Introduction to In-Circuit Testing

Introduction
to
In-Circuit Testing

© GenRad, Inc. 1984
Concord, Massachusetts, U.S.A. 01742
January 1984

The following are trademarks of GenRad, Inc.

SCRATCHPROBING
GRnet
BUSBUST

The following are trademarks of Digital Equipment Corporation, Maynard, Mass.

DEC
RSX
VT

Contents

Foreword iv

Chapter 1
Introduction 3

Chapter 2
Techniques for In-Circuit Testing 17

Chapter 3
A Look at an In-Circuit Tester 53

Chapter 4
Using the Tester 89

Glossary 121

Foreword

The purpose of this book is to provide the background information necessary to understand the purpose, theory, and operation of in-circuit testers. While primarily aimed at those who will develop test programs for the system, it should prove useful to anyone responsible for specifying, approving, managing, or supporting such a system.

The book is styled for easy reading. Emphasis is placed on general concepts and overall flow rather than on specific details of operation.

The book's four chapters have the following purposes:

Chapter 1 — **Introduction** describes how the in-circuit tester fits into the manufacturing process and what types of faults it can detect.

Chapter 2 — **Techniques For In-Circuit Testing** presents the critical concepts of access via the bed-of-nails fixture and isolation of the components on the board by guarding for analog and back-driving for digital components.

Chapter 3 — **A Look at an In-Circuit Tester** describes the hardware and software components of an in-circuit tester and the functions these components perform.

Chapter 4 — **Using the Tester** outlines the step-by-step process used by a programmer to develop a test program and by an operator to test boards.

For easy reference, a **Glossary** of terms associated with in-circuit testing is included at the back of this document.

Chapter 1

Introduction

So, tell me about in-circuit testing.

Before getting into in-circuit testing, let's review some important aspects of printed circuit (pc) board manufacturing and testing.

As you know, the design and assembly of pc boards follow certain basic steps:

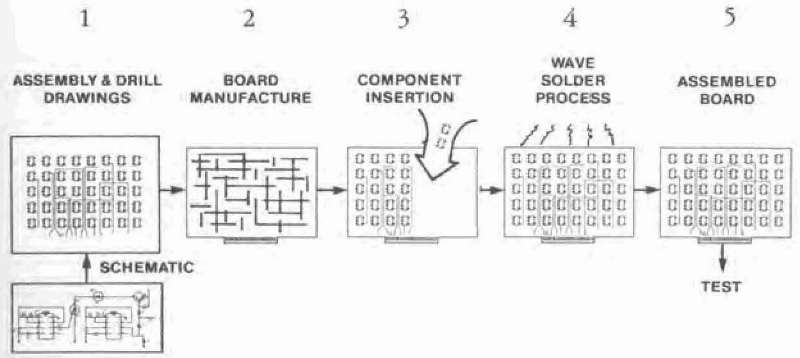
1. First, assembly and drill drawings of the board are developed from the schematic diagram. These drawings show where each component will be placed, where each track (wiring connection) will be etched, where each component mounting hole will be located, etc.

2. Then, holes for the component leads are drilled in a blank board and tracks are etched on the surface of the board to connect the components together.

3. Next, the components are mounted on the board, either by hand or by automatic insertion equipment.

4. Finally, the component leads are soldered to the tracks, usually by means of a wave-solder machine.

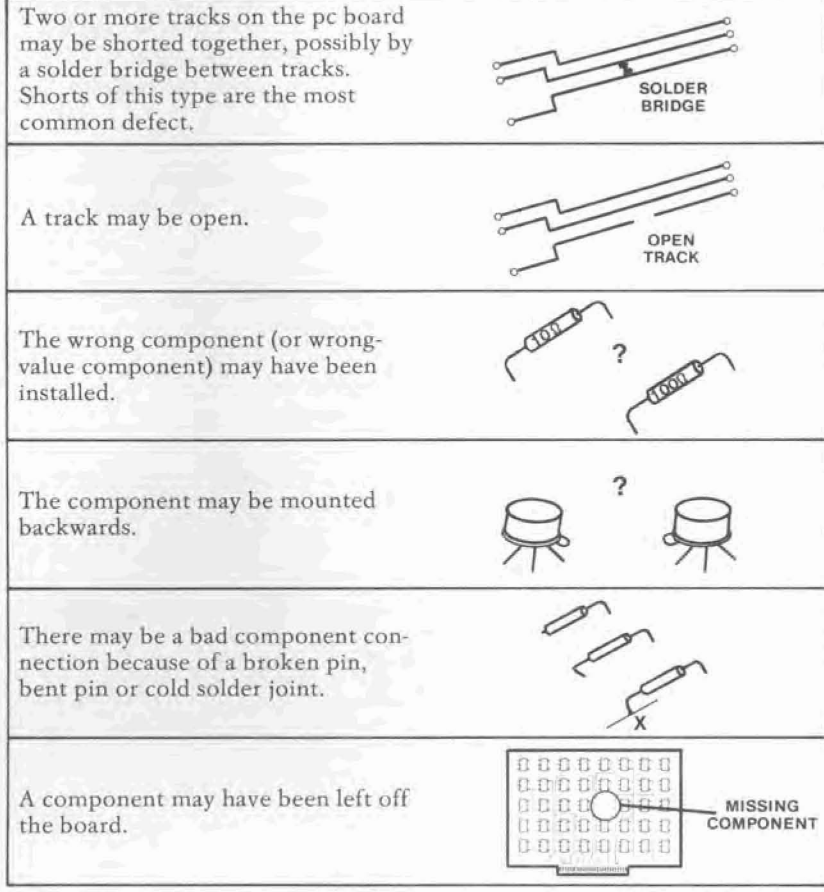
5. Once assembled, the pc board is then tested.



Unfortunately, these steps are not always performed flawlessly.

Shown below are some of the more common failures found in newly assembled pc boards.

Typical manufacturing faults



Fault	Illustration
Two or more tracks on the pc board may be shorted together, possibly by a solder bridge between tracks. Shorts of this type are the most common defect.	SOLDER BRIDGE
A track may be open.	OPEN TRACK
The wrong component (or wrong-value component) may have been installed.	?
The component may be mounted backwards.	?
There may be a bad component connection because of a broken pin, bent pin or cold solder joint.	X
A component may have been left off the board.	MISSING COMPONENT

Note that, even if the bare pc board had no failures and if you tested every component to make sure that each one was good, you could still introduce problems while assembling the board.

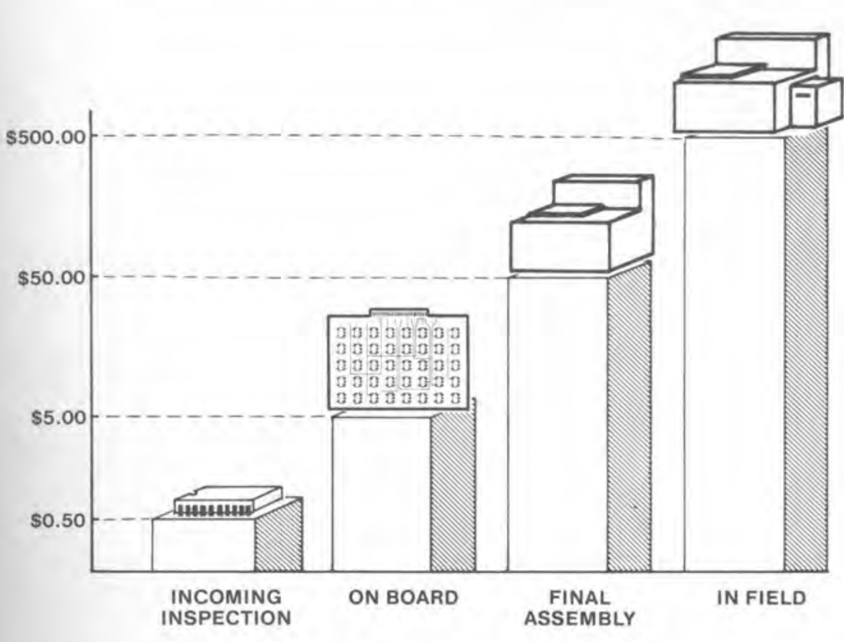
Now, consider the cost of finding and fixing failures.

To find a defective component
at incoming inspection might cost about $0.50

If a defect slips past this stage, the cost
of finding a bad part once it has been mounted
on the pc board goes up to about $5.00

The cost of finding and replacing that same
part once the pc board is passed on to final
(system) assembly now becomes $50.00

Finally, if the system finds its way into
the field with that defective part and
appears as a warranty repair, the expected
cost approaches $500.00



So who needs a fancy machine to test pc boards? A few test instruments and some common sense should do the job.

Maybe, but don't underestimate the job.

First, you have to understand how all the circuits on the board work before you can figure out how to test them. Once you've gotten over this hurdle, you have to write test procedures specifying exactly how the board will be tested. Remember, you want to test all the circuits; otherwise, you can't be sure the board will work properly under all normal operating conditions.

Once you've written the test procedures, you have to assemble all the necessary test equipment: oscilloscopes, voltage sources, current sources, meters, and so on.

Manual testing

Now, connect all this equipment to the appropriate points on the pc board. Some of those points may be hard to reach. If you have many boards to test, you may want to build a fixture to simplify connecting the test leads.

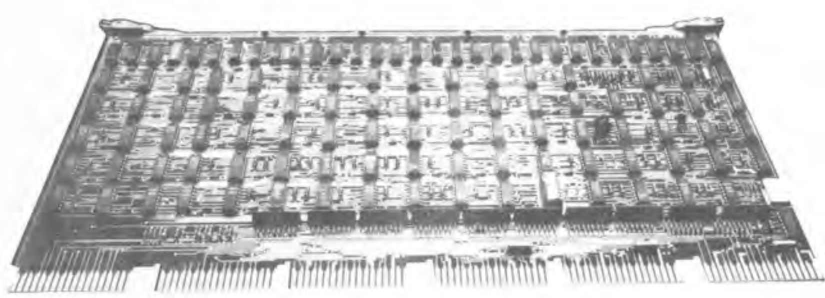
Finally, turn on the equipment, apply known inputs to the circuit, check the outputs and determine if the circuit is operating properly. And, oh yes, if the circuit is not operating properly, find the defective part and replace it.

I guess that could be quite a bit of work.

Don't go away. There's more.

Now, you have to turn off the equipment, re-connect the test leads to other points on the board, turn the equipment back on, drive some new inputs, check some new outputs, etc. You get the idea.

What if the pc board has hundreds or even thousands of circuit connections (nodes)?



And how will you test the more complex components like microprocessors and other LSI (Large-Scale Integration) chips?

Oh, and don't forget that all these components are interconnected on the pc board. So, if an output isn't what it should be, how will you know which component is causing the failure?

Could be quite a job, couldn't it? It seems that a test system with computer-controlled test instruments and the proper software could go a long way towards simplifying this process.

OK! OK! So tell me about automatic test equipment.

Thought you'd never ask.

The term automatic test equipment (ATE) applies to many forms of programmable, computer-controlled test instruments and systems. This equipment is used in many stages of the manufacturing process, including:

- Component manufacturing
- Quality assurance
- Incoming inspection
- PC board testing
- System testing
- Field service

This book concentrates on one specific category of automatic test equipment:

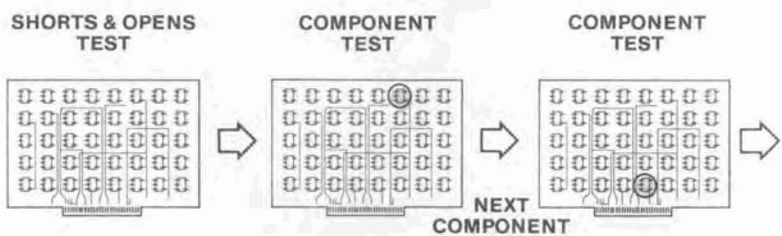
IN-CIRCUIT TESTERS

These testers are typically used to test fully assembled pc boards in a manufacturing environment.

What is an in-circuit tester?

It's a tester that tests each component on a pc board, one at a time. And it does this while the component is "in-circuit," that is, while it's connected to other components on the board.

A divide-and-conquer approach is taken. First, the tester checks the loaded pc board for unwanted shorts and opens. Then, it isolates and tests each separate component on the board, one at a time. The techniques used to do this are described in Chapter 2.



If all these individual tests are successful, the board is considered a good board.

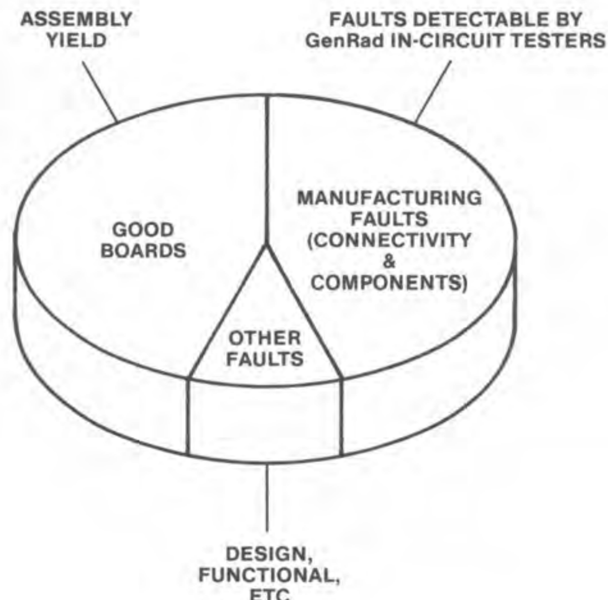
If one or more tests fail, the board is declared bad, and the tester reports the cause of the failure.

Since each component is tested separately, pinpointing the cause of a failure is a relatively easy job for an in-circuit tester.

What type of faults can in-circuit testers detect?

GenRad in-circuit testers can detect all the manufacturing faults described a few pages back. As shown in the chart below, these account for a very large portion of all the faults that can be found on a new board.

Note: Since actual percentages for assembly yield (good boards) and fault distribution depend on the complexity of the board, the quality of the manufacturing process, etc, specific number values are not given on this chart.



GenRad's in-circuit testers

All of GenRad's 227x Board Test Workstations are **in-circuit testers.**

For the sake of brevity, we'll refer to them simply as **testers** throughout this book.

The tester performs all the manual procedures that were described a few pages back, namely:

- Writes test procedures
- Connects the test equipment to the board
- Turns the equipment on and sets it up
- Applies known input signals and checks outputs
- Determines if the circuit is OK
- Locates defective components when the circuit is not OK

and last, but not least,

- Repeats this for all the circuitry on the board

GenRad's in-circuit testers (cont)

As shown below, GenRad has several different models of in-circuit testers (2271, 2272, 2275), and all of them use the same basic testing techniques.

To learn about these techniques, let's go on to Chapter 2.

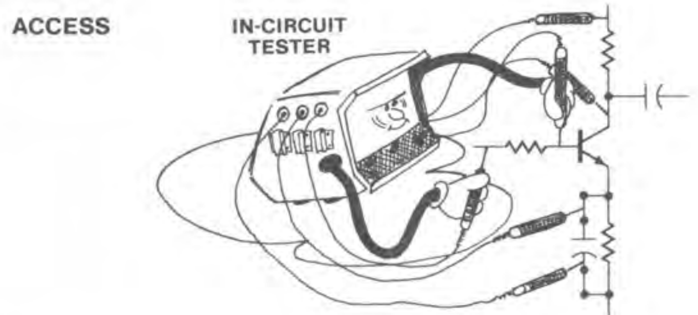
GenRad's 227X Family of Testers

Chapter 2

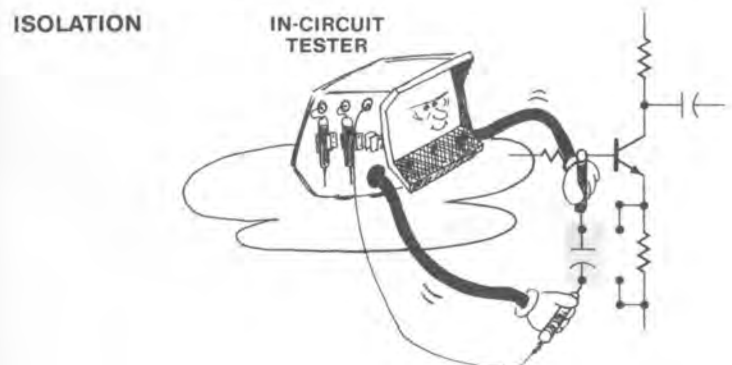
Techniques for In-Circuit Testing

To test individual components on a fully assembled pc board, an in-circuit tester must be capable of 2 things:

- It must have **access** to all the circuit nodes on the board. Obviously, to test each component individually, the tester must be able to connect test instruments to each pin of each component.



- Also, it must be able to **isolate** each component-under-test from surrounding components. Since components are interconnected on the board, some special isolation techniques are needed to prevent the component-under-test from being affected by other components



Now, let's look at how the tester performs these 2 functions. We'll start with the **accessing** of circuit nodes.

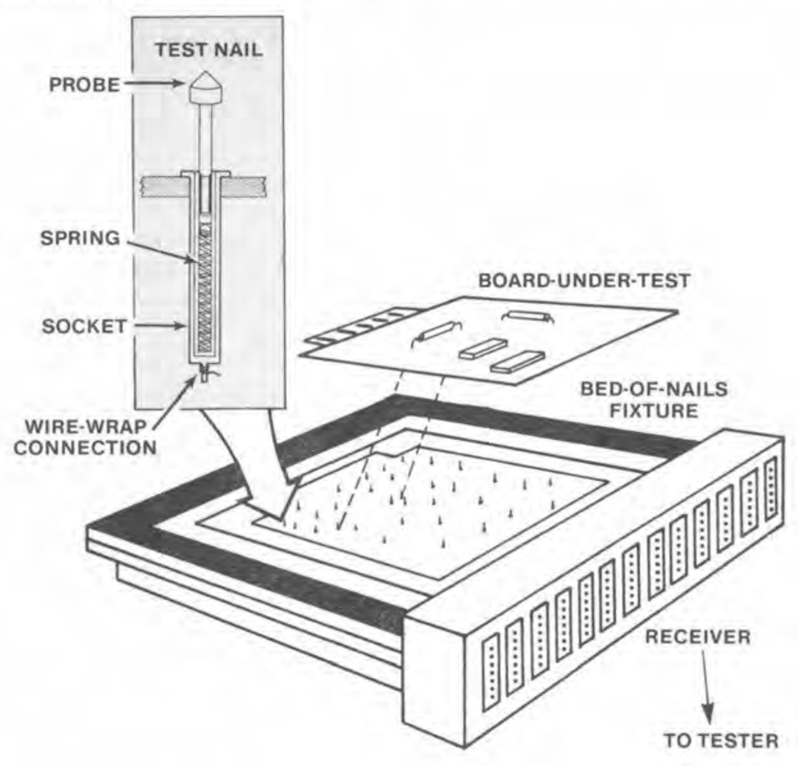
Accessing the circuit nodes

To access the circuit nodes on a pc board, the tester uses a special test fixture called (appropriately enough) a **bed-of-nails.**

The "nails" on this fixture are small spring-loaded probes that touch the tracks and component leads on a board during a test.

The nails are mounted in sockets properly located on the fixture so that each nail lines up with a circuit node on the board. Connections to the tester are made by wiring the other end of the sockets to connectors that plug into the system.

This physical interface between the tester and the fixture is called the **receiver.**

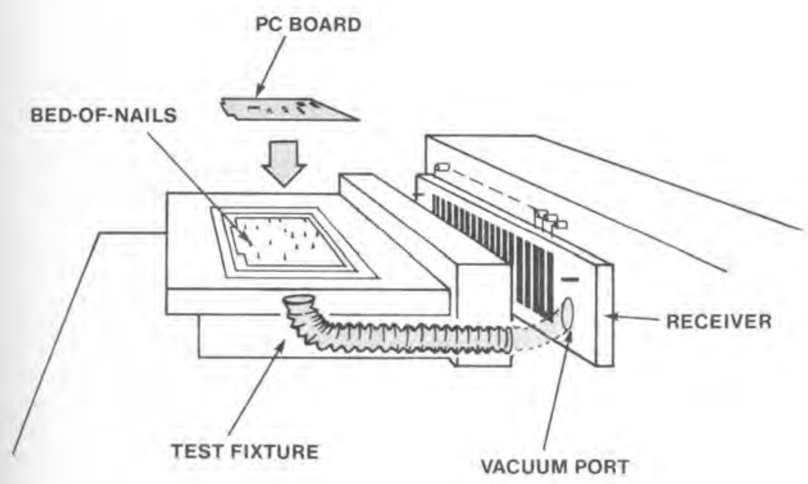


Bed-of-nails fixture

Each custom-built bed-of-nails fixture typically has hundreds of test nails properly positioned to come in contact with the solder pads and tracks on the bottom side of the pc board.

Prior to testing, the board is placed component-side up on the fixture and a vacuum is activated, pulling the board down onto the test nails.

The tester now has direct contact with every circuit node on the pc board.



Note: Chapter 4 describes how these bed-of-nails fixtures are made.

Some definitions

To prevent possible confusion later on, let's digress a moment and define (nail down?) some of the terms associated with the interface between the tester and the test fixture.

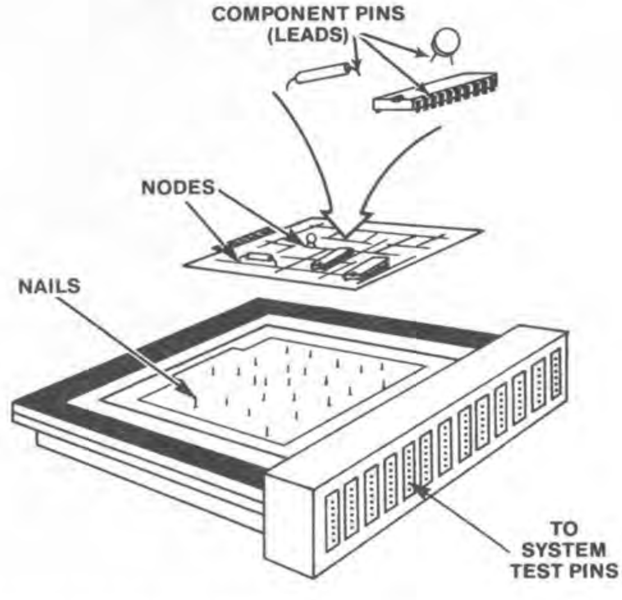
NAIL (or **TEST NAIL**)
Refers to the spring-loaded probes in the test fixture, which come in contact with the pc board under test.

NODE A circuit point (or track) on the pc board itself.

PIN When the tester is being referred to, the term **(test) pins** signifies the test points or test connections available for testing a board. For example, a tester may be said to have a total of 640 **test pins.**

When a component is being referred to, the term **(component) pins** signifies the leads attached to that component, for example, pin 4 of IC35.

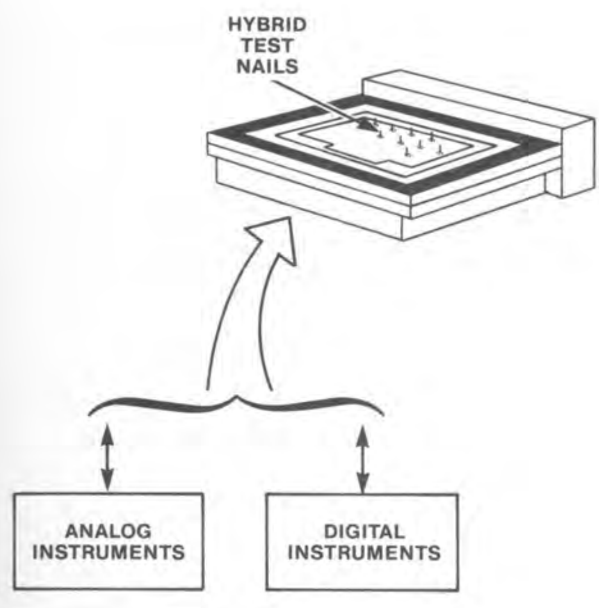
LEAD Usually refers to the connection attached to a circuit component (e.g., a capacitor has 2 leads, an IC has 16 leads, etc).



Now, back to accessing circuit nodes

Remember, the whole purpose of the bed-of-nails fixture is to connect the system test instruments to the circuit nodes.

Since all testing is either analog or digital, every test nail in most of GenRad's in-circuit testers can be connected to either the analog or digital test instruments, as directed by the test program. Because of this dual capability, these nails are referred to as **hybrid** nails. The obvious advantage of hybrid nails is that either analog or digital testing (or both) can be performed at any circuit node on the board, as the test program dictates.



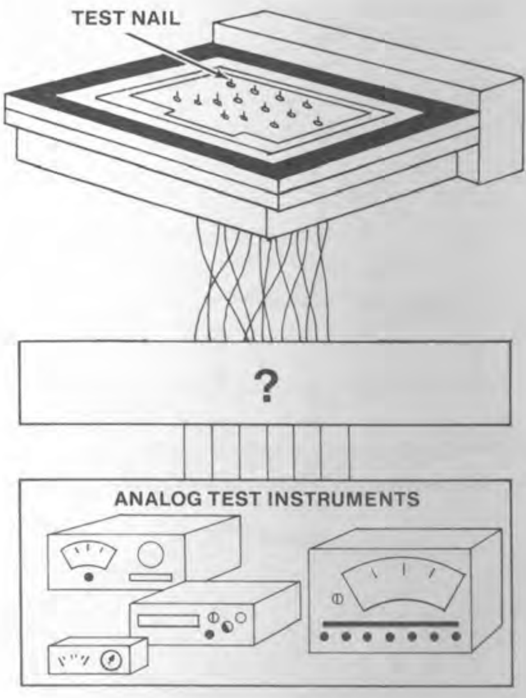
Analog test instruments

Let's start by taking a look at what analog test instruments are used by the tester, and at how these instruments are routed to the circuit nodes on the board.

To perform analog testing, the tester uses the following instruments:

DC Current Source
DC Voltage Source
DC Voltmeter
DC Ammeter (Current Meter)
AC Impedance Measurement Module

Now, the question is, how can the tester connect any one of these instruments to any one of hundreds of circuit nodes on the board?

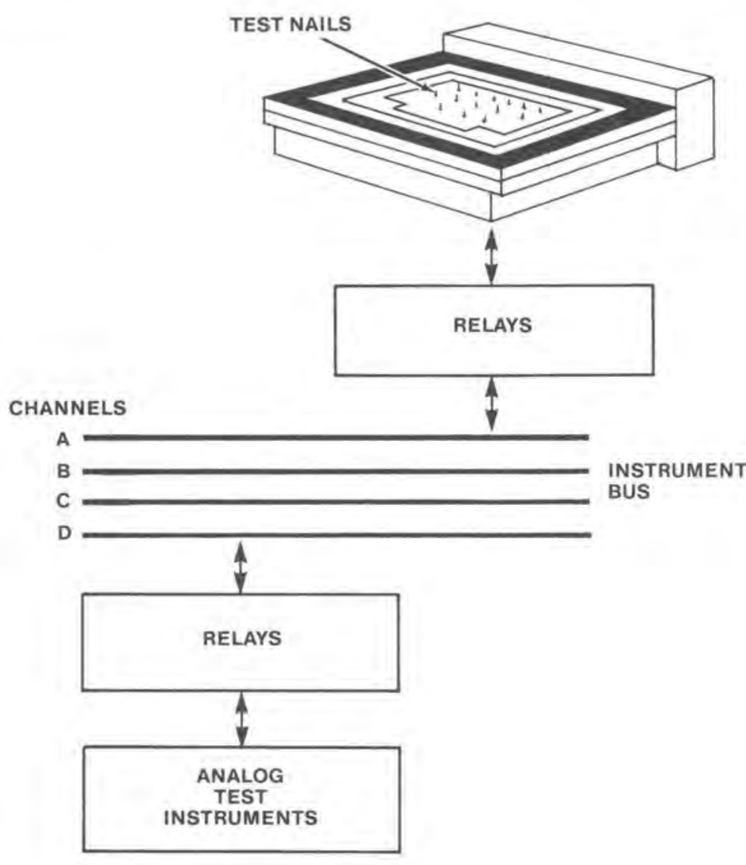


It's all done under computer control and directed by a test program uniquely developed by the tester for that board.

Connecting the analog test instruments to the nodes

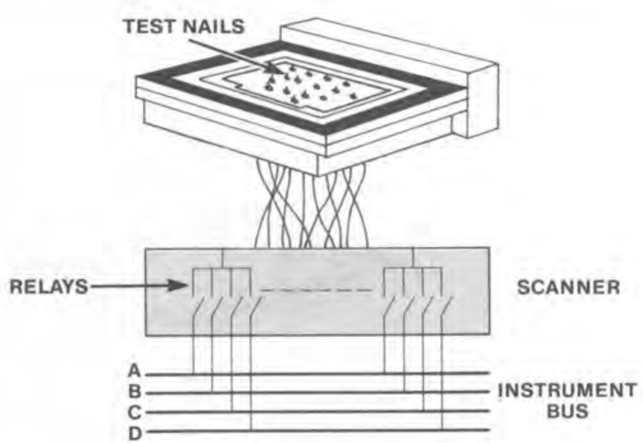
The actual connections between test instruments and test nails are made by program-controllable relays, i.e., switches that can be opened or closed by program instructions. These relays route both the test instruments and the test nails to a common set of 4 lines (or channels) called an **instrument bus.**

The 4 instrument bus channels are identified by the letters A, B, C, and D.

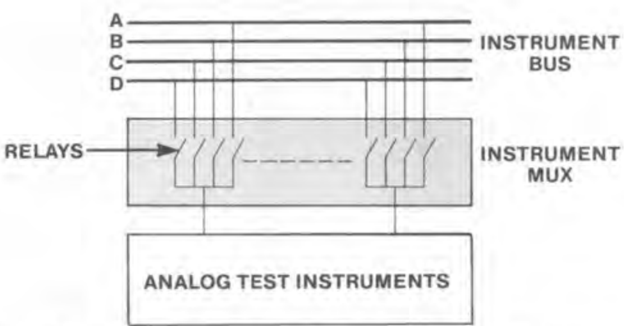


Connecting analog instruments (cont)

There are 2 sets of relays. One set, called the **SCANNER,** connects *any test nail* to any of the 4 instrument bus channels (A, B, C, D).



The other set of relays, called the **instrument multiplexer** or **MUX,** connects these same 4 channels to *any analog instrument.*



So, by controlling both the MUX and the SCANNER relays, the program can route any analog instrument through the instrument bus to any test nail.

Multiplexing is a way of sharing expensive test instruments among many test pins, making cost effective testing possible. As we'll see later, multiplexing is also used to connect the digital test circuitry to the test nails.

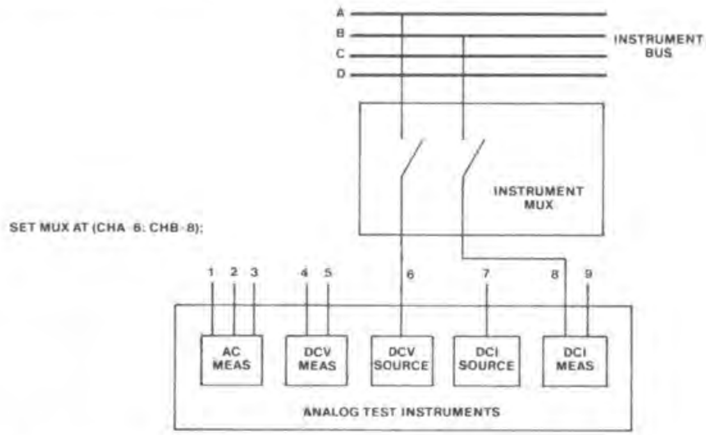
Controlling the MUX and SCANNER

The following example will illustrate how the system-developed test program controls the MUX and SCANNER relays.

The program statement

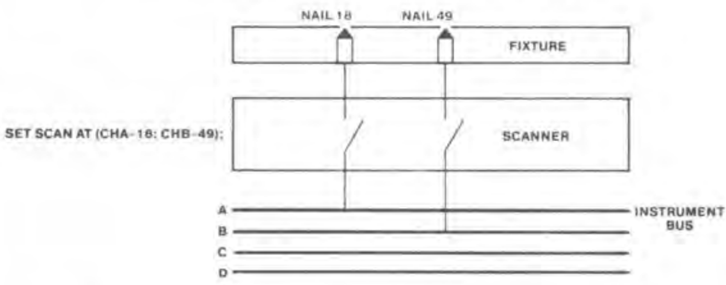
SET MUX AT (CHA=6: CHB=8);

closes the relays connecting pin 6 of the analog instrument unit to channel A and pin 8 of that unit to channel B. As shown in the diagram below, pin 6 of the analog instrument unit is the dc voltage source and pin 8 is the dc current meter.



Similarly, the statement

SET SCAN AT (CHA=18: CHB=49);



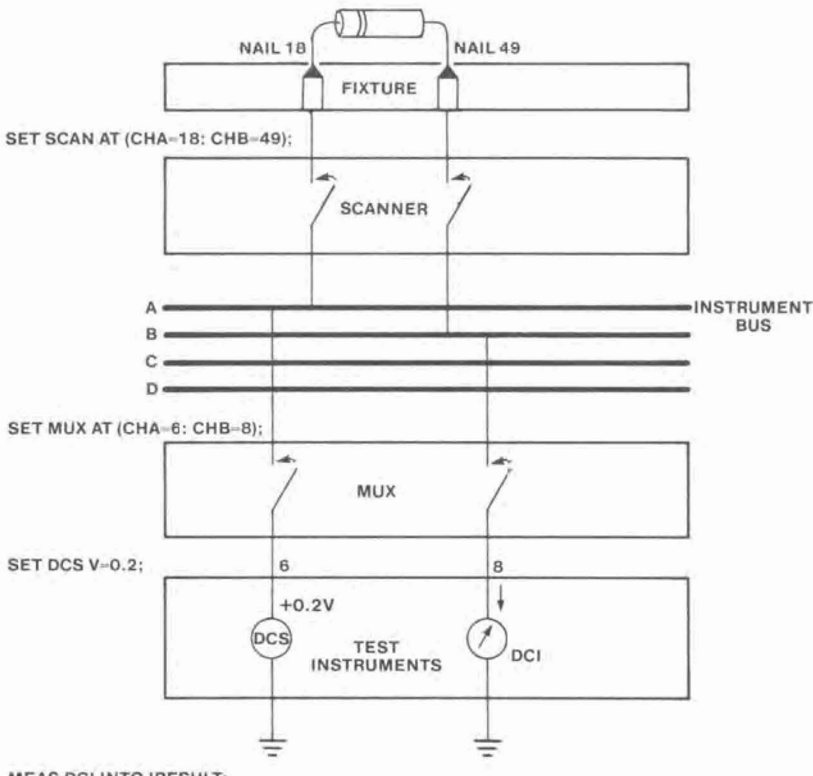
closes the relays connecting that same MUX channel A to test nail 18 and channel B to test nail 49.

Analog test example

To continue the example one step further, let's say that the pc board has a resistor connected between nails 18 and 49. And you want to apply 0.2 V across the resistor, measure the resulting current and store the value in a memory location (variable) called IRESULT.

The following instruction sequence would get the job done, and the diagram shows how.

```
SET MUX AT (CHA=6: CHB=8);
SET SCAN AT (CHA=18: CHB=49);
SET DCS V=0.2;
MEAS DCI INTO IRESULT;
```

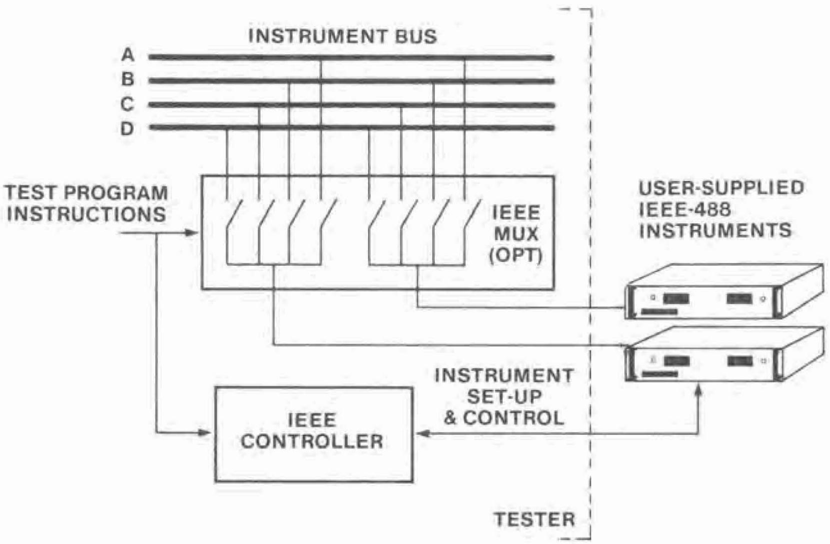


Can I use other test instruments that are not included in the tester?

Yes, provided that these instruments conform to a specification called **IEEE-488,** which defines a special digital interface for programmable instruments.

All of GenRad's in-circuit testers have an optional IEEE controller and multiplexer. In the 2275 tester, this IEEE option is contained on a single pc board.

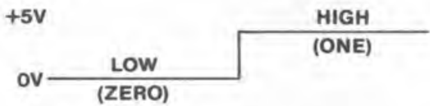
Under test program control, the MUX can connect external IEEE devices to the same instrument bus as the standard analog instruments, and the controller can handle the setting-up and operation of these instruments.



Now, how about the digital test instruments?

First, consider some of the typical features of digital components.

- Digital components deal with only 2 voltage levels: a LOW (ZERO) level and a HIGH (ONE) level. For example,

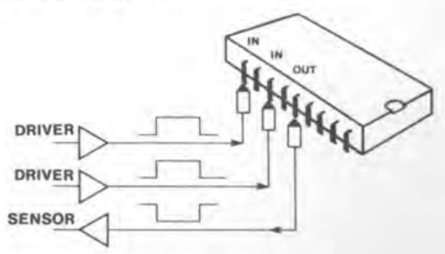


  The test program can define these 2 logic levels before digital testing begins and then simply refer to the 2 levels as HIGHs and LOWs throughout the digital test program.

- Digital ICs (Integrated Circuits) have many more pins than typical analog components such as resistors, capacitors, transistors, etc.

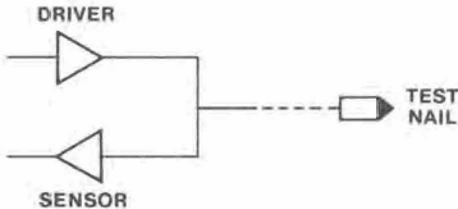
  It would be inefficient for the tester to keep switching analog source and measurement instruments from one IC pin to another during a test.

So, the tester has a set of digital **drivers** that it uses to drive the IC inputs to desired states and a set of digital **sensors** to check the logic levels at the IC outputs.



Driver/Sensors

These drivers and sensors (D/S, for short) always exist in pairs, so that the output of a driver and the input to a sensor are always tied together.



Drivers and sensors, however, are separately controllable by the program.

Therefore, when a D/S pair is used to force a logic input to an IC, the driver portion is enabled (connected) and its output forced to a specified state. At the same time, the sensor portion either can be enabled to sense that driver output or can be instructed to ignore it

Similarly, if the D/S pair is used to check a logic output from an IC, the driver is disconnected and the sensor is enabled.

Keywords, such as

IC — (Input Connect)
ID — (Input Disconnect)
OS— (Output Sense)
OI — (Output Ignore)

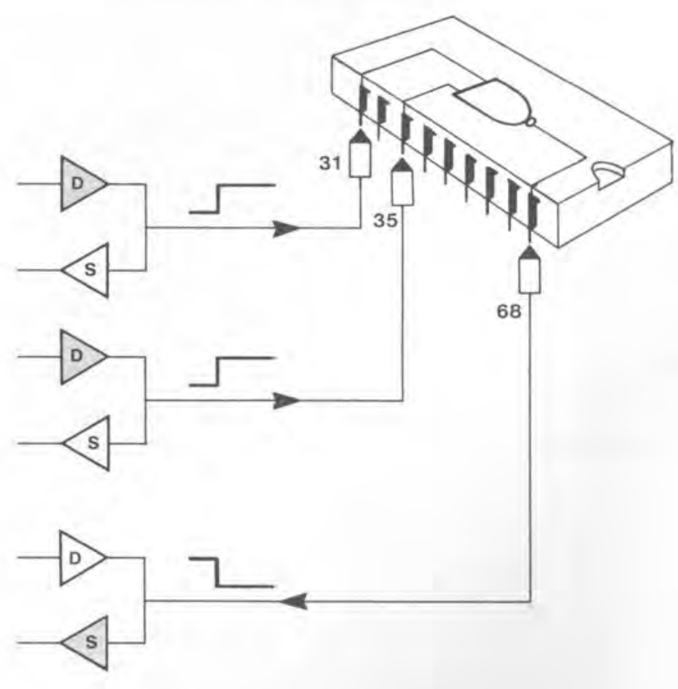
are used in the test language statements to control the driver/sensors (as shown on the next page).

Programming the driver/sensors

To illustrate how the driver/sensors are controlled by the program, let's look at a simple example.

In the NAND gate shown below, suppose you wanted to apply a HIGH level to both inputs while checking the output for a LOW level. You would use the following statements:

IC(31, 35)	Input Connect - enables (connects) the driver portion of the D/S associated with test nails 31 and 35.
IH(31, 35)	Input High - forces a HIGH level out of these 2 drivers and into the NAND gate.
OS(68)	Output Sense - enables the sensor portion of the D/S connected to nail 68.
OL(68)	Output Low - causes that sensor to check for a low output from the NAND gate.



The driver/sensor system

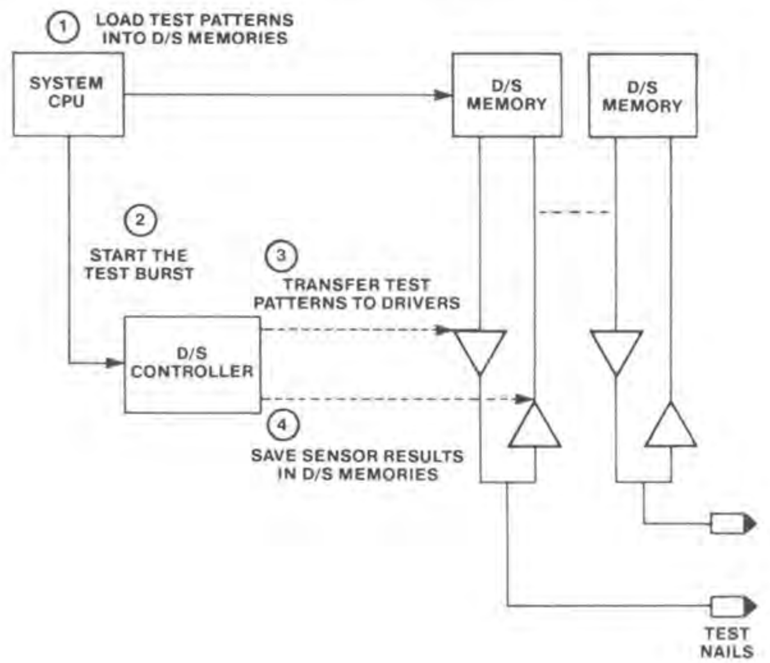
To be practical, digital tests must be conducted at a much higher speed than the system's computer (Central Processing Unit) can handle.

Therefore, before a digital IC is tested, the Central Processing Unit (CPU) loads a series of test patterns from the test program into a bank of memory cells attached to the driver/sensors (D/S). Each D/S has its own memory bank.

To start a test, the system CPU enables a special high-speed controller, which handles the transferring of these test patterns to the drivers and the storing of sensor results into the D/S memories.

This high-speed digital test procedure, illustrated below, is called a test BURST.

After the burst is complete the CPU transfers the results from the D/S memories back to the main memory for analysis.



One more thing about the driver/sensors

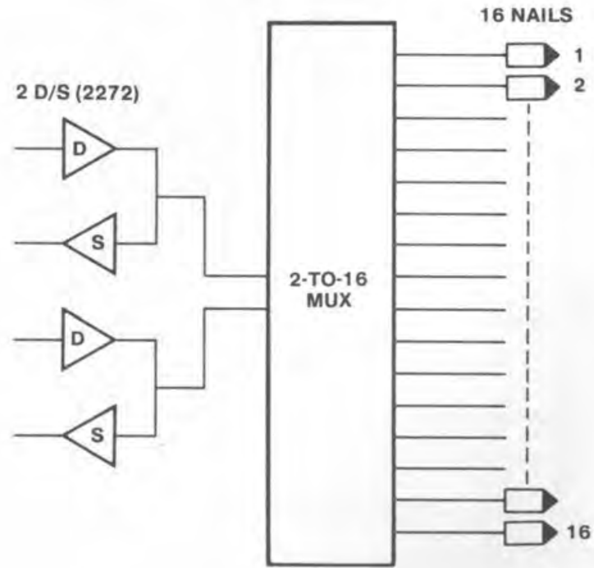
Testing large pc boards that have many digital ICs requires a test fixture with many digital test nails.

Luckily, not all of these nails are used at the same time. (All of them contact the board at the same time, of course, but the tester uses only a few of them during each test.)

In fact, during each test (or burst), the system uses only those nails in contact with the IC being tested.

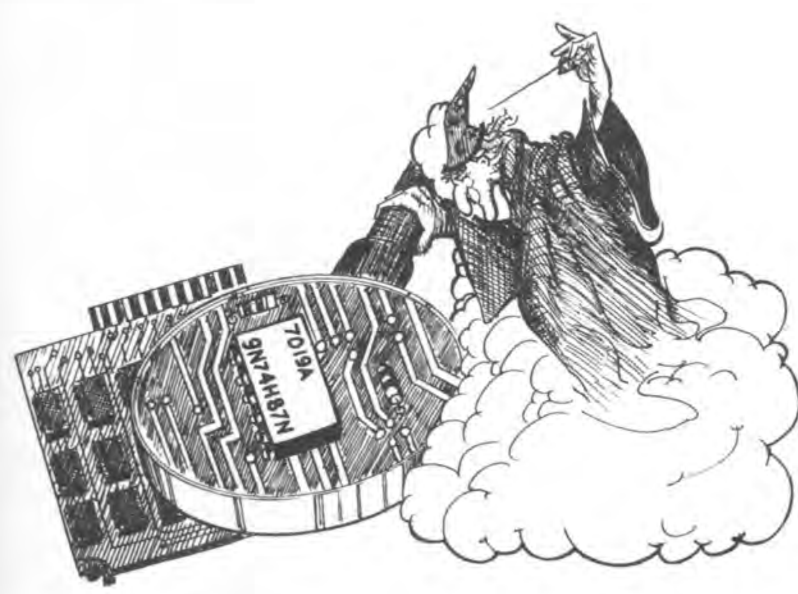
So, for economical reasons, each test nail does not usually have its own dedicated driver/sensor, but shares a few driver/sensors with a group of other nails. This technique is called **driver/sensor multiplexing.**

In the 2271 and 2275, 16 nails share 4 driver/sensors.
In the 2272 (shown below), 16 nails share 2 driver/sensors.



Isolating each component

Now that we've seen how the test instruments can access every circuit point on a board, let's see how the tester uses these instruments to isolate and test each component.



Isolation techniques used by the in-circuit tester differ depending on whether the component being tested is an analog device (resistor, capacitor, diode, etc) or digital device (gate, flip-flop, bus driver, etc).

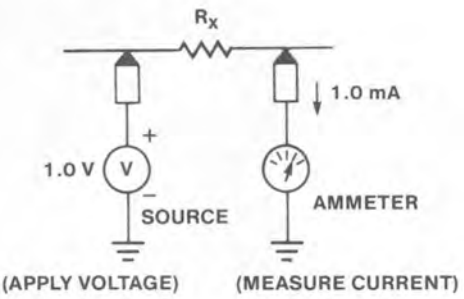
We'll start by looking at the isolation of analog components.

Isolating and testing analog components

Suppose you wanted to test an analog component, such as a resistor, that is not connected in a circuit.

You could apply a known voltage across that resistor, measure the resulting current, and calculate resistance by using Ohm's Law: R=V/I.

For example, if the applied voltage were 1.0 V and the measured current were 1.0 mA, the calculated resistor value would be

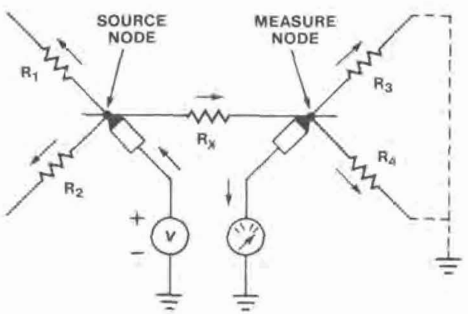


$$R_X = V/I = 1.0 / 0.001 = 1000 \text{ ohms}$$

This is called a **2-terminal measurement.**

Now, what happens when that same resistor is connected in a circuit?

There would very likely be some shunt paths around the resistor (R_X), which would divert some of that resistor current from flowing into the ammeter. (Resistors R_1, R_2, R_3 and R_4 represent these shunt paths.)

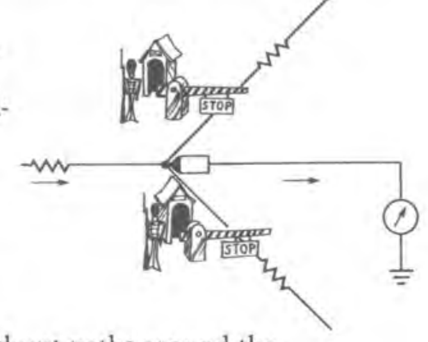


Instead of flowing directly to ground through the ammeter, the total current through R_X would now split up and flow through R_3 and R_4 as well as through the ammeter.

Depending on the resistance values of these shunt paths, the ammeter reading could be affected significantly and, therefore, the calculated value of R_X could be significantly in error.

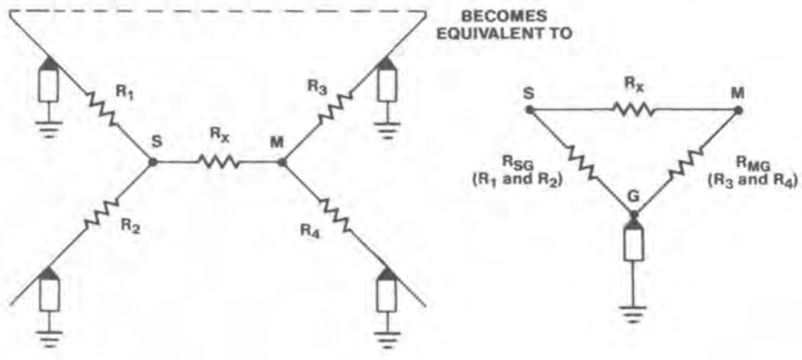
How can we "guard" against these unwanted shunt currents?

By using a technique known as **guarding**, which is the key to analog in-circuit testing. Guarding stops, or at least reduces significantly, the current flow through the shunt paths connected to the measure node.

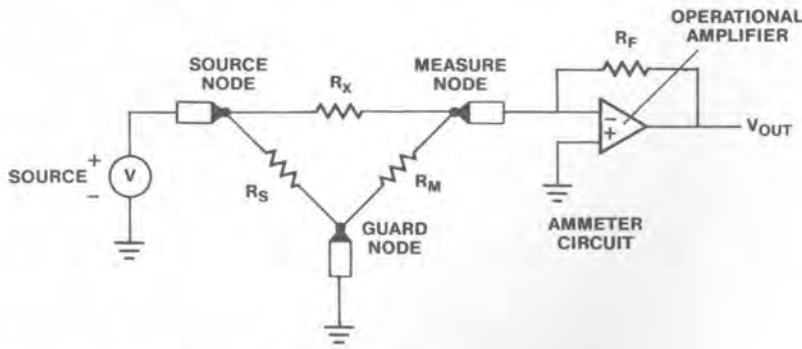


Guarding is accomplished by:

1. Temporarily connecting all shunt paths around the component-under-test to ground (the guard voltage), and



2. Using an operational amplifier (op amp) in the ammeter circuit.



This is called a **3-terminal measurement.**

How does this help?

First, let's consider a very important feature of the op amp circuit.

Because of the feedback resistor (R_F), the – input to the amplifier is forced to be very nearly equal to the + input. Since the + input is tied to ground, the – input is also very close to ground (virtual ground).

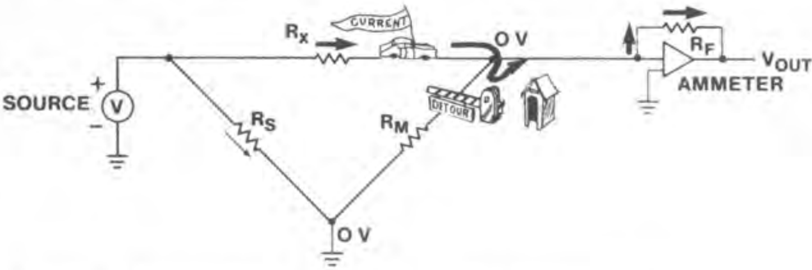
Now back to the guard circuit shown on the previous page.

In the ideal case:

The measure node is at 0 V (virtual ground) due to the op amp, and the guard node is also at 0 V because it is connected to ground by the tester.

Since there's no voltage across the measure and guard nodes, there's no current flow, right?

Voila! We've temporarily disabled these shunt paths and effectively isolated R_X from the rest of the circuit. All the current through R_X now flows into the ammeter circuit.



The tester knows the value of the feedback resistor (R_F) and can measure the op amp voltage V_{OUT}. Therefore, determining the current flow into the ammeter circuit and calculating R_X is a snap.

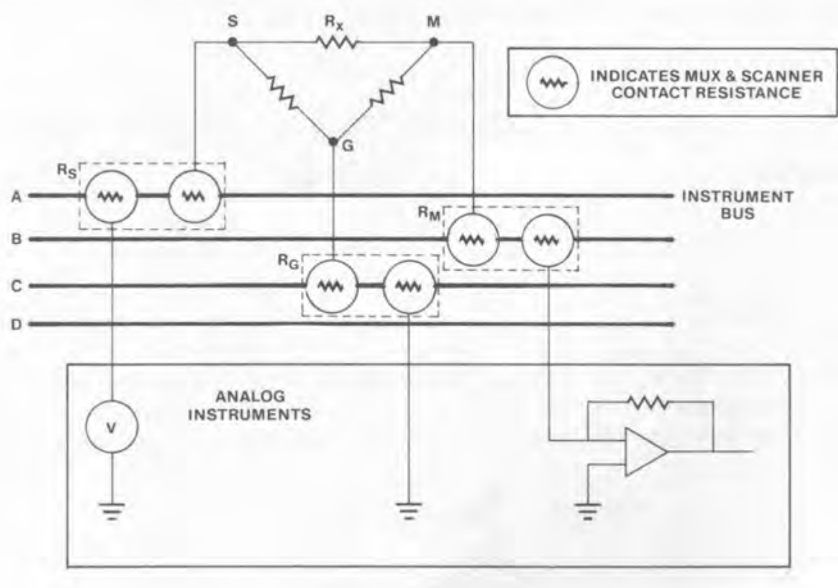
But, isn't there still a current flow from the source node to ground?

Yes, there could be.

But, if we ignore such things as cable resistance and switch contact resistance, this current has no effect on the measurement. After all, a known voltage is still being applied across R_X and the ammeter is measuring the resulting current.

In the real-world, however, such mundane things as contact resistance cannot always be ignored and can have an effect on the measurement.

Remember, in the guard circuit just described, we assumed that the guard terminal was at 0 V because it was tied directly to ground. In reality, however, the guard terminal is connected to ground through relay contacts in the instrument multiplexer and scanner. Similarly, the source and measure terminals are connected to the pc board through relay contacts.



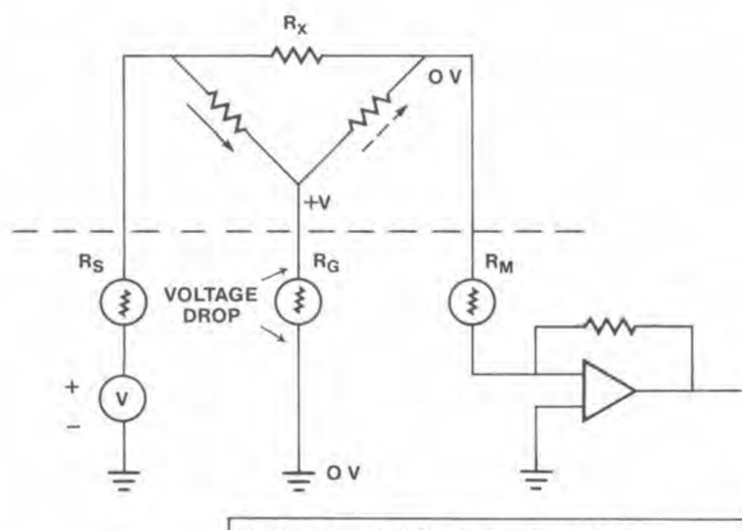
What effect does this have on the measurement?

Consider,

Any current from the source node to ground flows through this contact resistance.

Current through this resistance causes a voltage drop, right? So the guard node is not really at 0 V.

This means that there is a voltage difference between the measure node and guard node, and some of the shunt current that we tried to guard against is still trickling through.



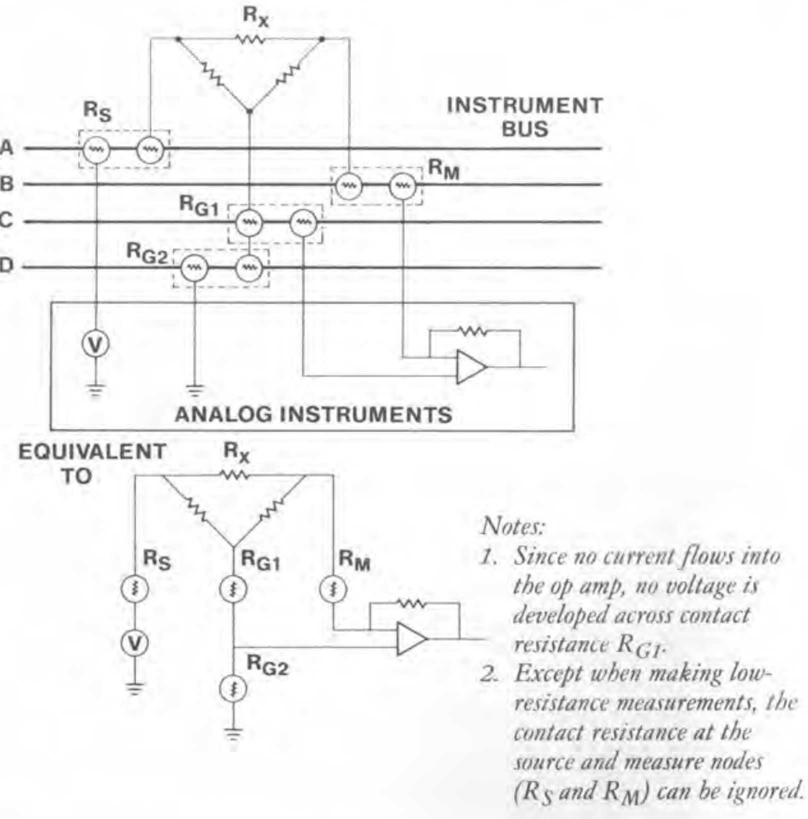
Oh, no!

Take heart. All is not lost.

By using a fourth terminal (in addition to the source, measure and guard terminals), this source of error can be greatly reduced.

Rather than connecting the ammeter reference (the + input of the op amp) directly to ground, this fourth terminal connects that reference closer to the actual guard nodes on the board.

With this set-up, any error voltage developed across contact resistance R_{G2} now appears at both the guard and measure nodes. This reduces the voltage difference between these 2 nodes and thus reduces that bothersome shunt current at the measure node.

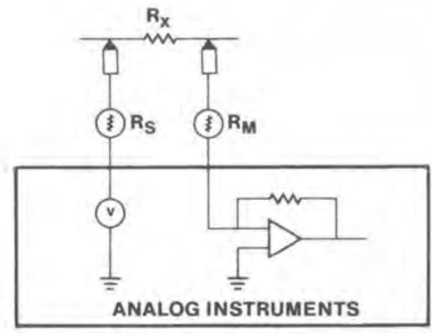


Notes:

1. *Since no current flows into the op amp, no voltage is developed across contact resistance R_{G1}.*
2. *Except when making low-resistance measurements, the contact resistance at the source and measure nodes (R_S and R_M) can be ignored.*

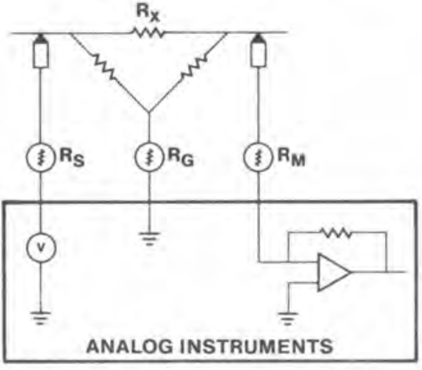
This is called a **4-terminal measurement.**

Review of 2- 3- and 4-terminal measurements



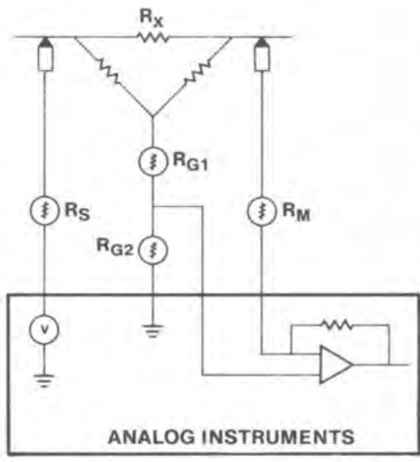
2-Terminal

Ignores the effects of surrounding components.



3-Terminal

Takes into account surrounding components, but ignores contact resistance in MUX and scanner.



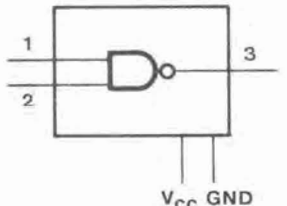
4-Terminal

Takes into account both the surrounding components and contact resistance in MUX and scanner.

Now, how about digital testing?

Let's start by reviewing how you would test a digital component not connected in a circuit.

To keep it simple, let's take the single NAND gate we used in a previous example.



To test this gate completely:

First, you would apply power (and ground) to the IC, to make the circuitry within the chip operational.

Then, you would apply all possible combinations of logic inputs (highs and lows) to the IC while checking the output for the proper logic levels.

This NAND gate has 4 possible combinations of inputs and, of course, a known output for each set of inputs.

Input Pins 1	Input Pins 2	Output Pin 3
Low	Low	High
Low	High	High
High	Low	High
High	High	Low

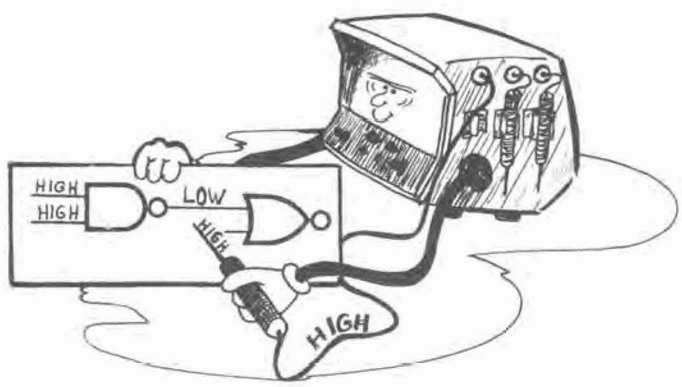
What happens when this IC is connected in a circuit?

For one thing, since power must be applied to the IC, circuit interconnections on the board will cause that power (and ground) to be applied to the rest of the ICs on the board, as well.

Remember, we did not need to apply power to the board when testing analog components.

Therefore, when the tester tries to apply a logic level to the IC input, it may find that the IC input pin is being held in the opposite state by the output of another IC.

For example, it may want to apply a high input to a pin that's being held low by another IC output.

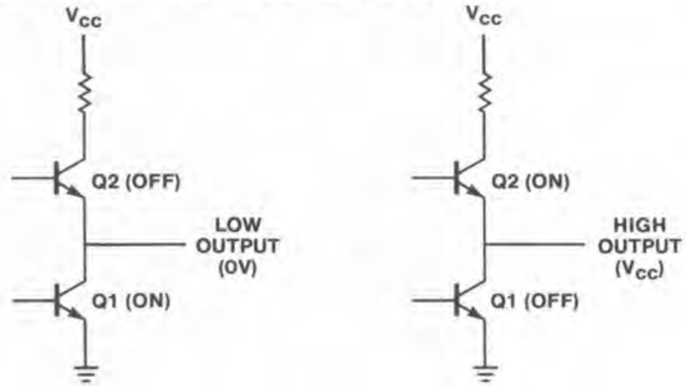


The digital drivers in the tester handle this problem by momentarily forcing the IC input to the desired state, regardless of what state it is being held in by another IC.

This technique of momentarily overriding an IC output is called **backdriving**.

What's involved in backdriving?

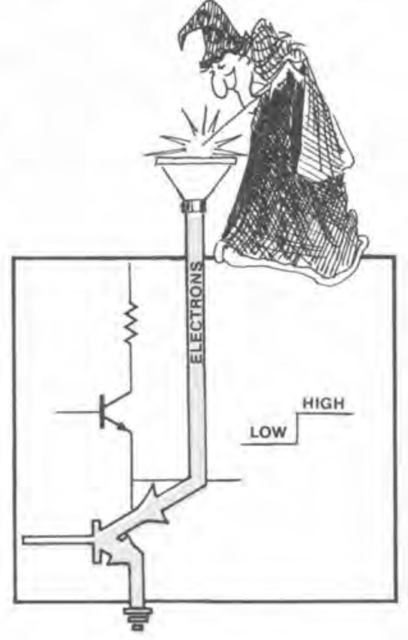
Consider the typical output stage of an IC, such as the TTL gate shown below. *Note: TTL (Transistor-Transistor-Logic) refers to the type of circuit architecture used in the design of the gate.*



A *low* output occurs when transistor Q1 is conducting (is on) and Q2 is off. To override this condition temporarily and develop a high output (say, about 3.5 V), the tester forces a current pulse of up to 300 mA back through Q1. This current flowing through the emitter-to-collector resistance of the transistor develops the high output.

Similarly, a *high* output exists when Q2 is conducting and Q1 is off. If the tester wants to force the output low, it applies a low level to the output and sinks the resulting current flowing through Q2.

Remember, the purpose of backdriving is to force a desired logic input to the IC-under-test, regardless of what any other IC outputs might be doing.

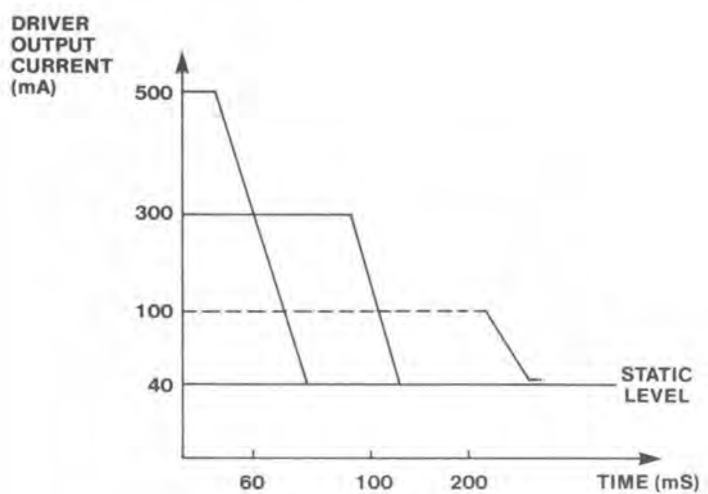


Doesn't this backdriving current harm the IC?

Not if the current is applied for only a very short time.

Since digital tests are conducted at high speeds, normally, the test current pulse is much less than 100 ms wide (typically 5-10μs) and, therefore, causes no problem.

If for any reason, however, a driver remains on and connected to the circuit for 100 ms or more, a sense circuit in the driver automatically reduces the current to around 40 mA, to protect both the IC and the tester electronics.



Now, back to the in-circuit digital test example.

Before any digital testing begins, the program first:

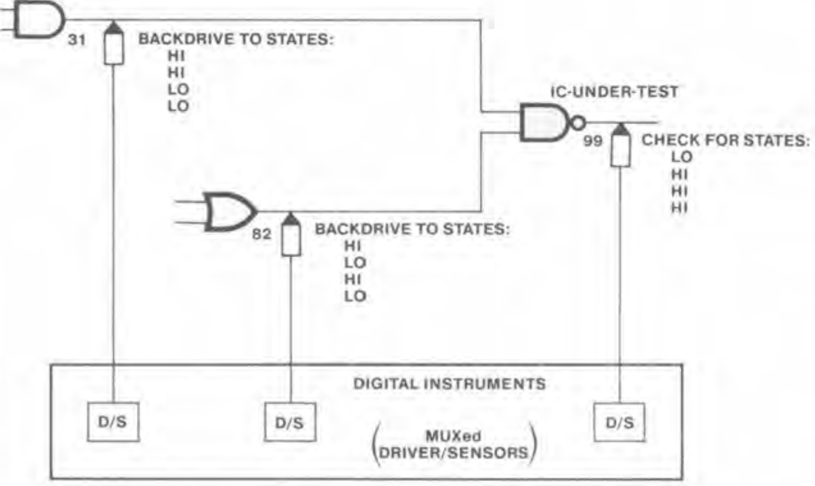
- Applies power and ground to the board.
- Defines the high and low logic levels.
- Tells the system what nails will be used during the test.

After this is done, test bursts can be executed to test each IC on the board.

For the single NAND gate we've chosen as an example, the following test burst statements will check the operation of that gate for all 4 combinations of inputs.

Statement	Drive Input Pins 31	82	Sense Output Pin 99
IC(31, 82) IH(31, 82) OS(99) OL(99);	H	H	L
IL(82) OH(99);	H	L	H
IL(31) IH(82);	L	H	H
IL(82);	L	L	H

IC(31, 82) connects the drivers to nails 31 and 82.
OS(99) enables the sensor on nail 99.
IH or IL drive the specified nails high (IH) or low (IL).
OH or OL indicate the expected high (OH) or low (OL) state of the specified output nails.

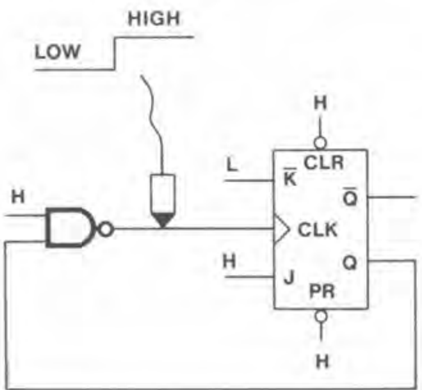


So that's all there is to digital testing?

Not quite.

Backdriving is certainly the key to digital testing but, as you might expect, other problems can arise. For instance, consider what can happen when testing an IC that's part of a feedback loop.

Let's say that the flip-flop shown below is initially cleared and is set-up to toggle to the opposite state whenever a high-going pulse is applied to its clock (CLK) input.

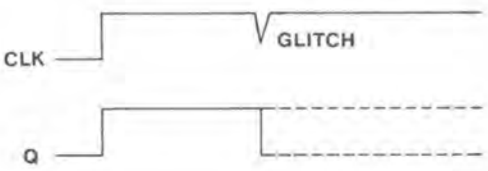


Part of the test on the flip-flop would be to pulse the CLK input and make sure that the flip-flop output changes state.

However, when this happens, the following unwanted action also takes place:

1. The sudden change in the flip-flop output (high-to-low) immediately feeds back to the CLK line, driving it low.

2. After a short time, the driver connected to the CLK line recovers and drives that node back high (where it's supposed to be).

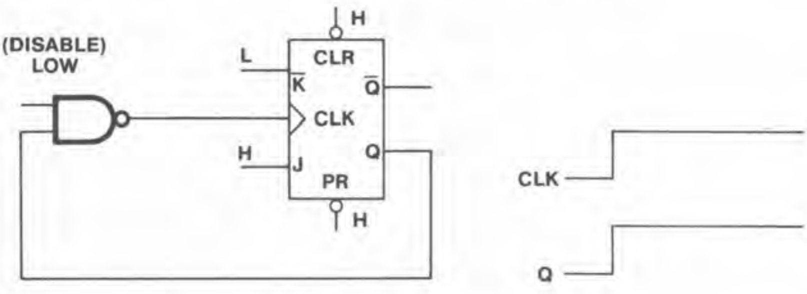


So, what's the problem?

The problem is the momentary glitch that appears on the CLK line while the driver is recovering. That glitch (depending on how big it is) might toggle the flip-flop back to the state it was in before the test.

If this happens, the tester will assume that the flip-flop did not toggle and, therefore, failed the test.

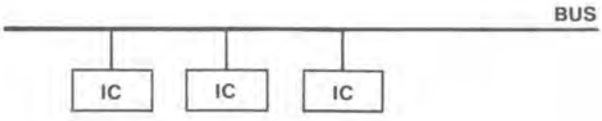
To prevent these glitches from becoming a problem, the tester analyzes the surrounding circuitry and tries to inhibit all devices in the feedback loop which may interfere with the test.



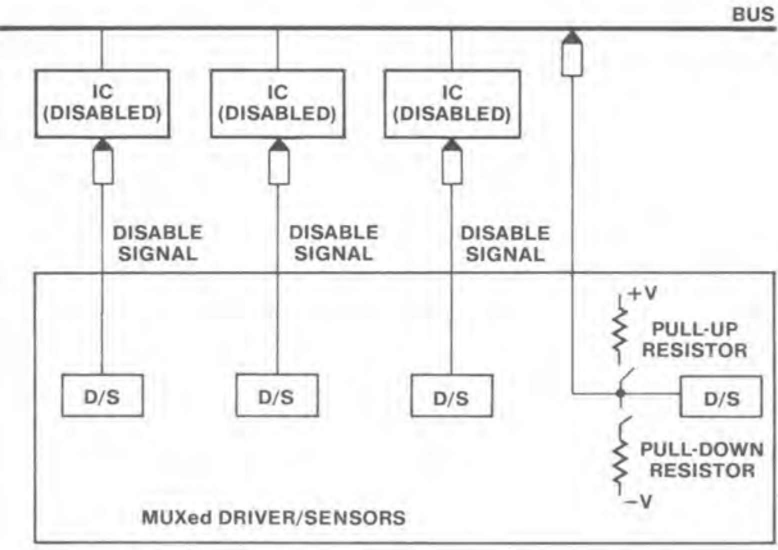
To protect against unwanted glitches during a test, GenRad's in-circuit testers **disable** all 3-state devices by placing them in their high-impedance state and **inhibit** all other devices (except the IC-under-test) by forcing their input pins to a state that effectively inhibits their operation. The output of a 3-state device has an output that can be high, low or in a high-impedance state.

Any other problems?

Another special situation arises when testing bused devices, i.e., several devices all tied to a common bus.



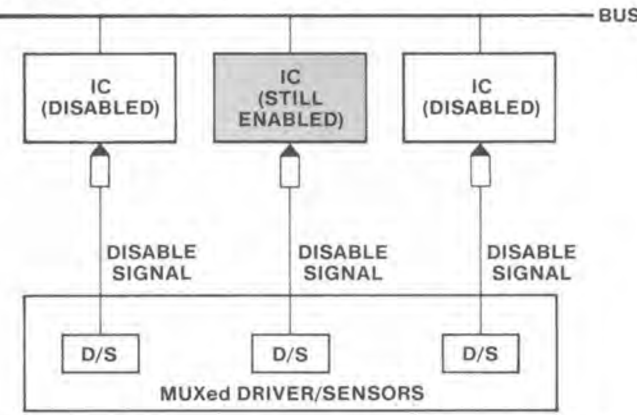
Before testing each individual device on the bus, the tester first disables them all, then connects the bus to a pull-up and then a pull-down resistor to make sure that the bus is not stuck in one state (either high or low).



If this pull-up/pull-down test is successful, the tester then checks each device individually to see if each one can control the state of the bus.

What happens if a bus is stuck in one state?

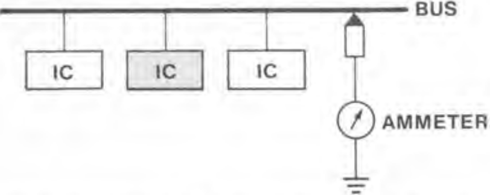
Good question. How can you tell which innocent-looking device refuses to be disabled? The device outputs are all tied together, so any one of them could be forcing the bus to this error state.



GenRad uses something called a **BUSBUST™** technique to find the bad bus device. The procedure goes something like this:

1. With all devices still "disabled," the tester measures the current at the failing bus node.

2. Then, it turns on one device at a time and applies logic inputs to that device to try to drive the bus node in the direction of the failure. For example, if the node is stuck low, the logic inputs to the device will cause its output to go low.



3. The tester again measures the current at the node.

4. If the current changes significantly, the device is deemed OK. However, if the current remains essentially the same as when the device was "disabled," then the IC is assumed to be bad.

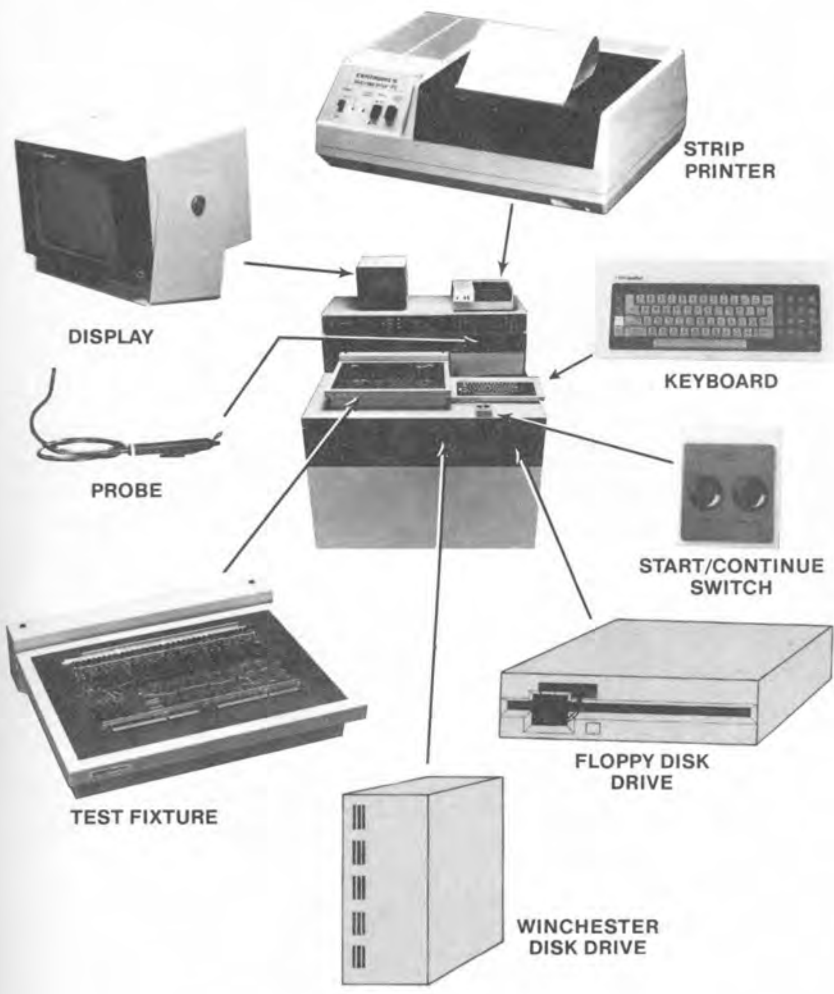
Now, let's go on to the next chapter and see what an in-circuit tester looks like.

Chapter 3

A Look at an In-Circuit Tester

Meet a real-live tester!

Shown below is the **2275 Board Test Workstation,** along with major components found in that tester.



This section describes the major hardware and software components used by test operators and programmers when operating the testers.

Let's start by looking at the hardware components.

What hardware components does a test operator use?

The operator who uses the system solely for testing boards would typically use the following system components on a regular basis:

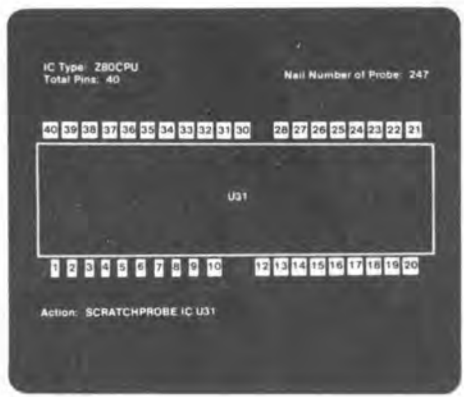
- Video Display
- Keypad Controls
- Message Printer
- Disk Units
- Fixtures
- Probe

The video display

The display looks like a small TV set mounted above the work surface (where it's out of the way, but easy to see). It's adjustable for your viewing comfort.

The programmer, of course, uses the display for preparing test programs and selecting various system options. These functions are described later in this chapter.

The display screen can also prompt the operator to do certain things during board testing. For example, it can guide the operator in probing the board whenever an IC failure is detected. The results of the probing are shown graphically on the screen. (More about this **SCRATCHPROBING™** technique later in this chapter.)



The keypad controls

The keypad controls are specifically designed for the test operator. It has all the control keys that an operator needs for production testing, but few keys for accessing the system.

On a 2271 or 2272 system, the keypad is a small calculator-size keyboard.

2271, 2272
KEYPAD

On a 2275 system, most keypad controls and indicators are built into the front control panel. The START/CONTINUE button, however, is kept portable, for convenience.

2275
START SWITCHES

2275
CONTROL
PANEL

When the operator presses the LOAD PROGRAM key, the system reads a pre-wired identification code from the test fixture, then locates and loads the corresponding test program into memory.

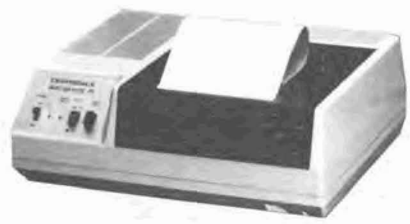
When the operator presses the START/CONTINUE button, the system starts (or resumes) testing.

The number keys (0-9) and the YES/NO keys can be used for selecting programmed test options and are not typically used by the operator unless the operator is prompted by the test program.

The message printers

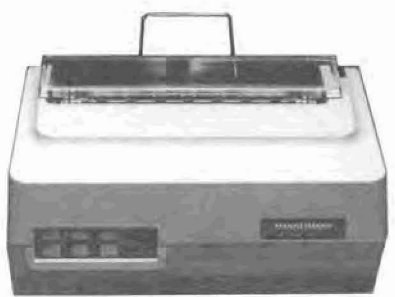
After a board has been tested, the tester prints appropriate test results, fault messages, and repair instructions for that board on a message printer. The user can then attach these messages to the board so that a record of the test results will accompany the board to the repair station.

The 2271 and 2272 systems use the **strip printer** shown here, in which messages are printed on narrow, 4.75-inch (12 cm) wide strips of aluminized paper.



STRIP PRINTER

The 2275 tester can use either the strip printer or a **combination printer** that can print program listings and reports, as well as operator messages, on plain 8½ inch (21.6 cm) wide paper.



COMBINATION PRINTER

The disk unit(s)

All the system software is stored on magnetic disks. Magnetic disks are ideally suited for bulk storage of information, since each disk is capable of storing large amounts of data.

GenRad testers use 3 different types of disk units (in various combinations): hard disk, Winchester, and floppy disk.

The **hard disk** units use both fixed and removable disk cartridges as the storage medium. The disk drive used by the 2271 and 2272 systems can store up to 56 million bytes (56 megabytes) of data.

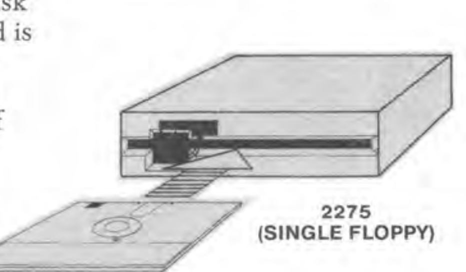
2271/2272
HARD DISK UNIT

The **Winchester** unit is essentially a high-speed sealed disk unit, with non removable disk. The 2275 can have up to two 21-megabyte or 55-megabyte Winchester units.

2275
WINCHESTER

The **floppy disk** units have a storage medium that resembles small 45-RPM records in their record jackets. Each floppy disk (or diskette) is easy to use and is highly portable. Floppy disk units used in GenRad testers can store up to 1 megabyte of data.

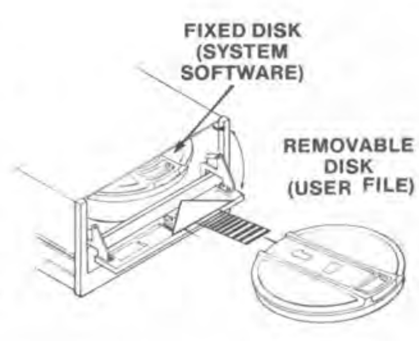
2271, 2272
(DUAL FLOPPY)


2275
(SINGLE FLOPPY)

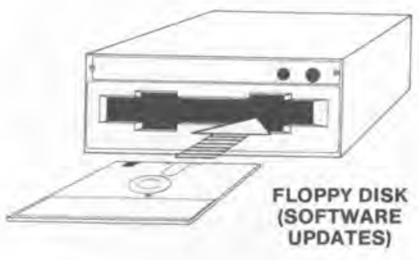
Disk units (cont)

The 2271 and 2272 systems use:

- a hard disk unit with both fixed and removable disk cartridges. The fixed disk contains the system software. The removable disk stores the user-created files and can also be used for reloading system software onto the fixed disk, running maintenance procedures, etc.

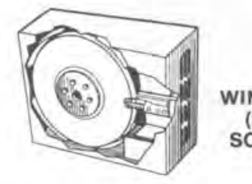


- a floppy disk unit on at least one system per installation for updating software on the hard disk.

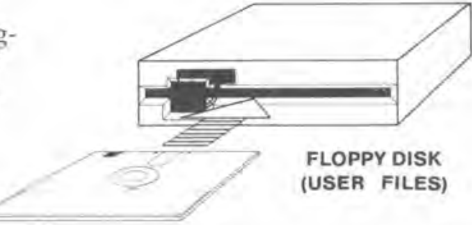


The 2275 system uses:

- a Winchester drive for storing the system software, test programs, user files, etc.



- a floppy disk unit for long-term, portable storage of programs and other files.



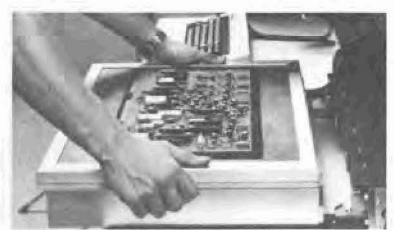
When the system is initialized, for example, when power is turned on, the tester automatically loads ("boots") the operating system software from disk to the internal memory of the computer. Then, during program preparation and execution, additional files are transferred between the specified disk and memory, as needed.

The bed-of-nails fixture

As you've already seen, a vacuum-driven bed-of-nails fixture is used to connect the pc board to the tester. Shown on this page are the single-board fixtures available for GenRad's in-circuit testers.

2271 - Two fixtures available:

Max board size -
13 in. × 17 in. (33 cm × 43 cm)
17 in. × 21 in. (43 cm × 53 cm)



2271

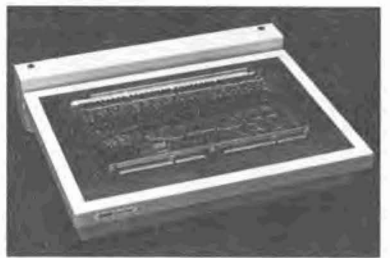
2272 - Three fixtures available:

Max board size -
12 in. × 26 in. (30 cm × 66 cm)
26 in. × 16 in. (66 cm × 41 cm)
26 in. × 26 in. (66 cm × 66 cm)

2272

2275 - Two fixtures available:

Max board size -
12 in. × 18 in. (30 cm × 46 cm)
20 in. × 22 in. (51 cm × 56 cm)



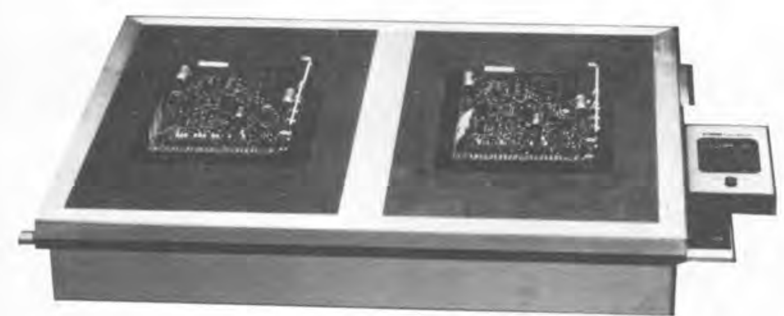
2275

Converters are also available which allow fixtures from one tester to be used on another type of tester. For example, with the proper converter, a 2271 fixture can be used on a 2275 tester.

Dual test fixtures

Dual fixtures can be used to speed-up testing in a high-volume test environment.

These dual fixtures have 2 separate beds-of-nails and 2 separate sets of controls. While one board is being tested on one fixture, another board can be loaded on the other fixture in preparation for its test.



As in the single-board fixture, various sizes of dual fixtures are available.

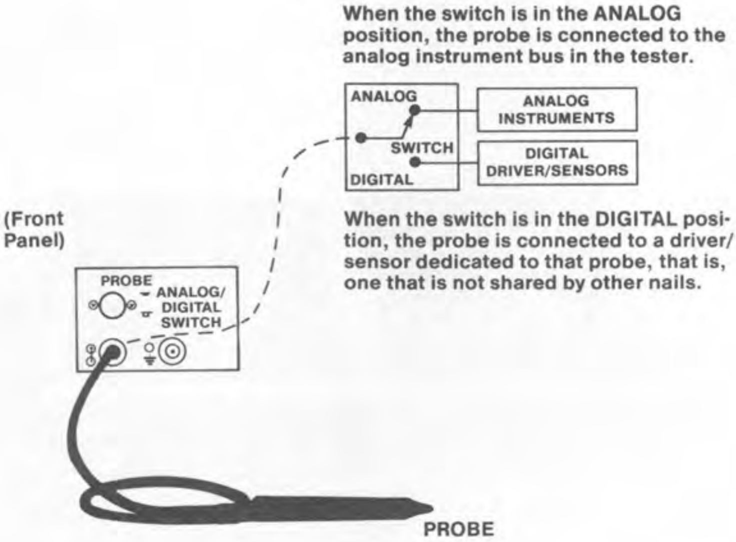
The 2271 dual fixture can accommodate 2 pc boards up to 9.5 in. × 17 in. (24 cm × 43 cm) each.

The 2272 dual fixture can accommodate 2 boards 16 in. × 11 in. (41 cm × 28 cm).

The 2275 has 2 dual fixtures available: one for 2 boards 12 in. × 8 in. (30 cm × 20 cm), the other for 2 boards 20 in. × 10 in. (51 cm × 25 cm).

The probe

Connected to the front panel of the tester is a test probe controlled by a switch labeled ANALOG/DIGITAL.



The probe, therefore, can be thought of as a movable test nail that the tester uses mainly for checking continuity. An example is shown on the following page.

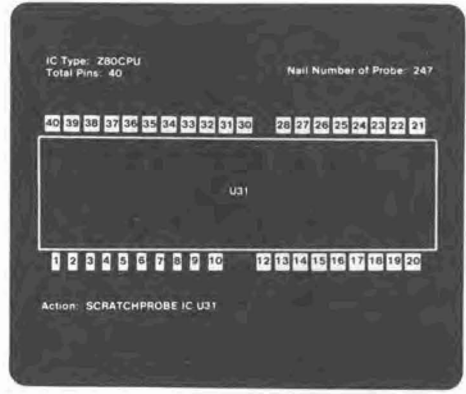
GenRad's Scratchprobing technique

A particularly useful application of the probe is GenRad's **Scratchprobing** technique.

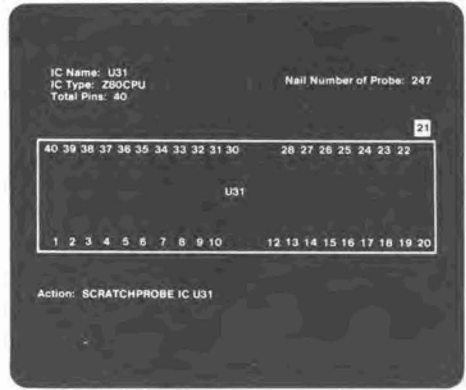
When the system detects a failure in an integrated circuit (IC), it doesn't really know if the IC itself has failed or if an open track on the board or a faulty test nail is the real cause of the problem.

So, the system prompts the operator to check the IC pins by taking the probe and quickly drawing it against the legs of the IC, in any direction.

An outline of the IC and its pins appears on the screen.



As the probe comes in contact with an IC pin, the number of that pin jumps inside the outline drawing. The numbers of the pins that do not make contact remain outside the drawing.



As a result, the operator is quickly made aware of a bad connection and the system automatically diagnoses the true fault and prints it out for the test/repair person.

What hardware components does a programmer use?

During the test-preparation phase, the programmer needs greater access to the system than does the operator. The programmer also typically needs printed copies of programs and reports.

Therefore, the programmer would use not only those items used by the test operator:

- Video Display
- Keypad Controls
- Message Printer
- Disk Units
- Fixtures
- Probe

but also the following additional components:

- Keyboard
- Line Printer (Optional)
- Background Terminal (Optional)
- Programming Stations (Optional)

The keyboard

The keyboard is indispensable to the test programmer and system manager, since it provides the only means of accessing all the system software options.

The typewriter-like keyboard has all the keys needed for preparing test programs, creating and editing files, selecting monitor options, etc.

The keyboard is attached to a flexible cable that plugs into a connector on the control panel. Some useful features result from this arrangement:

- You can move the keyboard around the work surface for your convenience.
- More important, perhaps, you can easily detach the keyboard from the system to prevent unauthorized or accidental access to the software files.

The line printer (optional)

Although it's possible to use the strip printer for obtaining printouts of programs and reports, the optional line printer or, in the case of the 2275, the combination printer is much better suited to that purpose.

The **line printer** handles standard 14-inch (35.6 cm) wide computer paper and prints up to 132 characters (standard 10-point type) per line. Its main purpose is for listing programs, reports, and data files that require *long* lines of data. This printer is optional on the 2271, 2272 and 2775 testers.

LINE PRINTER

The optional **combination printer** can print up to 80 standard 10-point type characters per line, on 8½-inch (21.6 cm) wide paper. Larger characters can also be printed for headings or for highlighting portions of text. This printer is well-suited for short messages to the operator as well as longer program listings, reports, etc.

COMBINATION PRINTER

The foreground/background option

A separate **background terminal** (and supporting software) is available which lets 2 separate tasks be performed simultaneously on the same test system:

The test operator uses the "foreground terminal," consisting of the video display and keypad, to test boards

while at the same time

the programmer uses the separate background terminal (a DEC VT10x) to prepare programs.

Central/Programming Stations (optional)

Off-line program preparation can also be performed on GenRad's optional 229x stations. These stations and the various GenRad testers can communicate with each other over a special networking facility called **GRnet™** networking system.

Shown below is GenRad's 2293 Central Station.

Any other hardware?

Other major hardware components in the tester include:

- The **computer** system that controls the whole operation of the tester. It consists of a DEC LSI-11/23 computer with associated memory and input/output circuitry for handling all peripheral devices such as keyboard, display, disk units, printer, etc.

- The **test instruments.** Both analog source and measure devices and the digital driver/sensor system.

- **User power supplies.** Several types of power supplies (both fixed and programmable) are available for applying power to the board during digital testing.

This hardware is packaged on printed-circuit boards and modules which plug into the various system cabinets (bays).

Now, for the software

What makes the difference between a manual tester and an automatic tester?

Why, the software, of course. (Ask any programmer.)

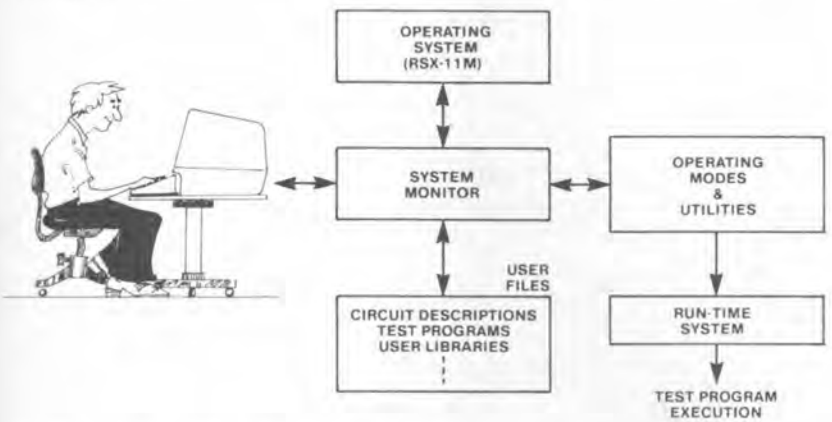
The software consists of all the program instructions and data used by the computer to control the operation of the test system.

To perform all the tasks expected of it, the board tester requires a sophisticated software system, made up of many programs, sub-programs and data files.

On the following pages, we'll look at how this software is organized, what it does and how it is used.

OK, where do we start?

We'll start by looking at the general organization of the system software.



At the top-most level, controlling the overall operation of the system is DEC's RSX-11M **operating system.** This program controls all input and output operations, assigns memory space to programs and data and, in general, lets the computer get at the many programs and files in the system software, easily and quickly.

While a board is being tested, the **run-time system** handles the operation of the test instruments, under control of the test program.

> *Both the OPERATING SYSTEM and RUN-TIME SYSTEM operate behind the scenes, and most users never get directly involved with these programs.*

The user, however, gets very much involved with the system **monitor.** It is through the monitor that the user accesses and controls the various programs available in the test system.

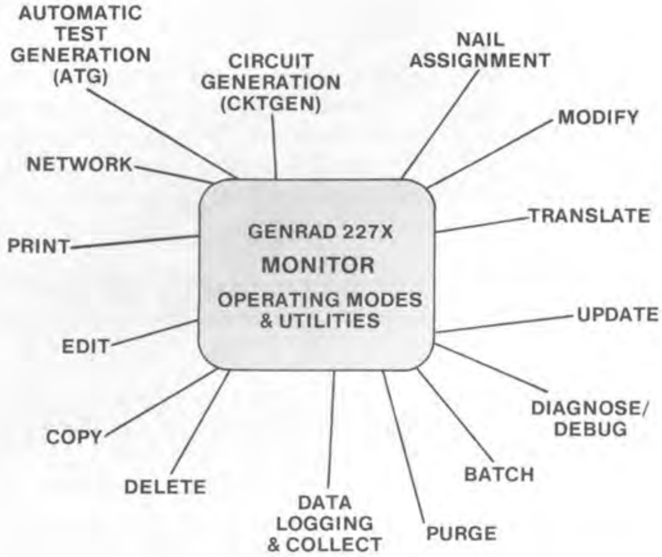
So, tell me about the monitor.

The monitor controls all of the many tasks or functions that the tester can perform. These tasks are organized into operating modes and utility routines. For each mode and utility routine, the monitor displays a menu-type page of user-selectable options. (We'll look at these monitor pages a little later.)

To run a task, you would:

- Call up the monitor options page for that task
- Specify the appropriate options
- Run the task

Let's start by looking at the many software tasks that the monitor handles.



Operating modes and utilities

It's beyond the scope of this document to describe in detail all the operating modes and utilities in the system. But, as an introduction, here's a brief description of most of them. They're grouped roughly according to the general function that each one performs.

PREPARING THE TEST PROGRAM

CKTGEN - Generates the circuit description from components list
ATG - Generates the test program from circuit description
ACL - Calls (special) analog components library
ATL - Calls analog test library
DTL - Calls digital test library

PREPARING THE TEST FIXTURE

NAIL ASSIGNMENT - Assigns test nails in multiplexed D/S systems

TRANSLATING THE TEST PROGRAM

TRANSLATE - Compiles test program into machine code
UPDATE - Handles changes to test program
MODIFY - Allows changes to machine-coded test program

RUNNING THE TEST PROGRAM

DIAGNOSE - Runs the test program for debugging and testing

FILE MANAGEMENT

COPY - Copies information from one file to another
PURGE - Removes old versions of files from disk
DELETE - Deletes files from disk
PRINT - Outputs files to printer

OTHER MODES AND UTILITIES

BATCH - Runs system under control of a batch file
EDIT - Lets you create or revise files
LOG - Displays data collected during board testing
NETWORK - Transfers files to/from remote systems using GRnet

Does the test operator need to use the monitor?

Yes, to some extent.

The monitor has 2 levels of access:

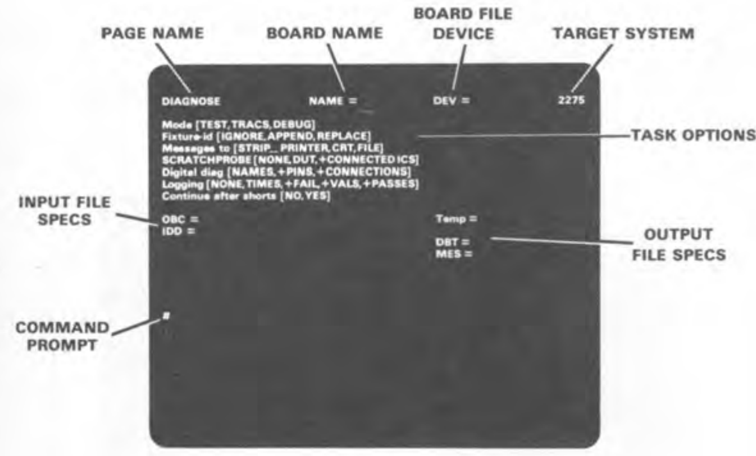
> When the system is being used for production testing *only,* the system manager (or authorized programmer) may set the monitor to a **LIMITED ACCESS** level. This not only reduces the number of options that the operator must select before testing a board, but it also prevents unauthorized access to the software files during testing. Even with a keyboard, the test operator is not able to access files if the monitor is set for limited access.
>
> The other monitor level is **UNLIMITED ACCESS** which provides full access to all the user software. The system manager and programmers operate the system from this monitor level.

The following discussion assumes that the monitor is in UNLIMITED ACCESS mode and that the full keyboard is being used (not just the keypad).

How do I use the monitor?

By filling in menu pages.

Once the operating system gets loaded into the tester and the system initializes itself, a DIAGNOSE mode options page (or menu page) automatically appears on the screen.



Remember, every monitor mode has its very own options page, similar to this one.

How about an example showing how to use the monitor?

OK.

Suppose you wanted to create a brand new data file, using one of the system editors.

The first thing to do is to call up the EDIT mode options page.

Simply type EDIT from the command prompt appearing at the bottom of the options page, and press RETURN. The EDIT mode options page will be displayed.

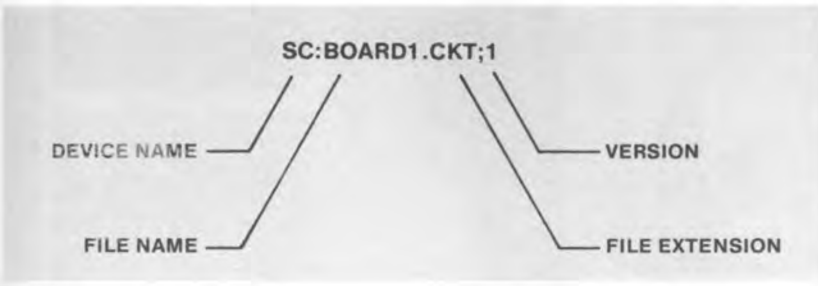
Then what?

You'll notice that some of the options are of the fill-in-the-blank variety (NAME= , DEV= , INPUT=), and some of the options are multiple choice (Mode [ED2, TECO]).

To specify an option, first move the cursor to that position on the screen (using the keyboard keys); then either fill in the blank or choose the desired option in the list by stepping the cursor to that option.

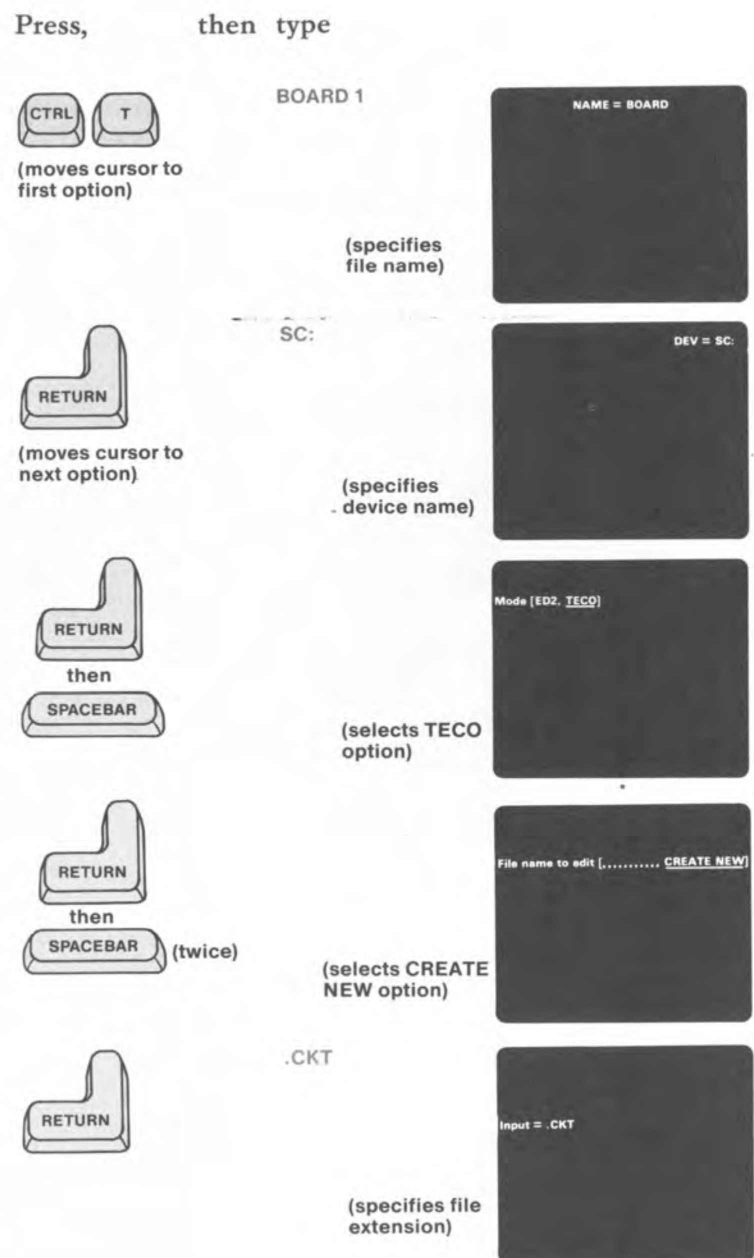
In this example, suppose you want to use the TECO editor (the system has 2 editors: TECO and ED2) to create a file, and you want to name the new file SC:BOARD1.CKT.

Note: Files are specified as follows:



Now, proceed as follows

Monitor example



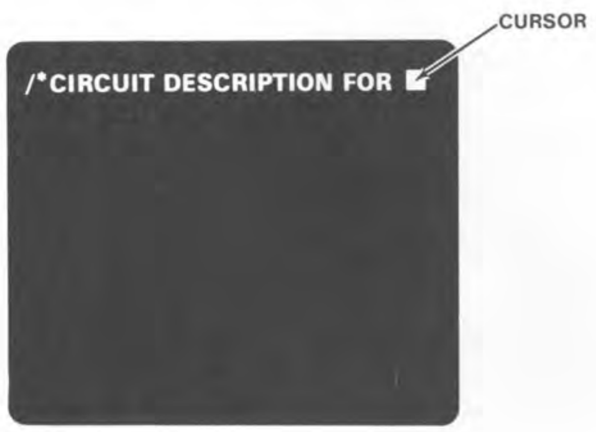
Monitor example (cont)

After the last option has been specified, press RETURN to move the cursor to the command prompt (#).

Then type RUN (or press the RUN key).

The screen will clear and remain empty (except for the cursor symbol).

The tester is now ready for you to start creating the file. Simply use the keyboard to enter the desired data.



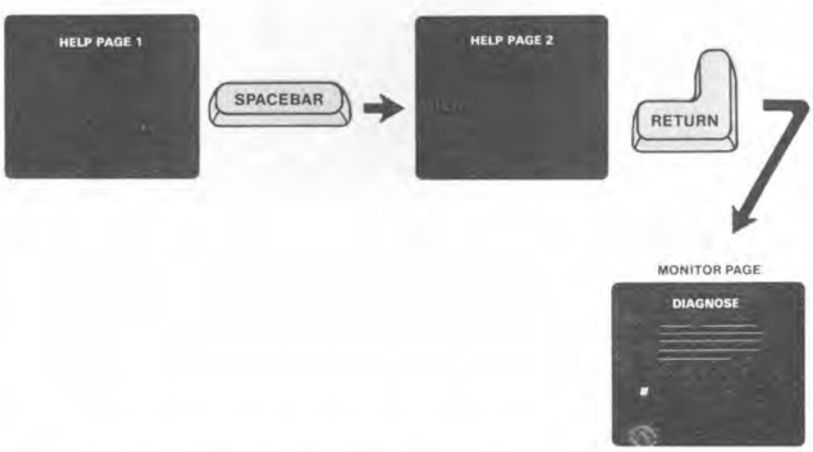
When you've finished with the file (and want to save it), simply exit TECO by pressing the ESC key followed by the CTRL and Z keys.

Can I get help filling out the monitor pages?

Every monitor page has a HELP facility. To obtain general information on a particular monitor mode (or on the system in general), simply type HELP from the command prompt (#) at the bottom of that monitor page, or press the HELP key on the keyboard.

Pressing the spacebar will step the display through a series of help pages.

Pressing the RETURN key will return the display to the monitor options page.



Help information for a specific option can be obtained either by moving the cursor to that option and pressing the HELP key or by typing HELP and the option name from the command prompt.

What about the other software?

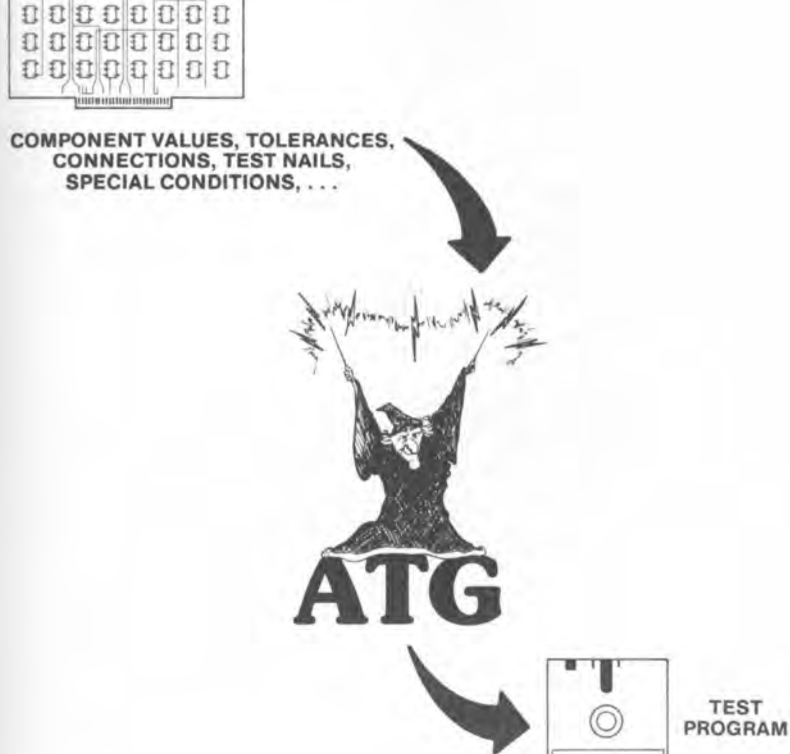
Probably the most sophisticated piece of software in the whole system is the software that develops the test program.

Even with the brief introduction to in-circuit testing presented here, you can appreciate the effort required to develop the test program for a large pc board. There are hundreds of nodes, components, tolerances, interconnections, test procedures, test limits, etc, to take into account.

In GenRad testers, a software task called

ATG – AUTOMATIC TEST GENERATION

performs this function automatically.



How does ATG work?

Very well.

You start the ATG process by inputting the circuit description of the pc board into the tester. This information consists of component names, values, tolerances, interconnections, etc, and is usually read directly from the schematic drawing of the board.

ATG uses this description to find a test for each component on the board. The system has an extensive library of component and test procedures. ATG checks through these library files to find an acceptable test procedure for each component.

ATG (cont)

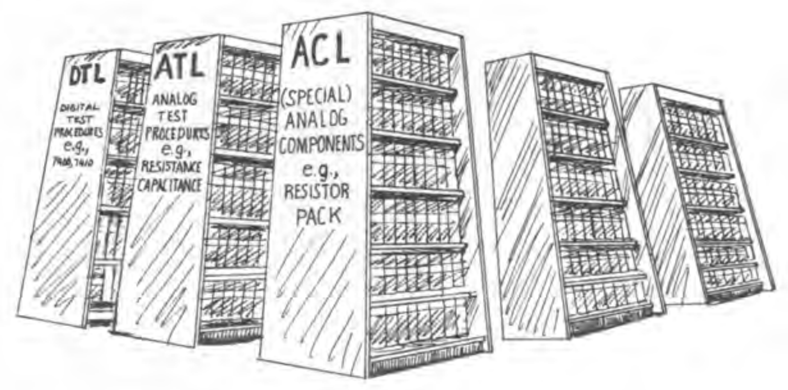
When an acceptable procedure is found, ATG defines the specific parameters (test voltages, guard points, delays, etc) for customizing that general procedure to the particular in-circuit configuration of the component being tested.

ATG then collects these individual tests and assembles them into a logical test program (i.e., shorts and opens testing first, followed by analog testing, then digital testing).

The ATG libraries

In the process of developing the test program, ATG has access to 3 types of library files:

> DIGITAL TEST LIBRARY (DTL) –
> Contains test procedures for checking hundreds of different types of digital ICs.
>
> ANALOG TEST LIBRARY (ATL) –
> Contains test procedures for checking basic analog characteristics such as resistance, capacitance, inductance, transistor gain, etc.
>
> ANALOG COMPONENT LIBRARY (ACL) –
> Contains circuit descriptions of complex or unusual analog components (e.g., a special resistor pack) that would not be directly recognized by the ATG software.



Can I change the ATG libraries?

You can customize and update the library procedures by creating your own **user library** files, but you *cannot* change the contents of the **system library.**

ATG will always search through the user library for a test procedure before it searches through the system library. If an acceptable test is found in the user library, ATG will use that test. If no acceptable test is found in the user library, then ATG will try to find one in the system library.

The assumption is that all test procedures and component descriptions that you put into your user library were put there to supersede corresponding procedures and descriptions in the system library.

In addition, GenRad testers also have an **update library** that allows GenRad to update system libraries in the field, on a periodic basis. With an update library in place in a user's system, ATG will always search through this update library before searching through the system library. New tests or replacement tests in this update library will, therefore, automatically appear in the test program.

Now that we've seen what components make up the tester, let's see how we actually use it.

Chapter 4
Using the Tester

Using the test system.

To tie all of this in-circuit testing information together, let's step through each phase of the in-circuit testing process. Remember, the process consists of 2 separate phases:

TEST SET PREPARATION

where the test program is developed and the test fixture wired.

BOARD TESTING

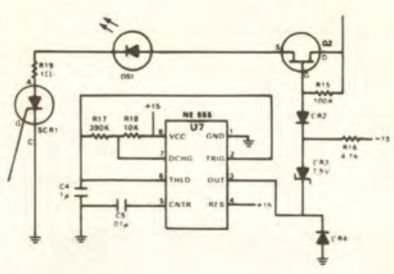
where the test set is used to test boards in a manufacturing environment.

Note: Since each board type has its own unique test program and test fixture, GenRad refers to the combination of these 2 items as a TEST SET.

What do I need to get started?

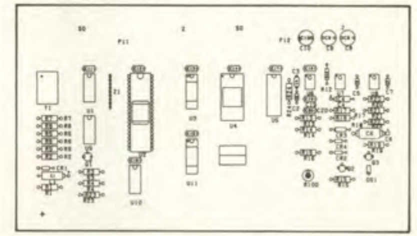
SCHEMATIC DIAGRAM AND PARTS LIST

These contain the component names, types, values, tolerances, and other information that you'll need for the circuit description.



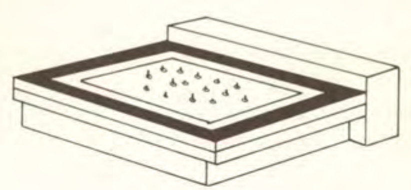
ASSEMBLY DRAWINGS

Used for locating precisely where to place the test nails.



TEST FIXTURE KIT

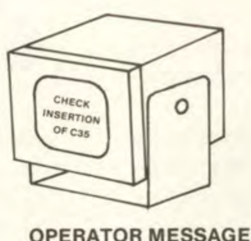
Contains all the hardware for assembling and wiring the fixture.



Anything else I need to do before getting started?

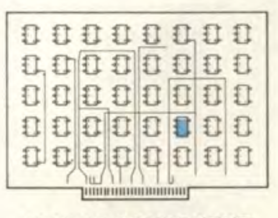
It's a good idea to examine the schematic diagram and try to anticipate any special testing requirements. For example

Are there any special instructions that you'd like to give the operator during a test?



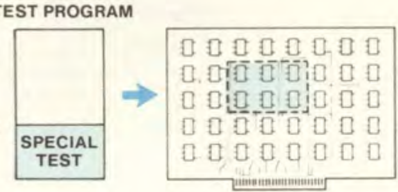
OPERATOR MESSAGE

Does the board contain special components that are not contained in the system library?



UNIQUE COMPONENT

Are there any special functional tests that you'd like to add for testing unique circuit segments?



SPECIAL FUNCTIONAL TEST

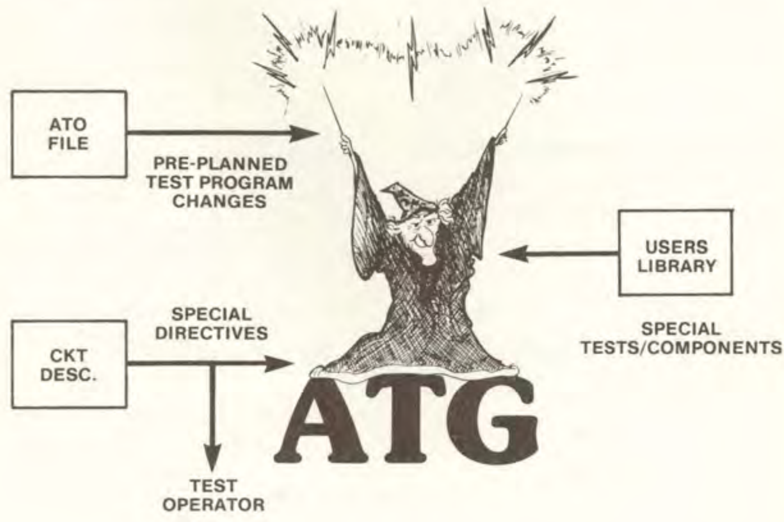
What if there are special testing requirements?

GenRad testers have several features designed to help you take care of those problems. For example

You can insert special directives (called **flagspecs**) in the circuit description, which can either guide the ATG process in developing the test program or can provide special instructions to the operator during testing.

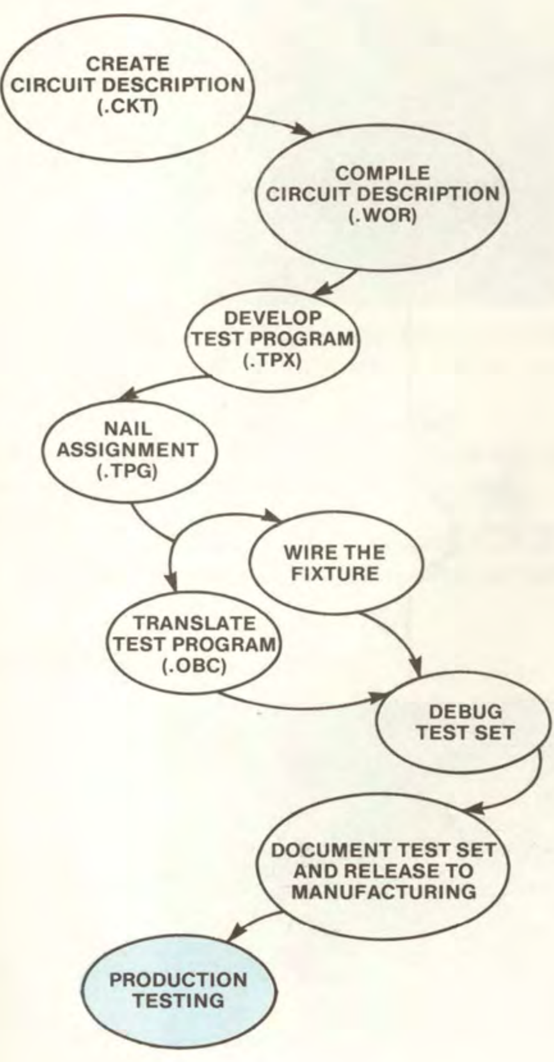
You can write an **Automatic Test Options (ATO)** file that will make pre-planned changes to the test program that ATG develops.

You can create new test or component descriptions for the ATG **User's Library.**



OK, now can I start preparing the test set?

Sure. The chart below shows the major steps along the way.



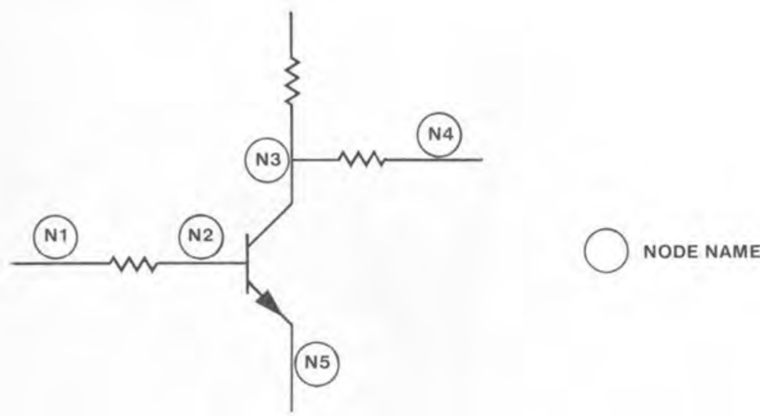
How do I start?

Take the schematic diagram of the board and write down a name for each circuit node, for example, N1, N2, etc.

These **node names** serve 2 purposes:

They help you describe how the circuit components are interconnected.

They can be used as **temporary nail numbers** if you don't know what real nail numbers to assign. (We'll explain this nail assignment function a little later in this chapter.)



How do I enter the circuit description information?

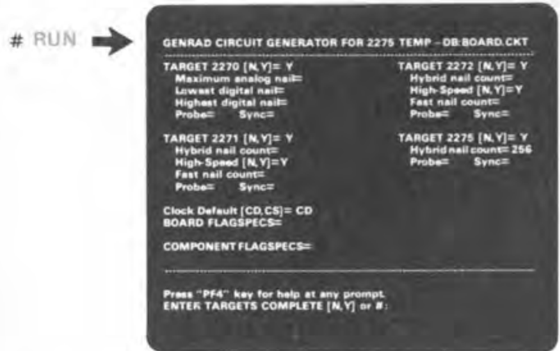
First, call up the CKTGEN monitor page by typing CKTGEN at the command prompt and pressing RETURN.

CKTGEN is an automatic circuit generator that prompts you for each piece of circuit information the tester needs to create a circuit description file.

Select the CREATE option to open a new database file called .CIR, specify appropriate file names, and then run the task.

Note: To avoid getting bogged down in details and losing track of the overall flow, monitor options will be covered only in general terms throughout this chapter.

The following BOARD page will appear.



Fill in the test nail information requested for the "target" system (the system on which the test will be run).

CKTGEN (Circuit Generator)

After you indicate that the target data has been entered, the tester will prompt you to enter a component type.

For each component type specified, CKTGEN will display a page similar to the one shown below for resistors. All you do is respond to the prompts and enter the correct circuit information.

Generating the test program using ATG

After you've compiled the circuit description, you're ready to let ATG develop the test program.

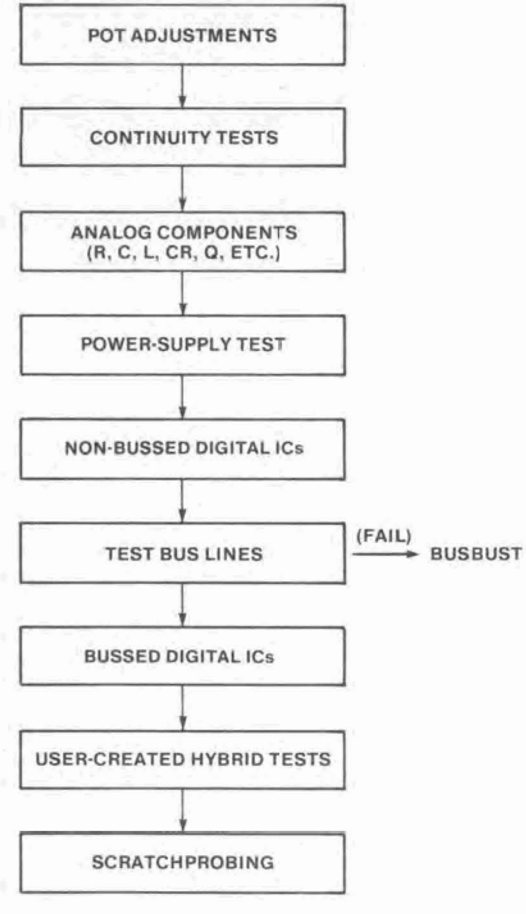
To do this call up the ATG options page and run the task.

The test program

Based on the circuit description information in the .CKT file and the available test routines in the library, ATG now attempts to find a suitable test for each component on the board.

The accumulation of all these individual tests forms the test program. This user-readable **test program** file is called **.TPG.**

The testing sequence is, of course, important. Starting with the most basic, each test level is designed to build confidence in the next higher level of tests. The testing sequence is shown in the flow chart below.



The ATG reports

When ATG finishes assembling the test program, it reports any known program deficiencies. For example, it will inform you of any components for which it could not find an acceptable test. You, the programmer, can then modify the program to account for these deficiencies or make other changes, as appropriate.

ATG produces 2 major reports: one for the analog tests (called .RPT) and one for the digital tests (called .MSG). Brief samples of each report are shown below.

The **analog test report .RPT** starts by identifying the board and the libraries used to generate the tests. Connectivity results (opens and shorts tests) are then reported, followed by test information for each type of analog component (capacitors, transistors, resistors, etc). A summary report rating the quality of the tests is also produced.

.RPT

```
ANALOG ATG(ATGMNA[illegible]) REPORT FOR TRAINBBD          31-OCT-83 11:48
       PAGE 1

Libraries used during analog ATG
Library SYS: LB1:[2,11]TSTLIB740.LIB

Connectivity results

Lib  Test    Page

SYS  OPENS   4 1

Pin list: 45,46: 0
Analog threshold         10 ohms
Digital threshold        45 ohms
Test voltage             100M

Board Test Statistics
     32 Superior Tests  (Average err  1.25%)   (86.49%)
      2 Adequate Tests  (Average err  29.43%)  (5.405%)
      3 Unsuccessful Tests                     (8.108%)
      0 Not Testable Characteristics
     10 Connectivity, Utility, and ATG Tests
     47 Total Board Tests Processed
```

The **digital test report .MSG** also starts by identifying the board and the libraries, then summarizes the digital test statistics and problem areas for testing that board. This is followed by test information for each digital IC on the board.

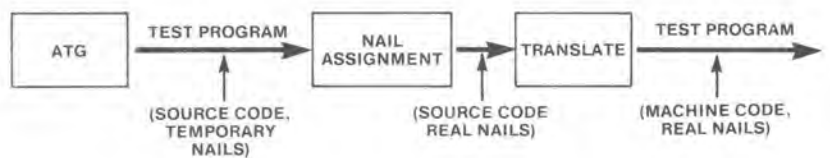
.MSG

```
ATG Report On Digital Test Section SC TRAINBBD DTP 31-OCT-83 12:11
(DTGMNA02) Libraries:   LB1:[2,11]DIGLIB740.DTL

BOARD STATISTICS SUMMARY

        Device count              51
        Digital IC count          13
        Undriveable ICs            0
Target tester: 2275           Adaptor specified for: 2275

  Begin Database Creation

   7 Digital ATG Option commands found in SC TRAINBBD.ATX
PROBLEM: No test program found for IC U5(NE5[illegible])
PROBLEM: No test program found for IC U6(LM1458)
PROBLEM: No test program found for IC Y1(XTL)
PROBLEM: No test program found for IC Z1(RPAK8)
Lowest nail used:    13
Highest nail used: 4018

Logic Level Assignments for Nails
     LHLA(13-20,49-52,78,82-85,109-[illegible],141-148,173-180,205-210,[illegible]
      -242,244,269-273,301-308,4001,4002,4009,4010,4017,4018)

Results for Non-bussed ICs

        PIN RESULTS FOR U1 (7400)
             Equivalent Set 1
         Pin  Type   Nail  Tested?  Comments
          1   IN     301    YES       A
          2   IN     302    YES       B
          3   OUT    303    YES       Q
```

When I finish fine-tuning the test program, then what?

Eventually, the program has to be **translated** into machine code, before it can be run.

But before that, the temporary test nails you assigned back in the circuit description, have to be converted to real nail numbers. This is done by running the test program through a **NAIL ASSIGNMENT** process.



Remember, because of driver/sensor multiplexing, each test nail does *not* have its own dedicated driver/sensor.

In the 2272, 2 driver/sensors are shared by 16 test nails.

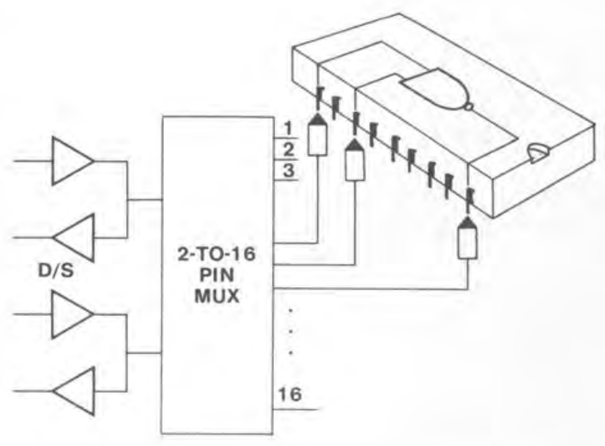
In the 2271 and 2275, 4 driver/sensors are shared by 16 test nails.

Why is nail assignment necessary?

Because, when a tester contains multiplexed driver/sensors,

you cannot arbitrarily assign just any test nail to any circuit node

Look at the following example of a 2-to-16 driver/sensor multiplexer.



There are 2 driver/sensors available for 16 nails.

If you connect 3 nails (from the same group of 16) to the NAND gate, the system cannot possibly drive both inputs and sense the output simultaneously, with only 2 driver/sensors.

To prevent this conflict, at least 1 of those 3 nails should come from another group of 16.

Do I have to worry about this?

Fortunately, no.

All you need to do is assign temporary nail numbers in the circuit description (such as N1, N2, . . N354, . . etc), use ATG to develop the test program, then run NAIL ASSIGNMENT.

To keep track of which files have temporary nail numbers and which don't, the following naming scheme is used:

Real Nails	Temporary Nails
.TPG	.TPX
.IDD	.IDX
.ATO	.ATX

Nail assignment analyzes the test program and other input files and automatically assigns real nail numbers so that testing conflicts will not occur.

It then modifies the input files to reflect these real nail assignments and produces several different reports on the results.

FILES THAT HAVE TEMPORARY NAILS: .CKT, .TPX, .ATX, .IDD

UPDATED FILES WITH REAL (TARGET) NAILS: .CKT, .TPG, .ATO, .IDD

NAIL ASSIGNMENT REPORTS

What are the Nail Assignment reports used for?

The **Nail Assignment Report (.NAR)** is the *basic one.* It provides a cross reference between temporary nail numbers and real nail numbers and indicates preferred locations for nail placements.

.NAR

```
ASSIGNED           TEMP                  Page: 15
  NAIL   NODE      NAIL  CONNECTOR     COMPONENTS
-------------------------------------------------
  H237   N130      H235  9-077       *U146,12, U163,13
  H238   N132      H236  9-078       *U163,11, U146,6
  H239   N278      H223  9-079       *U147,11, U148,18
  H240   N269      H227  9-080       *U158,8, U158,5, U159,9
  H241   N540      H158  9-081       *U84,2, U82,3, U98,17
  H242   N541      H159  9-082       *U84,15, U92,6, U98,6
  H243   N423      H187  9-083       *U88,9, U79,13, U198,6
  H244   N288      H149  9-084       *U125,6, U116,19
  H245   N728      H249  9-085       *U126,19, C8,1
  H246   N239      H219  9-086       ( 2-157)
  H247   N730      H245  9-087       *U126,12, CR3,C
  H248   N798      H246  9-088       *U105,18, U75,13
  H249   N17       H251  9-089       U7,15, U8,15, U15,15, U16,15, U23,15,
                                     [illegible]
```

The **Nail Fixture Report (.NFR)** is a wiring list for the test fixture. It specifies all test nail connections between the circuit board and the system.

.NFR

```
WIRE WRAP LISTING FOR CONNECTOR # 6: HYBRID NAILS
  CONN  ASSIGNED   NODE  COMPONENT     CONN  ASSIGNED   NODE  COMPONENT
 NUMBER  NUMBER    NAME    LEAD       NUMBER  NUMBER    NAME    LEAD
[illegible]
```

The **Nail Data Base (.NDB)** is useful in making future changes. You can modify this report file and use it as an input in a subsequent nail assignment run, to specify nail changes.

.NDB

/*
TEMP NAIL NUMBER -> ASSIGNED NAIL NUMBER

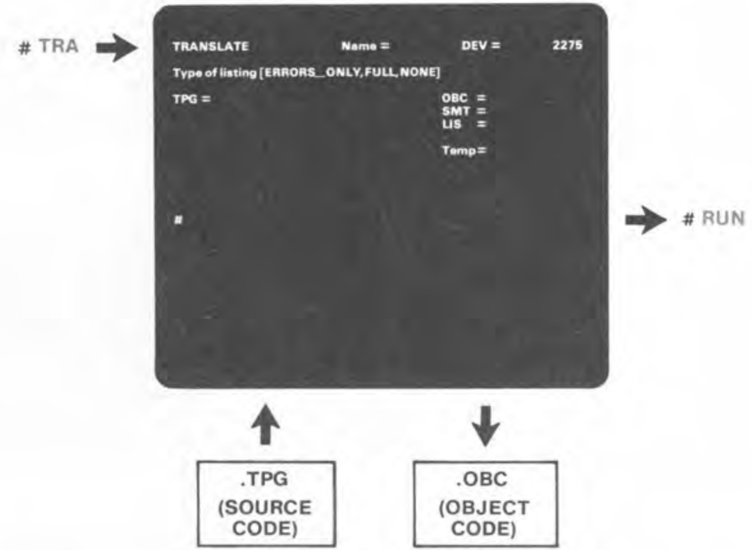
The **Nail Assignment Listing (.NLS)** is simply a progress report on the nail assignment process, as it occurs.

.NLS

```
              2271 NAIL ASSIGNMENT IN PROGRESS

                     Target Machine: 2271
                     Software ID:      NASMN275 A0
                     Mode:   NEW
                     Board Name:       R72RLY80
                     Date:   Mon 23-Aug-82 10:26:50

Process:      Copying Input TPG File
              Process completed:    0 errors detected
                                    0 warnings detected

Process:      Creating Nail Record File (.VPN)
              Process completed:    0 errors detected
                                    0 warnings detected

Process:      Creating Nail Assignment File (.VPR)
              Process completed:    0 errors detected
                                    0 warnings detected

Process:      Mapping old nails to new
              Process completed:    0 errors detected
                                    0 warnings detected
```

After nail assignment, can I translate the test program?

Yes. Simply call up the **TRANSLATE** mode options page, specify the input and output files and the type of listing you want, and then run the task.



The translator will let you know if it finds any syntax errors. Use EDIT mode for corrections. (Normally, syntax errors only exist if user modifications were made to the test program.)

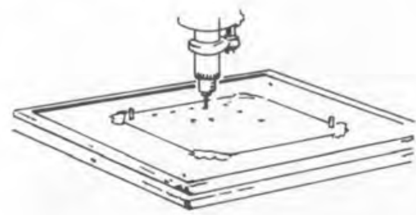
Once the errors are corrected, the translator converts the user-readable .TPG file to a machine-readable **object code file** called **.OBC.**

This .OBC file is the one used by the system when running the test program.

So much for the test program, now for the test fixture

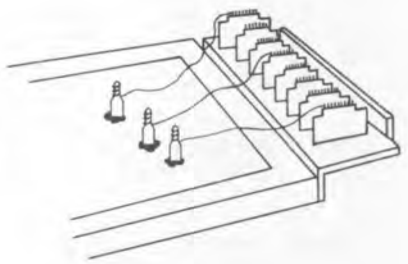
Building a fixture is strictly a mechanical operation.

First, holes are drilled in the fixture base plate to accommodate the test nail sockets. Typically, there is one hole (and one nail) for each circuit node on the board.

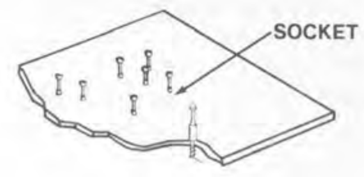


Nail sockets are inserted in these holes.

Each nail socket is wire-wrapped to the appropriate pin on the fixture connectors (the connectors that plug into the test system).



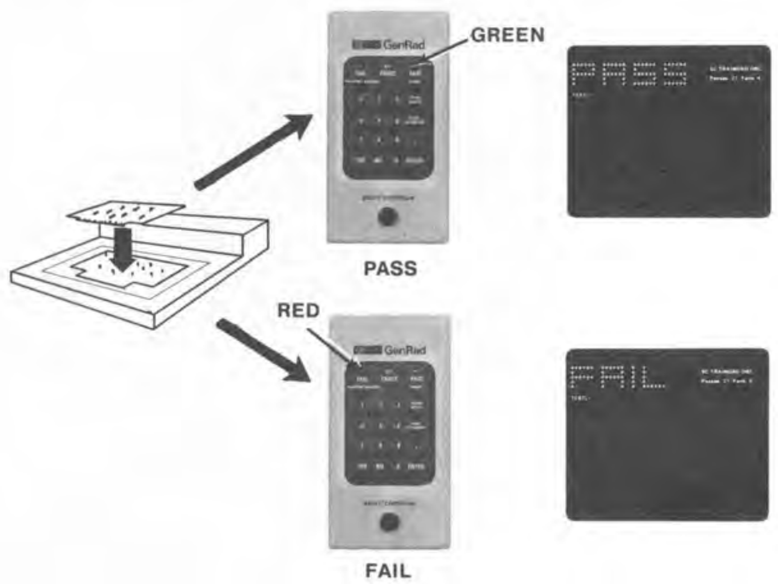
Test nails are inserted into the sockets.



OK! The test program and fixture are ready. Now what?

Now, you're ready for trial runs with several known good boards to see if your new test set is working properly.

Start by mounting the new test fixture on the system, then place a known good board on the fixture and run the test program.



If the system detects no failures, it will display a PASS message on the display screen. In systems with a portable keypad (2271 and 2272), it will also turn on the green **PASS** light on the keypad. This means that your new test set is OK for production testing.

Note: The term GREENLIGHTING is often used to indicate the successful testing of a board.

Repeat this trial run with each of the several known good boards to evaluate the performance of the test program properly.

If the system detects any failures, it will display a **FAIL** message and light the red FAIL indicator on the keypad. If this happens, you have to debug the test set.

How do I debug a test set?

The system has a special DEBUG mode available just for that purpose. You call this DEBUG mode from the DIAGNOSE mode options page at which time a **Begin label** and an **End label** option (not shown in this figure) appears on the screen.

Since debugging is usually done on one component test at a time, use the Begin Label and End Label options to specify the segment of the test program you want to work on.

```
U6:  BURST; /* (7400) */
          :
     IH(38) OL(60);
     IL(59) OH(60);
          :
     END BURST;
U7:  BURST; /* (7414) */
```

```
BEGIN LABEL = U6
END LABEL   = *
              (*INDICATES NEXT LABEL,
                WHICH IS U7)
```

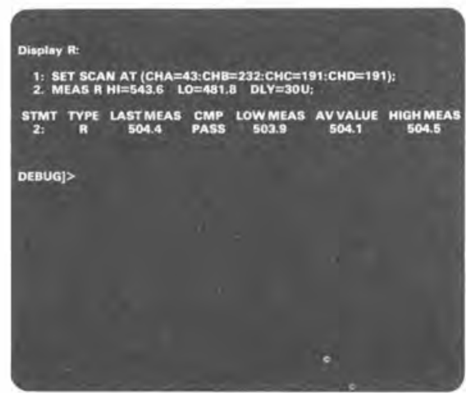
Select other options, as desired, to suit your purposes, then run the task.

```
DEBUG]> RUN
```

The system will run the entire test program in a normal fashion up to U6, then shift to debug mode for the U6: to U7: segment, displaying *detailed* results for you to analyze. After this test segment is complete, testing resumes in the normal fashion.

Debugging an analog test

After an **analog test** is complete, both the test statements and the results of each measurement are displayed.



After analyzing the test, suppose you decide that the 30 μs delay specified in statement 2 (DLY=30U) should have been 10 ms (DLY=10M) instead of 30 μs.

To make that change, type the statement number 2: followed by the new delay value DLY=10M.

```
DEBUG]> 2: DLY=10M
```

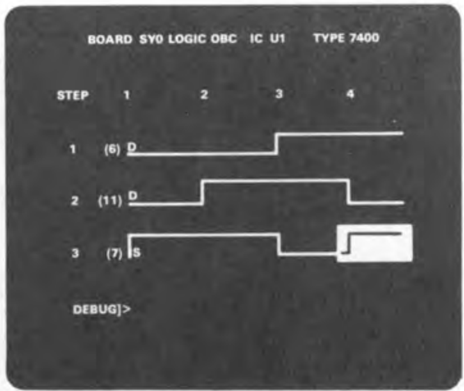
Then, rerun the test by issuing a RERUN command.

```
DEBUG]> RERUN
```

The system will rerun the test with the new delay parameter and display the new results.

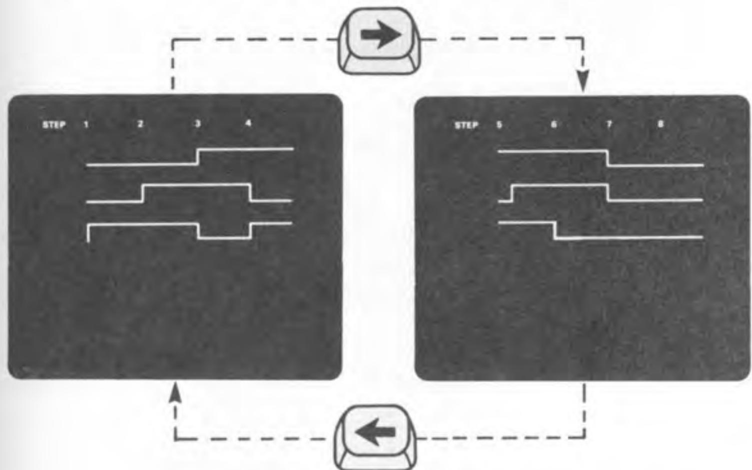
Debugging a digital test

After the tester runs a **digital test** in DEBUG mode, it automatically displays a detailed timing diagram of each test step.



The reverse video indicates an error condition. In this example, the output at nail 7 is in error because it was not expected to go high during step 4.

If there were more test steps than could be shown on the screen at one time, pressing the right arrow key would advance the timing display one frame. Pressing the left arrow key would move the timing display back one frame.



Making changes to a digital test

First, look at the program statements associated with the timing display shown on the previous page. To do this, issue an UNTRA (Untranslate) command.

After analyzing the results, suppose you decide that statement 4 is in error because it should have indicated a high output at nail 7.

Type the statement number 4: and the change OH(7) at the DEBUG prompt.

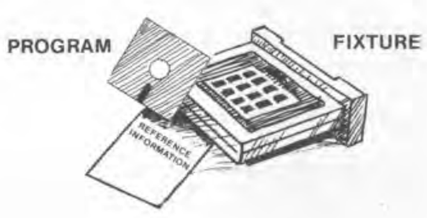
DEBUG]> 4: OH(7)

Then, rerun the test by issuing a RERUN command.

DEBUG]> RERUN

This finishes the test set preparation phase, right?

Almost. It's important at this time to make sure that the new test set is properly documented. A little extra work now could save you a lot of time later if you have to make future changes to the test set.



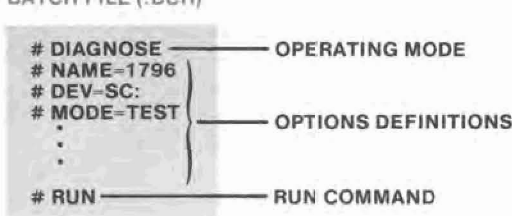
Also, you have to instruct the test operator on how to use the test set for production testing. For example, you have to tell him or her what files and fixture to use and what options to specify on the DIAGNOSE page. Include any other special instructions that the operator might need to run the test.

Setting-up for production testing

A convenient way of conveying test set-up information to the test operator is by means of a **batch** file.

A batch file defines a series of tasks for the system to perform. For each task, the batch file specifies the options to be used, followed by a RUN command.

You can use one of the system editors to write a batch file that defines all the DIAGNOSE options needed for the board test.



The system executes each statement in the file as if that statement were being typed directly from the keyboard.

To run the batch file, all you need to do is enter a period (.) followed by the name of the file. For example,

```
#.1796
```

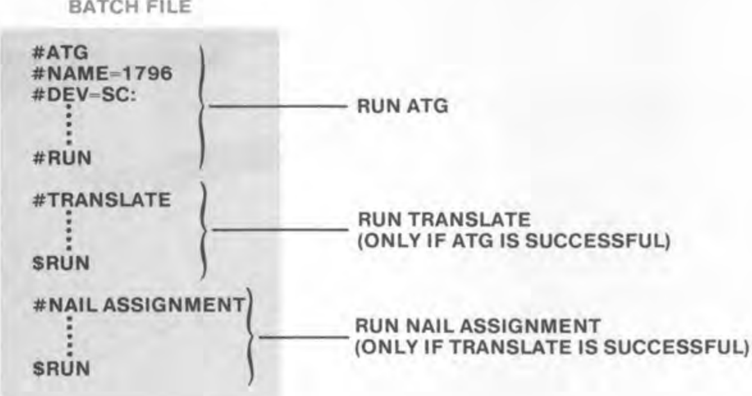
will run batch file SC:1796.BCH.

BATCH mode

BATCH mode has much wider application than that of simply defining DIAGNOSE mode options for the test operator.

For instance, it can be used to run operating modes that would normally be "off-limits" to an operator running the system in LIMITED monitor mode. It can also be used to run long jobs on an unattended system, for example, overnight jobs.

When running BATCH mode, the system automatically runs through each task in the file, sequentially. In the following example, the system would run ATG first, followed by TRANSLATE mode, then NAIL ASSIGNMENT.



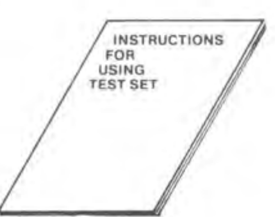
Note: If a $ precedes the RUN command, the system runs that task only if the last task was successfully completed. You wouldn't want the system to translate a program that had not made it successfully through ATG, would you?

Running the test program

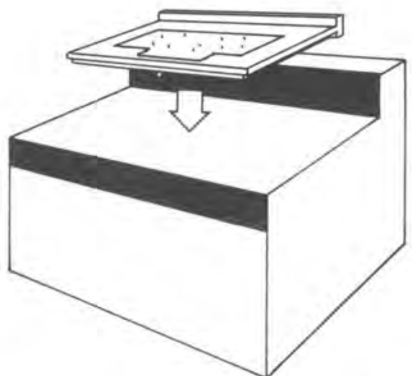
Once the programmer releases the test set to production testing, it's time for the tester to start earning its keep.

From here on, the board testing phase becomes a repetitive procedure, consisting of a few simple steps:

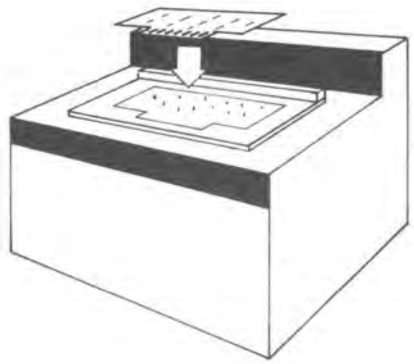
At the start of a testing session, the operator checks the instructions written for that test set.



The operator then mounts the fixture on the system,



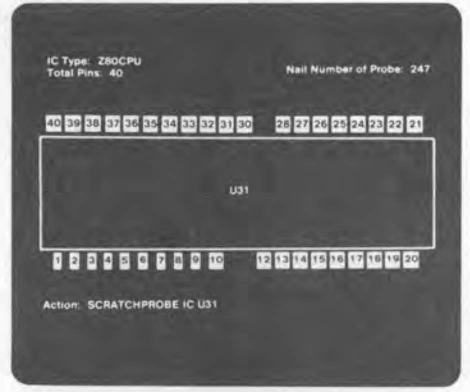
places the pc board on the fixture,



and runs the test program.

Diagnosing the test results

If the board fails, the operator performs whatever diagnostic procedures are called for by the displayed messages. For example, a Scratchprobing action may be requested (as described earlier in this book).

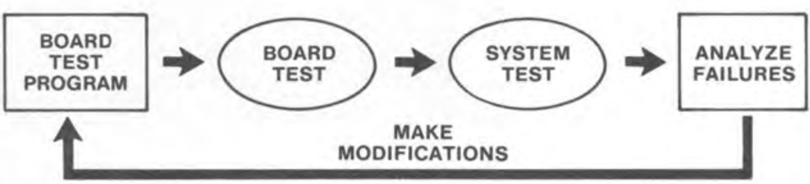


After a test run is complete, system-generated diagnostic messages can be attached to bad boards and forwarded to a repair station. Good boards can be forwarded to further assembly.

Continuing the test programming effort

It is important to monitor the production testing phase to make sure that the test program really finds all the faults that it is supposed to.

If it doesn't, analyze the failures detected at the next test stage (system test) and use the data to modify the test program, as needed.



You can also use the tester software to plot the distribution of test results for several boards. This will show how stable the analog measurements are.

An on-going effort

It's a good idea to continue working closely with the other test departments monitoring test results until all problems are resolved.

Understanding other people's test problems will help you to become a better test programmer and thus help you to use the tester more efficiently.

Glossary

Analog instrument multiplexer. A program-controlled relay matrix used to connect any analog test instrument to the instrument bus.

Archiving. The long-term storage of files on a mass storage medium, such as disk, for use at a later time.

ASCII. (American Standard Code for Information Interchange). A standard representation for encoding alphanumeric and special characters as binary values.

Assembly drawing. Document that describes the physical structure and layout of a circuit board.

ATE (Automatic Test Equipment). A system, typically computer-controlled, used to test electronic devices, pc boards and products.

ATG (Automatic Test Generation). Computer generation of a test program based solely on the circuit topology, requiring little or no manual programming effort.

Background terminal. An optional unit that allows a background task (program preparation) to take place simultaneously with a foreground task (board testing).

Backdriving. The process of forcing the output of a digital device to a selected logic level.

Base board. A fixed board in the test fixture upon which the nails and nail stops are mounted.

Batch mode. A feature whereby the system automatically executes a series of tasks predefined in a batch file.

Bed-of-nails fixture. A type of vacuum-activated test fixture in which the nodes on a pc board come in contact with the system test nails.

Binary file. A file containing binary data, not normally read by a user.

Bit. A single digit in the binary number system; can be a 0 or a 1.

Burst. A series of instructions for testing a particular digital device; executed at high speed under the control of the driver/sensor controller.

BUSBUST™ feature. A GenRad testing technique used to isolate the device(s) causing a bus failure.

Bus node. A circuit node that can be driven by more than one device.

Byte. A contiguous set of 8 bits, usually the smallest directly addressable unit in a computer memory.

Central Processing Unit (CPU). The main control portion of the computer.

Circuit description. Circuit board information supplied by the user to the tester, allowing the tester to characterize the circuit board to be tested.

Circuit description language. A high-level language in which the circuit description file is written.

CKTGEN (Circuit Generator). A software module designed to help the user input the circuit description into the tester.

Compiler. See Translator.

CPU. See Central Processing Unit.

Data logging software. Software used to accumulate data on board failures for future analysis.

Debug. In testing, the process of modifying a test set to get measured results from a known good board to agree with expected results from that board.

Diaphragm board. A movable board upon which the UUT is placed; when vacuum is applied to the fixture, the diaphragm board moves down against stops mounted on the base board, causing the UUT to make contact with the test nails.

Digital multiplexing. The sharing of a few common driver/sensors by a group of test nails (e.g., 2 driver/sensors shared by 16 nails).

Disk. A file-structured, mass-storage device which allows fast, random access of data.

Display. A CRT (Cathode-Ray Tube) device used for the display of data. When coupled with a keyboard, can also be used as a data entry terminal.

Drivers. The tester circuitry used to force selected logic levels to digital devices on the UUT.

Driver/Sensors (D/S). The tester circuits used to force logic level inputs and monitor logic level outputs from digital devices on the UUT.

Driver strobe. Defines the precise time at which an input test pattern is applied to the UUT.

D/S Controller. The tester module that controls the testing sequence during a digital test burst.

Dual fixture. A test fixture with 2 separate bed-of-nail units to speed up production testing. One board can be loaded on one fixture while another board is being tested on the other.

Editing. The process of creating or modifying a source file.

Fail. In testing, a term applied to a UUT or a device which has one or more failing test steps.

Fault. Any condition that causes a device or circuit to fail to operate in a proper manner.

Feedback loop. A circuit path that causes a device input to be affected by other devices further along the path.

File. A collection or set of records which resides on a mass storage device.

File directory. Information stored on a mass storage device describing, among other things, the name of each file on the device.

File maintenance utilities. A collection of independent software programs used to perform various useful operations on file-structured devices (e.g., copy, delete, etc.).

File-structured device. Any device which supports the storage and retrieval of data in files.

Fixed power supply. A power supply capable of producing only a single voltage level.

Fixture. The device that interfaces between the tester and the UUT. See Bed-of-Nails fixture.

Flagspecs. Special parameters and instructions supplied by the user to the ATG task.

Floppy Disk. A small, flexible magnetic storage medium.

Functional tester. ATE which tests a UUT as a complete, functional entity, typically by applying inputs and sensing outputs only through the UUT's edge connector.

Glitch. A small spurious pulse or spike on a signal line.

GO/NOGO test. A testing process which yields only a pass or fail condition.

Good board. A fault-free board.

Greenlighting. The pass condition achieved upon the completion of the debug process, or sometimes applied to any UUT which passes a test.

Ground plane. A metallized plate in the fixture which serves as a common ground point to reduce any noise generated by cross-coupling of test signals.

Guarding. In in-circuit testing, the process of ensuring that a shunt path does not interfere with the testing of a device.

Help pages. Displayed information describing how to use various features and options in the tester.

High impedance state. See Tri-State.

IC. See Integrated circuit.

IEEE-488 Bus. A data transmission bus which provides communication between the tester and external devices.

In-circuit tester. ATE which tests each separate device on a board by applying test signals directly to the device's inputs and sensing the results directly from the device's outputs.

Initialization. The process of applying inputs to a device or circuit until known states are obtained.

Input vector. A set of logic values to be applied to the complete set of input test nails at any one point in time.

Instrument bus. Four common lines (or channels) to which any analog test instrument can be connected via the Multiplexer, and any UUT circuit node can be connected via the Scanner.

Integrated circuit (IC). An array of interconnected circuits integrated into a single chip.

Interstrobe Time (IST). The time between a driver strobe and a sensor strobe.

Inter-Test Time (ITT). The time between two successive driver strobes.

Intermittent fault. A fault whose effect on a circuit appears and disappears at seemingly random intervals.

Keypad. Small keyboard that contains the controls and indicators typically used for running a test.

Known good board. A circuit board which is verified to be fault-free.

Learning. The process of determining the expected outputs from a device by applying inputs to a known good board and measuring the actual outputs.

Library. In in-circuit testing, a collection of predefined test programs or circuit descriptions for various types of components, all stored on a mass storage device.

Logic diagram. See Schematic.

Logic levels. Voltage levels representing a logic 0 and a logic 1.

Looping. The repeated execution of a sequence of statements.

Machine code. Binary patterns (ones and zeros) representing computer instructions.

Mass storage device. A medium used for the permanent storage of large amounts of data (e.g., magnetic disk units).

Monitor. The software module under control of the Operating System, which displays options and interprets commands from the user.

Multiplexer (MUX). See Analog multiplexer.

Multiplexing. In a tester, multiplexing refers to the sharing of test instruments among test nails to reduce the size and cost of the system.

Nails. Spring-loaded metal probes used in a bed-of-nails test fixture to make electrical contact with the nodes on a circuit board.

Nail assignment. The replacement of temporary nail numbers assigned in a circuit description with the target nail numbers used for the actual wiring of the fixture.

Node. The electrical interconnection between two or more device leads.

Object code. See Machine code.

Open. A fault which causes 2 electrically connected points to become separated.

Operating System (OS). The software module that supervises the operation of all other modules in the computer system.

Output vector. The set of logic values, either expected or measured, for all output pins of a UUT or device at a particular test step.

Pass. In testing, a term describing a device or a UUT which has no failing test steps.

Peripheral device. Any input/output device, such as a disk unit or line printer, that is connected to a computer system.

Power-up reset. Circuitry that automatically sets a device or circuit into a known state when power is applied to it.

Printer. A peripheral device that produces a hard-copy output.

Probe. A device used as a movable test nail to monitor various nodes on the UUT.

Programmable power supply. A power supply capable of having its voltage(s) programmed or selected over a range of values.

Programming station. A test system which does not contain any test hardware, typically used for test program development.

Receiver. That part of the interface between the tester and the fixture that is permanently attached to the tester.

Run-Time System. The collection of software programs required to perform the actual testing and diagnosis of a UUT.

Scanner. A program-controlled relay matrix used for connecting any UUT circuit node to the analog instrument bus.

Schematics. The set of drawings which shows the elements of a circuit and how they are interconnected.

SCRATCHPROBING™ technique. A GenRad diagnostic feature that directs the user to lightly move (scratch) the probe along the pins of an IC to check for continuity.

Sensors. Tester circuits used to monitor logic level outputs.

Sensor strobe. Defines the precise time at which output responses from the UUT are measured.

Short. A fault which causes two or more normally electrically separated points to become connected.

Sink. Typically refers to current flowing into a power source.

Source. Typically refers to current flowing out of a power source.

Source file. A file consisting of alphanumeric and special character data encoded in a standard format, such as ASCII.

Target nail. Actual nail numbers assigned by the nail assignment software to replace the temporary nail numbers assigned by the user in the circuit description.

Temporary nail. Arbitrary nail numbers assigned by the user when developing the circuit description file.

Test language. High-level language used to write the test program.

Test nail. See Nails.

Test program. The set of instructions to the tester which controls the testing of the UUT.

Test set. The unique combination of test program and test fixture used to test a particular UUT.

Test step. The application of a single input/vector.

Translator. A software module which analyzes and converts the test program from high-level test language to binary machine code. The input to the translator is a source file; the output is an object-code file.

Tri-state (or 3-state). In addition to a high and low state, some devices such as bus drivers have a third (high-impedance) state. This third state effectively disconnects the device output from all other circuitry on the board.

Truth table. A table showing the expected outputs from a digital device for all possible combinations of logic inputs to that device.

UUT (Unit-Under-Test). A term applied to any circuit board which is being tested by ATE.

Vector. See Input vector or Output vector.

Winchester unit. A high-speed mass-storage disk unit with non-removable disk.

Production Test Division
300 Baker Avenue
Concord, Massachusetts 01742

USA

***Boston** • (Sales) Waltham, MA 02254
Tel (617) 890-4900 • TWX 710 324 0893
(Service) Concord, MA 01742
Tel (617) 369-4400 • TWX 710 347 1051

***Chicago** • 1156 Shure Drive, Arlington Heights, IL 60004
Tel (312) 577-1881 • TWX 910 291 1209

***Cincinnati** • Suite A, 2002 Ford Circle, Milford, OH 45150
Tel (513) 831-9210 • TWX 810 460 8323

***Colorado Springs** • 6180 Lehman Drive, Suite B207, Colorado Springs, CO 80907
Tel (303) 593-8238

***Dallas** • 1121 Rockingham Lane, Suite 100, Richardson TX 75080
Tel (214) 234-3357 • TWX 910 867 4771

***Los Angeles** • P.O. Box 19500, 17361 Armstrong Avenue,
Irvine Industrial Complex, Irvine, CA 92714
Tel (714) 540-9830 • TWX 910 595 1762

***New York** • 22-08 Route 208, Fair Lawn, NJ 07410
Tel (NJ) (201) 797-8001, (NY) (212) 964-2722 • TWX 710 988 2205

***Orlando** • 3751 Maguire Blvd., Suite 170, Orlando, Florida 32803
Tel (305) 894-4303 • TWX 810 850 0270

***San Francisco** • 2855 Bowers Avenue, Santa Clara, CA 95051
Tel (408) 727-4400 • TWX 910 338 0291

***Washington, D.C.** • 1701 Research Boulevard, Rockville, MD 20850
Tel (301) 424-6224 • TWX 710 828 9783

CANADA

***Toronto** • 307 Evans Avenue, Toronto, Ontario M8Z 1K2
Tel (416) 252-3395 • TLX 06-967624
Montreal • Tel (514) 747-1052 • TLX 05-826652

EUROPE

***England** • Norreys Drive, Maidenhead, Berkshire SL6 4BP England
Tel (0628) 39181 • TLX 851-848321

Best • Tel (0) 49984240 • TLX 844-59535

***Milan** • Tel (02) 8466541 • TLX 843-320373

***Munich** • Tel (089) 41690 • TLX 841-529917

***Paris** • Tel (01) 7970739 • TLX 842 220991

Rennes • Tel (099) 535154

***Rome** • Tel (06) 455839

Wiesbaden • Tel (06121) 370057 • TLX 841-04186489

***Zurich** • Tel (01) 552420 • TLX 845-53638

ASIA

Japan • Tokyo Electron Limited, Shinjuku-Nomura Bldg.
1-26-2 Nishi-Shinjuku, Shinjuku-Ku, Tokyo 160, Japan
Tel 343-4411 • TLX 232-2240 Labtel J

Other countries in Asia and Pacific • Tel 344-9245 (Tokyo)

LATIN AMERICA

Contact WALTHAM, MA 02254 USA
Cable GENRAD Waltham (MASS.)

*Service Facilities Available

Part No. 2275-0102-A

Printed in USA