the eight-conductor cable was assigned to TP10.

Test the circuit at this stage in its construction before you mount any of the switches and jacks in the case. When you are satisfied that the circuit operates as described, complete the necessary hole drilling in the side walls of the case and its cover for mounting the switches and jacks.

Start first by marking the centers of the eight holes in a row in the side wall of the case

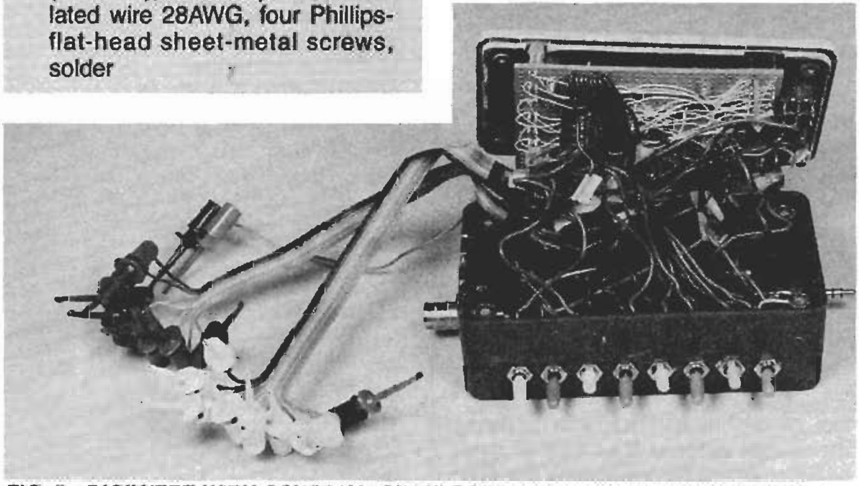


FIG. 5—DIGILYZER WITH COVER/CIRCUIT-BOARD ASSEMBLY REMOVED. The internal wiring is arranged so that the cover can be closed without interference.

row in the side wall of the case for switches S1 to S8, as shown in Fig. 4. You can simplify the task of drilling an even row of holes in the case for the switches by applying a strip of drafting tape to the case and marking the locations of the hole centers. Note: The ganged bodies of the eight switches selected occupied the space between the cover mounting posts inside the case.

Drill the eight holes for the switches as well as the holes for plug PL1 and jacks J1 and J2. Mount all the switches and the plug and jacks with the ring nuts provided or nut and bolt sets, as required.

Mark the locations of the eight holes in a row across the cover to admit the lenses of the LEDs. (In the prototype they were sized for the diameter of T1¼ LED lenses.) Tape a section of perforated board on the top surface of the cover and use the 0.1-inch matrix as a guide for locating the centers of the holes to be drilled. The spacing should correspond to the spacing of the LEDs on the circuit board.

Drill the eight holes for the LEDs, drill the four countersunk holes in the case cover for mounting the circuit to the underside of the cover, and drill the three holes in the cover for switches S9, S10, and S11, as shown in Fig. 4. You might want to apply decals to the cover to identify the switch functions.

Fasten the circuit board to the cover with the four spacers and suitable self-tapping screws. It will not be necessary to drill additional holes in the circuit board because the screws will pick up on matching holes in the board.

After fastening the circuit board to the case, attach the miniature test clips to the ends of the ribbon cables and separate twin lead. The test clips were color coded in the prototype: eight green clips on the eight-wire ribbon cable, eight white clips and one black and white clip on the nine-wire cable, and black and red clips to terminate the twin wires. The Digilyzer is now complete. Ω