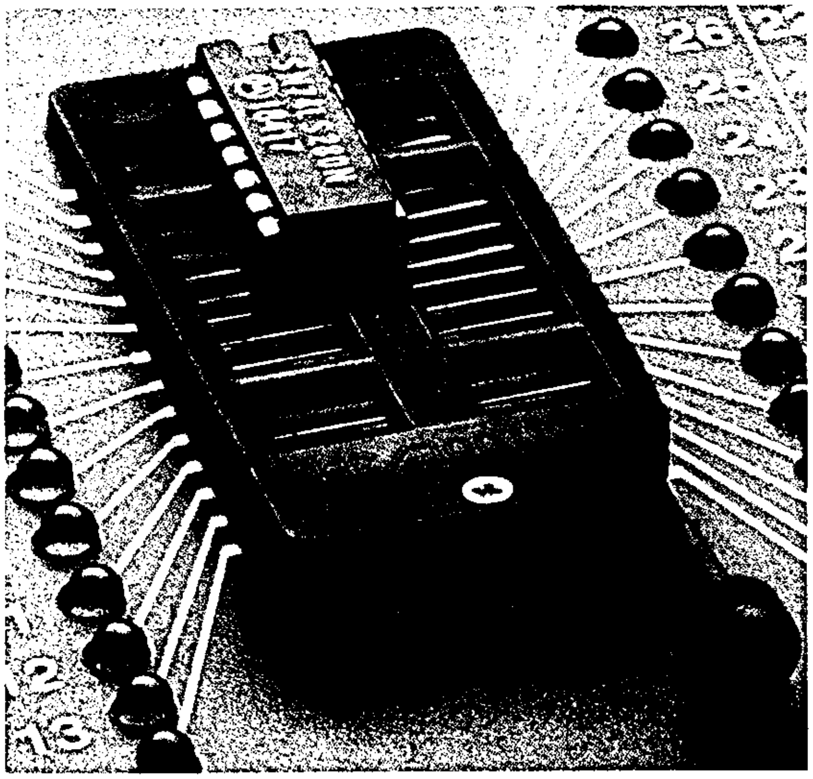
# 900 SERIES • SOLUTIONS

DYNAMIC TROUBLESHOOTING



FLUKE

# Manual Exercises

1. Test the following chips on the UUT in manual by making local parameter changes until they all pass consistently. Once you have a passing test, try to "tighten up" the test by reducing the number (or amount) of changes to the minimum that will still allow a consistently passing test.

U36 - (7404) combinatorial
U37 - (74244) combinatorial
U57 - (41256) programmable
DEMO3 - (PAL22V10) synchronous

2. ***Look through the rest of this worksheet before proceeding to familiarize yourself with what is available.*** Then, as necessary, refer back to the explanations and diagrams for help in selecting parameters for change, and understanding their purpose.

3. Below is listed the proper order for conducting this exercise.

STEP 1 - Enter Develop mode
STEP 2 - Load the on-board Library data for the DUT
STEP 3 - Change test parameters until the optimum test is determined
STEP 4 - Note changes made and Store the parameters for that board location
*(Repeat steps 2 - 4 for each of the four test chips)*
STEP 5 - Store the whole test Sequence developed so far
STEP 6 - Test the validity of the new Sequence by trying it on the demo UUT
STEP 7 - View the file containing the parameter changes

4. To assist you in selecting the appropriate parameters to adjust, consider that all devices fall into three general catagories, as described below.

**Combinatorial**

Those devices whose output pins reflect immediately the conditions on the input pins. They have no internal memory, but may have tristate output pins (e.g. NAND gate, Bus Driver).

Common parameter adjustments include: FMASK, Threshold, Ignore

**Synchronous**

Devices with internal memory whose state is brought to an output pin or can be definitely inferred. They always have a clock input line (e.g. up/down counter, J-K Flip Flop).

Common parameter adjustments include: FMASK, Sync Time, Reset

**Programmable**

Devices which require data to be written into them before outputs are valid (e.g. UART, RAM).

Common parameter adjustments include: Sync Time, Trigger, Offset

# Exercise Steps

**Step 1** Do your testing in the DEVELOP MODE by entering the keystrokes below. You will find that this mode, once past the first few screens, looks and acts like manual mode. Refer to the prompter sheet and follow the flow chart for developing a new sequence.

| **KEY** | **ACTION** |
|---|---|
| <F3> | selects Develop |
| <F2> | selects New Sequence |

(Type in a file name for this new sequence, such as TEST)

| | |
|---|---|
| <F5> | selects system memory for storage - instead of the cartridge |
| <ENTER> | enters the selected name and storage location |

**Step 2** Load the on-board library data for the first chip.

| **KEY** | **ACTION** |
|---|---|
| <F1> | selects Manual |
| 7404 <ENTER> | calls the library data for the first chip, 7404 (or U36) |

**Step 3** Change test parameters until the optimum test is determined.

| **KEY** | **ACTION** |
|---|---|
| <F1> | selects Local |

The screen should now look familiar. Adjust the parameters for the first chip until you are satisfied with them.

Note the changes to the default parameters that you found necessary for each chip below.

a. 7404 (U36) ____________________

____________________

b. 74244 (U37) ____________________

____________________

c. 41256 (U57) ____________________

____________________

d. PAL22V10 (DEMO3) ____________________

(check sum = ____________________)

**Step 4** Enter the following keystrokes to store the data as the first test in a new sequence.

| KEY | ACTION |
|---|---|
| <ESC><ESC> | returns up two menus |
| <F3> | selects define location, (defn_loc) |
| U36 <ENTER> | names the test you developed for the 7404 as the parameters for testing location U36 |

**Repeat Steps 2 - 4** of the procedure for each of the chips until all four have been tested and stored as "named locations."

**Step 5** When you have stored the last location end the development of this sequence and prepare the files related to it.

| KEY | ACTION |
|---|---|
| <F4> | selects END, which stores the named locations you developed, and compiles the new test sequence |
| <ESC> | exits the Develop mode back to the main menu |

**Step 6** Test your newly developed sequence by running it in sequence mode on the demo UUT.

| KEY | ACTION |
|---|---|
| <F2> | selects Sequence |
| <F1> | selects the option to run a sequence |
| TEST | identifies the sequence to be run, (or type in the name you selected if it was not TEST) |
| <F5> | indicates that it is in system memory, not the cartridge |
| <ENTER> | starts the selected sequence |

(Proceed through your test like normal to ensure that it works. Then continue as follows.)

| | |
|---|---|
| <ESC> | exits the run sequence mode |
| <F5> | acknowledges that you are deliberately quitting the mode |
| <ESC> | exits Sequence back to the main menu |

**Step 7** View the location file (.LOC) that you created.

| KEY | ACTION |
|---|---|
| <F3> | selects Develop mode |
| <F1> | selects Files |
| <F2> | selects Edit |
| TEST | identifies the file to be edited, (or type in the name you selected if it was not TEST) |

**Step 7** (Continued)

| KEY | ACTION |
|---|---|
| <F3> | selects Type of file |
| <F3> | selects location file (.LOC) for editing |
| <F5> | identifies this file as residing in System memory (;SYST) |
| <ENTER> | enters the above information |

Cursor down through the information that was stored as specific parameter changes for the board locations listed. They should match the notes that you made on each in STEP 4. When done, enter the following.

| | |
|---|---|
| <ESC> | quits Edit mode without saving |
| <ESC> | returns to the Develop mode menu |
| <ESC> | returns to the main menu |

* * *

## DEVICE LEVEL

The following is excerpted from the Operator Manual, section 2.7, Testing Applications, and explains the use of each adjustment.

### Loading Adjustments

When a device drives a high fanout or capacitive load, the signal rise time will be extended. Increasing the FMASK setting will compensate for this. Loading is normally within 50nsec of the response time for a device. If a greater FMASK is necessary, the user should investigate other causes.

### Floating Inputs

Inputs floating or connected to a tristate source will cause an apparent fault when the signals are floating at an indeterminate level between 1 and 0. The DUT and Interface Buffer may see logic 1 at different levels. The best solution is to enable an external gate on a control signal that is active when device inputs are valid.

Note: Adjustments to threshold that may appear to solve the problem are probably not universal. Problems reappear on other boards because different devices have different characteristic thresholds.

### Noise

To ignore noise and ringing on the inputs of a device, Threshold may have to be increased. Moving GND lead of the Interface Buffer close to DUT may also assist. Power supply spikes from the AC line input are a particularly difficult form of noise to ignore.

### Race

When the inputs to a sequential device (e.g. clock and data) are within a few nanoseconds, the slight skew of the System may cause the RD to clock in wrong data. If this is the cause of problems, you will generally observe failures on all output pins. Try to move Threshold up and down to separate signal edges.

### Asynchronous Inputs

An external interrupt or any asynchronous signal acquired by latching with a synchronous board clock may exhibit a race problem if the edges are very close. The latching device in this case is not testable with DRC.

### Bus Contention

This occurs when more than one device is enabled on a tristate bus at the same time. Bus drivers such as 74244, 74245 may exhibit this problem. The solution is to gate comparison testing from a signal which indicates valid data. Most microprocessor designs provide this signal on the bus controller chip.

<u>PIN DEF</u> (*H, L Status*)

H and L status checks are performed with 10K pullup/pulldown resistors in the Interface Buffer. Pullups are present by default to bring unused input pins to a definite state. For an L status check, the pullup is present; for an H status check, a pulldown resistor is switched in.

<u>PIN DEF</u> (*F Status*)

Use to monitor ungated periodic clock signals.

A Reset is issued with each pin frequency check.

<u>PIN DEF</u> (*IGN/COMP*)

When a single pin on a device requires a large FMASK, that pin should be ignored while the device is tested a second time with a lower FMASK on the rest of the device.

<u>RESET</u>

Occasionally, a simple Reset fails to duplicate power down reset. Testing can still proceed, except for the few devices with uninitialized registers.

Very rarely, multiple Reset pulses cause such problems as blowing a fuse. The extra pulses used for synchronization may be disabled with the command SYNC_RESET_OFF in the .LOC file of the Sequence. Alternatively, Trigger may be used instead of Reset.

Use of a negative <u>Offset</u> to wait for initialization on a device can be useful in trying to understand the operation of a good board. On a bad board, however, it is susceptible to false failures. Trigger is a more effective initialization parameter in this case.

Use of a positive Offset is recommended for those devices that generate a board's reset so that DRC testing is done during the Reset pulse.

<u>STATE</u>

The state screen shows the pin activity during test and can be a good qualitative indicator of how active a device was.

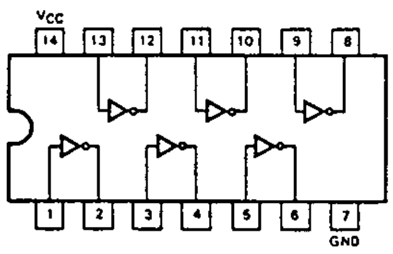
<u>THRESHOLD</u>

It is recommended to keep Threshold constant and vary FMASK and Gate to make devices pass during Sequence Creation.

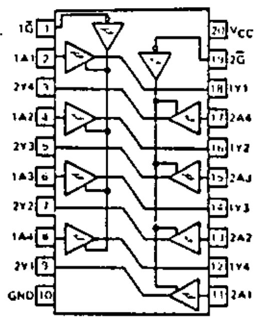
Threshold sensitive circuits are subject to board-to-board variations.

On a signal with a long rise time, lowering Threshold can permit a lower FMASK setting and thus a closer comparison on other pins of a device that should not be compared using a larger value for the FMASK.

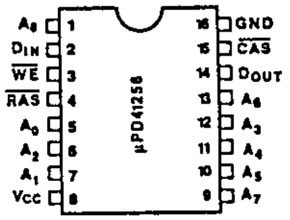
7404



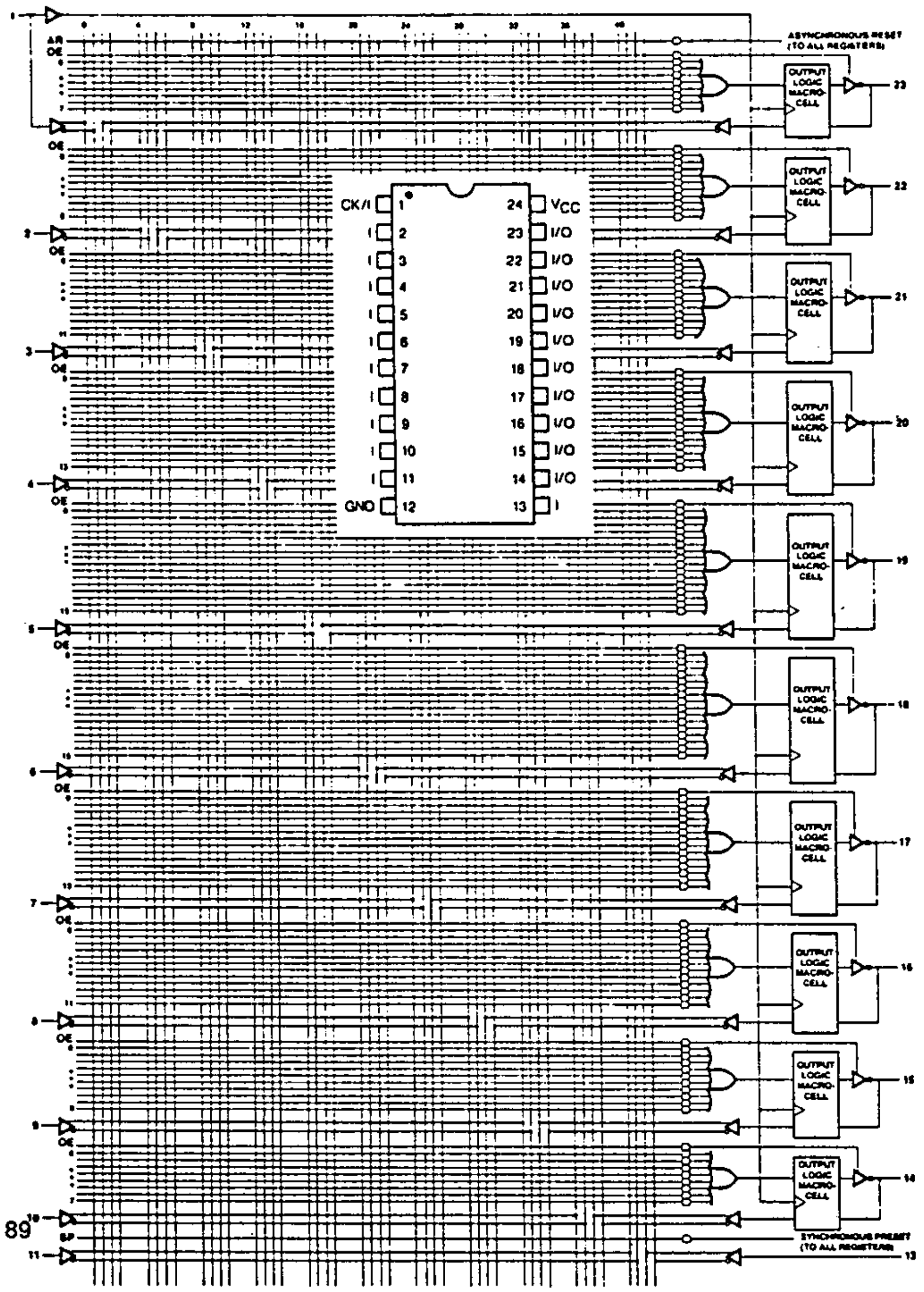
74244



41256



PAL22V10



U36 - (7404) combinatorial - Should pass with a negative Reset Offset of as little as -1, all other settings at default. Alternately, needs increased Fmask and probably decreased Threshold. Encourage the use of two tests here. Test 1 is with the slow pin Ignored, all else at default settings. Test 2 is with the slow pin tested, all settings such that the slow pin will pass on a normal board.

U37 - (74244) combinatorial - Needs Gate signal. The best test is with Gate set on the EXT signal (On the Demo UUT, EXT is a combination of four signals: I/O or Memory, Read or Write - High is active). Also good to add a gate requirement on the chip enable signals, but not needed here.
Beware - because an increased Fmask will hide times that the bus is tri-stated during the power-up self test, increase the TTIME setting to several seconds. A long enough TTIME and no Gate setting will show that there are times when this component is not driving the bus and a compare will fail, but only because of an invalid test.

U57 - (41256) programmable - Must insure that both RD and DUT contain the same data. Best way is to set a Trigger for a completed write to the last location in the RAM. Alternately you can use a negative Reset Offset to allow time for the same to have occured, but if the board is dead or malfunctioning you will not know that the proper write never occured, and invalid test results will be displayed. To enable a proper Trigger : Word one - set all address lines to high (logic 1), and write enable to low (logic 0).
Word two - set write enable to high (logic 1).
Beware - the power-up self test in the ROM reads and writes to RAM as part of it's testing. After the first time the test is run on any one chip, the contents of the RD and DUT will be the same at the beginning of the next test. This will result in your test giving passing results with almost any test parameters at all. Thus, to verify that your test is valid try running your test alternately between U57 and any other chip in the bank. U57 is the Parity chip and should have different contents than the other eight chips, though all nine RAM chips in the row use exactly the same test.

DEMO3 - (PAL22V10) synchronous - Needs a Checksum calculated and entered in the RD-Test menu, then only needs adequate Synchronization time - usually 3000 is enough. Remember, too much sync time is ok. The test will start as soon as sync is achieved.

# Student Information Form

*Please print clearly*

Date: ____________________

Name (as you would like it spelled on your Training Certificate):

____________________
*first* *middle* *last*

Title: ____________________

## Technical Background

Voc. Tech. School: ____________________ *grad. date*

Military Tech. Schools: ____________________ *date(s)*

____________________ *date(s)*

____________________ *date(s)*

College: ____________________ *date(s)*

____________________ *date(s)*

Degrees Earned: ____________________ *date*

____________________ *date*

## Employment

Company Name: ______________________________
Your Mailing Address:

______________________________
*street* *m/s*

______________________________
*city* *state* *zip*

( ____ ) ______________________________
*area-code* *Telephone No.* *ext.*

Name the individual in your organization to whom additional course offerings, data sheets, and schedules can be sent:

Name: ______________________________
Title: ______________________________
Mailing Address:

______________________________
*street* *m/s*

______________________________
*city* *state* *zip*

( ____ ) ______________________________
*area-code* *Telephone No.* *ext.*

If you have had prior experience with the equipment or systems, (or similar equipment and systems) pertaining to this training course, please list and indicate below your level of involvement by circling the appropriate activities:

| | | | |
|---|---|---|---|
| ______________________ | Programmed | Maintained | Operated |
| ______________________ | Programmed | Maintained | Operated |
| ______________________ | Programmed | Maintained | Operated |
| ______________________ | Programmed | Maintained | Operated |

Have you had any prior experience with other FLUKE products? If so, please list the particular products below:

______________________________

______________________________

______________________________

*Thank you.*

# Course Critique

It is requested that you complete this form and return it to the instructor after the completion of the training course.

Course Title: ______________________________

Training Location: ______________________________

______________________________

Date of Completion: ______________________________

Instructor's Name: ______________________________

Please answer the following questions to the best of your ability. The answers you give will be used in improving the course in the future so that FLUKE can continue to provide what you, our valued customer, needs. Please answer each question as completely and as specifically as possible, and provide a rating from 1 (for poor) to 10 (for excellent).

1. How close did the training come to meeting your expectations of the course?

______________________________
______________________________
______________________________
______________________________
______________________________
______________________________ Rating: ____

2. How well did the training you received meet the learning objectives as stated in the training materials?

Rating: ____

3. How helpful were the workbooks and manuals provided to you in supporting the learning objectives?

Rating: ____

4. How appropriate were the training aids (overheads, UUT, equipment, etc.) in support of the learning objectives?

Rating: ____

5. How prepared was the instructor to present this course?

Rating: ____

6. How would you evaluate the instructor's presentation style?

Notes:

Rating: ____

7. How well do you feel this course will help you in performing your job more effectively?

Rating: ____

8. Have you had previous experience with the situations or equipment covered in this course? If so, in what capacity and with what equipment?

9. Do you have any suggestions for the improvement of the training course that you have not previously mentioned in the critique?

Your candid, well-thought-out comments are greatly appreciated.

Thank you.

370[illegible]

[illegible]

[illegible]

# 900 SERIES • SOLUTIONS

DYNAMIC TROUBLESHOOTING

FLUKE

John Fluke Mfg. Co., Inc.

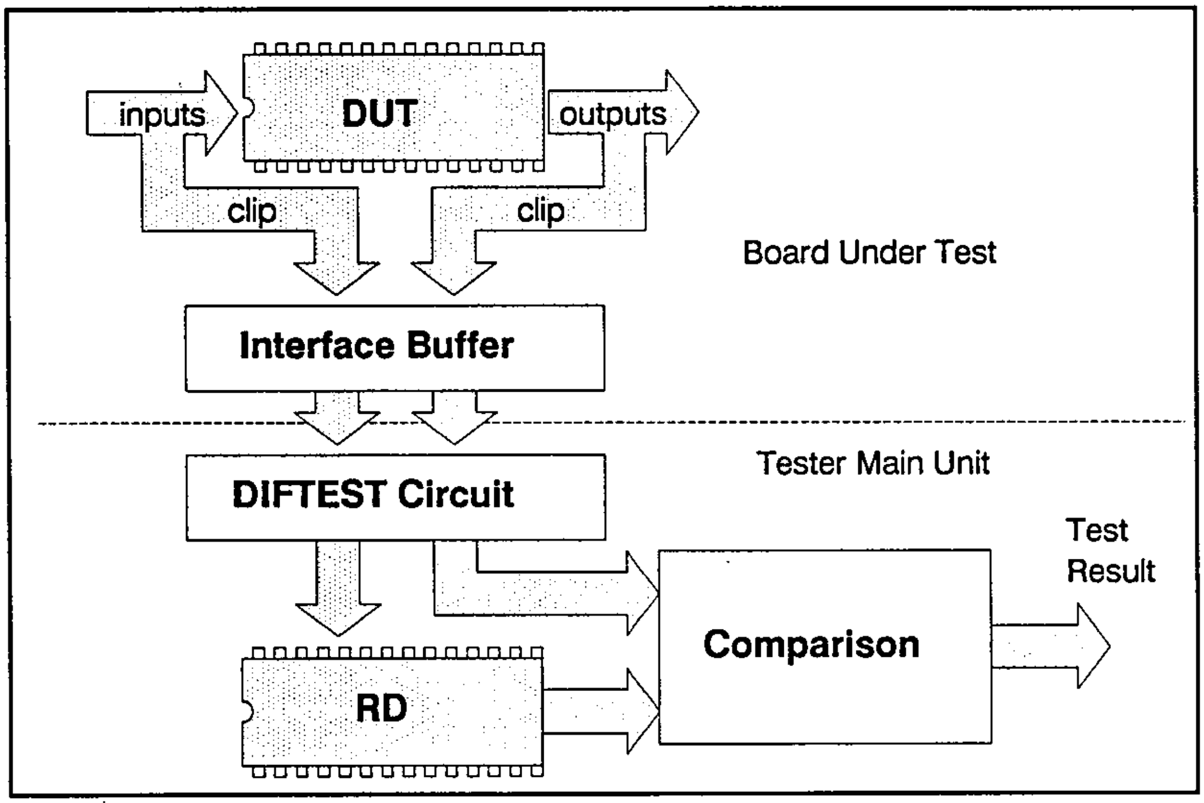
*Customer Support Services*
*P.O. Box C9090 MS 239D*
*Everett, Washington 98206*
*(206) 347 6100 or*
*(800) 44-FLUKE Ext 73*

# Table of Contents

# Section 1
# Introduction and Basic Concepts

*The FLUKE 900 Dynamic Troubleshooter was designed for a repair center environment that has a variety of board types, varying levels of documentation and a range of technical skills. It can be configured relatively quickly to achieve a high level of fault diagnostic accuracy with, or even without, schematics.*

The FLUKE 900 is used where you now use an oscilloscope

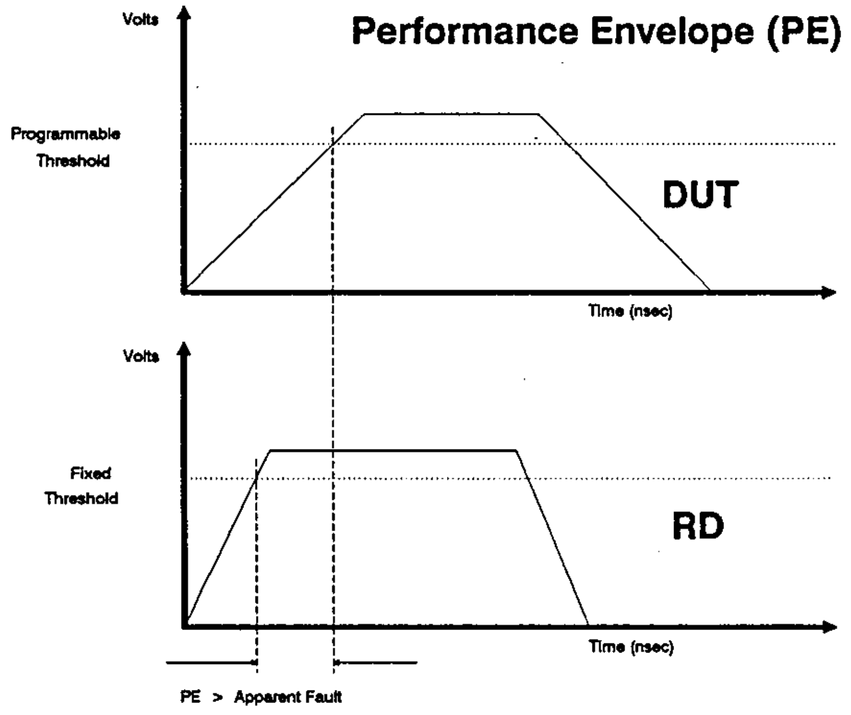


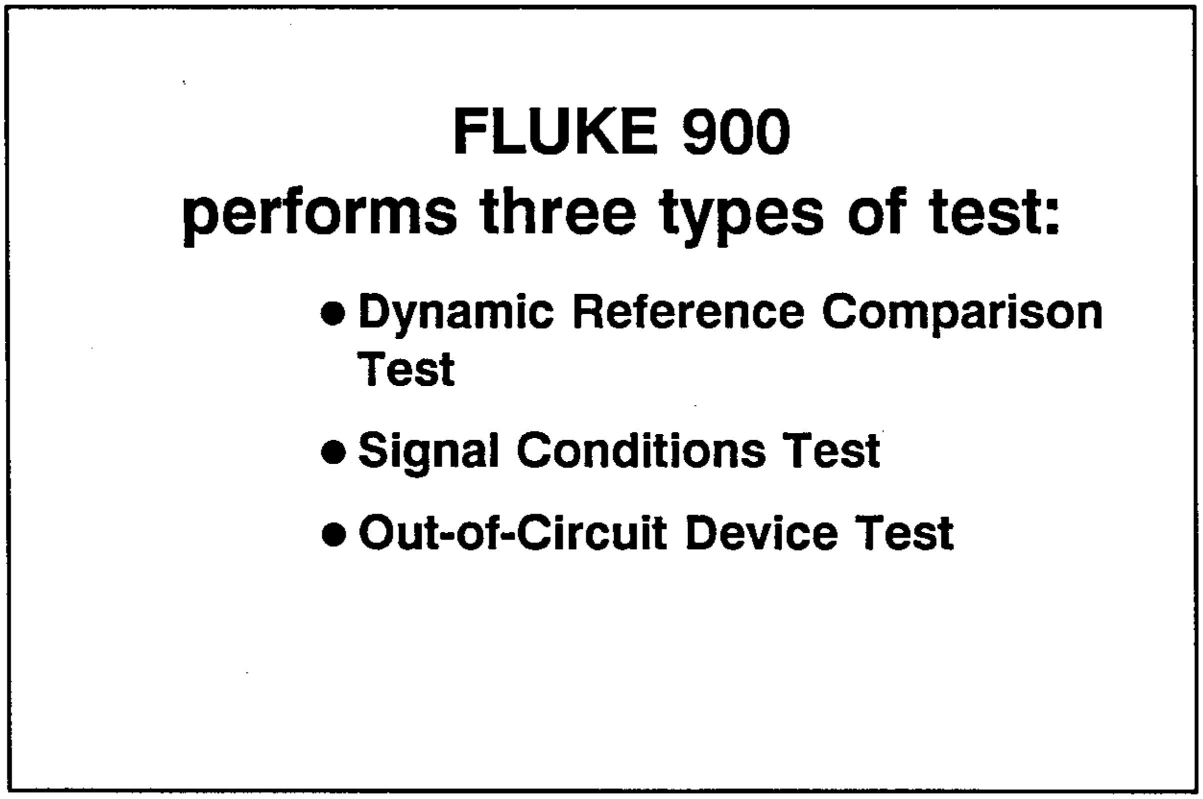
## FLUKE 900 is Used Where You Now Use an Oscilloscope

The basis of the method is to use, as the diagnostic stimulus, the functional go/no-go test that confirms a board is failing. In other words, testing is done in a hot mockup, a system, a standalone micro-based board, or stimulated through a microprocessor emulator pod.

## Dynamic Reference Comparison Principle (DRC)

The FLUKE 900 monitors activity without affecting the board under test. While it can isolate faulty devices in their circuit, it does not artificially backdrive devices like other in-circuit testers. Instead, signals are extracted from a suspect IC device with a clip and applied to the input pins of a known good Reference Device (RD) residing in a ZIF (Zero Insertion Force) socket. The two devices are operated in parallel and the output signals of the Device Under Test (DUT) are compared to the RD.





**FLUKE 900**
**performs three types of test:**

- **Dynamic Reference Comparison Test**
- **Signal Conditions Test**
- **Out-of-Circuit Device Test**

## Performance Envelope (PE)

The PE establishes a time and voltage template for the DUT waveforms. It is programmable within 10 ns time resolution (Fault Mask parameter) and 100 mv voltage resolution (Threshold parameter). This is used to mask out the apparent fault due to normal loading on the DUT. Any pin whose signal exceeds the PE will show a failing test result.

## FLUKE 900 Performs Three Types of Test

1. **Dynamic Reference Comparison Test**
   - Real time functional comparison on digital devices up to 28 pins.
   - All you need is a good Reference Device to be able to test the same IC on a circuit board.
2. **Signal Conditions Test**
   - Measure frequency, logic status, trigger state (like a logic analyzer).
   - Permits you to verify presence of critical signals (i.e. clocks, chip selects, etc.) which can shorten board diagnosis.
3. **Out-of-Circuit Device Test**
   - Verifies the functional performance of Reference Devices from a pattern in the system library.
   - Full dynamic prescreening of the device in the ZIF socket can be done by clipping on a good functioning device on a board.
   - Identify mode matches a generic number to an unknown device.

## Testing is defined by Performance Envelope Parameters

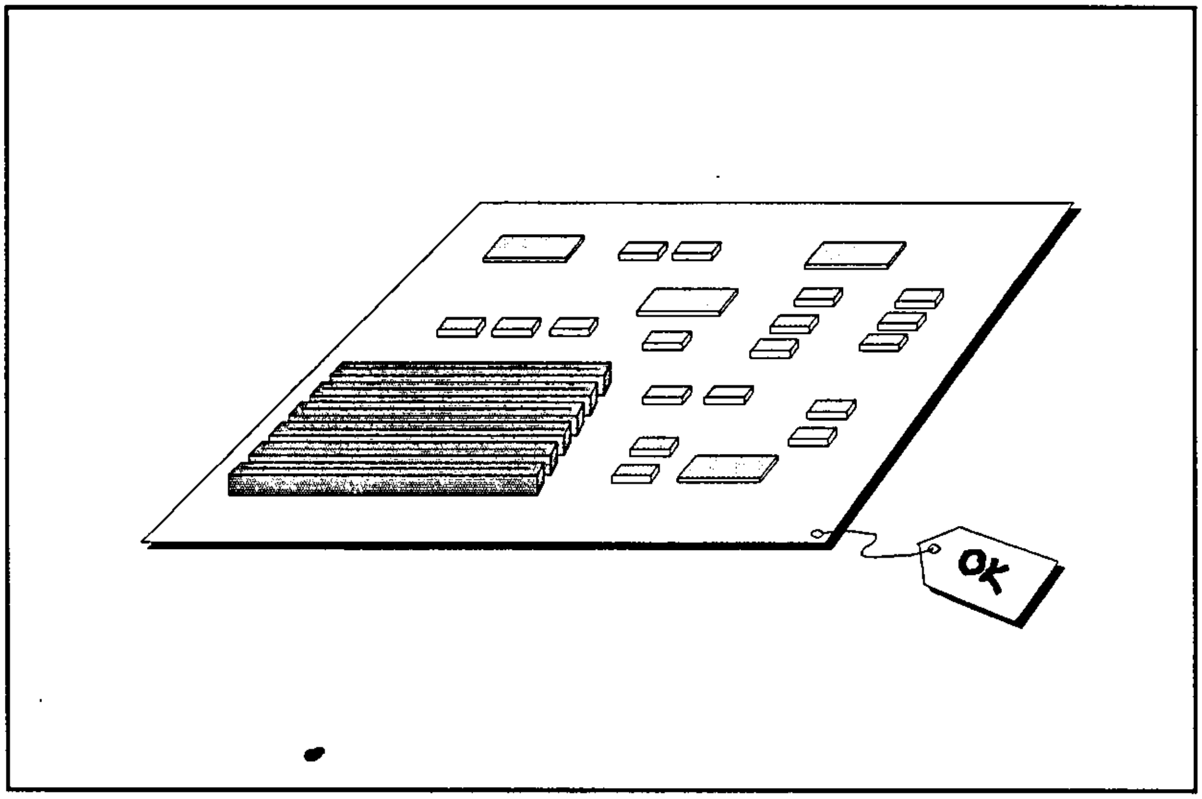
## Hitting the TEST button results in the following actions:

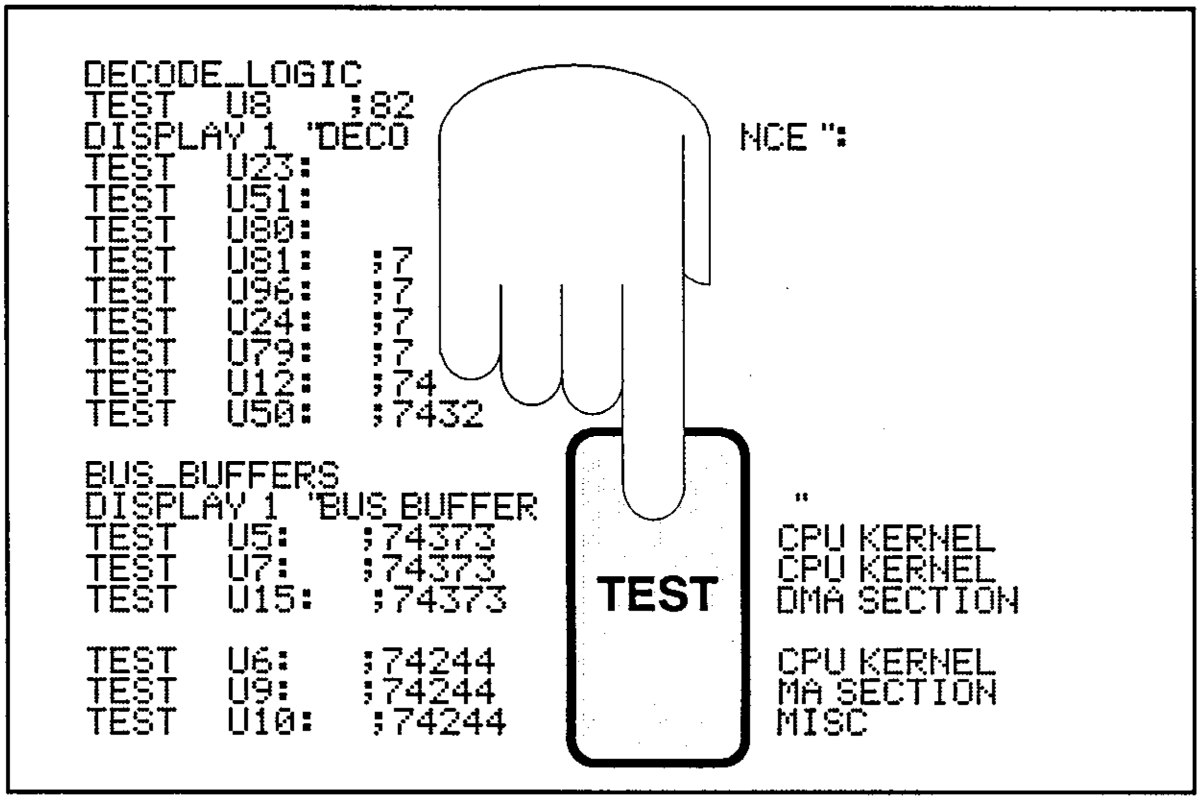
- RD Check
- Clip Check
- Initialization
- Synchronization
- Status Reporting

## Testing is Defined by Parameters

Parameters such as device size, Performance Envelope (Threshold, fault mask), functional type and others, have defined default values that are good for most devices. With the correct parameters set up, you clip onto a suspect device, insert an RD in the ZIF socket, press the test button and read the result. The FLUKE 900 has performed the following steps:

- Verified that the RD is indeed functioning properly
- Verified that the clip is positioned correctly
- Ensured, if necessary, that the proper board activity has initialized the DUT
- Ensured that the RD and DUT are in synchronism before a valid comparison is possible-identified any signal exceeding the expected PE
- Provided status information about DUT pins, including levels,whether signals were active and their frequencies.





```
DECODE_LOGIC
TEST   U8    ;82
DISPLAY 1  'DECO                     NCE ":
TEST   U23:
TEST   U51:
TEST   U80:
TEST   U81:   ;7
TEST   U96:   ;7
TEST   U24:   ;7
TEST   U79:   ;7
TEST   U12:   ;74
TEST   U50:   ;7432

BUS_BUFFERS
DISPLAY 1  'BUS BUFFER                  "
TEST   U5:   ;74373                     CPU KERNEL
TEST   U7:   ;74373                     CPU KERNEL
TEST   U15:   ;74373                    DMA SECTION

TEST   U6:   ;74244                     CPU KERNEL
TEST   U9:   ;74244                     MA SECTION
TEST   U10:   ;74244                    MISC
```

## Programming from a Good Board

As with most types of ATE equipment, programming setup of the FLUKE 900 is done by "learning" a good board. This is a simple matter of testing all devices to determine the few parameters that must be changed from default values to allow all devices to pass. This now becomes a guided clip troubleshooting procedure stored as a Sequence in a removable memory cartridge. A typical board of 100 ICs will take four to six hours to sequence.

## Testing with a Preprogrammed Sequence

A sequence can make an inexperienced test person productive, but it is also intended to complement the skill of any technician. An experienced user's judgement and intuition about a failing board may direct him to a particular area. He can choose to test those few ICs. Either a fault will be located, or we will know where the fault is not, and be directed elsewhere by activity results. (Note: The FLUKE 900 automatically keeps track of devices not yet tested to help narrow down the fault isolation.)

## Paperless Documentation

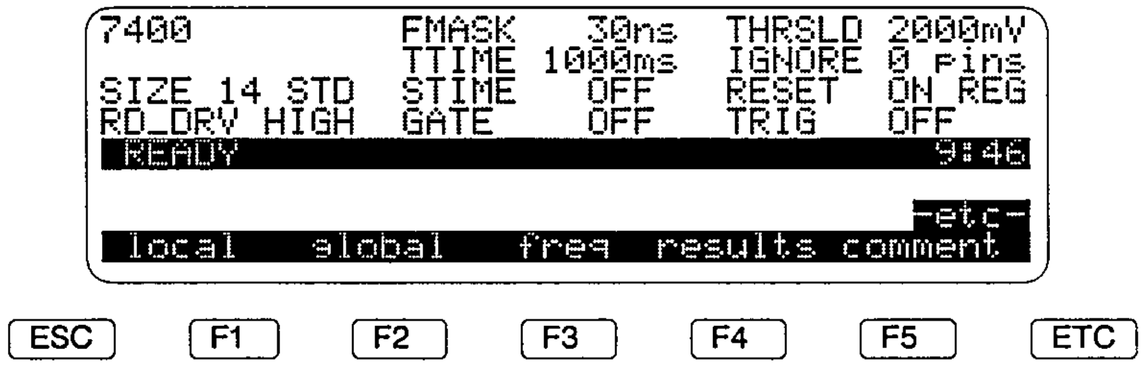
- IC Parameters
- Clipping Order
- Comments
- Results
- Devices not yet tested

## Paperless Documentation

The FLUKE 900 automatically keeps track of things when testing with a sequence such as:

- the parameters for each IC and its in-circuit application
- a guided clipping order for the operator
- troubleshooting comments entered previously by a skilled technician to assist less experienced ones
- test results and operator actions
- devices not yet tested to help narrow down the fault isolation

## Hands-On Exercise



```
7400               FMASK   30ns   THRSLD 2000mV
                   TTIME 1000ms   IGNORE 0 pins
SIZE 14 STD        STIME    OFF   RESET  ON REG
RD_DRV HIGH        GATE     OFF   TRIG   OFF
 READY                                     9:46

                                          -etc-
 local    global    freq   results  comment
```

## Hands-On Exercise Using the Demo UUT

Let us proceed to test a combinatorial device (a 7400 NAND gate) on the Demo UUT.

The Demo UUT and FLUKE 900 should be already prepared to begin this exercise. If they are, proceed to Step 1. If not, set up the 900 and kit as follows:

Connect the two ground (G) patch leads to the GND post on the Demo UUT. Two ground leads connected to different ground points on a test board will improve noise immunity where this is a problem.

Connect the Reset lead from Interface Buffer to Demo UUT post. Vcc is not connected here since we will generate a 5V reset pulse internally in the FLUKE 900. Optionally, we could have used Vcc or another on-board voltage to drive the reset to a level required on the board.

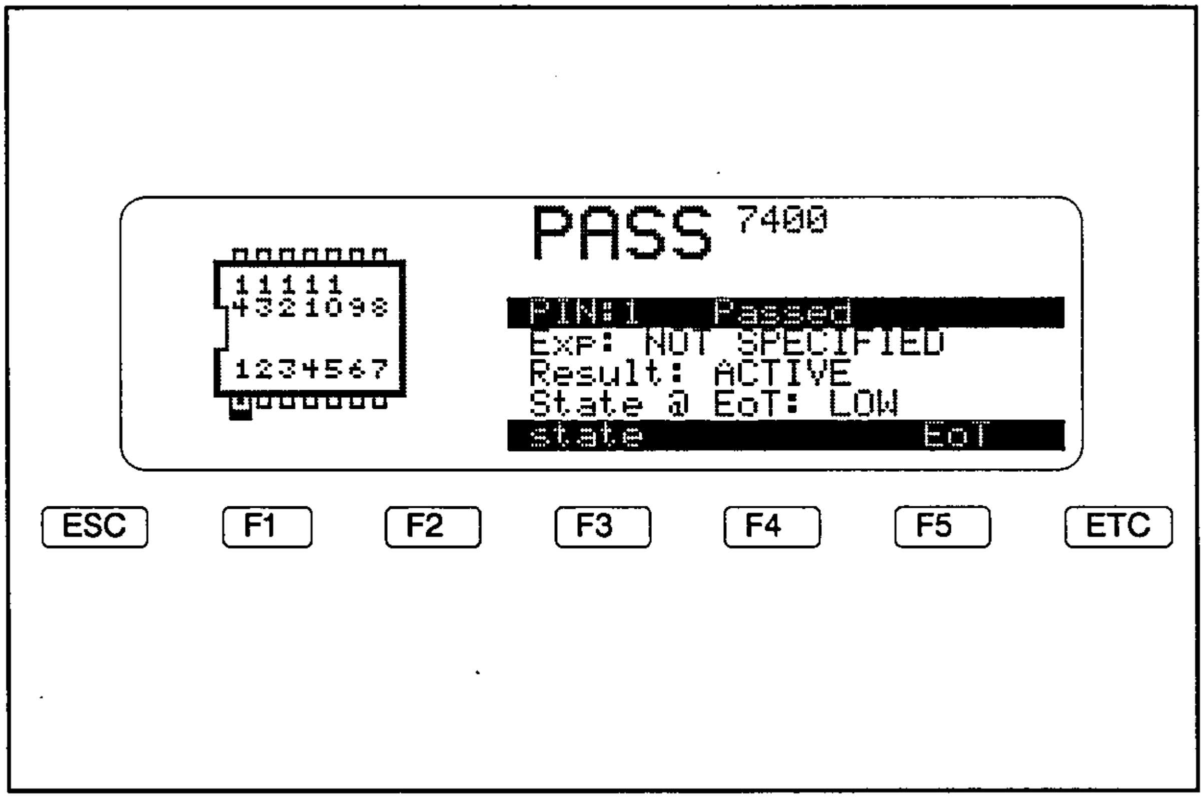
Plug the 16 pin test clip into the Interface Buffer.

### Step 1

Press manual from the main screen level.

Follow the instruction on the command line of the display and enter a device. Try typing 7 4 0 0 ENTER.

The screen should look like the one on the facing page.



PASS 7400

PIN:1 Passed
Exp: NOT SPECIFIED
Result: ACTIVE
State @ EoT: LOW

This combinatorial device is a quadruple 2 input NAND gate. It was recognized by the standard library in the tester and default parameter values displayed. Without clipping on anything, press the [TEST] button.

Notice how the test will not proceed until a functioning 7400 is properly inserted in the ZIF socket. The RD test is a truth table of test vectors for the Reference Device. For ROMs, a checksum is calculated based on a partial reading of the device. For PALs, a checksum is calculated based on a standard stimulus word to the input pins. RD test gives a high assurance that the Reference Device is functioning correctly.

Take a 7400 from the RD tray, put it into the ZIF socket and clamp the lever into place.

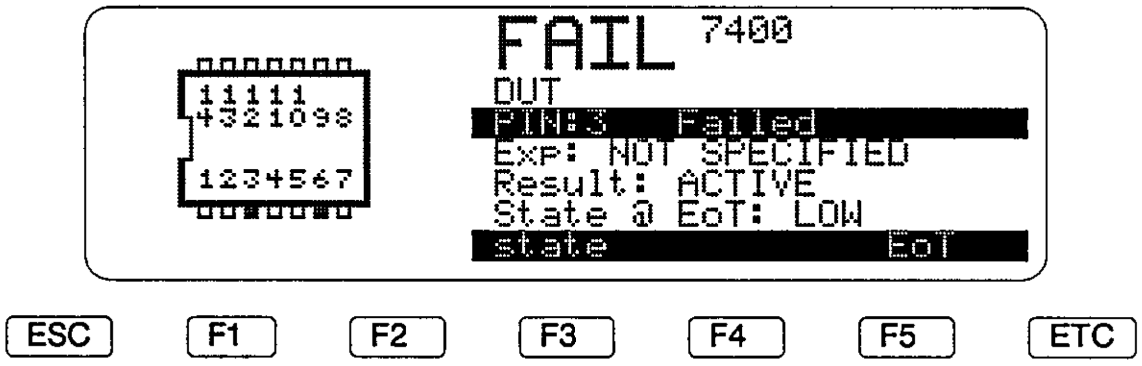
Let's try a test now, without clipping, and observe the clip check pretest results.

Now attach the clip to U14 on the Demo UUT, and after making sure pin 1 on the clip matches pin 1 on the chip, press [TEST]. If this did not PASS, make sure you are firmly clipped on the DUT.

Now, for the purpose of this exercise *reverse* the clip so pin 14 contacts pin 1 on the device. Press [TEST] and observe the result.

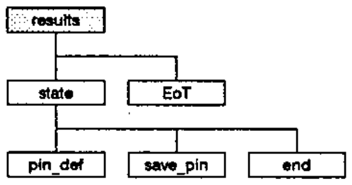
Reorient the clip properly and test again. After your second PASS, examine the result in more detail by pressing results. This graphic depiction appears automatically with a failure; to see this PASS screen the results softkey must be pressed.

The PASS you see should appear similar to the facing frame.



FAIL 7400
DUT
PIN:3 Failed
Exp: NOT SPECIFIED
Result: ACTIVE
State @ EoT: LOW
state EoT

## Results Screen Hierarchy



Cursor control keys move the pointer from pin to pin and the reverse video line displays the comparison result followed by three lines of additional information (if enabled under pin_def).

- an expected activity: high, low, active or specific frequency
- an actual activity: observed levels or value
- level observed precisely as the test ended.

## Step 2

Now press local, then thrsld. We will change the programmable voltage-one threshold of our Device Under Test from the default 2000 mV to 3200 mV. Press [3][2][0][0][ENTER]. Then press [TEST].

After a FAIL result, the facing graphic screen is automatically displayed. (Your result may differ slightly.)

Notice that there are two softkeys with the results screens: state and EoT. These show a picture of the activity seen during the test and state at end of test for all pins. The time-to-fault is also shown on the EoT screen.

The results screen hierarchy is shown on the facing frame.

Before proceeding, answer the following questions regarding the Hands-On exercise just completed:

1. Why did raising the threshold result in a FAIL?
2. Why is the RD Test important?
3. How did the Clip Check determine that you reversed the clip?
4. How does the Results screen help isolate a problem?

## Questions

The following exercises will review and add to the basic concepts of the FLUKE 900 we have just covered in this section.

1. **Recall that the principal of Dynamic Reference Comparison is to place a known good Reference Device (RD) in the same state as a suspect device (DUT) ensuring synchronism and initialization, and then proceeding to compare the output response of both devices to the same stimulus. Keeping in mind that this requires a functioning board that is failing, which of the following test setup(s) are not appropriate for Dynamic Reference Comparison fault isolation?**

   a. An operating multiboard product with extender cards.
   b. A go/no go functional tester driving a board through its card edge.
   c. A microprocessor-based board with an emulator pod testing through the CPU socket.
   d. A PC option card without a CPU and no motherboard to plug into.
   e. A single board system with a ROM-based selftest.

2. **The FLUKE 900 is a tool to troubleshoot digital devices up to 28 pins in size, operating on the board at speeds up to 20 MHz. It captures faults that exceed a defined Performance Envelope by 10 ns and measures frequency on any device pin to 25 MHz. A good indicator of how effective the FLUKE 900 will be on a given circuit board is to see what percentage of the devices fall within the tester's specifications. This is known as coverage. Classify each of the following boards as having high, medium or low coverage:**

   a. Small board with a 120-pin gate array, two 40-pin LSI devices and a small amount of SSI/MSI logic.

b. A board with half analog devices and half digital devices having less than 28 pins running at 10 MHz.
c. A board with a 40 MHz clock signal used on one quarter of the devices, then counted down to 10 MHz for use on the remaining devices. All devices have 28 pins or less.
d. A digital bus-based board with a 20 MHz crystal, three 40 pin LSI's, 20 other devices having 28 pins or less.

3. **Recall that a sequence is created by learning the profile of a good board and storing a troubleshooting procedure in a cartridge for future use. The three major modes of operation are described below. Which modes would require a greater knowledge of how a test board operates?**

    a. Develop mode is used to create a preprogrammed troubleshooting sequence.
    b. Sequence mode is used to test a board with a sequence.
    c. Manual mode is used to test individual devices.

4. **The FLUKE 900 locates static and timing faults to the node on digital ICs under real time in-circuit conditions. It is not a total solution, but complements other equipment and methods. In the list of the following other diagnostic tools, which could be used <u>after</u> the FLUKE 900 for further fault isolation, and which could be used <u>before</u>?**

   a. Emulator, like a FLUKE 9010, to test in functional blocks around a board's CPU.
   b. Shorts finder, such as a high resolution ohmmeter, to locate the device pin or trace point holding down a signal.
   c. Oscilloscope to locate timing or data-dependant faults.
   d. A junction tester, such as Huntron, for testing passives, discretes.

## Answers for the Questions in Section 1

1. d. Is the only setup that cannot be tested with Reference Comparison since the test board cannot be operated.

2. a. Low
   b. Medium
   c. Medium High
   d. High

3. a. Develop mode is best done with knowledge of the board.
   b. Sequence mode can be executed by less experienced operators.
   c. Manual mode requires either knowledge of the board, experience with the tester, or both.

4. a. Before. Emulator identifies a board area, FLUKE 900 isolates to the node.
   b. After. Shorts finder will determine if a driving or receiving device is holding down a stuck node.
   c. Before and after to verify the presence of major signals on the board (i.e. helps in detection of open tracks)
   d. After to isolate source of fault signals to devices.

# Section 2 Operating Features

*The FLUKE 900 is ready to use after connecting its component parts, the power line and executing a built-in Selftest. In this Section you will learn the correct way to do this and become familiar with major physical and electrical specifications.*

## Fluke 900 Terminology

- Main Unit
- Interface Buffer
- Patch Leads
- ZIF Socket
- Signal Monitor
- Cartridge
- RD Tray

## Vocabulary

New terms we will use during the course are:

**Main Unit** The part with the keyboard and display.

**Interface Buffer** The rectangular pod with two ribbon cable connectors, J1 and J2, used to plug IC clips into. It translates voltage levels on a board into FLUKE 900 internal levels.

**Patch Leads** The 5 removable black leads that insert into the Interface Buffer and attach to nodes of a test board. GND, EXT, R, V labels mean Ground, External Input, Reset and $V_{CC}$.

**ZIF Socket** The socket on the Main Unit that accepts a Reference Device (RD).

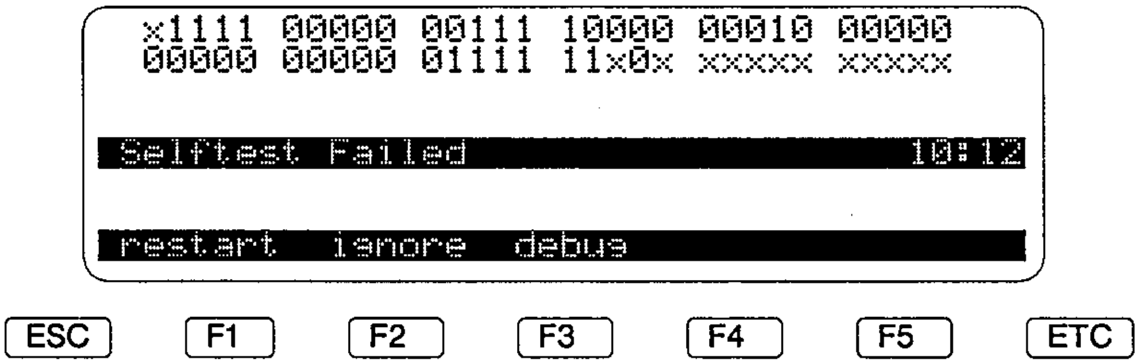
**Signal Monitor** The LEDs around the ZIF socket that display logic status of the RD pins. Also, the offending pin LED remains flashing after a failure is captured. The horizontal row of LEDs identifies the column of numbers that apply to pins of a specific Reference Device.

**Cartridge** Black plastic pack containing 32K of non-volatile memory storage that inserts into the right side of the Main Unit.

**RD Tray** A convenient antistatic foam retainer for Reference Devices and Cartridges. One tray per board type tested is recommended.

## Main Unit Preparation

- Green Power LED
- Memory & Selftest
- Beep



```
x1111 00000 00111 10000 00010 00000
00000 00000 01111 11x0x xxxxx xxxxx

Selftest Failed                         10:12

restart  ignore  debug
```

## Main Unit Preparation

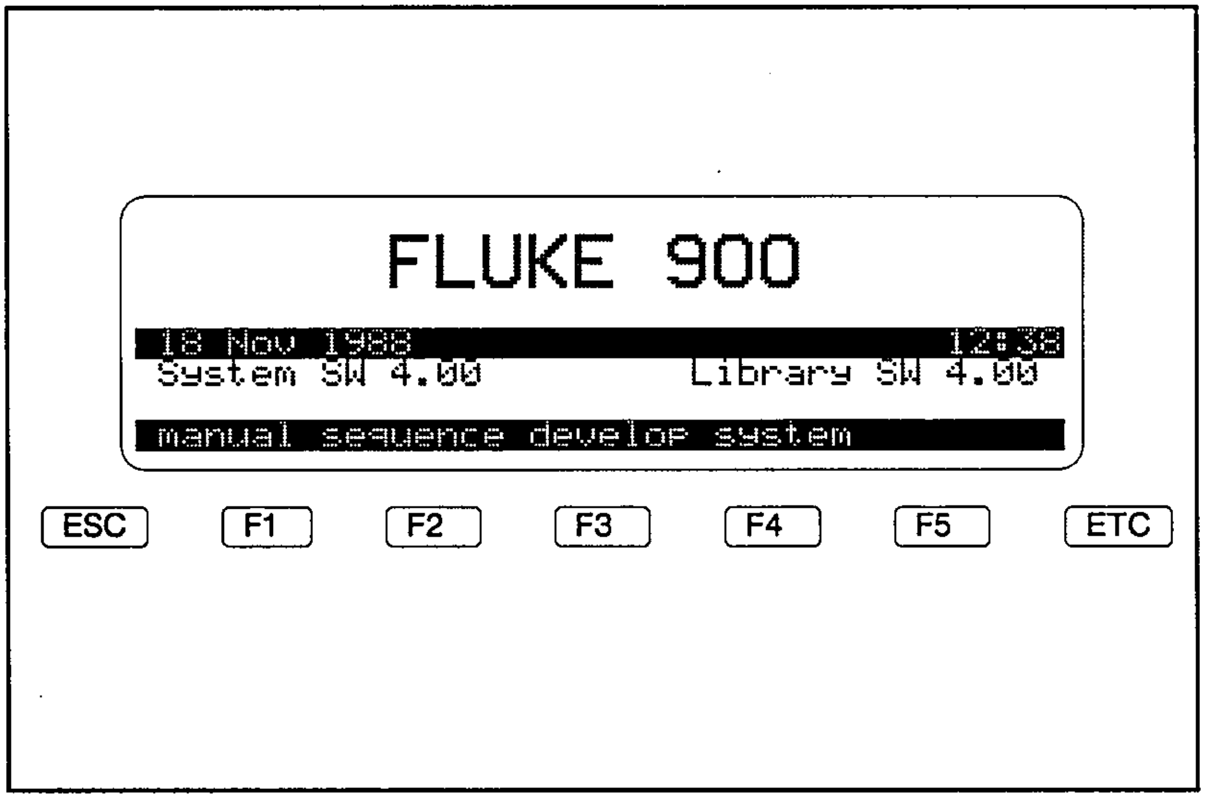
To prepare the FLUKE 900 for use, first make sure that the Interface Buffer is plugged into the Main Unit through connectors J1 and J2. They are labelled and keyed and *not* interchangeable. Connect the power line cord and turn on with the switch on the left side. Observe the following:

- Green power LED illuminated
- Screen reads "Memory Test" followed by a rapid cycle of over 50 hardware tests
- After approximately a minute of Selftest, there is an audible beep and the main menu screen appears.

In normal operation, the most common cause of Selftest failure is powering on with either a Reference Device inserted, a Test Clip on a device or the EXT Patch Cord connected. This causes Selftest to fail, but does no damage to the tester or device.

The function keys of the FLUKE 900 always correspond to a set of key labels printed directly above on the Liquid Crystal Display (LCD). In this instance, the key label ignore corresponds to the function key F2 on the keyboard. Press ignore to ignore the selftest failure or run a Selftest again by simultaneously pressing CNTR and ESC, or by pressing restart. A true Selftest failure is diagnosed from the hardware test failure number and the rows of 1s and 0s displayed on the screen which represent the tester's signal channels. Service personnel from FLUKE use the debug mode to isolate and repair this type of failure.

Try this yourself by taking a device from your RD tray and placing it in the ZIF socket. Pin 1 of the chip is always in the upper left. Start Selftest by rebooting the system by pressing CNTR ESC. For a true hardware problem, the procedure is to press debug then prt_res to print out a detailed diagnostic result on a printer attached to the serial port. Call an authorized service location for further assistance.


FLUKE 900

18 Nov 1988 12:38
System SW 4.00 Library SW 4.00

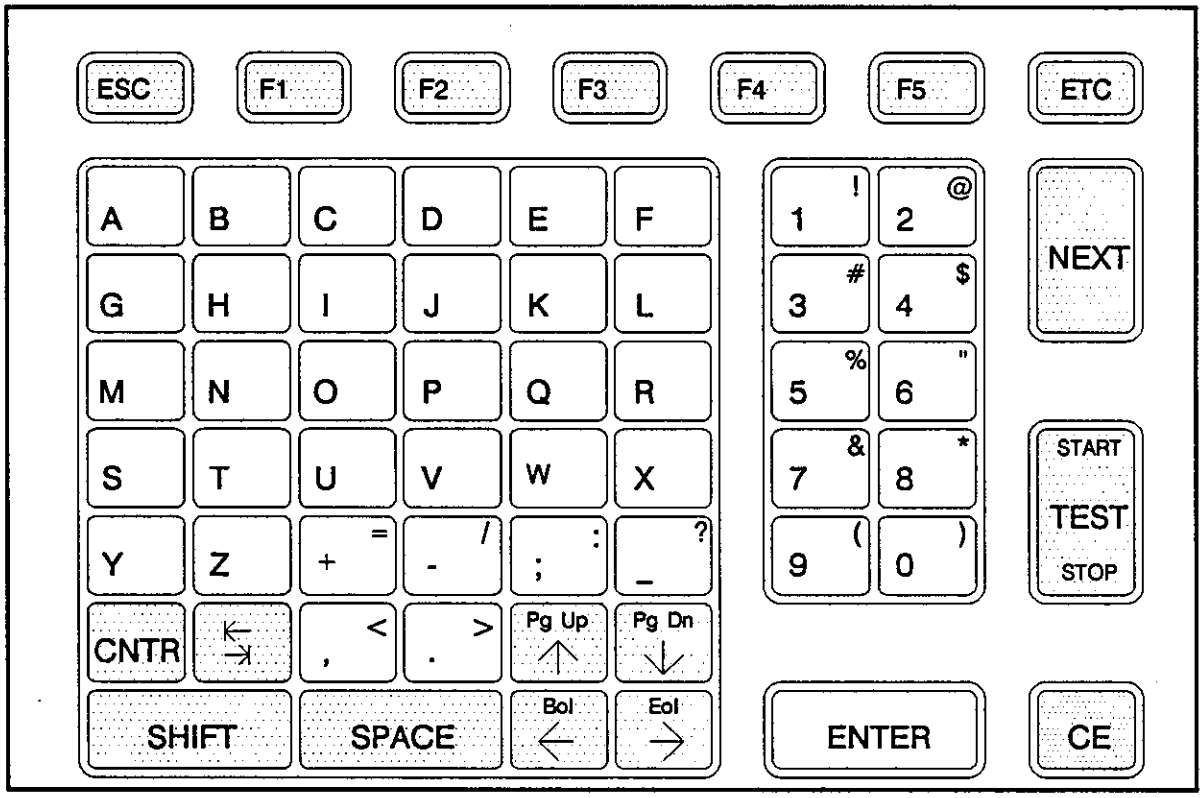
manual sequence develop system

Press ignore to continue with the normal power up screen. You should now see on the screen a System Software revision number. It is associated with the unit's operating features defined in a set of internal ROMs.

Note that ignore will allow us to proceed with keyboard and file operations, but will not permit actual DUT testing unless Selftest first passes.

Finally, observe a Test Clip which inserts into the Interface Buffer. It is keyed and contains only passive components. Larger clips may be used successfully on smaller components as long as Pin 1 (see label) is properly aligned. For places where the standard 16, 24 and 28 pin clips cannot fit onto tightly packed devices, optional 8, 14, 18 and 20 pin sizes are available.

The five patch leads are for the Ground (2), External Input, Reset and $V_{CC}$ jacks. The use of both ground leads is recommended to minimize noise.


ESC
F1
F2
F3
F4
F5
ETC
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
+ =
- /
; :
_ ?
CNTR
, <
. >
Pg Up
Pg Dn
SHIFT
SPACE
Bol
Eol
1 !
2 @
3 #
4 $
5 %
6 "
7 &
8 *
9 (
0 )
NEXT
START
TEST
STOP
ENTER
CE

## The FLUKE 900 Keyboard & Screen

In the following Section you will become familiar with the use of the keyboard, the screen hierarchy and data entry/manipulation. The keyboard layout is shown to the left with the keys explained in functional groups.

### Alphanumeric

Used for data entry, they also print lower case using the Shift key. After pressing for two seconds, the keys repeat.

CE (Clear Entry) performs a backspace function to delete the last character.

SHIFT CE Deletes the last *word* entered.

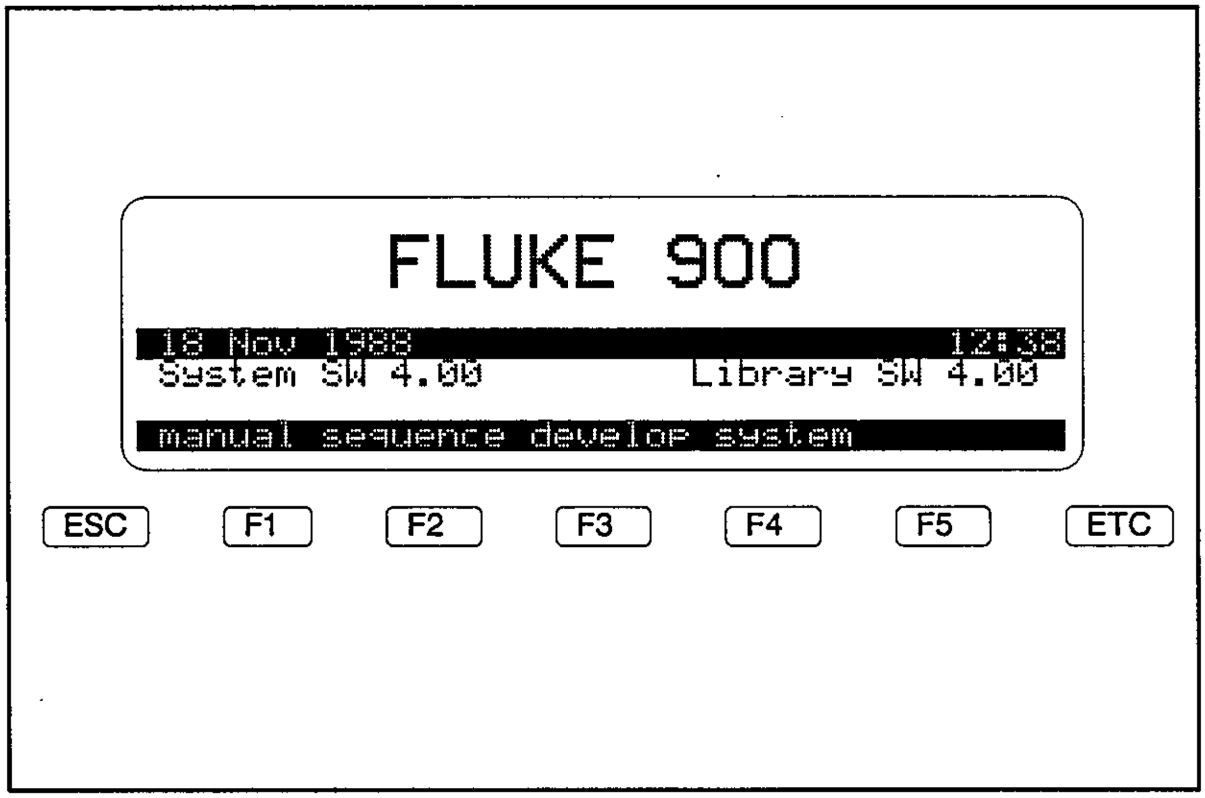
CNTR CE Deletes the last *line* entered.

ENTER Terminates a line of data. As you will see, it typically puts user inputs into the fields of the main display.

### Cursor Control

Movement by space, tab field, page and line

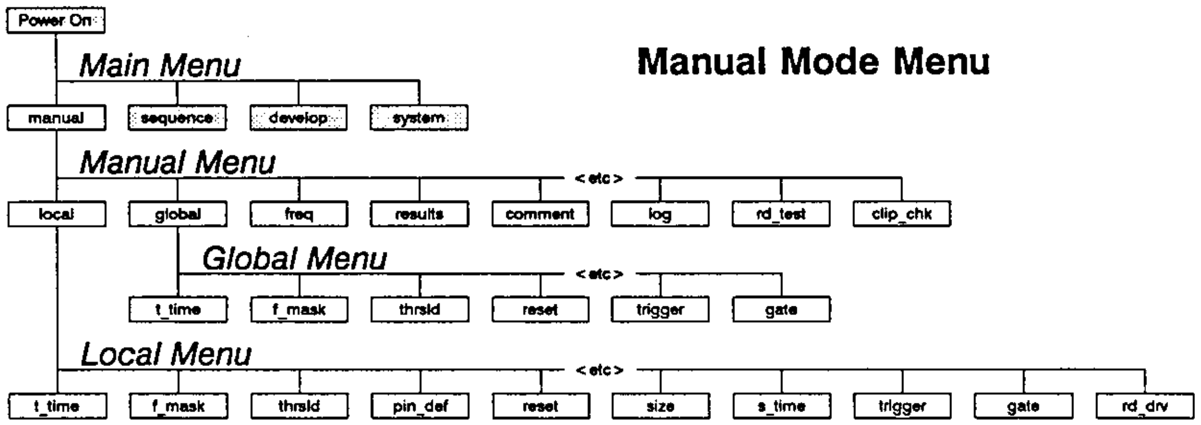
| | |
|---|---|
| SHIFT ← | Move cursor to *beginning* of line |
| SHIFT → | Move cursor to *end* of line |
| SHIFT ↑ | Page *up*. |
| CNTR ↑ | Go to *top* of file |
| SHIFT ↓ | Page *down*. |
| CNTR ↓ | Go to *bottom* of file |



FLUKE 900

```
18 Nov 1988                                12:38
System SW 4.00              Library SW 4.00
```

## Manual Mode Menu



### Test Execution Keys

The TEST and NEXT keys are used in manual and sequence modes to test and step from device to device. Duplicate keys are located on the Interface Buffer.

### Function Keys

Labels for F1 through F5 are found on the bottom line of the display.

| | |
|---|---|
| ESC | Returns control to a previous key level. |
| ETC | Displays additional labels that are present on a current key level. |

## Manual Mode

The main screen has four function key labels on the bottom of the screen, organized into levels that are linked in a tree structure. The hierarchy of sublevels under the manual key is shown on the left.

## Hands-On Exercises

## Manual Mode

## Manual Mode Menu Exercises

1. Match the letter of the function associated with each label from the main menu listed below:

**Function**

a. A mode for testing a single device with device number and test parameters to entered by the user.
b. A mode for setting the tester's configuration for time/date, serial port, audible beep, printer and data logging formats.
c. A mode for testing a board using preprogrammed troubleshooting prompts.
d. A mode for creating new sequences and using file utilities including edit, copy, delete and store.

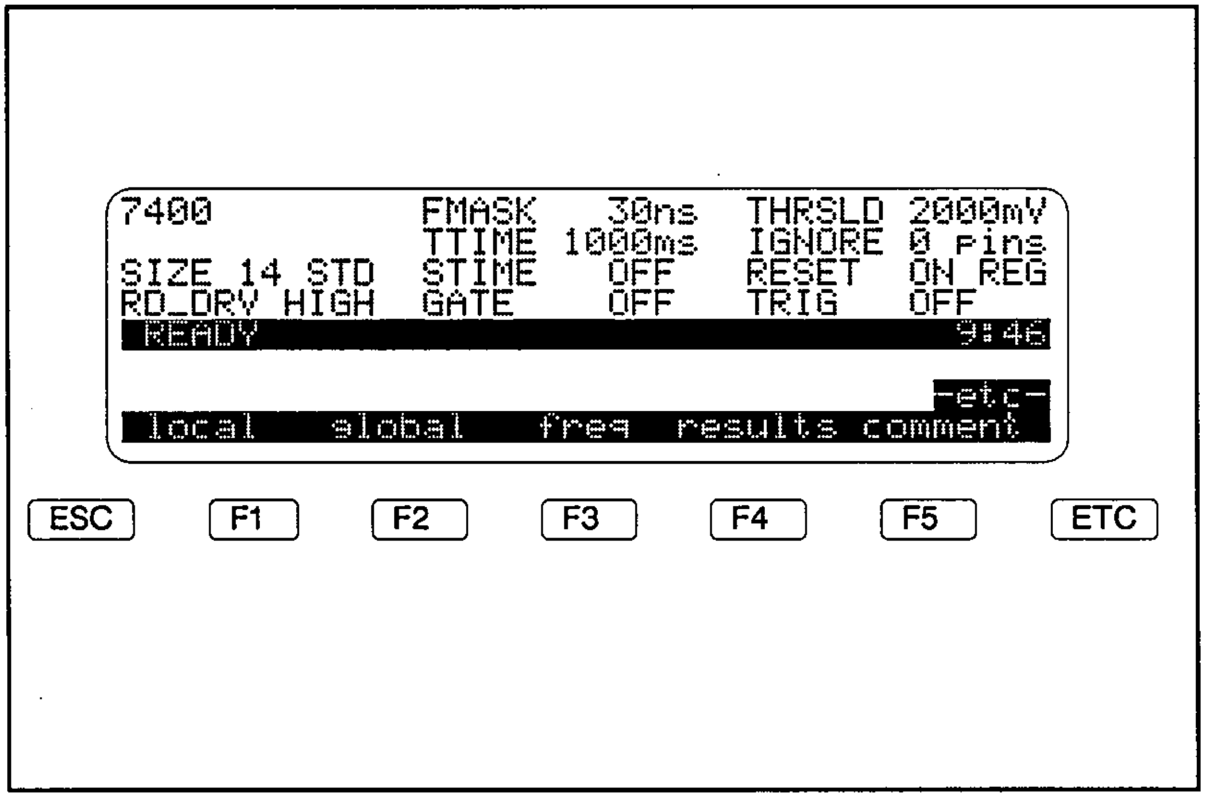
**Main Menu**

1. manual
2. sequence
3. develop
4. system

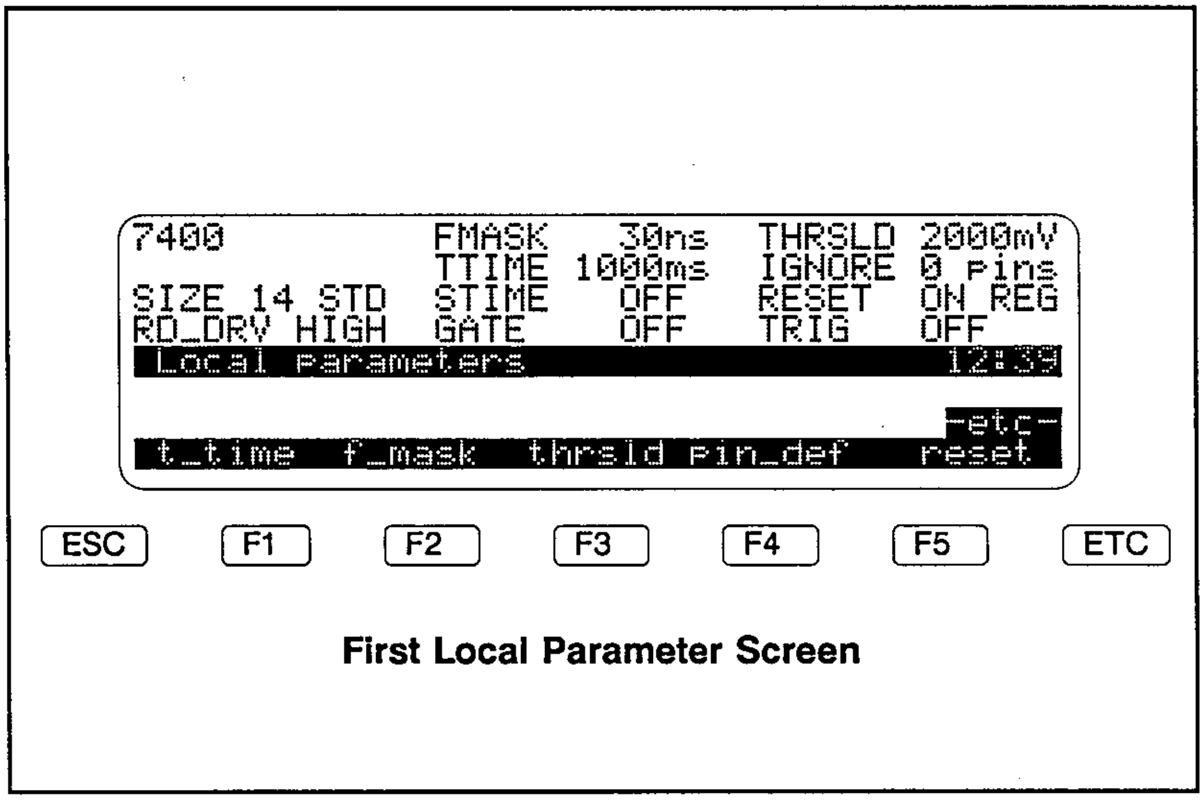
Now press F1 to show the manual Menu. Note that there is an ESC (escape key) on the left and an ETC (etcetera key) on the right of the function keys.

ESC always returns to the previous higher level menu. For example, pressing it from manual mode returns to the Main Menu.

ETC always displays additional labels from the same menu level.



```
7400              FMASK    30ns   THRSLD 2000mV
                  TTIME 1000ms    IGNORE 0 pins
SIZE 14 STD       STIME    OFF    RESET  ON REG
RD_DRV HIGH       GATE     OFF    TRIG   OFF
 READY                                     9:46
                                          -etc-
 local   global   freq  results  comment
```



```
7400              FMASK    30ns   THRSLD 2000mV
                  TTIME 1000ms    IGNORE 0 pins
SIZE 14 STD       STIME    OFF    RESET  ON REG
RD_DRV HIGH       GATE     OFF    TRIG   OFF
 Local parameters                         12:39
                                          -etc-
 t_time   f_mask   thrsld pin_def    reset
```

**First Local Parameter Screen**

Manual mode allows you to test a device while varying parameters for general testing, such as the test duration, and specific parameters unique to the DUT, such as the Performance Envelope. Additional features include the ability to measure frequency and display or record test results.

To specify a device to test, type in the generic number followed by [ENTER]. Try this for a 7400. Notice the test parameters listed on the screen and the LED near the ZIF socket defining a 14 pin device (opposite page).

2. Press `local` to show the test parameter menu. It has two lines of labels. The upper half of the screen has parameters and values corresponding to function key labels (opposite page).

Try the following matchup, and as a class, experiment with changing each parameter (1) through (10). Note that you must press ENTER after changing any setting to confirm the change.

**Definition**

a. Issues a pulse on the patch cord labeled R to force a device to a known state before testing.
b. Defines a length of time that a DUT will be compared to the RD before a pass result is given.
c. Defines a voltage level, above which is logic 1 and below which is logic 0.
d. Defines how close to compare the output signals of DUT and RD.
e. Defines a logic state word on the DUT that selectively masks out normally occurring indeterminate DUT conditions.
f. Defines a time that the tester will try to synchronize DUT and RD into the same state before testing.
g. Allows user to disable or re-enable comparison testing on each pin of a device.
h. Specifies the number of pins on the DUT.
i. Defines logic state words on the DUT pins that trigger the start of test.
j. Allows testing of devices that are weaker than the RD specification of 1 LS load.

**Local Menu**

1. t_time
2. f_mask
3. thrsld
4. pin_def
5. reset
6. size
7. s_time
8. trigger
9. gate
10. rd_drv

The size parameter can be used to specify a device instead of entering the generic number. For example, by entering 14 for size with standard power pins, the tester is configured the same as when we previously entered 7400. Standard power pins are taken as 7 & 14 in this case but you can optionally specify other pins by entering 14 NSTD for the size.

It is useful to think of the test parameters in categories:

**Initialization of DUT and RD**

s_time
reset
trigger

**DUT Functionality**

| | |
|---|---|
| f_mask | defines how close to compare |
| thrsld | defines logic 1 |
| t_time | defines how long to test |
| pin_def | ignores specified pins |
| gate | masks out specified states and ignores them. |

A third category relates to the signals between devices:

**Circuit Conditions**

pin_definition of active, high, low and a specified frequency of observed signals

trigger and gate detection of multichannel signal events.

We will return to discuss these parameters in detail later.

3. Now press [ESC] manual to show the Manual menu and match up the definition and label columns below:

**Definition**

a. General purpose counter to measure signals on any pin or External Input patch lead.
b. Change test parameters for a specific device from their default values.
c. Change default values of test parameters for all devices.
d. Display picture of device with test results highlighted.
e. Permits user comments to be entered from the keyboard (e.g. "U48 replaced")
f. Permits testing or identifying reference device in the ZIF socket.
g. Creates a file or printout of test results and operator activities.
h. Enables/disables the Vcc-Gnd check for clip orientation.

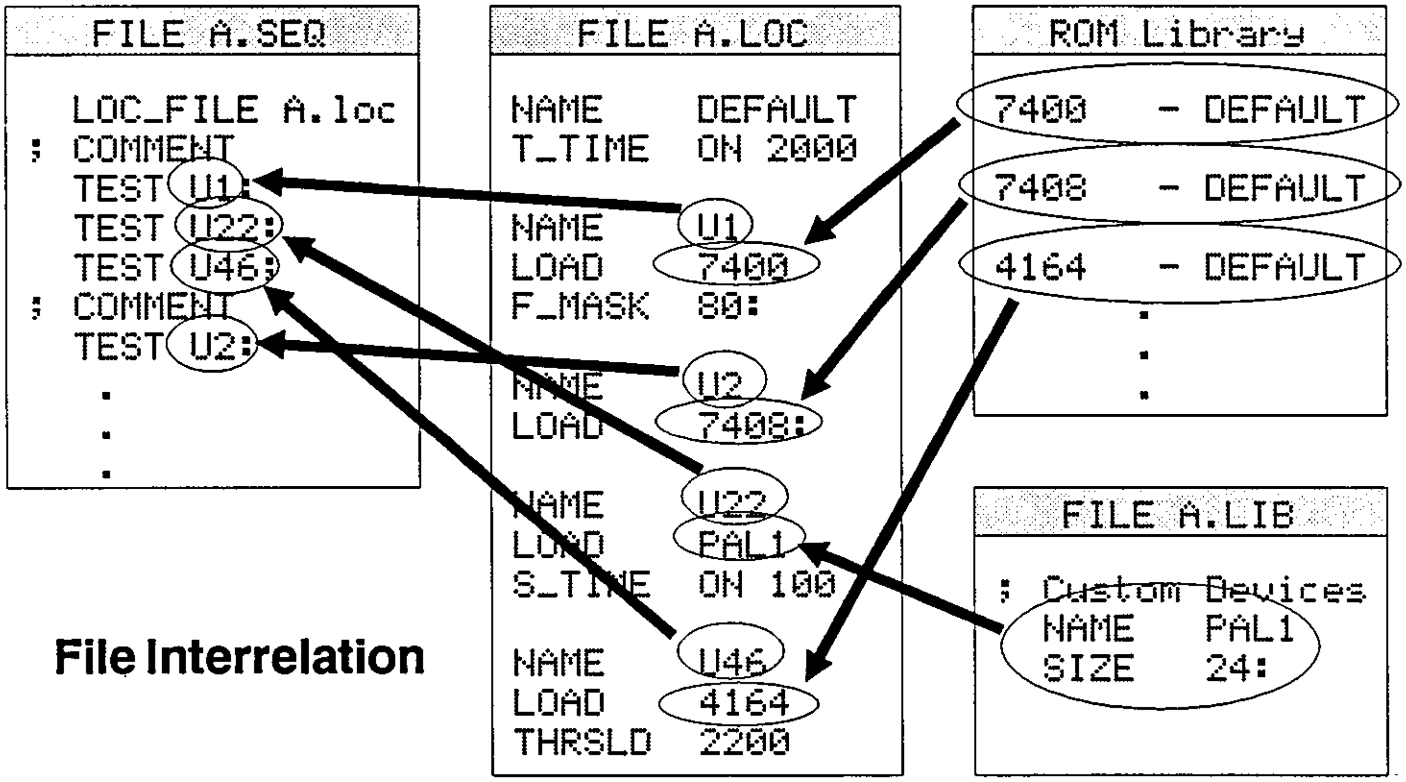
**Manual Menu**

1. local
2. global
3. freq
4. results
5. comment
6. log
7. rd_test
8. clip_chk

4. Select freq from the manual mode menu screen. Select pins 1 through 8 and clip U91. Observe the different frequencies on the various pins. Press [ESC] to back-up to the manual menu.

5. Select at random one of the 14- or 16-pin ICs from the foam block containing all of the RDs. Insert the chip into the ZIF socket and press etc followed by rd_test identify. Answer the prompt(s) depending on the selected chip size and proceed to identify your "unknown" chip.

Let's return to the Main Menu. The usual method is to press ESC to return up the screen hierarchy. Another way to go directly to the main level from any mode and reinitialize parameters to default is to press, simultaneously, SHIFT and ESC. (Note: No stored memory files are lost, unlike CNTR ESC which clears memory through Selftest.) Press SHIFT ESC at this time.



```
FILE A.SEQ

  LOC_FILE A.loc
; COMMENT
  TEST U1:
  TEST U22:
  TEST U46:
; COMMENT
  TEST U2:
    .
    .
    .
```

```
FILE A.LOC

NAME     DEFAULT
T_TIME   ON 2000

NAME     U1
LOAD     7400
F_MASK   80:

NAME     U2
LOAD     7408:

NAME     U22
LOAD     PAL1
S_TIME   ON 100

NAME     U46
LOAD     4164
THRSLD   2200
```

```
ROM Library
7400    - DEFAULT
7408    - DEFAULT
4164    - DEFAULT
            .
            .
            .
```

```
FILE A.LIB

; Custom Devices
  NAME    PAL1
  SIZE    24:
```

**File Interrelation**

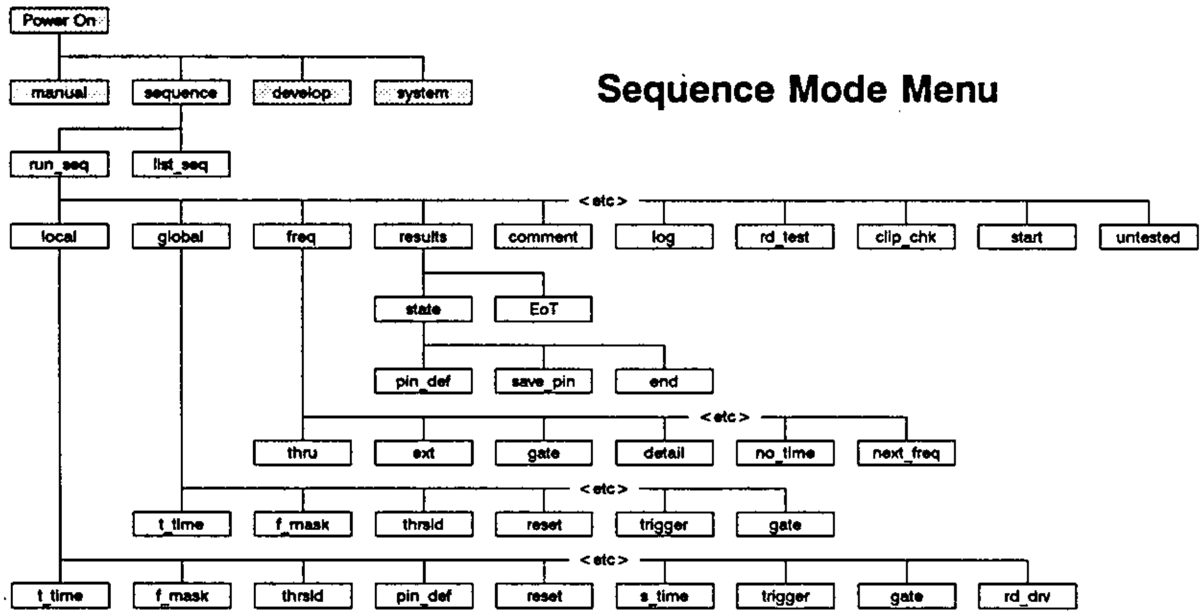
## File Interrelation

Notice the filenames in the opposite frame have upper case extenders (A.SEQ, A.LOC, and A.LIB) as opposed to the lower-case extenders shown on the display. The difference between upper-case and lower-case extenders is the former belongs to the ASCII *source* file and the latter belongs to its compressed and *compiled* counterpart. Only the upper-case source files are user-readable and alterable.

Note that a single .LOC file may be used in several .SEQ files to test the same devices in a different order for troubleshooting purposes. In this case, the first line of the display would show a different .seq name from the .loc name.

Observe in the file diagram the user-generated library file A.LIB containing the device named PAL1. This can be a very convenient way to cross-reference devices with custom part numbers to their standard device equivalents.

## Sequence Mode Menu



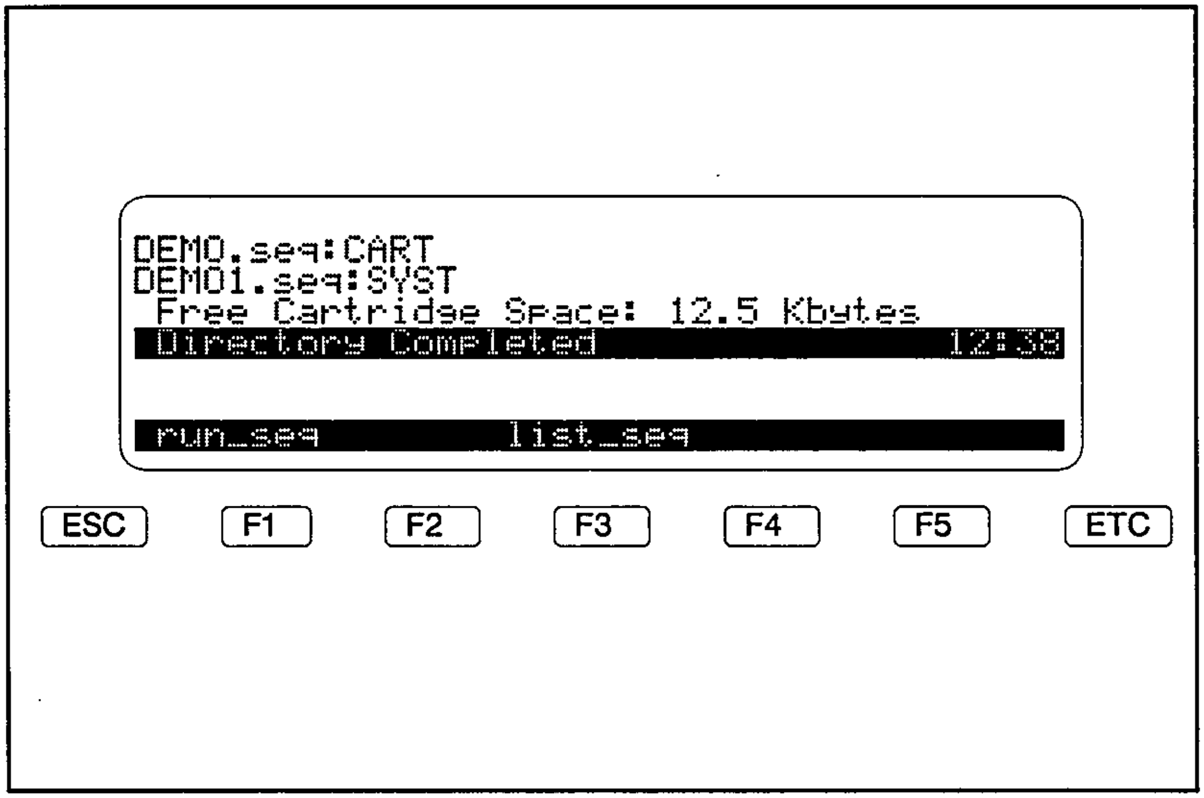
## Sequence Mode

We will now examine a preprogrammed sequence and do some simple file manipulation with the Cartridge. The Sequence mode menu tree is shown at the left.

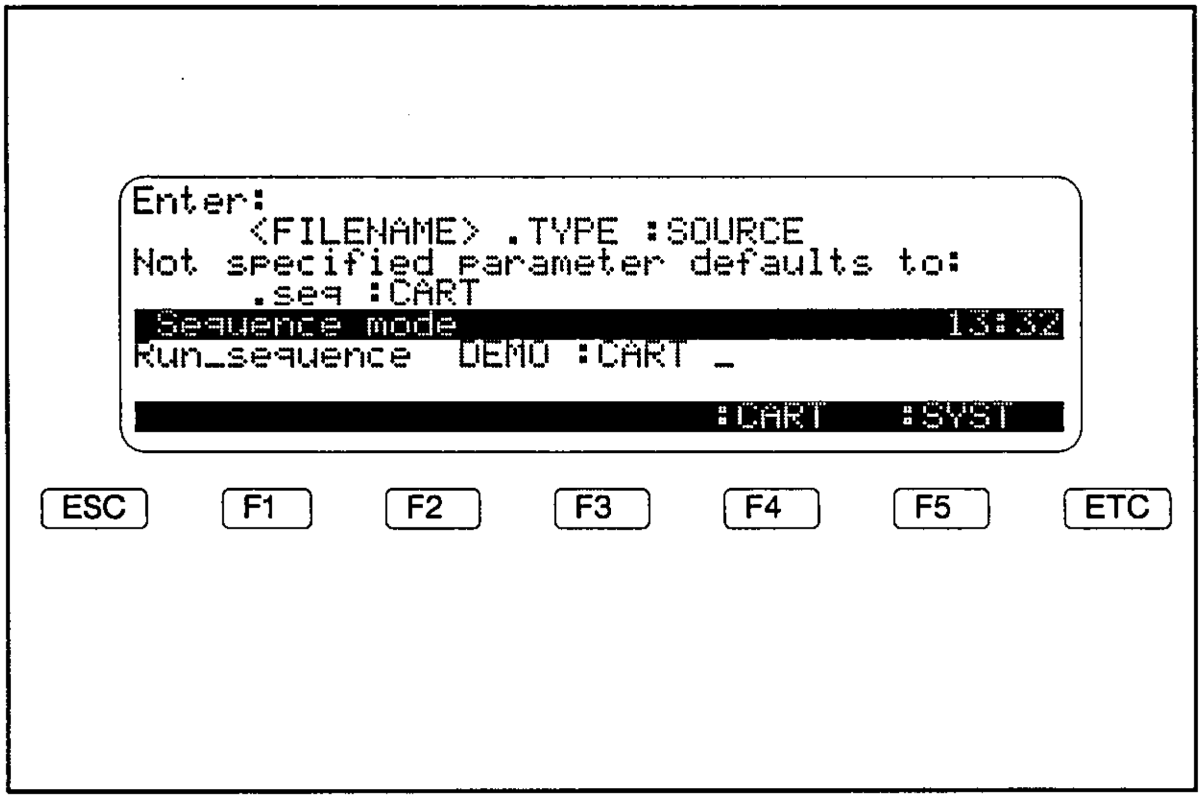
First, insert the black cartridge with its label facing up into the right side slot of the FLUKE 900. Press se-quence from the Main Menu screen and observe the new menu that gives two choices:

list_seq
: Provides a directory of sequences on cartridge.

run_seq
: Will step an operator through testing a board.



```
DEMO.seq:CART
DEMO1.seq:SYST
 Free Cartridge Space: 12.5 Kbytes
 Directory Completed                          12:38

 run_seq          list_seq
```



```
Enter:
     <FILENAME> .TYPE :SOURCE
Not specified parameter defaults to:
     .seq :CART
 Sequence mode                                13:32
Run_sequence  DEMO :CART _

                        :CART    :SYST
```

Proceed to list the sequences on the cartridge. The sequence for the Demo UUT appears as:

```
DEMO.seq:CART
```

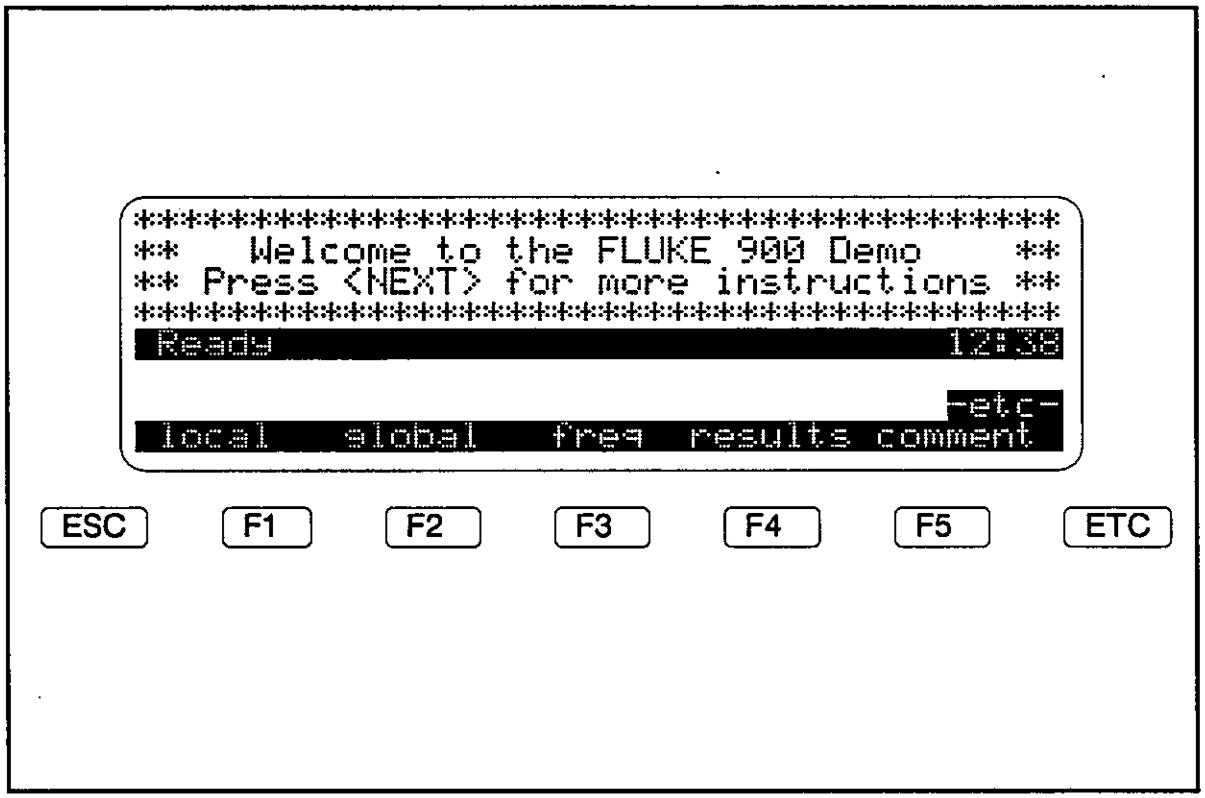
Now press `run_seq` and an explanation of the format appears. Simply respond to the prompt

```
Run sequence  _
```

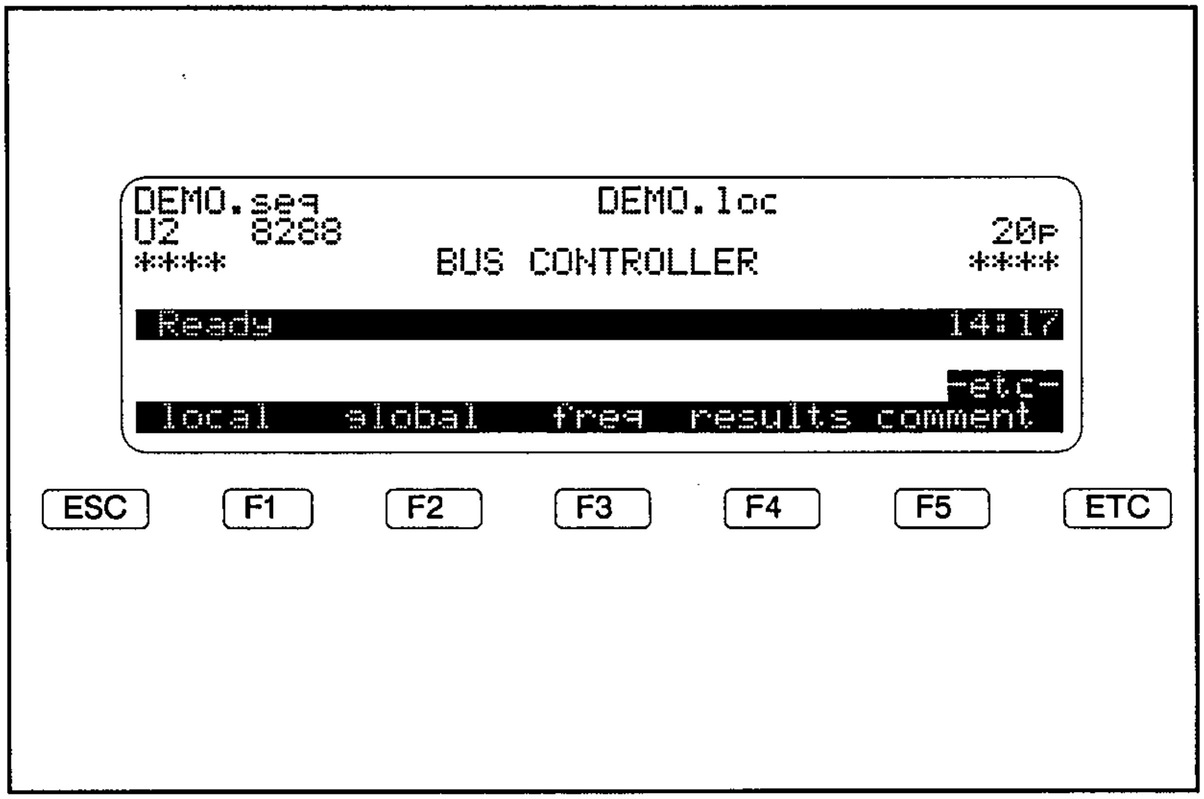
by typing

[D] [E] [M] [O] [ENTER]

The series of screens so far should appear as shown on the left and the resulting screen on the next page.



```
*****************************************
**    Welcome to the FLUKE 900 Demo    **
** Press <NEXT> for more instructions  **
*****************************************
 Ready                             12:38
                                   -etc-
 local   global   freq  results comment
```



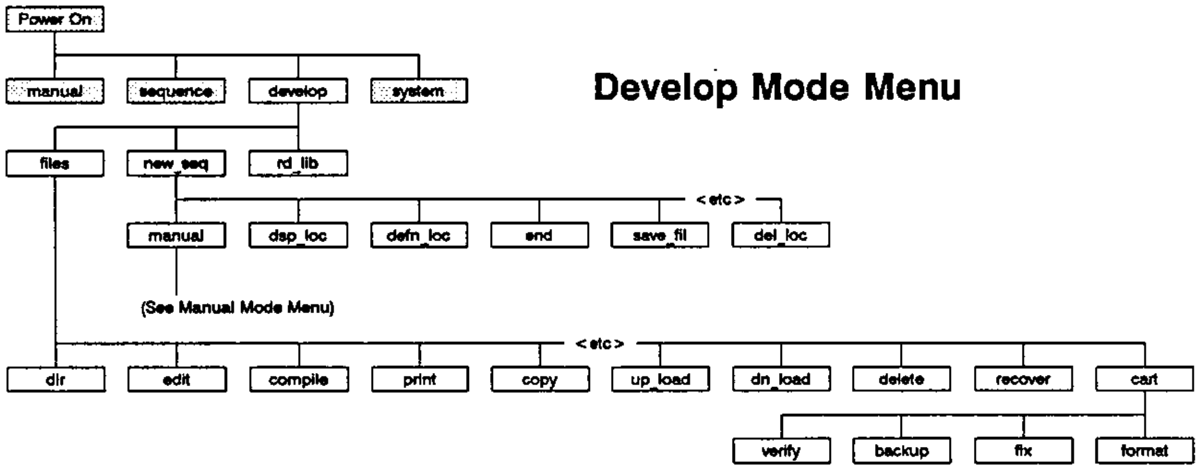
```
DEMO.seq                DEMO.loc
U2    8288                          20p
****        BUS CONTROLLER          ****

 Ready                             14:17
                                   -etc-
 local   global   freq  results comment
```

Press the [NEXT] key five or six times and observe how a sequence prompts you to test a series of devices and follow instructions. Now press [U][2][ENTER] on the keyboard. Notice how the display changes to prepare for testing U2. This illustrates that a sequence is only a nominal order of devices for troubleshooting. An experienced user will want to override this to go directly to the devices he suspects on a failing board. You can decide how you want to use the system:

1. Follow a preordered sequence.
2. Jump around according to observed failure mode.
3. Break up a single sequence into several smaller ones according to board functions and failure modes.

Note that the sequence name appears twice on the first line. This is because a sequence is really a combination of two files:

1. Location (**.loc**) file which is a listing of each board location (e.g. U2) with its associated device and local parameters.
2. Sequence (**.seq**) file which is an ordered list of the device locations for troubleshooting plus added comments.



## Develop Mode Menu

- dir
- edit
- compile
- print
- copy
- up_load
- dn_load
- delete
- recover
- cart
  - verify
  - backup
  - fix
  - format

## Develop Mode

At this point, let's exit the sequence run mode by pressing the ESC key. You will have to confirm that you wish to exit the sequence mode by pressing yes. Now press ESC again to return to the Main Menu.

We will now briefly cover the sub-menu under develop as shown in the frame opposite. Full operation will be described in greater detail later.

develop has the following sub-menu:

files
: Edit, copy, delete and perform other file utilities.

new_seq
: Generate new sequence files.

rd_lib
: Create and run test pattern vectors on the RD ZIF socket.

The following are the main functions that can be accessed from files level:

dir
: Provides directory of files in the cartridge and system memory (i.e. :CART, :SYST)..

edit
: Create and/or modify source files. Source files have uppercase extensions (e.g. .SEQ, .LOC, .LIB, .LOG, .LST)..

compile
: Convert source files (.SEQ, .LOC and .LIB) into FLUKE 900 executable format. The file produced has the same name and device as the source file, but has a lower case extension (.seq, .loc, .lib).

print

Print source files ( .SEQ, .LOC, .LIB, .LOG, .LST) through the RS232C port. The print file format, such as page width and height, and the RS232C communication standard are specified through system mode.

copy

Create a duplicate of the specified file. Both upper case and lower case files can be copied.

up_load

Copy the specified file to the RS232C port, in order to store it for future retrieval on another device, such as a PC. Unlike the print command, no formatting of the file data takes place.

dn_load

Copy the data input through the RS232C port into a specified file. This is used to retrieve files previously stored or generated on an external device such as a PC.

delete

Delete the specified file from the cartridge or system memory. Deleted files can be recovered as long as memory space on the device permits, via the recover command.

recover

Recover a previously deleted file.

cart

Access the following four cartridge utilities:

verify

Check out cartridge data integrity. Verify is a non-destructive operation.

backup

Create a duplicate cartridge. The backup command formats the destination cartridge prior to writing on it.

fix

Fix any damaged (hidden) files that were corrupted during writing to the cartridge. This could occur if the FLUKE 900 is accidentally powered down during a cartridge access function, or a LOG file was left open.

format

Format the data cartridge. This is a **destructive** operation that will **permanently** remove all files from the cartridge.

**NOTE:** The cartridges have an ON/OFF write protect switch on them. It is recommended that the switch be left in the ON position at all times when data is *not* being written into the cartridge.

## File Manipulation Exercise

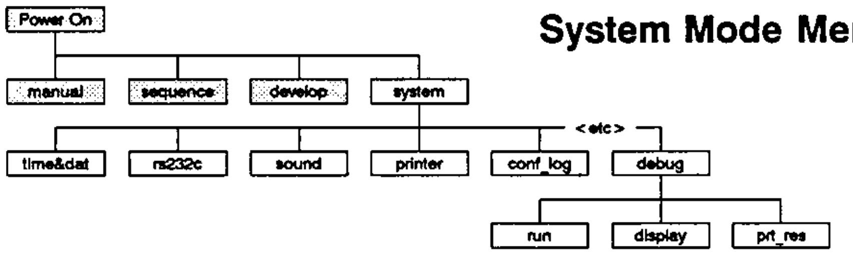
With a cartridge in place, list the contents of the cartridge with the dir softkey. Next, copy the contents of the cartridge to System Memory with the copy command and list the contents of System Memory to confirm the transfer took place.

Delete one or all of the files from System Memory.

Use the recover softkey to "undelete" one or all of the files previously deleted.

Lastly, make a copy of the ROM cartridge to Non-Volatile RAM cartridge using the backup softkey.

## System Mode Menu



- time&dat
- rs232c
- sound
- printer
- conf_log
- debug

## System Mode

The menu below system relates to tester configuration and has the menu tree on the facing page.

time&dat
: Sets the time and date displayed on the status line and test result log

rs232c
: Sets the communication port baud rate and protocol

sound
: Sets the audible beep to off, low or high

printer
: Sets the format to be used when printing source file via the print command

conf_log
: Defines up to five different log format files, which determine what type of test results will be stored into the user-selected log file

debug
: Used to initiate the Selftest procedure.

The system parameters are stored in a battery-backed-up RAM in the main unit and therefore are retained during power down. At this point, if you wish, change the time/date and sound parameters of your FLUKE 900.

## Answers for the Questions in Section 2

1. a. manual
   b. system
   c. sequence
   d. develop

2. a. reset
   b. t_time
   c. thrsld
   d. f_mask
   e. gate
   f. s_time
   g. pin_def
   h. size
   i. trigger
   j. rd_drv

3. a. freq
   b. local
   c. global
   d. results
   e. comment
   f. rd_test
   g. log
   h. clip_chk

# Section 3
# Testing In-Circuit Devices

- **Combinatorial**
  - **NAND Gates**
  - **Bus Drivers**
- **Synchronous**
  - **Up/Down Counters**
  - **J-K Flip-Flops**
- **Programmable**
  - **UARTs**
  - **RAM**

**ROM**

**4-bit Counter**

**Shift Register**

**Interrupt Controller**

## Digital Device Categories

For the purpose of comparison testing, digital devices are classified into three groups as follows:

1. **COMBINATORIAL**
   Those devices whose output pins reflect immediately the conditions on the input pins. They have no internal memory, but may have tristate output pins (e.g. NAND gate, Bus Driver).
2. **SYNCHRONOUS**
   Devices with internal memory whose state can be observed or inferred from an output pin. They always have a clock input line (e.g. up/down counter, J-K Flip Flop).
3. **PROGRAMMABLE**
   Devices that require data to be written into them before outputs are valid (e.g. UART, RAM). In general, these devices have internal hidden memory states.

Which categories would these devices fall into?

- ROM
- 4-bit counter
- Shift register
- Interrupt Controller

*(answers at end of chapter)*

## Performance Envelope Categories

1. **Parameters for defining a device**
2. **Parameters for initializing RD and DUT to the same state**
3. **Parameters of the Performance Envelope**

## Performance Envelope Categories

It is important to know what category a certain device falls into so that the proper test parameters may be set. Recall that the submenu below local was a series of Performance Envelope parameters (f_mask, t_time, reset, etc.). We can put them into three groups as follows:

1. **Parameters for defining a device:**

   IC or chip name (i.e. 7400)

   size (number of pins)

   rd_drv (reference device ability to drive 1 LS load).

Most devices and associated parameters are predefined.

2. **Parameters for initializing RD and DUT to the same state:**

   reset (A pulse issued on the R patch lead to reset a board. Pulse duration and offset from the comparison time may be set.)

   s_time (Sync Time synchronizes RD and DUT devices. All synchronous and some programmable devices use this.)

   trigger (starts comparison after a user-defined state appears on DUT)

3. **Parameters of the Performance Envelope:**

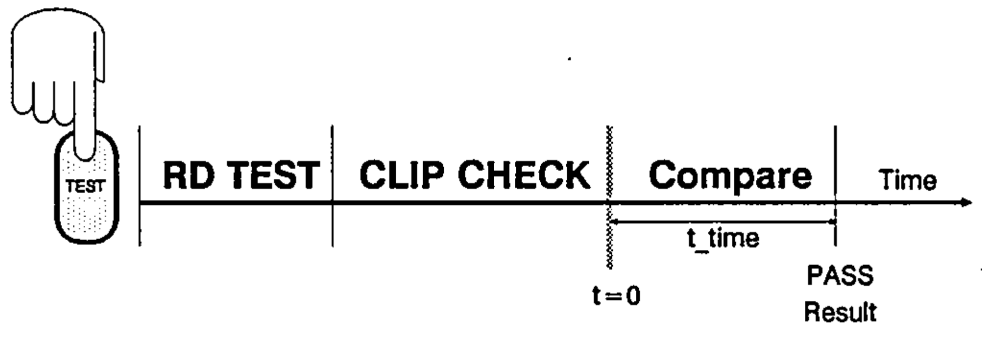
   t_time (Test Time defines the duration of comparison)

   f_mask (Fault Mask defines how close to compare RD and DUT)

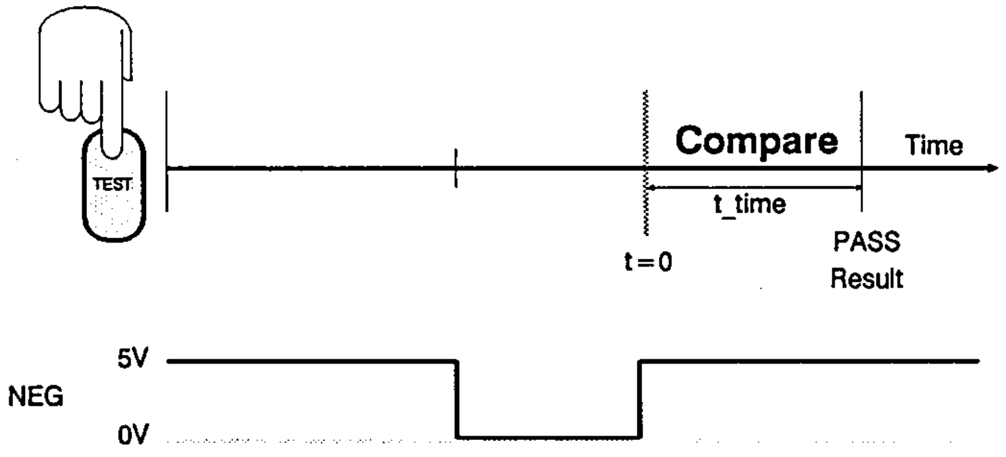
   thrsld (Threshold defines logic 1 level)

   pin_def (Pin Definition permits specified DUT pins to be ignored).

   gate (enables and disables comparison from user-defined valid state).



Test Cycle



Reset Pulse

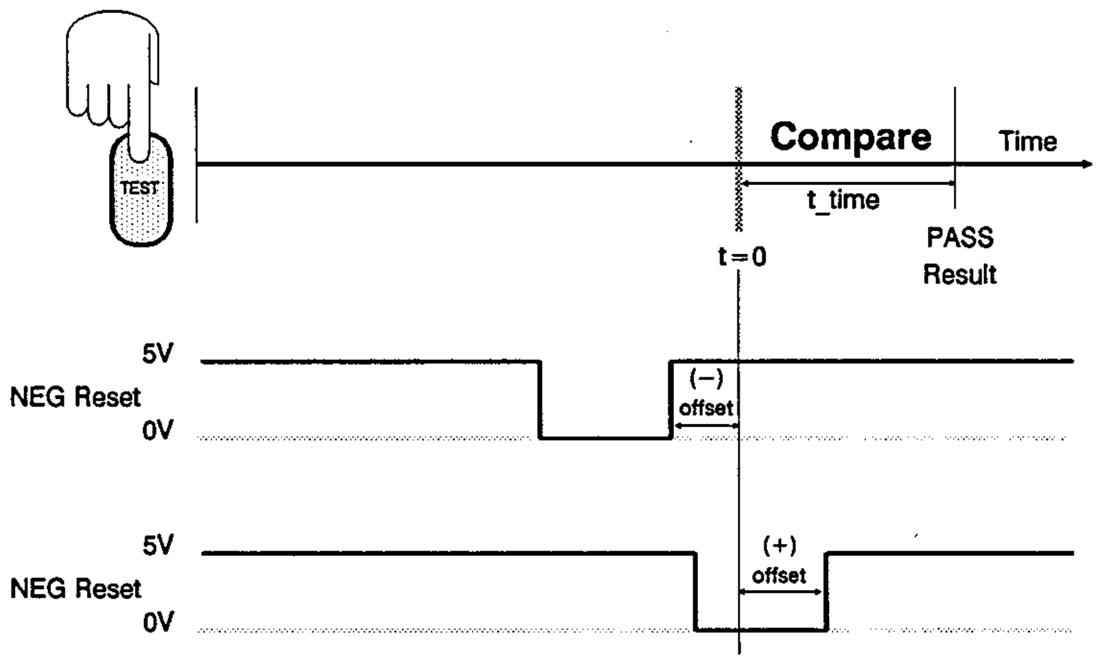
## Test Cycle

Let us draw a timing diagram of what happens after you clip onto a DUT and press TEST. It is known as a test cycle diagram shown on the facing frame.

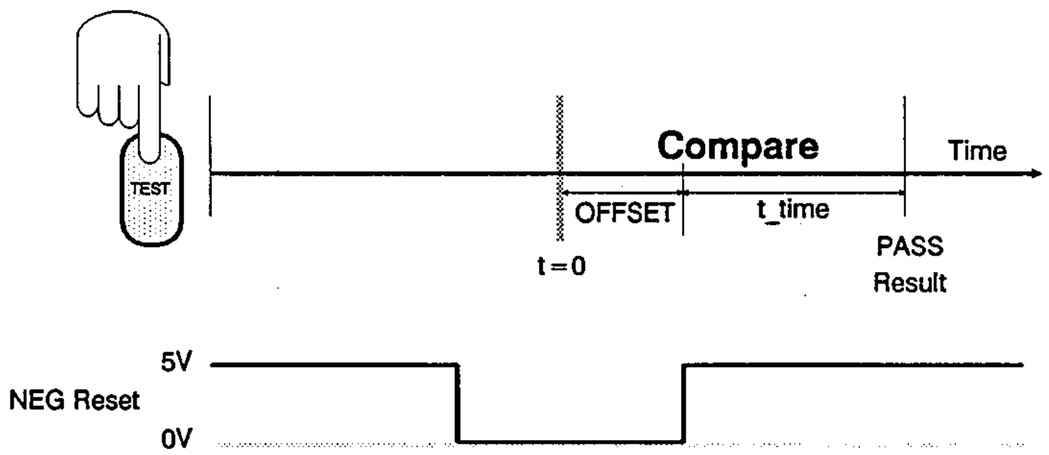
Notice that Clip Check and RD Test are performed first. This always occurs (unless rd_test or clip_check are off) so we will omit showing it in further test cycle diagrams. For the purposes of the Time-to-Fault value in the test result, it is measured from the start of the comparison (t = 0).

## Reset Pulse

For most boards and DUTs, a reset pulse is chosen as the main initialization parameter.



**Offset Parameter Diagram**



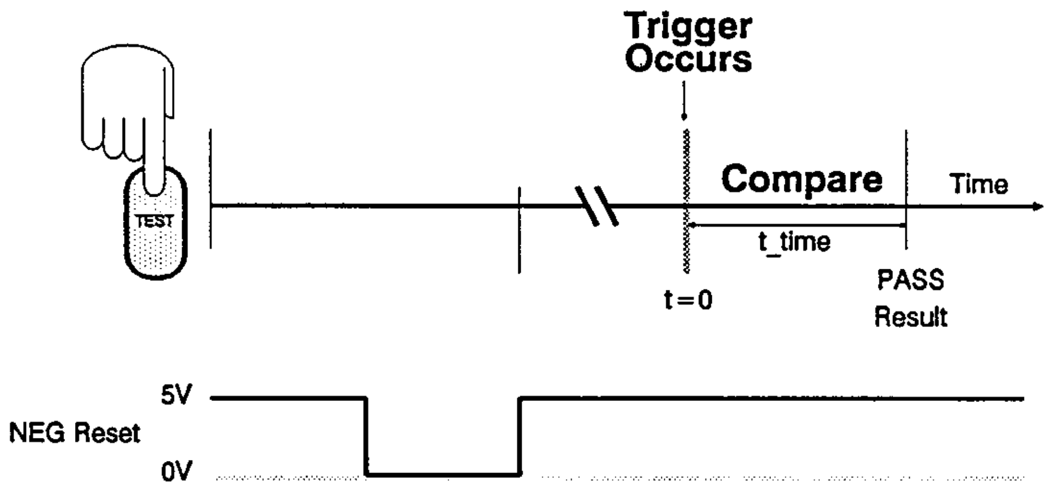
**Comparison during the Reset Pulse**

## Reset Offset

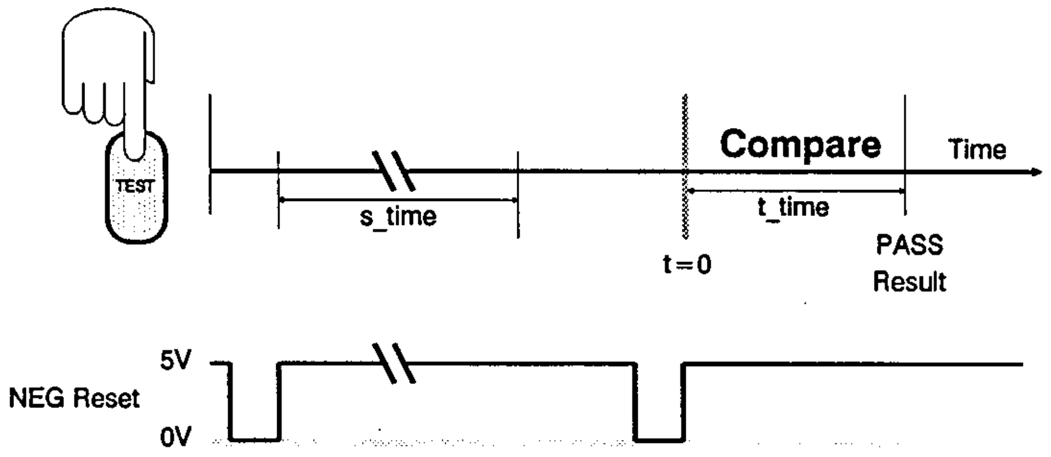
The Reset pulse has a definable duration (100 ms by default) and a polarity which can be positive or negative. The pulse may also be shifted relative to the start of comparison by assigning a value to the offset parameter. Shown in the example to the left is a negative offset (e.g. -30 ms).

This time margin after Reset, during which comparison is inhibited, is used for certain programmable devices which must be initialized by onboard activity.

Shown in the following frame is a positive offset. This test cycle is used only for devices that must be compared during the reset pulse itself (e.g. the board's reset circuit).



Trigger Parameter



Synchronization Time

## Trigger Parameter

Comparison may also be offset for initialization purposes using the trigger parameter.

Trigger effectively offsets comparison until two specified logic state events appear on the DUT pins or the EXT patch lead.

## Synchronization Time

When the s_time parameter is enabled, multiple resets are issued while library-resident synchronizing methods are executed.

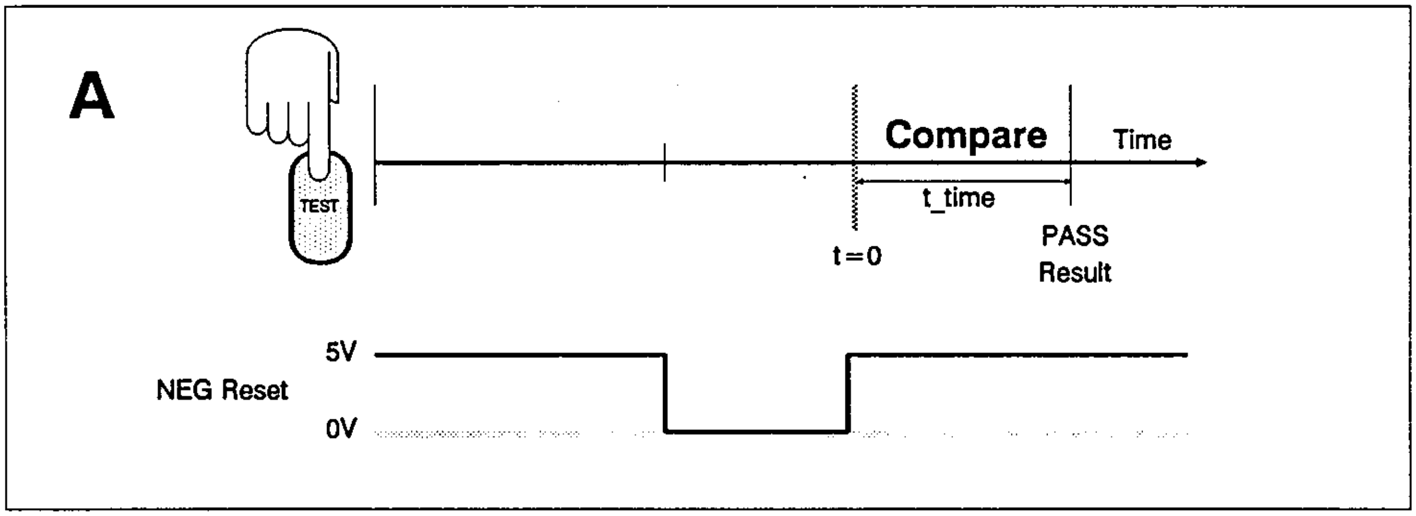
s_time is a time interval that lasts only long enough for the DUT and RD to be put into the same state. Various techniques are employed during s_time which fall into two classes:

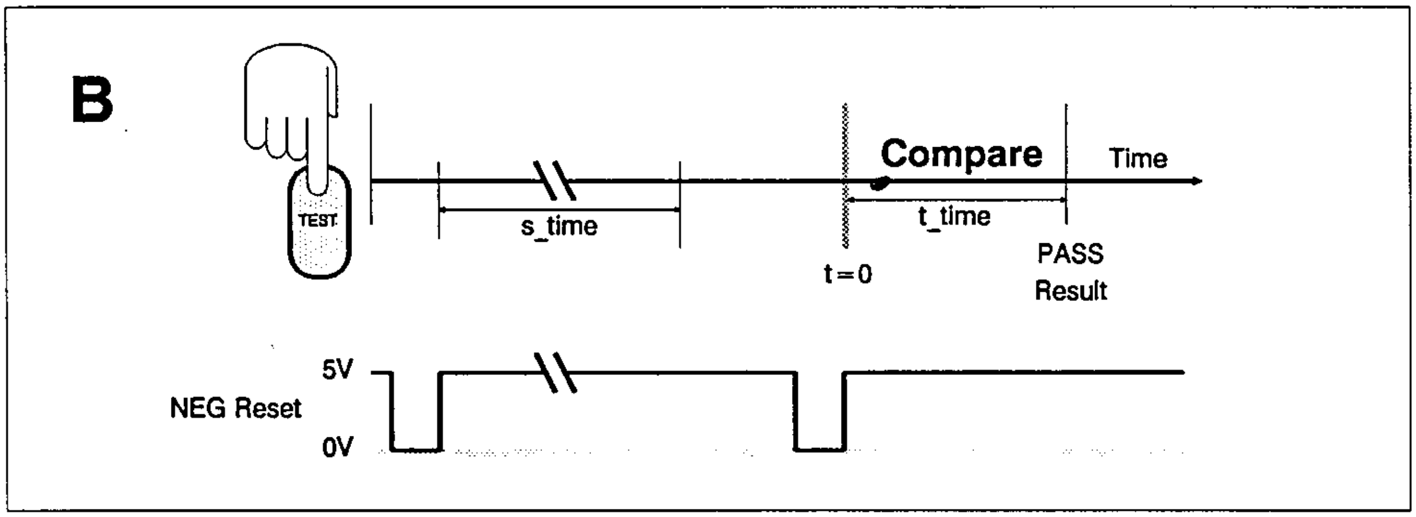
**Method 1:**
For an active DUT, specified signal activity is checked that would guarantee DUT and RD to be synchronized (e.g. pulse on Clear pin)

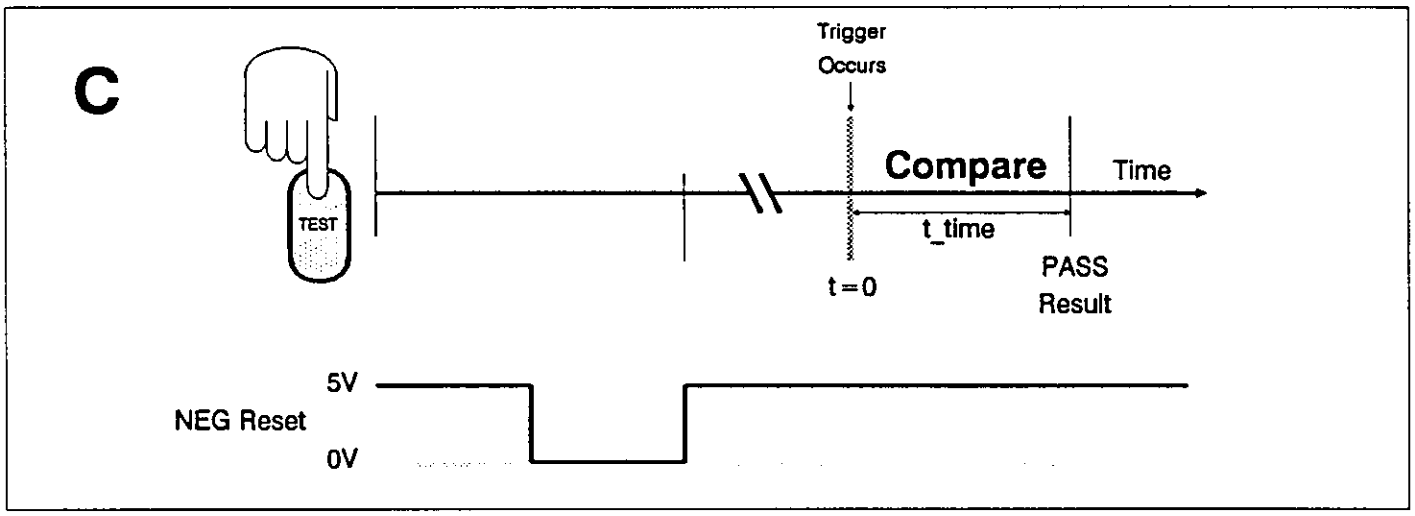
**Method 2:**
For an inactive DUT, the RD is separately stimulated through all its states until its output pins match the DUT.

## Gate

During the comparison interval for all the test cycles, the gate parameter can selectively enable or disable comparison based on a specified state of the DUT pins and the EXT patch lead. This has applications for defining a window of valid activity on some bus devices.


A
TEST
Compare
Time
t_time
t=0
PASS
Result
5V
NEG Reset
0V


B
TEST
s_time
Compare
Time
t_time
t=0
PASS
Result
5V
NEG Reset
0V


C
Trigger
Occurs
TEST
Compare
Time
t_time
t=0
PASS
Result
5V
NEG Reset
0V

## Default Test Cycles

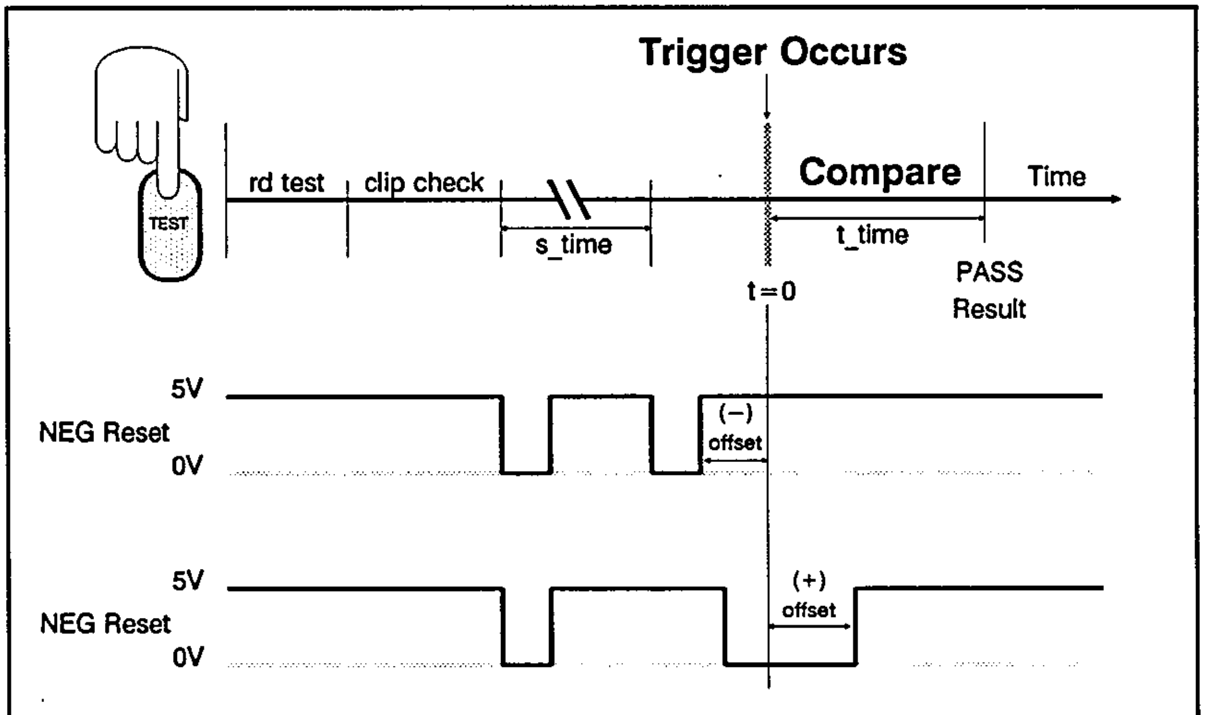
Recall that devices fall into three general categories:

- Combinatorial
- Synchronous – clocked IC's without unknown states
- Programmable – clocked IC's with unknown states.

When a device is selected by generic number, the resident library information sets up the appropriate test cycle. Match the test cycles (A, B, or C) on the opposite page with the type of device below:

| | |
|---|---|
| **Synchronous:** | ______ |
| **Programmable:** | ______ |
| **Combinatorial:** | ______ |

Library Test Cycle definitions may be expanded by the user as will be shown in Section 5.



**Test Cycle Master Diagram**

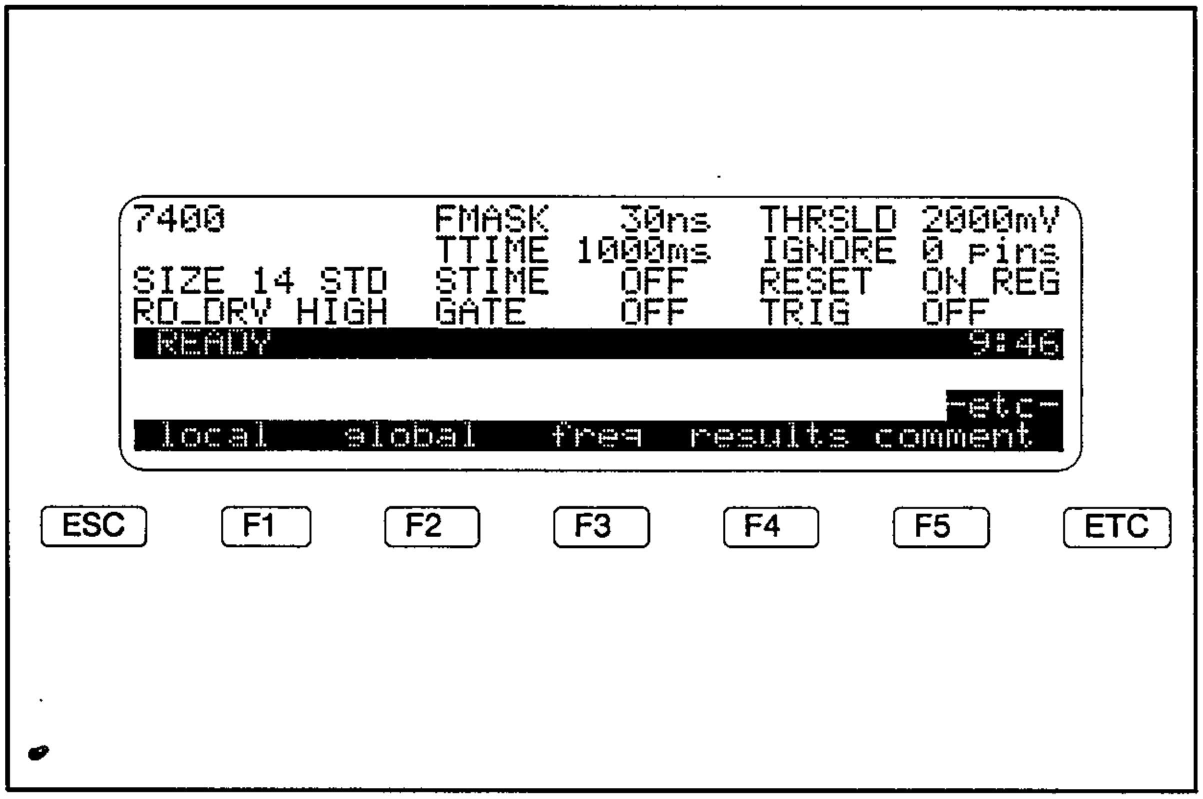
## User Definable Test Cycles

Variations in test cycle from the library default settings are usually only necessary for some programmable devices because of their special application in a circuit.

The parameters s_time, trigger and reset offset are for adjusting the test cycle. They are also the main setup parameters for programmable devices not found in the standard library. Consider, as an example, an 8259 Interrupt Controller which must receive command word data to define its mode of operation. Prior to this, the pins are indeterminate and unable to be compared. Three ways to mask out this indeterminate period are:

1. s_time enabled while checking for the occurrence of command word initialization. This is normally part of the library data.
2. trigger set to start comparison after the occurrence of command word initilization.
3. reset offset set to a negative value sufficient to allow board activity to initialize the 8259. This is the simplest to specify but potentially misleading for boards that are inoperative.

The time-to-fault reading is measured from the start of comparison, which is also the trigger point when this parameter is enabled. Some advice on applications to microprocessor-based boards is appropriate here. Since there are a number of clock cycles that make up an instruction, the micro will take a slightly different amount of time to reset, depending upon when the pulse edge occurs. The time-to-fault will only be consistently repeatable if it is measured from an EXT trigger attached to the processor **Reset Out** or **Memory Read** signal.



```
7400               FMASK    30ns  THRSLD 2000mV
                   TTIME 1000ms   IGNORE 0 pins
SIZE 14 STD        STIME    OFF   RESET  ON REG
RD_DRV HIGH        GATE     OFF   TRIG   OFF
 READY                                     9:46

                                          -etc-
 local   global   freq   results  comment
```

## Hands-On Exercises Using the Demo UUT

Let us proceed to test some combinatorial, synchronous and programmable devices on the Demo UUT. We will try a couple of devices in manual mode and then continue under develop new_seq mode to store our parameter settings in a cartridge sequence.

Prepare the Demo UUT and FLUKE 900 to begin testing individual devices.

Put the FLUKE 900 into manual mode.

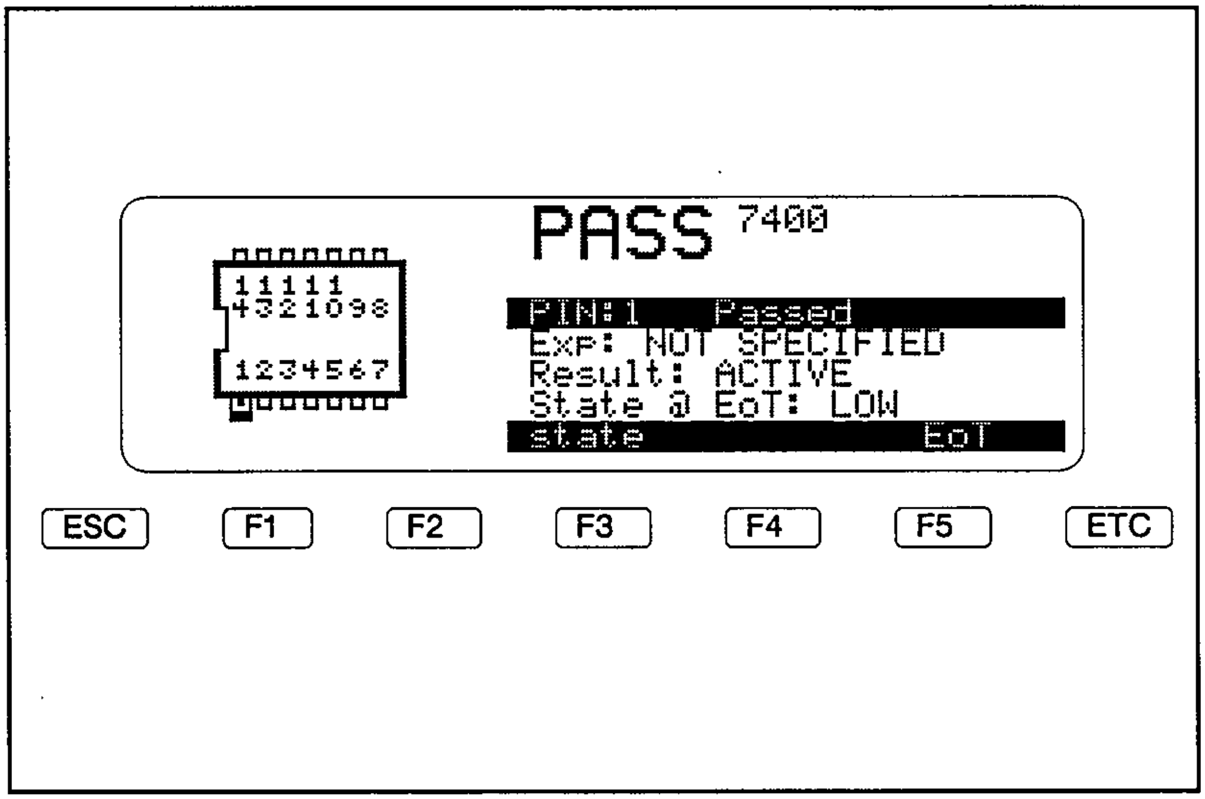
Connect the two ground (G) patch leads to the GND post on the Demo UUT. Two ground leads connected to different ground points on a test board will improve noise immunity where this is a problem.

Connect the R lead from Interface Buffer to Demo UUT Reset post. V is not connected here since we will generate a 5V reset pulse internally in the FLUKE 900. Optionally, we could have used Vcc or another on-board voltage to drive the reset to a level required on the board.

Plug the 16 pin test clip into the Interface Buffer.

Press manual from the main screen level.

Follow the instruction on the command line of the display and enter a device. Try typing [7][4][0][0][ENTER].



PASS 7400

```
PIN:1    Passed
Exp: NOT SPECIFIED
Result: ACTIVE
State @ EoT: LOW
state              EoT
```

This combinatorial device is a quadruple 2 input NAND gate. It was recognized by the standard library in the tester and default parameter values displayed. Without clipping on anything, press the [TEST] button.

Notice how the test will not proceed until a functioning 7400 is properly inserted in the ZIF socket. The RD test is a truth table of test vectors for the Reference Device. For ROMs, a checksum is calculated based on a partial reading of the device. For PALs, a checksum is calculated based on a standard stimulus word to the input pins. RD test gives a high assurance that the Reference Device is functioning correctly.

Take a 7400 from the RD tray, put it into the ZIF socket and clamp the lever into place.

Let's try a test now, without clipping, and observe the clip check pretest results.

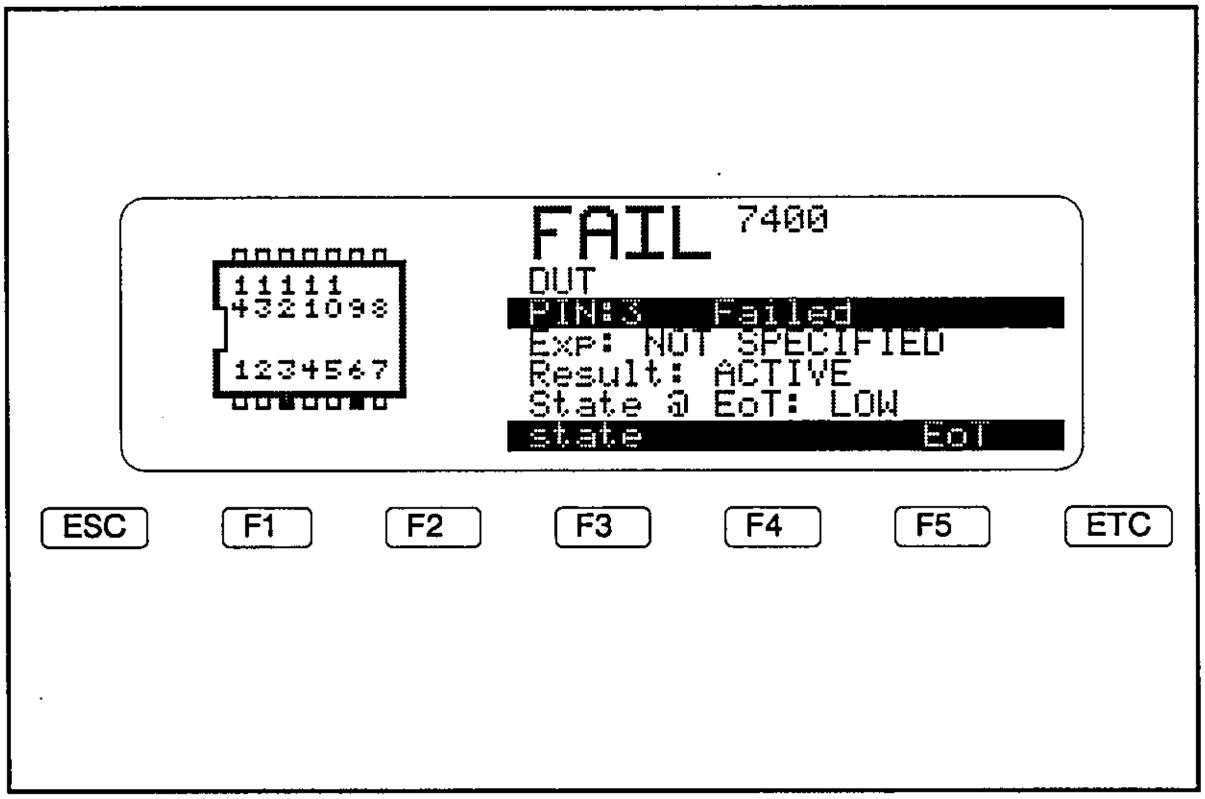
Attach the clip to U14 on the Demo UUT, however, for the purpose of this exercise *reverse* the clip so pin 14 contacts pin 1 on the device. Press [TEST] and observe the result.

Reorient the clip properly and try again.

Our first test result!

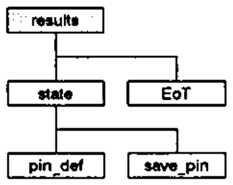
(Note: If this did not PASS, make sure you are firmly clipped on the DUT.)

After a PASS, examine the result in more detail by pressing results. This graphic depiction appears automatically with a failure. The PASS you see should appear similar to the facing frame.



FAIL 7400
DUT
PIN:3 Failed
Exp: NOT SPECIFIED
Result: ACTIVE
State @ EoT: LOW
state EoT

## Results Screen Hierarchy



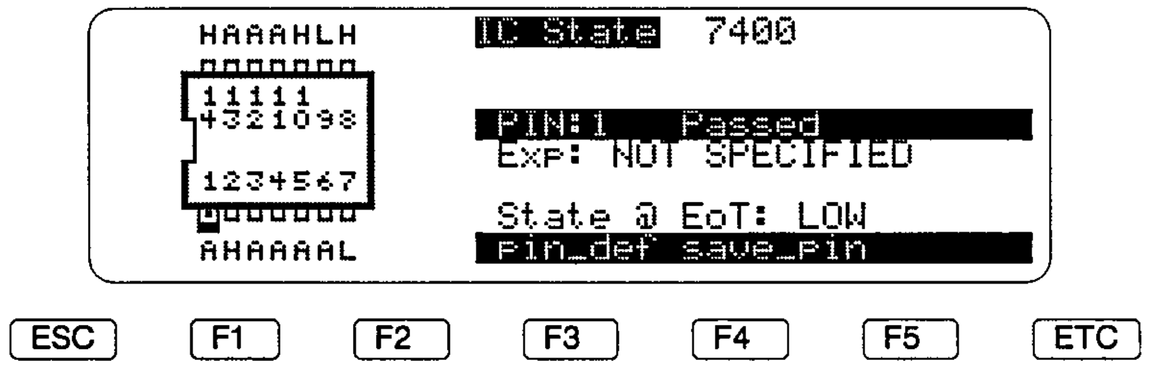
Cursor control keys move the pointer from pin to pin and the reverse-highlighted line displays the comparison result followed by three lines of additional information (if enabled under pin_def).

- an expected activity: high, low, active or specific frequency
- an actual activity: observed levels or value
- level observed precisely as the test ended.

After a FAIL result, the facing graphic screen is automatically displayed.

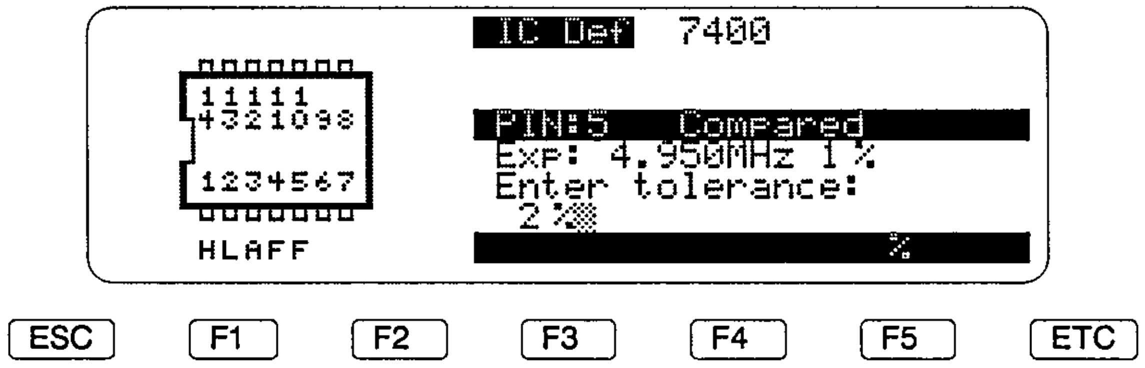
Notice that there are two softkeys with the results screens: state and EoT. These show a picture of the activity seen during the test and state at end of test for all pins. The time-to-fault is also shown on the EoT screen.

The results screen hierarchy is shown on the facing frame.



```
IC State    7400

PIN:1    Passed
Exp: NOT SPECIFIED

State @ EoT: LOW
pin_def save_pin
```

L = always low (logic 0)
H = always high (logic 1)
A = active (high and low)



```
IC Def    7400

PIN:5    Compared
Exp: 4.950MHz 1 %
Enter tolerance:
 2 %
                  %
```

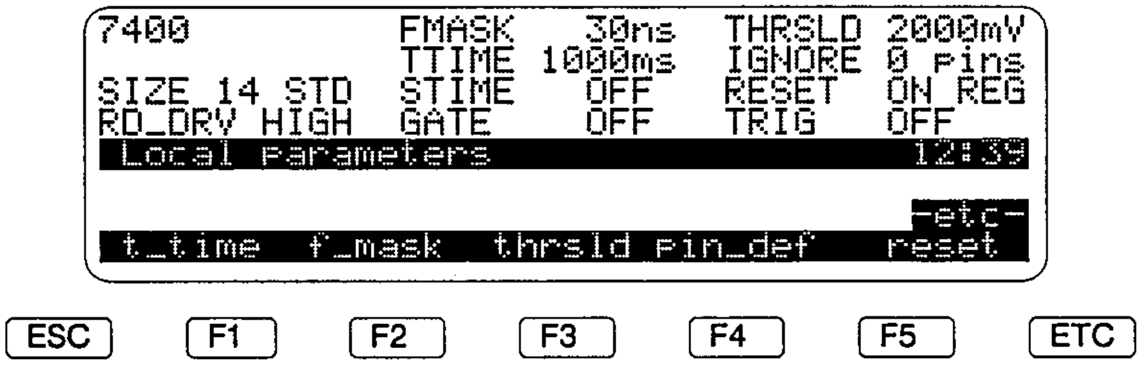
**Defining Frequency on a Pin**

## Activity Profile

You can learn the activity profile of a good device by pressing state after a test result and saving the status of any pin(s) you wish. Be certain to press end to exit this mode to confirm all the pin saves. (Note: If ESC is used, the pin status will not be saved.) The state screen appears on the facing frame.

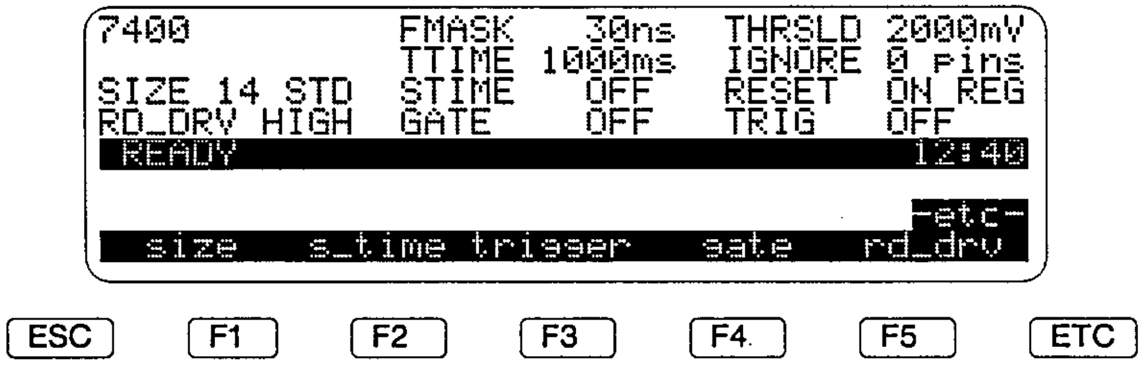
The "save_pin" feature is a convenient way to define the pin status from your observation of a good device, but it can also be done using the local parameter pin_def. With this more comprehensive parameter you can program further expected results for a pin:

| | | |
|---|---|---|
| ignore | = | disable comparison |
| compare | = | enable comparison |
| A | = | pin has to be active |
| L | = | pin has to be always low (logic 0) |
| H | = | pin has to be always high (logic 1) |
| F | = | pin has to have the user defined frequency, +/- the specified tolerance |



```
7400              FMASK    30ns   THRSLD 2000mV
                  TTIME  1000ms   IGNORE 0 pins
SIZE 14 STD       STIME     OFF   RESET  ON REG
RD_DRV HIGH       GATE      OFF   TRIG   OFF
 Local parameters                         12:39

                                          -etc-
 t_time  f_mask  thrsld pin_def          reset
```

**First Local Parameter Screen**



```
7400              FMASK    30ns   THRSLD 2000mV
                  TTIME  1000ms   IGNORE 0 pins
SIZE 14 STD       STIME     OFF   RESET  ON REG
RD_DRV HIGH       GATE      OFF   TRIG   OFF
 READY                                    12:40

                                          -etc-
  size   s_time trigger    gate          rd_drv
```

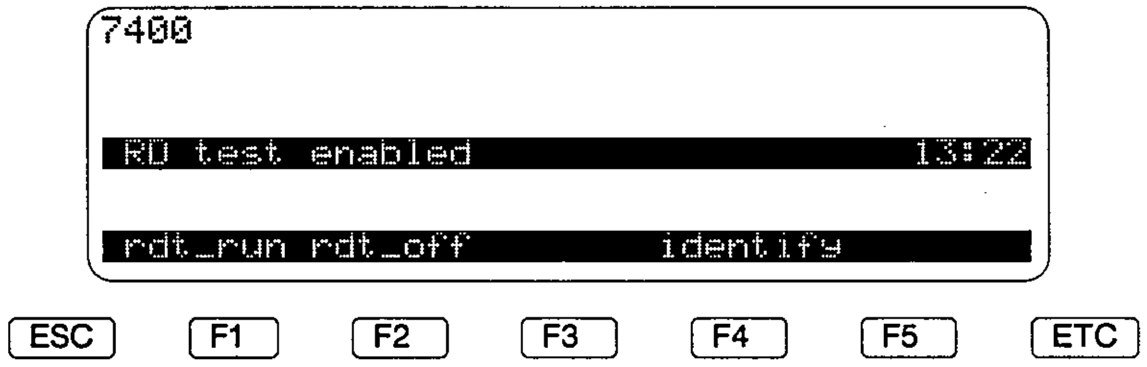
**Second Local Parameter Screen**

Now, let's change some testing parameters. Press local to display the two menus shown opposite. (Pressing [ETC] displays the second set of parameters.)

What happens when you raise threshold to 3000 mV? (Press thrsld and type [3][0][0][0][ENTER] at the prompt. Then press [TEST])

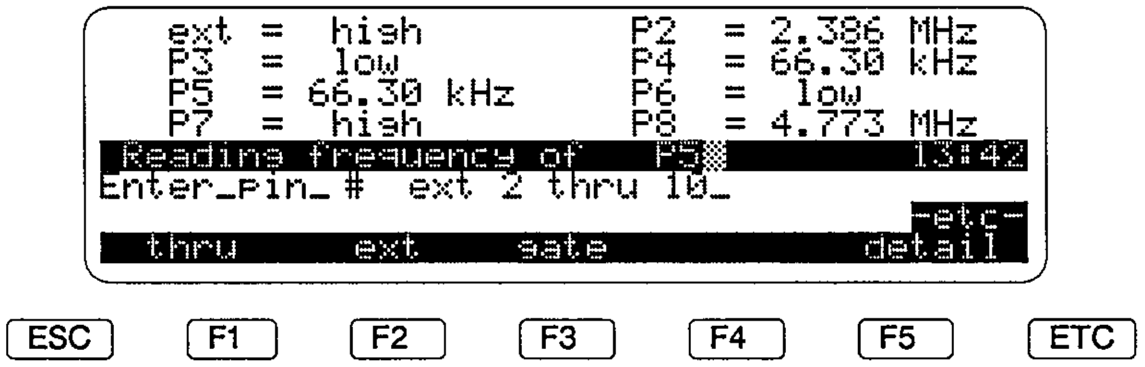
Press [ESC], then expand f_mask until the DUT passes at the new threshold.

Try a few other parameter changes to familiarize yourself with the procedure. For example, ignore a failing pin with the pin_def key and set test time (t_time) to continuous (remember to press [ENTER] after the t_time value).



```
7400


 RD test enabled                              13:22

 rdt_run rdt_off          identify
```

**RD Test Screen**



```
    ext  =   high              P2   = 2.386 MHz
    P3   =   low               P4   = 66.30 kHz
    P5   = 66.30 kHz           P6   =  low
    P7   =   high              P8   = 4.773 MHz
 Reading frequency of   P5                 13:42
Enter_pin_#  ext 2 thru 10_
                                            -etc-
  thru       ext      gate                 detail
```

**Frequency Counter Screen**

Let us examine the RD Test for this device that is done automatically every time you press TEST. From the Manual Mode (as indiated on the status line, press ETC to bring up the rd_test key.

You may test a 7400 in the ZIF socket by pressing rdt_run or disable the automatic RD Test by pressing rdt_off. The identify key will match an unknown device to the standard library equivalent as we will do later. Note that ROMs, PROMs and PALs will have another key at this level, c_sum, which defines a custom RD Test result.

Next, we will use the frequency counter feature to monitor the clock generator chip located at U91 on the board. From the manual mode, select the frequency option freq and clip onto U91. We can specify the pins we wish to monitor using SPACE as a separator and ENTER as a line terminator or specify a range using the thru key. For example, to monitor pins 2,3,4,5,6,7,8,9,10 of U91 and the external lead T, enter ext 2 thru 1 0 ENTER and the resultant screen will appear similar to the format shown opposite.

The cursor control keys will scroll the screen to show further pins.

Additionally, pressing detail provides pulse width and duty cycle information on the signal characteristics. The no_time option must be selected to reliably measure below 100 Hz. Normally, after .01 seconds of inactivity, the counter will time out and assume there is no signal on the pin. With the no_time option selected, a key designated next_freq appears, allowing you to manually move the frequency counter to another pin if it is stuck monitoring an inactive pin.

## Demo UUT Kit

| | |
|---|---|
| U14 | 7400 |
| U2 | 8288 |
| FU3 | PAL22V10 |
| U62 | 8259 |
| U91 | an analog/digital oscillator chip |

1. **Start new sequence.**
2. **Change global parameters, if necessary.**
3. **Define IC and local parameters, if necessary.**
4. **Define location (repeat steps 3 & 4 for each device.**
5. **Terminate sequence creation.**

## Creating a Small Sequence

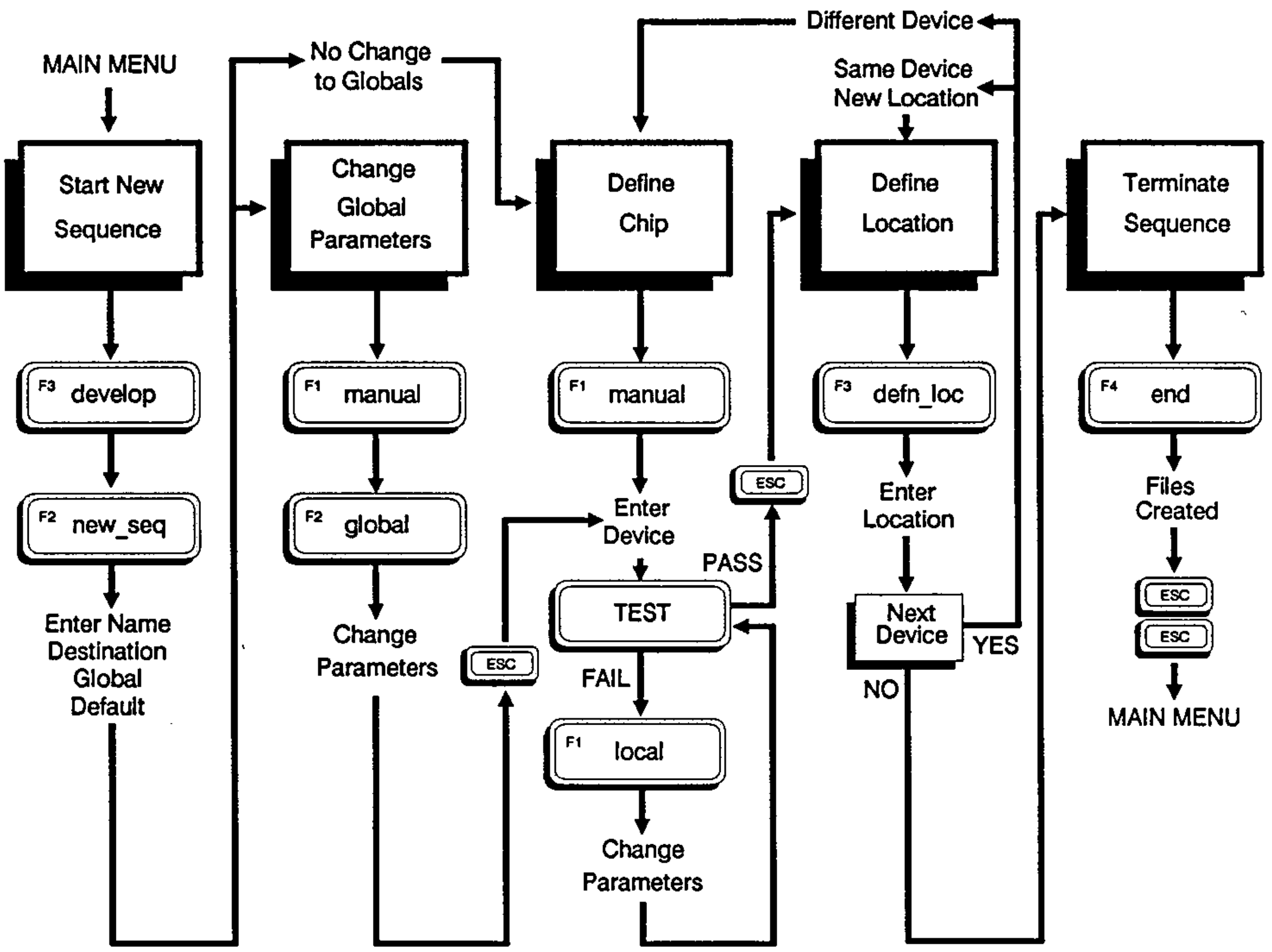
Testing a board with a sequence is a way of storing all the parameters available in manual mode for each device and arranging them in an order that is a sensible troubleshooting method. A sequence complements the skill of a user by providing a standard troubleshooting order and allowing him to jump out of the prescribed order if he wants to verify his hunches about where the fault is.

As an example, let us take five devices, ensure the parameters will allow them to pass on the good board (Demo UUT) and store the procedure as a cartridge sequence. The devices are:

| | | |
|---|---|---|
| U14 | 7400 | a simple combinatorial NAND gate |
| U2 | 8288 | a complex "combinatorial" Bus Controller |
| FU3 | PAL22V10 | a synchronous registered PAL (custom device located on Fault daughterboard) |
| U62 | 8259 | a programmable Interrupt Controller |
| U91 | 8284 | an analog/digital oscillator chip |

The procedure consists of five steps, beginning and finishing with the main menu:

1. Start new sequence.
2. Change global parameters, if necessary.
3. Define IC and local parameters, if necessary.
4. Define location (repeat 3, 4 for each device).
5. Terminate sequence creation.

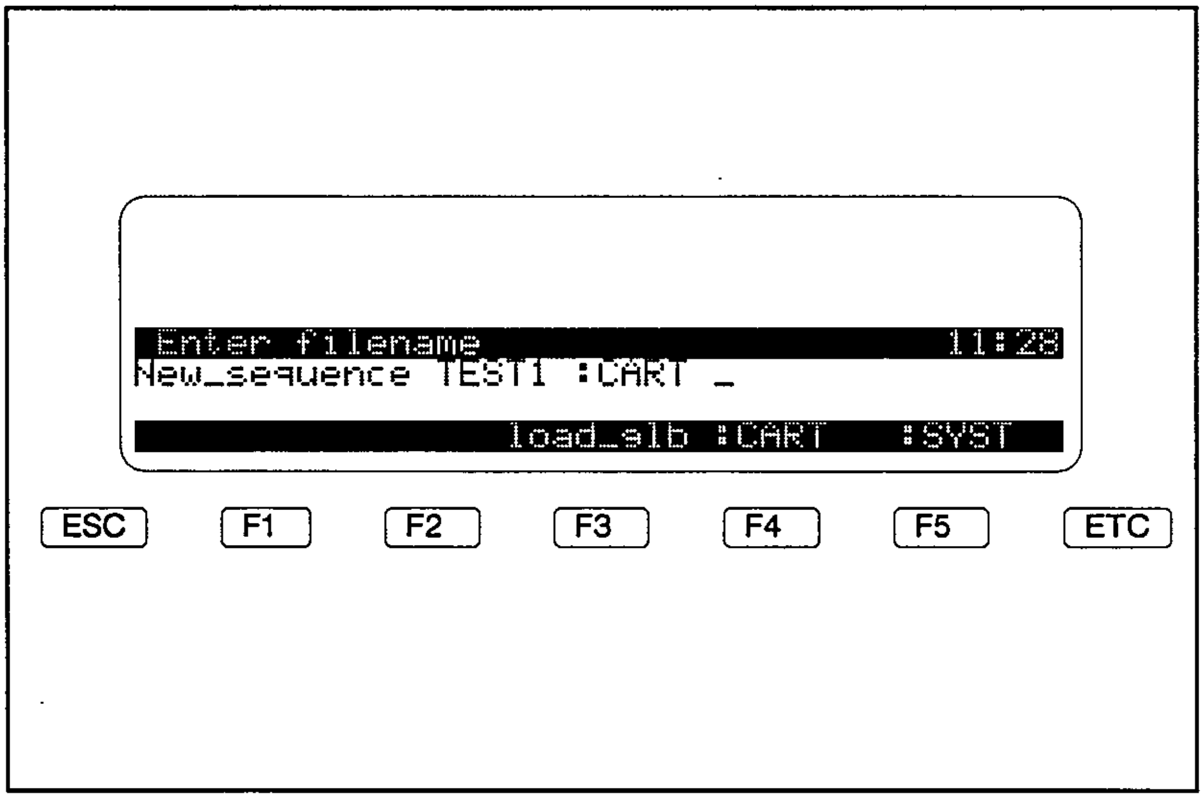


**Flow Diagram of New Sequence Procedure**

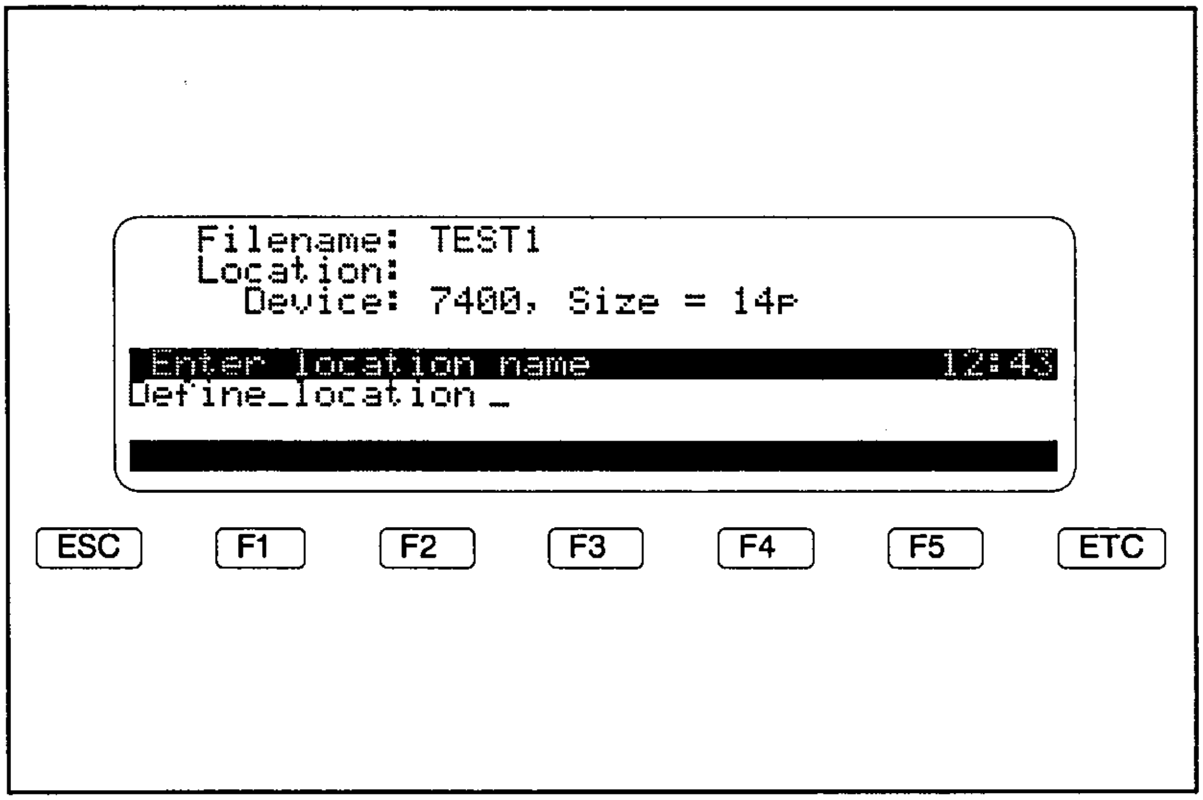
## Programming a New Sequence

The flow diagram on the opposite page shows the steps required to program a new sequence.

Proceed with the keystrokes shown in the flow chart and enter the name `TEST1`, with a destination specified as cartridge (ensure your cartridge is inserted with the write protect switch Off). Your screen should appear as on the following page before you press ENTER.



```
 Enter filename                          11:28
New_sequence TEST1 :CART _

                        load_alb :CART    :SYST
```



```
   Filename: TEST1
   Location:
     Device: 7400, Size = 14P

 Enter location name                     12:43
Define_location _
```

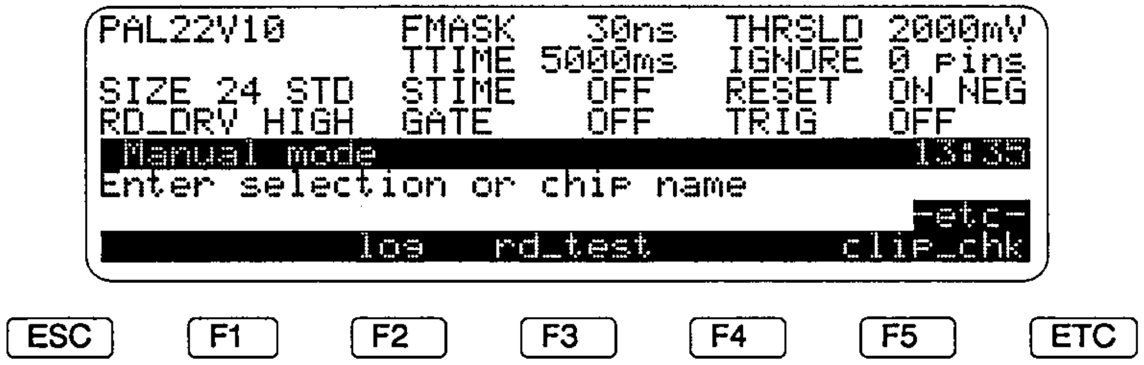
Note that, when naming your sequence, you have the option to select the global parameters from another sequence with the load_gbl key. Recall that globals are parameters that will apply to every device in the sequence. If you are creating several sequences for different parts of a board, this is a convenient way to define standard test times, reset, etc., among the multiple sequences.

By not pressing load_gbl we have chosen the System's standard default global parameters. Adjust our global parameters now to have a test time of 5000 ms. The key sequence is:

manual global t_time
5 0 0 0 ENTER ESC

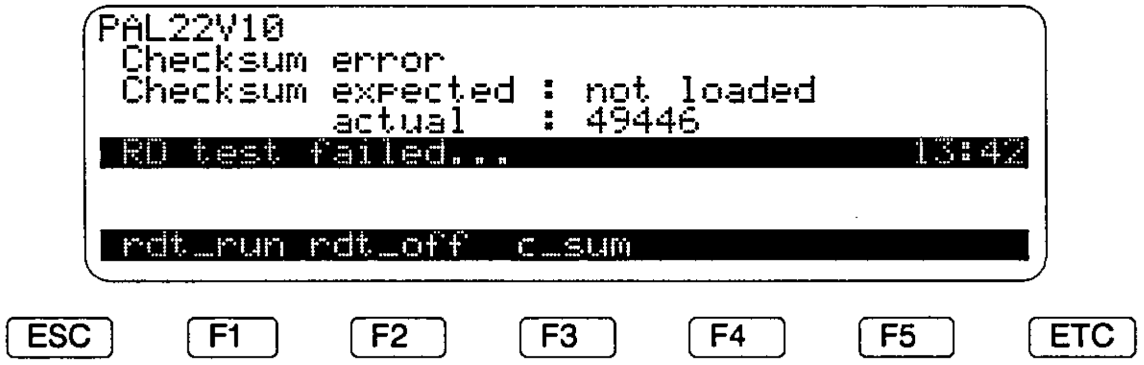
You can see the same keystrokes from the flow chart and we will refer to it in subsequent examples.

Follow the flow chart to define our first IC, verify that it passes on the Demo UUT board and define its location as U14 using defn_loc. The screen appears as on the facing frame while doing this.



```
PAL22V10        FMASK    30ns   THRSLD 2000mV
                TTIME  5000ms   IGNORE 0 pins
SIZE 24 STD     STIME    OFF    RESET  ON NEG
RD_DRV HIGH     GATE     OFF    TRIG   OFF
 Manual mode                              13:35
Enter selection or chip name
                                          -etc-
              los   rd_test           clip_chk
```

**PAL UF3 Parameters**



```
PAL22V10
 Checksum error
 Checksum expected : not loaded
          actual   : 49446
 RD test failed...                        13:42

 rdt_run rdt_off  c_sum
```

**PAL Checksum Generation**

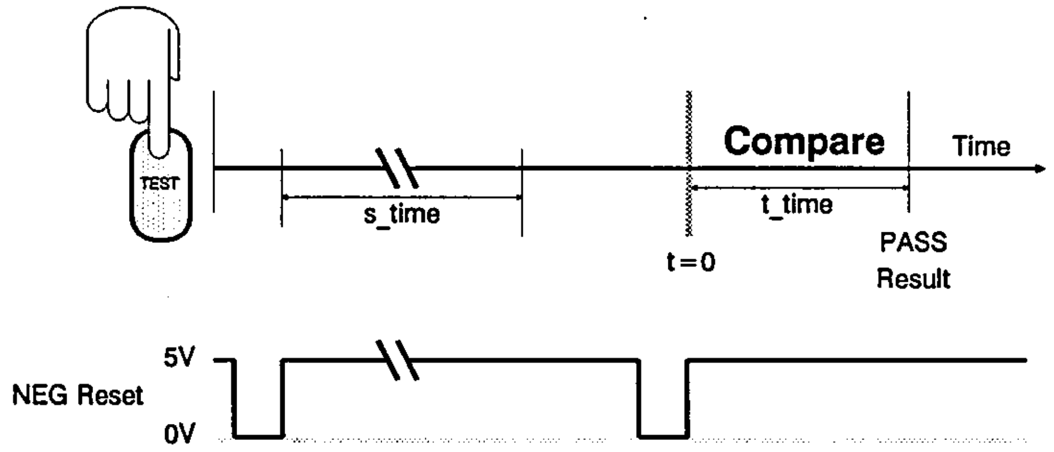
Two key labels appear that are not shown in the flow chart. One is dsp_loc, a short form of "display locations". You may press this at any time to examine the device locations defined so far in your sequence. The locations are ordered left to right, line by line on the screen and may be scrolled with the cursor control keys. ESC returns to the previous screen.

The other key label is save_fil. This key performs a quick storage of your current temporary file (**TEST1.nsq**) to the destination you chose when you named it. Normally this would be the cartridge and, by frequently saving your work-in-progress, you will have a backup copy if you are interrupted by a power loss. Each time you save in this manner, the old file copy is overwritten, so you always have only the most recent copy.

Proceed to add the second device to the sequence, the 8288 located at U2. Note that manual must be pressed for each new device type and defn_loc for each separate physical device.

Now define the custom PAL22V10 (labelled DEMO3) located at UF3 on the Fault Insertion Board by pressing manual and entering the device generic number. The FLUKE 900 only recognizes that it is a 24 pin device without any RD test or special test cycle parameters prestored. Place a Reference Device in the socket and we will generate an RD test from the standard PAL RD test vectors. Select RD test by first pressing ETC, then rd_test.

Press rdt_run and a checksum is generated which doesn't yet match the expected value. Press c_sum and enter the new checksum. Running the RD test should now give a RD test Passed result.



**Synchronous Test Cycle**

## Checksums

Note that all combinatorial and many sequential PALs will produce a unique checksum value. Complex PALs may, however, produce multiple checksums. The only recourse to create a consistent RD Test is to write explicit RD Test Vectors based on a knowledge of the PAL. RD Test generation is covered later in this course.

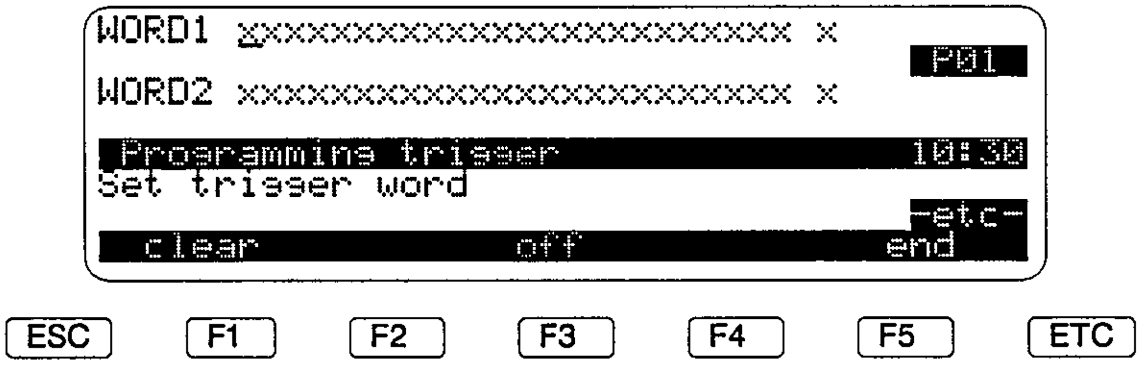
## Synchronization Time

Now we must define the test cycle for this custom device. We don't know its function, but it is a sequential PAL with output pins driven by registers. This will require a synchronization interval (S_TIME) to allow the DUT and RD to cycle through to the same initial state.

Try testing, first with S_TIME off, then with S_TIME set to 5000 ms.

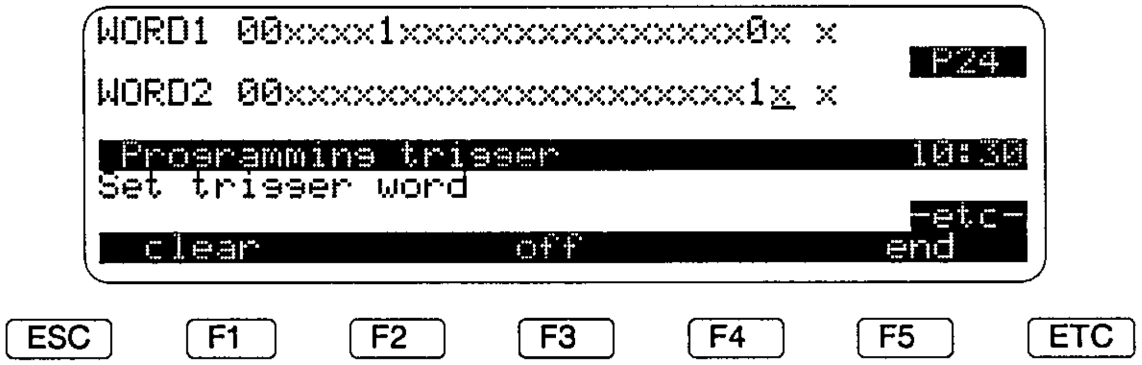
The test cycle for this device is shown on the facing page.

Remember to press defn_loc to save the RD test and test cycle parameters.



```
WORD1 xxxxxxxxxxxxxxxxxxxxxxxx x
                                   P01
WORD2 xxxxxxxxxxxxxxxxxxxxxxxx x

 Programming trigger              10:30
Set trigger word
                                  -etc-
  clear          off            end
```

**Trigger Programming Screen**



```
WORD1 00xxxx1xxxxxxxxxxxxxxx0x x
                                   P24
WORD2 00xxxxxxxxxxxxxxxxxxxx1x x

 Programming trigger              10:30
Set trigger word
                                  -etc-
  clear          off            end
```

## Using Trigger on Complex LSIs

Let's look at testing a complex LSI, the 8259 Interrupt Controller located at U62. This is a programmable device which must receive mode control data from the CPU before it has valid output signals. Selecting this from library memory by its number 8259 will bring a setting for the test cycle that includes an S_Time enabled (with synchronization method 1). Ensure that S_Time is off and we will make sure we have a trigger setting that works instead.

We are going to use trigger to ensure that the 8259 has been initialized. From the Intel Datasheet it states that the condition for an initial command word is

A0 (pin 27) = 0 and D4 (pin 7) = 1,

during a write cycle:

CS (pin 1) = 0, WR (pin 2) = 0.

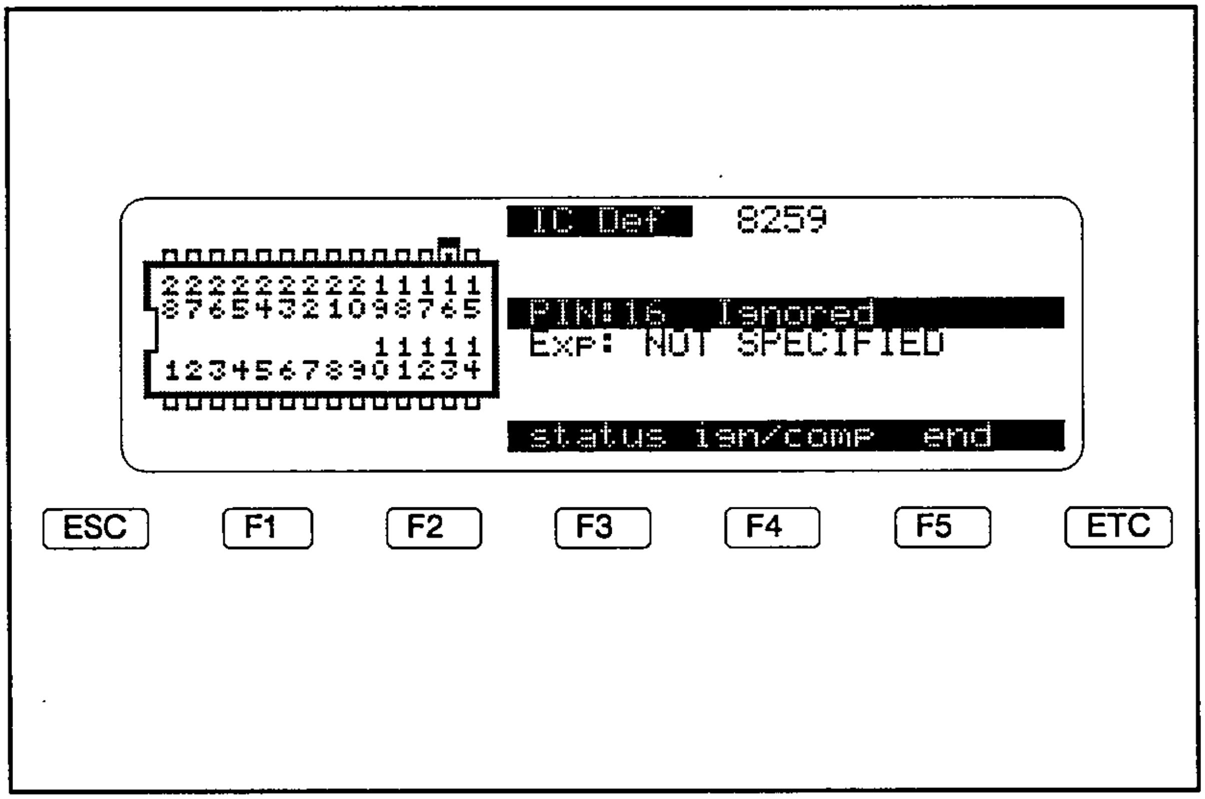
The second command word has

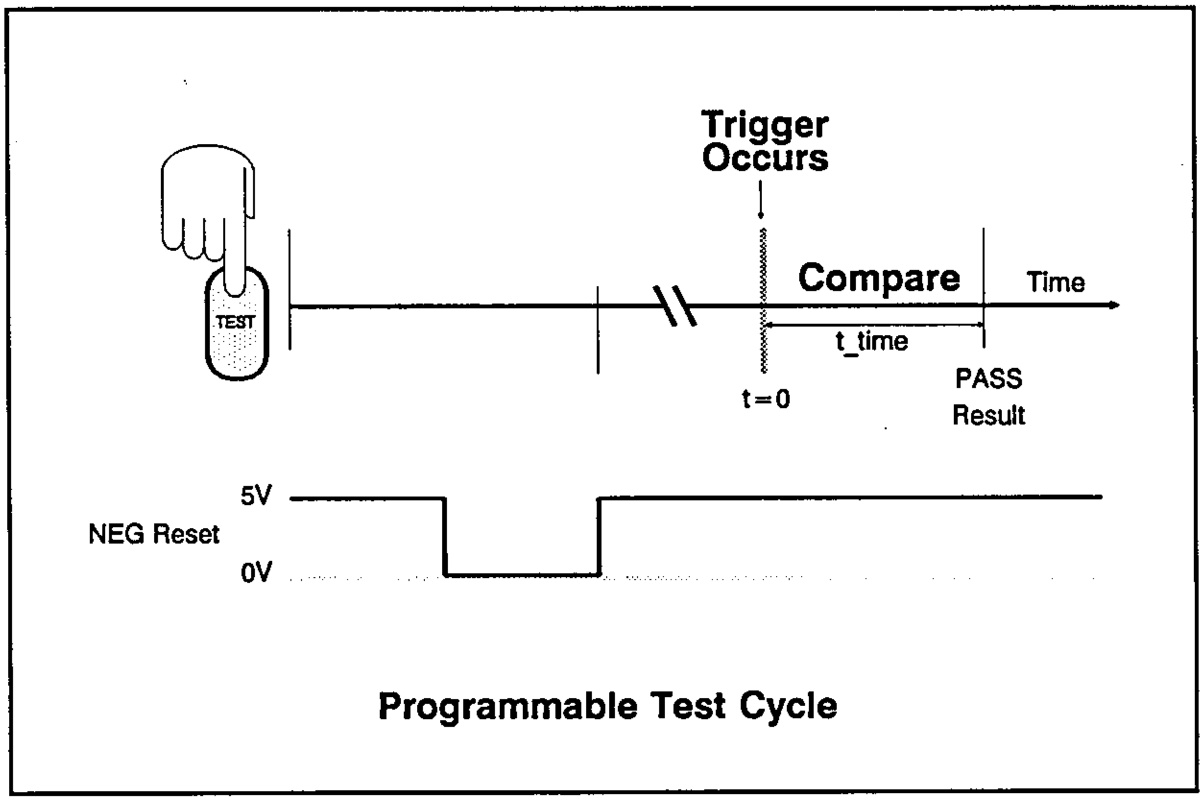
pin 27 = 1, pin 1 = 0 and pin 2 = 0.

We want to trigger the start of comparison on these conditions. Set trigger by pressing trigger in local parameter mode to specify the two 28 bit words.

Using cursor control arrows, move the cursor to the four respective pin number positions and press [0], [1] or [X], as desired.

Now press end and we have enabled a trigger on command word initialization.



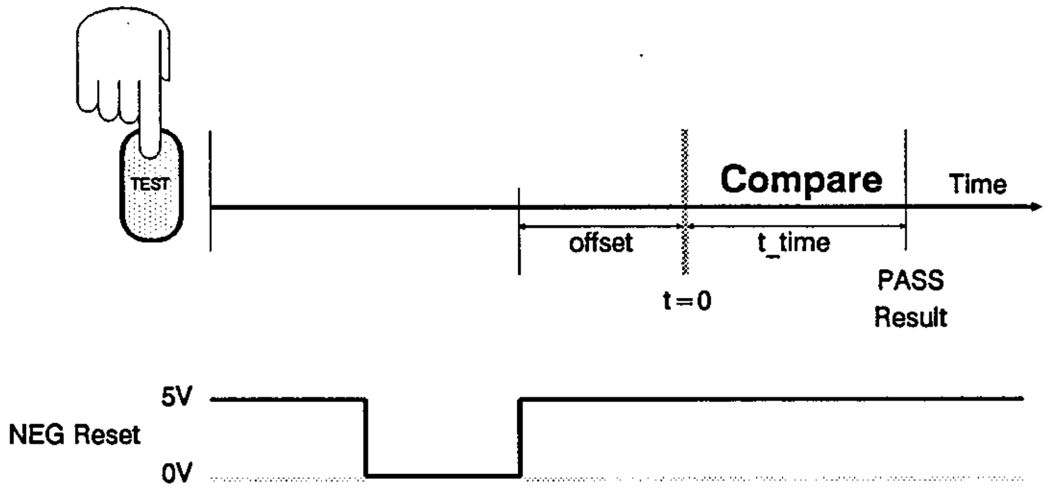


Programmable Test Cycle

Press TEST. An apparent failure occurs on pin 16 because this is an optional output that is not initialized or used on the board. Choose to ignore it with the pin_def key. Observe a graphic depiction of the IC. Use cursor control keys to point to pin 16 and then press the ign_comp key. Note that any pin can be specified to require high or low levels, activity or a frequency value as we discussed previously.

In addition, many LSIs on a bus will require a slightly larger f_mask setting of 40 or 50 ns as is the case here.

Press TEST again. The Test Cycle is shown opposite.



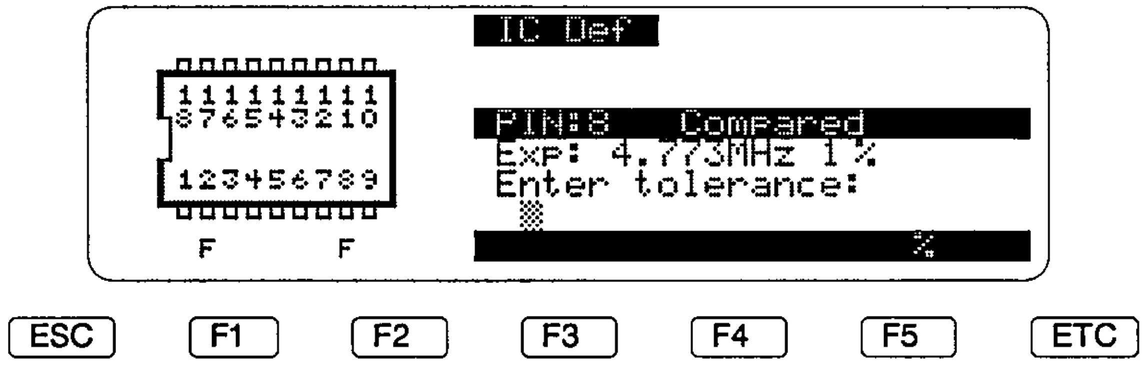
**Reset Offset Parameter**

## Reset Offset Parameter

Another way to initialize programmable devices of which we have little or no information about is to use the reset offset parameter instead of trigger. To illustrate, turn trigger off (it can be re-enabled by entering the trigger level and pressing end).

Now assign a value of -100 ms to the offset parameter found after pressing reset. When testing with this setting, the test cycle looks like the facing frame.

Note that trigger is a preferable initialization technique because a dead board will always result in a Fail test result for the 8259 even if it is good. Confirm this for yourself by pressing Fault Key 2 on the UUT to ground the Reset before testing. Disable offset, re-enable the trigger and try this again to compare results.


IC Def

PIN:8    Compared
Exp: 4.773MHz 1%
Enter tolerance:

**Pin Definition Screen**

## Frequency Check

Finally, we will test the 8284 at U91. Recall that this is a clock generator chip with an analog crystal input. Since we cannot compare this type of DUT to a reference device, we will, instead, define the frequencies of its output pins. The first step is to define the size of the IC. Press:

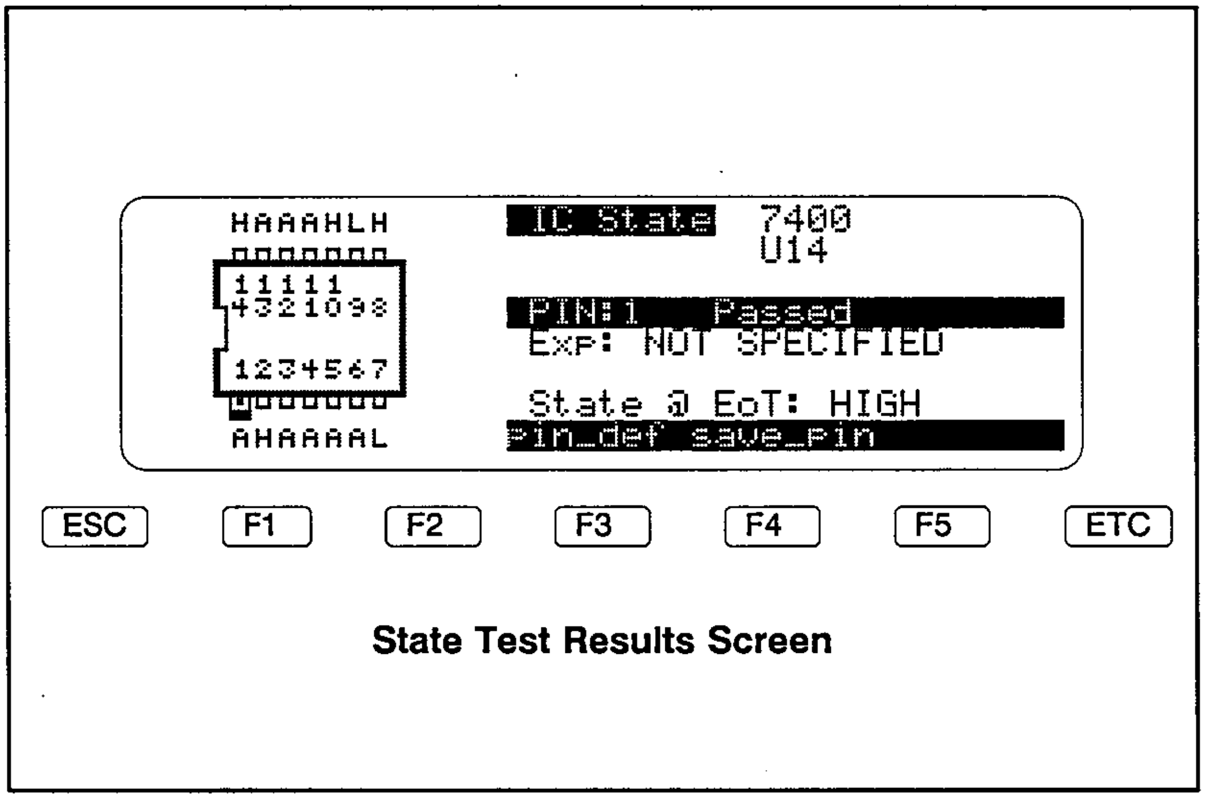
manual local ETC size
1 8 ENTER.

Second, disable comparison on all pins, disable rd_test and set the frequency check as follows:

pin 2 - 2.386 MHz, 1% tolerance
pin 8 - 4.773 MHz, 1% tolerance

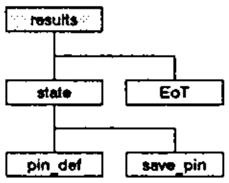
Note that, to do this, you press pin_def and use the cursor to point to each pin. Repeatedly press status until you are prompted to enter a frequency value. After specifying all the frequency checks, define the device location as U91.

Up to this point, you have created a temporary sequence file in the tester's working memory. You may also have stored a backup copy in the cartridge if you pressed the save_fil key. The file contains our five devices in the order they were entered and with the necessary parameter adjustments to make them pass on a good working board. Press end to terminate this initial sequence creation and notice the automatic generation of sequence source and compiled files.



```
IC State   7400
           U14

PIN:1    Passed
Exp: NOT SPECIFIED

State @ EoT: HIGH
pin_def save_pin
```

State Test Results Screen



Results Menu Tree

## Summary

We have created a small five-device sequence named TEST1 on the cartridge which includes a variety of simple and complex devices. We verified on a good board, by trial and error, that the parameters were set so each device passed.

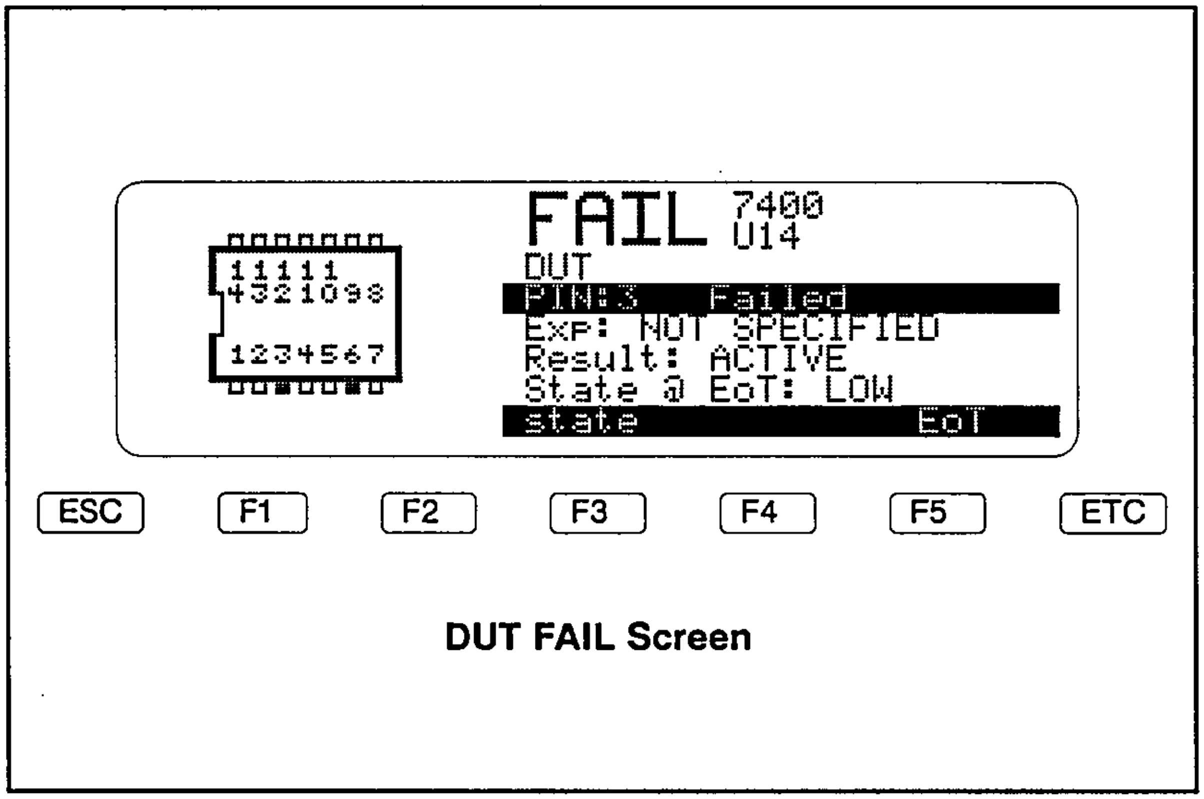
Some apparent failures were corrected with parameter adjustments. Now we could test these same devices on a bad board for actual faults.

## Test Results

1. "RD test failed..." appears on the status line when no RD, the wrong RD or a faulty RD is inserted in the socket. Failing to lower the contact clamping lever will also produce this message. (Severely shorted RDs produce the message "Excessive RD current.".
2. "No clip inserted" and "Clip size incorrect" are operator alerts that cannot be overridden by turning Clip Check off. "Vcc-GND check failed" and "No signals from clip"indicate a wrongly oriented clip or lack of power to the DUT.
3. "Test aborted" appears when TEST is pressed again to stop the test cycle before comparison begins.
4. "Pass..." indicates DUT within Performance Envelope

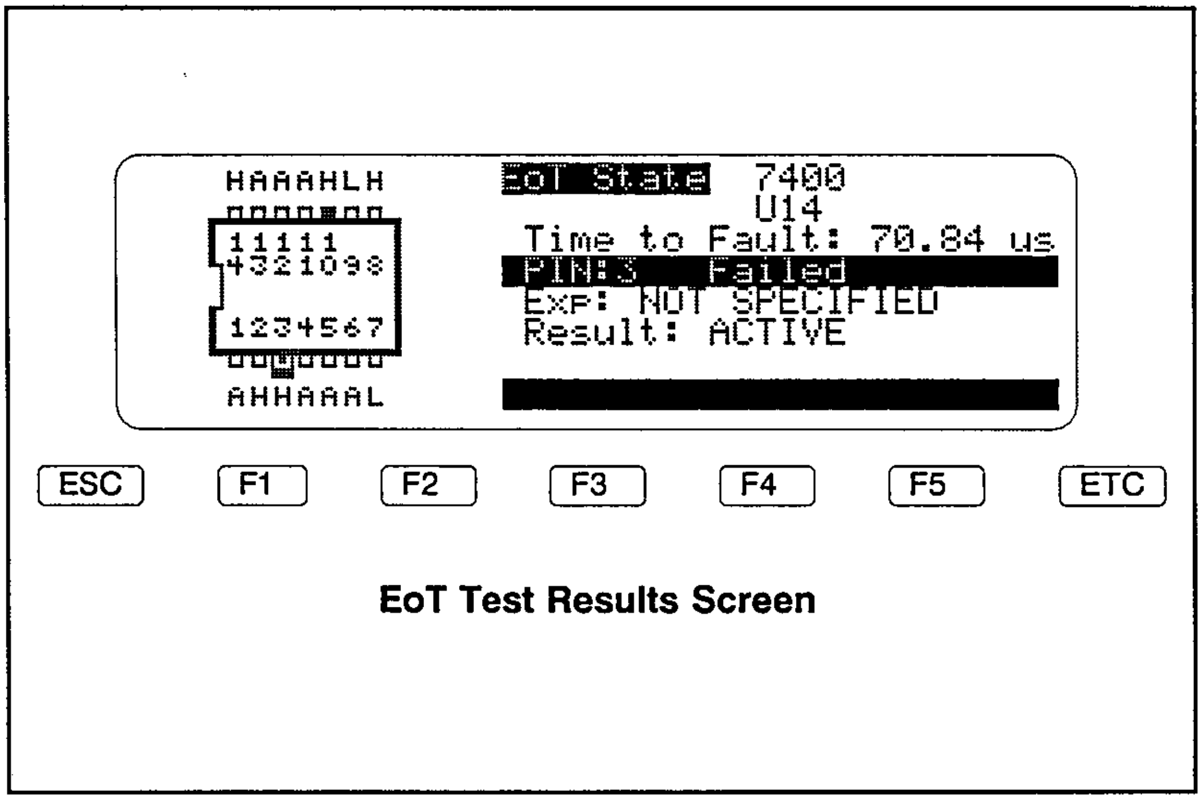
   "Pass..." on the status line indicates that the DUT did not exhibit a fault and any specified signal conditions were satisfied. Press results to show a detailed view.

   The cursor control arrow keys are used to point to a pin of interest as highlighted in the status line. Expected preprogrammed conditions are listed as "Exp:", actual results are listed as



```
FAIL 7400
     U14
DUT
PIN:3    Failed
Exp: NOT SPECIFIED
Result: ACTIVE
State @ EoT: LOW
state              EoT
```

**DUT FAIL Screen**



```
EoT State   7400
            U14
Time to Fault: 70.84 us
PIN:3    Failed
Exp: NOT SPECIFIED
Result: ACTIVE
```

**EoT Test Results Screen**

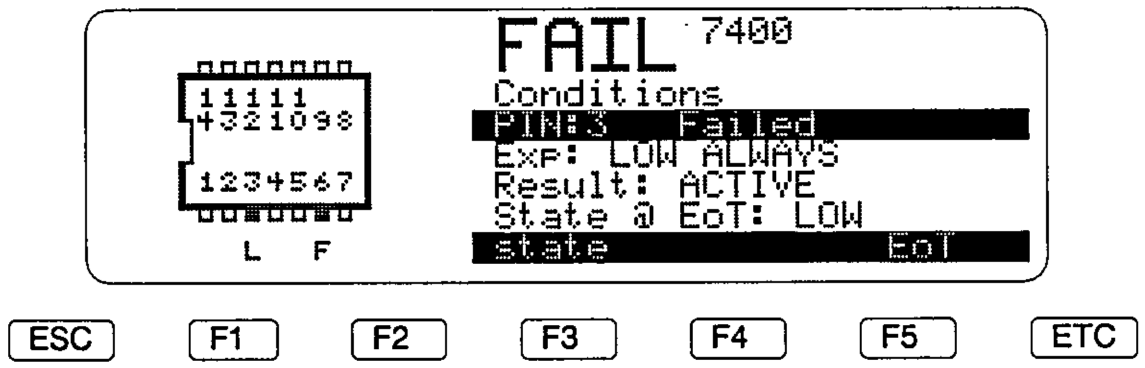
"Result:", and the state of the pin at the end of test is listed as "State @ EoT:". A complete snapshot of all pin signal conditions during the test is viewed by pressing state.

A complete snapshot of all pin states at the end of test can be viewed by pressing EoT. Note that from the state submode you can change expected signal conditions using pin_def, the same parameter accessible from Local Mode. Alternatively, it is possible to assign actually observed H, L or A conditions to each pin using save_pin.

5. "FAIL..." indicates DUT exceeded Performance Envelope

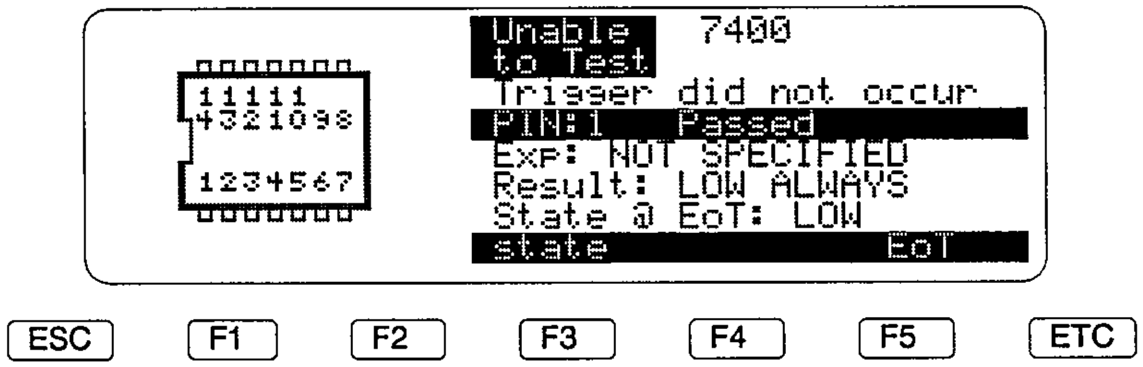
The graphic results screen then appears along with "FAIL" in large letters when the DUT failed comparison as defined by the PE parameters. The failed pins are shown in reverse highlight and are flashing on the signal monitor.

As with the PASS screen, snapshots may be viewed of all pin activity and state at end of test. Also shown is the "Time to Fault:" as measured from the start of comparison testing, with an accuracy of 40 ns or within the last digit shown (for ms, s). Note that most processor-based boards respond slightly differently each time to a reset pulse. Therefore, time-to-fault readings will only be consistent if an external trigger is connected to the processor Reset Out or Memory Read signal (i.e. first op code fetch).



```
FAIL 7400
Conditions
PIN:3    Failed
Exp: LOW ALWAYS
Result: ACTIVE
State @ EoT: LOW
state          EoT
```

Signal Condition Fail Screen



```
Unable    7400
to Test
Trigger did not occur
PIN:1    Passed
Exp: NOT SPECIFIED
Result: LOW ALWAYS
State @ EoT: LOW
state          EoT
```

Unable to Test Screen

6. Fail Conditions - Operator Interpretation

The graphic results screen appears along with "Fail Conditions" when the DUT passed comparison but specified signal conditions were missing. These are indicated by letters (H,L,A,F) adjacent to the pins concerned.

Depending on whether the pins are input or output and how they were programmed, it may indicate a faulty DUT or be a backtracing indicator. Snapshots may be viewed of pin activity with state and EoT.

7. Unable to Test - Operator Intervention

There are four conditions that will cause this message without ever doing a comparison test of the DUT.

a. "Failed to synchronize"

Board activity is not sufficient to initialize DUT. Look elsewhere for the source of the problem.

b. "Trigger did not occur", or "Trig W1 did not occur", or "Trig W2 did not occur".

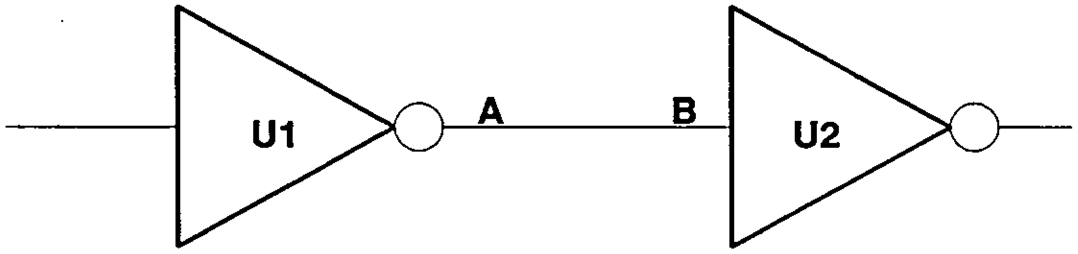
Specified initialization trigger is not present.

c. "Gate did not occur"

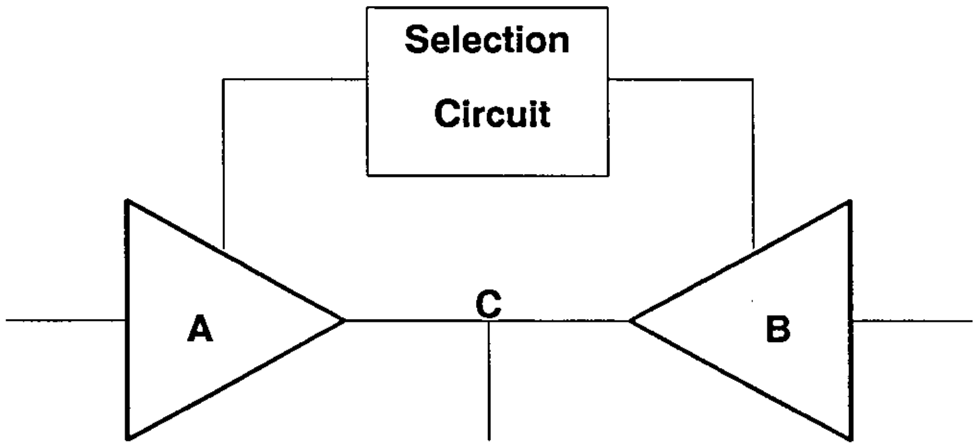
Specified valid gate did not occur.

d. "Synchronization timeout"

s_time setting is too small for the specified sync vectors (Refer to Section 5)



Stuck Output (A)/Stuck Input (B)



Bus Faults

## Interpreting Results

When verifying that a device is really failing and not just slightly exceeding a very tight f_mask setting, care should be taken not to mask out all activity with a large f_mask. For example, with f_mask = 200 ns, any pulse (good or faulty) smaller than this will be ignored on an output pin.

## Stuck Signals

Refer to the illustration on the opposite page.

Both of these fault modes appear as "U1 FAIL, U2 PASS". Often a shorts finder ohmmeter will locate which is the offending current- sinking pin. In practice, it is found that U1 is faulty most of the time, so this should be the first choice for rework replacement.

## Bus Faults

A failing device A will cause a fault on common node C and make device B also appear faulty. The only way to determine which one is holding down the bus is to use a shorts finder or high resolution DVM.

Note that if selection circuit D is failing and causing A and B to appear together on node C, the Time-to-Fault will be lowest for D since it fails first and improperly selects A and B.

## Open Traces/Missing Signals

Active signals may be monitored using the activity check. It is recommended this be done for signals originating off the board and clock and enabling signals to each device. Key signals that are missing point to a problem with a prior sourcing device or the interconnection.

## Answers to Questions on Page 3 - 1

1. ROM is treated as a "combinatorial device" since the inputs (address lines) cause an immediate repeatable state on the outputs (data lines).
2. Counter is a "synchronous device" since its internal states are observable from the output pins.
3. Shift Register is a "synchronous device" if it has a clear or parallel load line since they will allow us to infer the internal states. In addition, for an eight-bit shift register, we can infer its internal states after observing eight Serial In pulses.
4. Interrupt Controller is a "programmable device" since it has many hidden registers and requires a command word to be written to it before its outputs are valid.

# Section 4 Applications

*In this section we will cover some basic application examples and considerations. The Demo listings referred to in some examples are found in Appendix A.*

## Board Level Considerations

## Board Level

The general rule for making a troubleshooting sequence is to automate the common sense techniques used in manual troubleshooting. This usually involves testing for critical signals and devices first and following the flow of signals through a board. On a microprocessor-based board, critical signals include CLK, RESET, HLT and bus control signals. One approach to such a board is to check first for the presence of these signals with condition tests (e.g. freq, H, L, Active states) and then proceed with DRC testing. Alternatively, the devices responsible for clocking and control signals can be given a first priority in the sequence. On a multiprocessor board, the area of the master CPU is normally tested first.

Bus structured board testing proceeds from the bus outwards. Often, by merely testing 20-pin devices first, this is achieved. Twenty pins is the size of drivers, transceivers and latches. For nonprocessor cards, testing proceeds from the card edge through the board. Note that any sequence is only a nominal troubleshooting order that the user can override to shorten the process. He may want to focus initially on failure-prone areas or verify diagnostic messages from a functional test.

The diagnostic stimulus should be chosen to provide maximum activity. Ideally it can be repeatedly induced by resetting the board, or in the case of a multicard system, resetting the main board. Very occasionally, a reset will not effect a restart or multiple reset pulses cannot be handled by a board. Testing can still proceed under partial reset conditions except for certain programmable devices or RAM that may not be initialized. In some cases, it may be better to use the External Trigger instead of Reset to start DRC testing from a signal indicating start of diagnostics.

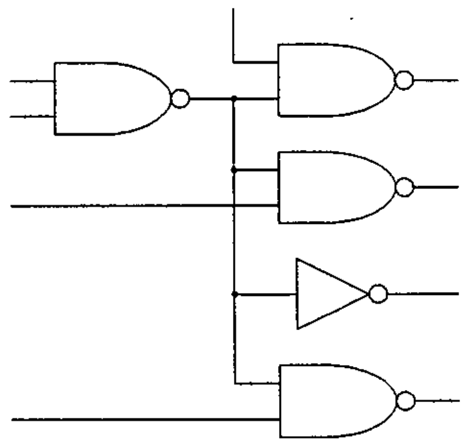
## Device Level Considerations

- reclip and retest to confirm the failure
- press state to see if the failing pin is active or stuck
- press EoT to see if elapsed time to fault is consistent
- increase FMASK to check the hardness of the fault
- for a bus device, test a few others on the bus before concluding which one is bad.

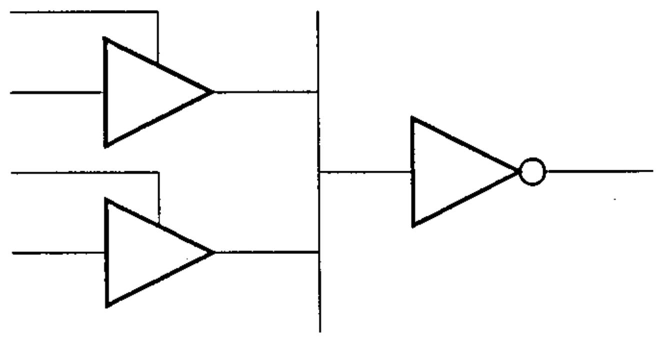
## Device Level

When a failure is detected, there are a number of things that may be done to give more insight about the device:

- reclip and retest to confirm the failure
- press state to see if the failing pin is active or stuck
- press EoT to see if elapsed time to fault is consistent
- increase FMASK to check the hardness of the fault
- for a bus device, test a few others on the bus before concluding which one is bad.



Loading Adjustments



Floating inputs

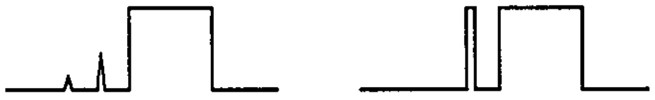
## Loading Adjustments

When a device drives a high fanout or capacitive load, the signal rise time will be extended. Increasing the FMASK setting will compensate for this. Loading is normally within 50 ns of the response time for a device. If a greater FMASK is necessary, the user should investigate other causes.

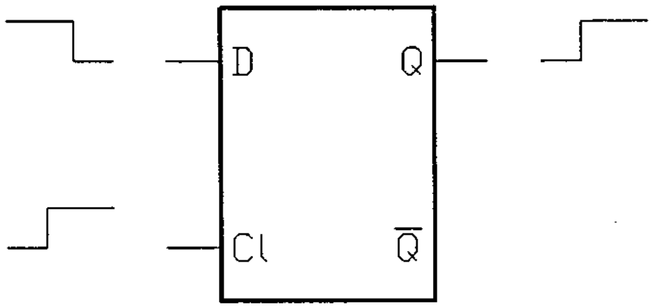
## Floating Inputs

Inputs floating or connected to a tristate source will cause an apparent fault when the signals are floating at an indeterminate level between 1 and 0. The DUT and Interface Buffer may see logic 1 at different levels. The best solution is to enable an external gate on a control signal that is active when device inputs are valid.

Note: Adjustments to bias that may appear to solve the problem are probably not universal. Problems reappear on other boards because different devices have different characteristic thresholds.



Noise



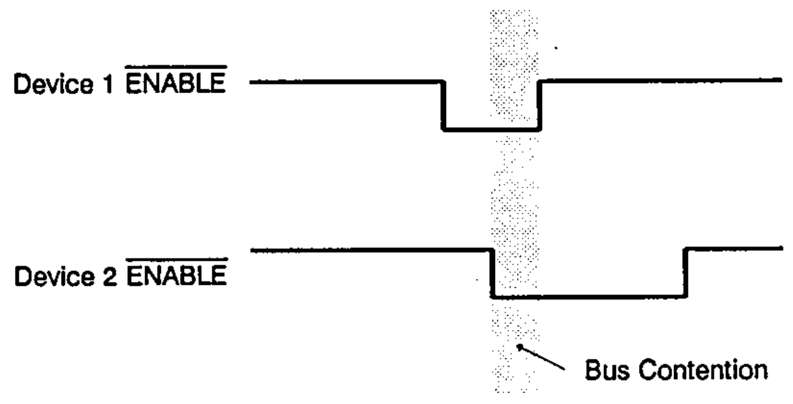
Race

## Noise

To ignore noise and ringing on the inputs of a device, Threshold may have to be increased. Moving GND lead of IB close to DUT may also assist. Power supply spikes from the AC line input are a particularly difficult form of noise to ignore.

## Race

When the inputs to a sequential device (e.g. clock and data) are within a few nanoseconds, the slight skew of the System may cause the RD to clock in wrong data. If this is the cause of problems, you will generally observe failures on all output pins (e.g. Q, $\overline{Q}$ of a flipflop). Try to move Threshold up and down to separate signal edges.


Device 1 $\overline{\text{ENABLE}}$
Device 2 $\overline{\text{ENABLE}}$
Bus Contention

**Bus Contention**

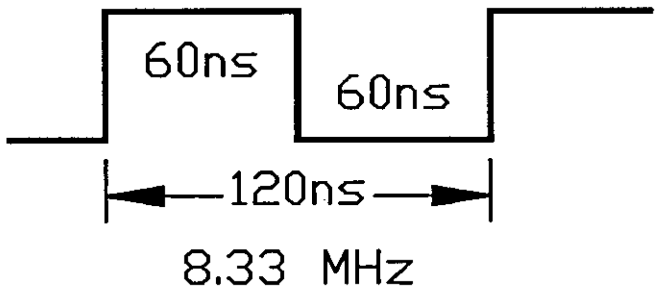
## Asynchronous Inputs

An external interrupt or any asynchronous signal acquired by latching with a synchronous board clock may exhibit a race problem if the edges are very close. The latching device in this case is not testable with DRC.

## Bus Contention

This occurs when more than one device is enabled on a tristate bus at the same time. Bus drivers such as 74244, 74245 may exhibit this problem which is illustrated in the timing diagram on the opposite frame.

Output data for device 2 is indeterminate before the last part of its output enable signal. The solution is to give comparison testing from a signal which indicates valid data. Most microprocessor designs provide this signal on the bus controller chip.



High-Speed Signals

## High Speed Signals

Recall that FMASK will ignore faults that are less than its duration. This implies that an FMASK setting of 60 ns cannot test signals faster than about 8 MHz at 50 percent duty cycle). To test 20 MHz signals, FMASK should be set to 20 ns. The general rule is that a signal's pulse width must be larger than FMASK.

## Initializing Programmable Devices

Programmable devices are indeterminate and will fail if they are compared before on-board activity initializes them. Synchronization data built into the library normally handles this. For new devices or RAM, it is left to the user to mask out indeterminate states using Trigger. Some systems read from RAM that is uninitialized and the contents of the RD will not be the same. The solution is to trigger the start of comparison from a point after the read-before-write occurs. For example, set a trigger on the writing to the highest memory address since this could be the last step of a system's initialization routine.

## EoT

## FMASK

## APPLICATION ADVICE

The testing features are listed alphabetically below and peculiarities of their application are described.

### EoT

The end of test screen shows the states of all pins at the moment of failure. This is useful to determine the failing RAM address when solving read-before-write initialization difficulties of sequence creation.

The time-to-fault reading will only be consistent if an External Trigger is employed on initial microprocessor activity. This is because CPU devices act on a Reset pulse with an inconsistent response time.

Time-to-fault readings in the nanosecond range are useful for first fail conclusions. Readings in μs and ms ranges are too coarse to be useful.

### FMASK

A lower setting provides a closer comparison and therefore higher quality of test.

The setting should be high enough to pass the typical range of good boards.

An abnormally high setting (i.e. 80 ns +) should be investigated to see if a Gate is more appropriate.

## FREQUENCY GATE

FREQUENCY

Use to monitor ungated clock signals.

Output pins are evaluated with DRC, so condition testing of all types (H, L, A, F) is normally only done on input pins.

A Reset is issued with each pin frequency check.

Duty cycle information is only available through freq and it is affected by the Threshold setting for narrow pulses.

GATE

Use as an alternative to large FMASK settings.

Bus control devices usually provide suitable signals for gating bus contention.

The frequency and duty cycle of the gate setting may be observed to monitor complex timing relationships (e.g. a gate set on RAS and CAS of a RAM).

**PIN_DEF**

**RESET**

## PIN_DEF

Activity check will reflect pin status even while a gate is not true. (i.e. an unselected bus device will still show active pins).
Use sparingly since a bad board will have many missing signals and be confusing.
Use mostly on input pins since outputs are checked with DRC. Exceptions include one shots, line drivers and any other device not testable with DRC.
H and L status checks are performed with 10K pull-up/pull-down resistors in the Interface Buffer. Pull-ups are present by default to bring unused input pins to a definite state. For an L status check, the pull-up is present; for an H status check, a pull-down resistor is switched in.

## RESET

Occasionally, a simple Reset fails to duplicate power down reset. Testing can still proceed, except for the few devices with uninitialized registers.
Very rarely, multiple Reset pulses cause such problems as blowing a fuse. The extra pulses used for synchronization may be disabled with the command `SYNC_RESET_OFF` in the .loc file of the Sequence. Alternatively, Trigger may be used instead of Reset.
Use of a negative Offset to wait for initialization on a device can be useful in trying to understand the operation of a good board. On a bad board, however, it is susceptible to false failures. Trigger is a more effective initialization parameter in this case.
Use of a positive Offset is recommended for those devices that generate a board's reset so that DRC testing is done during the Reset pulse.

## STATE THRESHOLD

STATE

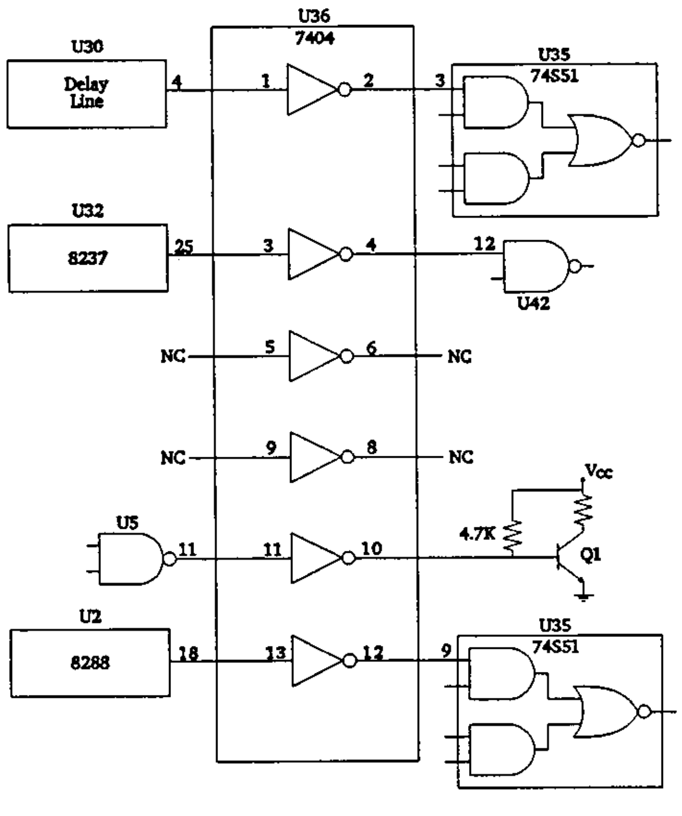
The state screen shows the pin activity during test and can be a good qualitative indicator of how active a device was.

THRESHOLD

It is recommended to keep Threshold constant and vary FMASK and Gate to make devices pass during Sequence Creation.
Threshold sensitive circuits are subject to board-to-board variations.

## Hands-On Applications Exercises

**DUT Loading**



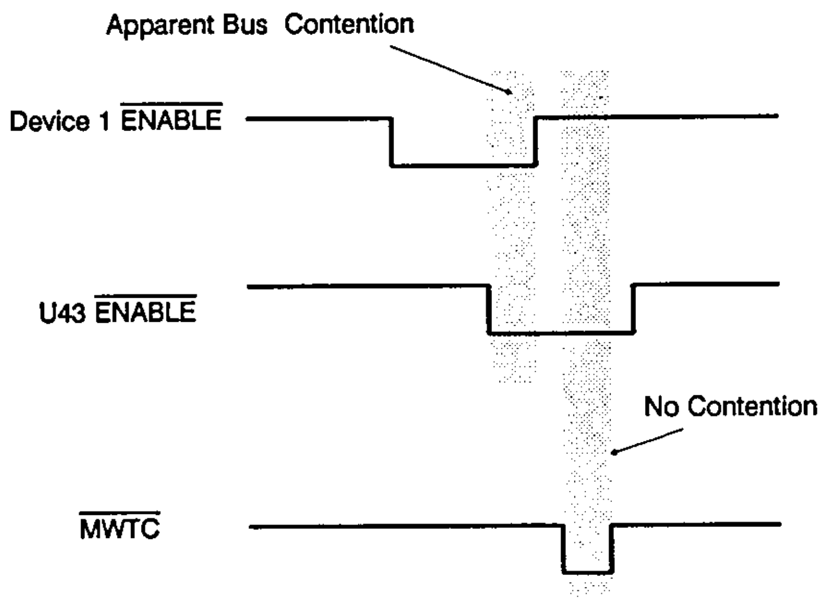
## Hands-On Applications Exercises

In the following exercises we will test a few devices in Manual Mode on the known-good Demo UUT, making the parameter adjustments necessary to make them pass. Then we will insert some faults and interpret the results.

**Loading Example**

1. Test the 7404 located at U36. Raise FMASK to try and make pin 10 pass. Try lowering THRESHOLD. Note than when we can pass pin 10, other pins fail. We need to test part of the device at one setting and part at another.
2. Now ignore pin 10 and find the correct Performance Envelope. This becomes the Performance Envelope for part of our test.
3. Ignore pins 2, 4, and 12 while comparing pin 10 and find the correct Performance Envelope for this part of the test.

   Testing at two different settings results in a high quality of test for the entire device.


Apparent Bus Contention
Device 1 $\overline{\text{ENABLE}}$
U43 $\overline{\text{ENABLE}}$
No Contention
$\overline{\text{MWTC}}$

**$\overline{\text{MWTC}}$ as a Gate**

## Gating Example

1. Test the 74244 bus driver located at U37. Find the FMASK setting that allows it to pass. Is this a loading or slight contention problem that can be handled by FMASK without sacrificing the quality of test?

2. Test the 74244 located at U43. Is the FMASK setting required for a PASS result excessively high? This seems to be a contention problem suitable for solving with the Gate parameter.

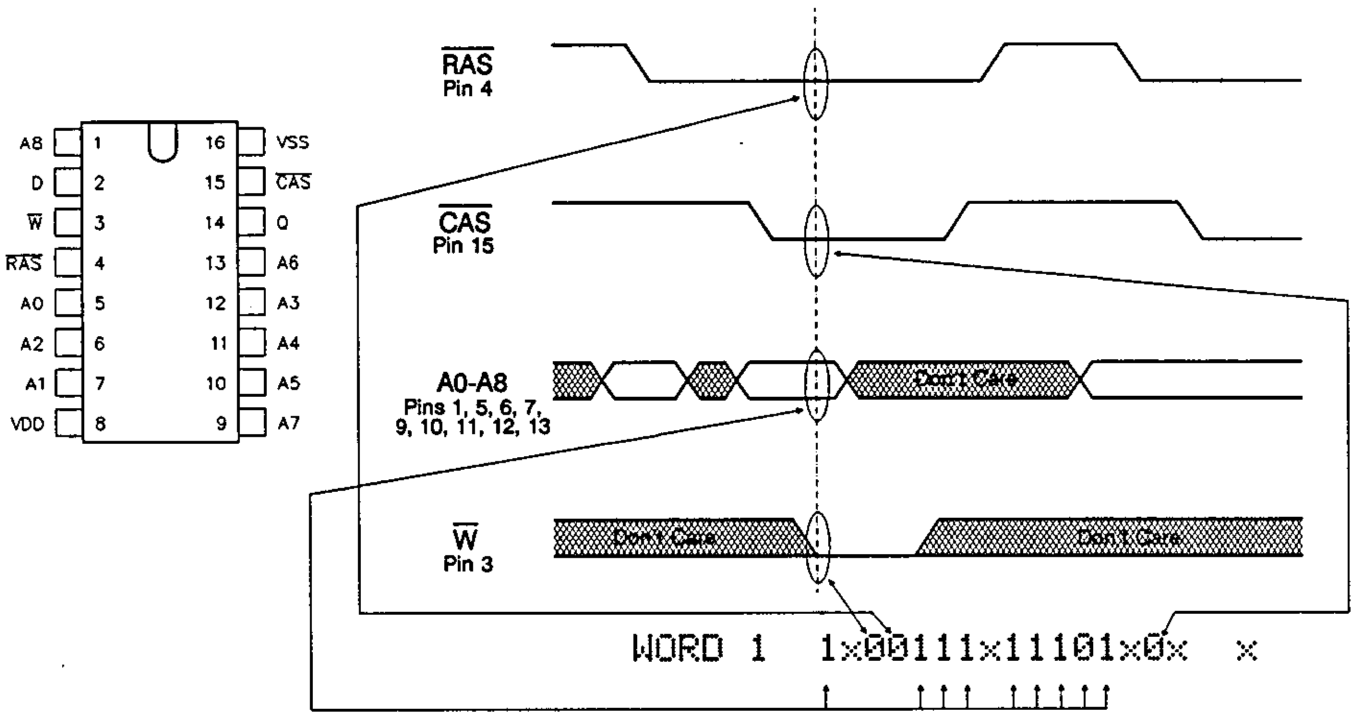
3. Select a signal on the 8288 Bus Controller for the External Gate lead. One method would be to use trial and error on the various outputs of the 8288. Note that if you do this, you should verify not only that U43 passes with the selected gate, but also that a fault inserted on U43 is captured.

   Observe the schematic of the circuit to provide a clue for the proper gating signal. U43 buffers the address lines from the DMA Controller and is enabled by the $\overline{\text{DMAEN}}$ line. Therefore, possible gating signals are those controlling memory, such as:

   | | |
   |---|---|
   | $\overline{\text{MWTC}}$ - | pin 9 of 8288 |
   | $\overline{\text{MRDC}}$ - | pin 11 of 8288 |

4. Try testing U43 with an External Gate set to logic 0 and the EXT lead clipped onto the Bus Controller. What FMASK setting will give a PASS result now?

5. Note that on the Demo UUT, several control signals are ORed together to form a general purpose gating signal for various busses. It has a true logic convention, so the External Gate setting in the DEMO.LOC file is logic 1 or H.



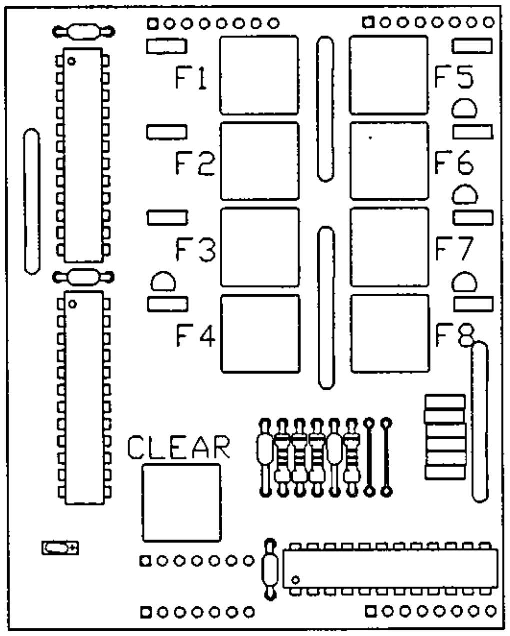
**41256 Write Cycle Timing**

**Setting Trigger Word 1**

## RAM Initialization Example

1. Try testing a 41256 in the bank of RAMs. Increase FMASK slightly to see if we have a loading or access time problem. Can we draw any conclusions from the failing address shown on the pins of the EoT screen? What is the column address (A0 through A8) captured at the fail point?
2. Try testing with a Reset Offset of -100 ms. We appear to have a "read before write" initialization problem that is solved by using the Offset parameter. Note that some RDs may characteristically power up to have the same contents as the DUT at the offending address. In this case, a PASS result will be obtained.
3. Put the Reset parameters back to their default settings and enable a Trigger to bypass the "read-before-write" condition we observed in step one: Set word 1 to detect a write cycle to this location. With this trigger setting the RAMs should PASS.

   Other guidelines that may help for other boards are to set triggers on the top two or bottom two RAM addresses since these locations are commonly the first ones written for a RAM selftest.
4. Now that we have the correct Trigger setting for testing the RAMs, let's try something else — a measurement of the access time allowed by the circuit driving the RAMs. It should exceed the characteristic access time labelled on the 41256s. Disable the Trigger setting and set the Gate to: RAS = 0, CAS = 0, W = 1. Now press freq and choose gate detail. The time high value is the time that the Gate is true which should be an interval greater than the RAM access time.



**Demo UUT Fault Insertion Board**

## Fluke 900 Demo UUT and Fault Insertion Board

The FLUKE 900 Demo UUT is based on an XT-Turbo Clone motherboard, based on the 8088 Microproccessor. The board can operate at 2 different clock speeds, 4.773 MHz and 10.00 MHz, and has on-board memory of 256K DRAM.

The Fault Insertion Board (FIB) is mounted on the lower right hand corner of the UUT, and provides status LEDs that indicate the boards operation, and 8 different Fault conditions.

The Fault buttons are numbered [F1] thru [F8], on the silkscreen adjacent to the individual blue fault buttons.

Each button inserts a different type of Fault on the UUT. Please note that under most fault conditions the UUT Crashes, however it can be Restarted by clearing the Fault via the red CLEAR button.

- FLT5 RAM appears to fail. The fault is actually inserted in the decoding circuit at U25, which also fails. This illustrates the necessity of testing address decoding and bus driver devices in a Sequence before the RAM itself to avoid ambiguous results.
- FLT6 `Unable to test - Waiting for trigger` result. The UUT is crashed before RAM is even exercised.
- FLT7 This fault causes bad data to go into memory. Every time the board is reset it is active for a time before locking up. This exemplifies how DRC will pinpoint the source of the bad data and not capture the fault as it propogates through memory.

## Fault Insertion Exercise

Enable the Trigger setting for RAMs that permits them to pass on a good board. Retest a 41256 while FLT5, FLT6, and FLT7 are inserted. A complete description of these faults appear at the end of this section.

The results of your test may be interpreted as follows:

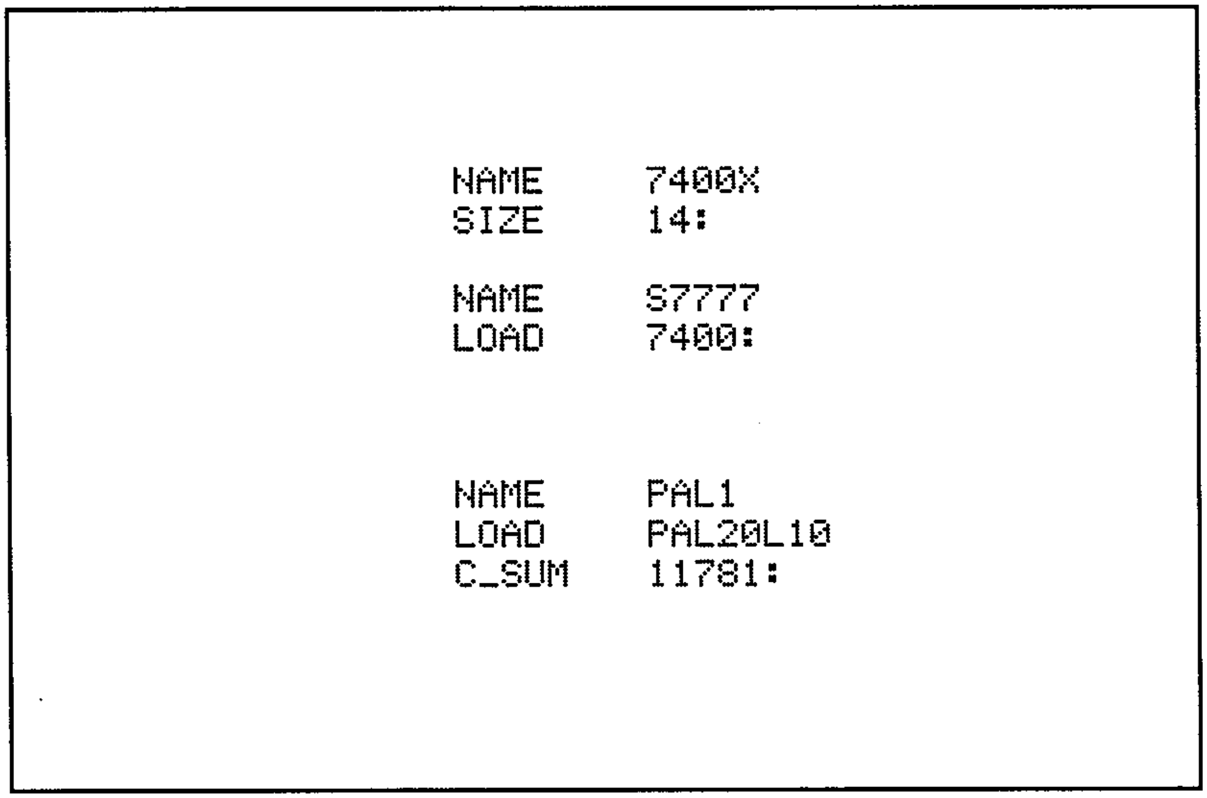
FLT5 RAM appears to fail. The fault is actually inserted in the decoding circuit at U25, which also fails. This illustrates the necessity of testing address decoding and bus driver devices in a Sequence before the RAM itself to avoid ambiguous results.

FLT6 `Unable to test - Waiting for trigger` result. The UUT is crashed before RAM is even exercised.

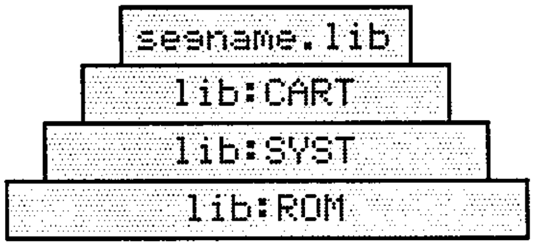
FLT7 This fault causes bad data to go into memory, yet the RAM itself passes. Every time the board is reset it is active for a time before locking up. This exemplifies how DRC will pinpoint the source of the bad data and not capture the fault as it propogates through memory.

# Section 5
# Device Libraries

*The device library is a database containing information for in and out of circuit testing. It is comprised of a ROM-resident file containing a large number of standard devices and expandable user files resident in system RAM or cartridge. The ROM library contains test patterns for verifying the functional integrity of reference devices in the ZIF socket. The revision level of the library is indicated on the main power up screen.*



```
NAME      7400X
SIZE      14:

NAME      S7777
LOAD      7400:


NAME      PAL1
LOAD      PAL20L10
C_SUM     11781:
```



## Library Format

The first command in a command group is NAME followed by the device number designator. The next command either specifies the device size or references an equivalent device that is already contained in the library.

Note that the first example shown defines a 14-pin device known as 7400X, unrelated to a 7400. The second example defines a device known as S7777 to be equivalent to a 7400.

Test parameters may also be added to the command group to account for characteristics such as drive capability (RD_DRV), parameters and RD Test specifications. Note that parameters specified in a .LIB file should be general for all applications while those in a .LOC file are unique to a particular circuit.

RD test information is listed to verify the device in the socket. For PROMs and PALs, a checksum (C_SUM) is calculated and verified. Approximately one hundred cells throughout the PROM address range are combined for its checksum (enough to characterize, but not fully evaluate it). A standard pattern is applied to a PAL. For all combinatorial and many sequential PALs, this generates a single characteristic checksum. For more complex PALs, a unique checksum is not obtainable by this method. The only recourse would be to write specific vectors based on a knowledge of the PAL, as described later. An example of a custom PAL library entry with C_SUM is shown on the left.

As a final general word on libraries, when a user enters a device type number, the system first checks cartridge-resident .LIB files, then system-resident .LIB files, before checking the ROM-resident library. Thus, device numbers duplicated on both ROM and cartridge will be chosen from cartridge, effectively ignoring the ROM library.

## RD/DUT Synchronization

Two main elements of the library for a device are synchronization (conditions and vectors) and RD test vectors.

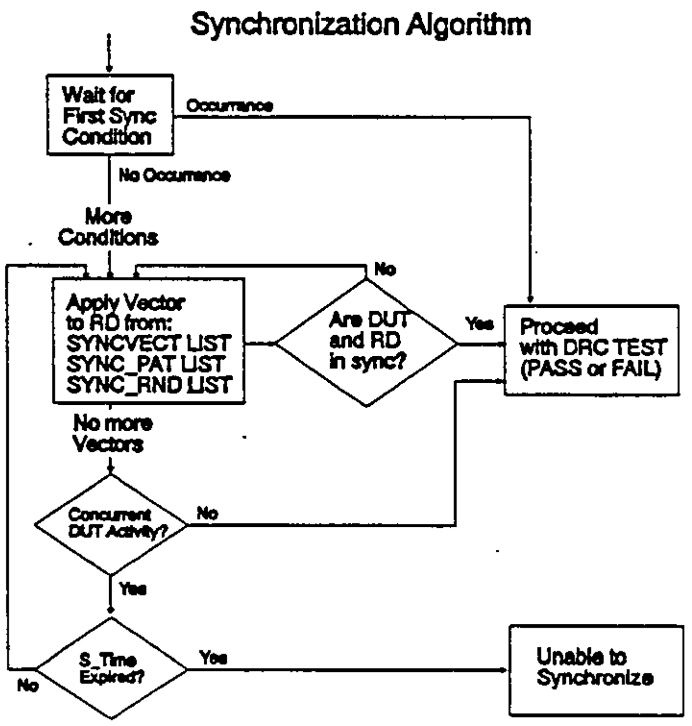
## Specifying RD/DUT Synchronization

For most devices other than combinatorial and RAM chips, it is necessary to define the synchronization requirements between RD and DUT. This is because the two devices must first be in synchronism before a PASS/FAIL determination can be made.

To ensure synchronization, we must be able to observe or infer all the internal states of each device. For example, an octal latch can be checked for synchronization by merely observing its outputs when they are enabled. However, an 8-bit FIFO shift register would require 8 pulses on its serial-shift-in line before we can infer the internal hidden states.

## S_Time Synchronization

1. Monitor DUT for activity that ensures synchronism.
2. Stimulate RD until it "catches up" to a static DUT.


Synchronization Algorithm
Wait for First Sync Condition
Occurrence
No Occurrence
More Conditions
Apply Vector to RD from: SYNCVECT LIST SYNC_PAT LIST SYNC_RND LIST
No
Are DUT and RD in sync?
Yes
Proceed with DRC TEST (PASS or FAIL)
No more Vectors
Concurrent DUT Activity?
No
Yes
S_Time Expired?
Yes
No
Unable to Synchronize

One of the failure modes of a DUT can be that it is in an illegal state or simply cannot be synchronized. Every attempt is made, therefore, to synchronize RD and DUT before a PASS/FAIL test result is presented. Under certain rare board conditions, the DUT is not compared because of an inability to synchronize (Unable to Synchronize result). Two methods are employed during the S_Time to synchronize RD and DUT:

1. Monitor DUT for activity that ensures synchronism.
2. Stimulate RD until it "catches up" to a static DUT.

## Synchronization Algorithm

The synchronization algorithm during S_Time is shown on the opposite page.

As can be seen from the diagram, synchronism is first attempted with a series of Sync Conditions. These are chosen as the necessary and sufficient conditions that would guarantee the RD and DUT would be in the same state.

For example, two 7474 D flip flops would be in the same state after logic 0 on the clear or preset lines or after a rising edge on the clock lines. If any of these occur, the devices are assumed to be synchronized and comparison testing is enabled.

For devices with internal hidden memory states, Sync Conditions are the only effective technique for synchronizing and care must be taken to ensure that they are specified completely to avoid falsely assuming synchronization.

If these conditions do not occur, the second method is tried, stimulating the RD. The stimulus vectors may be defined explicitly (SYNC_VECT), chosen from the RD Test pattern (SYNC_PAT), or created randomly (SYNC_RND). If no conditions or vectors are specified, SYNC_RND is automatically chosen for the

duration of S_Time (beware of false synchronization if there are hidden states).

For our 7474 example, a suitable SYNC_VECT would be a pulse on both clock lines. SYNC_PAT can be an effective synchronization definition, but it should only be used if the RD Test pattern completely exercises a device. SYNC_RND is the easiest to specify although it is less effective.

After each vector is applied, RD and DUT are checked to see that their outputs are the same and, if so, DRC testing proceeds. For tristatable devices, a further parameter, SYNC_GATE, defines when it is valid to check for synchronism on an output pin. This is usually a logic level on the output enable pin.

1. **What activity could initialize a device into a known state? The Sync Conditions should be specified accordingly.**
2. **How might a device be stimulated to cycle it through all possible states, or given a known state of DUT, how would an RD be put into the same state? The vectors should be specified accordingly.**

## Synchronization Commands

```
S_TIME
SYNC_COND
SYNC_VECT
SYNC_PAT
SYNC_RND
SYNC_IGNORE
SYNC_PINS
SYNC_GATE
SYNC_RESET_OFF
SYNC_GR_END
```

When defining the synchronizing parameters for a new library device, it is best to ask yourself two questions:

1. What activity could initialize a device into a known state? The Sync Conditions should be specified accordingly.
2. How might a device be stimulated to cycle it through all possible states, or given a known state of DUT, how would an RD be put into the same state? The vectors should be specified accordingly.

For programmable devices that have hidden states, method 1 should be used alone without method 2.

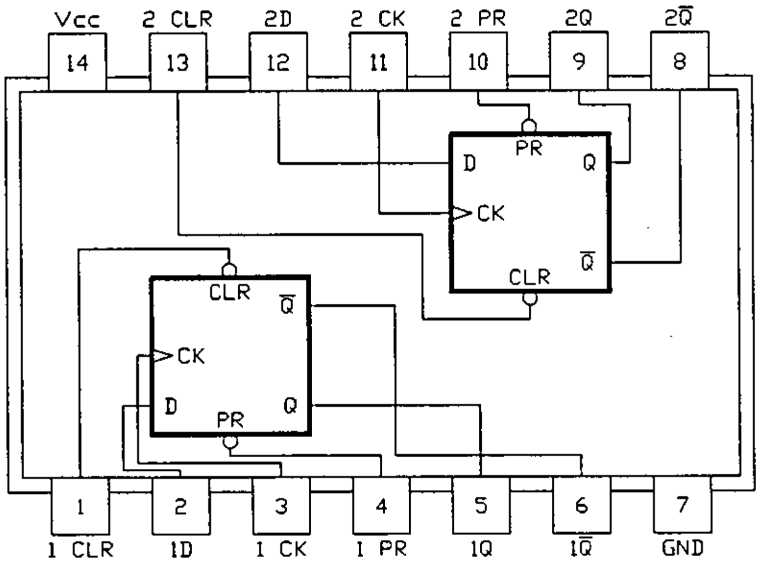
## Synchronization Commands

The actual commands that define synchronization are:

| | |
|---|---|
| S_TIME | duration of synchronizing attempts |
| SYNC_COND | defines conditions sufficient to synchronize from on-board activity |
| SYNC_VECT | defines vectors to be sent to RD |
| SYNC_PAT | specifies RD Test pattern instead of sync vectors |
| SYNC_RND | specifies random pattern instead of sync vectors |
| SYNC_IGNORE | specifies any pins that should be ignored while checking for synchronism |
| SYNC_PINS | specifies the DUT output pins which should be inactive while RD vectors are applied during synchronization. |
| SYNC_GATE | defines how tristate outputs are enabled on DUT. Syntax is similar to GATE. |
| SYNC_RESET_OFF | disables the Reset pulse. Normally, a Reset is issued before checking for each sync condition. |
| SYNC_GR_END | identifies the end of the group of sync parameters. |

Syntax for the preceding commands is found in Appendix II of the Operator's Manual.

```
NAME                 7474
SIZE                 14
SYNC_COND            <1 P1=L + P4=L + P3=R>
SYNC_VECT            1 <1 H P3=C P2=D5>
SYNC_PINS            1 3 4
SYNC_GR_END:
```



**7474 Flip-Flop**

The example on the facing frame is a library definition of a device (a 7474 flip-flop) with synchronization parameters. The pin diagram can be found on the lower frame.

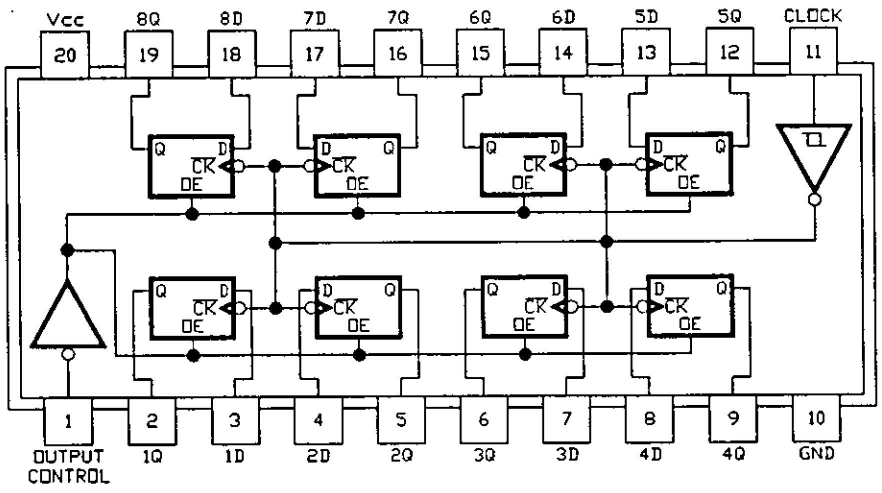
Notes:

1. < and > are called *brackets*. (For additional information refer to Appendix II page 69).
2. For illustration purposes, only half of this dual flip-flop has been specified.
3. A number before the first bracket specifies repetitions of the entire expression. A number after an opening bracket specifies repetition of the bracketed portion.
4. P #=R means check for a rising edge.
5. H alone means unspecified pins are driven high.
6. P #=C means provide a clock pulse of proper polarity.
7. P #=D # means apply a level to the RD pin that is the same as the specified DUT pin number.

## Hands-On Exercise

Complete the Library Definition for the 7474 flip-flop on the opposite page by adding the missing synchronization parameters for the second half of the chip.

```
NAME           74374
SIZE           20
SYNC_COND      <1 P11=R>
SYNC_VECT      1 <1 L P11=C P3=D2 P4=D5 P7=D6 P8=D9
                    P13=D12 P14=D15 P17=D16 P18=D19>
SYNC_GATE      P1=L
SYNC_PINS      11
SYNC_GR_END:
```

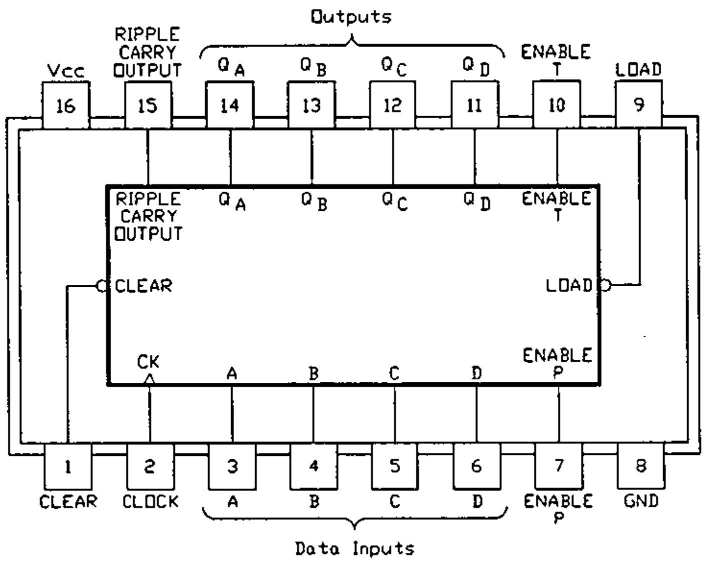


74374 Octal Latch

Notes:

1. Sync Vectors are specified on one line in the editor.
2. A "Synchronization timeout" message appears if the S_TIME value is too low for the complete vector list to be executed. Possible causes include: S_TIME too small, vector list too long, insufficient activity on SYNC_GATE.
3. The DUT clock (Pin 11) must be inactive while vectors are applied to RD (SYNC_PINS command).

```
NAME           74161
SIZE           16
SYNC_COND      <1 P1=L+P9=L*P2=R>
SYNC_VECT      1 <1 H P9=L P7=L P10=L P2=C P3=D14
                 P4=D13 P5=D12 P6=D11>
SYNC_PINS      2 7 9 10
SYNC_GR_END:
```

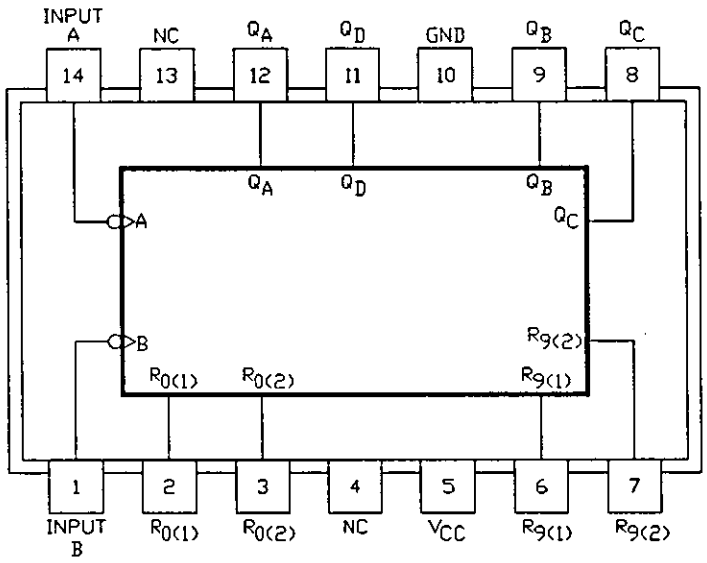


**74161Synchronous 4-Bit Counter**

Notes:

1. The expression <A+B> means A and B must occur simultaneously. <A>+<B> means A and B must occur, independently, in any order.

```
NAME            7490
SIZE            16
SYNC_COND       <1 P6=H*P7=H+P2=H* P3=H>
SYNC_VECT       2 <1 L P14=C P1=D1>/<5 L P1=C P14=D14>
SYNC_PINS       1 14
SYNC_GR_END:
```



**7490 Decade Counter**

Notes:

1. Sync Vectors are applied to this Decade Counter (RD) when the DUT is inactive. Clock pin 14 receives a pulse followed by 5 pulses on clock pin 1. This is repeated twice to cycle through all 10 states.
2. The general rule for devices that have dual internal structures is to hold one in the same state as DUT while stimulating the other (eg. P1=D1).
3. ⋰ forces execution of the expression on the left, then the expression on the right.

## Specifying RD Test Vectors

1. A definite stimulus produces a predictable response.
2. A definite stimulus produces a response that:
    - is not predictable but is consistent (e.g. PROMs)
    - is not consistent (e.g. DRAMs)

| | |
|---|---|
| 0 | logic 0 for an input pin |
| 1 | logic 1 for an input pin |
| L | logic 0 for an output pin or Gnd |
| H | logic 1 for an output pin or Vcc |
| Z | tristated output pin |
| D | disable comparison on all output pins for this vector (used for devices with internal memory that power up in indeterminate states until initialized). |

## Specifying RD Test Vectors

Test patterns for RD Test may be specified using conventions for two categories of devices.

1. A definite stimulus produces a predictable response.
2. A definite stimulus produces a response that:
   - is not predictable but is consistent (e.g. PROMs)
   - is not consistent (e.g. DRAMs)

Category 1 comprises most devices and the vector assignment conventions are:

| | |
|---|---|
| 0 | logic 0 for an input pin |
| 1 | logic 1 for an input pin |
| L | logic 0 for an output pin or Gnd |
| H | logic 1 for an output pin or Vcc |
| Z | tristated output pin |
| D | disable comparison on all output pins for this vector (used for devices with internal memory that power up in indeterminate states until initialized). |

```
NAME        7400
SIZE        14
RDTEST      VECTORS
00H 00HL    H00H00H
10H 10HL    H10H10H
01H 01HL    H01H01H
11L 11LL    L11H11H
END_VECTORS
```

| | |
|---|---|
| 0 | logic 0 for an input pin |
| 1 | logic 1 for an input pin |
| L | undefined pin: input, high, low or tristate output (try to drive pin low, but allow any state on pin) |
| H | undefined pin: input, high, low or tristate output (try to drive pin high, but allow any state on pin) |
| + | output pin with a high or low level (not tristate). Vcc and Gnd are specified with +. |
| * | output pin with a high, low or tristate level |
| Z | tristated output pin |

In a .LIB file created from the Editor, the pins are listed in vector rows with optional spaces to improve readability. The starting command is `RDTEST VECTORS` and the last command is `END_VECTORS`.

Example:

```
NAME       7400
SIZE       14
RDTEST     VECTORS
00H 00HL   H00H00H
10H 10HL   H10H10H
01H 01HL   H01H01H
11L 11LL   L11H11H
END_VECTORS
```

Category 2 has two subcategories (`CHECKSUM`, `PRESENCE`) and uses the following vector asignment conventions:

| | |
|---|---|
| 0 | logic 0 for an input pin |
| 1 | logic 1 for an input pin |
| L | undefined pin: input, high, low or tristate output (try to drive pin low, but allow any state on pin) |
| H | undefined pin: input, high, low or tristate output (try to drive pin high, but allow any state on pin) |
| + | output pin with a high or low level (not tristate). Vcc and Gnd are specified with +. |
| * | output pin with a high, low or tristate level |
| Z | tristated output pin |

For PALs and PROMs, the header command for the vector table is:

```
RDTEST VECTORS CHECKSUM
```

The footer command is:

```
END_VECTORS
```

For DRAMs and devices with an inconsistent C_SUM, the header command is:

```
RDTEST VECTORS PRESENCE
```

The footer command is:

```
END_VECTORS
```

Note that normally the RD Test Vectors checksum would only be created by the user for new types of PALs and PROMs. Types that are in the library already have their RD Test specified using C_SUM as described in Section 3.3.2 of the Operator's Manual.

The RDTEST VECTORS PRESENCE table used on new DRAMs checks that the RD is able to drive its outputs and that it is not driving any of its inputs. The vector rate of RD Test is not fast enough to meet the timing requirements for reliable data.

## VERIFYING AND PRINTING RD TEST VECTORS

RD Test may be run on a device from the Manual Mode that is a sublevel of develop rd_lib. Shown on the screen will be:

- line number that failed
- pin number that failed
- C_SUM error if applicable

The RD Test patterns for user libraries and the ROM-resident system libraries may be printed. Press rd_lib in Develop mode and select the device whose pattern is to be printed under manual. Press ESC and prt_pat which will send the formatted test pattern to the RS232C port.

## Answer to exercise on page 5-13

```
SYNC_COND <1 P10=L+P13=L+P11=R>
SYNC_VECT 1<1 H P11=C P12=D9>
SYNC_PINS 10 11 13
```

# Section 6
# Board Testing

*In Section 3, a basic sequence was created without thinking about file handling. We will now cover in greater depth the process of enhancing Sequence Files, Programming, and Data Logging.*

TEST1.nsq

TEST1.SEQ
TEST1.LOC

TEST1.seq
TEST1.loc

LIB

LOC

SEQ

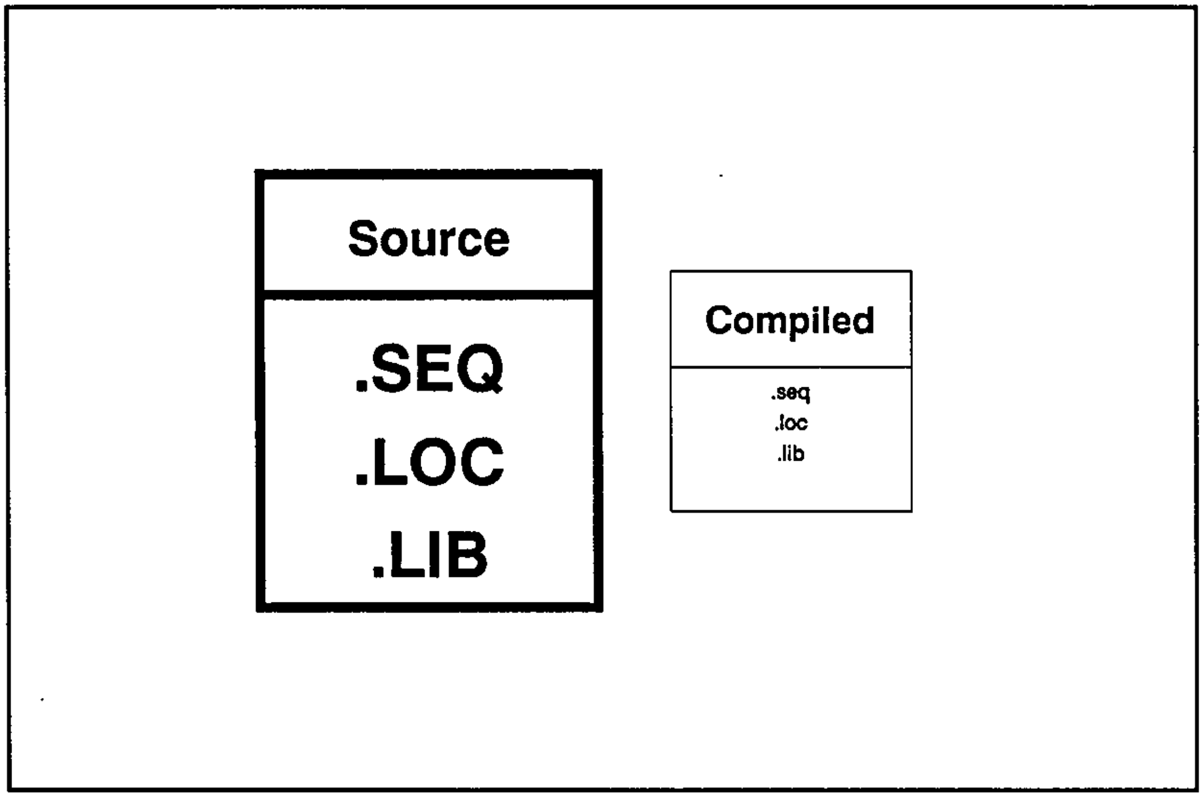
## Filetypes

Initial sequence development is done in a mode that requires no knowledge of the file structure, syntax or editor. The new_seq in develop mode creates a temporary file (**.nsq**) that is transparent to the user and automatically creates basic source and compiled files. If you list the file directory from develop files mode, you will observe the following:

| | |
|---|---|
| **TEST1.nsq** | may be further modified under develop new_seq mode |
| **TEST1.SEQ**<br>**TEST1.LOC** | source files that may be further refined using the editor |
| **TEST1.seq**<br>**TEST1.loc** | compiled files that may be used to test a board. |

Recall that there are three types of interrelated files that make up a sequence:

- **LIB** - library files are resident in ROM and define default parameters for most standard devices. New library files may also be created and stored in the cartridge by a user.
- **LOC** - location files contain all the changes a user made to the default parameters on a device. These are referenced to the board location designator. The parameters may be thought of in two groups: device functionality (e.g. P.E., initializing, qualifying), surrounding activity (e.g. frequency, input pin states).
- **SEQ** - sequence files define the order of test using individual device LOC files. Also, user comments can be inserted.

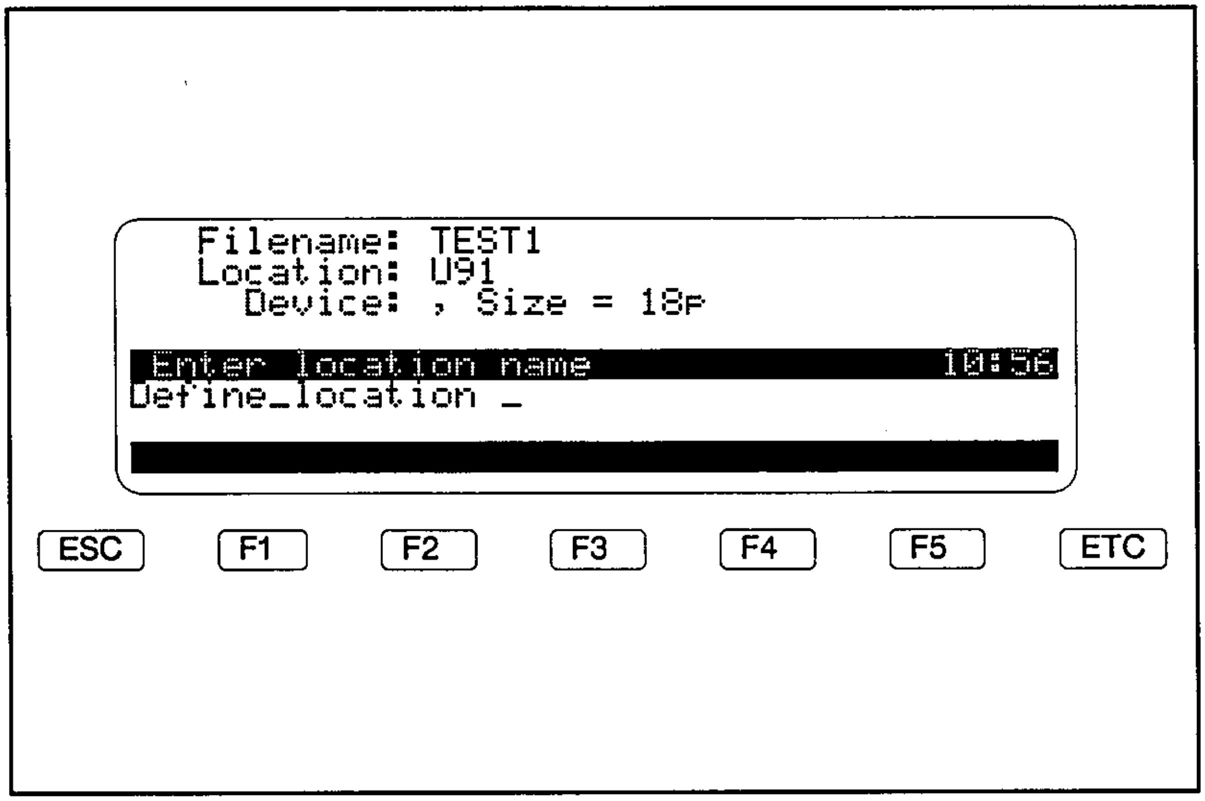

Source
.SEQ
.LOC
.LIB
Compiled
.seq
.loc
.lib

Creation of a Sequence and its component files is similar in some ways to a software computer program. A .SEQ file will call .LOC files which in turn call .LIB files. Instead of lines of computer code, of course, the FLUKE 900 files are made up of the keystrokes as they were entered. These .SEQ, .LOC and .LIB files are designated as Source files and they can be edited by adding or deleting keystroke commands.

The tester actually runs compiled versions of the files and these are designated by lower case, .seq, .loc and .lib names. Compiled files typically occupy only one third of the cartridge memory space of their Source versions but are not modifiable (run only). Thus, after inputting a new Sequence from the keyboard, a compile operation is done on the files when end is pressed in develop new_seq mode.

To summarize the file structure and its application, let's follow the complete procedure for our sequence TEST1.

| Procedure | FLUKE 900 Mode | Active Files |
|---|---|---|
| 1. ICs tested on known-good board with parameter adjustments to make them Pass. Source files **TEST1.LOC** and **TEST1.SEQ** are automatically created as well as compiled versions **TEST1.loc** and **TEST1.seq.** | develop new_seq | TEST1.nsq |
| 2. To improve the guided diagnostics the editor is invoked to reorder the sequence and add user comments. | develop files editor | TEST1.SEQ TEST1.LOC |
| 3. The refined source files **TEST1.SEQ** and **TEST1.LOC** are compiled to produce the run-only files **TEST1.seq,** and **TEST1.loc.** | develop files editor compile | TEST1.seq TEST1.loc |



```
   Filename: TEST1
   Location: U91
     Device: , Size = 18p

 Enter location name                     10:56
Define_location _
```

## Temporary Files

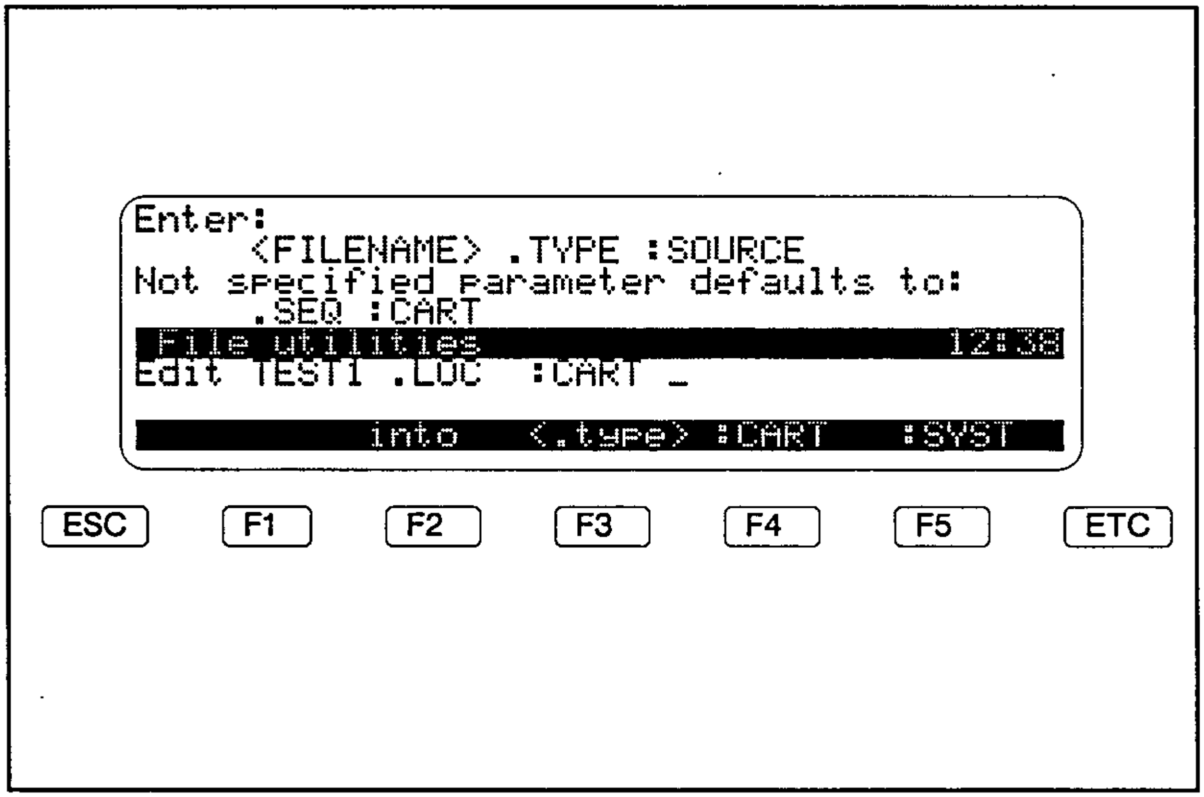
Note that a straightforward sequence can be created by initial development under new_seq in the desired order, without manually editing or compiling files. Usually, however, the temporary file created under new_seq is used as a master to develop other ordered troubleshooting sequences from. Once further development is done on a source file using the editor, it no longer resembles the temporary file from which it was derived. There is no way to update the temporary file, so the source copy becomes the new working development file.

A method for reordering the sequence without using the editor is to re-enter the location numbers for an existing new_seq file in the desired order. The order in TEST1.nsq is now:

U14
U2
UF3
U62
U91

Let us reorder it so that U91, the oscillator chip, is the first on the list. Select develop new_seq mode and type T E S T 1 ENTER (note warning that the new_seq file exists already). Enter in the location of the first device we want by typing U 9 1 ENTER, and observe the screen.

Now redefine it as U91 by pressing def'n_loc, and entering the same location. You will have to confirm that U91 will replace U14 as the first device by pressing yes. Repeat this procedure for each device in our new desired order U91, U14, U2, UF3, U62 and press end to create new SEQ and LOC file versions. (Note that you are asked to confirm this since it will wipe out the old SEQ and LOC files.) Two other methods of reordering are to use the built-in Editor or download a PC-generated SEQ file.



```
Enter:
     <FILENAME> .TYPE :SOURCE
Not specified parameter defaults to:
     .SEQ :CART
 File utilities                          12:38
Edit TEST1 .LOC  :CART _

           into   <.type> :CART    :SYST
```

## Editing Source Files

Let us now use the Editor to make a couple of refinements to our sequence. We will insert an operator prompt into the .LOC file advising the operator not to use a reference device when testing the 8284 at U91, and we will reorder the .SEQ file through the editor this time so the last two devices are interchanged.

First select files under the develop mode. The resultant screen is shown on the left.

These labels, and the second menu available through pressing ETC, are file utilities described in detail in the User's Manual. The one we want now is edit to change the TEST1.SEQ and TEST1.LOC files. Press edit and follow the syntax instructions in the display, using TEST1.LOC :CART as our filename (files are assumed to reside on the cartridge if the last name field is left blank). Your screen will look similar to that shown left when you type the following:

edit
T E S T 1
.type .LOC :CART

Note that when creating a new file you must edit into a filename. For an existing file you can merely edit a filename. This syntax permits you to keep a backup file of the original file you are changing (i.e. edit A into B will define B as an edited version of A while retaining the original A).

```
NAME        U91
SIZE        18
RDT_ENABLE  OFF
IGNORE      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
ACTIVITY    P2=2.386MHz 1 % P8=4.773MHz 1 %:

NAME        U14
LOAD        7400:

NAME        U2
LOAD        8288:

NAME        UF3
LOAD        PAL22V10
C_SUM       49446
S_TIME      ON 5000:

NAME        U62
LOAD        8259
S_TIME      OFF 3000
GATE        OFF
TRIGGER     ON P1=LL P2=LL P7=HX P27=LH
IGNORE      16
F_MASK      50:

NAME        DEFAULT
T_TIME      5000:
```

**TEST1.LOC Listing**

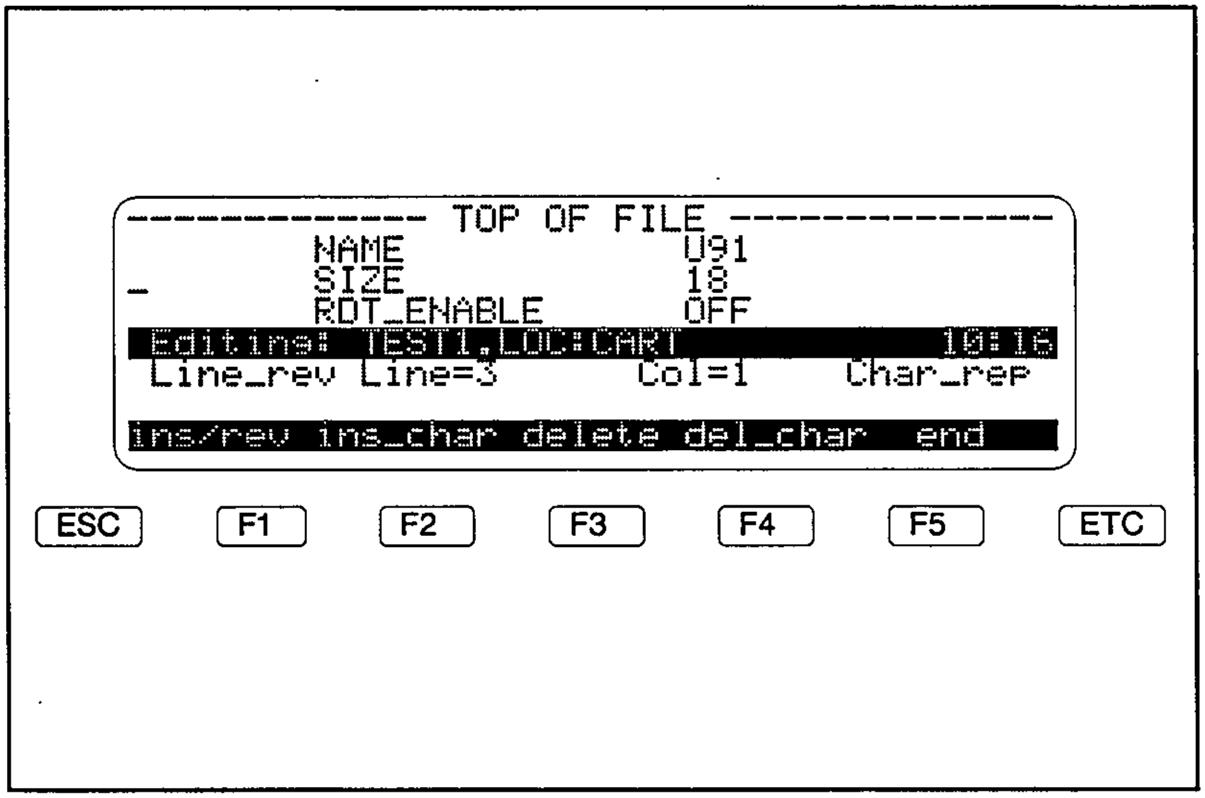
Press [ENTER] and you will see the top-of-file line, the device and parameter entries, and the global test time change we made. You can scroll through the file to observe the total listing as shown on the left.

Each line is known as a command and they are grouped by location into command groups.

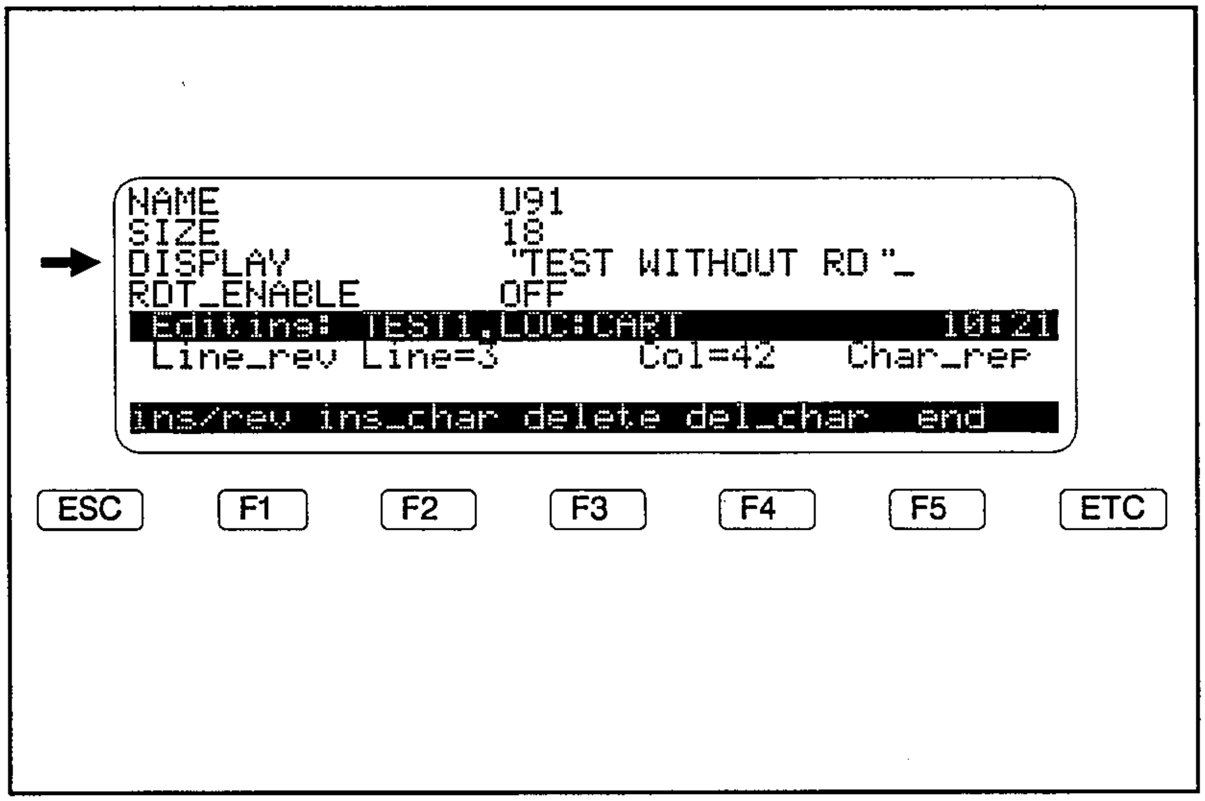
There are a few rules of structure and syntax to both .SEQ and .LOC files:

- Instruction line **labels** must begin in column 1.
- Labels, instruction commands and parameters are separated by at least one space. In practice the [→] key produces an easily readable spacing.
- ";" identifies the rest of the characters on the line to be a comment that will merely be listed and not executed.
- ":" is used to terminate each command or group of commands. For a **.SEQ** file this means every line that requires operator action (i.e. pressing [TEST]) ends with ":", and for a **.LOC** file every group of commands associated with a device ends with a ":".
- Blank lines make a file listing more readable, but do not affect its execution.

The standard format for a .LOC file uses the NAME command to specify a device location, the LOAD command to specify the device at the location, followed by appropriate parameter commands. One location in a .LOC file may be chosen as DEFAULT and, in this case, the parameter commands are taken to be global values that apply to all devices in the file.



```
------------- TOP OF FILE -------------
        NAME                U91
_       SIZE                18
        RDT_ENABLE          OFF
Editing: TEST1.LOC:CART                10:16
Line_rev Line=3       Col=1    Char_rep

ins/rev ins_char delete del_char  end
```



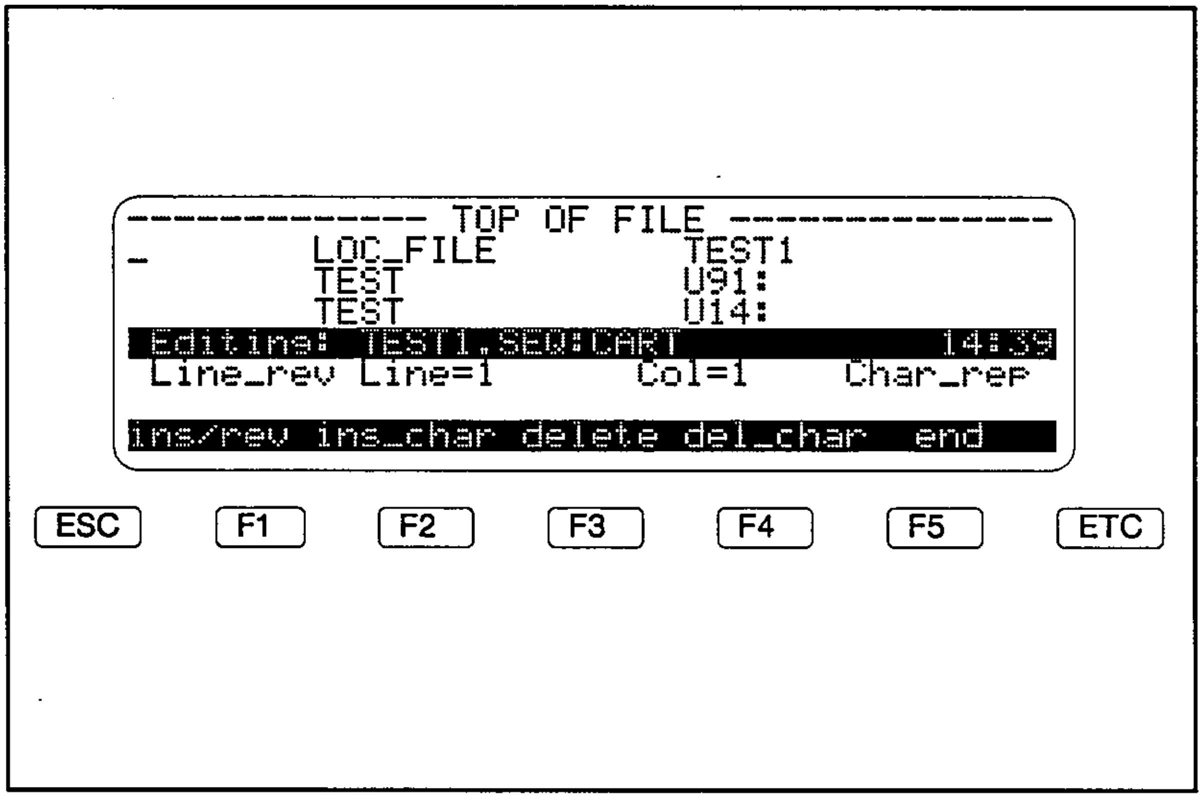
```
NAME                U91
SIZE                18
DISPLAY             "TEST WITHOUT RD "_
RDT_ENABLE          OFF
Editing: TEST1.LOC:CART                10:21
Line_rev Line=3       Col=42   Char_rep

ins/rev ins_char delete del_char  end
```

Note that the [F1] key has the label ins/rev. This alternately chooses to insert a blank line when [ENTER] is pressed or to revise (write over) the line that the cursor points to. The [F2] key has the label ins_char and alternately selects to insert or replace characters. delete removes and entire line and del_char a single character.

To add the operator prompt, position the cursor as shown on the bottom frame on the facing page and press ins/rev so we can insert a command line. Refer to the screen on the left.

Press [ENTER] to open up a line within the file and type in the DISPLAY command shown in the next frame. Note that the text scrolls to the left as you enter the text.

Press end to exit the Editor.



```
------------- TOP OF FILE -------------
_        LOC_FILE          TEST1
         TEST              U91:
         TEST              U14:
 Editing: TEST1.SEQ:CART              14:39
 Line_rev Line=1        Col=1      Char_rep

ins/rev ins_char delete del_char  end
```

We have finished editing the location file and now wish to reorder the sequence file. Invoke the Editor for TEST1.SEQ. We observe the following listing and screen at left:

```
LOC_FILE     TEST1
TEST         U91:
TEST         U14:
TEST         U2:
TEST         FU3:
TEST         U62:
```

The standard format for a .SEQ file starts with the LOC_FILE instruction to specify its associated .LOC file. The use of an END command as the last command of a .SEQ file is good practice, but one that is not strictly required. It restarts the sequence from the beginning after displaying an "END OF SEQUENCE" message.

Proceed to insert and delete characters or lines until the listing appears as:

```
LOC_FILE     TEST1
TEST         U91:
TEST         U14:
TEST         U2:
TEST         U62:
TEST         FU3:
```

Now, terminate editing of TEST1.SEQ with end. Note that comments may be just as easily inserted into the .SEQ file as the .LOC file. Both source files must now be compiled. Do this by using the compile key for TEST1.SEQ and TEST1.LOC.

Verify that you have a finished sequence by trying to run it under sequence mode. Try and test part of the Demo UUT using your new sequence, TEST1.

*Congratulations!!* You have completed the full process of developing your first sequence.

**Example:** **INTERFACE.SEQ file**

```
        LOC_FILE   INTERFACE

START   TEST       U1
        DISPLAY    "FREQ OF OSCILLATOR=4MHZ ":

        TEST       U3:        ; TEST the BUS first
        TEST       U4:
        TEST       U2:

; This whole line is a comment

        TEST       U5:
        TEST       U7:
```

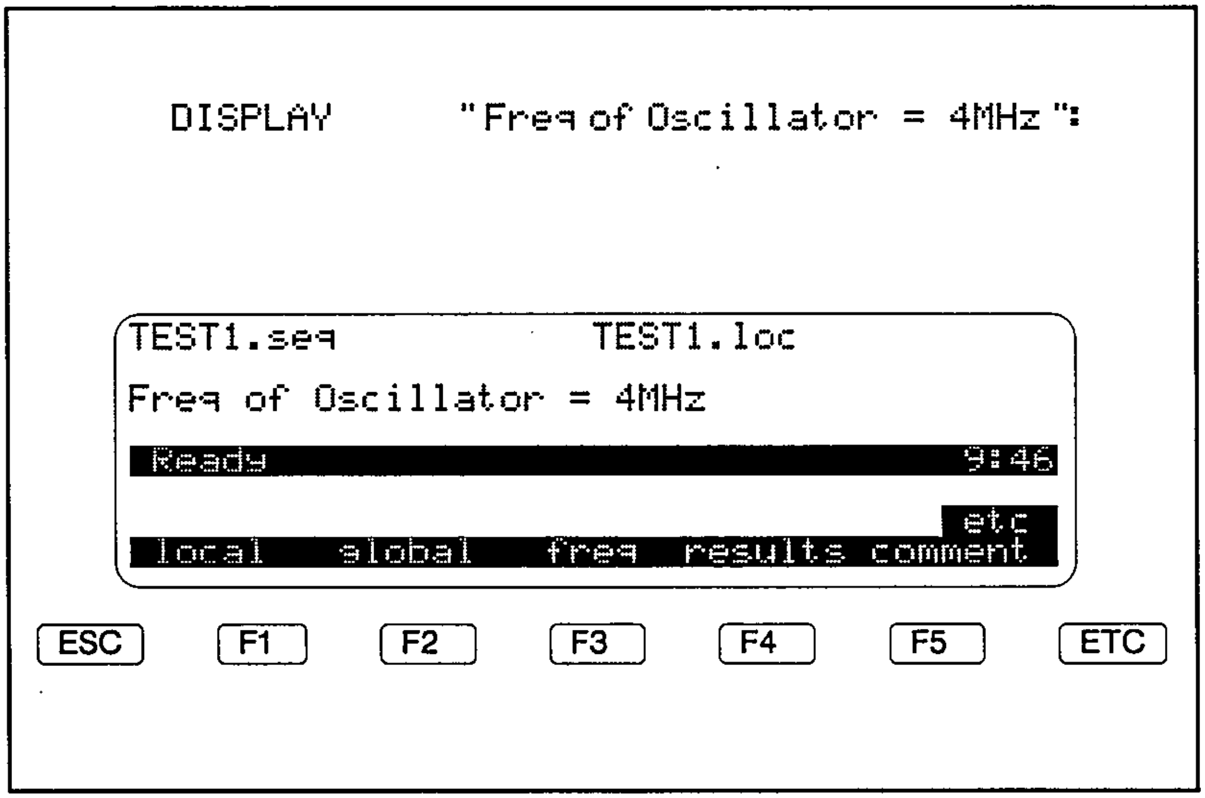
**Example:** **INTERFACE.LOC file**

```
        NAME       DEFAULT  ; Global parameters
        F_MASK     40       ; F_MASK=40 nsec
        THRSLD     1500:    ; logic 1=1.5 V

        NAME       U1
        LOAD       74244:

        NAME       U2
        LOAD       74245:

        NAME       U3
        LOAD       74373:
```

## Sequence Enhancement

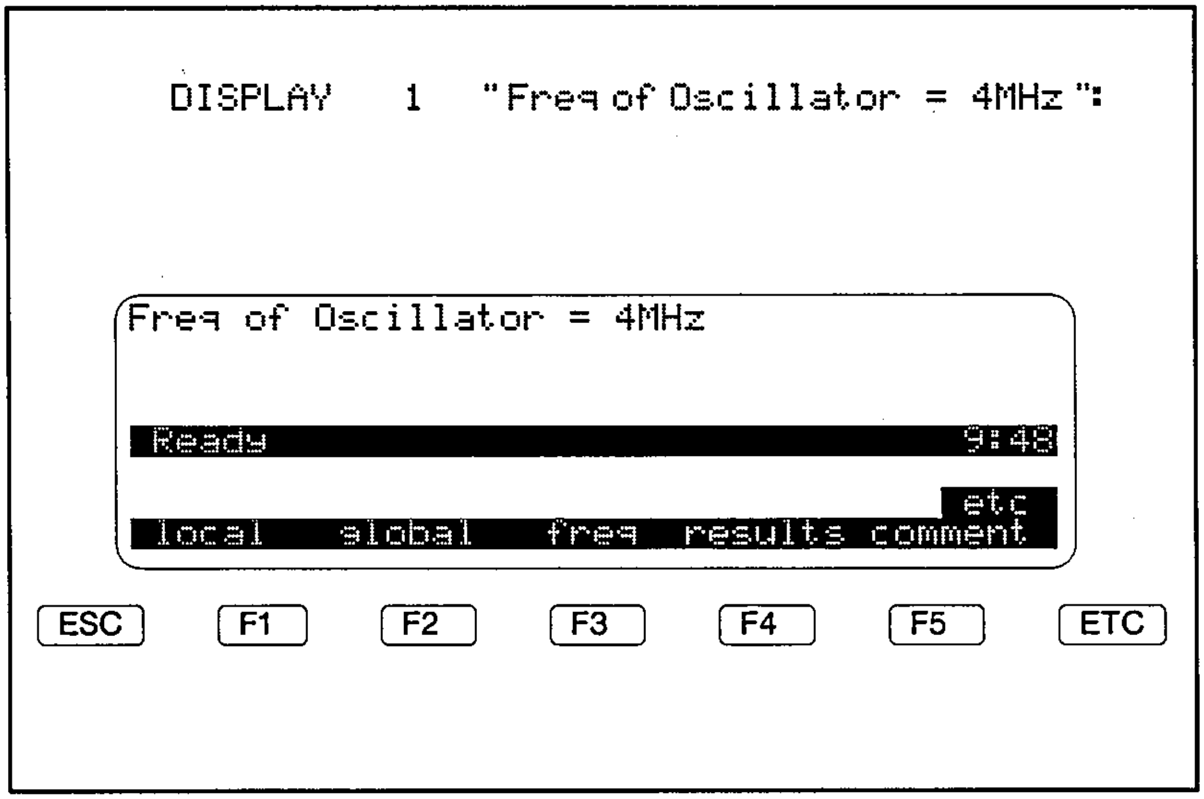
We are going to examine source files in detail and explore some advanced capabilities of the programming language itself. On the facing page are typical .SEQ and .LOC files using the standard syntax rules.

Notice that the function of the .SEQ and .LOC files are quite different. The .SEQ file is like a computer program that acts on data while the .LOC file is a database of locations with associated devices and parameters. We will shortly introduce some .SEQ commands that permit programmed jumps, conditional commands and function calls.



```
DISPLAY      "Freq of Oscillator = 4MHz ":
```

```
TEST1.seq                 TEST1.loc
Freq of Oscillator = 4MHz
Ready                                   9:46
```



```
DISPLAY   1   "Freq of Oscillator = 4MHz ":
```

```
Freq of Oscillator = 4MHz

Ready                                   9:48
```

## Display

The DISPLAY command puts a message, typically an operator prompt, on the LCD screen. The DISPLAY command is usually placed in the .LOC file, however, it also may be placed in a .SEQ file.

When the DISPLAY command is used without a line number, as shown in the upper frame on the opposite page, the message is automatically placed on line three. If the message is longer than 40 characters, the message is wrapped to the following line. Messages longer than 80 characters are truncated.

Use of the optional line number (1 through 4) will place your message on the specified line as seen in the lower frame.

## Functions

```
         TEST        U5:

         FUNCTION    MESG-1    ; display first prompt
         TEST        U6        :

         FUNCTION    TRIG-1    ; setup external trigger_1
         FUNCTION    MESG-2    ; prompt to attach trigger
         TEST        U7:
         END:

MESG-1   DISPLAY      "Run selftest No 2 "
         END_FUNCTION:

MESG-2   DISPLAY      "Attach trigger input lead to 8088 pin 28 "
         END_FUNCTION:

TRIG-1   TRIGGER     ON E=01   ; trigger on rising edge
         END_FUNCTION:
```

## Functions

When a series of commands are needed in several places in the sequence, they can be grouped together into a FUNCTION, and then called as a subroutine from another command group. The TEST command and the " : " *cannot* be part of the function.

Functions are ideal for parameter settings that may be used with more than one DUT, such as an external trigger, or for operator prompts (reminders) that have to be repeated in more than one place. After a Function has been called and executed, control is returned to the command following the call.

A function is defined by a label, and a command group ending with an END_FUNCTION statement. Any specified function can be called by using the command FUNCTION followed by its name.

## Jump

```
         TEST         U5:

         JUMP   CONT-2      ; Jump to CONT2

CONT-1   TEST         U6:
         TEST         U87:

CONT-2   DISPLAY       "Run selftest No 2 "
         TEST         U7:
         END:
```

## Jump

This command will cause the sequence to continue execution at the specified *label*. A valid label starts in column 1 of the source file.

## If...Then

```
IF  PASSED THEN DISPLAY "BOARD IS GOOD"

IF  FAILED + P4=A * (P22=2.098kHz 2 % + P11<1MHz) THEN JUMP VEC1
```

Conditions:

| | |
|---|---|
| PASSED | DUT passed |
| FAILED | DUT failed |
| RD_SHORTED | Short in ZIF |
| RD_TEST FAILED | Bad RD |
| CLIP_TEST_FAILED | Clip check fail |
| FAILED_TO_SYNC | Unable to sync RD with DUT |
| NO_GATE | Gate word did not occur |
| NO_TRIGGER | Either trigger word did not occur |
| NO_TRIGGER_WORD1 | Trigger word 1 did not occur |
| NO_TRIGGER_WORD2 | Trigger word 2 did not occur |

Operators:

| | |
|---|---|
| + | OR |
| * | AND |
| < | Pin freq less than value |
| > | Pin freq greater than value |
| = | Pin freq equals value |
| <> | Freq not equal to value |

## If...Then

The IF command is used to test if the specified set of conditions are "true", and if so it will execute the command defined by THEN, otherwise the program will continue executing at the next command line.

The IF command has the following syntax:

IF *condition* THEN *command*

where *condition* is one of the states listed on the left or optionally, one of the conditions listed followed by an *operator* followed by another condition. Conditions can be chained in this manner to satisfy multiple circumstances with parentheses used to specify precidence.

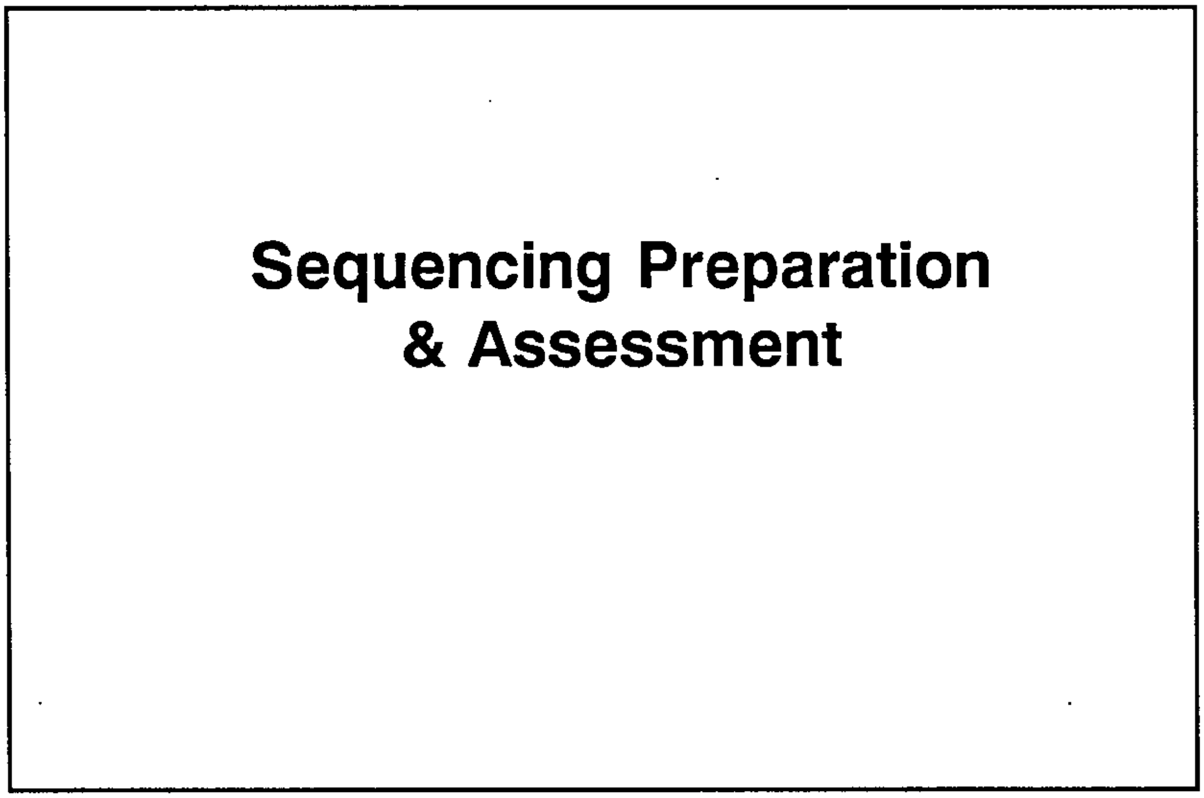
Note: The units for frequency must be spelled Hz, kHz, and MHz.

```
         LOC_FILE   DEMO
       TEST      U2:

       IF P2=4.773MHz 2 %THEN JUMP BUS
       TEST      U91:

       END:

  BUS  TEST      U37:
       TEST      U43:
       TEST      U48:
       END:
```

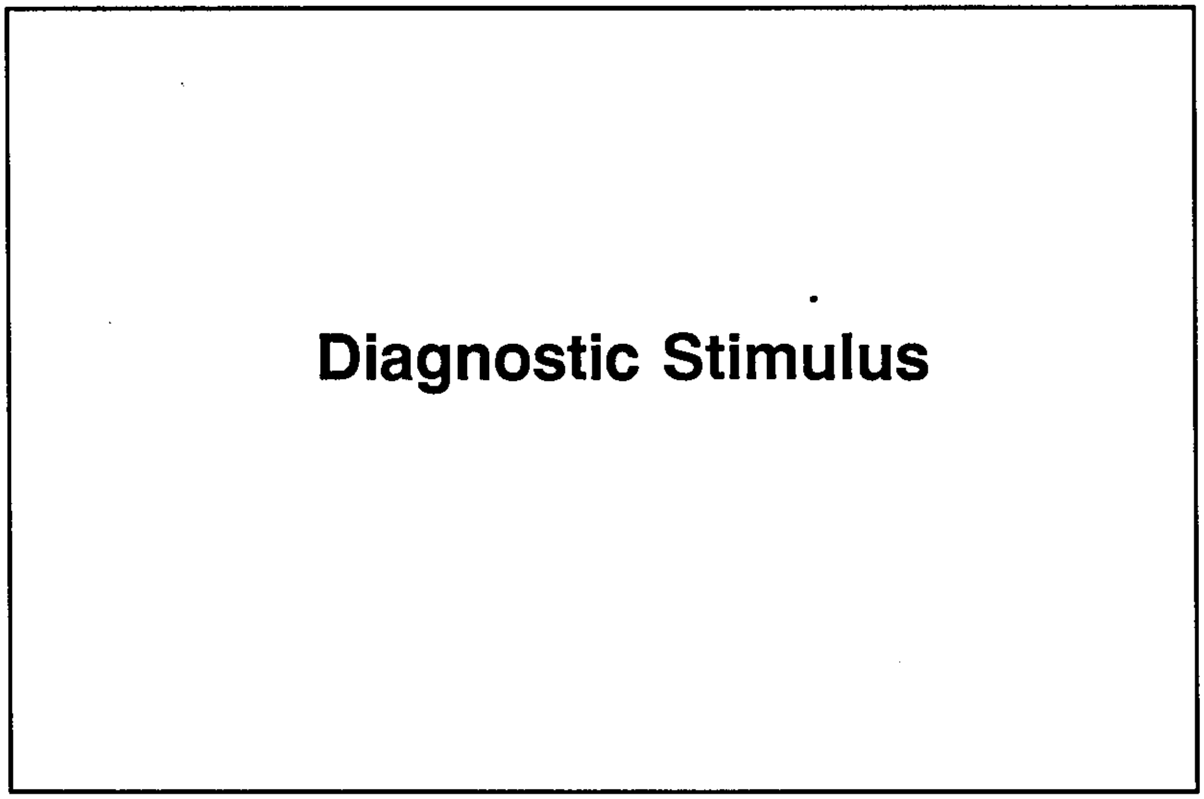
**Sample Sequence File**

As a further illustration of the application of a conditional jump in a sequence, consider the Demo UUT board. Before testing devices on the data and address buses, we probably want to test the bus controller itself at U2.

As part of the test for U2, the 8288, suppose we specify (in the .LOC file) that a 4.773 MHz clock must exist on pin 2. If pin 2 of U2 shows the presence of the 4.773 MHz clock we want to continue testing the other devices on the bus. If the clock is not present, we want to direct the operator to check pin 8 of the oscillator chip at U91. The sequence on the facing frame shows how this would be accomplished.



## Sequencing Preparation & Assessment



## Diagnostic Stimulus

## Board Testing Considerations

The remainder of this Section provides a detailed procedure that is recommended for the creation of effective board test Sequences.

## Sequencing Preparation and Assessment

Decide the quality of diagnostic testing desired, the degree of automation to be included and how much time is available for programming. A 100-IC board will generally take from two to four days to complete the entire sequencing process. Assess the device coverage, percentage of analog components on the board and any areas that exceed the tester speed specifications.

Collect Reference Devices, schematics and any required fixtures including loopback plugs, I/O connectors and extender cards. Arrange RDs in ascending numerical order in an RD retainer tray. One possibility is to have a unique tray for each board type. Another is to have one or two trays that have all devices for a range of boards.

Ensure that clip contacts are clean. Check oxidation on device pins and clean with alcohol if necessary to avoid intermittent contact problems.

Check that +5 volt power supply is good on the board and not at a marginal level that may cause inconsistent results.

## Diagnostic Stimulus

Set up a reliable repeatable stimulus, ensuring that the activity is as extensive as possible. Use Reset wherever possible and verify that it really does initialize the board. Look for a signal that comes from the power on function, such as a capacitor to $V_{CC}$ connected to the microprocessor reset line. On a single board system this is usually straightforward; on a multiple card system, the reset may even be located on another board.

## Verifying that All Devices Pass DRC

1. Verify each device in turn from one end of the board to the other.
2. Verify each device in numerical order (e.g. 7400, 7402, ...) placing a sticker on each IC to keep track of those done.

## Verifying that all Devices Pass DRC

Verify in new_seq Mode that each device on the board passes and, if not, make the required parameter changes. Refer to the .LOC file listing on page A-6 for an example of what parameter changes were necessary. Note that since fmask is set per device, not per pin, it is advisable to test twice when only one pin has a large setting (eg. 150 ns). Test once at 150 and a second time at 30 ns, with the offending pin ignored.

On devices that pass, pressing results state, to highlight active pins will give an indication of how well a device is stimulated. Two convenient ways of performing the new_seq procedure are:

1. Verify each device in turn from one end of the board to the other.
2. Verify each device in numerical order (e.g. 7400, 7402...) placing a sticker on each IC to keep track of those done.

## Adding Signal Conditions Tests

- Fixed Frequencies
- System Clocks
- Activity Presence
- Levels
- Triggers and Gates

## Structuring the Sequence

## Adding Signal Condition Tests

In general, the more you know about a board, the more conditions you can test for. Use secondary signal conditions sparingly; activity checks and triggers are most useful in a sequence structured by signal flow. A missing condition should direct you to a specific fault (e.g. a missing clock should point to the oscillator section).

Adding many condition tests throughout a board, however, will be confusing since there are many missing signals on a dead board. Suitable conditions in order of increasing knowledge of the board operation are:

- Fixed frequencies
- System clocks
- Activity presence
- Levels
- Triggers and Gates

Check conditions only on input pins since outputs are already checked through DRC.

Enable H and L status checks for pins that are pulled up or down through resistors. Measure frequencies on fixed, ungated periodic signals. Check for active signals coming from off the board.

## Structuring the Sequence

The next major step in Sequence creation is to reorder the devices for troubleshooting. This may be done by redefining all locations in new_seq mode, or creating a new .SEQ file with the editor or via the download utility from a PC generated file. The order could be by RD type, fault occurence based on past experience or signal flow. It is advised that the order be by signal flow, since occasionally, failure modes induce apparent faults further down the chain. As an example, failing memory addressing logic can cause all memory devices to appear bad.

## Typical Sequence Order

- reset circuit
- clock, timing generation circuit
- bus control, address decoding
- address bus drivers and logic
- databus drivers and logic
- memory devices
- I/O, remaining circuitry

A typical micro-based board would have the following recommended Sequence order:

- reset circuit
- clock, timing generation circuit
- bus control, address decoding
- address bus drivers and logic
- databus drivers and logic
- memory devices
- I/O, remaining circuitry

To test PC option cards, first check the cardedge interface devices.

If the board diagnostics are effective in first isolating a failure to a part of the board, several small Sequences appropriate to each part could be a good implementation. These smaller .SEQ files will share the same .LOC file for the whole board.

## Adding Operator Screen Messages

## Compiling the Files

## Adding Operator Screen Messages

Employ the editor to add operator prompt messages to the .SEQ file. This can be done also by uploading to a PC, using a full screen editor, and downloading back to system RAM. The added messages could be advice on connecting the patch leads, the type of RD or troubleshooting hints. Function calls are a convenient way to add duplicate messages.

## Compiling the Files

All edited files must be recompiled into executable versions. Press compile and enter the filenames with extensions. Do this for all edited .SEQ, .LOC and .LIB files.

## Documentation

- printout listings of the files
- comment sheet on why parameters were changed
- date, revision level, description of diagnostics
- board statistics: number of ICs, number untested, % covered.

## Typical file sizes for a 100 device board

| | |
|---|---|
| FILE.LOC | 5000 |
| FILE.SEQ | 5000 |
| FILE.loc | 1500 |
| FILE.seq | 1500 |
| FILE.nsq | 3000 |
| Total | 16000 bytes |

## Final Verification and Documentation

Verify the completed sequence by running it on other good boards. Edit any parameter changes (usually fmask in the .LOC file) and recompile.

One way to allow for a range in performance among several good boards is to create a new_seq temporary file (.nsq) with the default fmask of 30 ns. After creating a .LOC file, go into it and edit a global fmask of 40 ns to give an extra margin in comparison resolution.

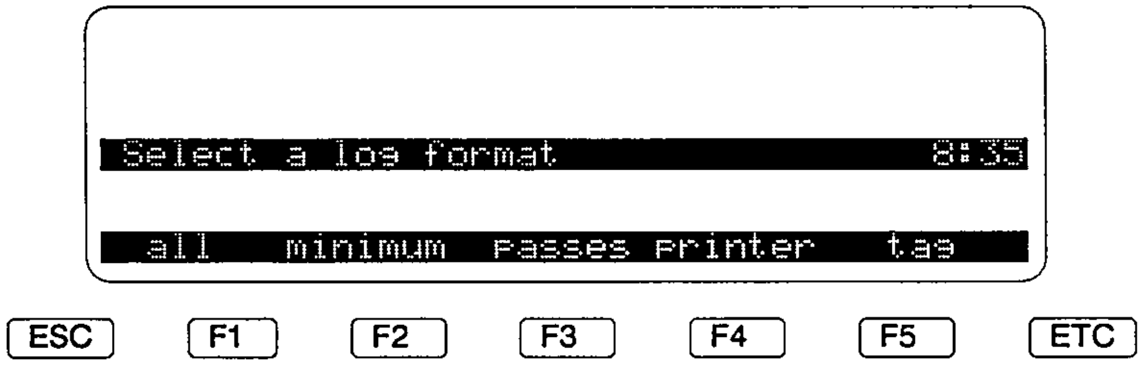
Fully document the newly created sequence including:

- printout listings of the files
- comment sheet on why the parameters were changed
- date, revision level, and description of diagnostics
- board statistics: number if ICs, number untested, and percentage covered

It is a good practice to retain a master copy of all source files on a diskette or cartridge and file it away for safekeeping.

A good rule for file management is to keep a master cartridge or floppy disk containing source and temporary files so they may be modified. Run-only compiled files covering a number of board types may be put on another cartridge. They occupy less space, cannot be readily modified and may be distributed safely to the test operators.

## Getting the Most from Your Test Results


Select a log format 8:35

Log Format Screen

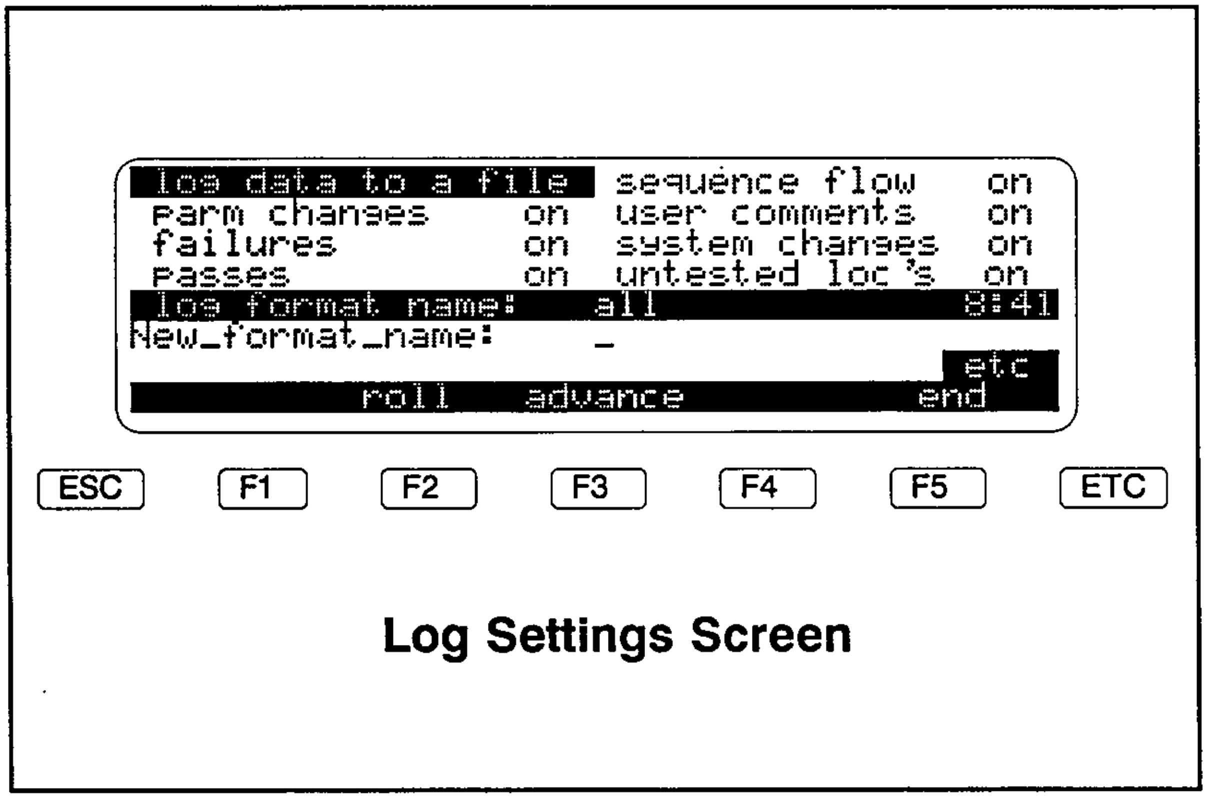
## Getting the Most from Your Test Results

The data logging function of the FLUKE 900 provides a paperless reporting environment to record test results, rework instructions and operator/testor activities. This data can be directed to a cartridge or a printer through the RS232C port. Under system mode, there are three menu keys relevant here:

rs232c used to set baud rate, parity and handshake signals.

printer used to format data stream, if necessary, for a printer with leading spaces and line terminator characters.

config_log permits the user to define five "data filters" and give them names. The factory-configured names appear as function key labels shown to the left.



```
 log data to a file  sequence flow    on
 parm changes     on user comments    on
 failures         on system changes   on
 passes           on untested loc's   on
 log format name:    all            8:41
New_format_name:     _
                                     etc
          roll    advance          end
```

Log Settings Screen

Select all and observe the log configuration on the facing page.

Note that there are eight parameters which may be turned on or off. They indicate the types of data that will be routed to a log file or printer. You may enable them in any combination and assign a new name which will appear as a function key label. roll toggles the on/off flag, advance moves the parameter pointer and end exits the mode.

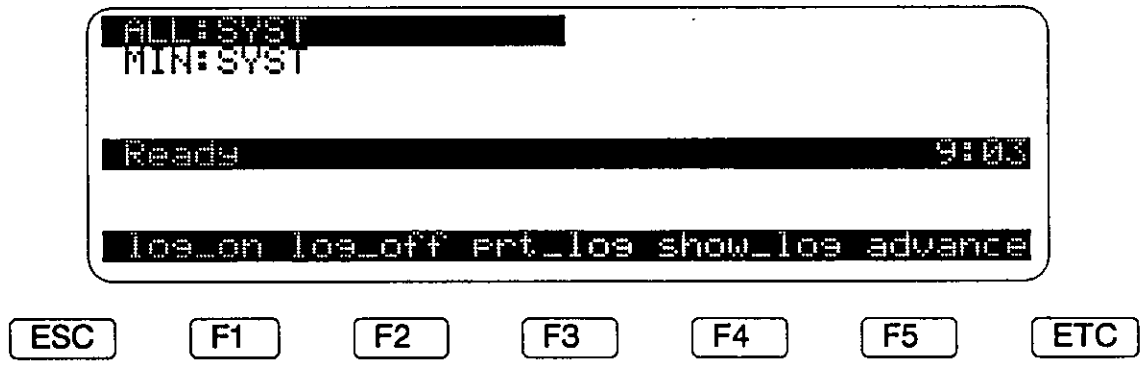
For a repair tag printout to send with a board to rework, you would use the tag format which logs failed locations and user comments. When a user is troubleshooting a board with a sequence, pressing the comment key brings up the next screen.

A simple rework instruction such as:

> "board repaired",
> or
> "not replaced"

will then appear beside the faulty device location on the repair ticket.

The user may want to set up a log configuration to monitor the order that he tested devices in and those he did not test to help him analyze test results. A log of sequence flow and untested locations will then be configured.



```
ALL:SYST
MIN:SYST


Ready                                            9:03


 log_on  log_off  prt_log  show_log  advance
```

**Log Menu Screen**

Data that has been stored in a log file may be retrieved in manual or sequence mode through the log key which brings up this screen.

You may choose to start or stop the logging of data, print the file to a printer or show the file on the screen.

Up to eight log files can be open simultaneously, each using one of the five user defined log formats specified using conf_log key under system mode. Test results data gets logged to all the open log files simultaneously, however, all log commands, such as log_off and show_log, only operate on the currently active log file. The advance key is used to select the active log file. (Note: The active log file is shown in reverse highlight.)

## Hands-On Exercise

Run the Demo Sequence using a log file and upon completion, examine the subsequent log file.

## Sample Log File

```
-----------------------------------
Log_on:                   22:26 14 Jan 1988
Sequence: XT:CART

Start logging to: SN44063:SYST

U99 7400   ----   passed


U8 8288    ----   passed
>Jump to U4

U4   PAL20RA10 ----   unable to test
  RD test failed
Local change: C_SUM 54940

U4   PAL20RA10 ----   failed 40.0 ns
  Pin(s): 15 16 19 20
>Jump to U48

U48 8259   ----   failed 983.0 ms
  Incorrect frequency on pin(s): 5
Stop logging to: SY44063:SYST

Untested locations:
U1        U5
U7        U15
U6        U9
U10       U11
U16       U2
U13       U14
U22       U85
U96       U24
U97       U18
```

## Synopsis

To develop a sequence, the following steps should be taken:

1. Set up repeatable, comprehensive activity on the board to be tested. Use Reset whenever possible, either on the board under test or on another conrolling board. Alternatively, arrange to manually cycle the diagnostics and use the external trigger line to define the start of the test.
2. Enter new_seq Mode and verify that each device passes comparison testing. Make adjustments in the local and global parameters to achieve this.
3. Reorder the device locations to structure the troubleshooting, for example by signal flow. This can be done with keystrokes in new_seq Mode or the file listing can be rearranged with the screen editor. A PC editor is another possiblity.
4. Edit operator prompt messages into the file.
5. Verify that all devices pass on other good boards.
6. Document the sequence, reasons for parameter changes and the test setup details. Retain a master copy of the files on cartridge or disk.

# Appendix A
# Demo UUT Sequence Listings

## DEMO.SEQ Listing

```
File: DEMO.SEQ:CART     John Fluke Mfg. Inc. - FLUKE 900     08 Feb 1989,  07:39     Page    1

        LOC_FILE        DEMO
        SIZE            20

;******************************************************************************
;*                        SIGNON MEASSAGES                                    *
;******************************************************************************

        FUNCTION        SIGNON:
        FUNCTION        MSG_1:
        FUNCTION        MSG_2:

;******************************************************************************
;*                        20 PIN DEVICES                                      *
;******************************************************************************

        TEST            U15:        ; 74244
        TEST            U17:
        TEST            U37:
        TEST            U43:
        TEST            U48:

        TEST            U8:         ; 74245
        TEST            U23:
        TEST            U58:
        FUNCTION        ANL_U58:
        TEST            U66:

        TEST            U24:        ; 74373
        TEST            U25:
        FUNCTION        ANL_U25:
        TEST            U54:

        TEST            U91         ; 8284
        DISPLAY 3        "*****       CLOCK GENERATOR (MHz)      *****"
        DISPLAY 4        "*****         P2=2.38  P8=4.77         *****";

        TEST            U2          ; 8288
        DISPLAY         "*****          BUS CONTROLLER          *****";

;******************************************************************************
;*                        14 & 16 PIN DEVICES                                 *
;******************************************************************************

        TEST            U5 :        ; 7400
        TEST            U14:
        TEST            U42:

        TEST            U13:        ; 7402
        TEST            U87:

        TEST            U10:        ; 7404
        TEST            U20:
        TEST            U36:

        TEST            U4:         ; 7407

        TEST            U28:        ; 7408
        TEST            U47:
```

```
File: DEMO.SEQ:CART     John Fluke Mfg. Inc. - FLUKE 900     08 Feb 1989,  07:39    Page    2

          TEST             U6:        ; 7410

          TEST             U86:       ; 7420
          TEST             U21:

          TEST             U81:       ; 7427

          TEST             U41:       ; 7432

          TEST             U35:       ; 7451

          TEST             U11:       ; 7474
          TEST             U19:
          TEST             U27:
          TEST             U34:
          TEST             U82:

          TEST             U44:       ; 74125

          TEST             U39:       ; 74138
          TEST             U46:
          TEST             U50:
          TEST             U52:

          TEST             U22:       ; 74139

          TEST             U33:       ; 74157

          TEST             U40:       ; 74158
          TEST             U45:

          TEST             U3:        ; 74175
          TEST             U12:
          TEST             U29:

          TEST             U70:       ; 74280

          TEST             U26:       ; 74670

          TEST             U57        ; 41256 DRAM
          FUNCTION         RAM256:

          TEST             U61
          FUNCTION         RAM256:

          TEST             U65
          FUNCTION         RAM256:

          TEST             U69
          FUNCTION         RAM256:

          TEST             U74
          FUNCTION         RAM256:

          TEST             U78
          FUNCTION         RAM256:

```

```
File: DEMO.SEQ:CART    John Fluke Mfg. Inc. - FLUKE 900     08 Feb 1989, 07:39    Page    3

        TEST          U80
        FUNCTION      RAM256:

        TEST          U85
        FUNCTION      RAM256:

        TEST          U90
        FUNCTION      RAM256:

;*****************************************************************************
;*                            24 & 28 PIN DEVICES                            *
;*****************************************************************************

        TEST          U38:       ; 2764

        TEST          U71        ; 8253
        DISPLAY       "*****        Interval Timer          *****":

        TEST          U62        ; 8259
        DISPLAY       "*****      Interupt Controller       *****":

        TEST          U16A       ; 8088  PROCESSOR - 1st Pass
        DISPLAY 3     "***** U16 tested in 2 passes. Pass 1 *****"
        DISPLAY 4     "Line Up Pin 1 of clip with pin 1 of Chip":

        TEST          U16B       ; 8088 PROCESSOR - 2nd Pass
        DISPLAY 3     "***** U16 tested in 2 passes. Pass 2 *****"
        DISPLAY 4     "Line Up Clip pin 14 with Pin 20  of Chip":

;*****************************************************************************
;*                          FAULT INSERTION BOARD CHIPS                      *
;*****************************************************************************

        TEST          FU1        ; DEMO1 PAL - PAL22P10
        DISPLAY       "*****            PAL22P10            *****":

        TEST          FU2        ; DEMO2 PAL - PAL22P10
        DISPLAY       "*****            PAL22P10            *****":

        TEST          FU3        ; DEMO3 PAL - PAL22V10
        DISPLAY       "*****            PAL22V10            *****":

        END:


;*****************************************************************************
;*                                  FUNCTIONS                                *
;*****************************************************************************

RAM256  THRSLD        1800
        TRIGGER       ON P3=LH P4=LL P5=HH P6=HH P7=HH P9=HH P10=HH P11=HH P12=HH P13=HH P15=LL
        DISPLAY 3     "***** 256K DRAM - TRIGGER Set ON 1st *****"
        DISPLAY 4     "***** Memory Location WRITE/READ cycle****"
        END_FUNCTION

SIGNON  DISPLAY 1     "******************************************"
        DISPLAY 2     "***   Welcome to the FLUKE 900 Demo    ***"
        DISPLAY 3     "*** Press <NEXT> for more instructions ***"
```

```
File: DEMO.SEQ:CART   John Fluke Mfg. Inc. - FLUKE 900     08 Feb 1989,  07:39   Page 4

          DISPLAY 4      "******************************************"
          END_FUNCTION

 MSG_1    DISPLAY 1      "***  The  UUT is a  Turbo  PC-XT Main ***"
          DISPLAY 2      "***  Board  with Fault Insertion Keys ***"
          DISPLAY 3      "***  F1 thru F8.The Red Button clears ***"
          DISPLAY 4      "***  Inserted Faults & Resets the UUT ***"
          END_FUNCTION

 MSG_2    DISPLAY 1      "******************************************"
          DISPLAY 2      "***   Attach both GND Leads, EXT and   ***"
          DISPLAY 3      "***   Reset  Leads to Demo UUT posts   ***"
          DISPLAY 4      "******************************************"
          END_FUNCTION

 S8288    IF NO_TRIGGER_WORD2 THEN  DISPLAY 1 "*****   CPU HALT - CHECK RESET LINE  *****"
          IF P2<4.77MHz + P2>4.78MHz THEN DISPLAY 2 "*****   CLK Error - CHECK OSCILLATOR *****"
          END_FUNCTION

 ANL_U58 IF FAILED * P9=L THEN DISPLAY 2 "*****       DATA BUS D0 Stuck Lo      *****"
          IF FAILED * P9=A * P11=A THEN DISPLAY 2 "*****       DATA BUS D0 LOADED      *****"
          END_FUNCTION

 ANL_U25 IF (FAILED + FAILED_TO_SYNC) *  P12=L THEN DISPLAY 2 "*****       ADDRESS A0 STUCK Lo    *****"
          END_FUNCTION
```

## DEMO.LOC Listing

```
File: DEMO.LOC:CART     John Fluke Mfg. Inc. - FLUKE 900     08 Feb 1989,  07:41     Page    1

            NAME             U25
            LOAD             74373
            GATE             ON E=H:

            NAME             U24
            LOAD             74373
            GATE             ON E=H:

            NAME             U54
            LOAD             74373
            GATE             ON E=H:

            NAME             U23
            LOAD             74245:

            NAME             U15
            LOAD             74244:

            NAME             U8
            LOAD             74245:

            NAME             U58
            LOAD             74245
            F_MASK           60:

            NAME             U66
            LOAD             74245
            GATE             ON E=H:

            NAME             U17
            LOAD             74244
            F_MASK           60:

            NAME             U37
            LOAD             74244
            GATE             ON E=H
            F_MASK           40:

            NAME             U43
            LOAD             74244
            GATE             ON E=H:

            NAME             U48
            LOAD             74244
            GATE             ON E=H:

            NAME             U91
            LOAD             8284
            T_TIME           100
            ACTIVITY         P2=2.386MHz 1% P8=4.773MHz 1% :

            NAME             U2
            LOAD             8288
            F_MASK           40
            RESET            ON NEG INT OFS=-1 DUR=100
            TRIGGER          ON P3=X1 P18=X1 P19=X1 E=LH
            ACTIVITY         P2=4.773MHz 1%:
```

```
File: DEMO.LOC:CART     John Fluke Mfg. Inc. - FLUKE 900      08 Feb 1989,  07:41     Page   2

          NAME        U26
          LOAD        74670:

          NAME        U70
          LOAD        74280
          GATE        ON E=H:

          NAME        U81
          LOAD        7427:

          NAME        U86
          LOAD        7420:

          NAME        U21
          LOAD        7420:

          NAME        U3
          LOAD        74175:

          NAME        U12
          LOAD        74175:

          NAME        U29
          LOAD        74175:

          NAME        U82
          LOAD        7474
          F_MASK      50:

          NAME        U11
          LOAD        7474:

          NAME        U19
          LOAD        7474:

          NAME        U27
          LOAD        7474:

          NAME        U34
          LOAD        7474:

          NAME        U13
          LOAD        7402
          F_MASK      40:

          NAME        U87
          LOAD        7402
          F_MASK      40:

          NAME        U4
          LOAD        7407
          F_MASK      40:

          NAME        U10
          LOAD        7404
          F_MASK      100:

          NAME        U20
```

```
File: DEMO.LOC:CART    John Fluke Mfg. Inc. - FLUKE 900    08 Feb 1989, 07:41    Page  3
        LOAD        7404:

        NAME        U36
        LOAD        7404
        RESET       ON NEG INT OFS = -1 DUR = 100:

        NAME        U5
        LOAD        7400
        F_MASK      120: ; PIN 11 LOADED

        NAME        U14
        LOAD        7400:

        NAME        U42
        LOAD        7400:

        NAME        U33
        LOAD        74157:

        NAME        U39
        LOAD        74138
        F_MASK      40:

        NAME        U50
        LOAD        74138:

        NAME        U46
        LOAD        74138:

        NAME        U52
        LOAD        74138:

        NAME        U22
        LOAD        74139:

        NAME        U44
        LOAD        74125:

        NAME        U6
        LOAD        7410:

        NAME        U28
        LOAD        7408:

        NAME        U47
        LOAD        7408:

        NAME        U40
        LOAD        74158
        GATE        ON E=H:

        NAME        U45
        LOAD        74158
        GATE        ON E=H:

        NAME        U41
        LOAD        7432:

```

```
File: DEMO.LOC:CART      John Fluke Mfg. Inc. - FLUKE 900     08 Feb 1989,  07:41      Page  4

          NAME          U35
          LOAD          7451:

          NAME          U62
          LOAD          8259
          IGNORE        16
          F_MASK        40:

          NAME          U38
          LOAD          2764
          C_SUM         45145
          RD_DRV        HIGH
          F_MASK        50:

          NAME          U71
          LOAD          8253
          RESET         ON NEG INT OFS = -1 DUR = 100:

          NAME          U57
          LOAD          41256:

          NAME          U61
          LOAD          41256:

          NAME          U65
          LOAD          41256:

          NAME          U69
          LOAD          41256:

          NAME          U74
          LOAD          41256:

          NAME          U78
          LOAD          41256:

          NAME          U80
          LOAD          41256 :

          NAME          U85
          LOAD          41256:

          NAME          U90
          LOAD          41256:

          NAME          FU1
          LOAD          DEMO1
          S_TIME        ON 3000
          RESET         ON NEG INT OFS = -5 DUR = 100:

          NAME          FU2
          LOAD          DEMO2
          S_TIME        ON 3000:

          NAME          FU3
          LOAD          DEMO3
          S_TIME        ON 9000:

```

```
File: DEMO.LOC:CART     John Fluke Mfg. Inc. - FLUKE 900     08 Feb 1989,  07:41     Page   5

 233         NAME          U16A
 234         LOAD          8088_1
 235         T_TIME        100:
 236
 237         NAME          U16B
 238         LOAD          8088_2
 239         T_TIME        100:
 240
 241         NAME          DEFAULT
 242         GATE          OFF
 243         TRIGGER       ON E=LH
 244         T_TIME        4000:
```

## DEMO.LIB Listing

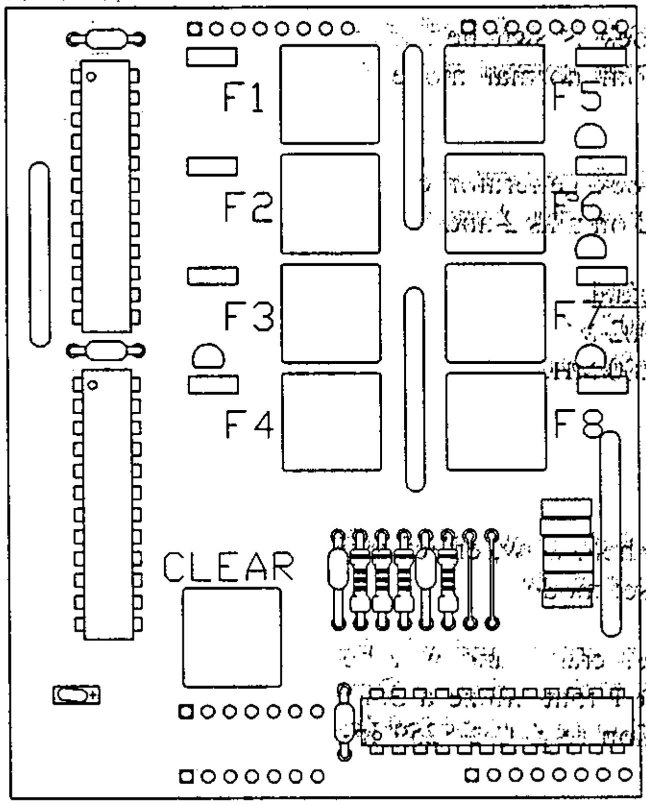
```
File: DEMO.LIB:CART     John Fluke Mfg. Inc. - FLUKE 900     08 Feb 1989,  07:40     Page   1
        NAME    74670
        SIZE    16
        SYNC_COND <1 P12=L*P13=L*P14=H>*<1 P12=L*P13=H*P14=L>*<1 P12=L*P13=H*P14=H>
        S_TIME  ON:


        NAME    8284
        SIZE    18
        RDT_ENABLE  OFF
        IGNORE  1-18:

        NAME    8088_1
        SIZE    28
        CLIP_CHK        OFF
        RDT_ENABLE      OFF
        IGNORE          1-28
        ACTIVITY        P1=L P2=A P3=A P4=A P5=A P6=A P7=A P8=A P9=A P10=A P11=A P12=A P13=A P14=A P28=H
                        P27=A P26=A P25=A P24=A P23=A P21=L P20=A P16=A P15=A:
        :
        NAME    8088_2  ;PROCESSOR
        SIZE    28
        CLIP_CHK        OFF
        RDT_ENABLE      OFF
        IGNORE          1-28
        ACTIVITY        P1=A P2=A P3=A P4=A P5=A P6=A P7=A P8=A P9=A P10=A P13=4.773MHz 1% P14=LP28=A
                        P27=L P26=A P22=A P21=A P20=A P19=A P18=A P16=A P15=A:


        NAME DEMO1
        SIZE 24
        SYNC_VECT 1 < 1 L P2=D23 P3=D21 P5=D17 P4=D15 P8=D8 P9=D9 P11=D11 >
        S_TIME ON
        SYNC_RND
        RDTEST VECTORS
;                1111111 11122222
;       12345678 90123456 78901234
        01111001 100LOHHZ HLZZHZHH ;1
        01111001 101LOHHZ HLZZHZHH ;2
        00111001 101LOLHZ HLZHHZLH ;3
        01111101 101LOHHZ HLZLHZLH ;4
        01111001 101LOHHZ HLZHHZLH ;5
        01111100 101LOHHZ HLZZHZHH ;6
        01111001 101LOHHZ HLZZHZHH ;7
        01011001 101LOLHZ HLHZLZHH ;8
        01111011 101LOHHZ HLLZLZHH ;9
        01111001 001LOHHZ HLZZHZHH ;10
        01111011 101LOHHZ HLZZHZHH ;11
        01101001 101LOLLL HLZZHZHH ;12
        01111001 101LOHLL HLZZHZHH ;13
        01111001 100LOHHZ HLZZHZHH ;14
        01111001 101LOHHZ HLZZHZHH ;15
        01110001 101LOLHZ LHZZHZHH ;16
        01111001 101LOHHZ LHZZHZHH ;17
        01111000 101LOHHZ HLZZHZHH ;18
        01111001 101LOHHZ HLZZHZHH ;19
        END_VECTORS:



        NAME DEMO2
```

```
File: DEMO.LIB:CART    John Fluke Mfg. Inc. - FLUKE 900    08 Feb 1989, 07:40    Page 2

          SIZE 24
          SYNC_VECT 1 < 1 L P5=D22 P6=D21 P8=D19 P9=D20 P4=D4 P7=D7 P10=D10 >
          SYNC_RND
          S_TIME ON
          RDTEST VECTORS
 ;                 1111111 11122222
 ;        12345678 90123456 78901234
          00001111 110LOZLL ZLHHHHHH  ;1
          00011111 110LOZLL ZLHHHHHH  ;2
          00010111 110LOZHL ZLHHHLLH  ;3
          00011111 110LOZHL ZLHHHLHH  ;4
          00001111 110LOZLL ZLHHHHHH  ;5
          00011111 110LOZLL ZLHHHHHH  ;6
          00011011 110LOZLL ZHHHLHLH  ;7
          00011111 110LOZLL ZHHHLHHH  ;8
          00011101 110LOZLL ZLHHHHHH  ;9
          00011111 110LOZLL ZLHHHHHH  ;10
          00011111 010LOZLH ZLHLHHLH  ;11
          00011111 110LOZLH ZLHLHHHH  ;12
          00011111 100LOZLL ZLHHHHHH  ;13
          00011111 110LOZLL ZLHHHHHH  ;14
          00011110 110LOZLL LLLHHHLH  ;15
          00011111 110LOZLL LLLHHHHH  ;16
          00001111 110LOZLL ZLHHHHHH  ;17
          00011111 110LOZLL ZLHHHHHH  ;18
          END_VECTORS:

          NAME DEMO3
          SIZE 24
          SYNC_VECT 1 <257 L P1=C >
          SYNC_PINS 1
          S_TIME ON
          RDTEST VECTORS
 ;                 1111111 11122222
 ;        12345678 90123456 78901234
          00000000 000L1LLL LLLLLZZH    ; 0
          00000000 000LOLLL LLLLLZZH
          10000000 000LOLLL LLLHLZZH    ; 1
          00000000 000LOLLL LLLHLZZH
          10000000 000LOLLL LLLLHZZH    ; 2
          00000000 000LOLLL LLLLHZZH
          10000000 000LOLLL LLLHHZZH    ; 3
          00000000 000LOLLL LLLHHZZH
          10000000 000LOLLH LLLLLZZH    ; 4
          00000000 000LOLLH LLLLLZZH
          10000000 000LOLLH LLLHLZZH    ; 5
          00000000 000LOLLH LLLHLZZH
 ;                 1110111 11122222
 ;        12345678 90123456 78901234
          10000000 000LOLLH LLLLHZZH    ; 6
          00000000 000LOLLH LLLLHZZH
          10000000 000LOLLH LLLHHZZH    ; 7
          00000000 000LOLLH LLLHHZZH
          10000000 000LOLLL HLLLLZZH    ; 8
          00000000 000LOLLL HLLLLZZH
          10000000 000LOLLL HLLHLZZH    ; 9
          00000000 000LOLLL HLLHLZZH
          10000000 000LOLLL HLLLHZZH    ; 10
```

```
File: DEMO.LIB:CART     John Fluke Mfg. Inc.   FLUKE 900     08 Feb 1989,  07:40     Page 3

  117        00000000 000LOLLL HLLLHZZH
  118        10000000 000LOLLL HLLHHZZH   ; 11
  119        00000000 000LOLLL HLLHHZZH
  120        10000000 000LOLLH HLLLLZZH   ; 12
  121        00000000 000LOLLH HLLLLZZH
  122        10000000 000LOLLH HLLHLZZH   ; 13
  123        00000000 000LOLLH HLLHLZZH
  124 ;               1110111 11122222
  125 ;      12345678 90123456 78901234
  126        10000000 000LOLLH HLLLHZZH   ; 14
  127        00000000 000LOLLH HLLLHZZH
  128        10000000 000LOLLH HLLHHZZH   ; 15
  129        00000000 000LOLLH HLLHHZZH
  130        10000000 000LOLHL LLLLLZZH   ; 16
  131        00000000 000LOLHL LLLLLZZH
  132        10000000 000LOLHL LLLHLZZH   ; 17
  133        00000000 000LOLHL LLLHLZZH
  134        10000000 000LOLHL LLLLHZZH   ; 18
  135        00000000 000LOLHL LLLLHZZH
  136        10000000 000LOLHL LLLHHZZH   ; 19
  137        00000000 000LOLHL LLLHHZZH
  138        10000000 000LOLHH LLLLLZZH   ; 20
  139        00000000 000LOLHH LLLLLZZH
  140        10000000 000LOLHH LLLHLZZH   ; 21
  141        00000000 000LOLHH LLLHLZZH
  142 ;               1110111 11122222
  143 ;      12345678 90123456 78901234
  144        10000000 000LOLHH LLLLHZZH   ; 22
  145        00000000 000LOLHH LLLLHZZH
  146        10000000 000LOLHH LLLHHZZH   ; 23
  147        00000000 000LOLHH LLLHHZZH
  148        10000000 000LOLHL HLLLLZZH   ; 24
  149        00000000 000LOLHL HLLLLZZH
  150        10000000 000LOLHL HLLHLZZH   ; 25
  151        00000000 000LOLHL HLLHLZZH
  152        10000000 000LOLHL HLLLHZZH   ; 26
  153        00000000 000LOLHL HLLLHZZH
  154        10000000 000LOLHL HLLHHZZH   ; 27
  155        00000000 000LOLHL HLLHHZZH
  156        10000000 000LOLHH HLLLLZZH   ; 28
  157        00000000 000LOLHH HLLLLZZH
  158        10000000 000LOLHH HLLHLZZH   ; 29
  159        00000000 000LOLHH HLLHLZZH
  160 ;               1110111 11122222
  161 ;      12345678 90123456 78901234
  162        10000000 000LOLHH HLLLHZZH   ; 20
  163        00000000 000LOLHH HLLLHZZH
  164        10000000 000LOLHH HLLHHZZH   ; 31
  165        00000000 000LOLHH HLLHHZZH
  166        10000000 000LOLLL LHLLLZZH   ; 32
  167        00000000 000LOLLL LHLLLZZH
  168        10000000 000LOLLL LHLHLZZH   ; 33
  169        00000000 000LOLLL LHLHLZZH
  170        10000000 000LOLLL LHLLHZZH   ; 34
  171        00000000 000LOLLL LHLLHZZH
  172        10000000 000LOLLL LHLHHZZH   ; 35
  173        00000000 000LOLLL LHLHHZZH
  174        10000000 000LOLLH LHLLLZZH   ; 36
```

```
File: DEMO.LIB:CART     John Fluke Mfg. Inc. - FLUKE 900     08 Feb 1989,  07:40     Page    4

          END_VECTORS:
```

# Appendix B
# Demo UUT Fault Switches



**Demo UUT Fault Insertion Board**

The following is a list of the various Faults and their effect on the UUT:

## Fault No. 1

*Description:* The UUT system clock is switched from 4.773 MHz to 10.00 MHz (i.e. from normal mode to Turbo mode).

*Result:* When testing U91, the clock generator chip a **Condition Failure** will be detected on pins 2 and 8.

| Pin No. | E-xpected | Actual |
|---|---|---|
| 2 | 2.38 MHz | 5.002 MHz |
| 8 | 4.773 MHz | 10.00 MHz |

## Fault No. 2

*Description:* The UUT RST line is held LOW, effectively keeping the UUT in permanent reset mode.

*Result:* This will cause a kernel crash, and will be detected on several ICs on the UUT that utilize a Start Trigger or a Trigger for intialization (e.g. U62 8259 Interrupt Controller).

The fault can be detected as a **Condition Failure** on the Bus Controller chip U2 8288. The condition that will be not available is the Trigger, which detects the first bus cycle and the Status pins (S0, S1 and S2) will all be High, indicating that the 8088 CPU is in Passive (Reset) state.

## Fault No. 3

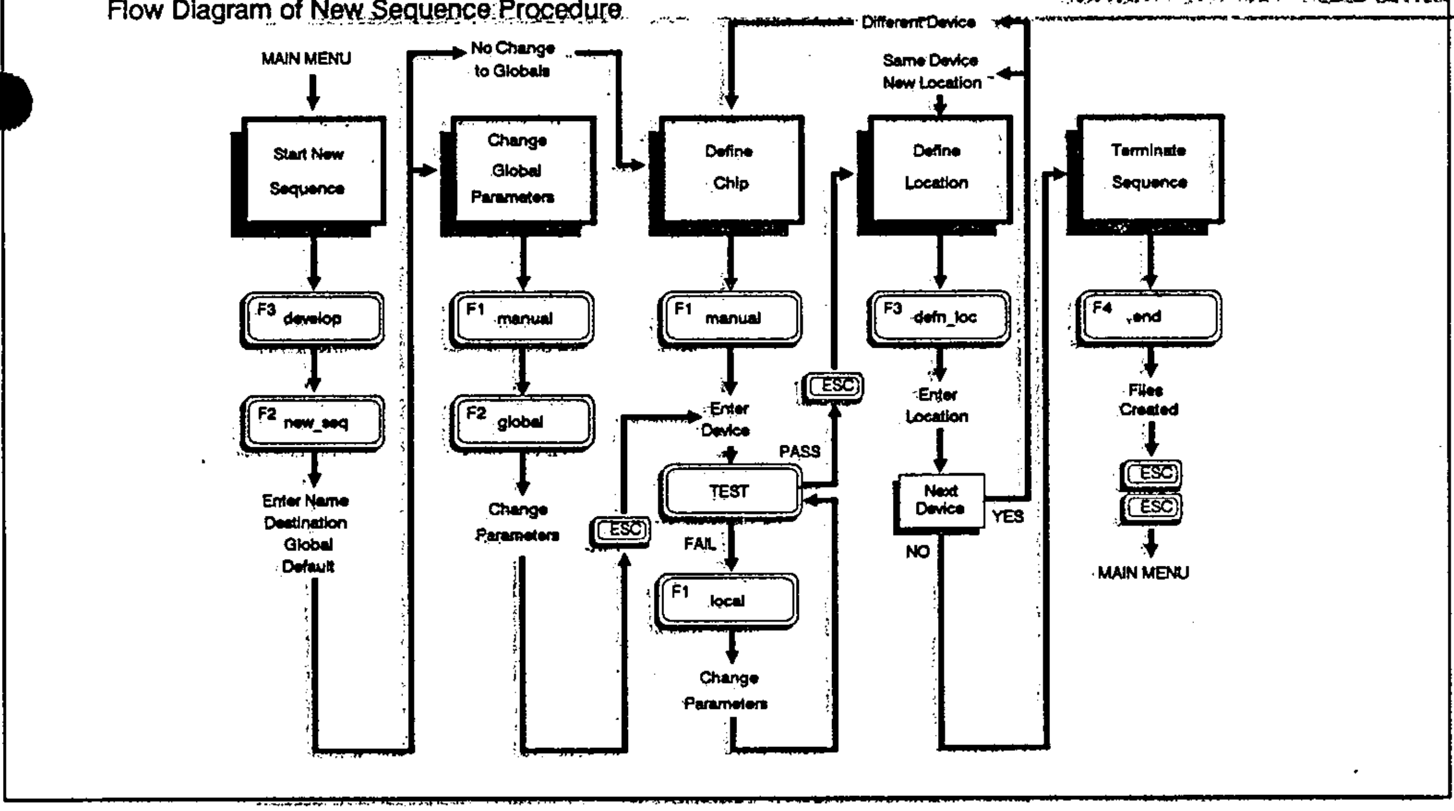
*Description:* The UUT Data Bus Bit 0 (D0) is held Low. This simulates the effect of a bus short.

*Result:* The UUT will crash, and the Fault will be detected on U58 Pin 9.

The Fault can be ignored via the pin_def ignore pin command, in order to allow further testing of the chip and bus, if so desired.

## Demo UUT Fault Switches

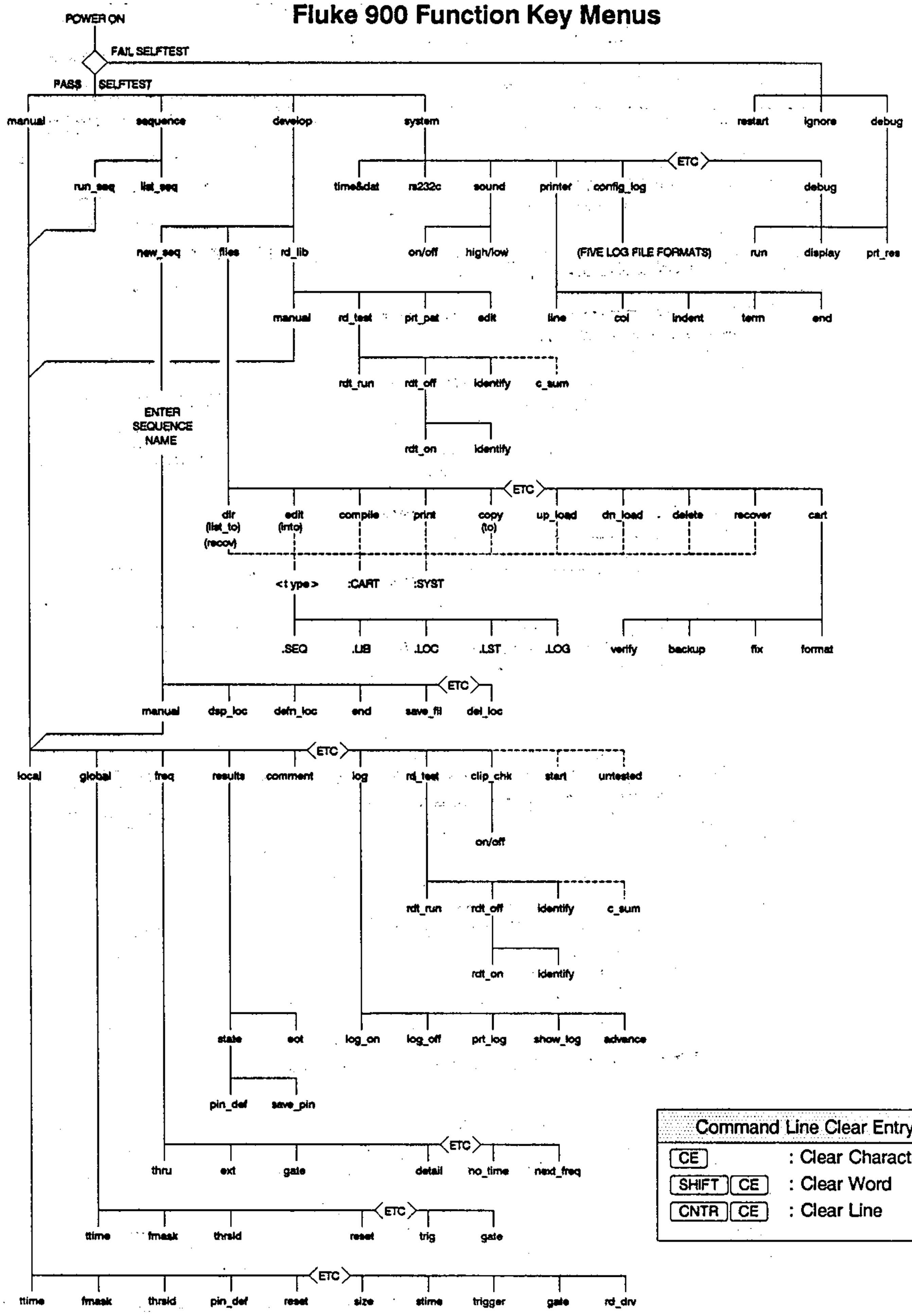
Flow Diagram of New Sequence Procedure



## Glossary

| | |
|---|---|
| sum | a ROM checksum or the PAL output signature from a standard input stimulus |
| Conditions | signal status (H is high, L is low, A is active, F is frequency) |
| defn_loc | "define location" key for specifying a device U number |
| del_loc | "delete location" key for removing a device and its U number |
| DRC | Dynamic Reference Comparison |
| dsp_loc | "display locations" key for listing devices in a new_seq file |
| DUT | Device Under Test |
| EoT | "End of Test" key for displaying device state at fault capture |
| mask | "fault mask" key permitting setting of comparison resolution |
| global | a key to set parameters for all devices that follow |
| LIB | extension on a source file name meaning Library |
| LOC | extension on a source file name meaning Location |
| local | a key to set parameters for a single device |
| log | a key to enable recording of test results, usually to a file |
| new_seq | "new sequence" key to create a Sequence with simple keystrokes |
| nsq | extension on a file meaning a New Sequence created in new_seq |
| NSTD | nonstandard power pins (i.e. Vcc is not highest pin number) |
| PAL | Programmable Array Logic |
| PE | Performance Envelope: the f_mask, thrsld setting for comparison |
| pin_def | "pin definition" key for ignoring or setting conditions on pins |
| RD | Reference Device |
| rd_test | a key to test or identify an RD in the ZIF socket |
| SEQ | extension on a source file name meaning Sequence |
| Sequence | a stored troubleshooting procedure with parameters and prompts |
| STD | standard power pins (i.e. Vcc is highest pin number, Gnd is half) |
| s_time | "synchronizing time" key to put RD and DUT into the same state |
| save_fil | "save file" key to store a volatile New Sequence on cartridge |
| thrsld | "threshold" key for setting logic 1 |
| t_time | "test time" key for setting the duration of comparison |
| ZIF | Zero Insertion Force socket |

# Fluke 900 Function Key Menus



| Command Line Clear Entry | |
|---|---|
| CE | : Clear Character |
| SHIFT CE | : Clear Word |
| CNTR CE | : Clear Line |