

Z1800-Series

Programmer's Guidebook



Teradyne, Inc
Assembly Test Division
2625 Shadelands Drive, Walnut Creek, California 94598-2597
Part Number M00-098-00 Copyright Teradyne, Inc.

Product Warranty

THE STANDARD TERADYNE WARRANTY CONSTITUTES THE ONLY REPRESENTATION OR WARRANTY MADE BY TERADYNE WITH RESPECT TO ANY EQUIPMENT, GOODS OR SERVICES SUPPLIED BY TERADYNE. TERADYNE MAKES NO OTHER WARRANTIES OR REPRESENTATIONS, EXPRESSED OR IMPLIED, IN FACT OR IN LAW, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL TERADYNE BE LIABLE FOR INCIDENTAL, SPECIAL OR CONSEQUENTIAL PENALTIES OR DAMAGES, INCLUDING LOST PROFITS, OR PENALTIES AND/OR DAMAGES FOR DELAY IN DELIVERY OR FAILURE TO GIVE NOTICE OF DELAY, EVEN IF TERADYNE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Due to an ongoing policy of constantly updating equipment and procedures, the contents of this document are subject to change without notice.

Teradyne assumes no responsibility for errors or for any damages that result from the implementation of the procedures described in this publication. Teradyne also reserves the right to make changes in its products without incurring any obligation to incorporate such changes in units previously sold or shipped. Teradyne makes no commitment to update nor to keep current the information contained in this document.

Teradyne assumes no responsibility for the use of any circuitry other than the circuitry embodied as a Teradyne product. No other circuit patent licenses are implied.

This software system consists of computer software and documentation. It contains trade secrets and confidential information which are proprietary to Teradyne, Inc. Its use or disclosure in whole or in part without the express written permission of Teradyne, Inc. is prohibited.

This software system is also an unpublished work protected under the copyright laws of the United States of America. If this work becomes published, the following notice shall apply:

Copyright © 1994, 1995, 1996, 1997, 1998, 1999, 2000 Teradyne, Inc. All Rights Reserved

Trademarks

The following are trademarks or registered trademarks of Teradyne and may be used to describe only Teradyne, Inc., Assembly Test/Walnut Creek products:

Z850	Z1860VP®	FrameScan Plus
Z875	Z1866	HostLink
Z8000	Z1880	MultiScan II
Z8500	Z1884	PRISM-Z
Z1800	Z1888	ProcessWatch
Z1803 and Z1803 Plus	Z1890	Safecracker
Z1805	APC	Test Toolbox
Z1808	AutoLoad	Tester-Aided Instruction
Z1820	Boundary Scan Intelligent Diagnostics (BSID)	TestQA
Z1840	BusScan	VICTORY
Z1840VP®	CapScan	WaveScan
Z1850	DeltaScan	Z1800-Series Manufacturing Test Platform
Z1850VP®	Digital Function Processor (DFP)	
Z1860	FrameScan	

Other product names are trademarks of their respective owners.

Z1800-SERIES PROGRAMMER'S GUIDEBOOK

Manual History

- Fourteenth Edition, June 2000, G.0 system software
- Thirteenth Edition, December 1998, F2b system software
Component Test Reference and Board Test Tutorial have been removed and are separate documents. Former volumes 1 and 2 are combined into one volume.
- Twelfth Edition, July 1996, F.1 system software
- Eleventh Edition, November 1995, F.0 system software
- Tenth Edition, April 1995, E.4 system software
- Ninth Edition, November 1994, E.3 system software
- Eighth Edition, August 1994, Version E.2 system software
- Seventh Edition, May 1994, Version E.1 system software
- Sixth Edition, February 1994, Version E.0 system software
- Fifth Edition, December 1992, Version D.2 system software
- Fourth Edition, August 1992, Version D1.1 O/S.
- Preliminary Fourth Edition, May 1992, Version D.1 O/S.
- Preliminary Third Edition, March 1992, Version D.1 O/S.
- Preliminary Second Edition, December 1991, Version D.0 O/S.
- First Edition, September 1991, Version C.1 operating system.

CONTENTS

Preface

About the Z1800-Series Manual Set	xi
Quick Help	xii
Tester Safety Issues for Programmers	xiii

Chapter 1—Introduction to System Software

Starting with the Basics	1-1
The Menus	1-1
The Windows	1-2
Saving Your Work	1-4
Quitting from a Menu	1-5
Understanding the Directory and File Systems	1-6
MS-DOS	1-6
File and Directory Structure	1-6
\MOS Files	1-7
\PGT Files	1-9
\TPD Files	1-9
Viewing the Main Menu	1-11
Setting Logins and Permissions	1-14
Setting Up Logins after Installation	1-14
Changing Permissions	1-16
Default Permissions	1-17
Changing Passwords	1-18
Setting the Default Login	1-19
Using the Files Menu to Manage Programs	1-20
Selecting Programs	1-21
Selecting ICT/FST Program Types	1-22
Toggling Between Test Program Directories	1-22
Removing a Program	1-22
Creating a New Program Directory	1-23
Copying a Program Using Multipanel	1-24
Converting to Different Software Versions	1-25

Chapter 2—System Setup

Set Up Overview	2-1
Setting Up Data Paths	2-2
Path Data Fields	2-2
Changing Program Directories	2-3
Setting Up Communication Channels	2-3
Setting Up System Variables	2-8
Setting Up Environment Variables	2-11
Setting Up General Factory Interface	2-12
Setting Up Validate	2-16
Setting Up Serial Communications	2-20
Com Parameters	2-21
Changing Screen Colors	2-22
Using the Color Window	2-22

Chapter 3—Editing Overview

Navigating in Step Worksheets	3-1
Navigating in the 18xx Editor: Mouse	3-1
Navigating in the 18xx Editor: Keyboard	3-1
Navigating in Options Fields Using the Keyboard	3-2
Introduction to Programs, Categories, and Test Steps	3-3
Introduction to Step Worksheets	3-5
Editing Basics	3-5
Relationships Between Test Page Elements	3-7
Categories/Sections/Test Types	3-8
Component Select Menus	3-11
Revision Control	3-11
Enabling Revision Control	3-11
Header/Trailer Step Worksheet Editing	3-13
Editing Headers and Trailers	3-13
General Variables	3-17
Header/Trailer Message Step Fields	3-19
Merging	3-21
Step Worksheet Editing	3-23
Step Worksheet Menu Bar	3-25
Analog Step Worksheet	3-25
Analog Test Properties Editing	3-30
Gray Code Step Worksheet	3-38
Vector Step Worksheet	3-41
Setting the Step Worksheet Options	3-42
Test Step Controls	3-49

Chapter 4—Controlling Test Execution

Understanding the Test Environment	4-1
Program Flow Tools	4-4
Setting Up the Environment	4-4
Test Step Flow Control	4-6
Flow Control Options	4-6
Other Test Step Flow Controls	4-10
Using Test Control Buttons	4-10
Using Option Flags Switches to Control Flow	4-14
Setting Up a Branching Condition	4-14
Setting Up Chaining	4-16

Chapter 5—Test Techniques and Strategies

Introduction to Test Applications	5-1
Developing a Program Strategy	5-2
Understand the Tester's Capabilities	5-2
Momentum	5-3
MultiScan	5-3
Prism-Z	5-4
Understanding the Board	5-4
Evaluating the Test	5-5
Auditing the Program	5-6
Power Control and the Effects on Board Test	5-6
Power Supply System Architecture	5-6
Power and Program Execution	5-10
Power Step Worksheets	5-10

Updating D.X Programs to Run With Programmable PS	5-13
Analog and Digital Switching	5-13
Analog Test Techniques	5-17
Analog Measurement Theory	5-17
Analog Wire Mode and Measurement Concepts	5-21
ATB Test Configurations	5-26
Analog Wire Modes	5-27
Digital Test Techniques	5-36
Digital Guarding	5-36
Disables and Guards in Digital Tests	5-37
Disables and Guards in Vector Tests	5-39
Summary of Stimulus Hierarchy	5-40

Chapter 6—Program Generator Input List Reference

Introducing the Program Generator Menu	6-1
Overview of the Pgen Menu System	6-3
Understanding Input List Structure	6-4
IPL Records	6-4
IPL Tokens	6-5
Control Tokens	6-6
Creating Test Programs with C.1 Input Lists	6-25

Chapter 7—Program Generator Tools

Using PGEN.CFG to Reassign Default Values	7-3
PGEN.CFG Hierarchy of Influence	7-3
How to Edit PGEN.CFG	7-3
Setting Up PEP for Input List Analysis	7-11
Using Clean and Build	7-12
Clean	7-12
Build	7-12
Learning Interconnect Information	7-12
Generating the Program	7-14
Automatic Program Generation	7-14
Incremental Program Generation	7-14
Combining Parallel Components	7-15
Manual Mode	7-16
Automatic Mode	7-16
Updating the Test Program	7-17
Validating the Test Program	7-17
Generating Reports in Pgen	7-18
Managing Topology	7-19

Chapter 8—Managing Template Libraries

Managing the Template Library	8-1
Setting Up Library Permissions	8-2
Creating a New Library	8-2
Using Templates Menu Edit Functions	8-3
Extracting Templates	8-6
Accessing Template Revisions	8-9
Listing Aliases and Templates	8-12
Translating Vectors	8-13
Creating Templates	8-17
Analog Templates	8-17

Gray Code Templates	8-18
Vector Image Templates	8-20
Creating Models Using ASCII Vector File Structure	8-20
NPins Section	8-20
Test Configuration Section	8-22
Vector Section	8-23
Guard and Disable Sections	8-26
Tabular/IVL Formats	8-26

Chapter 9—Test and Debug Tools

Test Development Tools	9-1
Using the Test Jack Panel in Debug	9-7
Test Envelope and Listen Window	9-9
Using Test Jacks in Digital Debug	9-10
Displaying Test Results	9-10
Using Test Jacks in Analog Debug	9-11
Using Digital Tracer to Test Connectivity	9-11
Setting Up Tracer in PRGMVARS	9-12
Evaluating Test Quality with Fault Inject	9-17
Digital Test Quality	9-17
Evaluating Fault Coverage at Board Level	9-24
Interpreting the Fault Coverage Report	9-25
Fault Coverage Techniques	9-31
Supplemental Fault Coverage	9-33
Software Utilities	9-35
Learning Component Lead Node	9-35
Embedded Step Worksheet Nodefinder	9-36
Locating a Node Number on a Board	9-38
Learning a CRC	9-38
Locating a Driver/Receiver Board	9-39
DOS Shell	9-40
Diagnostics	9-40
Supplemental Diagnostics	9-40
Calibration	9-40
Documenter	9-40
Global Editor	9-44
Hardware Configuration	9-71
Validate	9-73
Validate Process	9-73
Establishing Guards	9-75
Other Options	9-79

Chapter 10—Special and Advanced Applications

Measuring Frequencies	10-1
Example 1—Higher Frequencies.....	10-1
Example 2—Middle Frequencies.....	10-1
Example 3—Lower Frequencies	10-1
Example 4—50,000 Hz.....	10-2
Testing Switches.....	10-2
Testing Relays	10-3
Testing SCRs/Triacs	10-4
Testing Operational Amplifiers.....	10-6
Developing a Bus Test.....	10-7

Testing Gates.....	10-8
Testing Flip-Flops	10-9
Testing Counters.....	10-10
Testing Line Transceivers	10-11
Testing Memory Devices	10-12
Burst Time and Duty Cycle Supervision for Backdrive Protection	10-13
Effects of THC and VP on Backdrive.....	10-15
Accessing External Programs	10-15
Using Test Type	10-15
Using Pre-Test or Post-Test Options Menu.....	10-16
Using IEEE Instrumentation.....	10-18
Digital Function Processor	10-21
Setting Up DFP in PRGMVARS	10-21
Generating DFP Test Properties	10-23
Test Properties Fields.....	10-24
External Synchronization & VP Tests	10-25
VP External Clock.....	10-25
VP Hold	10-26
Multipanel Testing.....	10-27
Entering Master Program Data.....	10-27
Entering Panel Program Data.....	10-28
Editing PANEL.DAT	10-33
Vector Performance Strategy Options	10-34
ASCII/VLSI/PAL Strategy Option Summary	10-35
VLSI Model Exists in Library.....	10-35
VLSI Model Resembles the Library Model	10-36
ASIC/VLSI Patterns from CAE Simulation.....	10-37
ASIC/VLSI Patterns from Another Tester	10-39
Teradyne Z8000-Series In-Circuit Tester: Protocol Test	10-39
Teradyne Z8000-Series Vector Test	10-40
GENRAD 22XX In-Circuit Testers	10-40
Factron 3XX In-Circuit Testers	10-40
Teradyne J941, J971 Component Testers	10-41
Sentry 20 Component Testers.....	10-42
ASIC/VLSI/PAL Functional Descriptions Available.....	10-43
ASIC/VLSI/PAL Board Schematic Only	10-44
PAL JEDEC Test Pattern Available	10-45
PAL Equation Available	10-45
Victory Boundary Scan Test Vector.....	10-46
ASIC Test Pattern Design Criteria	10-46
Disable Requirements	10-46
Test Pattern Restrictions	10-47
Pattern Coverage Guidelines	10-48
Developing an ASIC Test.....	10-48
Overview.....	10-48
ASIC Test Development Process	10-49
Transporting Z875/Z850 Test Applications	10-52
Examine the Fixture.....	10-52
Transfer and Edit the Input List	10-52
Generate the Program.....	10-52
Debug the Program	10-52

Appendix A

18XX Error Messages	A-1
C-SCAPE ERROR MESSAGES	A-30

Appendix B

Command Line Options	B-1
----------------------------	-----

Index

PREFACE

About the Z1800-Series Manual Set

Your Z1800-Series tester comes with an extensive array of technical manuals to support the system hardware and software. They are listed below with brief descriptions of contents.

Z1800-Series System Software Release Notes contain information about the features and performance characteristics of the latest software release including corrected software deficiencies, software warnings and problem reports, user documentation notes, and C.X to G.X program update notes.

The **Getting Started** quick reference guide tells you at a glance where to look in the documentation set for particular information. It also provides summaries of several basic procedures to get you started using the 18xx system software.

The **Z1800-Series Programmer's Guidebook** contains the information programmers need about Z1800-series system software to develop test applications for their boards. A test application is a complete test solution comprising a fixture and a program.

The 18xx software environment speeds test program development by using language-free, menu-driven worksheets for every test step.

The Programmer's Guidebook covers both digital and analog test. If you have an analog-only test system, information regarding digital test is not pertinent to your test situation, and you will notice in the 18xx system software that digital functionality is not available. However, the analog functionality documented here and in other Z1800-Series manuals is the same for analog-only test system as for any other test system in the Z1800-Series family of testers.

The Programmer's Guidebook contains the following chapters:

- Chapter 1, "Introduction to System Software." Menu format and operation, directory and file system, logging in and setting permissions.
- Chapter 2, "System Setup." Using the Setup menus to establish system parameters and global controls.
- Chapter 3, "Editing Overview." Software navigating techniques, analog and digital worksheet editing.
- Chapter 4, "Controlling Test Execution." Specifying program flow of control, branching, chaining.
- Chapter 5, "Test Techniques and Strategies." Introduction to test applications, program strategy, power control, analog and digital test techniques.
- Chapter 6, "Program Generator Reference Input List Reference." Introduction to the pgen menu, input list structure, and creating test programs.
- Chapter 7, "Program Generation Tools." Introduction to the tools available in the Pgen menu, including pgen.cfg.
- Chapter 8, "Managing Template Libraries." Managing and creating libraries and templates, ASCII vector language format.
- Chapter 9, "Test and Debug Tools." Test jacks, Digital Tracer, Fault Inject, software utilities, including Global Editor.
- Chapter 10, "Special and Advanced Applications." Frequency measurements, Multipanel test, clock synchronization for vector tests, vector performance strategy options, among other topics.
- Appendix A. "Error Messages"
- Appendix B. "Command Line Functions"

The Z1800-Series tester-specific System References provide safety information for technicians and explain and illustrate the tester's physical characteristics, and installation.

Note that information about the Relay Array Board (RAB), formerly in the **Relay Array Board Option**, has been incorporated into the Programmer's Guidebook, Chapter 3, and the Maintenance Reference.

The **Z1800-Series Component Test Reference** is an alphabetical listing of testable analog and digital devices, menu access, and test editing.

The **Z1800-Series Board Test Tutorial** is a step-by-step guide to developing a test program based on the self-test assembly.

The **Z1800-Series Maintenance Reference** provides information about EMC and electrostatic prevention, diagnostics, periodic and preventive maintenance, and specifications. For calibration support call 1.800.HLP.TEAM (1.800.457.8326).

The **Parts Lists Manual** and **Engineering Reference Drawings Manual** relevant to your system contain parts lists for your tester and engineering reference drawings.

The **Z1800-Series Operator's Guide** contains the information operators need to perform board production test. Always consult the operator's guide for safety cautions and operating instructions before using the tester.

The **ProcessWatch User's Guide** tells you how to use the ProcessWatch™ test data collection and reporting system.

The **MultiScan User's Guide** contains information programmers and technicians need for developing MultiScan component tests and operating and maintaining FrameScan, WaveScan, and DeltaScan hardware.

The **Z1800-Series Fixturing Guidebook** provides fixture wiring information for program development and guidelines for building fixtures.

The **Z1800-Series Template Library Manual** contains library installation information in addition to a complete listing of Gray code and VLSI templates.

The **Z1800-Series Computer Configuration and Software Installation Guide** gives up-to-date information on how to configure your PCI or ISA PCIO adapter board, modify AUTOEXEC.BAT and CONFIG.SYS files, and install System software.

Technical Support Bulletins (TSBs) contain the latest information about problems with hardware or software, applications, and other important test issues.

Documentation about the installation and use of optional hardware and software is provided in the options kits.

Quick Help

The F1 key is available at all times for on-line help. When you press F1, the Quick Help utility will provide all help messages and cross references in an easy-to-use, menu-driven format. To exit back to the Z1800-series programming environment press *x* or select Exit QuickHelp from the Files menu.

Tester Safety Issues for Programmers

CAUTION: Consult the safety sections of your System Reference and Maintenance Reference for all cautions before attempting to program the tester. Failure to do so could result in damage to the tester, the board under test, and to test personnel.

Verifying Fixture Power Wiring

Your fixture's power supply wiring connects the board-under-test to significant amounts of electrical power. Damage to the board, fixture, and tester can result from incorrect power wiring and programming. Take the time to check wiring thoroughly before turning on a supply with a loaded board on the fixture.

If test results are not what you expect, you may have to troubleshoot the fixture. While troubleshooting, be prepared to release the vacuum from the fixture and receiver in the event of a serious problem. Press the Cancel button to reset the tester and its power supplies.

If you suspect that relays in the tester have been damaged by a power wiring accident, run the diagnostics and the fixture self-test program before continuing.

Extensive information about fixturing may be found in the **Z1800-Series Fixturing Guide** including a troubleshooting section and application notes.

Transformer Testing

Stimulating one coil of a transformer may produce much larger and possibly hazardous voltages at the secondary coil of the transformer. Therefore use caution when testing a transformer with Test V Stim V from the Analog worksheet.

Programmable Power Supplies

External power supplies should never be connected to the digital drivers.

Section Abort and Debugging

There is potential for damage on new production boards if you execute a test with Section Abort off while debugging. Edit the Header Flags page to turn Section Abort on.

Power Supply Selection

The system software does not prevent you from using the +5 volt power supply to power devices that should be tested at 3 volts. Be sure that you have set up your program's PGEN.CFG file to prevent such an occurrence.

Voltage for Logic IC Testing

The +5 power supply is ordinarily used to power logic ICs on the board-under-test. The Z1800-series' drivers are designed to operate with this supply only. Never test logic ICs at any other voltage.

Turning Power Off Under Emergency Conditions

In an emergency, you can quickly turn the power off while a board is under test:

- Press Cancel, either on the keyboard, screen, or tabletop.
- Turn off the tester power.



1 INTRODUCTION TO SYSTEM SOFTWARE

Chapter 1 is an introduction for programmers and supervisors to menu-based operations of Teradyne Z1800-Series testers. It also provides an overview of the directory and file system. The Windows-based software comprises a set of menus, windows, and worksheets from which you may easily make selections for program creation, modification, and execution. The 18xx system software delivers programs with a minimum of development time for precision measurement of a large number of device types.

The menu-based design and color formats enable you to quickly view and select the many software features. You can also change menu colors and access many debugging tools from Pulldown or Pop-up menus. Worksheets offer a simple arrow-toggling technique for changing test steps and for viewing the pages of a multipage test.

The useful test controls such as Start and Cancel are available on primary screens, as well as on the keyboard and tester console.

A Board Name field makes program identification easy, and a test Status Line immediately reports the results of test debugging runs.

When you feel comfortable with the menu-based operations explained in this chapter, refer to the **Z1800-Series Board Test Tutorial** for a step-by step program development tutorial.

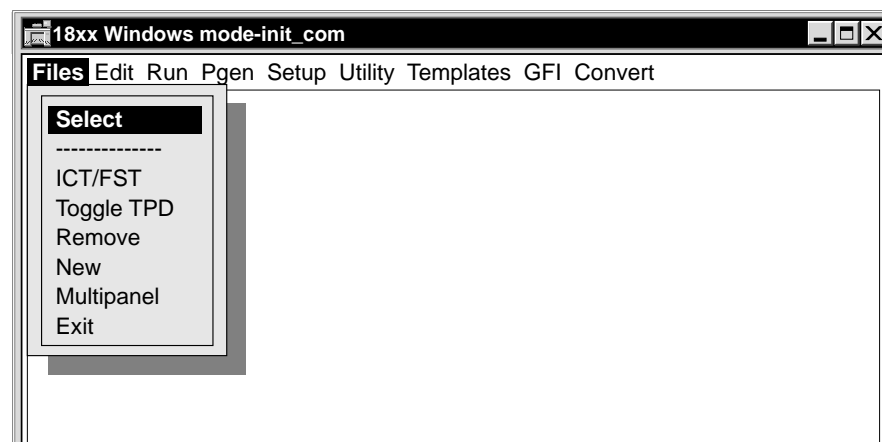
Starting with the Basics

The 18xx software is a DOS-based, menu-driven system which can operate either in a Windows[®]95 or DOS environment. The graphical user interface consists of menus and a variety of windows. You can use the mouse or the keyboard to perform operations in the interface. See chapter 3 for information about mouse and keyboard operation in worksheets.

The Menus

Commands are presented either in horizontal menu bar or vertical pulldown formats. The mouse is the preferred way to select from a menu, and navigational instructions in the document assume that you are using a mouse. However, you can also use the keyboard or use both methods in combination. Users familiar with mouse operations will have no trouble with clicking and pulldown maneuvers.

When you click one of the Main menu categories such as Files, for example, the Files pulldown menu appears.



To select a Main menu item using the keyboard, simply type the item's red letter (the default color)—often the first letter of the menu item. To select the Files menu, for example, you type the letter “F”. To exit the Main menu and enter the DOS environment, you type “X”.

Keyboard selection letters are red by default, but you may change their color in the Color setup menu. Refer to the Color menu in the section, Setting Up System Controls, in chapter 2.

You can also move through the menus with the arrow cursor keys. Press Enter to select the highlighted menu or menu item.

The following table shows the letter to type when selecting from the Main menu. All menu bars permit keyboard selection by typing a single red letter. (Keyboard selection by letter is not case-sensitive.)

Type this Letter	To Get this Menu
F	FILES
E	EDIT
R	RUN
P	PGEN
S	SETUP
U	UTILITY
T	TEMPLATES
G	GFI
C	CONVERT

You can also use the keypad's left, right, Home, and End cursor keys to move through the menus and to select commands. The cursor movements wrap around. For example, with the right cursor on the right-most selection, press the cursor and it wraps to the left-most selection. Press the Enter (Return) key to complete the selection process when you use the cursor keys to select.

See chapter 3, “Editing Overview,” for more information about using the mouse and keyboard to navigate in the 18xx editor.

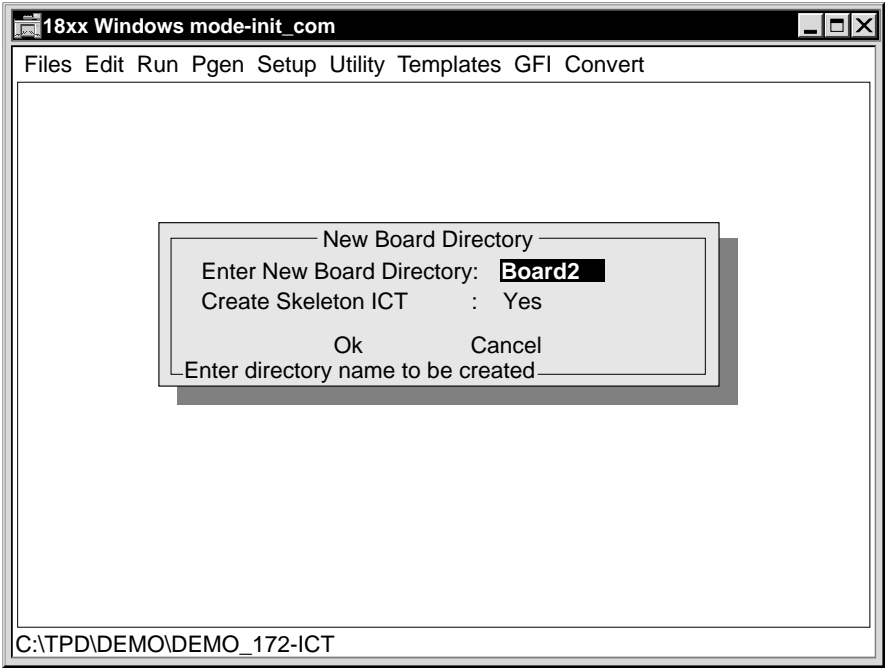
The Windows

The system software is comprised primarily of the following

- Dialog boxes
- Process windows
- Pop-up windows

A **dialog box** requires that you respond by typing information, and/or selecting an answer or command. It can be as complex as a worksheet or as simple as a window that requires only one click.

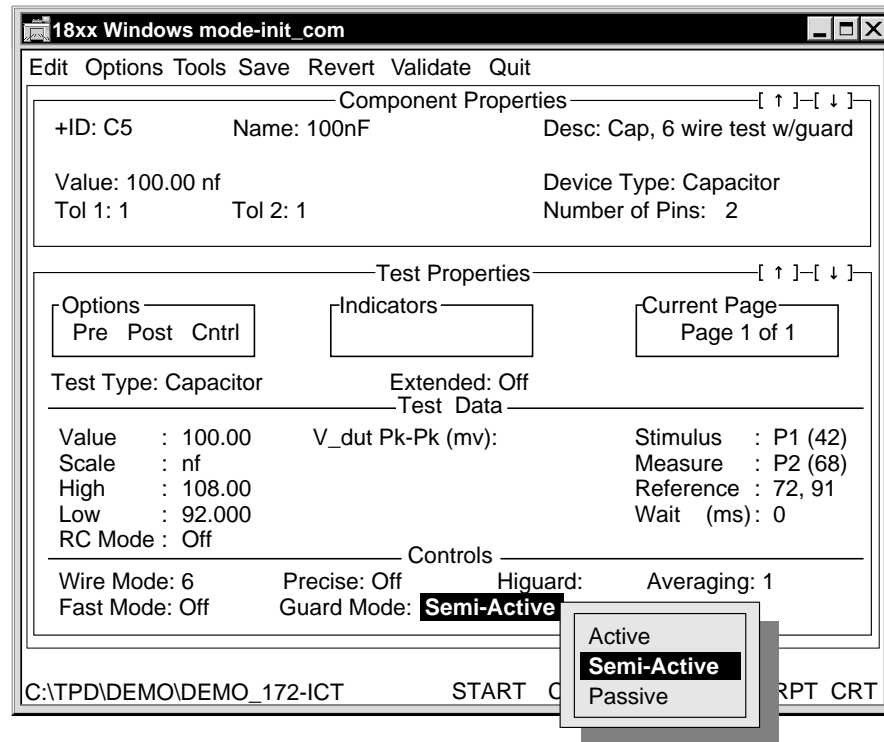
For example, when you select New as shown below, the tester prompts that you either type the new board name or select Cancel.



Most **process windows** are visible for the short period of time required to communicate information. For example, the window “Removing the Board Directory” shown below is a process message that appears for a short time when you remove a board directory.



As a result of user action, a **pop-up window** overlays a portion of the window to which it applies. Pop-up windows generally display a list of selections from which you can make a selection. When you make your selection and move control back to the primary window, the pop-up window disappears.



You can move and/or resize some windows. To move a window, click its border and hold down the mouse button while you drag the window to a new location.

To change the size of a window, click one of the window's corners, hold down the mouse button, and drag to enlarge or decrease the window's size.

Many windows display a status line along the bottom edge which contains a brief instruction for the window or the current field.

Saving Your Work

Saving your work is essential. With regard to saving, the 18xx system software has two modes of functionality—setup and program development. Steps to save your system setup differ from those for saving your program edits.

Saving System Setup

The Setup menu has two levels of saving—to memory and to disk. The various windows in the Setup environment—Data, Environment, GFI, Validate, Color, Serial, and the login windows—have OK buttons to enable you to save changes to memory. However, you must also use the Save selection in the Setup pulldown menu to save Setup environment changes to the disk file—18XX.CFG. If you do not Save those changes, they are lost the next time you reboot the system.

The Setup menu bar also has a Revert function. Revert restores setups to their last Saved state.

Saving Program Edits

Edit offers two ways of saving your work. Both ways save your edits to disk. You can save your edits using the Save selection in the Edit menu bar, or you can use the pop-up query window that appears when you select Quit if you have not already saved your edits using the menu bar Save.

As test steps are saved, they are written to a temporary edit file. An index file (ICT.NDX) is used to determine if any given test step is in the test program file (ICT.TST) or the temporary edit file (EDI.TMP). When you exit the editor to return to the main Menu, the test program is “packed.” This involves taking all edited steps from the edit file and putting them back into the test program file. While this is taking place, the “Packing User File ...” message is displayed.

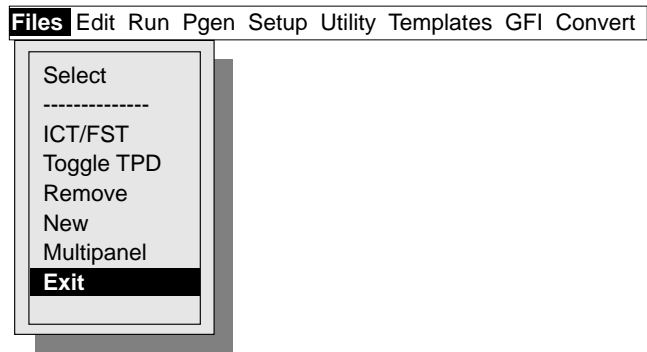
The Edit menu bar also provides a Revert function which when selected returns the test step to the most recently saved version.

Quitting from a Menu

There are several ways to quit a menu. There is only one way to fully exit the software system to enter the DOS environment.

Select Exit

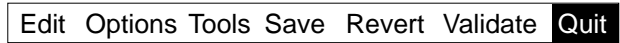
The Main menu's Files window contains an Exit command that returns you to DOS. Select Exit or press X to return to DOS. You can also enter DOS through the Utility menu's DOS Shell



command, but you do not exit the menu system when doing so. To return to the Main menu system from DOS once in the DOS Shell, type exit.

Select Quit

Most submenus offer a Quit command that allows you to quit a menu level and return to a previous level. The Quit command is found at the far right of these submenus as shown in the Setup menu below.



Press F10

F10 quits most windows and returns control to the menu. F10 accepts changes to a field as you exit. F10 does not allow you to exit a menu. ESC or quit must be used to exit a menu. F10 is also used to exit a program run session and the CRT window during Run.

Press ESC

The ESC key also returns control to a menu. Pressing ESC is an alternative to the Quit command; however, any unsaved edits are lost. The ESC key is also used to exit a program run session and the CRT window during Run.

IMPORTANT: The windowing package used by the 18xx defines the ESC key as “abort without registering edits.” Do not use the ESC key to return to the menu from Test Properties in the Step Worksheet; your edits will be lost.

Understanding the Directory and File Systems

Understanding the directory and file systems help you manage your test program development. These systems and the relationship they have to each other are explained below. Following the overview of the directory/file system is an explanation of the Files menu, which gives you access to board directories, test programs and fixture self-test programs, in addition to enabling you to create new directories and copy test programs for multipanel tests.

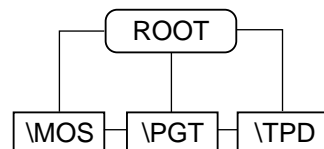
MS-DOS

The system software is designed to run on IBM-compatible computers with MS-DOS 6.0 or later.

MS-DOS contains certain files and directories that may require editing of the CONFIG.SYS and AUTOEXEC.BAT files. The degree of editing required depends on how you have configured your computer to run MS-DOS. Consult the **Computer Configuration Guide** for details regarding the installation of tester software and the management of MS-DOS files. Refer also to the documentation files provided on the software distribution disk for the latest release information.

File and Directory Structure

The tester's software system is comprised of 3 main directories: MOS, PGT, and TPD.



The MOS directory contains the tester's software system files and directories.

The PGT directory contains the system template database files in the TEMPL_DB subdirectory. The system \TTM subdirectory, under \PGT\TEMPL_DB, contains user-created vector and template files with .TTM, .JDC, .IVL, and .ASC extensions. The \PGT directory also optionally contains a user-created PGEN.CFG file. You can change the location where the 18xx system software looks for the system TEMPLATE_DB by using the PGEN.CFG mechanism.

See chapter 10, “Managing Template Libraries,” for more details if you need to have a common system library on an NFS drive.

The TPD directory contains test programs. For example, \TPD\DEMO is a directory containing all the files associated with the board test DEMO.

The following tables list the files in \MOS, \PGT, and \TPD.

MOS Files

MOS File	Description
GCIO.DLM	Dynamic Link Module included in the TSR based on hardware configuration
VPIO.DLM	Dynamic Link Module included in the TSR based on hardware configuration
VPTHC.DLM	Dynamic Link Module included in the TSR based on hardware configuration
NOIO.DLM	Dynamic Link Module included in the TSR based on hardware configuration
GCTHC.DLM	Dynamic Link Module included in the TSR based on hardware configuration
NOTHC.DLM	Dynamic Link Module included in the TSR based on hardware configuration
18XX.EXE	Binary 18xx system software executable
PRISM.IMG	Executable image loaded onto PRISM
JTOOL.EXE	Jedec to 18xx vector translator
QH.EXE	Help system's executable
UPDT_MOD.EXE	18xx template library installation tool used to update modified template
TDS18XX.EXE	Translates TDS ASCII vector files to Z18xx ASCII vector models
NUM_MOD.EXE	18xx template library installation tool
PALAMOD.EXE	Palamod (color set up) executable
D18XX.EXE	Diagnostics executable
DIAPRISM.IMG	Diagnostics executable image loaded onto PRISM
ALCONV.EXE	Library conversion tool (Alias Convert) converts D.X to E.X library format
ATBCAL.EXE	Analog test board calibration executable
CLKCAL.EXE	Test head controller and vector processor master clock calibration executable
DSCAN.EXE	Execute, validate, edit DeltaScan
DTRAN.EXE	Program generation for DeltaScan
PCIO.EXE	TSR containing 18xx hardware interface
PRISMCAL.EXE	PRISM calibration executable
DS_CONV.EXE	Convert E.4 level DeltaScan databases to current revision
INIT_COM.EXE	Executable used by 18xx to initialize COM ports
CONVD2.EXE	Used by 18xx to convert D.2 programs
ATMPLCON.EXE	Stand-alone executable for converting analog templates
TELLME.EXE	Executable that reports system configuration information

WOS File	Description
VP_12.IMG	68K executable for vector processors 1 and 2
VP3.IMG	68K executable for vector processor 3
18XX.ERR	18xx error file
WORK.BIN	Binary file that contains 18xx Step Worksheet data
PCIO.TXT	Document file that explains how to install the 18xx's pcio TSR
IPL_SPEC.TXT	Document file that describes the 18xx input list
RELNOTES.TXT	Online version of Release Notes documenting characteristics of latest version of system software.
TOOLS.HLP	Edit tools Help file
INDEX.HLP	Help index file
ANALOG.HLP	Analog Help file
DIGITAL.HLP	Digital Help file
PROGRAM.HLP	Programming Help file
CMPHDR.TXT	Text header for COMPLST file
IPLHDR.TXT	Text header for IPL.LST
NODHDR.TXT	Text header for NODE.LST
ASYNC.TXT	ASYNC.SYS driver documentation
MM.COM	DOS memory map tool to display memory usage
18START.BAT	Batch file used to launch PCIO and 18xx executables
ANALYZE.BAT	Batch file to launch DeltaScan's DTRAN.EXE to analyze an input list
CAGE_SCN.DAT	Scan file for the board ID scan routine
18XX.CFG	18xx configuration file. Contains driver/receiver configuration plus data from Setup menus. Update using Main menu/Utility/DR configuration.
BD.VEC	Vector file for driver/receiver diagnostics
FD.VEC	Vector file for diagnostics
18XX.ICO	Desktop icon for 18xx
ASYNC.SYS	Device driver for 18xx
PALAMOD.SET	Data file containing current palette settings

\PGT Files

Directory	Subdirectory or File	Subdirectory or File	Description
\PGT			
	PGEN.CFG		Global pgen configuration file. The statements in this file are applied to all programs generated unless overridden by a local PGEN.CFG file.
	\TPL		Contains user-created snapshots of sections from 18xx test programs.
	\TEMPL_DB		Default path for the system library.
		DEVICE.LST (Binary)	Entry point for all library files. Contains all device names. Points into alias.lnk.
		ALIAS.LNK (Binary)	Contains all device names grouped by aliases. Contains power data for each alias.
		TEMPL.IND (Binary)	Contains up to one pointer into each library for each device.
		ANTEMPL.LIB (Binary)	Analog template library, accessed via ATMPL token.
		GCTEMPL.LIB (Binary)	Gray code template library, accessed via IC token.
		VRSNAP.LIB (Binary)	Vector snapshot library, accessed via VIMG token.
		VRTEMPL.LIB (Binary)	Vector template library, accessed via VEC token.
		PRINT.DAT (ASCII)	Contains output from latest Template/List commands.
		ALIAS.CFG (ASCII)	User-created file made by editing and/or choosing the Template/Utility/Export.
		\TTM	Directory for system library templates.

\TPD Files

Refer to the TPD Files table for descriptions of test program directory files. All of these files are one level down from \TPD, such as \TPD\DEMO.

\TPD File	File Type	Description
IPL.DAT	ASCII	Input list file from CAD or manual entry. The input list is a complete description of the board's component parts, interconnection topology, and nodal connections. Its structured format is required by the program generator to create a program. See chapter 6, "Program Generator Input List Reference."
IPL.DBF	Binary	Component database (board topology) file. Internal use only. Created by Pgen/Build, modified by Pgen/Update.
IPEXCEPT.LST	ASCII	Documentation file. Record of screen output messages occurring during C1TODX transversion process.
ICT.TST	Binary	In-circuit test data file with Test Properties and test step data, etc. Internal use only. For more information see chapter 6, "Program Generator Input List Reference."

TPD File	File Type	Description
ICT.BCK	Binary	Back up of ICT.TST automatically made by editor each time editor is invoked.
ICT.NDX	Binary	Index file pointing to contents of ICT.TST. Internal use only.
ICT.DSC	ASCII	Optional board description file which may be used to document test program and changes. Does not exist until user creates it by entering data, either through use of F11 default or by DOS command. As a default, the F11 key opens file for editing.
IPL.LST	ASCII	Text input list output from IPL.DBF file.
COMP.LST	ASCII	Documentation output from IPL.DBF listing components and their associated nodes.
NODE.LST	ASCII	Documentation output from IPL.DBF listing nodes and their associated components.
IPL_NOD.NDX	Binary	Node Index file for internal use only.
IPL_TOK.NDX	Binary	Token index file for internal use only.
IPL_ID.NDX		
DISABLE.VEC	Binary	Board level vector disable test step file. Internal use only.
EXCEPT.LST	ASCII	Documentation file. Record of screen output messages during program generation. Records Pgen configuration if option is enabled in the environment. For more information see chapter 7, "Program Generator Tools."
GFILE.DAT	Binary	Digital guard data file containing guard data sorted by node number. Internal use only. Do not manually delete this file! Used for subsequent incremental program generation.
ICT.B0	Binary	Automatic back up file created during a "Convert" operation. This name is used whether a B.0, C.0, or C.1 file is converted because they are the same format.
FST.B0	Binary	Automatic back up file created during a "Convert" operation. This name is used whether a B.0, C.0, or C.1 file is converted because they are the same format.
ICT.DO	Binary	Automatic backup file created during "Convert" from software revision D.0 to some newer revision.
ICT.D1	Binary	Automatic backup file created during "Convert" from software revision D.1 to some newer revision.
ICTD2.X2	Binary	Automatic backup file created during "Convert" from software revision D.2, D.2.3, E.0, E.1, or E.2 to some newer revision.
ICT.E4	Binary	Automatic backup file created during Convert from E.3/E.4 to a newer revision.
VF123456.TXT	ASCII	Typical file name in test directory. Contains statistics information on vectors.
DISABLE.IVL	ASCII	Vector disable in IVL syntax. During Generate this is merged with the Vector Disable step.
TOKENLOG.DAT	ASCII	Tokenized test results file. Automatically cleared before each test cycle. Viewable only by an external program.
VF123456.VEC	Binary	Typical file name for vector test step data file. Contains vector patterns for testing a single device. Internal use only.

\TPD File	File Type	Description
VF123456.GRD	Binary	Typical file name for vector guards for Gray code test steps. Contains vector guard patterns for single test step. Internal use only.
PGEN.CFG	ASCII	Special program configuration requirements. Local file—affects only current board; overrides global \PGT\PGEN.CFG. For more information see chapter 9 “Program Generator Tools.”
NODE3V	ASCII	Contains list of nodes which require 3 V power. File is added to by both global and Step Worksheet generations. Deletions must be done by hand.
PANEL.DAT	ASCII	Node and relay exceptions for multipanel test. For more information see chapter 10, “Special & Advanced Applications.”
DATALOG.DAT	ASCII	Failure data logged by the test program, if enabled. Default condition is enabled.
PROGRAM.DOC	ASCII	Documenter utility output file. “Program” is the name of the board directory as default.
VECFILES.DOC	ASCII	Documenter utility output file. Default file name for documentation of vector files.
TOPOLOGY.LST	ASCII	Documentation output from Pgen/Reports/Topology Report.
IPL_NDX	Binary	ID index file for internal use only.
DOT.RST	Binary	Result file from batched program executions. Internal use only.
DSCAN.BAK	Binary	Backup of DSCAN.DB
DSCAN.DB	Binary	Contains test data for all components tested with DeltaScan.
PANEL.CFG	Binary	Exists in the source panel directory of a Multipanel job and stores the Multipanel configuration.
SUPP.CVG	ASCII	Optional input to the Board Fault Coverage algorithm. Contains supplemental fault coverage data.

Chapter 6, “Program Generator Input List Reference,” provides further details about the input list format.

Viewing the Main Menu

After installation, start the 18xx system software by clicking the 18xx icon on your desktop.

The Main menu with Teradyne's copyright message appears. You can press any key to remove the copyright message.

IMPORTANT: Do not scan cages with the fixture and board on the fixture receiver.

The Files and Setup selections are the only available menus at this time. The additional menus available to the Programmer, Operator, and Technician depend upon the permissions set by the Supervisor. Refer to the Setting Logins and Permissions section in this chapter for details.

The Main menu provides access to all other menus, commands, windows, and worksheets in the 18xx interface.

The current directory and the type of test (ICT or FST) appear in the Board Name field on the bottom line of the screen. The current directory above is C:\TPD\DEMO. The type of test is in-circuit test (ICT). Fixture self-test (FST) is the other type of test that can appear on the bottom of the window.

The software design, from left to right in the Main menu, contains the following features:

Files menu:

- Select
- ICT/FST
- Toggle TPD
- Remove
- New
- Multipanel
- Exit

Use the Files menu for program selection, change, removal, creation, and exit-to-DOS capabilities. Refer to the sections, Understanding the Directory and Files Systems, and Managing Programs with the Files Menu, in this chapter for details.

Edit menu:

- Header
- Interconnect (Intc)
- Passive
- Semi
- Power-Off (PwrOff)
- Board Power (Brd_Pwr)
- Linear
- Digital
- Trailer

Use the Edit menu for component selection, editing, and setting test step options. Refer to chapter 3, “Editing Overview.”

Run command:

Run executes test programs. See chapter 4, “Controlling Text Execution”.

Pgen menu:

- Edit pgen.cfg
- PEP
- Clean
- Build
- Learn
- Generate
- Combine Parallel
- Update
- Validate
- Dig Guard
- Reports

Use the Pgen menu for program generation and special reports. Refer to chapter 6, “Program Generator Input List Reference” and the **Z1800-Series Board Test Tutorial**.

Setup menu:

- Path Data
- Device & Channel Data
- System Variables
- GFI Setup
- Validate Configuration
- Color Settings
- Serial Port Setup
- Login
- Change Permissions
- Change passwords
- Set Default Login
- Save
- Revert

Use the Setup menu to set the testing parameters; logins, passwords, and user permissions; color; and serial communications parameters. Refer to the Setting Logins and Permissions section in this chapter, and chapter 2, “Setting Up System Controls.”

Utility menu:

- Nodefinder
- Reverse Nodefinder
- CRC
- Board
- DOS Shell
- Diagnostics
- Supplemental Diagnostics
- Calibration
- Board Fault Coverage
- Documenter
- Hardware Configuration
- Global Editor

Use the Utility menu for test development tools, system diagnostics, and easy DOS access. Refer to the section, Software Utilities, in chapter 9.

Templates menu:

- System Library
- User Library
- Local Library
- Create Library
- Library Permission

Use the Templates menu for managing and creating libraries and templates. Refer to chapter 8, “Managing Template Libraries.”

GFI (General Factory Interface) menu for selecting GFI programs. Refer to the section, Setting Up General Factory Interface, in chapter 2 and the optional **Test Toolbox User's Guide**.

Convert menu:

- To F.X
- F.X to E.3/E.4
- DeltaScan DB

Use the Convert menu to convert previous-version programs to current version programs. Convert can “un-convert” a program from the most recent software version to E.3/E.4 and convert the DeltaScan DB from an earlier version to an F.x-compatible format. Refer to the section Converting to Different Software Versions at the end of this chapter for detailed information.

See the **Multiscan User's Guide** for additional information about DeltaScan DB.

Setting Logins and Permissions

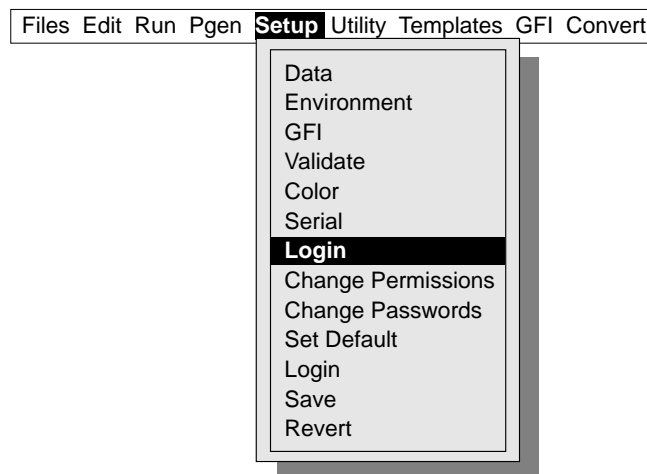
The Login command under the Setup menu allow a user to log in to the menu-based software system. Users are Supervisor, Operator, Technician, or Programmer. The supervisor sets all user logins, permissions, and passwords.

Setting Up Logins after Installation

After software installation, the supervisor can change passwords and permissions or simply accept default passwords and permissions. The supervisor cannot change the login names.

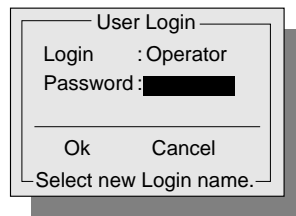
To setup user login accounts, the designated supervisor must first login as “Supervisor.”

- 1 First, click the **18xxWIN** icon on your desktop or type **18xx** at the DOS prompt to bring up the Main menu.
Teradyne's copyright message appears.
- 2 Click the mouse or press any key to redraw the screen without the copyright.
Only the Files and Setup menus are available at this time.
- 3 To access all Supervisor permissions, select **Setup** from the Main menu.
The pulldown menu appears.



1 Select Login.

The User Login window appears.



Operator is preselected in the User Login window as shown above.

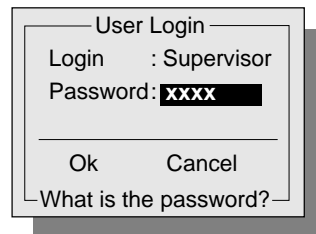
2 Click Operator,

The user type window appears.



3 Select Supervisor from the window.

Supervisor appears in the User Login window.



4 Move to the Password field and type the Supervisor's default password.

You can find all user default passwords on the Password card included in your software kit.

If you type the wrong password, the software displays the message

Invalid Password

5 Move your cursor to the OK field and the status line reads

Change to new system login.

6 Click OK to complete the Supervisor login or click Cancel to abort the process.

Once the Supervisor is logged in, all menu features are available. Supervisors should change the default password as soon as possible for software security. Limit passwords to 12 alphanumeric characters. Passwords are case-sensitive. At this point, supervisors may set passwords and permissions for all other users.

Changing Permissions Change Permissions allows a Supervisor to specify which of the menu functions users may access. Refer to the table below for details.

This Permission	For Menu/Item	Accessed From	With This Capability
Editor	Edit	Main menu	Modify and save test
	Tools	Step Worksheet's menu bar	All worksheet tools
	Files	Main menu	Remove, New, Copy
File Manipulation	Edit	Main menu	Modify but not save test
	Utility	Main menu	DOS Shell
Program Generator	Pgen	Main menu	Program generation
	Convert	Main menu	Prior version generation
	Generate	Tools menu	Test step generation
Validate	Validate	Edit menu	Test program validation
	Validate	Step Worksheet's menu bar	Test step validation
Program Select	Select	Files menu	Select a test program
Program Execute	Run	Main menu	Program run
	Start	Edit menu	Section run
	Start	Step Worksheet	Test step run
Setup	Setup	Main menu	All setup menus
Template Library	Templates	Main menu	All template menus
	Create Template	Tools menu	Take a snapshot
Node Probe Tools	Nodefinder	Utility menu	Enter nodes/probing
	Reverse Nodefinder	Utility menu	Identify pin/probing
	Edit Device Pins	Tools menu	Same as Nodefinder
Diagnostics	Diagnostics	Utility menu	Run system diagnostics
Calibration	Calibration	Utility menu	Calibrate ATB & Clock
GFI Execute	GFI	Main menu	Run GFI applications
Template Lib Write Permission	Templates	Main menu	Change library permissions
Auto-Convert	Edit	Main menu	Convert program
	Run	Main Menu	Convert program
Stop On Fail	Run	Test Control	Stop on Fail
	Start	Buttons	Edit
Revision Control	Edit	Edit menu	Record program edits

Default Permissions

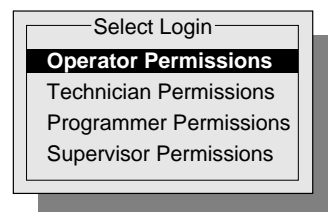
The following table lists the default permissions for all users.

Permission	Operator	Technician	Programmer	Supervisor
Editor			X	X
File Manipulation	X	X	X	X
Program Generator			X	X
Validate			X	X
Program Select	X	X	X	X
Program Execute	X	X	X	X
Setup		X	X	X
Template Library			X	X
Node Probe Tools	X	X	X	X
Diagnostics		X		X
Calibration		X		X
GFI Execute	X	X	X	X
Template Lib Write				X
Auto-Convert				X
Stop On Fail	X	X	X	X
Revision Control				X

The procedure to change permissions is as follows:

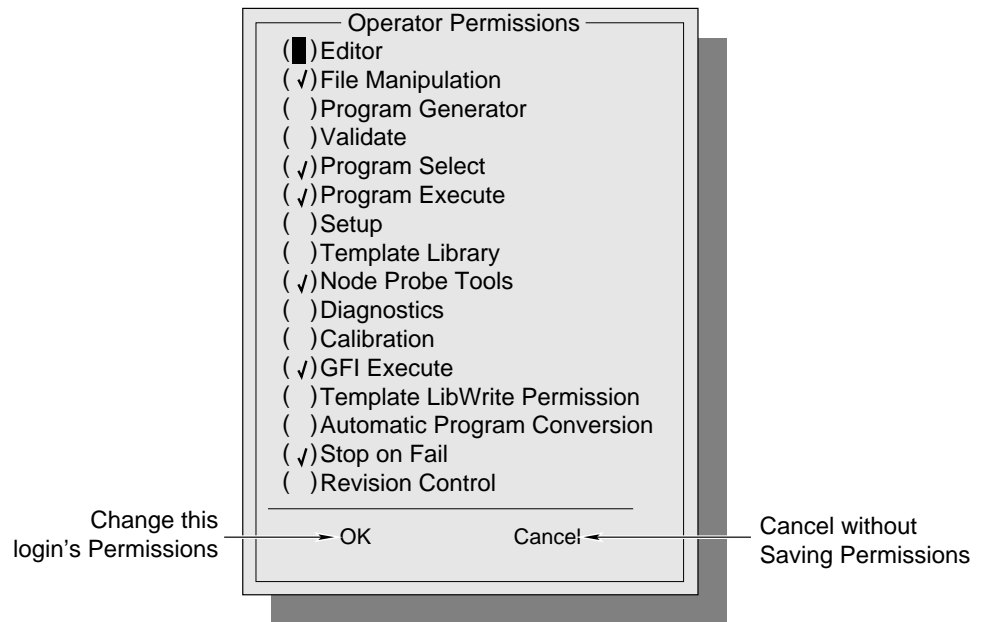
- 1 Select **Change Permissions** from the Setup menu.

The Select Login window appears. (Operator Permissions is highlighted.)



- 2 Select the user for whom permissions are to be changed.

The Permissions window appears. The items checked represent Operator default permissions.



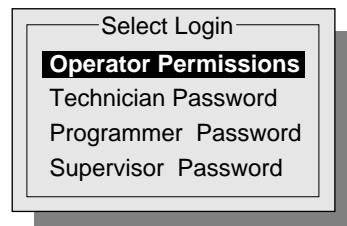
- 3 To change a permission, toggle the field preceding the permission.
Clicking a check mark removes a check mark, thus deselecting the permission. Clicking a blank field selects the field and inserts a check mark.
For example, to grant system diagnostics permission to an Operator, the Supervisor selects Diagnostics from the Operator Permissions window.
- 4 Once you have set permissions, select **OK**. The changes take effect immediately. To close the window without changing permissions, select Cancel.
- 5 To make the changes permanent, return to the Setup menu and select **Save**.

Changing Passwords

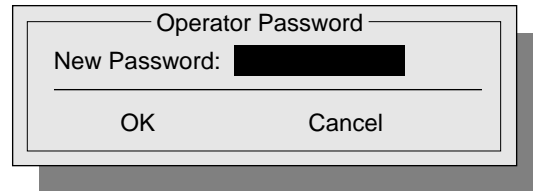
Only the Supervisor may change passwords. Refer to the password card for Operator, Technician, and Programmer default passwords. The password card comes with your software kit.

The procedure to change passwords is as follows:

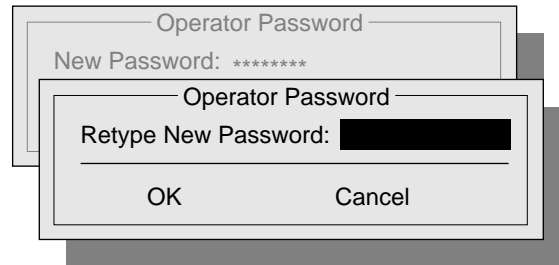
- 1 Select **Change Passwords** from the Setup menu.
The Select Login window as shown below appears. (Note that Operator Password is highlighted.)



- 2 Select the user for whom the password is to be changed.
The New Password window appears as shown below.
- 3 Type the new password.
The typed characters appear on the screen as asterisks. The password may be no more than 12 alphanumeric characters.



Remember that the password is case-sensitive. Select **OK** to confirm or Cancel to abort.
The password confirmation window below appears when you select OK.



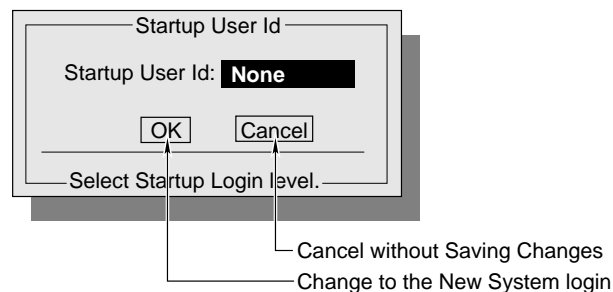
- 4 Retype the new password and select **OK** to confirm. Select Cancel to abort.
- 5 To make the changes permanent, return to the Setup menu and select **Save**.

Setting the Default Login

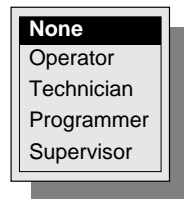
With Set Default Login, the Supervisor can configure the software to start with one of the four user logins (with set permissions), or with no permissions at all. For example, in a multitask production environment, a Supervisor might configure production test systems only for operator logins, and software-only programming stations for programmers.

To set the default login,

- 1 Select **Set Default Login** from the Setup menu.
The Startup User Id window appears. None is the default, meaning that at startup the user is able to access only the Setup or Exit menus.



- 2 Click **None** (the default) in the Startup User Id field.
- 3 Select a user from the pop-up window below.



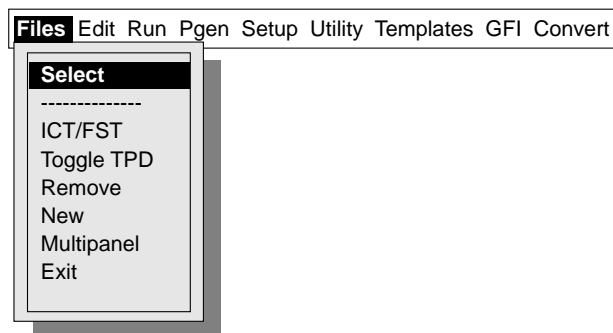
Your choice appears in the Startup User Id field.

- 4 Select **OK** from the Startup User Id field to set the default login, or select Cancel to exit without saving changes.
 - 5 To make the changes permanent, return to the Setup menu and select **Save**.
- After the Supervisor sets a user login as the default, the permissions of the default login are in effect each time the software starts.
- To save all settings and changes permanently to a file so that they remain in effect after you reboot, you must select Save from the Setup menu.

Using the Files Menu to Manage Programs

The default test program directory, \TPD, contains the board test subdirectories, which contain the test program as related files.

The Files pulldown menu accesses data in the \TPD.



It lets you select, add, and remove programs, toggle between a board's in-circuit and fixture self-test programs, change to a higher or lower directory, and exit to DOS or the Windows environment.

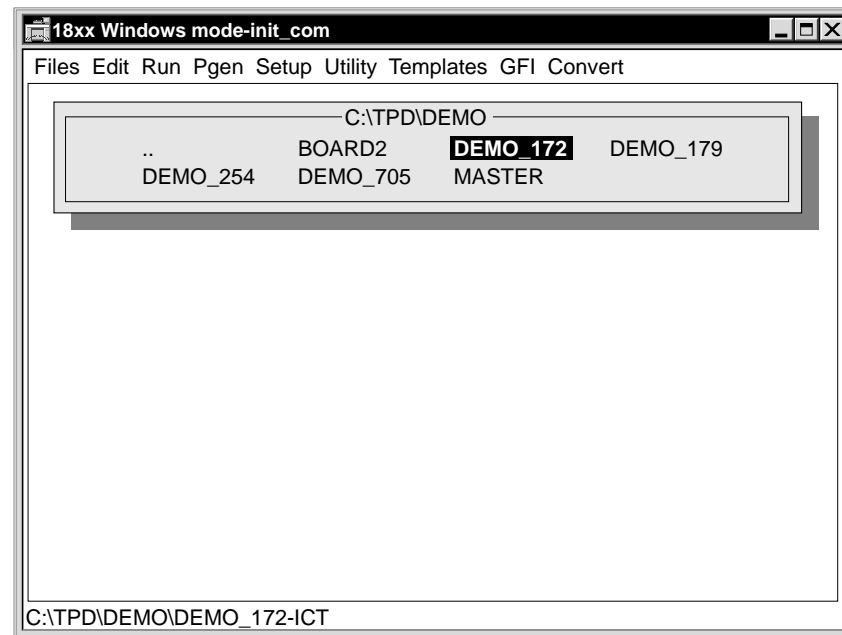
Menu Selection	Function
SELECT	Display and choose programs from the current directory and change directory level.
ICT/FST	Toggle between in-circuit and fixture self-test program types.
TOGGLE TPD	Change to the alternate test program directory.
REMOVE	Remove a program directory.
NEW	Create a new program.
MULTIPANEL	Copy (or "clone") selected program for multipanel tests.
EXIT	Exit to the DOS environment.

Selecting Programs

To select test programs from the current test program directory or to change paths to a different directory level, click **Select** from the Files menu. The current program path is stated in the Board Name field in the bottom-left corner of each window.

Note that Select has the same function in all menus.

When you choose Select from the Main menu, a directory window appears.



In the illustration above, the directory for C:\TPD is displayed listing the Board programs residing in the directory. The two dots displayed in the window enable you to access the parent directory.

The Board program names, similar to those shown in the window, follow DOS conventions and consist of two parts separated by a period. The first part of the program name is limited to 8 characters. The second part of the name, the file extension, is limited to 3 characters, such as NewBoard.xxx (notice that the Board program's extension does not display in the directory window). DOS does not allow spaces in file names but accepts an underscore to separate words.

Board programs are subdirectories within \TPD containing the necessary files for the test. The necessary files for the board program reside in that directory. (Refer to the File and Directory Structure section in this chapter for additional information.)

Move the mouse to a desired test program and observe that the program name changes color indicating that your choice is ready to select. Single click the mouse to select or press Enter.

A double click, or the key combinations Alt-Enter or Ctrl-Enter, changes directory into the selected name (cd dir_name). If the selected directory does not have a subdirectory, the change directory does not take place. To change to a parent directory (next higher), place the cursor on the double dots field (..) and double click or press Alt-Enter.

The Files/Select menu affects three Setup/Data variables: the Primary Program Directory, the Alternate Program Directory, and the Currently Selected Board.

In DOS, you can also list a directory's files and select a program using the DIR command. To exit to DOS from 18xx, click Utility in the Main menu, and then click DOS Shell.

See, Changing Program Directories, in chapter 2, for more information.

Selecting ICT/FST Program Types

Use **ICT/FST** to select a board's in-circuit or fixture self-test program. When you select ICT/FST to toggle between the programs, a query window appears asking you to confirm your selection.

The Board Name field in the lower-left corner of the screen shows the current program type with either an ICT or FST suffix, as in

C:\TPD\DEMO – ICT

Both program types reside in ICT.TST. You may only edit or run the test steps for the selected test type.

Toggling Between Test Program Directories

Use Toggle TPD to change from the master test program directory to the alternate. Before you attempt to toggle from one TPD to another, you must have created a second test program directory in DOS (a default name is \TPD_2) and set up the alternate directory in the Setup and Environment menus.

See the section, Setting Up Environment Data, in chapter 2.

If you have not created a second TPD or have not set the alternate directory, the software displays the message

Cannot change directory to C:\TPD_2

Removing a Program

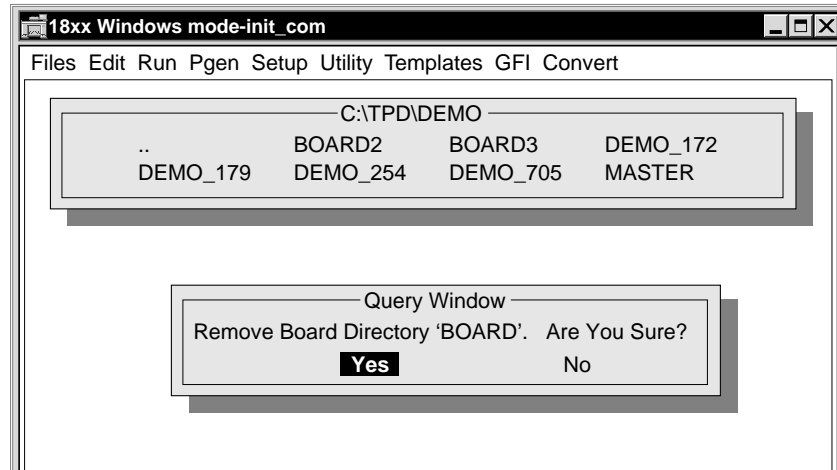
Use the Remove feature to remove a board program.

- 1 Click **Files**, and then click **Remove**.

The Select window appears listing the programs resident in the directory.

- 2 Click the program name you want to remove.

The a window appears requesting that you confirm or deny removal.



- 3 Select **Yes** to remove the directory, or select **No** if you decide not to remove the directory. If you select Yes, the software system displays the message "Removing Board Directory." If you select No, the software system returns to the Files menu.

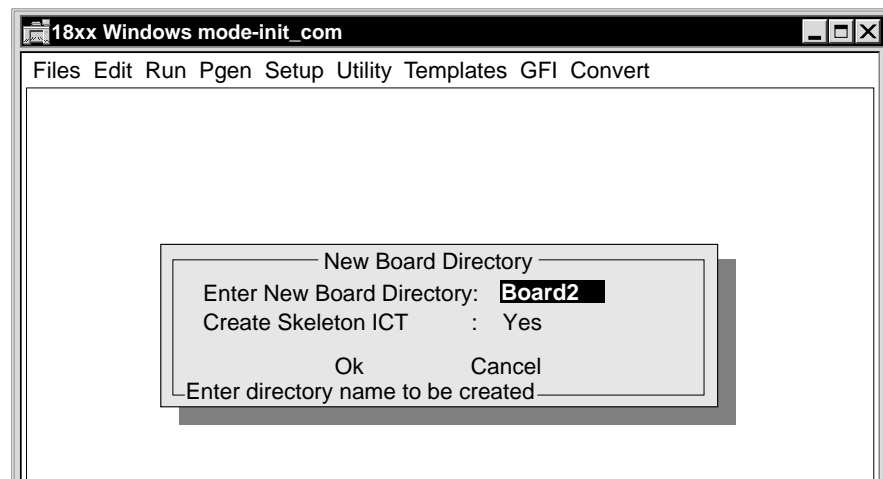
Creating a New Program Directory

Clicking **New** creates a board program directory containing skeleton test steps. This new program directory resides in the current test program directory (default is \TPD).

To create a new board program directory,

- 1 Select **New**.

The New Board Directory window appears.



- 2 Type the name of the new program directory.

DOS restricts file names to 8 characters. If you include a dot “.” in your file name, you can add an additional 3 characters after the dot. The maximum length of a program name can be 11 characters, as in, USRTEMP3.TTM. Use alphanumeric characters only. Do not use slashes or asterisks. The names are not case-sensitive.

- 3 Set up the contents of the new board directory by clicking the **Create Skeleton ICT** field.

To add the skeleton in-circuit test to the program directory, click Yes.

To create a program directory without the skeleton test, click No.

- 4 Click **OK**.

The new program directory appears in the select window when you view the current test program directory's contents. The new program directory also becomes the currently selected directory.

Clicking the Cancel button cancels creation of a new program directory.

If the software system cannot create the directory for some reason (for example, a directory already exists with that name), it displays a message indicating that it cannot do so, and returns control to the Files menu.

Copying a Program Using Multipanel

Multipanel has two associated functions:

- Making a duplicate of the selected program for situations requiring a copy

Multipanel is the program copy utility for 18xx. Use Multipanel when you need an exact copy of the program to use, for example, as a base for creating a test on a new board.

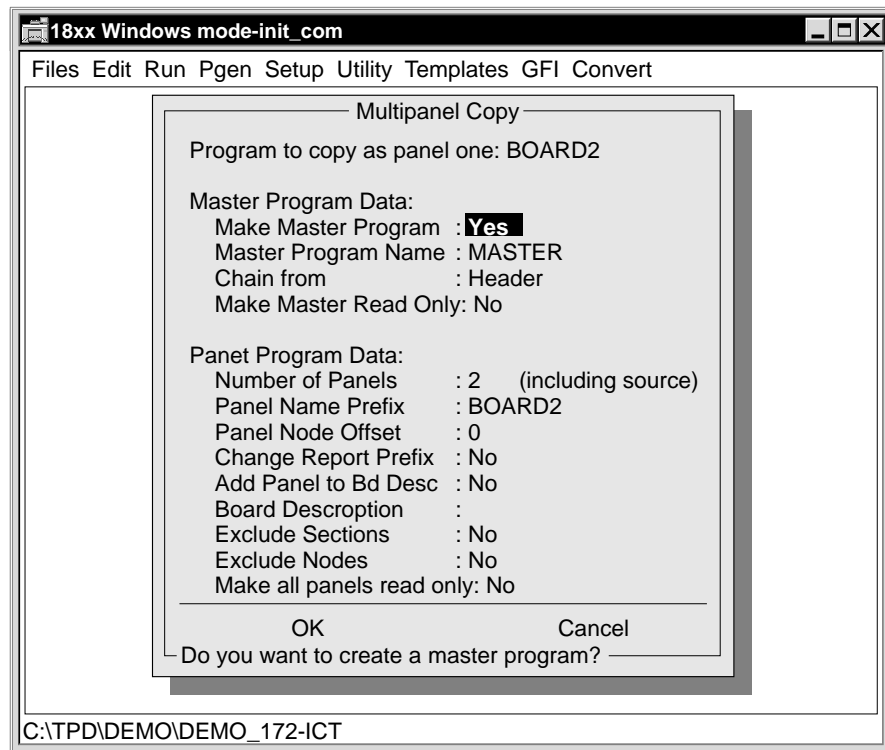
- Making one or more copies of the program with different node offsets for a multipanel test program

Multipanel reduces the time it takes to generate programs for fixtures that have been designed for multiple boards of the same type. Generate a program for one of the boards (master) and, using appropriate node offsets, copy this program using Multipanel to generate programs for the other boards in the panel.

To copy or “clone” the currently selected program,

- 1 Select **Files** and then **Multipanel**.

The Multipanel window appears.



- 2 In the Make Master Program field, select **No**.

Selecting No disables the other Master Program Data fields. Do not enter data in the Panel Program Data section.

- 3 Select **OK**.

Once made, the duplicated program resides in the TPD directory at the same level as the master program.

For detailed information about using Multipanel with different node offsets see chapter 10, “Special and Advanced Applications.”

Select Exit from the Files menu, to return to the DOS shell.

Converting to Different Software Versions

Convert allows you to convert and update in-circuit test programs and fixture self-test programs from version B.0/C.1 to the current software version. You can also “unconvert” from revision F.x to E.3/E.4.

If you have a DeltaScan database created with a pre-F.x version of the system software, the utility permits you to convert that database to the integrated DeltaScan. The DeltaScan database conversion should be run only after you have converted your program to F.x.

See the **Multiscan User’s Guide** for additional information about DeltaScan DB.

The 18xx system software has two modes for program conversion—automatic and manual.

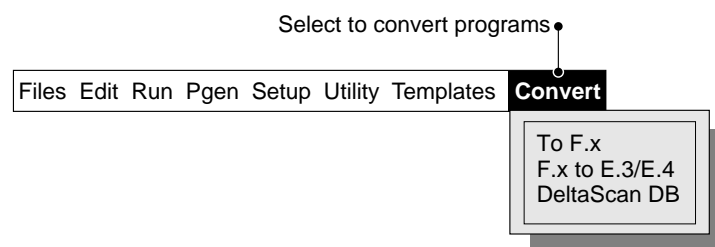
Each Login has a permission for Automatic Program Conversion. If this permission is enabled, then the system software automatically converts a program to the current revision instead of issuing an error message indicating that the program is not at the expected revision. This automatic conversion takes place when an ICT program is opened in Edit, Run, or Documenter.

If you have not enabled Automatic Program Conversion, the system software issues an error message if you need to convert your program. In this case, run Convert from the Main menu. If you try to run Convert on a program that already is at the correct version, a message appears informing you of that fact.

To run Convert

- 1 Select the correct board test from the Select window of the Files menu.
- 2 Select **Convert** from the Main menu.

The following menu appears.



- 3 Select To F.x to convert to the current revision or F.x to E.3/E.4 to unconvert from F.x back to the software revision preceding F.x.

If you select To F.x, a message similar to the following appears:

```
Converting <board name>
Converting ICT from E.3/E.4 to Revision F.x
Converting FST from E.3/E.4 to Revision F.x
Creating backup file ICT.E4
Hit any key to continue ...
```

- 4 If your program contains an external call to DeltaScan (called from an E.4 or earlier version of the system software):

Disable the step where your program makes the external call, and
Select Convert/DeltaScan DB from the Main menu.

- 5 If you want to go from F.x back to E.3/E.4:

Copy DSCAN.E4 over DSCAN.DB, and
Re-enable the external program call wherever it may be in your program.

IMPORTANT: If your original program contains corrupted files or other problems that prevent conversion from E.3/E.4 to F.x, an error message appears telling you which section could not be converted and whether that section was in ICT or FST. Then Convert continues to convert as much of the program as possible.

▼▼▼

2 SYSTEM SETUP

The Setup menu provides facilities for setting up parameters that control tester performance on a global level. Therefore, the settings you make affect every test program you open and run. It is a convenient way to set up default preferences for all users and/or all test programs.

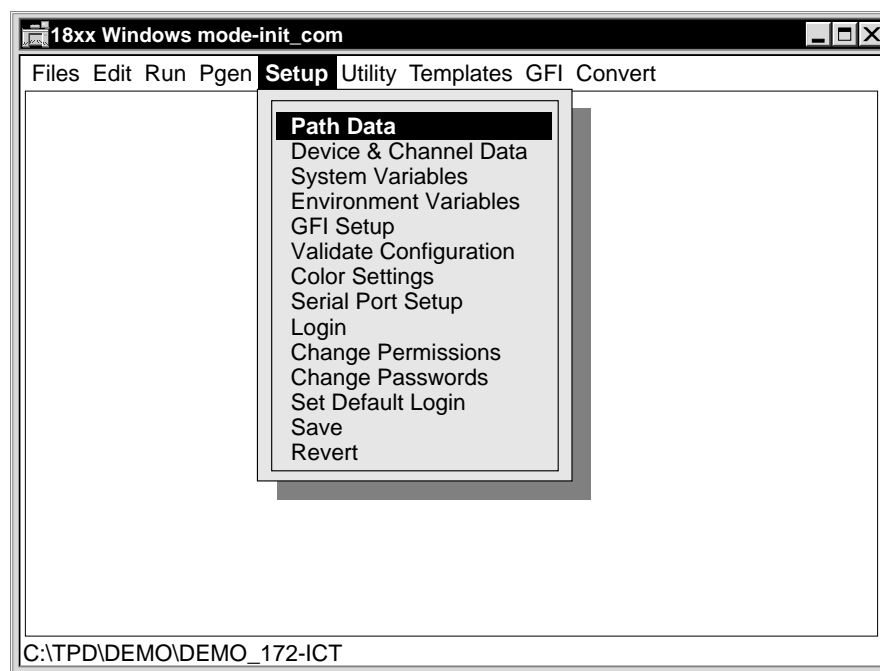
The Setup menu includes control of data management, tester environment, General Factory Interface data, validation of capacitor, resistor, and MultiScan tests, color setup for the user interface, and serial communications ports. In addition, the Setup menu enables the supervisor to set up logins and permissions.

For detailed information about Logins, see chapter 1, “Setting Logins and Permissions.”

Set Up Overview

The Setup menu allows you to setup and modify the tester’s environment control and test variables, color schemes, and communications ports. You may configure the environment at any time after installation.

The Setup menu resides on the Main menu bar and contains the functions shown in the illustration below.



Check the settings in the Setup menus before developing or running a test program. Test and communications parameter defaults may require change for your particular application and for those with restricted permission levels, such as operators and technicians.

If, in the process of making changes, you want to return to the settings established since the most recent Save, select the Revert command from the Setup menu. When you have made changes to any of the Setup menus and wish to preserve them, select Save from the Setup menu. After you have saved your changes, you cannot revert to the previous settings.

Each Setup screen has an OK and a Cancel button. Selecting OK causes the data in that window to be written to the TSR's memory and exits the window. Selecting Cancel exits the window without writing the data to the TSR's memory and leaves the TSR's image unchanged.

Upon start-up of the 18xx, the data structures within the TSR are initialized from data in a binary file called 18XX.CFG. The 18XX.CFG file is the permanent storage space for Setup data and the TSR is local storage (while the TSR is installed).

Save at the Setup menu level causes all Setup data currently in the TSR to be written to the 18XX.CFG file. The next time the TSR is installed, this data is loaded into the TSR's memory buffer.

Setting Up Data Paths

When you click **Setup** and then **Path Data**, the following window appears:

Path Data Fields

Field	Default	Description
1800 Operating System Drive	C	Disk drive partition where the system software resides.
Primary Program Directory	C:\TPD	Full path name of the primary test program directory.
Alternate Program Directory	C:\TPD_2	Full path name of the alternate test program directory.
Currently Selected Directory	BASIC	Name of the currently selected board program. This field cannot be edited.
F11 Command	EDIT ICT.DSC	Invokes the specified DOS command. Default is DOS editor, which can be invoked in edit, SOF edit, and vector edit. Accepts any legal DOS command. Completion of command, restores 18xx interface. If field is empty, F11 invokes the DOS shell. Provides "scratch pad" for programming notes.
DOS EDITOR	EDIT	The editor invoked for editing DOS files. May be Edit or any other resident editor.

Changing Program Directories

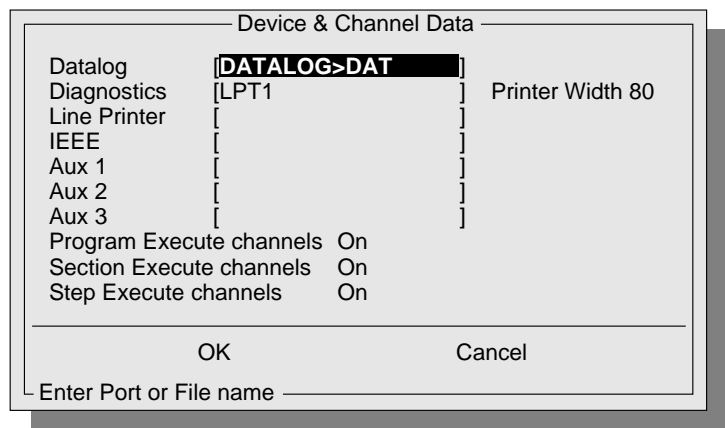
The Setup/Path Data variables—Primary Program Directory, Alternate Program Directory, and Currently Selected Board—are affected by changes you make in the Files/Select menu.

- When you select a file in the Files/Select menu (with a single click or by pressing Enter), the Currently Selected Board variable is updated.
- When you change a directory in the Files/Select menu (with a double click, or by pressing Alt-Enter), the active (either Primary or Alternate) Program Directory from the Setup/Path Data window is updated.
- When you type “18xx” at the DOS prompt, the Primary or Alternate Program Directory and Currently Selected Board variables are used to determine which program to change to.

When these variables are updated in the Files/Select mode, the system software starts up with the most recently selected program directory if the PCIO.EXE TSR has not been removed. After you have turned off the power to the PC or typed pcio -r (to remove the PCIO.EXE file), the system software starts up with the program directory last saved under Setup.

Setting Up Communication Channels

To set up 18xx communication channels, click Setup on the Main menu and then click Device & Channel Data. The following window appears.



Field	Default	Description
Datalog	DATALOG.DAT	If Datalog is on in the Header, failure messages are appended to the file or device specified by file name. If no file is specified, Datalog is inactive.
Diagnostics	LPT1	Diagnostics Printer Port to receive operator's failure messages. May be the same as Line Printer Port. Can also specify file name.
Line Printer	(Null)	Port to receive information from program or DOS Print function. May be the same as Diagnostics port. Can also specify file name.
IEEE	(Null)	Port to send and receive IEEE data and messages.
AUX 1 through 3	(Null)	Auxiliary ports.
Program Execute channels	On	Channels and output devices to be active for entire program execution (Run mode).

Field	Default	Description
Section Execute channels	On	Channels and output devices to be active for section Run mode.
Step Execute channels	On	Channels and output devices to be active for step execution mode.
Printer Width	80	Number of columns on the diagnostic printer. Longer lines begin on new line.

The first seven fields allow you to enter a DOS device or file name. A dialog line at the bottom of the window explains each field.

Click OK to confirm the Device & Channel data after making selections. Click Cancel to exit the window without making changes.

The tester's software structure employs a channel system to communicate results and messages. There are nine possible channels, seven of which are fully configurable.

You can establish a hierarchy of control over data output depending on which channels you enable at the systemwide level.

Then, for each mode of operation (program execute, section execute, and page execute), you can select which of the channels are available for use by test programs. Note that if a particular channel is not enabled at the systemwide level or is not selected for the current mode of operation, data is not sent to that channel even if selected by the test program. Program and test step controls for routing output data are explained below.

Setting Up Channels: Systemwide

To enable data output systemwide through any or all of the first 7 channels, you must configure the channels to any DOS file name or DOS character device type through the Setup menu's Device & Channel Data window.

See your DOS documentation for valid device names such as LPTx, COMx, PRN, etc.

To automatically disable a channel systemwide, do not enter a file or device name in the channel field.

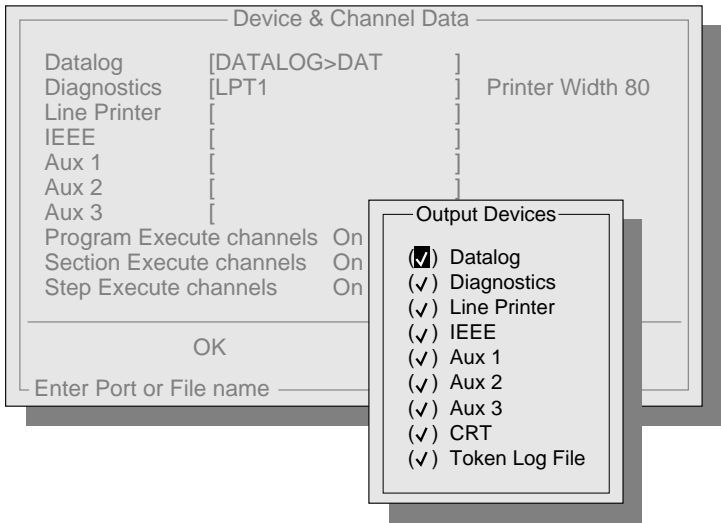
You cannot configure the CRT channel (since this channel always refers to a screen) or Tokenlog. The Tokenlog channel is accessible in the Device & Channel Data window only through Program Execute Channels.

Setting Up Channels: Program, Section, and Step Execute Modes

Program Execute, Section Execute, and Step Execute in the Device & Channel Data window allow you to set up output channels at the system level. To specify channels in any of these modes, click On. The Output Devices window appears.

When a channel is checked, it is available for any test program to route data to for that mode. The opposite is true if a channel is not checked. For example, if Aux 1 is not checked in the Program Execute channels, then no test programs can send data to Aux 1 during program execute mode even if Test Properties explicitly instructs an output to Aux 1.

To specify the output devices in Program mode, for example, click the Program Execute channels field. The Output Devices window appears.



Click in the parentheses to select or deselect an output device. With a keyboard, use the spacebar to toggle your choices. A check mark indicates a selected device.

The following table shows the valid settings for output devices in program, section, or test step modes.

Output Device	Program	Section	Test Step
Datalog	Y	Y	N
Diagnostics	Y	Y	N
Line Printer	Y	Y	Y
IEEE	Y	Y	Y
AUX 1	Y	Y	Y
AUX 2	Y	Y	Y
AUX 3	Y	Y	Y
CRT	Y	Y	N
Tokenlog	Y	N	N

Datalog and Diagnostics are normally excluded from the Section and Step execution modes since these modes are usually used for debugging.

Setting Up Channels:
Program and Test Step Levels

Output Devices windows in Step Worksheets allow you to disable or enable output channels at the program and test step level. There is no Output Devices control at the Section level.

Program Level Set Up. Output events—report messages output from Header and Trailer test steps or the Pre-Test Options, for example—are fully configurable in the Output Devices window. You may send such data to one or more channels at any time.

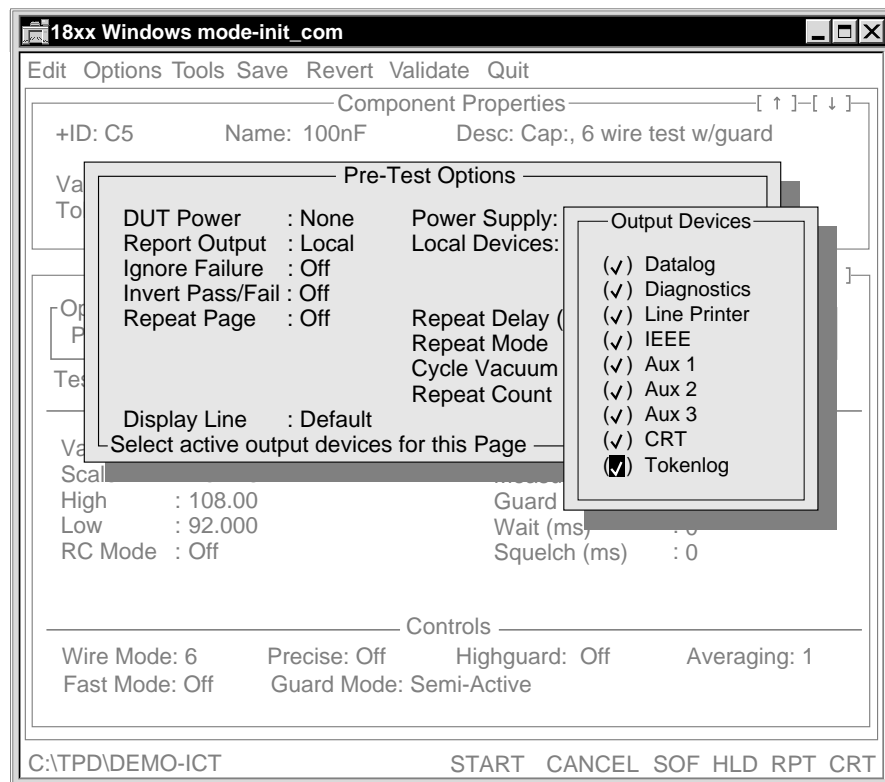
To select output channels at the program level, click Header/PRGMVARS, then click the Report Output Device(s) field. The Output Devices window appears in which you can check the desired output devices. Remember that if a device is not checked in the Setup/Device & Channel Data window, enabling it here will have no effect.

By default, the output data is sent to the Datalog, Diagnostic, and CRT channels. If you want to suppress report generation for an entire program, do not select any channels.

Test Step Level Set Up. To modify the program-wide report data distribution system on a step-by-step basis using the Pre-Test, Option Variables and the Pre-Test or Post-Test I/O Control,

In the Pre-Test Option Variables window,

- 1 Click the **Report Output** field, and then click **Local** from the popup window.
The Local Devices field becomes active.
- 2 Then click the **Local Devices** field.
The Output Devices window appears.
- 3 Check output devices to route messages to at the test step level.



- 4 Close out of the list.

Upon closing, On will be highlighted, indicating that some output devices are enabled while others may be disabled.

The channels selected here supersede the channels specified from the program level.

Unlike other output devices, Tokenlog is enabled only at the system level—there is no program-level control. If Tokenlog is enabled at the system level, using Report Output in Pre-Test Options is the only way to prevent data from being routed to Tokenlog.

To prevent data from being routed to Tokenlog,

- 1 Choose Pre-Test Options/**Options Variables**.
- 2 Click the **Report Output** field in the Options Variables window.
- 3 Click **Local** in the Report Output pop-up.

The Local Devices field becomes active, Off is highlighted, and data is not routed to any output devices, including Tokenlog.

Note that if Tokenlog routing has not been activated at the system level, you cannot activate it by using this function.

To route Conditional I/O from Pre- or Post-Test options to the appropriate output devices, select I/O Control from either the Pre- or Post-Test Options windows. Select an I/O Condition other than Unused; the Output Device field becomes enabled. Click the Output Devices field to popup the Output Devices window and check the desired output devices.

To route Conditional I/O from Header and Trailer test steps to the appropriate output devices, click Output Devices and select the output devices.

Remember that, if a device is not enabled in the Setup/Device & Channel Data window and at the program level, you cannot enable it here.

Datalog Setup Example

Datalog is under the full control of the test program through the channel system.

To set up a program for Datalog operation,

- 1 Assign the Datalog channel to a file name in the Setup/Device & Channel window.

The default file name is DATALOG.DAT. In the default case, the Datalog file field creates a file, DATALOG.DAT, in each program directory where Datalog is used. The field may be set to an absolute path such as C:\TMP\DATALOG where all Datalog data from all programs would be stored. The Datalog channel might be set alternatively to a peripheral device such as COMx where COMx could be a serial port connected to a different computer.

- 2 Enable the Datalog channel for the required execution modes from the Setup/Device & Channel window.

The Datalog channel is enabled by default for the Program Execution mode.

- 3 Enable the Datalog channel for a given program in the program's Header.

Click PRGMVARS.

Click Report Variables.

Click the Datalog field, and then click OK.

Combined with steps 1 and 2 above, this setting opens the program's Datalog channel.

- 4 Click **Report Output Devices** in Header/Report Variables to send the failure report to the datalog channel.

The Output Device window appears.

- 5 Click **Datalog**.

The default for the report tag will be sent to Datalog, the Diagnostics printer, and CRT. You can customize the report by selecting from the report features following the Report Output Devices: Report ID, Report Description, Report Meas nodes, Report Values.

IMPORTANT: The Datalog Begin and Valid messages are program Header and Trailer test steps which you can customize.

Tokenlog Setup Example

The Tokenlog file contains test results in a more easily machine-readable format. It is created to allow external programs to easily parse the results into internal structures. The token format makes it easier for the GFI custom programs to parse test results. The output will:

- Identify the beginning and end of each block of test data for easier machine readability
- Identify each data item by adding a prefixing token to each item
- Mark each block of test data with the type of test being performed.

Always in token format, the file is enabled and disabled through Setup/Program Execute Channels. On every test cycle the file is cleared so that at any one time it includes only one board's worth of test data.

To set up a program for Tokenlog operation.

- 1 In the Setup/Device & Channel Data window, click the Program Execute channels field.
The Output Devices window appears.
- 2 If Tokenlog File has not been checked, click in the Tokenlog field to create a Tokenlog file and then click OK to exit from the Setup menu.
System test messages are automatically directed to the Tokenlog file.
- 3 To direct custom messages from a Header or Trailer message step (named <message>) to the Tokenlog file,
 - Select Header/<message> or Trailer/<message>.
 - When the Header or Trailer window appears, click the Output Device field to activate the Output Devices window.
 - Select Tokenlog File.
- 4 To direct custom messages from individual test steps to the Tokenlog File,
 - Select the desired test step from the Edit menu.
 - Select Options/Pre- or Post-Test Options.
 - From the Options window, select I/O Controls making sure that the I/O Condition selected is something other than "Unused."
 - Select Output Device from the Conditional I/O field.
 - Click Tokenlog File.

The Tokenlog content is affected by Allprint. When Allprint is enabled, all passing and failing information is sent to the Tokenlog File and, when it is off, only failing information is sent.

Setting Up System Variables

System variables apply to the tester and influence measurement data and tester operation even though, as global settings, they are not part of explicit test step parameters.

To set up 18xx system variables, click Setup on the Main menu and then click System Variables. The following window appears.

System Variables

Highest System Node	0	
ATB System Resistance	1.8	Ohms
ATB System Capacitance	22	PF
Prism System Resistance	0.5	Ohms
Prism System Capacitance	22	pF
Prism Backplane Capacitance	1.2	nF
Prism Backplane Inductance	10	uH
Prism Line Frequency	60	Hz
Prism Trim Interval	60	Minutes
Prism Memory Test Level	1	
Nodefinder Impedance	4	
Backdrive Timeout	3	
Record Pgen Configuration	00	mS
Duty Cycle	0	Percent
Test Jack Panel	Off	
VP Timeout	2000.0	mS
Clear test page setup at	End of the Page	

OK

Cancel

Enter the System Resistance (0–20)

Refer to the table below for more information about the System Variables window.

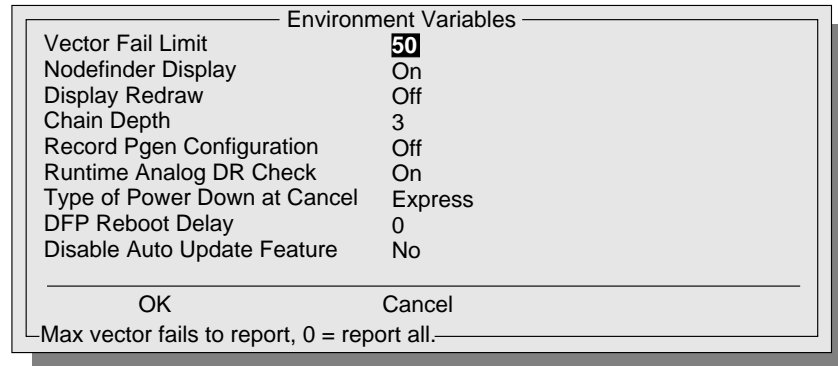
IMPORTANT: The System Capacitance default value of 22 pf is not to be understood as accurate for all, or even most, test systems because of the wide range of fixturing possibilities. It is important that system capacitance in the Setup/System Variables window be set to the correct value for your particular system. To find the appropriate value for system capacitance, use the procedure outlined in the Capacitor Testing chapter in the *Z1800-Series Component Test Reference*.

System Settings	Default	Description
Highest System Node	System's Highest Node Number	The highest node found in the test cage based on the placement of driver/receiver boards. Full complement of boards in a model Z1888-1/80/40/00 = 639; Z1890/88-2, 60/20 = 2047; Z1884 = 5119. Node count starts at 0, e.g. maximum node count for Z1840 is 640. Number is displayed in non-edit field. Does not account for missing or nonsequential placement of driver/receiver boards.
ATB System Resistance	1.8 Ohms	Subtracted from all non-precise resistor measurements before display or comparison against programmed limits. Range = 0–20 Ohms.
ATB System Capacitance	22 pF	Subtracted from all capacitor measurements before display or comparison against programmed limits. Range = 0–1000 pF. In Cap Phase measurements, is subtracted if "Use Sys Cap" is Yes.

System Settings	Default	Description
Prism System Resistance	0.5 Ohms	Deducts effect of relay or trace resistance in signal path which is not corrected by Prism sensing arrangement.
Prism System Capacitance	22 pF	Defines a value for capacitance between E and F poles during a Prism test. Used to correct Prism measurement results.
Prism Backplane Capacitance	1.2 nf	Default value is worst-case value. This is the capacitance which must be driven to change a voltage level on the backplane. It affects default wait times used by the system for Prism tests—the higher the value, the longer the default wait time.
Prism System Inductance	10 uH	Defines a value for capacitance between E and F poles during a Prism test. Used to correct Prism measurement results.
Prism Line Frequency	60 Hz	Specifies the input power frequency for the tester. The value is used by Prism test code to determine how long to sample an output when Line Reject is enabled.
Prism Trim Interval	60 minutes	Specifies time interval between realignment of the two Prism stimulus sources and a new placement of the measure strobe for maximum sensitivity to voltage. Trim is performed only before bridge mode measurement and not until 30m minutes after system power up. Trim process takes about 5 seconds.
Prism Memory Test Level	1	Used to check Prism memory chips. The value affects the number of patterns used in testing memory. Range 1 to 5. With values 1–5, all memory locations are checked. Higher numbers check memory with more patterns.
Nodefinder Impedance	4.0 Ohms	Maximum resistance for Nodefinder function. Range = 0–50 Ohms.
Backdrive Timeout	0 ms	Maximum time a digital stimulus is applied. If any digital burst exceeds this time, the burst will not occur and an error is issued. Enter a value of 0 to turn off the function. Range = 0–9999.9 ms. PRGMVARS/Backdrive Timeout overrides the value specified here.
Duty Cycle	0%	Ratio of burst time to wait time between bursts for digital tests, expressed as percentage. Value of 0 turns off. Range = 0–100%. PRGMVARS/Duty Cycle overrides the value specified here.
Test Jack Panel	Off	Enable test jacks/test envelope in Edit or debug mode. Select On or Off. Test jacks/test envelope are always Off in Run.
VP Timeout	2000.0 ms	Sets the maximum time which a vector test step may take to complete. If the test step cycle is not complete within the specified time, a VP Timeout error message is issued. Range 0–64000 ms.
Clear test page set up at	End of page	Clears and sets node relays between tests.

Setting Up Environment Variables

To set up 18xx Environment Variables, click Setup on the Main menu and then click Environment Variables. The following window appears.



Refer to the table below for more information about the Environment Variables window.

System Settings	Default	Description
Vector Fail Limit	50	Controls the amount of vector failure information reported when in vector editor. Maximum is 32767. A high number in this field slows execution time.
Nodefinder Display	Off	Enable display for embedded (Test Properties) Nodefinder. Select On or Off.
Display Redraw	Off	Forces the display to redraw upon return from subprogram or external program call. For user programs which modify display. Because save and redraw take time, do not use if subprogram does not modify screen.
Chain Depth	3	Specifies the depth for chaining subprograms. Max is 5.
Record Pgen Configuration	Off	Records Pgen configuration to exception list file during program generation.
Runtime Analog DR Check	On	When On, compares the driver/receiver configuration to selected program. It directs you to EXCEPT.LST file for conflicts between nodes selected for a digital test and the presence of an analog-only or functional interface board in the slot where the nodes are located. If no analog-only driver/receiver cards are present in the system, the compare routine will not be executed.
Type of Power Down at Cancel	Express	Use only with Cancel. When On, executes the power-down function that turns the power supplies off in reverse order of their on sequence; does not clear the power supplies with a reset command.
DFP Reboot Delay	0	Time in seconds to wait for the DFP to reboot. This delay is only applied at the start-up of the 18xx program.
Disable Auto Update	No	Prevents auto update by operations requiring data from a modified IPL database.

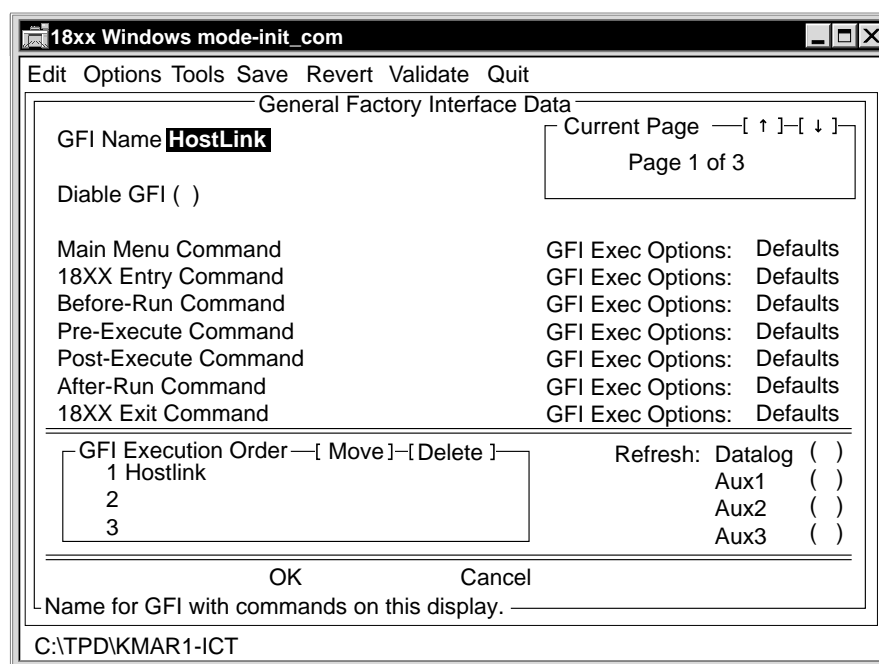
Setting Up General Factory Interface

General Factory Interface (GFI) provides a set of connections in the standard 18xx software to allow you to write custom factory communication modules for Z1800-series systems. Teradyne provides several optional GFI applications including HostLink and BoardWatch. See the Z1800-Series option documentation for further information about these options.

The optional Test Toolbox enables you to develop applications of your own for adapting a Z1800-series tester to an automated environment. See the **Z1800-Series Test Toolbox User's Guide** for further information about developing your own custom programs. These applications are separate from the system software and communicate with the tester only through GFI.

To set up GFI to use your custom modules, select GFI from the Setup menu.

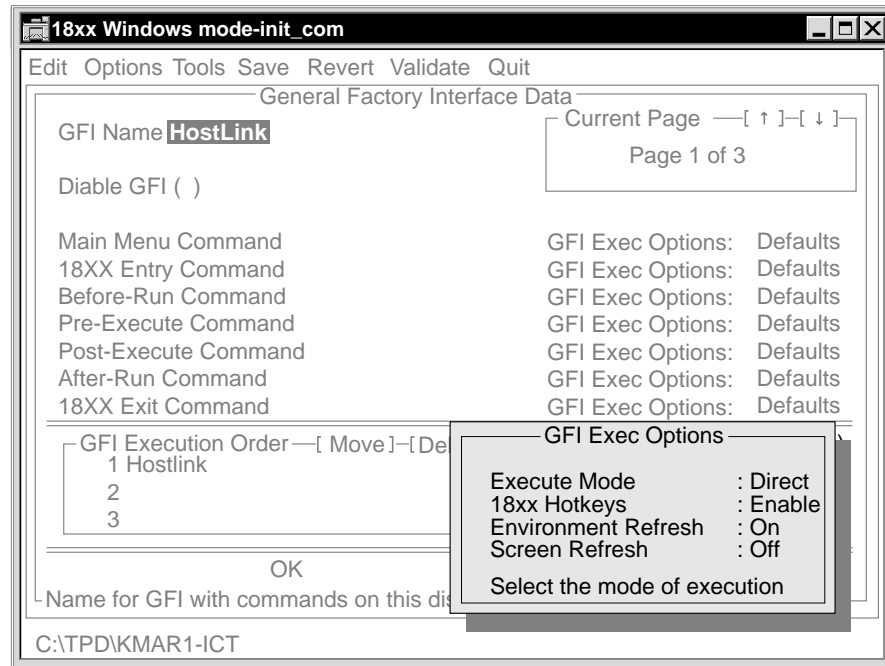
The General Factory Interface window appears. To provide flexibility and yet not use excessive space, the interface permits the use of no more than three custom GFI applications.



To fill out the General Factory Interface window:

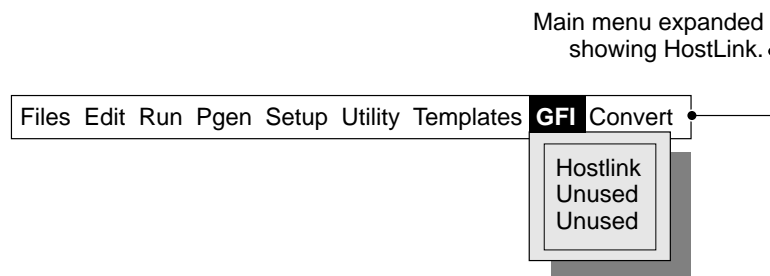
- 1 Move the cursor to the GFI Name field and type in the name of the program module, HostLink, for example.

As soon as you move the cursor to the next field, the name you typed in the GFI Name field appears in the GFI Execution Order field.



- 2 Move the cursor to the Main Menu Command field and type in the name of the appropriate executable program in \MOS.

You must fill in this field for the GFI Name to appear in the expanded GFI menu in the Main menu as shown below.



For detailed information about the type of information required in the 18xx Entry Command through the 18xx Exit Command fields, refer to the **Test Toolbox User's Guide**.

- 3 Move the cursor to the GFI Exec Options to specify settings for Execute Mode, 18xx Hotkeys, Environmental Refresh, and Screen Refresh.
- 4 To select the Execute mode, click the **Execute Mode** field.
A pop-up window appears allowing you to select Direct or Command.com.

For external executables, generally use the Direct execution mode. Use COMMAND.COM when you want to call DOS internal commands such as COPY, DIR, CD, and BATCH files since COMMAND.COM interprets these commands directly. It is not advisable to use COMMAND.COM to execute external (.EXE) programs because more memory is consumed and return values from the executable are suppressed.

- 5 To disable the 18xx Hotkeys, click the **18xx Hotkeys** field.
A pop-up window appears allowing you to select Disable or Enable. The default is Enable. Hotkeys should be disabled for the duration of the execution of the external program in order for the external program to receive key strokes from the F3–F8 keys and mouse events from the Start, Cancel, RPT, HLD, SOF, and CRT fields.
Software versions prior to F.0 did not have user control for this feature, and hotkeys were always on during a swap. However, GFI external executables make use of the hotkeys directly. For these executables, 18xx Hotkeys must be on during the swap.
- 6 To refresh the environment setup, click **Environment Refresh** and select **On**.
There are two copies of the 18xx environment data in the system. One copy resides in the TSR and is modified and read by the external programs. The other is part of the protected mode executables. When an external program modifies the TSR environment, the data needs to be copied to the protected mode location upon return of the swap for the 18XX.EXE to see the change. The copy is necessary only when the external program modifies the Environment. The default state of the Environment Refresh is on. Throughput is increased when Environment Refresh is off.
- 7 To enable Screen Refresh, click **Screen Refresh** and select **On** from the pop-up window.
Screen Refresh refreshes the screen upon return from the swap. This is the same functionality as the Environment/Display Redraw, but it is local to the current Test Properties only.
- 8 Click **OK** to save the changes to GFI or on Cancel to cancel without saving changes.
- 9 To add a second GFI to your test program, move your cursor to the Current Page field and click the down arrow.

The field will then read Page 2 of 3, and the values in the top portion of the new page will be blank. Fill in the top portion of the second page with information appropriate to the second GFI. (You may add a third GFI.)

Disabling a GFI Application

To disable a GFI application the Disable GFI field. When the check mark appears, the name of the GFI application is greyed out in the Execution Order box. The GFI application is also disabled in the GFI pulldown in the Main menu. Additionally, the GFI name will not appear on the GFI line during Run mode.

Changing the GFI Execution Order

If you have more than one GFI application and wish to change the order of execution:

- 1 Click Move in the GFI Execution Order field.

The Move GFI window appears allowing you to specify the order in which you want the GFIs to execute.



- 2 Select the GFI to move.

The message line asks you to select the location for the GFI.

- 3 Click the name of the GFI where you wish to place the selected GFI.

Deleting GFI Applications

To delete a GFI application:

- 1 Click Delete in the GFI Execution Order window.



The Delete GFI window appears allowing you to specify the GFI application you wish to delete.

- 2 Click that GFI application.

A check mark appears beside it. The GFI application will be removed when you select OK. You can undelete the GFI before selecting OK or by selecting Cancel.

Refreshing Output Device Files

The Refresh option can be set in the GFI interface to have the Datalog, Aux 1, Aux 2, and Aux 3 files overwritten after each program execution cycle. These files will then contain only one program execution cycle's worth of data at a time. This feature is useful for allowing a GFI to pick up and process one board's worth of data at a time.

The GFI interface is set by default to not overwrite the files thus allowing each board's test results to be appended.

This feature applies only if there is an active GFI application.

IMPORTANT: In most cases where a GFI application needs to read test results, it uses the Tokenlog file. This file is automatically refreshed.

Overriding GFI From the Command Line

To override GFI from the command line use the option

`/o`

as in `> 18xx /o`

This switch allows entry into the system software so that you can disable the GFI in the Setup menu. This command line option overrides only the entry process.

Setting Up Validate You can activate Validate from the Setup menu for capacitor, resistor, and MultiScan tests.

See the **Multiscan User's Guide** for additional information.

For resistors and capacitors, Validate allows the program to automatically:

- Find guard points
- Find the proper wait and squelch times
- Find the number of measurements to take for signal averaging (if averaging results in better stability)
- Swap poles to produce a better result
- Assign 4-, 5-, or 6-wire modes (if nodes are available)
- Evaluate precise mode

For MultiScan tests, Validate attempts to find the correct frequency, bias current, and measurement threshold for each pin. It is essential that you run Validate for all MultiScan tests. If you do not, the results will not be accurate.

DeltaScan has a built-in check to prevent the generation of erroneous results. If you do not validate DeltaScan tests in Test Properties, the test will fail, and you will be alerted by an error message. In production mode, the DeltaScan test will fail.

Several environmental variables in Validate, including the Automatic Accept function, enable you to control the process, trading off speed for increased accuracy or flexibility. The enhanced functionality of Validate permits faster validation with a higher passing turn-on rate if you are using "Pass-If-Good" strategies.

For detailed information about the Validate function see the section, Validate, in chapter 9.

To run Validate you must have a complete and correct component database and an in-circuit test program with passing interconnect sections (jumpers, continuities, SCs, and shorts).

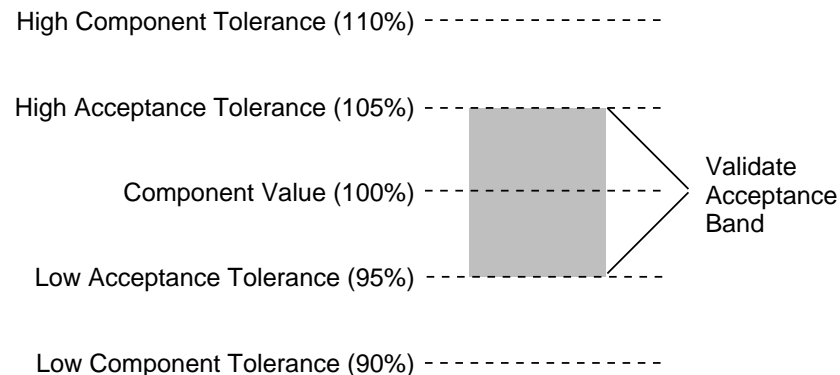
To set up Validate, click Validate in the Setup menu.

The following window appears.

Validate Configuration			
Acceptance Range	50%		
Calculate Wait Times	()	Automatic Accept	()
Average Samples	()	Automatic Exclude	()
Learn Cap Offsets	(√)	Learn with :	Fixture Only
Validate Resistors	(√)	Samples:	32
Validate Capacitors	(√)	Samples:	32
Validate DeltaScan	(√)	Minimum Parallel Resistance (KO):	10
		Minimum Series Resistance (O) :	200
		Nominal (100%) Learn Threshold :	150
		Threshold for First measure of a delta:	1024
		If Threshold exceeded:	Warn
Validate WaveScan	(√)	Threshold Variance Reported :	2
		Minimum Threshold Allowed :	50
Validate FrameScan	(√)	Threshold Variance Reported :	2
		Minimum Threshold Allowed :	30
Validate FScanPlus	(√)	Minimum Threshold Allowed :	25
Validate CapScan	(√)		
OK		Cancel	
Range for accepting measurements (10%–100%)			

Acceptance Range. Acceptance Range sets the thresholds for a “good” reading inside Validate. It is expressed as a percentage of the limits shown on each Step Worksheet. The range tolerance is then used to determine if the component needs validation or not. For example, if the range is set to 20% and a component is measured with a $\pm 10\%$, Validate is not necessary when the actual measurement falls within $\pm 2\%$ (20% of 10%). The range is also used to determine if Automatic Accept needs to be done. The default for Acceptance Range is 50%.

The acceptance band for a 100 Ω resistor with a 10% tolerance and a 50% acceptance range is shown below.



Calculate Wait Times. Validate attempts to attain stability and accuracy for resistor tests by finding appropriate wait times. Calculating wait times, however, increases execution time. If your goal is increased accuracy rather than speed of execution, click within the parentheses to enable Calculate Wait Times.

Average Samples. For resistors or capacitors, Validate tries to attain a more stable result by using averaging. However, averaging increases execution time. The default is that no averaging will take place. If you want to enable average samples, click within the parentheses to enable this function.

Automatic Accept. When you select Automatic Accept, Validate modifies tests that it cannot debug to the given limits so that they will work based on the best readings it can achieve. Validate does this by accepting the measured result as the component's value and then calculating new high and low limits using the original tolerance percentage(s).

The Automatic Accept function executes as the last step of Validate after finding guards and swapping poles processes have been exhausted. Automatic Accept is triggered only if the final result from Validate does not fall within the acceptance band.

Tests "automatically accepted" are marked in the Options/Test Step Controls window both as automatically accepted with the Accepted flag and disabled from further validation by the Validate/Disable flag.

Generating the test step in Test Properties reverts automatically accepted steps to their original status.

A transcript of all the automatically accepted tests can be found in the EXCEPT.LST file.

If Validate can't achieve a passing test and AutoAccept is on, one of two things will occur.

1. If a stable shorthand test is not possible, the test is disabled, and a message is written to the EXCEPT.LST file indicating that the test has been disabled.
2. If a stable test is obtained, but the test results are outside the test limits:
 - The test limits are shifted.
 - The test is marked as validated.
 - The test is marked as AutoAccepted.
 - A message is written to the EXCEPT.LST file saying that this component's test limits have been changed.

If Validate can't achieve a passing test, and AutoAccept is Off, the original test parameters as specified in the Test Properties portion of the Step Worksheet are restored.

A message is added into the EXCEPT.LST file indicating that Validate can't bring test results within the goal tolerance. This allows you to run the section in Stop on Fail mode and manually examine each failing test in the section to determine what action to take.

Automatic Exclude. The Automatic Exclude function excludes from further validation a test step on which Validate has already been run. The function sets the Validate disable flag in Test Step Controls.

Validate/Samples. The Validate/Samples fields in the Validate Configuration window allow you to enable or disable validation for resistors and capacitors and to specify the number of samples to be taken by Validate for each resistor or capacitor measurement. Validate takes between 16 and 255 samples. Placing a check mark between the parentheses after Validate Resistors or Validate Capacitors enables Validate for those components. In the Samples field, enter the number of samples to be taken for each resistor test debugged by Validate.

Taking many samples adds to the execution time of Validate, especially on larger capacitors and resistors. When you specify the number of samples, you must determine whether your goal is speed or reliability in Validate's output. The specified default is 32.

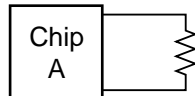
Learn Cap Offsets. Use the Learn Cap Offsets field to enable or disable validation for Cap Phase and PRISM-Z tests which measure small capacitors and RC combinations; however, it

exists for learning only the Offset Cap value for Cap Phase and PRISM Capacitor Test Properties.

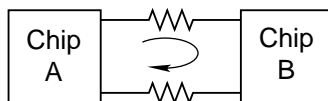
Note that disabling Learn Cap Offsets does not affect learning cap offsets at program run. Refer to the section, Header/Trailer Test Step Editing, in chapter 3 for additional information about Learn Cap Offsets.

Validate DeltaScan. The Validate DeltaScan field allows you to enable or disable validation for DeltaScan tests. DeltaScan validation attempts to find the best combination of pin pairs and thresholds to test each component.

The Minimum Parallel Resistance field enables you to specify the minimum resistance allowed before pins are considered “no path.” The default is 20 Kohms, with a range of 1 to 20 Kohms.



The Minimum Series Resistance field enables you to specify the minimum resistance between two chips with two pins. The default is 200 Ohms, with a range of 50 to 1000 Ohms.



The Default Learn Threshold enables you to specify the minimum validate threshold under which a test or a given pin pair is not used. The default is 150, with a range of 0 to 2000.

Validate WaveScan, FrameScan, and FrameScan Plus. The Validate WaveScan, Validate FrameScan, and Validate FrameScan Plus fields allow you to enable or disable validation for WaveScan, FrameScan, and FrameScan Plus fields tests. The validation attempts to find the correct measurement threshold for each WaveScan, FrameScan, or FrameScan Plus pin. In addition, Validate for WaveScan finds the best bias current to use for testing each pin.

The Threshold Variance field enables you to specify the variation that the system will use to determine whether a message should be written into the EXCEPT.LST file when Validate changes the measurement threshold in a FrameScan or WaveScan test. If Validate determines that the proper measurement threshold varies from the value specified in the Test Properties portion of the Step Worksheet by more than the “Threshold variance reported” value specified in the Validate Configuration Test Properties, it writes a message into the EXCEPT.LST file. If the variance is less than the specified value, a message is not written. The range of acceptable threshold variance reported values is 0 to 255 for WaveScan and FrameScan and the default is 2.

Minimum Threshold for WaveScan and FrameScan Validate. In the Setup/Validate Configuration window, you can specify the minimum threshold allowed; that is, the minimum threshold below which Validate will not allow a test to be generated.

To specify the threshold:

- 1 Click Setup in the Main menu; then click Validate.
The Validate Configuration window appears.

- 2 Enter the appropriate minimum threshold in the “Minimum threshold allowed” field.
For WaveScan, the default is 50, and the range is 50 to 32000. For FrameScan, the default is 30 and the range is 30 to 32000.
For FrameScan Plus, the default is 30 and the range is 30 to 32000.
- 3 Run Validate in a WaveScan or FrameScan test.
See the **Multiscan User’s Guide** for additional information about DeltaScan, WaveScan, and FrameScan validation.

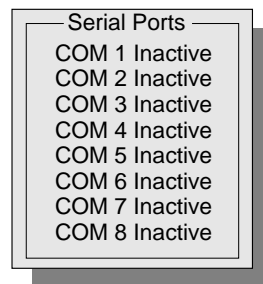
Validate CapScan. The Validate CapScan field allows you to enable or disable validation for CapScan Tests. CapScan tests for the correct orientation of polarized capacitors. Validation for CapScan attempts to produce workable test parameters for a particular device on a particular board, this includes finding an acceptable strong signal to weak signal ratio, and selecting the test frequency to use (ACZ or ACI).

See the **Multiscan User’s Guide** for additional information about CapScan validation.

Setting Up Serial Communications

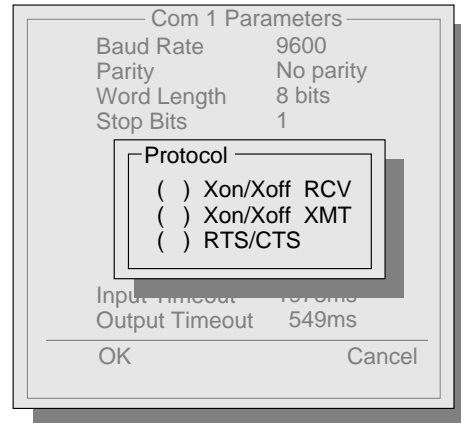
To set up the serial communications port parameters:

- 1 Choose Serial from the Setup menu.
The following pop-up window appears.



If you have not enabled the Com ports (communication ports) in the ASYNC.SYS file, the name of the port is followed with the word, Inactive. If a Com port has already been enabled, the word Active appears after that port. Whether the Com port is active or inactive, the Com Parameter window pops up when you select any Com port so that you can modify the parameters for that port.

- 2 Modify the Com parameters using the selections in the Com Parameters window.



The assignment of communication ports are defined in the CONFIG.SYS file (DOS). Consult the **Computer Configuration and Installation Guide** and ASYNC.DOC file in the \MOS directory for information about ASYNC.SYS com settings and the CONFIG.SYS file.

Click the Baud Rate, Parity, Word Length, and Stop Bits fields to edit them. To enable a field in the Protocol section, click in the open parentheses. To disable a field, click the check mark. To set the Input and Output Timeout fields, click a numerical field and type the appropriate number.

Refer to the following table for more information about the Com Parameter fields.

Com Parameters

Variable	Default	Description
Baud Rate	9600	Sets baud rate of the selected port. Select one of the following: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600.
Parity	Even parity	Sets parity of the selected port. Select one of the following: No parity, Odd parity, Even parity, Sticky Odd parity, Sticky Even parity.
Word Length	7 bits	Sets the word length of the selected port. Select one of the following: 5 bits, 6 bits, 7 bits, 8 bits.
Stop Bits	1	Sets the number of stop bits for the selected port. Select either 1 or 2.
Protocol Section		Click to enable the following fields:
Xon/Xoff RCV	disabled	Enables or disables RCV (receive) xon/xoff flow.
Xon/Xoff XMT	disabled	Enables or disables XMT (transmit) xon/xoff flow.
RTS/CTS	disabled	Request to Send/Clear to Send handshake protocol to supply RS-232 signal by device connected to computer port.
Input Timeout	0 ms	Enter a value from 0 to 14010 ms.
Output Timeout	0 ms	Enter a value from 0 to 14010 ms.

Input timeout is the time the software waits for input characters to arrive when a read request has occurred before returning with an error condition. Output timeout is the time the software waits for the receiving unit to read the output characters before the software returns with a “buffer full” error.

IMPORTANT: Valid times are in 1/18 sec (54.9 ms) increments. The number you enter will be rounded to the nearest 1/18 of a second (in whole milliseconds).

Press Cancel to exit the window without making changes. Press OK to make changes temporarily until you reboot the computer. On the Main menu bar, Save saves the settings made since the last Save. Revert returns to the last settings saved.

Changing Screen Colors

You can change the software system’s color scheme two ways:

- In the Color window.
- From the command line with Palamod

Palamod is a TSR you can use to change the colors of the VGA card. It has a palette of 16 active colors, each of which can have up to 256 shades. You must load Palamod from the command line in DOS before you can use it.

In addition, you can imbed a color code in an output string with the control character \c using pre-test or post-test options as explained above in the section, Customizing Messages in chapter 3. The most recently imbedded color code overwrites the previous one.

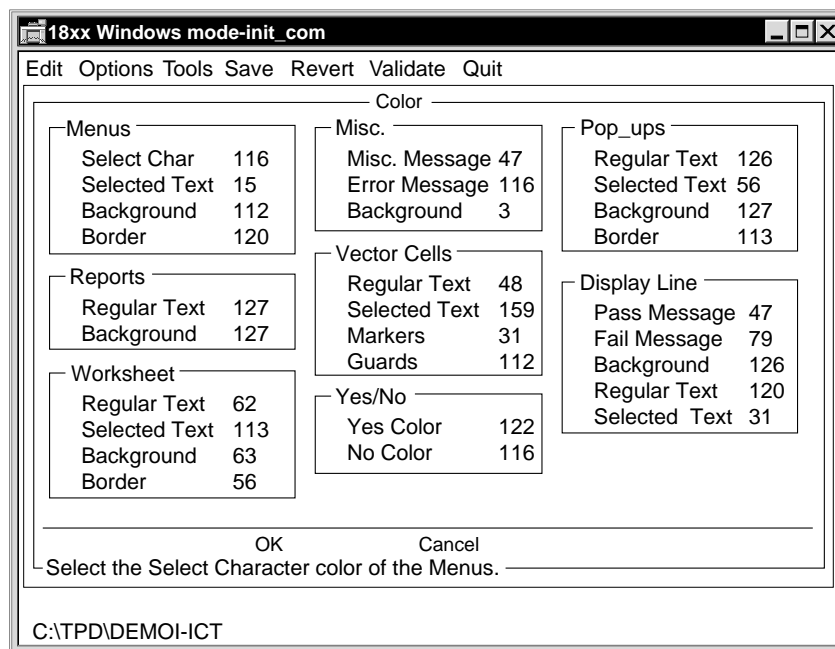
Using the Color Window

Using the Color window is easier than using Palamod to change colors because you can immediately see the results of your color selection in the chosen environment.

To use the Color window to change colors:

- 1 Select Setup from the Main menu and then select Color.

A window similar to the one on the following page will appear.



The Color window has eight sections listing within each the display elements (such as regular text, background) of the software that you can change. When first selected, the Color window displays an example of a menu's current color scheme, as shown above.

- 2 To view a color scheme, move the mouse over an element in one of the eight sections.

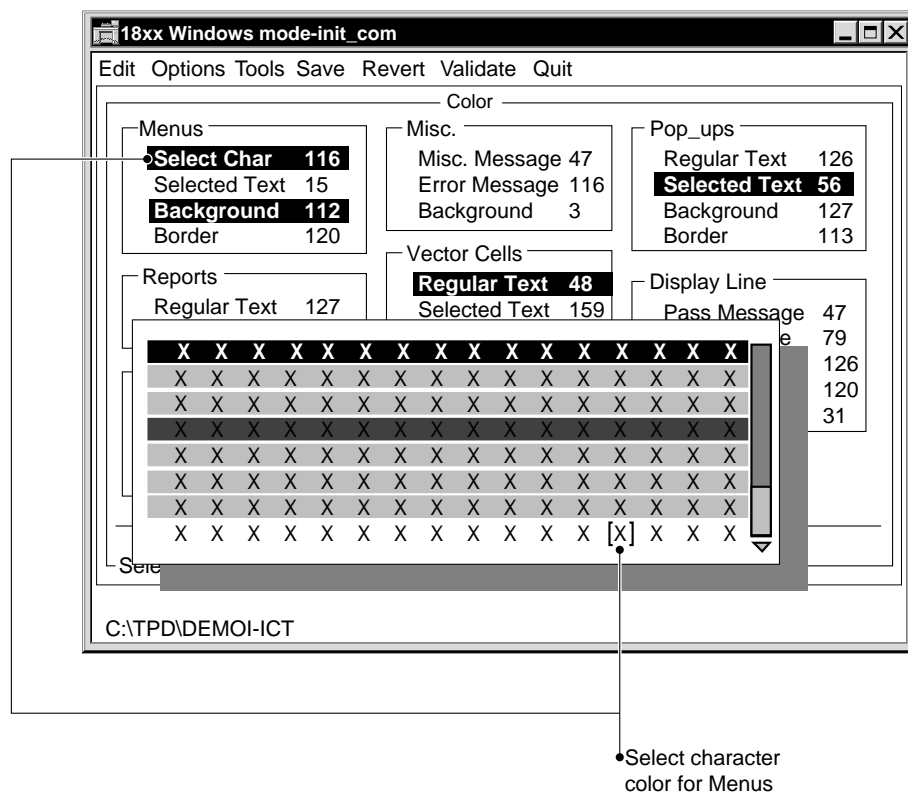
A pop-up window displaying an example of the section's current settings appears.

- 3 To change the color of a screen element, click the mouse in the section again.

A scale of color choices appears. The underlying strip of color in the scale is background color; Xs represent text color. Brackets enclose the currently selected color combination, and the combination also shows up in the pop-up example window.

- 4 Click the X to select the desired combination.

The element in the color window changes to the selected color combination, and the color number, which refers to the Palamod palette, changes to reflect the new combination.



- 5 To select flashing text colors, use the scroll bar on the right side of the scale. The Selected Text field in the Vector Cells section flashes by default.



3 EDITING OVERVIEW

Chapter 3 introduces you to the mechanics of editing a program. It shows how to modify Header and Trailer test steps, access and edit a Step Worksheet, set up Pre-and Post-Test Options, and generate a test to produce the test step.

Navigating in Step Worksheets

You can use both the mouse or keyboard to perform edits in the Step Worksheet. The location of highlighted text shows where in a particular window your cursor is active. Users familiar with the mouse will have no trouble with clicking and pulldown operations. Using the keyboard to effect the same results requires pressing a variety of keys. Instructions for both methods follow. Moving through the Main menu bar and making selections from it are covered in the section, Selecting From a Menu in chapter 1. The following discussion is not exhaustive but is meant to provide general information about mouse and keyboard techniques.

Navigating in the 18xx Editor: Mouse

When you use a mouse in the Component Select window, merely move the cursor to the desired component and click once to produce the Step Worksheet.

After you have selected the desired component from the Component Select window, the component's Step Worksheet appears. With a mouse you can move the cursor to any field. Clicking on the field has the following possible effects:

- It activates a field so that you can type text in it (for example, ID, Name, Desc fields)
- It invokes a pop-up menu in which you can use the mouse to select from a list (for example, Device Type, Value Scale)
- It invokes a pop-up window into which you can enter data (for example, Number of Pins)

To close a pop-up window, click anywhere outside of the window. The window disappears.

Navigating in the 18xx Editor: Keyboard

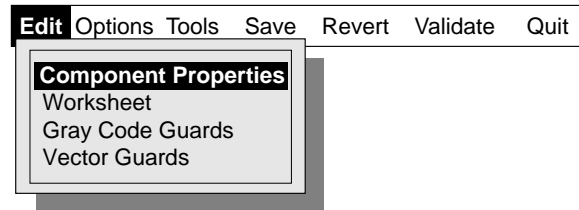
The following keyboard controls have varying effects depending on your location in the interface:

- alphabetic characters,
- the Enter key
- spacebar
- function keys
- numeric keypad plus key
- directional arrows

When you use the keyboard, press Enter to move the cursor from any item in the menu bar into the Component Select window. Once in the window, use the arrow keys to move through the list of components, and press Enter to select the desired component.

After you have selected the desired component from the Component Select window, the component's Step Worksheet appears, and in the menu bar, Edit is highlighted.

Press Enter to expand the Edit menu, and press Enter again to select the highlighted choice, Component Properties.



The up arrow in the Component Properties portion of the Step Worksheet is highlighted showing that cursor control is now in Component Properties. To move through this part of the Step Worksheet, use the arrow keys. When a text field is highlighted, you can type in the desired information.

Pop-up windows lying behind certain fields (Device Type and Number of Pins) can be invoked by pressing the Plus key “+” on the keypad. In the Device Type field use the arrow keys to scroll through the list until you reach the desired choice. Then press Enter to select it. Pressing Enter also returns control to the Edit menu. From the Edit menu either make another selection from the menu bar or activate Component Properties again to make further edits. In the Number of Pins field, use the arrow keys or press Enter to move through the Number of Pins fields. After you have made your edits in that field, you can continue to press Enter or press F10 to close the window and return to the Number of Pins field.

To invoke a pop-up menu from Test Type, press Enter. Use the arrow keys to highlight your selection and then press Enter to select it. Pressing Enter also returns control to the Edit menu.

In the Test Data section for a capacitor, for example, use the space bar to toggle through selections in the Scale and RC Mode fields or press the keypad plus key to see the pop-up window.

In the Controls section, use the space bar to toggle through selections or press the keypad plus key to see the pop-up windows for Wire Mode, Precise, and Higuard. Averaging is a text field.

To get to the other pages of a multipage Step Worksheet, use the Page Up or Page Down keys.

After editing the Test Properties, press F10 to return to the Edit menu.

To invoke the pop-up window behind the Value/Scale field, press the plus key “+” on the keypad. To close a pop-up window, keeping the changes, press F10 which returns you to Edit in the Edit menu. From there either make another selection from the menu bar or activate Component Properties again to make further edits. You can also toggle through the Value/Scale menu without invoking the pop-up window by pressing the space bar.

After you have finished editing the Component Properties portion of the Step Worksheet, press F10 to return to the Edit menu.

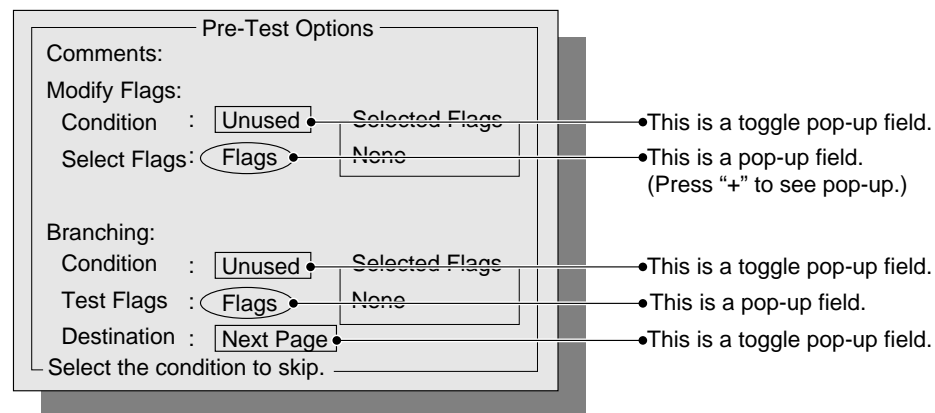
Navigating in Options Fields Using the Keyboard

To navigate in Options, invoke the pop-up windows, and make selections,

- Highlight Pre, Post, or Cntrl and press the plus key on the keypad.
- When the pop-up window appears, make your selections by moving through the list of options with the arrow keys until you highlight the desired field, and then press Enter.
- For Pre- or Post-Test Options, in a text field such as Comments, type in the desired text. Then, press Enter or the down arrow to accept the selection and move to the next field.

- In a Condition or Destination field, you can (1) toggle through the choices without invoking the pop-up window by pressing the space bar, or (2) press the plus key on the keypad to invoke the pop-up window to see the choices. In the first case, either use the arrow or press Enter to accept the selection and move to the next field. In the pop-up window, use the up or down arrows to highlight the desired selection, and press Enter to accept the selection and return to the Options window.
- In a Flags or Select Relays field, press the plus key on the keypad to invoke the Flags or Relay Control windows. To make selections in these windows, use the space bar to toggle through the choices or press the plus key to invoke the pop-up window to display the choices, Clr, Set, or ---. To move through the Flags window, press Enter to proceed through the System Flags, then the Options Flags, and lastly the User Flags. Pressing Enter on User Flag 16 returns you to the Options window.
- To close these windows and return to the previous window press F10.

The following illustration indicates which fields you can toggle through with the space bar and which ones require that you press the keypad plus (+) key to edit the underlying pop-up window.



Introduction to Programs, Categories, and Test Steps

The tester's menu-based system divides programs into categories, sections, and test steps. You can run an entire program, section, or test step, and edit individual test steps as required for debugging purposes.

You can access the test categories by selecting Edit from the Main menu. The following table shows what the categories are and which sections are contained in them.

Category	Section
HEADER	Program header (global variables and messages)
INTC	Discharge, jumpers, continuities, ignores, special cases, shorts, and opens
PASSIVE	Switches/potentiometers, inductors, capacitors, resistors, miscellaneous (connectors), and Rpacks
SEMI	Diodes, transistors, and zeners
PWROFF	Analog power-off devices, DeltaScan, Frame/WaveScan, FS+
BRD_POWER	Board-under-test power
LINEAR	Analog (non-digital) power-on devices
DIGITAL	Digital components and disables
TRAILER	Program trailer (messages)

The actual parts of a test program—Component Properties (test step identification) and Test Properties (test parameters)—reside in the Step Worksheet. The Step Worksheet is the vehicle used to produce a test step.

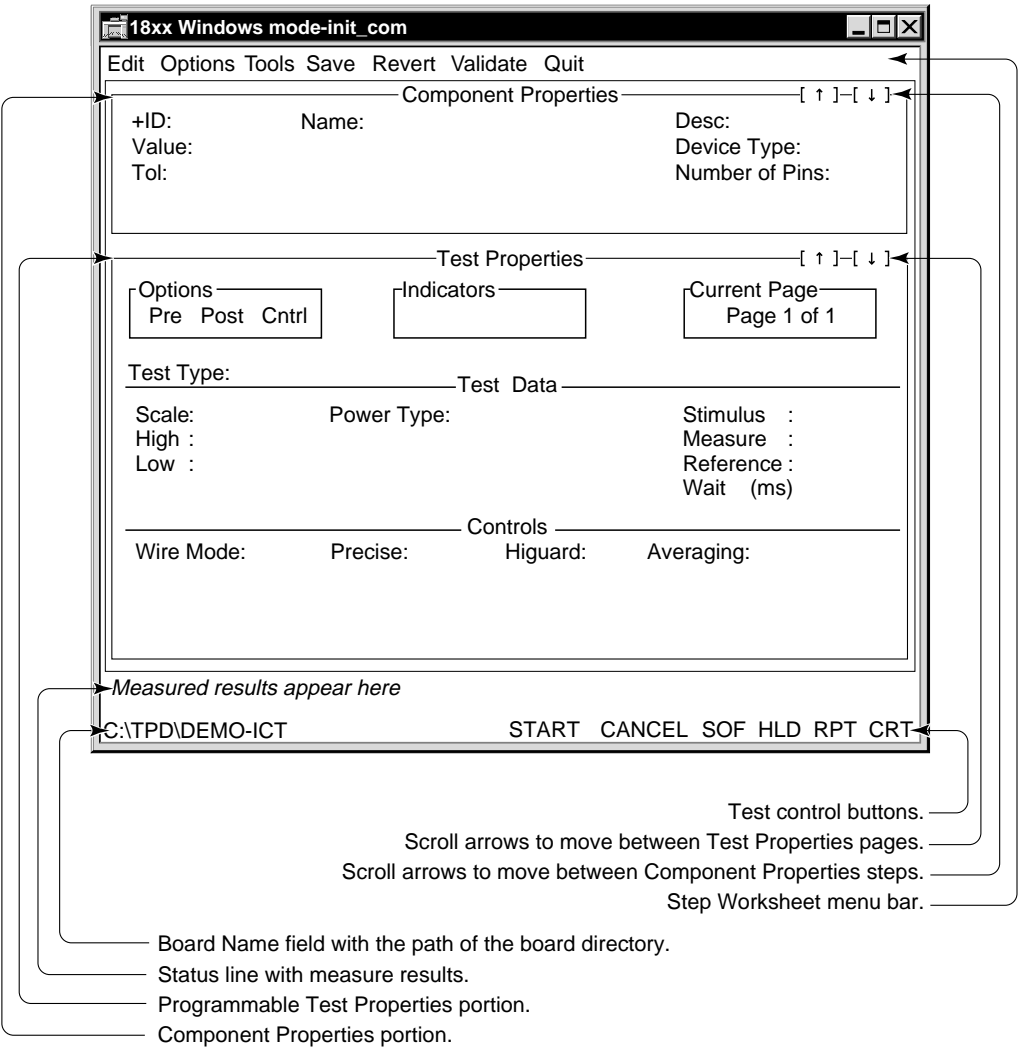
To view or edit a Step Worksheet from the Edit menu

- Select a category (Passive, for example)
- Select a passive component from the pulldown menu (Resistors, for example)
- Select a component from the Component Select window (R2, for example)

Introduction to Step Worksheets

A Step Worksheet contains Component Properties and Test Properties data used to produce the test step. Step Worksheet fields vary among components.

An example of a Step Worksheet with a list of typical features appears below. Details regarding Step Worksheets for each component type appear in the **Z1800-Series Component Test Reference**.



Editing Basics

Program editing chiefly involves modifying a program's test steps in an environment called the Step Worksheet to obtain optimal test results. The Step Worksheet comprises a menu bar, Component Properties and Test Properties. Step Worksheet menus vary in format from device to device, but their overall design will quickly become familiar.

The Edit menu, the second choice on the Main menu, is the starting point for program editing and debugging.

Files **Edit** Run Pgen Setup Utility Templates GFI Convert

From Edit, you can enter a component's Step Worksheet, change test parameters and Component Properties and Test Properties data, run a test, and create new test steps.

Select Edit from the Main menu and the following Edit menu appears.

Header Intc Passive Semi PwrOff Brd_Power Linear Digital Trailer Quit

The menu allows you to select from a category of components tested in the program. When you select a category, either a pulldown menu or Component Select window appears, depending on the complexity of the device type.

The component test categories appear in the order of execution—left to right, then top to bottom. For example, in the Header section of an Edit menu, PRGMVARS appears first and executes first.

For example, when you select Header, the Component Select window lists the Header Step Worksheets contained in the board program. (The board program's path appears at the bottom of each screen.)

If you select Passive, for example, you may select a type of passive component from the pulldown menu below.

Header Intc **Passive** Semi PwrOff Brd_Power Linear Digital Trailer Quit

Switches/Pots..
Inductors
Capacitors
Resistors
Miscellaneous
Rpicks

Relationships Between Test Page Elements

The following table describes the Edit menu's component categories and which sections, test steps or functions fall into them.

Category	Section Description
HEADER	Program Header test steps.
INTC	Interconnect tests: discharge, jumper, continuity, ignores, merging of special cases, shorts, and opens tests.
PASSIVE	Switches/potentiometer, inductor, capacitor, resistor, miscellaneous, and Rpack tests.
SEMI	Diode, transistor, and zener tests.
PWROFF	Analog power-off tests, DeltaScan, Frame/WaveScan, FS+.
BRD_POWER	Unit-under-test power Step Worksheets.
LINEAR	Analog power on Step Worksheets.
DIGITAL	Digital disable and component Step Worksheets.
TRAILER	Program Trailer Step Worksheets.
QUIT	Return to Main menu.

Device Type to Test Type

The Device Type field is in the Component Properties portion of the Step Worksheet. The Test Type field is in the Test Properties portion of the Step Worksheet. When you change the Device Type, the Test Properties page is removed. To get the proper Test Properties for a new device, you must select Tools/Generate Test. Each component test section has a default Device Type.

Within each section, the Test Type selection in Test Properties governs the style of Test Properties and the type of test to be performed. Components can be tested using from only one to one of several Test Types depending on the test situation. However, only one Test Type may be specified on a given Test Properties page.

Test Type to Sections and Categories

The following table shows the relationship of standard Test Types to sections and categories. A standard Test Type is one that is recommended as the most generally suitable type for a device because of built-in test parameters. However, a special test case may require the use of another Test Type. See the section, Shorthand and Longhand Theory, in the **Z1800-Series Component Test Reference**.

Categories/Sections/ Test Types

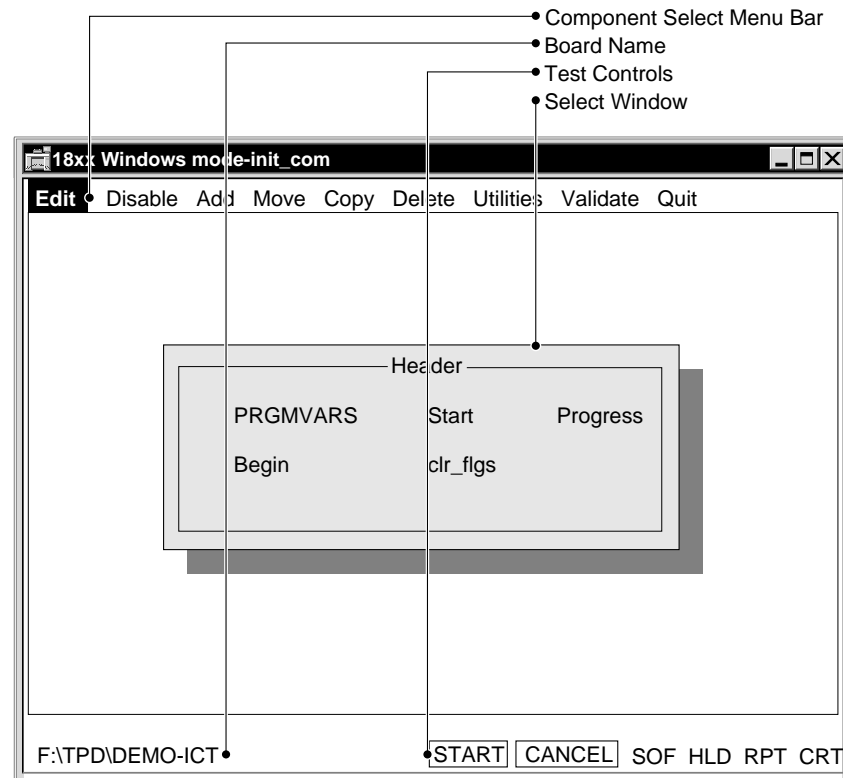
Category	Section	Standard Test Types
Intc	Discharge	Discharge
	Jumper	Jumper
	Continuities	Continuity
	Ignores	Ignores
	Merge_SC	Special Case
	Shorts	Shorts
	Opens	Opens
Passive	Switches/Pots	
	Rheostats	Resistor
	Potentiometers	Resistor
	Inductors	Inductor, Resistor, TVSI, TISV
	Capacitors	Capacitor, Cap Phase, TVSI, TISV
	Resistors	Resistor, TISV
	Miscellaneous	No Test
	Rpacks	Resistor
Semi	Diodes	Diode, TISV, TVSI
	Transistors	Transistor, TISV, TVSI
	Zeners	Zener, TISV, TVSI
PwrOff	Analog	TISV, TISVSV, TV, TVSI, TVSISV, TVSV, TVSVSV
	DeltaScan	DeltaScan
	Frame/WaveScan	WaveScan, FrameScan
Brd_Power		Power, Power 5V, Power Prog 5.5V, Power Prog A, Power Prog B, Power Prog Slaved
Linear		Test V, Test V Stim V, Test V Stim V Stim V
Digital	Disable	
	Components	Gray Code, Vector

IMPORTANT: For discussions of the IEEE and LabWindows Test Types, see the IEEE User's Guide and the LabWindows User's Guide respectively.

Component-Level Test Selection

The Component Select window appears when you have selected a category from the Edit menu. This window allows you to select from all components in a category, run the entire section, and perform other editing functions on the components listed, all without leaving the menu. The Component Select window contains a menu bar, the actual Select window, Board Name field, and Test Control buttons.

A Header's Component Select window is shown below:



The Test Control buttons in the bottom right-hand corner (and corresponding keyboard function keys) allow you to run all the component steps listed in the select window. For example, select Start to run the Header test steps: PRGMVARS, Start, Progress, Begin, and clr_flg. The results of the section run appears in an output window.

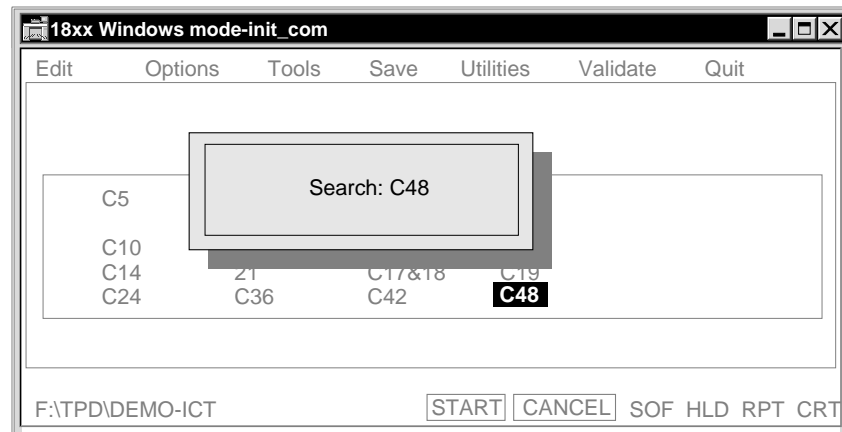
(The Pre/Post options such as Modify flags, and Conditional I/O are also executed.)

Select Window Search Feature. All Component Select windows have a Search feature which is useful for locating components when a large listing scrolls off the screen. To activate the Search window,

- 1 Move the cursor to the Component Select window, thus giving control to this window.
- 2 Type any alphanumeric character to display the Search window.
The Search window appears.
- 3 To locate a particular component, enter its ID.

You do not need to type the entire ID. Search begins looking for a match on the first letter. The Search window accepts only first characters which are first letters in the Component Select window. For example, if you type R in the Search window in a capacitor section containing components beginning only with C, the R will not show appear in the Search window.

When there is a match, Search highlights the component. If that is not the one you are looking for, use the mouse or control arrows to highlight the desired component.



- 4 To bring up the Step Worksheet for the highlighted component, click the component's ID or press Enter.
- 5 To escape from the Search function, select either Edit or Quit, or press F10 or ESC.

Component Select Menus

Selection	Function
EDIT	Select a component from the window for editing.
DISABLE	Disable a test step. Select Disable; then select test step's component ID. Asterisk indicates disabled step. Click component ID to toggle Disable off. To get out of the Disable function, make another selection from the menu bar.
ADD	Add a new component's ID for a Step Worksheet. Choose Add, then choose which component's ID the new ID is to follow. Using this function puts you into the added component's Step Worksheet.
MOVE	Change order in which components are displayed and tested. To move a component, choose Move from menu. Use cursor keys to highlight the component to precede it and press Enter.
COPY	Make a copy of the selected component's Step Worksheets. Using this function puts you into the Step Worksheet. Save on Quit to Copy.
DELETE	Delete unwanted component from program. Deletion does not occur until you reach the Main menu level. If you mistakenly mark a test for deletion, repeat Delete to reverse the process before quitting menu. Tests selected for deletion are marked with a check mark.
UTILITIES	Allows you to build programs from sections of existing programs. Write saves entire section "as is" to a file. You cannot merge or copy one component—the entire component section is copied with all Test Properties. An 8-character field is available to type the file name under which to save the entire section. Press Return after typing name. After saving a component section, Select, Replace, Append, Delete, and Quit commands are available. To merge saved component section, select Quit and return to Main menu to select another board test program. Select board test to which you are copying new component section. Select the corresponding section to which new copied section will be replaced or appended. When you have finished a saved section, select Remove to delete the file. See the Merging section in this chapter.
VALIDATE	Under control of the Validate Setup menu, automatically analyzes resistor, capacitor, and MultiScan tests and makes edits to improve performance.
QUIT	Return to the Edit menu.

Revision Control

Revision Control records all edits made in a particular ICT program. By default, ICT programs are not under Revision Control. If you have the proper 18xx permission, you can enable revision control for any ICT program.

When Revision Control is enabled for an ICT program, all modifications to the ICT program are recorded to an "edit log file." When you exit the editor, the edits are recorded in the edit log file. The entries to the edit log file take place during the pack file process.

The edit log file contains the Section, Step Number, Type of Edit, ID, Name, and Description of each edited test step. The edit log file is a text file that resides in a subdirectory of the board directory. Each time an entry is added to the log file, you are asked to enter your User name and a description of the changes.

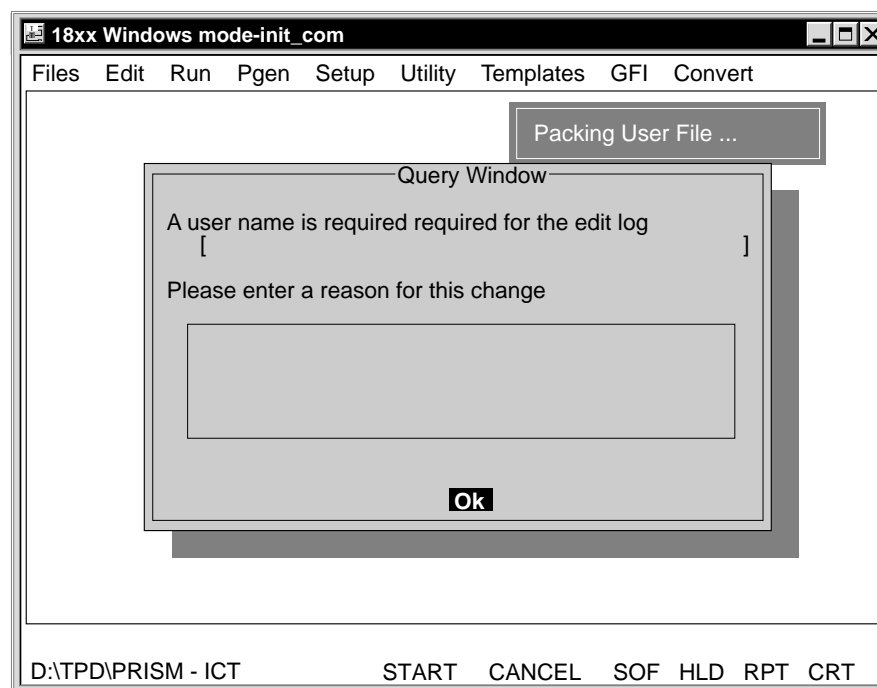
Enabling Revision Control

To enable revision control for a particular ICT program,

- You must have the Revision Control permission
- Revision Control must be On in the PRGMVARS Report Variables window.

Normally the Supervisor has Revision Control permission, which lets the supervisor decide which ICT programs should be placed under revision control. If your login does not have the Revision Control permission, then you cannot edit the PRGMVARS Revision Control field, and the text "<permission req.>" will be displayed next to the field.

Prior to recording the edits to the log file, a Query Window prompts you for your User Name and a Description of the edits.



You must fill in the user name field. The description field is optional. Both fields are text entry fields.

The 1800-Series system software does not provide a mechanism to prevent the logging of edits. If the Revision Control feature is enabled and edits have been made to the program, an entry to the edit log file will be created. The edit log file is named EDITLOG.TXT and resides in a subdirectory to the board directory named ZRCS. This text file can be viewed using a text editor. An example edit log file entry is shown below:

```
Edited by: Joe
Edit Reason: Enabled the Revision Control mechanism.
Edit Date: Thu Jul 2 10:02:16 1998
Section   Step   Action      Id           Name
=====
Header    0      Modified    PRGMVARS     Joe
```

The Section, Id, and Name fields speak for themselves. The Step field is the number of the test step within the section beginning at 0. Step 0 of the Header section is the step named PRGMVARS. The Action field identifies what type of edit has taken place and can be one of Added, Deleted, or Modified.

Header/Trailer Step Worksheet Editing

Step Worksheets fall into seven broad categories—Header/ Trailer, interconnect, MultiScan, board power, analog, Gray code, and vector. Each type of Step Worksheet has features peculiar to its functions. This section discusses editing the Header (PRGMVARS) and Trailer Step Worksheets.

Features common to analog Step Worksheets are discussed in Editing Analog Step Worksheets. Gray code and vector features are discussed in sections Editing Gray Code Step Worksheets and Editing Vector Step Worksheets.

See the **Z1800-Series Component Test Reference** for more information about interconnect and board power Step Worksheets.

See the **MultiScan User's Guide** for detailed information about MultiScan Step Worksheets.

Editing Headers and Trailers

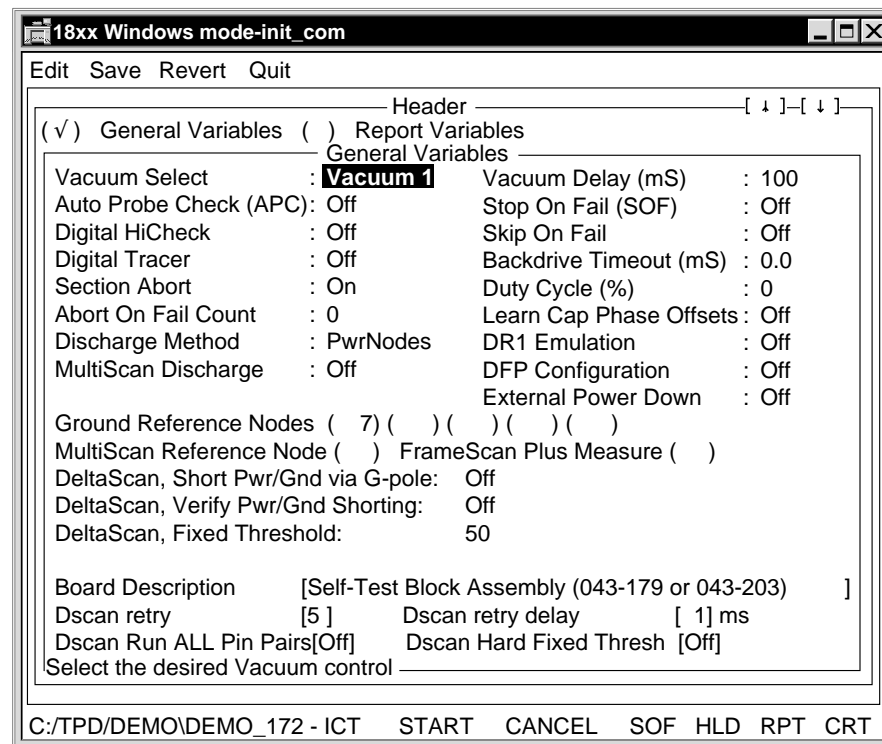
The first category to run in the test program is the Header; the last is the Trailer. Header and Trailer categories may contain up to 1200 Step Worksheets. In the Header category, the first step is reserved as the program variables (PRGMVARS) step. This step contains global flags and variables used throughout the test program. If you turn on a variable in this step, it's always on until you change the setting. (In some cases, variables set in the PRGMVARS section may be overridden by local Step Worksheet controls.)

Setting Up PRGMVARS

Program variables are distributed between two windows:

- General Variables control test program performance and behavior.
- Report Variables control report output.

When you select PRGMVARS, the General Variables window first appears.



You select a field in order to type in it or select from a pop-up window.

General Variables Functions

Field	Choice	Description
Vacuum Select	None Vacuum 1 Vacuum 2 Both Alternate Ignore	If None is chosen, neither vacuum valve is actuated. Vacuum 1 programs the right-hand vacuum port, Vacuum 2, the left. Both (used with large dual-ported fixtures) turn both ports on simultaneously. Alternate (used with tandem fixtures) programs the right port first, then the left, alternating with each run of the program. Ignore causes the system to not actuate any vacuum regardless of the vacuum control switch settings (used with mechanical fixtures). When Alternate is selected, pressing Cancel in the stopped position in the Header, returns vacuum to the right-hand port. If you press Cancel while program is running, vacuum does not return to right-hand port.
Vacuum Delay (mS)	0 to 10000	100 (ms) is the default. Specifies a wait time of the stated duration between actuating the vacuum valve(s) and continuing the program.
Auto Probe Check (APC)	Off On	Controls execution of APC (Automatic Probe Check). A pop-up is displayed : Enable Auto Probe Check On/Off. APC Threshold (nA) determines probe contact; default is 10, range is 5 to 100 nA. DeltaScan APC Threshold (mA) determines probe contact; default is 5, range is 1 to 20 mA.
Stop On Fail (SOF)	Off On	Stops the test step when a failure occurs. Functions the same as the SOF screen button and F5 key. However, if you set SOF on in this menu, SOF will always be on. Normally off during production testing.
Digital HiCheck	Off On	Reports CRC signatures for stuck-high failing measurements as high rather than as the actual signature.
Skip On Fail	Off On	When Skip On Fail is on and a failure occurs, the balance of the test step is skipped. For example, a missing R-Pack would fail on the first test page, skip all remaining pages, and generate only one failure message. If you set Skip On Fail on in this menu, Skip On Fail will always be on. Skip On Fail does not work in Section Run.
Digital Tracer	Off On	Enables Digital Tracer to verify pin contact between fixture probes and DUT for failing digital devices. Uses Nodefinder probe. Enabling Digital Tracer allows you to specify Chip Type (DIP or QUAD), Orientation, Chip Top pin number, Pin 1 Offset, and Second Pass for contact. See chapter 11, "Test & Debug Tools."
Backdrive Timeout (mS)	0.0	Maximum time a digital stimulus may be applied. If any digital burst exceeds this time, the burst will not occur and an error is issued. Enter a value of 0 to default to the Setup/Environment window variables. Range = 0–9999.9 ms. PRGMVARS/Backdrive Timeout overrides the value specified in Setup/Environment window.
Section Abort	On Off	Prevents failures from interfering with subsequent measurements causing false failure messages. Also prevents the tester from applying power to a board that is known to be defective. It sets these flags: Discharge, Interconnect (Jumpers, Continuities, Ignores, Merge SC, Shorts, Opens categories); "Analog (Passive, Semi, Power-Off sections); APC; MultiScan, or Brd_Pwr. When on, a failure occurring in on of these categories or sections sets the abort flag, and the test goes to the Trailer. Turn Section Abort off while debugging your program on a known-good board. Remember to turn it on after completing debug.

Field	Choice	Description
Duty Cycle (%)	0	Ratio of burst time to wait time between bursts for digital tests, expressed as percentage. Range = 0–99%. PRGMVARS/Duty Cycle overrides the value specified in Setup/Environment window. 0 defaults to Setup/Environment window variables.
Abort on Fail Count	0	Specifies the number of failures between 0 and 255 after which the test program aborts. When set at 0, Abort on Fail Count is disabled.
Learn Cap Offsets	Off w/Wired Fixture w/Bare Board	Default is Off. When enabled, either with Wired Fixture or with Bare Board (and after you select Run from the Main menu), 18xx searches for each Cap Phase test in the program, makes measurements and displays/stores the results in the Offset Cap field of Test Properties where you can view all learned values. The subprograms inherit the learned values. When disabled, the subprograms will do what the corresponding PRGMVAR variable instructs.
Discharge Method	PwrNodes DischrgSect None	PwrNodes is the default for pre-E.2 programs. Cannot be used with programmable power supply test types. DischrgSect is default for E.2 programs or later and is preferred method because it runs complete discharge. A "Section Discharge in Progress" message appears during power down even though there may be no entries in the Discharge section. None suppresses discharge phase of power down. See chapter 5, "Test Techniques & Strategies."
DR1 Emulation Mode	Off On	Applies only if you have DR2d boards. Enables you to select whether DUT power comes from backplane or fixture interface. When Off, power comes from fixture interface; when On, power comes from the backplane (source of power for DR1 boards).
MultiScan Discharge	Off On	When On, instructs system software to perform the selected Discharge Method at beginning of WaveScan and DeltaScan sections. Default is On.
DFP Configuration	Off On	When On, enables the Digital Function Processor. Pop-up window allows you to specify source file directory path and communication channel.
External Power Down	Off	Enables custom power-down sequence for external power supplies in older systems. See "Power and Program Execution" in chapter 5 and also chapter 10.
MultiScan Reference Node	9999	Unused tester node dedicated as a reference for MultiScan tests.
Ground Reference Nodes	Empty fields	List of nodes the tester can safely connect to ground while power is on the board or unit-under-test. In power-on analog Test Properties, specifying a reference node that is not in this list enables the differential measurement feature.
FrameScan Plus Measure Node	Empty field	Enter number of node that will be used as the measurement node for all FrameScan Plus and CapScan tests.
DeltaScan, Short Pwr/Gnd via G-pole	Off On	When On, specifies for DeltaScan that if there are not enough ground wires, then use G-Pole for any that are missing.
DeltaScan - Verify Pwr/Gnd Shorting	Off On	Default is Off. Verifies that all power and ground buses have proper grounding. Failure is set up if problems are detected.
DeltaScan - Fixed Threshold	999	The fixed test threshold value used for all DeltaScan tests.
Board Description	String field	A comment that applies to the program as a whole, used in %BOARD macro.

Field	Choice	Description
Dscan retry	5	Specifies the number of times DeltaScan retries a pin pair before marking test as failed. The higher the number, the slower the throughput.
Dscan retry delay	1ms	Sets delay between retries of a pin pair. Range—1 to 99ms).
DScan Run All Pin Pairs	Off	Runs all possible pin pairs. Increases output data; reduces throughput.
Dscan Hard Fixed Thresh	Off	Use to specify that DeltaScan is to use the fixed threshold for all pins rather than the lower of the fixed threshold and one-third of the learned value.

Report Variables. Clicking in the parentheses to the left of Report Variables displays the Report Variables window.

18xx Windows mode-init_com

Edit Save Revert Quit

Header [↑] [↓]

() General Variables (✓) Report Variables

ReportVariables

Datalog	Off	Shorts Locator	0
Status Display Line	Off	Revision Control	Off
Report Prefix	[\n.....\n]		
Report Output Devic(s)	On	Exclude Allprint Device(s)	Off
Report Id	On	Report Component Name	On
Report Description	On	Report Limits	On
Report Meas Nodes	On	Report Nominal	Off
Report Values	Actual	Report Analog Stim Nodes	Off
Allprint	Off	Tokenize Reports	Off
Allprint Message [\c116FAILED\n]		Report Dscan Pin Pair Data	Off

This string is prepended to the report for each page

C:/TPD/DEMO\DEMO_172 - ICT START CANCEL SOF HLD RPT CRT

You select a field in order to type in it or select from a pop-up window.

Report Variables

Field	Choice	Description
Datalog	Off On	If On, appends failure messages to the datalog device specified in Setup Data menu. If Off, sends nothing. Can be overridden by "Report Output" controls in the Pre-Test option variables of a Step Worksheet.
Status Display Line	On Off	Displays test results on the status line. Selecting Off speeds test execution.
Report Prefix	\n____\n	String sent to output devices ahead of test step-specific information on failure.
Report Output Device(s)	OnOffDatalogDiagnosticsLinePrinterIEEEAux1Aux 2Aux 3CRT	With output device choices selected, indicates that at least one is selected. Off disables failure report generation. Off is achieved by deselecting all choices.
Report ID	OffOn	Includes ID string in report message when On.
Report Description	OffOn	Includes Desc string in report message when On.
Report Meas Nodes	OnOff	Reports the measurement node number with all failure messages. Reports the IC's pin number with the node number when On.
Report Values	ActualOffPercent	When set to Actual, the actual failing value is reported in a failure message. When Percent, the failing value is a percentage of the nominal test value. When Off, the failure message will not include a value. Gray code tests report the signature with either Percent or Actual.
Allprint	OffOn	When On, the "failure message" prints for all tests whether they pass or fail. If a test fails, optional message (see below) also prints. Useful for viewing messages and checking program execution. Exclude Allprint Device(s) in PRGMVARS and Test Page Allprint in Edit/Options/Controls provide further datalogging flexibility.
Allprint Message	String field	Additional Allprint string appended to Allprint message when component fails.
Shorts Locator	0-31	Shorts locator depth. 0=off. When enabled, includes in diagnostics device names and pin numbers where shorts occur. Selected number indicates number of component associations for each node number. Useful to limit to range of 1–10 to avoid printing lengthy repair tags.
Revision Control	OffOn	Specifies that revision control is activated for the test program. Monitors test step edits and reports them to a file in the ZRCS subdirectory of the board directory.

Field	Choice	Description
Exclude Allprint Device(s)	OffOnDatalogDiagn osticsLine PrinterIEEEAux 1Aux 2Aux 3CRT	Allows Allprint-generated messages to be blocked from output on specified devices while allowing non-Allprint messages to continue to go through.
Report Component Name	OffOn	Includes Name string in report message when On.
Report Limits	OffOn	Adds the Pass/Fail limits of the test, i.e., High and Low values which the measurement is compared against.
Report Nominal	OffOn	Includes nominal test value in reports.
Report Analog Stim Nodes	OffOn	Adds the analog stimulus nodes to the failure data.
Tokenize Reports	OffOn	Enables output of tokenized reports to datalog file.
Report Dscan Pin Pair Data	OffOn	Saves pin pair data for reporting more than pass/fail results. AllPrint and Tokenlog affect the extent of the reporting. May reduce throughput.

Editing Message Steps

A Header or Trailer message step format allows you to send messages to the operator, control system, Option or User Flags, and/or wait for operator actions such as pressing the Start key—prior to component tests with Header steps and after component tests with Trailer steps. To edit a Header or Trailer message step, select a Header or Trailer message step from the Component Select window.

When you select a message, a window similar to the following appears.

Click a field to activate it.

18xx Windows mode-init.com

Edit Options Tools Save Revert Validate Quit

Header (or Trailer) [↑] [↓]

Step Name: Start (or Cancel, for example, for Trailer step)

Execute this Step On Condition Flags Selected Flags
None

Conditional Execution

Modify Flags: Flags Selected Flags
None

Output Device: On

OutputString: \\n\\t\\t\\c47Press Start to test a board\\n

Input Type: Start Variable: 0

Chain Board Name:

External Programming: Off

Vacuum Control: Keep

C:/TPD/DEMO - ICT START CANCEL SOF HLD RPT CRT

Header/Trailer Message Step Fields

Variable	Choice	Description
Step Name	<string>	Step name. Has no effect on the functionality of the test step. Appears in Component Select window.
Execute this Step on Condition Flags	System Flags Option Flags User Flags	Permits conditional execution of each Header/Trailer message. If conditions are met, the rest of the test step is executed. Otherwise, the program will skip to the next test step. Default is all flags ignored and test step is always performed.
Modify Flags	System Flags Option Flags User Flags	Modify/initialize the selected flags.
Output Device	Off On Datalog, Diagnostics, Line Printer, IEEE, Aux1–3, CRT Tokenlog	Where to send Output String. On indicates an output device is selected. Off indicates no output device selected. Select the Output Device field to pop up the Output Devices window.
Output String	<string>	A message from the program to selected output devices.
Input Type	None Start Alpha Int	Wait for the operator to press the Start button to enter data on the keyboard. None is the default condition.
Variable	0–9	Applies to Int and Alpha only (Input types). Specifies which of the ten variables for the Input Variable are to receive the entered data.
Chain Board Name	String Field	A main program can chain other programs. When the chained program ends, it returns to the main program at the point where the chaining took place. Enter the name of the program to be chained. Chaining depth is controlled in Setup/Environment and is limited to 5. Default depth is 3.
External Programming	Off On	Click or press Off to get pop-up for entering external program call. On indicates the external program call is enabled.
Vacuum Control	Off On Keep	Permits explicit control of vacuum activation and release. Normal mode is On for last message step of the Header section and Keep for all other message steps in the program.

The Selected Flags box in the Header/Trailer message window informs you whether flags are set or cleared. If all flags are ignored (set with “---”), “None” appears in the box.

Customizing Messages

Using Output Macros. Output macros and control characters assist you in creating custom messages in the Header, Trailer, as well as in pre-and post-options windows. These macros and control characters signal the program to insert specific data into the messages when the string is sent through a channel. The macros and control characters are case-sensitive. The following table lists the recognized output macros and the information they provide.

Output Macro	Expands to the ...
%ALPHAn*	nth. Alpha variable (from input Alpha)
%CLEAR_ALPHAn	Clears one of the alpha strings. The n can be a number from 0 to 9 and identifies which string should be cleared.
%INTn	nth. Integer variable (from input Integer)
%CLEAR_INTn	Sets to zero one of the integer variables. The n can be a number from 0 to 9 and identifies which integer should be cleared.
%TIME	Current time and date
%ELAPSED	Elapsed time since the last time Start was pressed.
%CLEAR_ELAPSED	Does not generate any output. Clears the elapsed time variable which is an accumulation of time since the last time Start was pressed.
%MEAS	Last executed analog test's measured value
%HIGH	Last executed test's high limit value
%LOW	Last executed test's low limit value
%ENODE	Last executed test's stimulus nodes
%FNODE	Last executed test's measurement nodes
%GNODE	Last executed test's guard nodes
%BOARD*	Current board description from PRGMVARS header step
%STAT	Last executed test's Pass/Fail/OVFL status
%IEEE*	Last result string from IEEE channel
%NAME*	Current step's Name field from Component Properties portion of the Step Worksheet
%ID*	Current step's ID field from Component Properties portion of the Step Worksheet
%DESC*	Current step's Desc field from Component Properties portion of the Step Worksheet
%FAILURES	Current number of failed steps in the program
%OPTn <text>	Modifies the text associated with any one of the option switches. The option switches are the "buttons" shown on the Display Line when the system is waiting for Start. The default text for the option switches is Option 1 to Option 7. The n in the macro can be a number from 1 to 7 and identifies which option switch to modify. The <text> is the actual text that will be displayed on the Display Line for this option switch. A maximum of 10 characters can be specified as the new text.

*Do not use recursively in field. For example, do not use %NAME in Name field.

Using Control Characters. In addition to the output macros, the following control characters are authorized.

Character	Description
\n	Newline
\t	Tab
\a	Alarm (beep)
\f	Form feed (clears the CRT)
\cnnn	Color, where nnn is a decimal color code value from the color setup menu
\nnn	Any ASCII character where n is a decimal value from 0 to 255

The procedure to add a custom message to a Header or Trailer or in a Pre- or Post-Test window is similar. For example, to add a custom message specifying date and time in a Pre-Test Options window:

- 1 Open the Pre-Test **I/O Control Options** window.
- 2 Select an I/O Condition other than Unused to activate the Output String field.
- 3 Click the **Output Device** field.
A pop-up window listing the output devices will appear.
- 4 Select the output devices.
- 5 Highlight the **Output String** field and type in a message such as:

```
\c47Today's date and time is %TIME:\n
```

where \c47 is the control character specifying that the message appear in white letters on a green background and \n specifies newline.

When you run the program, the following message will appear if the condition is met:

```
Today's date and time is Thu July 11 08:40:05 1996
```

You can fully customize the standard failure report tag with the output macro capability. For example, turn off the Report Meas Nodes and Report Values fields located in the Header/PRGMVARS window. Enter the following string into the Desc field of the Component Properties portion of the Step Worksheet:

```
Test %STAT - Result = %MEAS\nMeasured from %ENODE to %FNODE\n
```

To customize report tags for an entire program, enter the quoted string above into the input list's description field for each component and generate the program.

Merging

The merging function expands program editing capabilities by enabling you to import special routines, custom Headers or Trailers or incorporate test steps from other programs or programmers in your test programs, thus saving yourself an appreciable amount of work. Program merging involves manipulating files and working between several directories to move a section from one test to another. These directories include

- the source program directories (usually in \TPD)
- the \PGT\TPL directory
- the target test program directory in \TPD

In brief, the procedure involves using

- Utilities/Write to write the current section out to a storage area—the \PGT\TPL directory. The sections Step Worksheets are stored in a file in this directory. For the purposes of this manual, we call this file a “section snapshot.”
- Files/Select (or Files/New) to specify the target directory
- Utilities/Overwrite (or Append) to replace (or add to the) current section with the copied section

Writing Out Files. To write out a Header for transfer from one program to another

- 1 Select **Header** from the Edit menu of the source program.
- 2 Select **Utilities**.
The Component Select window becomes empty if this is the first time this has been done in a Header, and only Write is enabled in the menu bar.
- 3 Click **Write**.
The Merge File Name dialog box appears.
- 4 Enter the file name that you want to write the section to in the Component Section File Name field.
Specify an 8-character file name. You do not need to give an extension to the name; the file extension is automatically created by the system software based on which section you are writing the file from.

IMPORTANT: For digital components sections with vector tests, all vector-related files are placed in a subdirectory of \PGT\TPL.

- 5 Click **OK** to save your edits or **Cancel** to terminate the procedure.
The entire Header section including PRGMVARS and all the message steps are written to the file \PGT\TPL\<FILENAME>.HDR where <FILENAME> is the name you have given to the file.

The file is written to a storage area in \PGT\TPL where it is available to merge into other test programs.

Custom “section snapshots” stored using the Write function in \PGT\TPL as individual files are not managed as part of the template library. The name of the “section snapshot” file does not appear in the template library listing.

Merging the File. To merge the file you have written out to \PGT\TPL

- 1 **Quit** to the Main menu.
- 2 Select **Files/Select** and choose the target board directory.
- 3 Select **Edit/Header/Utilities**.
<FILENAME>.HDR appears in the list of files.
- 4 Select **Overwrite** to have the test steps from <FILENAME>.HDR replace the previous Header test steps.

IMPORTANT: If you select Append from the Utilities menu, the system does not give any messages indicating that action has been performed. If you click Append a second time, it will write duplicates of all the test steps in <FILENAME>.HDR.

Step Worksheet Editing

All in-circuit programs are made up of individual Step Worksheets. Each component to be tested has its own Step Worksheet. Analog tests can have up to 32 individual Test Properties pages; digital tests have only one.

A Step Worksheet consists of two main portions—the Component Properties and Test Properties. The Component Properties portion of the Step Worksheet, which derives its data from the input list, contains all the information about the component such as Name, Device Type, Value, and Nodes. However, it does not define how a component is to be tested. It is the Test Properties portion of the Step Worksheet, filled in by the program generator or by the programmer, that controls how a component is tested.

Component Step Worksheets have three general configurations—Gray code, vector, and all other tests—to accommodate different test methods. Interconnects pages are different from standard analog Step Worksheets in that you edit one page, a list, or node groups that run when you execute (press Start) on a page.

For information about MultiScan test steps and Step Worksheets see the **MultiScan User's Guide**.

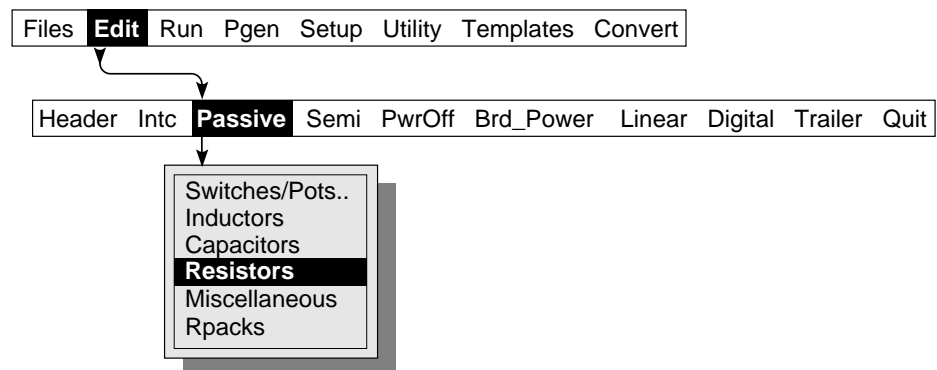
When you select a component from the Component Select window, the Step Worksheet appears. No matter what type of component test you are performing, a Step Worksheet has 3 primary portions:

- menu bar
- Component Properties
- Test Properties

See below for detailed examples of each type of Step Worksheet page.

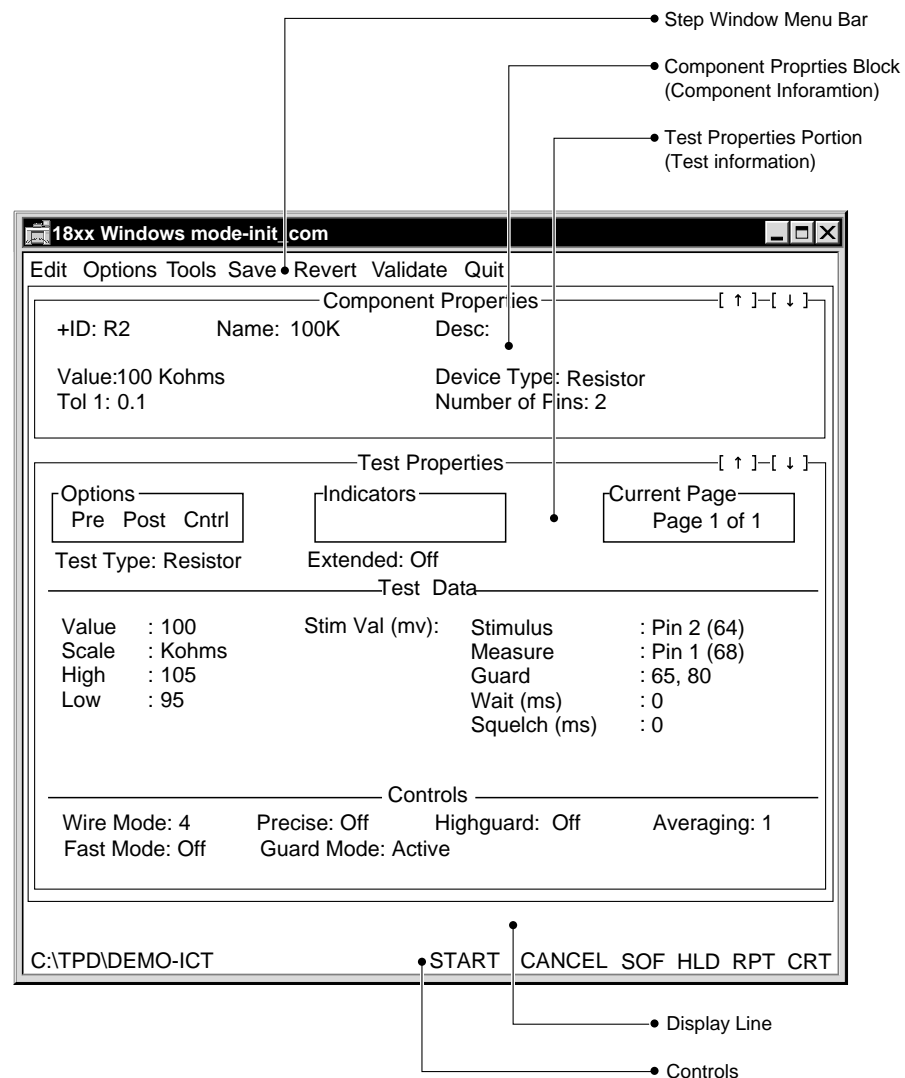
You can edit a test step at any time from the Step Worksheet. To access a specific Step Worksheet, in this case for a resistor—

- 1 Select **Edit** from the Main menu of the board directory.
- 2 Select **Passive**.
- 3 Select **Resistors**, for example.



4 Select a resistor from the Component Select window.

A window similar to the following appears.



IMPORTANT: The Save command saves all changes to all pages of multipage Step Worksheet. Select Revert at any time to revert to the Step Worksheet's test step variables last saved on disk. Remember, if you press the Escape key to return to the menu, you will lose all edits you have made to Test Properties.

Step Worksheet Menu Bar

The Step Worksheet menu bar allows you to edit and administer Component Properties and Test Properties.

Refer to the following table for details.

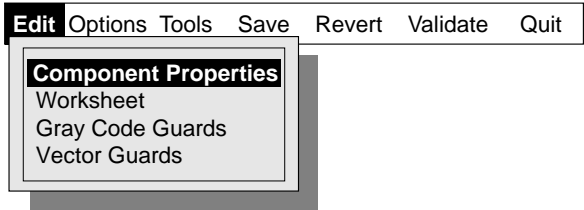
Selection	Description
EDIT	Edit the Component Properties (which, in turn, modifies the input list record), Test Properties, Gray code guards, or vector guards. The last two are available only in the digital component section.
OPTIONS	Edit pre- and post-test variables, flow of control, and I/O control options, and test step controls.
TOOLS	Edit device nodes, get template pins, edit device pins, generate a test, calculate tolerance, create a template, add a Step Worksheet page, delete a page, copy a page, swap test poles, and convert to longhand test, view test parameters, and view fault coverage statistics for Frame/WaveScan tests, inject faults; learn expected data.
SAVE	Save changes made to Test Properties or Component Properties. Stores all page changes. If one page has been modified and another added and saved, the modified page is also saved.
REVERT	Restore the values last saved.
VALIDATE	Find guards for a resistor or capacitor test, correct bias current and measurement threshold for Frame/WaveScan pins, and best combination of pin pairs and thresholds for DeltaScan test.
QUIT	Return to the Component Select window.

Analog Step Worksheet

Editing Component Properties

Component Properties represents the data known about a component—it does not specify how the test is executed. Component Properties is created automatically upon program generation from an input list. An empty Component Properties is created and displayed when you select Add from the Component Select menu.

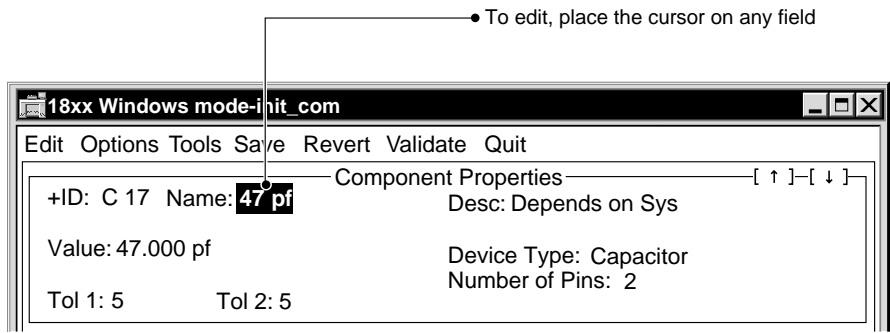
To edit Component Properties, select Edit from the menu bar, then Component Properties—or with a mouse, simply place the cursor on the data field you wish to edit.



Click the Component Properties field you wish to edit. You will either see a flashing cursor, depending on your Color setup, indicating that you may type in the field, or a pop-up menu.

Component Properties contain the following fields: Step Enabled/Step Disabled, ID, Name, Desc (description), Device Type, and Number of Pins. Component Properties for Analog devices may have additional fields for values, tolerances or other parameters.

With the keyboard, navigate through Component Properties by using Page Up/Page Down on the keyboard or click the up or down arrows to go backwards or forward.



The fields in a capacitor Component Properties are shown in the table below.

Field	Type	Description	Data Source*
Step Enabled Step Disabled	+ or *	Enables or disables the test step	
ID:	Text. Max=8	Capacitor test step identification	Input list ID field
Name:	Text. Max=20	Capacitor test step name	Input list CN field
Desc:	Text. Max=64	String for commenting	Input list quoted string
Value:	Text. Max=6 Pop-up	Capacitor value pf, nf, uf, mf	Input list VAL1
Tol 1: & Tol 2:	Text	Capacitor tolerance	TOL1 and TOL2 fields
Device Type:	Pop-up**	Type of device test	C field (capacitor)
Number of Pins:	Text**. Max=128 [†]	Number of device pins	NODES field

*Input list field(s) used by the program generator to fill in this Component Properties field. Refer to chapter 8, “Program Generator Input List Reference,” for input list specifications.

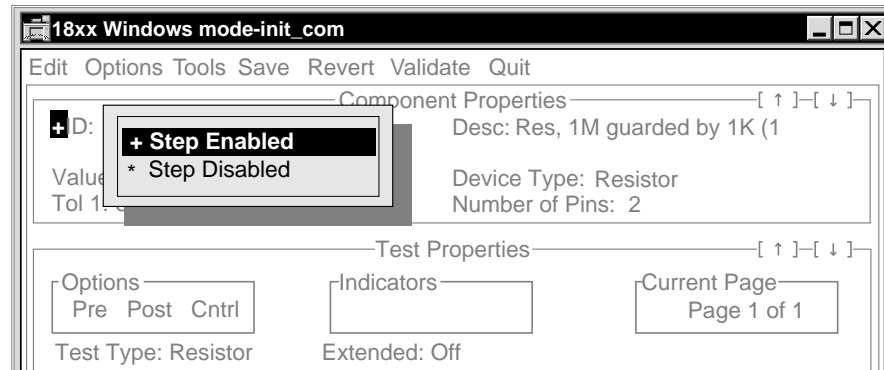
**Edit block appears when you select field.

[†]Maximum of 128 pins for this device.

Component Properties Fields

Step Enabled/Step Disabled Field. Use this field to enable or disable a test step in the worksheet's Component Properties.

This field, which is only one character wide, is just to the left of the ID field in the Component Properties section of the worksheet.



To enable or disable the test step,

- 1 Select the plus or asterisk
A pop-up window appears.
- 2 Select either **+ Step Enabled** or **– Step Disabled**.

When a test is disabled, an asterisk “*” is displayed, and when a test is enabled, a plus sign “+” is displayed. For example, the ID for a disabled capacitor step would look like “*ID: C1”. The “*” character is the same character used to depict a disabled step in the Component Select menu.

Note that Step Disabled is different from the “Test Page Execution” field in the Test Page Controls menus: the former disables an entire Step; the latter disables a single page.

ID Field. The ID field accepts and displays up to 8 alphanumeric characters. A typical ID entry for a component is its designated board/schematic name, such as C1 for capacitor 1.

Refer to the documentation on Pgen/Reports/Topology Report about how this field is used to identify duplicate components.

Name Field. The Name field is primarily an information field. For failure messages, this field is printed out according to the variable in the Header/PRGMVARS. For messages other than failure messages, the %NAME macro must be used to include the field in the message.

For most analog devices, the value or type of device is entered in this field. For analog and digital templated tests, the Name field is the template name.

The field is filled in during automatic program generation; it may also be manually edited. It accepts a maximum of 20 alphanumeric characters, 16 of which appear at a time.

Desc Field. The Desc (component description) field is primarily an information field. For failure messages, this field is printed out according to the variable in the Header/PRGMVARS. For messages other than failure messages, the %DESC macro must be used to include the field in the message.

The field is filled in during automatic program generation; it may also be manually edited. Typical information to include in this field is the part number or board location of the component or descriptive information regarding the component test which will assist the repair person in correcting the fault.

Desc accepts a maximum of 64 alphanumeric characters, 24 of which appear at a time. The Desc field most often contains program comments.

The ID, Name, and Desc fields route to the screen, diagnostic printer, and/or Datalog file when a test reports a failure. You may scroll through the Name and Desc fields to view additional text by holding the cursor down at either ends of the fields.

Value and Scale Fields. The Value field contains the numeric value of the component. When you select the Scale field, you can choose from the pop-up window units of Ohms, Farads, or Henries depending on the component type. In addition, you can choose 6-decimal precision or the default for all analog test types. This precision applies only to the test limits fields—Value, High, and Low.

When you select Scale in a capacitor worksheet, a pop-up window appears from which to choose Scale and Precision.

The screenshot shows the 'Test Properties' dialog box. At the top, there are three sections: 'Options' with buttons 'Pre', 'Post', and 'Ctrl'; 'Indicators' with an empty box; and 'Current Page' showing 'Page 1 of 1'. Below these, 'Test Type: Capacitor' and 'Extended: Off' are displayed. A 'Test Data' section contains a table with the following data:

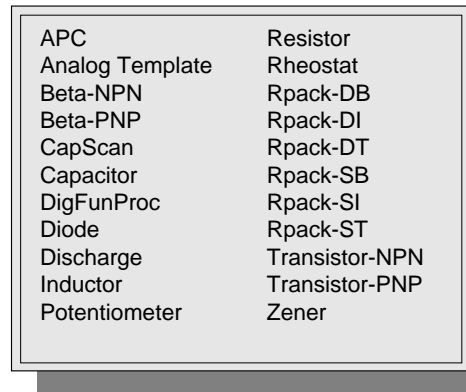
Value	: 100.00	V_dut Pk-Pk (mv):	Stimulus	: Pin 1 (42)
Scale	: nf		Measure	: Pin 2 (68)
High	: 108		Guard	: 72, 91
Low	: 92.		Squelch (ms):	0
RC Mode	: Off			

A pop-up window titled 'Scale Precision' is open over the 'Scale' field. It contains a list of units: 'pf', 'nf', 'uf', and 'mf'. The 'nf' option is selected, and a sub-pop-up shows '6 decimal' as the selected precision. At the bottom of the main dialog, 'Wire Mode: 6', 'Fast Mode: Off', 'Guard Mode: Semi-Active', and 'Averaging: 1' are shown. The status bar at the bottom reads 'C:\TPD\DEMO\DEMO_254 - ICT' and has buttons for 'START', 'CANCEL', 'SOF', 'HLD', 'RPT', and 'CRT'.

For Precision, all existing programs have Default, the backwards-compatible mode, selected. In Default mode, values greater than or equal to 100.0 have 2-decimal precision. All other values have 3-decimal precision.

Device Type Field. The Device Type field is usually filled in by the automatic program generator using the information provided in the input list. If you use instant generation (Tools/Generate Test), the Device Type field determines what kind of Test Properties is produced. Most analog devices—capacitors, diodes, potentiometers, resistors—produce a shorthand test, one that does not require a library template. Of the Device Types shown, Analog Template, keys the system to access the template library in order to generate the test.

To edit the Device Type field, move the cursor to the field and click.



A pop-up window like the one above displays all available device types although you may have to scroll through them using the scrollbar at the right side of the window. To select a Device Type, click the desired type.

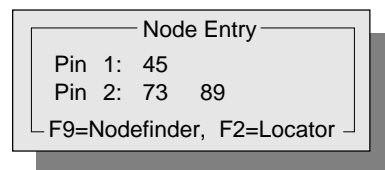
If you change the Device Type, the current Test Properties portion of the Step Worksheet is removed. To generate the proper Test Properties for the new device, select Tools/Generate Test.

Tolerance Field. The Tolerance field is used in conjunction with the Value field to establish the acceptable range of measured values for a passing test. You can enter a one- or two-digit integer number (no decimal point) representing the percentage tolerance to be used in testing the component. The Tol 1 field is for positive tolerance; the Tol 2 field is for negative tolerance.

In an automatically generated program, the value in the tolerance field is extracted from the input list and placed in the component database through the Pgen/Build function and then placed in the program through the Pgen/Generate function.

To alter the testing limits, do not modify the (Resistor) tolerance field in Component Properties. Instead, use the Tools/Tolerance Calculator menu to change the testing limits in Test Properties, or enter high and low limits directly into the fields in Test Properties.

Number of Pins Field. The Number of Pins field is used for test generation purposes both to specify the number of pins on the device to be tested in the test step and to enter the node(s) for each device pin. When you move the cursor to this field, it is highlighted, and you can then type in the number of pins on the device. When you click this field, the Node Entry pop-up window appears, enabling you to enter the node number(s).

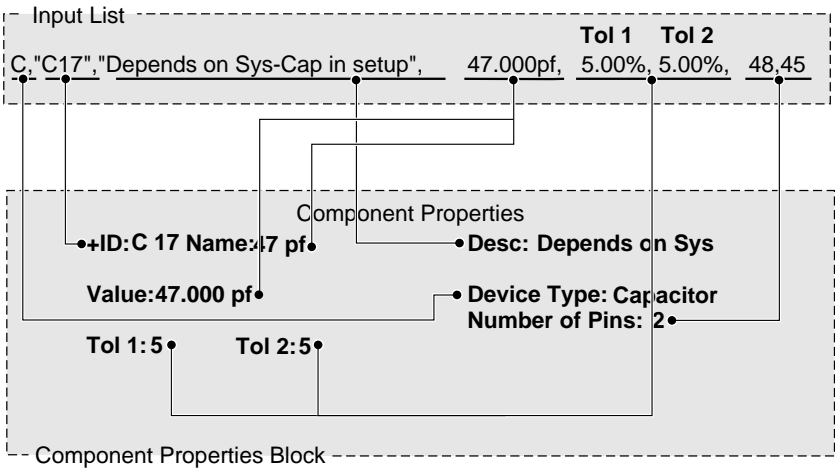


Pressing F9 or double clicking the node number field of the Node Entry box with the left mouse button, automatically invokes the Nodefinder. This action allows you to probe the board and have the system enter the node numbers for you. This technique is helpful in verifying the fixture layout or developing a program without having a noded schematic.

For analog tests, up to five node numbers may be entered for each device pin although typically only one or two pins are attached to a board net.

IMPORTANT: Pin Number refers to a particular lead on the device package. Number of Pins equals the total number of “legs” or device leads on a package/component.

The following illustration shows how the information in the input list is inserted into Component Properties during automatic program generation.



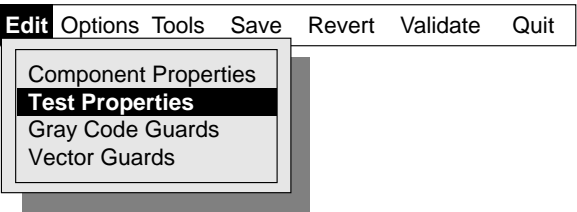
Analog Test Properties Editing

Execution data resides in the Test Properties portion of the Step Worksheet. The Test Properties portion is created after Generate makes the program from an input list, or Generate Test (Tools menu) creates a Step Worksheet. The fields in Test Properties contain the test execution data for a single test step, and they assume different forms depending on the type of component tested.

Analog Test Properties may contain up to 32 pages. Chapter 7, “Component Test Reference,” explains the component-specific Test Properties fields.

Some Test Properties have multiple test pages to perform several tests on one component. For example, transistors have a test page for each junction, and Rpacks have several resistor test pages, one for each Rpack element. Multipage tests are also available for all power-off and linear tests.

To edit Test Properties, select Edit from the menu bar, then Test Properties—or with a mouse, simply place the cursor on the data field you wish to edit.



Input list and Component Properties parameters entered into the Test Properties fields appear below.

Input List

Tol 1

Tol 2

5.00%

5.00%

Options

Pre

Test Properties

Indicators

Current Page

Page 1 of 1

Test Type: Capacitor

Extended: Off

Value :47.000

Scale :pf

High :56.400

Low :37.600

RC Mode:Off

Stimulus : Pin 2 (45)

Measure : Pin 1 (48)

Guard : 57, 38

Squelch (ms): 0

Wire Mode: 3

Fast Mode : Off

Precise: Off

Guard Mode: Active

Higuard: Off

Averaging: 10

C:\TPD\DEMO-ICT

START

CANCEL

SOF HLD RPT CRT

Test Properties Block

Test Properties consists of three parts: the Test Properties data area, the Test Data area, and the Controls area.

The following table lists the typical fields in shorthand analog Test Properties and provides brief explanations of the functions of these fields. See below for more detailed information. These fields vary according both to the device type which determines the configuration of Test Properties that appears after program generation and to the test type which you may select for specialized test purposes.

Test Properties

General Data	Description
Options	Pre-Test, Post-Test, and Controls.
Indicators	Shows page marked for deletion. Read only.
Current Page	Current page of this Step Worksheet. Read-only.
Test Type:	Type of test configuration used on current Step Worksheet.
Test Data	Description
Value	A numeric field specifying the nominal expected value of Scale. Range = 0 – 1000.000
Scale:	Value field's unit modifier based on Test Type, such as ohms, microfarads, volts, etc.
High	The calculated high tolerance in actual measurement value, not percent. Higher measurement than this sets failure flag.
Low	The calculated low tolerance in actual measurement value, not percent. Lower measurement than this sets failure flag.
Stimulus	Pin/node number(s) to which stimulus applied. 1 pin or 5 nodes max.
Measure	Pin/node number(s) at which measurement is taken. 1 pin or 5 nodes max.
Guard	Pin/node number(s) of the analog guard point(s). 1 pin or 10 nodes max.
Wait (ms)	0 to 32000 ms. Added to default Wait of 3 ms typical.
Squelch (ms)	0 to 32000 ms. Repeatable initial condition to remove stored energy.
Controls Area	Description
Wire Mode	Test configuration. Select either 3, 4, 5, or 6.
Precise	ON or OFF. When ON, opens 3/6-wire ATB bridge relays. Selectable for all wire modes
Higuard	ON or OFF. When ON, reduces effects of thermal EMF.
Averaging	Number of measurements averaged. Range = 1–255. Default = 1.
Guard Mode	Provides choice of Active, Semi-Active, or Passive guard modes. Active is default.

Test Properties Data Fields. The upper Test Properties area contains the Options Box, Indicators box, Current Page box, and Test Type. (In capacitor and resistor Step Worksheets, it also contains the Extended field for extended shorthand test.)

The **Options** box in Test Properties provides direct access to the Pre-Test, Post-Test, and Control Options menus. Click the desired item to access the menu. The options can also be accessed from the top menu bar Options selection.

The Options box items—Pre, Post, and Cntrl— change color from white to yellow (standard colors) to indicate that one of the default option variables in a particular option has been modified.

The **Indicators** box, a read-only field, shows if a page is deleted (Del). When selected, it marks the page for deletion, and the indicator lights up. In digital component steps it also shows if Gray code guards (GCgrd) or vector guards (Vgrd) are programmed.

Delete is available only for multiple-page tests. You can delete the last page; however, you cannot delete all pages. Use Delete from the section menu to remove an entire device entry.

The **Current Page** box, a read-only field, displays the active Test Properties page. To scroll from page to page, click the scroll arrows or press Page up or Page Down keys. Pages within a test are executed in numerical order—page 1 is executed before page 2, and so on.

The **Test Type** field states the type of test configuration used by the tester to test the device.

The program generator specifies the test type when Test Properties is generated. Most in-circuit tests are performed using shorthand tests with inputs from input list. Longhand tests like Test I Stim V have stimulus parameters and are programmed in terms of voltages and currents. Use longhand for devices not having a shorthand representation (transformers, relays, FETs, etc.) and for circuit configurations requiring special stimulus.

For most analog tests you may select any of the test types from the pop-up menu at left. Other Test Type fields are available or automatically set in the Interconnect, Board Power, MultiScan, and Digital test sections.

When you change test types, Test Properties fields change accordingly.

The Categories, Sections, and Test Types table shows which Test Types are appropriate for which component tests.

However, you can choose to do other types of tests in most analog sections. You are not limited to a specific Test Type except in certain sections.

Test Data Area Fields. In Test Properties Test Data area, the software checks numeric values for proper syntax and range. You cannot move the cursor off a numeric field if the value is unacceptable. An “Invalid Entry” message appears.

The **Value** field contains the numeric value for the component which is used to setup correct internal stimulus and measurement values. It is filled by the value placed in the corresponding field of the Component Properties portion of the Step Worksheet. The field should have a value between 0 and 1000. Each component has an acceptable range of values.

The **Scale** field is the multiplier option for Value field. Value and Scale fields combine to specify the magnitude of the nominal value of the device or measurement.

The high tolerance value (**High**), calculated automatically when the test is generated, is based on the nominal component value and a percent tolerance. A measurement greater than this high limit sets the failure flag. The High Tolerance (or for capacitors, Tolerance 1) is always stated in percent, but the High Limit (or High field) is always stated in actual value, not percent. Use the Tools/Tolerance Calculator to have the system automatically calculate the actual high and low limit values from the specified percent high and low tolerances. Note that the Tolerance Calculator changes only the High and Low (Limit) fields Test Properties; it does not affect the Tolerance field(s) in the Component Properties portion of the Step Worksheet.

The **Low** tolerance field has the same functionality as the High field except that it holds the low tolerance. A measurement smaller than this sets failure flag.

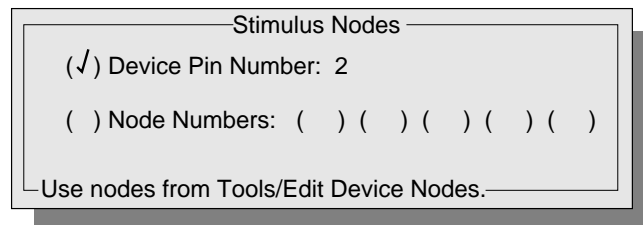
Stimulus, Measure, and Guard Node Fields. The 18xx system software allows you to specify stimulus, measure, and guard nodes in one of two ways. (1) You can use the stimulus and measure nodes associated with the device pins as displayed in the Number of Pins Node Entry window in Component Properties. (2) If they are not suitable for testing, the system software allows you to input any node number for stimulus, measure, or guarding using the stimulus, measure, or guard node fields in Test Properties.

The window operation mechanics are the same for all three types of nodes. The Device Pin Number field has to contain a pin number less than or equal to the number of pins displayed in Component Properties Number of Pins field.

To enter nodes in the Stimulus Nodes pop-up window

- 1 Click the Stimulus field.

A Stimulus Nodes pop-up window similar to the one below appears.



Stimulus Nodes

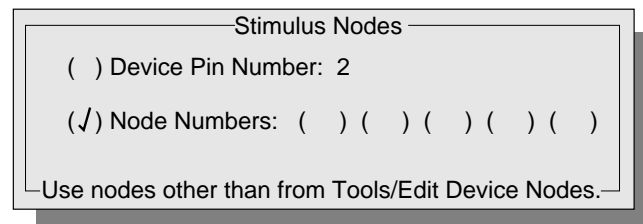
(✓) Device Pin Number: 2

() Node Numbers: () () () () ()

Use nodes from Tools/Edit Device Nodes.

- 2 Click in the parentheses to the left of the Node Numbers field.

When you have done this, the window looks similar to the following.



Stimulus Nodes

() Device Pin Number: 2

(✓) Node Numbers: () () () () ()

Use nodes other than from Tools/Edit Device Nodes.

A check mark appears next to Node Number and the entire Node Numbers field is activated.

- 3 Edit the node fields by placing the cursor on the field, typing the appropriate number, and pressing **Enter**.

The cursor then moves to the next position. After you have entered the maximum number of pins and pressed Enter, the Stimulus Nodes window disappears and Test Properties is displayed. The Stimulus field displays the node numbers you have entered in the Stimulus Nodes pop-up window.

In the Stimulus and Measure Nodes windows a maximum of five nodes may be entered in the range 0 to the highest specified system node. In the largest model Z1800-Series tester, the highest system node number is 5119. You may enter up to 10 nodes in the Guard Nodes window.

The **Stimulus** field represents a test step's stimulus pin and node numbers. During the test, the stimulus voltage is applied to the node(s) designated in the Stimulus field.

For most component tests, the Stimulus node resides on the end of the device attached to the most components. (The measurement pole resides generally on the more isolated end of the component.) Placing stimulus and measurement on these ends facilitates guarding when required, and reduces the effects of noise.

At times, you may want to swap (reverse) stimulus and measurement poles to get stable test results. You can experiment with this technique by using the Swap Poles command located in the Tools menu.

The **Measure** field represents the test step's measurement node and pin numbers. During the test, the measurement is made on the node(s) designated in the Measurement field.

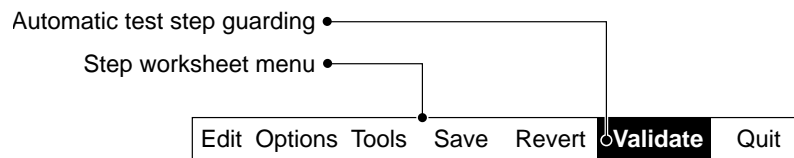
Except in Test V Stim V and Test V component tests in which guarding is not applicable, Teradyne recommends that you make this measurement on that node which is attached to the fewest components. Doing so facilitates guarding when required and minimizes the effects of noise interference on measurement.

At times, you may want to swap (reverse) stimulus and measurement poles to get stable test results. You can experiment with this technique by using the Swap Poles command located in the Tools menu.

The **Guard** field represents the test step's isolated pin and node numbers. Use the guarding technique to reduce or eliminate the effects of parallel current paths.

Edit the Guard nodes the same way that you edit the stimulus and measure nodes.

You may automatically generate a resistor or capacitor test step's guard nodes with the Validate command in the Step Worksheet.



As a rule, a properly guarded test would have the measurement node on the most isolated side of the device. For example, a pull-up resistor with the power rail on one end should have the measurement node on the most isolated end. The stimulus pole would therefore be on the power rail. Select a guard point around the measurement node, ideally one component away to prevent current feedback.

For almost all components, the voltage levels used in shorthand testing are kept very low (200 mV) to prevent turning on semiconductor junctions. Consequently, it is rarely necessary to “guard out” a current path through a semiconductor in a shorthand test.

Shorthand tests for large-value resistors, however, apply larger voltages. For resistors between 2 megohms and 20 megohms, the applied voltage is 1 volt, and for over 20 megohms it is 10 volts.

Squelch removes stored energy from capacitors and inductors on the board-under-test. During the squelch period, the software drives the component to a discharge state by holding stimulus at zero and clamping the measuring amplifier's feedback loop. Squelch always occurs at the beginning of the test step.

All component tests have appropriate preset minimum wait and squelch times specific to the category of test. The tester adds the value you specify in the Squelch fields to the built-in squelch periods. Note that the longer the squelch period, the longer the test time. The Squelch field can accept a value up to 65534 ms.

The **Wait** period is the time between the application of an analog stimulus or power and the taking of the measurement. It is useful for allowing capacitors to charge and readings to stabilize and especially to control the timing in mixed mode testing. The value in the Wait field may be between 0 ms and 32767 ms.

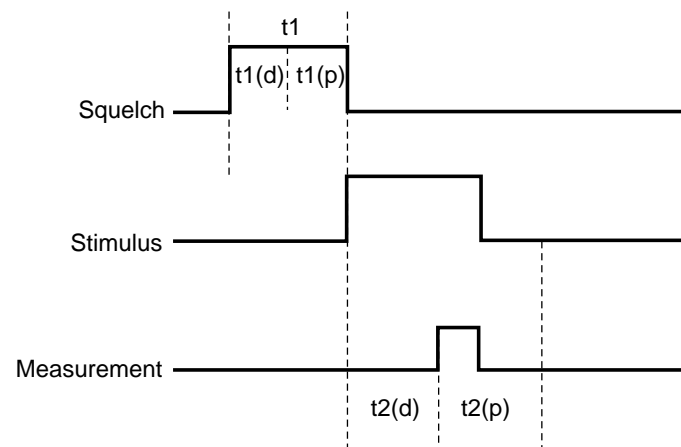
The actual Wait time is dependent upon the circuit being tested. In the case of resistor/capacitor networks, a rule of thumb is to make the Wait time at least 5 times the calculated RC time constant.

You may edit the Wait field in inductor and resistor tests, but not shorthand capacitor tests where Wait time is internally controlled and cannot be modified.

The following diagram shows the timing relationship of squelch, stimulus, and wait states at AC2.

t1= 2 to 10 ms default.

t2= automatic wait time



- t1(d) = default
- t1(p) = programmed squelch
- t1 = t1(d) + t1(p) total squelch time
- t2(d) = default (internally controlled wait time)
- t2(p) = programmed wait

Controls Area Fields. The Controls section allows you to control a test step's execution techniques.

The **Wire Mode** field allows you to select an analog test mode from a pop-up window. 3-wire mode is the standard analog test mode. It uses the E, F, and G buses. 4-, 5-, and 6-wire modes add analog test capabilities to the standard 3-wire test mode. 4-wire mode splits the G bus into 2 buses for separate drive and sense buses. 5-wire mode splits E and F buses into E-drive, E-sense, F-drive, and F-sense respectively for increased measurement accuracy in low impedance measurements. 6-wire mode splits the E, F, and G buses into drive and sense buses.

Wire Mode and Bus Configurations:.

Wire Mode	Stimulus Node		Measurement Node		Guard Node	
	E	Es	F	Fs	G	Gs
x						
3-wire	1	—	1	—	1	—
4-wire	1	—	1	—	1	1
5-wire	1	1	1	1	1	
6-wire	1	1	1	1	1	1

Refer to chapter 5, “Test Techniques & Strategies,” for explanations of Wire Mode and the Precise attribute.

Precise mode, when turned on, opens the 3/6 wire bridge relays on the analog test board. This causes the stimulus, measure, and guard nodes to be sensed remotely rather than locally on the analog test board. There is some variation in which relays open depending on which wire mode (3, 4, 5, or 6) that you use. See chapter 5, “Test Techniques & Strategies,” for detailed information.

Higuard is a measurement algorithm that reduces the effects of thermal EMF induced in some guarded measurements. In situations where the guard ratio is high, the stray currents generated by thermal EMF in the relay contact can cause error in the component test. By measuring and subtracting stray current, your test can determine component values more accurately.

Use the space bar to select On or Off, or click the field to activate the pop-up selection window.

The **Averaging** field digitizes and averages analog signals. Averaging, when on, allows you to take a number of measurements and report their results as an average arithmetic mean. Since multiple measurements take more time than single measurements, your program's test time increases as the number increases. You may enter a number in the range of 1–255. 1 is the default.

The **Guard Mode** field gives you a choice of Active, Semi-Active, or Passive modes of guarding analog devices to eliminate oscillation tendencies when there is a capacitor in the guard path.

In Active mode, the guard amplifier is referenced to the virtual ground (F_s pole) of the measurement op amp. This mode can cause oscillation whenever there is a capacitor between the G and F poles. Active is the default mode.

In Semi-Active mode, the guard amplifier is referenced to the analog ground. This mode does not cause oscillation whenever there is a capacitor between the G and F poles.

In Passive mode, no guard amplifier is used. The G pole is directly connected to the analog ground. You cannot use 4 or 6-wire modes in Test Properties when you are using passive mode.

Refer to chapter 5, “Test Techniques & Strategies,” for detailed information about analog test techniques. Refer the **Z1800-Series Component Test Reference** for information regarding Test Properties specific to each component type.

Gray Code Step Worksheet

A Gray code Step Worksheet has the same 3 primary sections as an analog Step Worksheet.

To edit Gray code Test Properties,

- 1 Select **Edit** from the Main menu for a desired board directory.
- 2 Select **Digital**, and then select **Components**.
- 3 Select a Gray code component from the Component Select window.

A Test Properties page similar to the following appears.

18xx Windows mode-init_com

EditOptionsToolsSaveRevertValidateQuit

Component Identifier

+ID: U1 (Gray)Name: 74LS240Desc: Gray Code Simulated test

Number of Pins: 20Device Type: Gray Code

Worksheet

Options

PrePostCntrl

Indicators

Test Type: Gray Code

Low Threshold : 1.6

High Threshold : 1.6

Clock Divisor : 1

Logic Level : 5V

Terminator : 500 O Pull-Up

Pin	Stimulus	Measure
1	LL	
2	F2	
3		CRC 9B1F
4	F3	
5		CRC C492
6	F4	
7		CRC 3D52
8	F5	

Pin	Stimulus	Measure
20	POWER	
19	LL	
18		CRC 8AD0
17	F5	
16		CRC 3D52
15	F4	
14		CRC C492
13	F3	

C:\TPD\DEMO-ICT

STARTCANCEL SOF HLD RPT CRT

Gray Code Component Properties

Component Properties for a digital component, typically has five fields.

18xx Windows mode-init_com

EditOptionsToolsSaveRevertValidateQuit

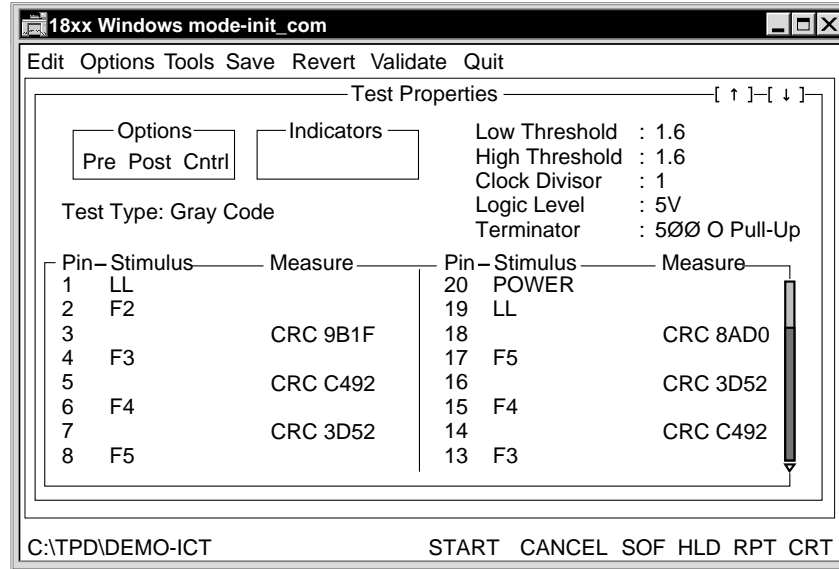
Component Properties

+ID: U1 (Gray)Name: 74LS240Desc: Gray code simulated test

Number of Pins: 20Device Type: Gray Code

Gray Code Test Properties

Unlike analog, Gray code Test Properties have a single page. The Step Worksheet contains the test execution data for a single step. To see the rest of the Pin/Stimulus/Measure portion of Test Properties, use the scroll bar to the right.



Gray code Test Properties consists of the following fields

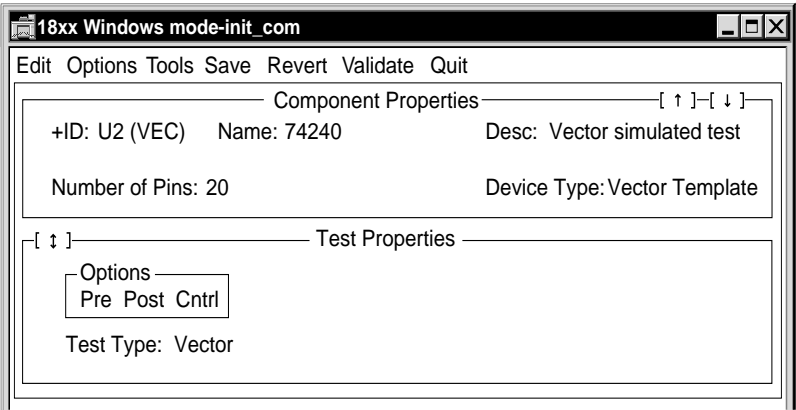
Field	Description
Options	See Editing Analog Test Properties.
Indicators	Shows Gray code guard present (GCgrd), and vector guard present (Vgrd). Guarding indicators, for Gray code tests only, are a quick reference showing if either vector or Gray code guards are present.
Low Threshold	Single or Dual, -2 to +9.5 volts. Default—1.6
High Threshold	Single or Dual, -2 to +9.5 volts. Default—1.6
Clock Divisor	Governs clock rate during digital bursts. Range 1–255. Default is Clock = 1 for a 2 MHz master clock.
Logic Level	For information only.
Terminator	Allows you to apply resistive loads to all measurements. Default—None
Pin	Lists device pin numbers.
Stimulus	Lists stimuli applied to device inputs.
Measure	Shows expected response data for each device output pin.

For further information about Gray code test steps and Step Worksheets see the **Z1800-Series Component Test Reference**.

Vector Step Worksheet A vector Step Worksheet consists of

- Component Properties and abbreviated Test Properties page and
- the expanded Test Properties.

When you first select a vector test step, a Step Worksheet similar to the following appears.

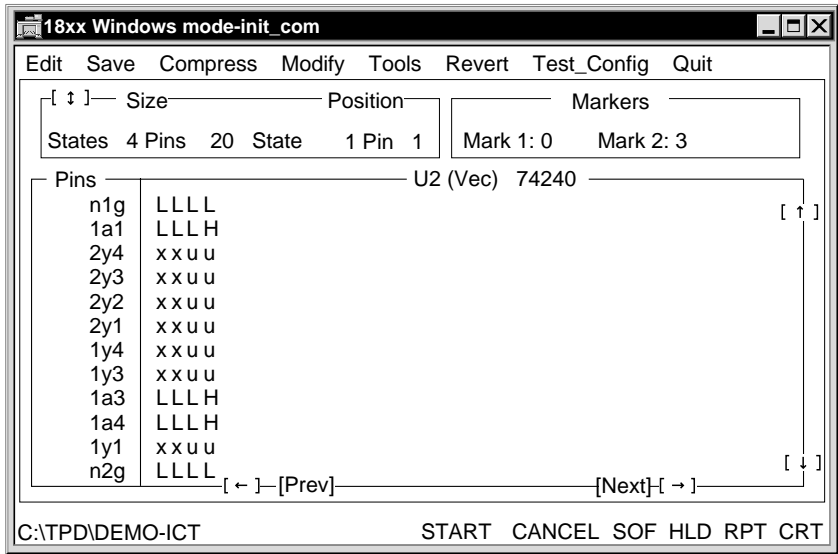


Vector Test Properties

Vector Test Properties (expanded) consists of the following fields

Field	Description
Size	Indicates the overall size of the vector in terms of the number of States and the number of pins used in the current test.
Position	Shows current cursor position within overall vector.
Markers	Placeholders to mark location of interest within vector. To drag, move cursor to top of edit window just above vertical marker bar, hold down the left mouse button, and drag bar to left or right.
Pins	Displays names associated with each device on the pin. Click pin name to edit.
Data Window	Displays vectors with pins on vertical axis and the state at each clock tick on horizontal axis. Using the arrows, scroll to view vectors which exceed window size.

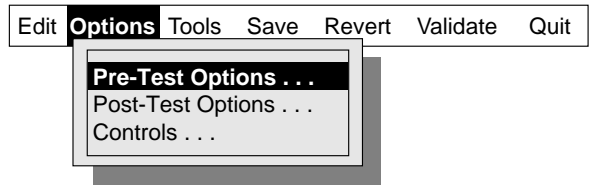
Clicking the toggle arrows displays the Vector Editor menu bar and expands Test Properties as shown below.



For further information about vector test steps and Step Worksheets see the **Z1800-Series Component Test Reference**.

Setting the Step Worksheet Options

Each test page of an individual component has an independent set of Options and Controls. You may edit the options and controls from the Options menu or from the Options box in Test Properties.

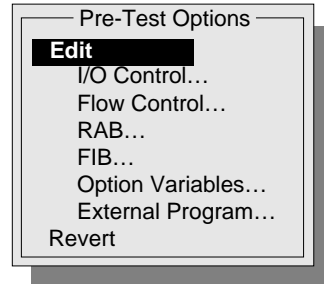


IMPORTANT: Each Test Properties page has an independent set of options fields. Select the correct page before you choose Options. Note that Copy Page also copies the option box contents, and the Add Page function adds a default Test Properties page with default values in the option fields. For more information, see, Order of Program Execution, in chapter 4.

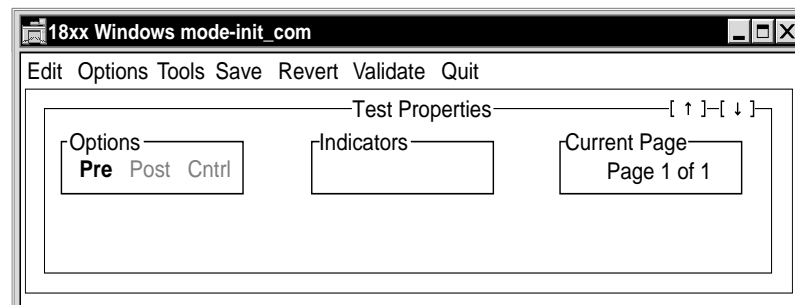
Pre-Test and Post-Test Options

Pre-Test and Post-Test Options submenus contain editing windows that allow you to set a variety of optional conditions for each test step. The default parameters and flags are generally acceptable for basic tests. Pre-Test options are executed before a test page measurement.

Post-Test options are executed after a measurement.



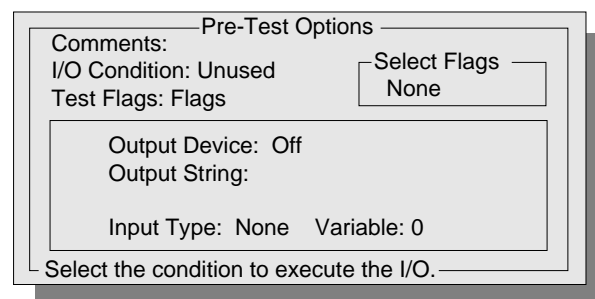
Certain options have default states. If you change a value or flag state, the options box in Test Properties indicates this by changing the color of the selected option—Pre, Post, or Cntrl—from white to yellow or as you have specified in the Color Setup menu. For example, when you make a Pre-Test modification, Pre in the options box changes color.



Controls enable or disable general system features on a page-by-page basis.

I/O Control Options

To edit the input/output control options, select I/O Controls from the Options window. The following window appears.



The window allows you to select the condition to execute the I/O. Select the field you want to edit and type or select from a pop-up menu.

The Pre- and Post-Test Edit windows also provide ways for your program to communicate with the production operator. The I/O is conditional like the Branching function described in chapter 4, “Controlling Test Execution. Its default is Unused. For simple messages that report the progress of the test program, select Always. For conditionals, you may compose combinations of Pass or Fail, and add the System, User, and Option flags.

Three lines allow you to program the I/O operation: Output Device, Output String (a text string with control codes permitted), and Input Type. Select a condition other than Unused in the Condition field to activate the choices.

Output Device field tells the program where to send Output String. String is a programmed message that is sent to the selected output device(s).

The Input Type field allows you to program a specified wait for input condition (pressing Start, for example), or no wait condition at all. The choices are None, Start, Alpha, and Int. None is the default. The Variable field choices are 0–9, and apply to Int and Alpha Input Types only.

Flow Control Options

For detailed information about setting up options in Flow Control, see chapter 4, “Controlling Test Execution.”

RAB

The relay array board (RAB) offers great flexibility in developing your test program by adding up to 32 relays to your tester. RAB allows you to control those relays.

For further information about the RAB, refer to the **Z1800-Series Maintenance Reference**.

You may program the window to Set, Clr (clear), or to ignore (---, the default) each relay.

The Selected Relays box informs you whether relays are set or cleared. If all relays are ignored (set with “---”), “None” appears in the box.

The following screens show the process of changing the relay settings for the Relay Array Board. A Capacitor test is chosen for this example.

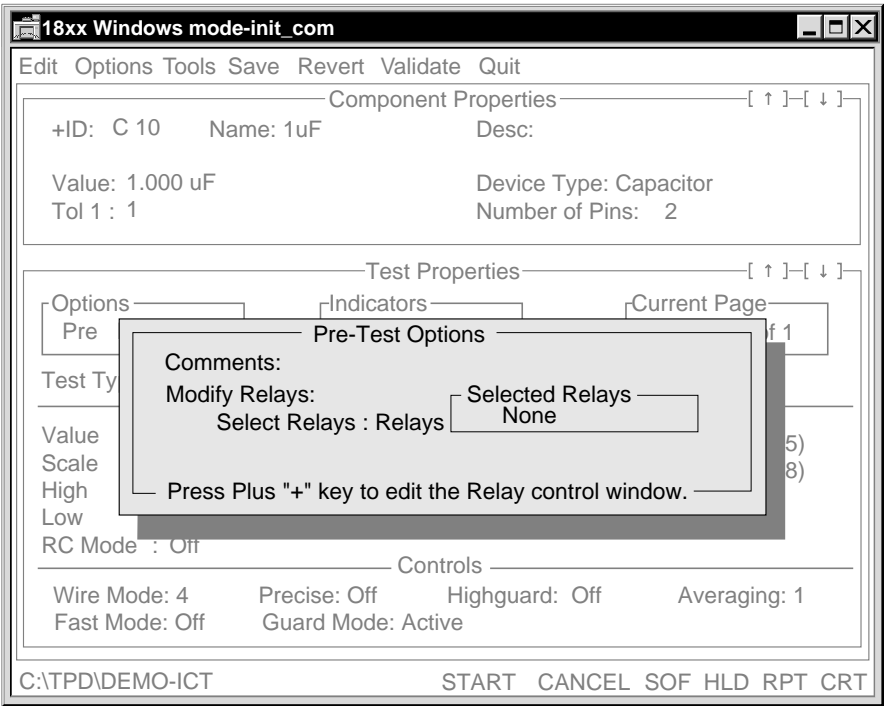
To set and clear relays on the RAB

- 1 Select **Pre-Test Options** from the Options menu.

IMPORTANT: You can program relays from either of the (Pre- or Post-) Options menus.

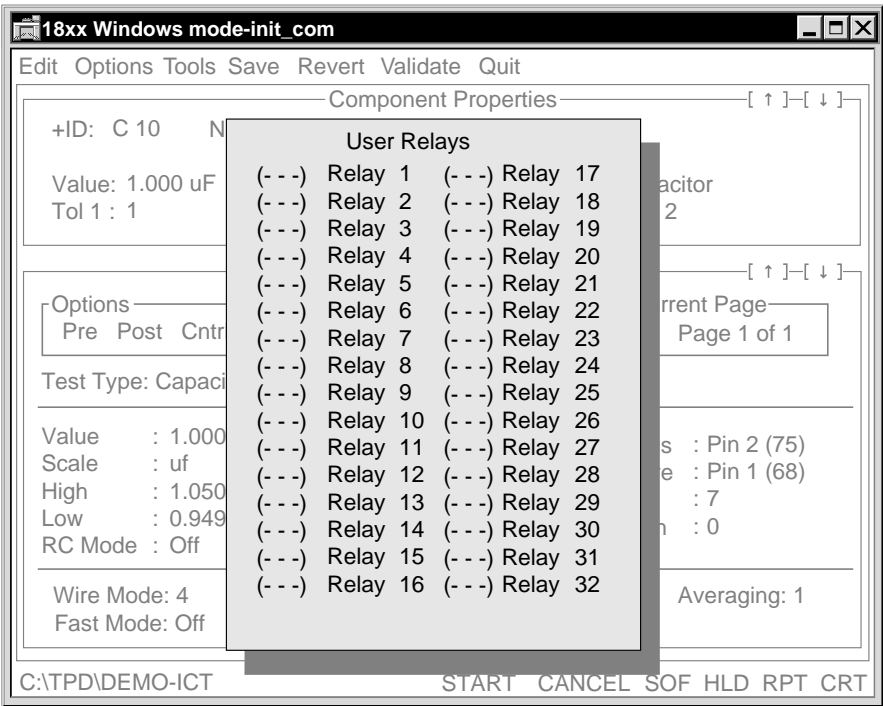
- 2 Select **RAB** from the Pre- or Post-Test Options menu.

The following window appears.

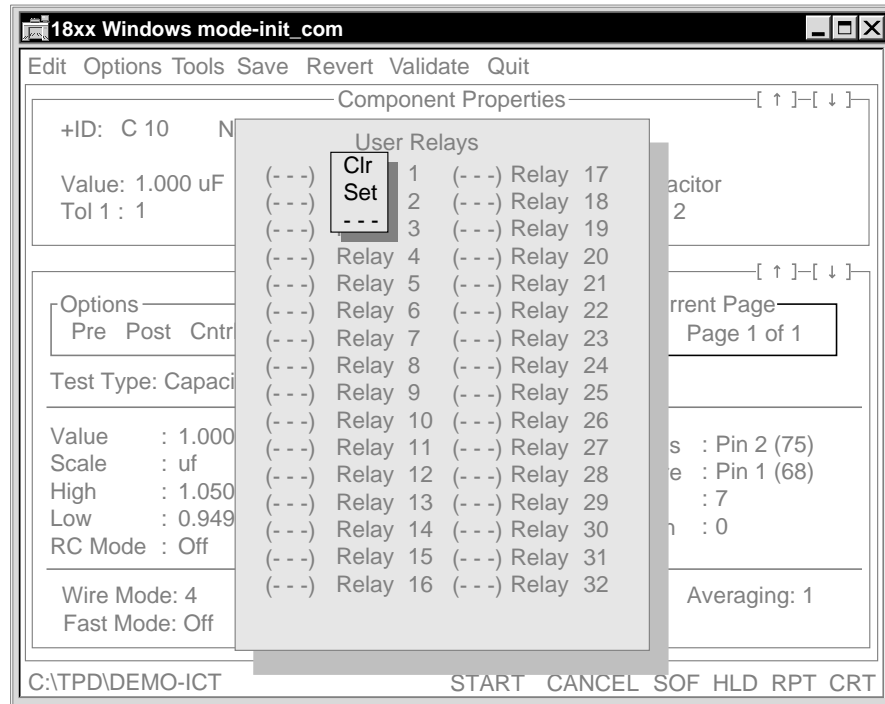


- 3 To modify the relays, click **Relays** in the Select Relays field or press the plus (+) on your keyboard's number pad.

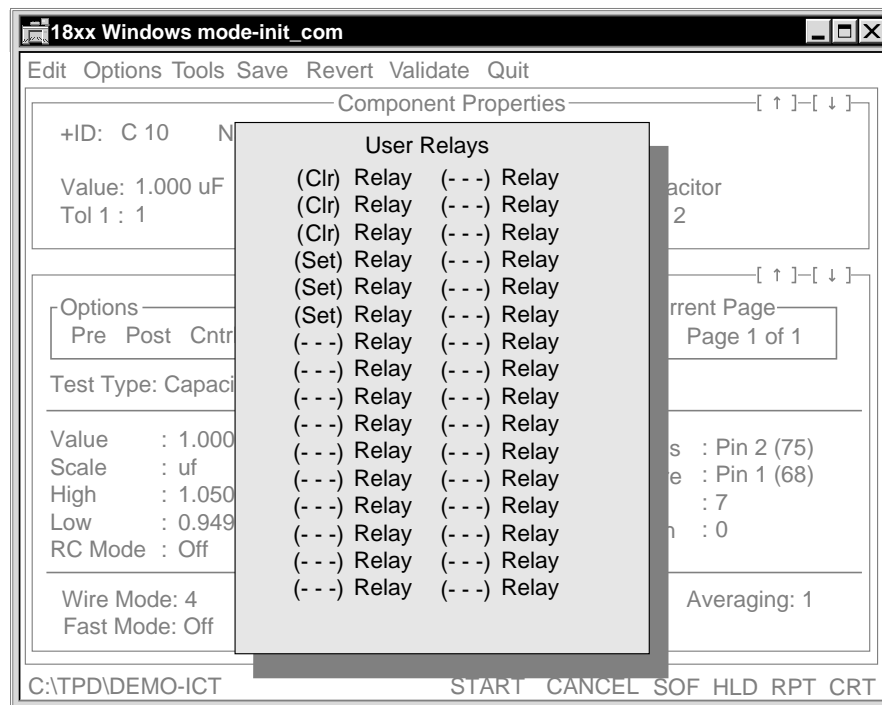
The User Relays window appears, showing entries for the individual relays. In the User Relay window you can set, clear (Clr), or ignore (---, the default) each relay.



- 4 Select the relay to modify by clicking within the parentheses to the left of the relay.
Relay 1 is selected for this example.

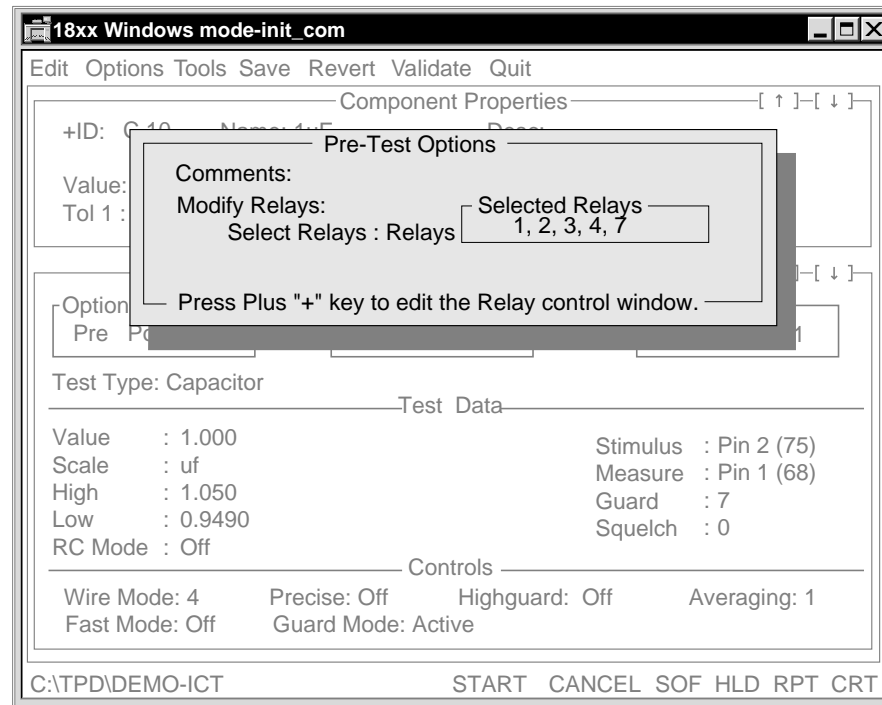


After you have programmed several relays, the User Relay window should look similar to the one shown below. In this example, Relays 1 through 3 are cleared, and Relays 4 through 6 are set. The remaining relays are ignored.



- 5 Click anywhere outside of the User Relays window to return to the Pre-Test Options window.

After you return to the Pre-Test Options window, the Selected Relays box shows that several relays have been modified. If all the relays are ignored, "None" appears in the box.



Relays are cleared automatically at the beginning of the Trailer regardless of where they were set in the program. Relays are also cleared whenever a program is chained and whenever a chained program returns to its master program. If relay changes are programmed, an automatic delay of 10 milliseconds is inserted to allow the relays to settle.

Functional Interface Board (FIB)

When you select FIB from the Pre- or Post-Test Options Window, the FIB Control panel for FIB HB Cluster is displayed. You can toggle between HB and HC cluster types by clicking the Change_Cluster_Type fields.

FIB Clear Control, at the top of the FIB Control panel, specifies whether to clear relays on the FIB at the same time as driver/receiver relays (between each test step) or to clear the FIB relays only

- through Cancel
- through reaching the end of the program, or
- through explicit function commands to clear certain relays.

If you enable the FIB Clear Control, the operating system clears the FIB relays whenever the system issues a Clear Cage command, in addition to when you press the Cancel switch or when the test program ends.

If you disable the FIB Clear Control, the FIB relay is cleared only if it is explicitly programmed to be cleared, or when you press Cancel, or when the test program ends.

If you specify Keep, the FIB Clear Control is kept at its current state whether it is enabled or disabled. The default state of the clear control is enabled.

On the bottom of the window, you can choose Add_Page, Del_Page, Change_Cluster_Type, and Quit.

FIB HB Cluster. You can set or clear all relays in one of the FIB high bandwidth (HB) clusters by using the FIB HB Cluster function panel. The Set/Clear FIB HB Cluster function can initially configure the FIB for testing devices.

Relays in a cluster can be modified incrementally (using Keep) or explicitly (specifying Set or Clear). If you want to modify just a few relays in a cluster and want the rest of the relays to remain unchanged, select question mark (?) for those you want to remain in their current state; Set/Clear for the others. If all the relays and the Clear Control are in the Keep State, the page will not be saved.

The relays in the HB clusters have a maximum current rating of 500 ma.

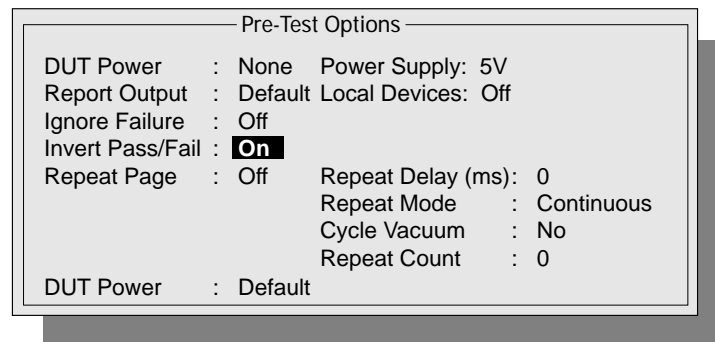
FIB HC Cluster. The FIB HC Cluster sets or clears all relays in one of the FIB high current (HC) clusters. The relays in the HC clusters are rated at 2 amps maximum.

Modify relays in the FIB HC cluster the same as you did in the HB Cluster.

Refer to the **FIB User's Guide** for detailed information about programming the FIB using Pre- and Post-Test Test Properties.

Option Variables

To edit the Option Variables, select Option Variables from the Pre-Test Options window. The following window with options defaults appears.



The screenshot shows a window titled "Pre-Test Options" with a list of settings. The "Invert Pass/Fail" option is highlighted with a black background and white text "On".

Pre-Test Options	
DUT Power	: None Power Supply: 5V
Report Output	: Default Local Devices: Off
Ignore Failure	: Off
Invert Pass/Fail	: On
Repeat Page	: Off Repeat Delay (ms): 0
	Repeat Mode : Continuous
	Cycle Vacuum : No
	Repeat Count : 0
DUT Power	: Default

Click each of the selection fields to view the other choices. Option Variables are not available to Post-Tests.

Power Variables

DUT Power and **Power Supply** have meaning only in power on tests, that is, Brd_Power, Linear, and Digital tests. Modifying these options has no effect if programmed in another section.

DUT Power allows you to turn off all or some of the power supplies for one or more power-on tests, and to restore the supplies to continue power-on testing. Programming this option involves editing the DUT Power field, which in turn affects the availability of the Power Supply field.

DUT Power has three options: None, Restore, and Down.

None is the default value that maintains the tester in its current state.

Restore reapplies a previously powered down supply.

Down turns off a specified supply.

If Down or Restore are selected, the Power Supply field becomes available. Power Supply allows you to specify the supply or supplies to be modified. Possible selections are 5V, 12V or Prog Slaved, 15V or Prog Slaved, ADJ or Prog A & B, ALL, or Prog 5.5V. The selections are used for both the old (PN 045-047-xx) and new (PN 051-002-xx) power supply controllers. 5V and ALL are the same for both controllers. 12V, 15V, and ADJ refer to the old controller whereas Prog Slaved and Prog A & B refer to the new controller. Prog 5.5V applies to the new controller only.

Invert Pass/Fail. Invert Pass/Fail is useful for testing for the absence of a component. The default state is Off. When Invert Pass/Fail is Off, if the measured value of the test page falls within the High and Low limits, it is reported as Passed. If the measured value is outside the High and Low limits, the test is reported as Failed. When On, Invert Pass/Fail reverses the Pass/Fail logic so that if the measured value of the test page falls within the High and Low limits, the test is reported as Failed. If the measured value is outside the High and Low limits, the test is reported as Passed.

Repeat Functions. See Chapter 4, “Controlling Test Execution,” for detailed information about Repeat functions which include Repeat Page, Repeat Delay, Repeat Mode, Cycle Vacuum, and Repeat Count.

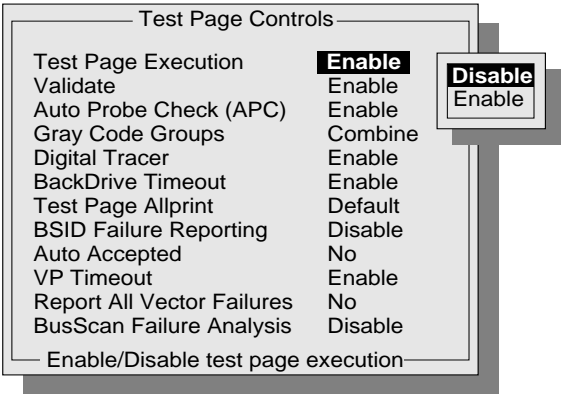
External Program

Select External Options from either the Pre- or Post-Press window, then press the Plus (+) key to display the External Programming edit window. Click an edit option to display available choices.

Test Step Controls

When you select Controls from the Options window, the Test Step Controls window appears. This window allows you to enable or disable Test Page Execution, Validate, APC, Digital Tracer, BackDrive Timeout, and VP Timeout. Clicking the Enable/Disable field brings up a pop-up window in which to make your selection. Clicking the Enable/Disable field for BSID Failure Reporting brings up the BSID Setup window. Auto Accepted is a read-only field that lets you know if the Validate/Automatic Accept process has been run on this test step. Not all fields are available in all test sections.

See below for further information about Gray Code Groups, Test Page Allprint, BSID Failure Reporting, and Report All Vector Failures. For information about BusScan, see the **BusScan User Guide**.



Gray Code Groups

Gray Code Groups controls the execution of Gray code bursts. A Gray Code Group is a logic element within a digital device package. For example, a 7400 chip has four 2-input nand gates; each nand gate is a separate group. Similarly, a 7474 has two D-type FF (two groups).

Combine allows multiple groups to be tested in a single burst; **Separate** allows one group per burst.

Combine is the default. The function combines groups for the minimum number of bursts to effect the most efficient test. Because it reduces the number of bursts, it reduces execution time. However, if there are too many node numbers in the same area (drive or sense) of a single driver/receiver card, they may use the same measurement pole. When this occurs the measurements cannot be combined.

Separate is generally used when the device configuration on the board is such that testing one group would interfere with testing a second group at the same time.

Test Page Allprint

Test Page Allprint allows you to specify the Allprint function on a test page by test page basis. This function facilitates focusing in on a smaller group of components when you are collecting both passing and failing data. For example, if you want to collect all Pass/Fail data on a particular component(s), select Always. Passing and failing data for the specified component(s) is then datalogged without logging such data for all components.

Default = Use the Header Allprint

Always = Override the Header Allprint

BSID Failure Reporting

Boundary Scan Intelligent Diagnostics (BSID) is used to enhance the Z1800-series ability to identify failures when testing boundary scan devices. It is the VICTORY software's link to the system software and enables you to specify diagnostic reporting for boundary scan tests VIT or BICT.

In order to use BSID, you must use the VIT or BICT software to produce the test vectors for the test and to prepare the necessary diagnostic data files for BSID. These files must reside in the directory for the current test program. The file TOPOLOGY.BSC represents the connectivity of the board. For each BICT or VIT test there is a file named the same as the name of the test with a suffix of .SVI (for Scan Vector Information). For details on these files, see the VICTORY manuals.

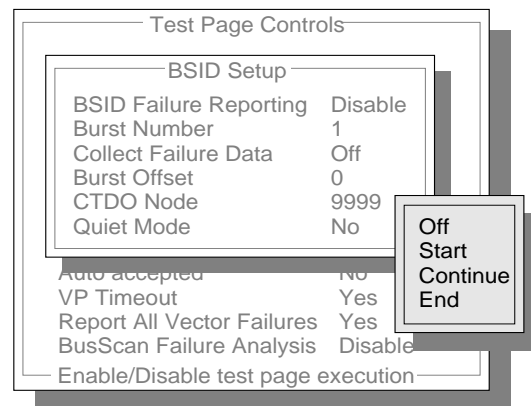
Because some large VICTORY tests require more than one test step, BSID has a Collect Failure Data function to allow you to collect data for the entire test across multiple Step Worksheets. BSID then analyzes the collected data.

BSID Failure Reporting can be selected only if the BSID option has been installed and if the device type in the current Test Properties is Vector. If these conditions are not met, BSID Failure Reporting appears in the Test Step Controls window with Disabled in black, indicating that it is not available.

To setup BSID Failure Reporting,

- 1 Click the **BSID Failure Reporting** field.

The BSID Setup window appears as shown below.



- 2 Click **Disable** in the BSID Failure Reporting field to change it to Enable.
- 3 The default Burst Number is set at 1. However, if you are using a test that involves multiple bursts in separate test steps, set Burst Number in the first test step to 1, in the second to 2, and so on.
- 4 If you have multiple bursts, click **Collect Failure Data** to set up the collection sequence. A pop-up box appears with the following selections..
 - Select Start for the first test step.
 - In the second and subsequent test steps up to but not including the last test step, select Continue.
 - In the last test step, select End to indicate the end of the collection sequence.
- 5 The Burst Offset default is 0. Set this to a number other than 0 only if you have added or subtracted patterns in the beginning of the test. In that case, enter a positive or negative number indicating the number of patterns you have added or subtracted from the test.
- 6 Set the **CTDO (Chain Test Data Out) Node** field to the node that represents the end of the boundary scan chain.
- 7 Enable the **Quiet Mode** function if you want a brief synopsis of diagnosed faults, rather than verbose output.

IMPORTANT: To enable Quiet Mode, you need VICTORY 2.2.

- 8 After setting up BSID, press **Escape** to return to Test Properties.
BSID is invoked only when the test is started from Run in the top level menu or by pressing Start from the digital Component Select window. When the test is run from within Test Properties, BSID is not invoked.

BSID produces its diagnostic report in a file called BSID.DIA, which is displayed in the 18xx software environment and/or is printed, depending on the environment settings. This report specifies which nets on the board failed and gives the device/pins connected to each of those nets. If there are errors in the diagnosis, a message indicating errors appears in the diagnostic report, and detailed error messages are placed in the file BSID.ERR.

For more information about BSID see the **Z1800-Series BSID User's Guide**.

Report All Vector Failures

Report All Vector Failures enables the reporting and display of all failures collected for a pin.

When you click the Report All Vector Failures field, a Yes/No pop-up window appears. No is the default. Select Yes to display the verbose vector failure report on all output devices specified to receive diagnostic reports. Specifying Yes, however, does not affect the Tokenlog data which always receives a truncated report.



4 CONTROLLING TEST EXECUTION

A number of functions in the system software allow you to control the flow of execution to meet individual programming requirements. These functions are found in Setup/Environment Variables, the Files Menu, Header/PRGMVARS, and Pre- and Post-Test options. You can program them to stop the test on a failure, to repeat steps, branch to other parts of the test, and to call or “chain” subprograms.

This chapter explains how the test environment, the function of pre- and post-test options, flow control options and the Multipanel function affect program execution. In addition, it provides examples of how to set up both branching within a program and chaining to a subprogram.

Understanding the Test Environment

Understanding the test environment will help you to fully utilize the tools available for controlling test execution and flow.

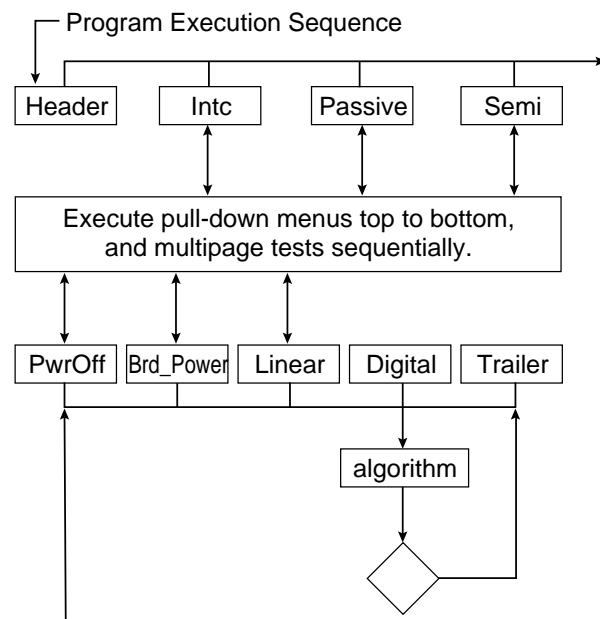
Hierarchy of Control

The structure of the 18xx system software establishes a certain hierarchy of control.

- Setup global tester controls in Setup/Path Data and Device & Channel Data window.
- Use the Header (PRGMVARS and messages) to abort sections on failures and to chain.
- Finally, use Pre-and Post-Test Options, and Controls in the Step Worksheet to govern the flow of the test program at the test step level.

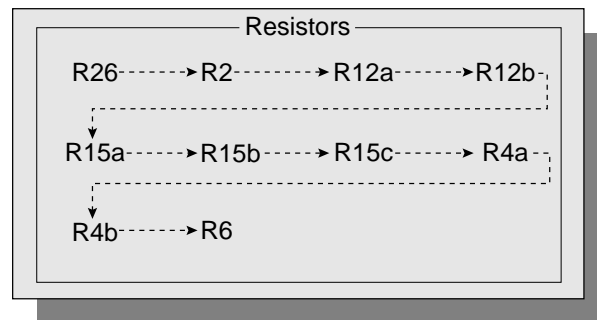
Order of Program Execution

Test programs are executed in the following structured manner without flow control modifications introduced by the programmer.



A program executes test categories in the order displayed across the Edit menu—for example, the Header first, the Trailer last. Within categories (Passive and Semi, for example) execution

proceeds from top to bottom. Within each section (Resistors, for example) execution begins with the component in the upper left hand corner of the Component Select window, and proceeds to the right, then down.



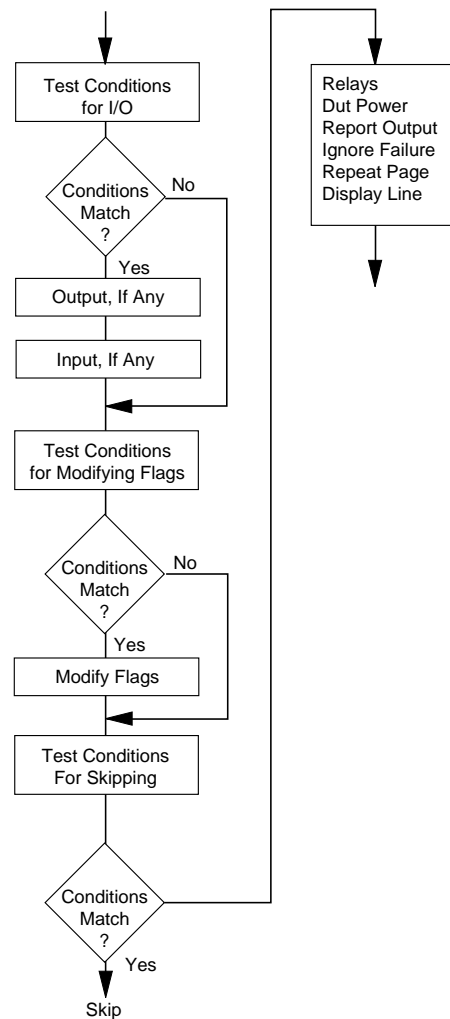
Multipage tests are executed sequentially—page 1 is executed first, followed by page 2, and so on.

Given the basic execution sequence, you can program the execution to branch on condition to the next page, next step, next section, trailer, header, or to repeat section, repeat step, and repeat page. You can also set up calls to external subprograms. Thus you can vary program flow to meet your needs.

Effect of Pre- and Post-Test Variables on Execution

At run time, the Pre- and Post-test variables which you have set up run in a prescribed sequence.

1. Conditional I/O (output first, then input)
2. Conditional setting of flags
3. Conditional jumping of the test. (If skipping occurs, the rest of the page is also skipped.)
4. Relays and DUT Power
5. Report Output (Default/Local)
6. Ignore Failure
7. Repeat and Display



In the Pre-Test box, the test has not yet taken place, and Pre-Test conditions depend on the previously executed test. In the Post-Test box, the current test has taken place, so the “Pass” and “Fail” conditions depend on the current test results. If skipping begins either in a Pre-test page or as a result of stop-on-fail rejection in the current test, the entire Post-Test page is also skipped.

Note that External Program is executed last in Pre-Test Options and first in Post-Test Options.

Master and Subprogram Relationships

You can use programs as subprograms without calling them from a master program. Likewise, you can chain any program that runs on its own from another program.

Chain Depth in Setup/Environment enables you to chain from subprogram to subprogram up to 5 times. The Chain Board Name variable in a Header or Trailer step specifies the board to which a master program may chain.

Start Button Operation. In systems of master and subprograms, you normally setup the program to begin with a prompt to press Start. If you use a program as both master and subprogram, the program should test the subprogram flag (System flag) and perform the Input: Start function only if the subprogram flag is Clear.

Cancel Key Returns to the Master Program. If the production operator presses the Cancel key while the tester is executing a subprogram, the software will return execution to the beginning of the master program's trailer section, which will then be executed.

Vacuum Operation in Subprogram. If you are chaining from the Header of the master program and have not yet activated vacuum, then the subprogram must contain a message step to turn vacuum on.

Rules for Shared Variables. Rule 1. If datalogging is to take place, turn on the Datalog function for both master and subprogram. All Datalog information goes to the master program's Datalog file if the master program has an open Datalog file, whether it comes from the master program or from a subprogram. If the subprogram has an open Datalog file and the master program does not, the subprogram's information goes to its Datalog file. Refer to the table below.

Master program Datalog	OFF	ON	OFF	ON
Subprogram Datalog	OFF	OFF	ON	ON
Master data goes to	—	Master data file	—	Master data file
Subprogram data goes to	—	—	Subprogram data file	Master data file

Rule 2. All system flags except the subprogram flag are passed on unchanged between a master program and a subprogram.

Rule 3. All option flags and user flags are passed on unchanged between a master program and a subprogram.

Rule 4. The alpha variables and the ten integer variables are passed unchanged between a master program and a subprogram.

Subprogram Control with Trailers. Power supplies are turned off in a Trailer, whether in a master program or in a subprogram. Relays are turned off automatically in a Trailer, whether in a master program or in a subprogram. Each program (master or sub-) must therefore manage its own relay and power supply needs.

Only in a master program are vacuum valves turned off automatically at the end of the Trailer.

Program Flow Tools

Setting Up the Environment

In Setup, Chain Depth in Environment Variables is the only parameter affecting flow control. With Chain Depth you can specify chaining up to five levels deep. You can chain any number of subprograms to the master program at the same level, but when a subprogram calls another subprogram, the chaining is said to descend one level.

If you have programmed four levels of chaining but have specified only two in Chain Depth, when the test tries to chain the third program, an error message notifies you that you have exceeded the chaining limit.

Using the Header/Trailer to Control Test Flow

The first category to run in the test step program is the Header; the last is the Trailer. The program variables step (PRGMVARS) in the Header and the message steps in both the Header and Trailer can affect program execution.

Effect of Program Variables on Test Execution. The PRGMVARS step contains flags and variables used throughout the test program. The variables that affect program execution are Section Abort and Abort on Fail Count, both in PRGMVARS/General Variables.

Section Abort prevents failures in an early test stage of a program (shorts, for example) from interfering with subsequent measurements and causing false failure messages. It also prevents the tester from applying power to a board that is known to be defective. Section Abort can occur at the end of Discharge, Interconnect, PwrOff, or Board Power sections if any failures exist. If you turn a variable on in this step, it will always be on until you change the setting.

Abort on Fail Count permits you to set a maximum number of component failures allowed per board test. If the maximum is reached during testing, the test aborts. The range is 0–255.

Effect of Message Steps on Test Execution. A Header or Trailer message step format allows you to send messages to the operator—prior to test with Headers and after a test with Trailers. These messages can instruct the operator to vary program flow when certain conditions occur. In addition, you can set up chaining using the Chain Board Name field in the Header or Trailer message step.

To edit a Header or Trailer message step, select a Header or Trailer from the Component Select window. When you select a message, a window similar to the following appears. Click a field to activate it.

Header

[↑] [↓]

Step Name: Start (or Cancel, for example, for Trailer step)

Execute this Step On Condition Flags

Selected Flags
None

Conditional Execution

Modify Flags: Flags

Selected Flags
None

Output Device: On
OutputString: \n\t\t\t\tc47Press Start to test a board\n

Input Type: Start Variable: 0

Chain Board Name:

External Programming: Off

Vacuum Control: Keep

See chapter 3, “Editing Overview,” for more information about editing message steps.

Test Step Flow Control

Test step flow control is managed for the most part in the Flow Controls function in the Pre-and Post-Test Options. Repeat Page in the Pre-Test Option Variables and Test Page Execution in Pre-and Post-Test also affect program flow.

Flow Control Options

Using the Flow Control options, you can affect the order of program execution by setting up a branching situation. Branching routes program execution to the next page, next step, next section, Trailer, or Header depending on the conditions you have set up. Using the Branching/Destination function, you can also repeat a section, test step, or test page. To see how to set up branching see the Branching example section later in this chapter.

To edit Flow Control options

- 1 Select Flow Control from either the Edit/Options menu or the Pre- or Post windows in the test page Options box.

A window similar to the following appears.

Pre-Test Options

Modify Flags:

Condition: **Unused** Selected Flags: None

Select Flags: :Flags

Branching:

Condition: :Unused Selected Flags: None

Flag Condition: :Flags

Destination: :Next Page

Select the condition to modify the flags.

- 2 To either enter text or select from a pop-up window, click the mouse on a field you wish to edit.

The Comments field allows you to insert program notes of general interest or unusual test configurations. The field accepts up to 64 characters.

Modifying Flags

The Modify Flags field lets you alter the state of the System, User and Option flags based on the pass/fail status of the prior test.

Modifying System Flags. System flags are set and cleared automatically when components are tested. System flags are initialized automatically with each run of the program. Generally it is best to let the program set and clear the System flags unless you have an extraordinary testing or operational requirement. Manually changing the flags can affect your program in ways which are difficult to foresee.

The System flags have the following functions:

- General Failure—if set, indicates to the rest of your program that at least one test has failed.
- Cancel Abort—if set, indicates that the Cancel key has been pressed, sending program control to the program's Trailer.
- Discharge Failure—Discharge failure indicator.
- Interconnect Failure—Interconnect failure indicator.

- Analog Failure—Analog failure indicator.
- Power Failure—Power failure indicator.
- APC Failure—APC failure indicator.
- Master Datalog On—if set, indicates that the master program is datalogging (valid only in chaining situations).
- SubProgram—if set, indicates a subprogram.
- Abort Count Reached—if set, indicates that the number of failures in the program has reached the maximum specified in the Header/PRGMVARS steps.
- MultiScan Failure—WaveScan, DeltaScan, or FrameScan failure indicator. See the **Multiscan User's Guide** for additional information about MultiScan Failure.

To edit System flag conditions,

- 1 Select the Condition field, and then select either Pass, Fail or Always to make the Flags to Modify field available.
- 2 Select the Flags field.

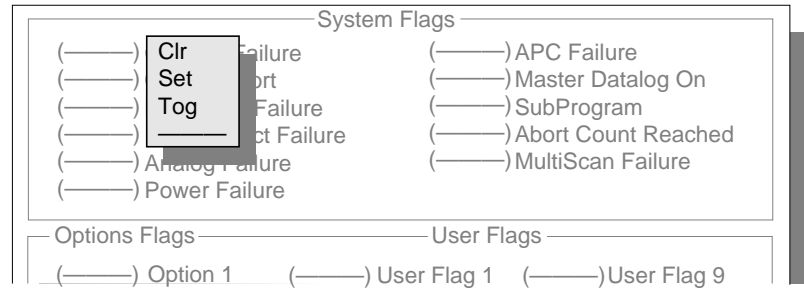
The System, Option, and User flags window pops up.

System Flags		
<input type="checkbox"/> General Failure	<input type="checkbox"/> APC Failure	
<input type="checkbox"/> Cancel Abort	<input type="checkbox"/> Master Datalog On	
<input type="checkbox"/> Discharge Failure	<input type="checkbox"/> SubProgram	
<input type="checkbox"/> Interconnect Failure	<input type="checkbox"/> Abort Count Reached	
<input type="checkbox"/> Analog Failure	<input type="checkbox"/> MultiScan Failure	
<input type="checkbox"/> Power Failure		

Options Flags	User Flags	
<input type="checkbox"/> Option 1	<input type="checkbox"/> User Flag 1	<input type="checkbox"/> User Flag 9
<input type="checkbox"/> Option 2	<input type="checkbox"/> User Flag 2	<input type="checkbox"/> User Flag 10
<input type="checkbox"/> Option 3	<input type="checkbox"/> User Flag 3	<input type="checkbox"/> User Flag 11
<input type="checkbox"/> Option 4	<input type="checkbox"/> User Flag 4	<input type="checkbox"/> User Flag 12
<input type="checkbox"/> Option 5	<input type="checkbox"/> User Flag 5	<input type="checkbox"/> User Flag 13
<input type="checkbox"/> Option 6	<input type="checkbox"/> User Flag 6	<input type="checkbox"/> User Flag 14
<input type="checkbox"/> Option 7	<input type="checkbox"/> User Flag 7	<input type="checkbox"/> User Flag 15
	<input type="checkbox"/> User Flag 8	<input type="checkbox"/> User Flag 16

- 3 Select the flag type to edit.

A pop-up window allows you to modify the flags.



4 To edit in the System Flags window, click System.

Ignore is the default for each item, represented by dashes (---). You may select ignore, clear (Clr), Set, or toggle (Tog) for each of the System flags by clicking in the parentheses of a flag's field.

The Selected Flags box informs you whether flags are set or cleared. If all flags are ignored (set with "---"), "None" appears in the box.

Modifying Option Flags. Option flags 1 through 7 are displayed on the display line during Run mode whenever the tester is waiting for the Start key. The Option flags can be set or cleared by the program, or manually by the production operator any time the tester is waiting for the Start key. The numeric keys 1 through 7 toggle the Option flags, or you can select them with the mouse. You can set the flags by clicking in the parentheses and selecting either Clr, Set, Tog, or ignore (---), in the same manner as System flags.

Modifying User Flags. User flags 1 through 16 are available for any purpose in your program. Select their settings in the same way as you set System and Options flags.

The Option flags and User flags are cleared automatically when an operator selects a new program. They are not cleared automatically from run to run of the same program—programmers are responsible for making the program initialize the flags used. Flags are not cleared when a program chains a subprogram in order to enable communication among a group of chained programs.

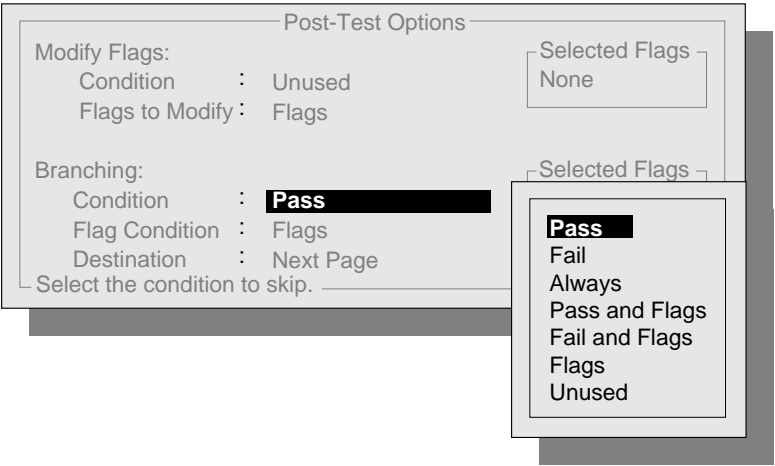
Branching

The Branching field lets you specify the conditions under which programs may branch. Branching has three fields:

- Condition
- Flag Condition
- Destination.

Depending on the condition you choose, the Flag Condition and Destination fields are either inactive or active.

To edit the Branching fields, select Condition. A window appears from which you can select a variety of conditions which trigger branching.

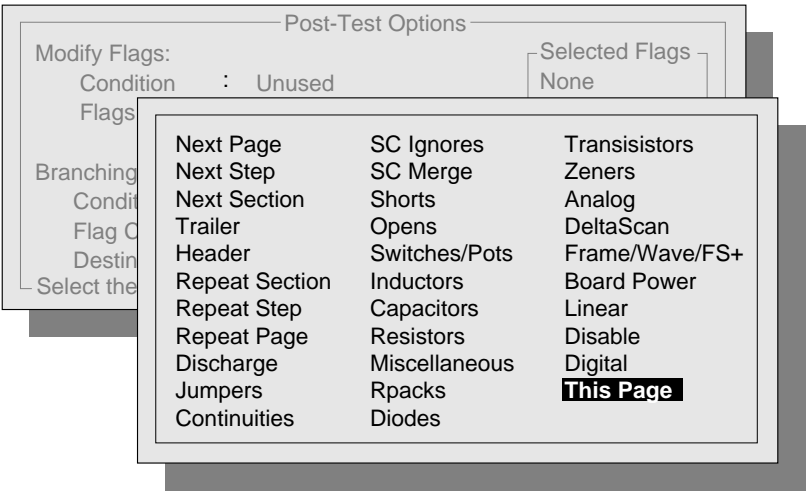


If the Condition field is set to Unused (the default), branching cannot occur. For branching to occur all the time, select Always. You can also use the result of the current test (previous if used in Pre-Test) combined with the status of the System, Option, or User flags to determine if branching will take place.

For example, if you select Pass as your branching condition, the test will branch to the specified destination only if the current test (previous if used in Pre-Test) passes. However, if you select Pass/Fail in conjunction with Flags, then for branching to occur, the Pass/Fail status must match that of the current (previous if used in Pre-Test) test, and the current value of the flags (System, Option, and User) must match those specified.

When you select a Condition that is a product of both Pass/Fail and Flags, the Test Flags field becomes available allowing you to define the flag states required for branching to take place.

When you select a branching condition other than Unused, the Destination field becomes available. When you select the Destination field, the following window appears.



The Destinations listed allow you to instruct the test program where to continue execution if the branching condition is met.

This Page executes the current test page based on pass/fail flag status. This feature is useful in ECO situations when a user flag is set for a particular ECO revision, and you want to execute a particular page only if the ECO is implemented on the board under test.

To execute This Page, set the Destination to This Page and set the Condition to the flag which indicates that the ECO is implemented.

See the section, Setting Up a Branching Condition, later in this chapter.

Other Test Step Flow Controls

Test step option in the Step Worksheet provide two additional tools for controlling test flow: Controls/Test Page Execution and Pre-Test Option Variables/Repeat Page and related functions: Repeat Delay, Repeat Mode, Cycle Vacuum, and Repeat Count.

Test Step Execution

When you select Cntrl from the Step Worksheet Options, the Test Step Controls window appears. Click the Test Page Execution Enable/Disable field to bring up a pop-up window in which you can toggle between the two choices.

Repeat Functions

Repeat Page places the test page in a repeat loop. The function is useful, for example, when the operator needs to adjust a potentiometer. You can modify the behavior of the looping with the other Repeat variables.

The other Repeat variables are enabled only when Repeat Page is On.

To use these functions

- 1 Select Option Variables from Pre-Test Options.

The Pre-Test Options window appears.

- 2 Click Repeat Page to bring up a pop-up that enables you to toggle between On and Off.
- 3 Select On to enable the other variables.

When Repeat Page is On, the test step remains in a continuous loop until you press Start.

- 4 Click Repeat Delay to specify the length of time in ms between loops.
- 5 Click Repeat Mode to specify whether you want the loop to repeat continuously or only while failing.

A pop-up box appears for you to choose “Continuously” or “While Failing.”

- 6 Click Cycle Vacuum (while repeating) to bring up a pop-up that enables you to toggle between On and Off.
 - 7 Click Repeat Count to specify the number of times to repeat the loop.
- Repeat Count is disabled if you enter 0.

Using Test Control Buttons

Programmers use Test Control buttons at the bottom of the Step Worksheet to configure test execution. Operators may also use the Test Control buttons, depending upon permissions levels set by the supervisor. (See Setting Logins and Permissions in Chapter 1.)

Using Test Control buttons you can

- Run a test, a test section, or a test step by pressing Start.

- Cancel a test run by pressing Cancel.
- Stop a test at the point of failure with SOF (stop on fail).
- Pause test execution temporarily with HLD. Combined with Start, Hold allows one to “single step” through a test.
- Repeat a test run with RPT until RPT is turned off.
- Show screen messages programmed for operators during a run with CRT.

The illustration below shows the Test Control buttons as they appear in a Step Worksheet, where you control the execution of a single test step.

The screenshot shows a window titled "18xx Windows mode-init_com" with a menu bar (Edit, Options, Tools, Save, Revert, Validate, Quit). The window is divided into several sections:

- Component Properties:** Fields for +ID, Name, Desc, Device Type, and Number of Pins.
- Test Properties:**
 - Options:** Pre, Post, Cntrl (radio buttons).
 - Indicators:** A text input field.
 - Current Page:** Page 1 of 1.
 - Test Type:** Resistor.
 - Extended:** Off.
 - Test Data:**
 - Value: 100, Scale: Kohms, High: 105, Low: 95.
 - Stim Val (mv):
 - Stimulus: Pin 2 (64), Measure: Pin 1 (68), Guard: 65, 80, Wait (ms): 0, Squelch (ms): 0.
 - Controls:**
 - Wire Mode: 4, Precise: Off, Highguard: Off, Averaging: 1.
 - Fast Mode: Off, Guard Mode: Active.
- Buttons:** A row of buttons labeled START, CANCEL, SOF, HLD, RPT, and CRT.
- Footer:** C:\TPD\DEMO-ICT.

Below the buttons, a row of function keys is shown: F3, F4, F5, F6, F7, and F8. Lines connect the buttons to their corresponding function keys: START to F3, CANCEL to F4, SOF to F5, HLD to F6, RPT to F7, and CRT to F8.

Running Tests—Start and Run. You can run a test step (the Test Properties portion of the Step Worksheet) by selecting Start.

- Click the green Start button in the Step Worksheet
- Press the F3 function key
- Press the green console Start key

To run subsequent Test Properties pages, select the down scroll arrow in the upper right area of Test Properties to move to the next page and select Start again. You can run only one Test Properties page at a time.

To run a complete test section—for example, all the resistors on a board—select the following sequence of menus from the Main menu: Edit, Passive, and Resistors. Next, click the Start button, the console Start key, or the F3 function key.

Section Run mode is a valuable test and debug capability.

To run a complete board test, enter the Main menu, select the desired board directory, and select Run, also from the Main menu bar.

Stopping Tests with Cancel. You can cancel a running test step (the Test Properties portion of the Step Worksheet), a complete test section, a Repeat command, or a complete board test by selecting Cancel. Generally, you will use Cancel to stop a complete board test or an entire section run.

To select Cancel on a running test step, section, or entire test, click the red Cancel button located in the Test Control section of the screen, or press the F4 function key or the red console Cancel button.

Cancel removes all power; relays, if engaged, are cleared. Program flow reverts directly to the trailer. Standard Trailers test the Cancel flag and report a cancel message.

Stopping a Test on Failure. You can stop a running test at the point of failure during debugging or production testing with the SOF Step Worksheet button or the F5 function key.

Use SOF to identify intermittent failures with Repeat (RPT), or to view all failing results in multiple burst Gray code tests. If you are running an entire section or program, SOF provides direct access to Test Properties when a component fails.

Stepping Through a Test. You can “single-step” a digital Test Properties page with multiple bursts by selecting the HLD button or pressing the F6 function key. Press Start to see the results of the next burst. Data from each burst often flashes across the screen too quickly to view. Holding a test at a desired point allows you to review the results. HLD works for both analog and digital tests.

Repeating a Test. Select the RPT button or the F7 function key to repeat a test endlessly. Press Repeat again to end the cycle, or press Start to advance to the next measurement, which will then also repeat. You may use RPT with SOF to identify intermittent test failures.

Viewing Test Runs. To view the execution sequence of a test run on the screen, select the CRT button or the F8 function key.

In the following example, the user has selected Run from the Main menu without the CRT button on. The output window responds with the text in the left column.

•	PRESS START TO TEST THE BOARD.	—	HEADER TEST STEPS BEGIN
•		—	HEADER TEST STEPS END
•	Testing in progress	—	RESISTOR SECTION BEGINS
•	R1 Resistor R1 1 K Ohm 2000% NODE PIN: 2	—	RESISTOR SECTIONS ENDS
•	Test Terminated, Analog problem!	—	TRAILER TEST STEPS BEGIN
•	Test COMPLETED - FAILED!	—	TRAILER TEST STEPS END

However, after having selected the CRT button, the output window responds with the text in the left column below. (Test steps are underlined for reference.)

●	PRGMVARS	HEADER TEST STEPS BEGIN
	START	
	PRESS START TO TEST THE BOARD.	
	<u>PROGRESS</u>	
	Testing in progress...	
	<u>Begin</u>	
	clr_flg	
●		HEADER TEST STEPS END
●	R1 Resistor R1 1 K Ohm	RESISTOR SECTION BEGINS

	R1 Resistor R1 1 K Ohm	
	2000%	
	NODE PIN: 2	
●		RESISTOR SECTIONS ENDS
●	<u>Dischrg</u>	TRAILER TEST STEPS BEGIN
	<u>Into</u>	
	<u>Analog</u>	
	Test Terminated, Analog problem!	
	Test COMPLETED - FAILED!	
	<u>Cancel</u>	
	<u>Valid</u>	
●		TRAILER TEST STEPS END

Using the Function Keys

System software function keys allow you to select many of the software's test execution and debug features discussed in the previous section. Refer to the following table for a summary of the function keys.

Key	Function
F1	Quick Help
Shift-F1	On-screen function key template. (Type any key to remove.)
F2	Locator. Does not apply in the Disable Test Properties or to digital guards section of Gray code Test Properties. (Node field double-click with the right mouse button.)
F3	Start. Same as tabletop Start key, footswitch, and screen Test Control Start button.
F4	Cancel. Same effect as tabletop Cancel key and screen Test Control Cancel button.
F5	SOF. Stop On Fail. (Run and Start modes.) Same as screen Test Control SOF button.
F6	. Pause after every measurement. Press Start to resume. Same as Test Control HLD.
F7	Repeat current measurement. Press Start to continue. Same as Test Control RPT.
F8	CRT. Display test step messages for operators during a run.
F9	Probe. Enable fixture probing during Test Properties node number entry. (Node field double click with the left mouse button.)
F10	Return to menu control from a field or return to previous menu.
F11	DOS command line. Pressing F11 executes the DOS command specified in Setup/Environment.
F12	Escape to online documentation—DOS version.

To help you identify and locate the function keys, press Shift-F1. The function key template shown in the illustration below will appear at the bottom of the screen.

You will not be able to use the software system when the function key template is visible. Press any key to remove the template from the screen and to return to software control.

Keyboard Template											
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
Help	Loc	Start	Cancel	Stop/Fail	Hold	Repeat	Crt	Node	Menu	Notes	Doc

On a standard keyboard the function keys are on the left. On an extended keyboard the keys are on the top row.

Using Option Flags Switches to Control Flow

The seven option switches that appear near the bottom of a screen in Run mode also allow you to control program execution. Use messages to prompt the operator to select an option if certain conditions exist. The operator may either select an option button with the mouse or press the keyboard key corresponding to the option number.

To set up the option flags

- 1 Choose the test step
- 2 Click Pre- or Post-Test Options **Flow Control** (or I/O Control).
- 3 Type the output message in the Comments field.
- 4 Select Modify Flags **Condition** and then click the condition (Pass, Fail, or Always) to trigger the message.
- 5 Click Modify Flags **Select Flags**.
The flags selection window appears.
- 6 Click the **Option** flag and select Set (Tog or Clr) from the pop-up window.
- 7 Click out of the Options windows and **Save** your edits.

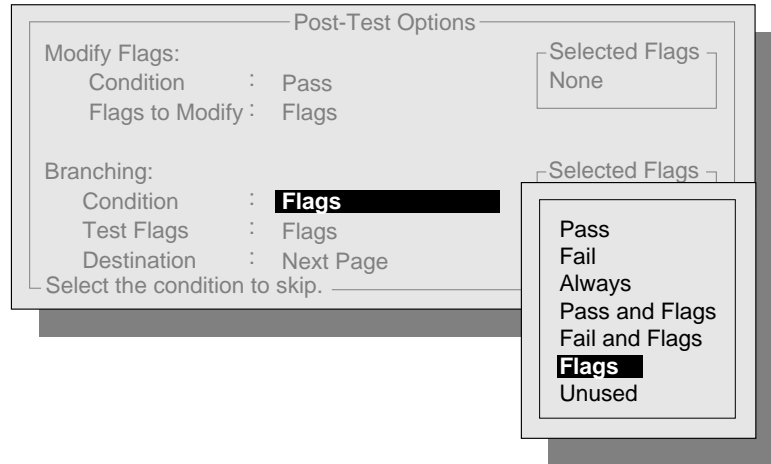
Setting Up a Branching Condition

When you develop your test program, you may want to change the flow depending on certain conditions. For example, if you are testing two revisions of a board with the same program and fixture, then you will need to test for flags and conditionally “branch around” the other revision.

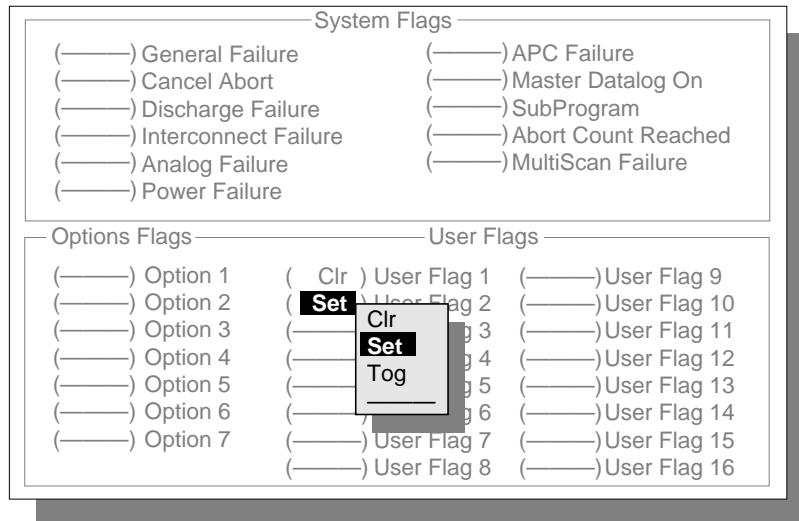
Thus, if revision A has a 2.2 Kohm resistor, and revision B, a 10 Kohm resistor, you can program with User Flag 1 and 2 respectively as follows:

- 1 Enter data as appropriate in the Component Properties portion of the Step Worksheet.

- 2 In the Value field of Component Properties, enter the revision A value—2.2 Kohms (or enter whatever value you are going to test on page 1).
- 3 Select Tools/Generate Test from the menu bar to create page 1 of Test Properties in the lower portion of the Step Worksheet.
- 4 In the Pre-Test Flow Control window in Test Properties, click the Branching Condition field and select Flags from the pop-up window.



- 5 Click the Flag Condition field and select Clr for User Flag 1 and Set for User Flag 2.



The Destination field should be Next Page.

- 6 Click the **Test Properties** page twice to remove the Flow Control options windows.
- 7 Select Tools **Copy Page** to create a second Test Properties page.
- 8 In the Value field, enter the revision B value—10 Kohms.
Use Tools/Tolerance Calculator to get the correct High and Low limit values on Test Properties page 2.

- 9 In the Pre-Test Flow Control window, click the **Branching Condition** field and select **Flags** from the pop-up window.
- 10 Click the **Test Flags** field and select **Set** for User Flag 1 and **Clr** for User Flag 2.
- 11 Click **Destination** field and select **Next Page**.
- 12 Click the **Test Properties** page twice to remove the Options window; then select **Save**.

Setting Up Chaining

To set up chaining

- 1 Create new main and subprograms using Files **New**.
- 2 Make the main program the current program.
- 3 Select Edit, then **Header**.
- 4 Move the clr_flg test step from the last place in the Header to the second step, immediately following PRGMVARS.
- 5 Add message steps telling the operator which number (or options switches) to select between the clr_flg and Start test steps.

Message steps look similar to the following.

The screenshot shows a 'Header' window with the following fields and values:

- Step Name: Menu 1
- Execute this Step On Condition Flags: [Selected Flags: None]
- Conditional Execution: [Modify Flags: Flags, Selected Flags: None]
- Output Device: On
- OutputString: \fFOR BOARD TYPE:\t\PRESS NUMBER:\n\n743A\t\
- Input Type: None
- Variable: 0
- Chain Board Name:
- External Programming: Off
- Vacuum Control: Keep

- 6 Add test steps for revisions or various chaining programs at the end of the Header after the Begin test step.

This step is necessary to allow the vacuum to be energized before chaining to a subprogram. You need one test step for each subprogram you wish to chain to. Each “evaluation” test step should conditionally execute based on the appropriate setting of Option switches for a given board revision or subprogram.

Such a test step may look similar to the following.

Header [↑]-[↓]

Step Name: 743B
Execute this Step On Condition Flags

Selected Flags
Opt

Conditional Execution

Modify Flags: Flags

Selected Flags
Opt Usr

Output Device: Off
OutputString:

Input Type: None Variable: 0

Chain Board Name: RevB

External Programming: Off

Vacuum Control: Keep

- 7 In the Modify Flags field, clear the Option switch and set the corresponding or related User Flag.

This step allows the Option switches to be utilized for another menu selection or another purpose.

- 8 Chain to the appropriate subprogram by entering the name of the subprogram in the Chain Board Name field.

If the main program is to be used to execute the test for one of the revisions, then don't chain on that menu selection; just set flags. In addition, you will need to add a test step, or steps, to check for the flag settings for the other revisions. This "FlgChk" test step must be the first step in Discharge prior to executing any actual tests on the board revision tested in the main program.

The following procedure ensures that the main program is not executed for a revision to be tested in a subprogram.

- 1 Add a **FlgChk** test step to the Discharge section.
- 2 In Pre-Test/Flow Control set the Condition field to **Flags**.
- 3 In the Flags field, select the appropriate **User Flag(s)** as condition.
- 4 Click the Destination field and select Trailer from the pop-up window.
- 5 Repeat steps 2, 3, and 4 in the Post-Test/Flow Control for a second revision to be tested in the subprogram.
- 6 Repeat steps 2–5 in additional test steps or additional Test Properties pages until all subprogram flag settings are handled.

▼▼▼

5 TEST TECHNIQUES AND STRATEGIES

Chapter 5 is an overview of test techniques and the recommended strategies to apply them.

Techniques are the hardware and software methods of testing analog and digital components and topological board features.

Strategies help identify the entire application and prepare you for the programming that lies ahead so that you can meet or exceed test goals.

Knowing the tester's techniques will disclose which of your board's components are testable and which are not. Having a sound strategy also saves application development time and allows the tester to perform to its maximum capability.

The **Z1800-Series Component Test Reference** provides the test specifications and techniques for all testable component types.

Introduction to Test Applications

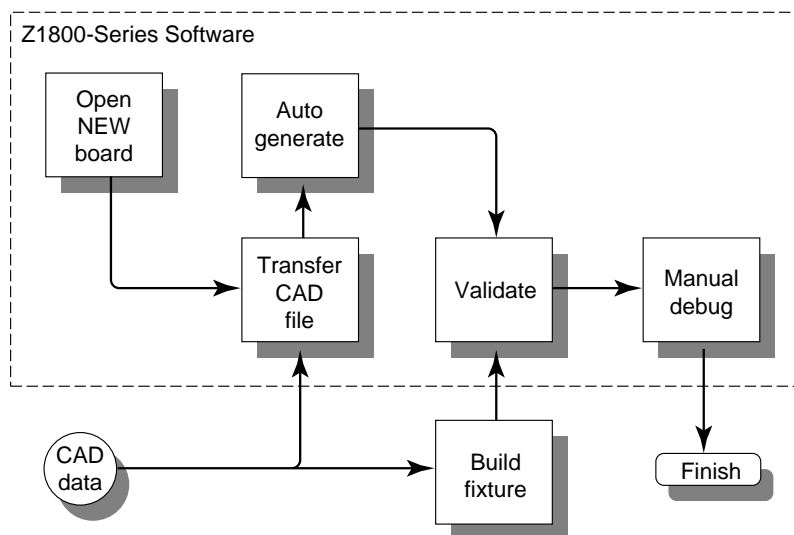
A test application is a test program and fixture for a board type that together produce accurate, repeatable results.

Fixture and program development sequence is dependent upon the type of information you have about the board. The information will be used to create an input list, which is a structured list of the components on the board, required by the program generator to create a program.

If you have a CAD-generated component list with wiring data, you will follow one path. If you have no CAD data, you must follow another.

With CAD data you can develop the fixture and program in parallel until the validation phase, when you will need the fixture to run Validate, an important debugging tool.

Refer to the following flow diagram.



Without CAD data—that is, with a manual entry procedure—it is best to build the fixture first so it can be used to help generate the input list one component at a time. You can use either of the following methods.

Method 1

- 1 Build the fixture. (See the Fixturing Guidebook.)
- 2 Create a program by selecting New from the Files menu.
The software system will create a skeleton program with Header and Trailer.
- 3 Learn the node range from the shorting plate with Learn.
- 4 Learn continuities from the bare board with Learn.
- 5 Learn special cases to ignore from a loaded board with Learn.
- 6 Enter components using the Edit function.
- 7 Use Nodefinder to obtain node assignments.
- 8 Update the database with Update.
- 9 Create analog guards and stabilize test with Validate.
- 10 Debug the test with the Step Worksheet editor.

See the **Z1800-Series Board Test Tutorial** for more information on steps 2–10.

Method 2

- 1 Manually create an IPL.DAT using a text editor.
- 2 Select Build in the Pgen menu.
- 3 Select Generate in the Pgen menu.

Developing a Program Strategy

A quality test program will tell you whether or not all devices on a board are manufactured properly. The tester should pass good components and fail those improperly manufactured, with repeatable results. You can achieve quality program standards by understanding the tester's capabilities, evaluating the test, and auditing the program.

Understand the Tester's Capabilities

Review the tester's specifications with the following questions in mind:

- Are all the components on the board testable?
- If not, what devices must be omitted from the test program?

The Programmer Efficiency Package (PEP) helps you analyze the testability of the components on your board.

PEP reads a Z1800 input list and checks each entry against a list of criteria to ensure that a test can be properly generated for that entry. It uses the SSI/MSI and the VLSI libraries and various tester specifications to make these checks. The goal of PEP is to identify problems and recommend solutions early in the test process when they are easier to identify and address.

Besides finding problems, PEP will provide additional information that will be useful during the program generation and debug phases of test program development. PEP may operate on a completed or a partial input list, and should be run as early in the process as is possible.

The primary output of PEP is a report that lists the findings of the program. From this report, you can make changes to the input list or to the libraries, and act on PEP's other recommendations. In addition, you have the option of having PEP automatically update the input list with its recommended changes, saving you the effort of doing so manually.

Momentum

Momentum, an option, is a comprehensive toolset that accepts data from a variety of sources, such as CAD data, Bill of Materials, or netlists. It uses the data to create a high turn-on rate program and generate the complete fixture design. Momentum gives you the ability to move through the complete program and fixture development process—from CAD data to testability analysis to Z1800-Series program generation and fixture design—in a single, integrated environment.

Momentum uses FABmaster as integration software to translate and input CAD data. FABmaster supports a wide range of CAD/CAE file formats and includes a user-configurable input processor for accommodating custom formats.

If you have already developed standard processes for generating Z1800-Series netlist (IPL.DAT) files you can use Momentum's testability-analysis and program-generation tools by importing the IPL.DAT file directly into Momentum.

Momentum analyzes all analog tests, taking account of the capabilities of the target Z1800-Series tester configuration. Momentum recommends multiwire tests or adjusting test limits, where required, to ensure test transportability. Momentum also analyzes the circuit configuration to determine the high-priority guard points for evaluation by the Validate automatic debug tools on the tester.

For packaged devices such as VLSI, Momentum checks for testability using MultiScan vectorless test, model availability (VLSI, template or user libraries), and logic drive levels required. Momentum then generates the final program for debug on the target tester.

The information generated during the analysis stage is fed back into the integrated FABmaster software for fixture design. This step includes automatic probe placement analysis and selection, followed by wire list and drill file generation. All of the recommendations approved by the user for multiwire testing, MultiScan testing (including FrameScan or WaveScan inducer placement), multi-panel testing, and power supply wiring are automatically included in the fixture design specification.

MultiScan

The MultiScan package includes five effective power-off test techniques: DeltaScan, WaveScan, FrameScan, FrameScan Plus, and CapScan. Each provides vectorless testing. The **Multiscan User's Guide** devotes a separate chapter to each technique.

DeltaScan detects opens on digital devices for Teradyne's popular Z1800-series board testers. It also detects broken protection diodes and bond-wires, and may detect "blown" input or output transistors or incorrectly oriented devices. DeltaScan can be employed without knowledge of the internal functioning of the device-under-test (DUT); however, it is essential that the DUT has protection diodes and a substrate resistance to the ground or power pins to identify opens.

WaveScan detects opens on digital device leads. It also detects broken bond-wires, and may detect blown input or output transistors or incorrectly oriented devices. It uses magnetic inducers mounted over the DUT; an oscillating magnetic field induces voltages in conducting paths in the DUT, verifying proper connection.

FrameScan detects opens on device leads. It also finds opens on connector pins without making physical contact with the connector. FrameScan uses electrostatic inducers mounted over the DUT to induce voltages in conducting paths; the voltage levels verify proper or improper connection. FrameScan Plus detects opens on device leads. It also finds opens on connector pins without making physical contact with the connector.

FrameScan Plus uses sensors mounted over the DUT to detect capacitance in conducting paths; the capacitance levels verify proper or improper connection.

CapScan detects incorrect orientation of polarized capacitors. CapScan uses sensors mounted over the capacitor to compare capacitance in each current direction, verifying the device orientation.

The five MultiScan tools provide complementary approaches to testing, yielding power-off, vectorless test coverage for a wide variety of devices.

Prism-Z

The PRISM-Z (PRecision Integrated Signal Measurement) module is a next generation analog measurement subsystem for in-circuit testers. It is a Digital Signal Processor-based (DSP) design that can accurately measure values and circuit configuration found on today’s products. The PRISM-Z instrument is standard with the Z1888, available as an option for the Z1803 Plus, and is also supported as an upgrade for existing 1805-1, 1808, 1840, 1860, 1880-1, 1880-2, and 1890 testers.

The PRISM-Z offers benefits in the following five major areas over existing systems:

- Throughput
- Accuracy/Repeatability
- Stability, both long and short-term
- Broadened range
- Flexibility/Potential

Systems which include the PRISM-Z option will retain the ATB to guarantee backward compatibility with existing programs as well as for ordinary tasks such as shorts, continuities, and NodeFinder. The software allows you to mix and match PRISM-Z and ATB tests throughout a test program to give the most flexibility.

For detailed information about installation and operation of the PRISM-Z board, see the **Z1800-Series PRISM-Z Reference**. Chapters 2 and 3 in this document discuss the software setup.

Understanding the Board

You can develop a sound test strategy if you understand the board. The board’s physical layout is critical to the fixturing process. Make sure you have the latest parts lists and schematics to determine the prospective devices to test. Data books will tell you how devices function.

The following table describes the primary test techniques used to test digital and analog devices.

Program Section	Purpose	Some Techniques
Digital	To detect pin faults on boards with digital components. Determines wrong or misoriented components, broken traces, or solder opens.	APC, Gray code, CRC, Vectors, Count, High, Nodefinder, Tracer, mixed mode.
Analog	To detect analog component manufacturing faults, continuities, shorts, passives, diodes, switches, jumpers, inductors, capacitors, resistors, potentiometers, Rpacks, zeners, transistors, power-off analog.	APC, Nodefinder, shorts; continuities, shorthand test types: resistor, capacitor, inductor, diode, transistor. Longhand test types TVSI, TVSV, TISV, TV

Keep a record of the testable devices and the technique required to test them. Your in-process program supplies most of the information. Update the record as you progress to the implementation phase. Your record's historical value also assists in subsequent development.

In the table below, the potential faults—component, device, or board feature—are listed in the left column. Include the percent of the total fault spectrum represented by this group in the next column. Percentages can be rounded off. The test technique, or strategy, is listed last.

Accuracy depends upon the board's complexity and your strategy.

Potential Fault	(Number and % of Total)	Technique/Strategy
Capacitors	(22, 12% of 179)	shorthand analog
Inductors	(8, 5%)	shorthand analog
Resistors	(40, 22%)	shorthand analog
Transistors	(2, 1%)	shorthand analog
Digital ICs	(1, .05%)	Gray code or vectors
Shorts	(50, 28%)	shorthand
Continuities	(50, 28%)	shorthand

Evaluating the Test

There are several key factors you must consider when evaluating the effectiveness of a test strategy:

- Coverage is usually a ratio of the number tested components versus the total number of components on the board, expressed as a percentage. Percentage is affected by component values and tolerances, and circuit configuration. Coverage is also the degree to which failures can be diagnosed.
- Throughput is test speed. A strategy for achieving throughput goals is easier to establish before programming begins than after. Manufacturing goals will determine if speed is more important than coverage.
- Repeatability means that a test program repeats results for a given board on different testers of the same family.
- Transportability is a program's ability to be used on testers of a different family.

The ideal strategy should provide a balance of coverage, transportability, throughput, and repeatability.

For example, if you are in a very high volume, multisite test environment, throughput and transportability may be more important than coverage. Your strategy might then contain the following rules:

- limit digital tests to 100 kHz (clock 5)
- expand analog tolerances
- delete or minimize use of 6-wire tests
- limit use and scope of diagnostic messages

If, on the other hand, you are in a low-volume, high-coverage test environment, coverage and repeatability may be more important than throughput and transportability. Consequently, your strategy might contain the following rules:

- use 1 MHz clock rates
- use tight analog tolerances
- use 6-wire tests for resistors valued under 100 Ohms
- use plentiful ground lines in the fixture
- use Precise mode for resistors and capacitors
- use descriptive diagnostic messages

Auditing the Program

You can exercise quality control on a program in many ways. The important thing is to establish an auditing procedure, both during development and after a board has been in production.

The purpose of a periodic quality audit is to ensure that the program is actually detecting faults. An easy way to spot check performance is to inject faults into the application by using the Fault Inject feature or temporarily altering the program or the fixture.

For detailed information about Fault Inject see chapter 9, “Test and Debug Tools.”

You should also maintain a program format and content specification that includes all quality control measures, a few of which are mentioned below.

- Document the tester’s operating bounds and capabilities. Set realistic tolerances accordingly, neither too tight nor too loose.
- Calibrate the tester periodically using the standard calibration procedure.
- Run system diagnostics at the beginning of each production shift.
- Check each program periodically with a known-good board.

Power Control and the Effects on Board Test

Several mechanisms control the application and removal of power on the board. These mechanisms include the power supply control board, Start and Cancel switches on the tester, software power controls built into the test program itself and into debug tools. This section explains the operation and effects of these mechanisms.

Power Supply System Architecture

Z1800-series testers equipped with the Programmable Power Supply Package have the Power Supply Control Board PN 051-002-xx, which allows program voltage control of the 55-volt and optional 3-volt power supplies. Earlier models of Z1800-series testers that have not been retrofitted have the Power Supply Control Board PN 045-047-xx, which requires manual adjustment of the current limit and voltage control on the DUT 0–55 volt power supplies. See chapter 2 in both the **Z1800-Series Maintenance Reference** and in your tester-specific system reference.

In addition to supplying more functionality, the Power Supply Control Board PN 051-002-xx fully emulates the earlier controller PN 045-047-xx, thus allowing D.X programs to run without any modifications with E-Level, or later, version system software. In addition to the existing power Step Worksheets, the E-Level and later software has been extended by adding a set of new power Step Worksheets to enable features of the Power Supply Control Board PN 051-002-xx.

IMPORTANT: A program can contain only “old” or “new” power Test Types. Should you write your test program to alternate between old and new Test Types, an error message will appear.

After you load a test program, the software determines which controller board is present in the system. If the programmable power Test Types are used, the Power Supply Control Board PN 051-002-xx is required. If the other power Test Types, non-programmable, are used, either control board is valid. Once the Test Type and power supply control board are validated, the system is locked into either the programmable or adjustable mode for the rest of the execution of the program. The first power Test Properties page of a Power section determines the mode of operation for the rest of the program—either adjustable or programmable.

Power-Up/Power-Down Sequence

The software has basically two mechanisms to control power to the board-under-test—the power-up and power-down sequences. No matter which power supply controller board you have, a program always starts out in a powered-down state. The power-up state is always reached by running the power section, either in an explicit or implied manner.

Power-Up. An explicit power-up occurs in normal run mode when the program reaches the power section and begins executing its steps, or when you directly execute a power section. An implied power-up occurs when, in the edit mode, you press Start on a test step which needs power to execute, and power has not been turned on. The first time this happens, you are queried if you really want to power on or not. An implied power-up also happens whenever power has been downed by either Digital Tracer, Nodefinder, Cancel or any other means of turning power off and a step requiring power is executed again.

Power-Down. There is no explicit place for power down. It is implied in the following circumstances:

- before the first test step of the Trailer section,
- when a powerless action such as Digital Tracer or Nodefinder starts up, or when a powerless test step is executed
- when you invoke the text editor (F11 Key) or exit the Component Properties editor, or
- when you press Cancel

There are two ways of powering down: orderly and express.

An orderly power down occurs at the appropriate places during normal test execution. It turns off the power supplies either in the reverse order of their on sequence (default) or in the “forward” mode. In Forward mode, the supplies are turned off in the same order in which they were turned on. To change the default operation, select Forward from the Power Down Sequence field in PRGMVARS/General Variables

Express mode shuts down all power supplies at once by sending a reset to the power supply control board and clearing the ATB. The power-off action is immediate. Express mode is the default action that takes place when Cancel is selected.

You can program the Cancel action in the Setup/Environment Variables menu to be orderly or express.

Power-Down/Discharge. The power discharge sequence immediately follows any power down sequence. The purpose of the discharge sequence is to remove charges from capacitors on the board. Using the Discharge Method control variable in PRGMVARS you can specify how you want the sequence to take place in a given program.

See chapter 3, “Editing Overview,” for detailed information about PRGMVARS.

There are two ways the power down discharge can be run with the addition of a third option which suppresses the discharge phase.

The **Power Nodes Discharge** method (PwrNodes) is the default mode for programs prior to E.2 level software. It emulates the power discharge method used in D.X level software and runs discharge on an internal list of nodes. These nodes are collected from the power steps of the “old style” Power type in the Power section. A maximum of 24 pairs of nodes can be collected.

IMPORTANT: A drawback to using this method is that it will not discharge capacitors not connected directly to the power supplies. Method 1 cannot be used with programs using programmable power supply test types.

The **Discharge Section Rerun** method (DschrgSect) is the default mode for programs developed using E.2 or later level software. It reruns the program discharge section and executes all Test Properties pages with the Discharge test type including the Pre- and Post-Test options. Failure reporting is turned off; however, the section fail flag is set, should one or more steps fail to discharge. “Section Discharge in Progress” messages may appear during power down operations even though there may be not entries in the Discharge section. Pre- and Post-Test options can be used to control relays and/or IEEE devices as needed.

IMPORTANT: Method 2 is the preferred option since a complete discharge takes place.

None, the third method in the Discharge Method pop-up box, suppresses the discharge phase of power-down.

IMPORTANT: It is normally not advisable to turn off power down discharge because doing so can damage the tester, fixture, and the board under test.

Power-Down/Restore Process. A power down/restore is an automated process. When power is on, and a test action which needs power turned off is about to be executed, a power-down/discharge sequence is executed first. Likewise, if a test action or a section boundary is executed which requires power to be on, a power-up/restore sequence is executed first.

IMPORTANT: Power-up/restore brings the power and relays back to the state that existed at the end of the Power section. If the program is subsequently modified in the Pre- or Post-Test of a digital step, the power/relays state changes outside of the Power section are lost and will not be restored.

The following table shows the section and functions which are keyed to Power Off and Power On.

Keyed to Power Off	Keyed to Power On
Header	All test types in Digital section:
Trailer	Vector
Discharge	Gray code
Continuities	Mixed
Ignores	Digital section boundaries
Merge SC	Linear section
Shorts	
Opens	
Switches/Potentiometers	
Inductors	
Capacitors	
Resistors	
Miscellaneous	
Rpacks	
Diodes	
Transistors	
Zeners	
Analog	
DeltaScan	
FrameScan	
WaveScan	
Nodefinder	
Tracer	
F11 (text editor)	

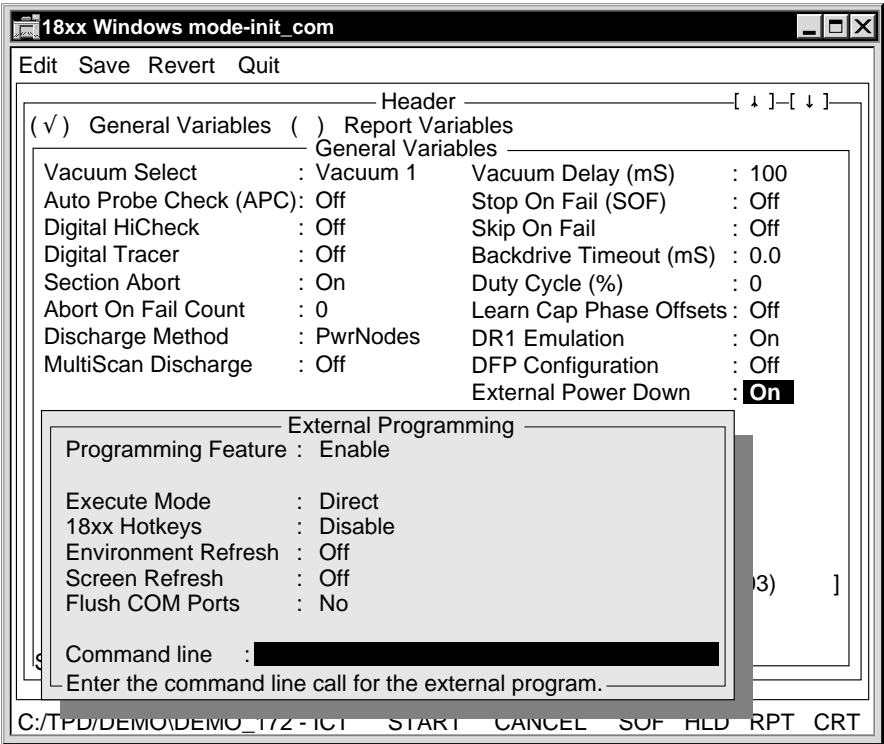
Power Supply Management

Z1800-Series Power Supplies. In earlier revisions of system software, the Adjustable Power Supplies were controlled under the Test Type, Power. In the Test Properties portion of the Step Worksheet for Test Type Power, a subtest identified which power supply was being controlled and in what power supply mode. In E-Level and later system software, the formerly Adjustable Power Supplies may be either in the adjustable mode or in the programmable mode. The programmable mode requires that the tester be upgraded with the Power Supply Controller Board PN 051-002-xx. The following chart shows the relationships between controller board PN 045-047-xx and controller board PN 051-002-xx and software functionality.

Controller Board	Power Supplies	Test Types	D.X Programs
PN 045-047-xx	Adjustable only	Power	No conversion to E.0 or later
PN 051-002-xx	Programmable and Adjustable	+5 V Fixed +0–5.5 V programmable A +0–55 V programmable B +0–55 V programmable ± 0–55 V slaved programmable	Must manually change D.X Power Step Worksheet for programmable functionality

External Power Supplies. Z18xx system software permits the powering down of external power supplies in an orderly fashion through the External Power Down control in the Header/PRGMVARS.

Selecting this variable lets you add an external program name to be called during the 18xx power down function. When an external program is called, external power supplies associated with the external program will be powered down as part of the 18xx power down procedure.



The data entry is just like the external program entry for Pre-/Post-Test Options, Header/Trailer, and External Program worksheets.

Some common places that the system software calls the power down function are

- Upon entry of the Trailer section during program Run
- Whenever you press Cancel, and the board has been powered up
- When you enter Nodefinder or Tracer

The power down execute sequence is

- All 18xx internal supplies are turned off in the order specified by the PRGMVARS Power Sequence variable.
- If one exists, the external power down program is called.
- The cage (DR,FIB,RAB...) is cleared.
- The discharge method selected in the Header/PRGMVARS Discharge Method field is executed.

Power and Program Execution

Generally speaking, tests are divided into power-off tests, which include analog tests, and power-on tests, which include digital and linear (analog power-on) tests. 18xx system software is designed to test all power-off devices first, followed by the power-on tests.

The standard test sequence is

- Interconnects (jumpers, continuities, shorts)
- Passive (inductors, capacitors, resistors)
- Semi (diodes, transistors, zener diodes)
- PwrOff (analog power-off devices and MultiScan tests)
- Board Power (power-on and power bus measurements)
- Linear (analog power-on devices)
- Digital (Gray code and vector)

Power Step Worksheets

Power supply Step Worksheets in the Power section of E-Level and later system software enable you to precisely control the application of the tester's power supplies to the requirements of the board-under-test. These Step Worksheets are divided into two sections—the Component Properties and Test Properties. The Test Properties portion of the Step Worksheet may have multiple pages.

With the Power Supply Control Board, PN 045-047-xx, E-Level and later system software controls the DUT +5 volt and the two 55 volt power supplies exactly the same way as D-Level system software. All in-circuit test programs developed with D-Level software will execute in the same manner with E-Level software with no modifications. The power supplies available for the board-under-test are the +5 volts, ± 12 volts, ± 15 volts, and two 0–55 volts, adjustable. A single page Test Properties controls and tests each of these power supplies.

With the Power Supply Control Board, PN 051-002-xx, the E-Level and later software has the ability to control the supplies in a programmable mode. The system software does not allow you to mix the two power supply control methods. The Test Properties for the power supplies in the programmable method are multipage. The first page programs the desired voltage, and the following page(s) measure the supplied voltage at the board-under-test. All measurement pages are the standard Test V Test Properties page.

Each physical power supply has its own Test Properties. The 0-55 V supplies have an extra (+/-) Test Properties to emulate the ± 12 V and ± 15 V operation of the power supply controller board (PN 045-047-xx). Slaved power supplies will program to the same voltage and connect out to the board-under-test at the same time. In other words, the slaved power supply Test Properties turns on the A and B power supplies at the same time with the A supply set to the positive programmable voltage and the B supply set to the negative voltage. The output of the slaved supplies is via the V_{ee} pins.

The Test Properties available are

- + 5V fixed
- + 0-5.5V programmable
- A 0-55V programmable
- B 0-55V programmable
- \pm 0-55V slaved programmable

All power supply Test Properties have the following mode selections:

- Off/On/Disable/Enable
- wait time (time after connect)
- measure Low/High threshold

Additionally, the programmable power supply Test Properties have these fields:

- Off/On/Disable/Enable/Reprogram
- Programming voltage

Mode—Off/On. Selecting On or Off from Mode in the Test Properties/Test Data portion of the Step Worksheet connects the power supply to the output pins via relays.

When the Test Properties mode is **On**, the power supply is connected to the board-under-test via the power supply isolation relay (cold switched). The power supply is programmed to the voltage in the Value field (programmable supplies only), turned on, and internally tested. The measured voltage must fall between the High and Low limits.

The sequence for turning a power supply On is as follows:

1. Connect the power supply to the board-under-test while the power supply is programmed to 0V.
2. Program the desired voltage.
3. Enable the power supply.
4. Wait.

When the test step is executed and the mode is **Off**, this power supply is programmed to 0 volts. The power bus on the board-under-test is discharged, and the power supply isolation relay is opened, disconnecting the power supply from the board-under-test. This mode also removes the power supply from the Restore All list in the Pre- and Post-Test options.

The sequence for turning a power supply Off is as follows:

1. Turn off the power supply.
2. Discharge the power supply with the internal discharge resistor.
3. Remove the power supply connections from the board-under-test.

For power supplies which are already turned on, power-on steps will not execute.

Disable/Enable. The **Disable** mode is similar to the Off mode except for the fact that the power supply may be enabled and is not removed from the Restore All list.

The **Enable** mode allows you to re-engage a power supply that has been Disabled. To enable any power supply, you must have previously turned it on.

These functions are primarily of use to temporarily shut down the power supplies without reprogramming the voltages.

Reprogram. The Reprogram selection changes the voltage for programmable power supplies without changing the state of the connection. Power supplies which have been turned off are not affected by the Reprogram option.

Measure Internally . For Power On, Enable, and Reprogram, the software makes an internal measurement of the power supplies. The voltage is measured right at the power supply output and may be slightly different from the voltage at the board-under-test. Measure Internally merely verifies the correct functioning of the power supply.

Measure Low/High. Measure Low/Measure High are the limits for the “measure internally” option. The display line indicates that an internal measurement is made. In addition, if a power supply internal voltage measurement fails, the diagnostic report indicates that failure.

Pre- and Post-Test Power Control. All steps have the ability to control the state of Power via fields in the Pre- and Post-Test Options menu (Option Variables/Power Supply). It is possible to power down and restore individual or all power supplies. However, the power-down function is possible only with an active power supply. Likewise, the restore function is possible only on a powered-down power supply.

When the All option is selected from the Power Supply pop-up in the Pre- or Post-Test Option Variables window, a regular power-down/restore process is invoked, including execution of Discharge for the power down.

If you have selected an individual power supply, only the specified power supply is modified.

Pre- and Post-Test power control can control either one power supply or all power supplies.

The Power Supplies pop-up in Pre- or Post-Test Option Variables is available when the DUT Power option is set to Restore. The Power Supply pop-up displays both the adjustable mode names and programmable names for the power supplies as shown below:

5V
12V or Prog Slaved
15V or Prog Slaved
ADJ or Prog A&B
ALL
Prog 5.5 V

Individual power supplies (5 V, 12 V or Prog Slaved, 15 V or Prog Slaved, ADJ or Prog A & B, Prog 5.5 V) and ALL power supplies are handled quite differently. When ALL power supplies are selected, a complete power-down/restore is executed just as it is when the Trailer or Power sections are run in the orderly power-up/down routine. When an individual power supply is selected, an action specific to the selected power supply is performed, and no discharge occurs.

The editor allows you to select only one power supply choice per Test Properties page. You can also specify whether you want power to be restored or down.

For further information about the Power Test Properties selections, see the **Z1800-Series Component Test Reference**.

Updating D.X Programs to Run With Programmable PS

D.X programs are compatible with the E-Level version of the system software unless you want to use programmable functionality in your D.X programs. In that case, you must update the Power Test Properties associated with those programs.

You can update Test Properties by modifying the PGEN.CFG file

PROGSUPWS Yes;

and generating each power Test Properties page individually or by changing the Test Type in Test Properties. The recommended method to ensure the maximum data conversion is to modify PGEN.CFG.

For more information about modifying the PGEN.CFG file see chapter 7, “Program Generator Tools.”

IMPORTANT: In addition to the steps already mentioned above, you need to change the Header/PRGMVARS entry for the discharge method to DschrgSect.

Analog and Digital Switching

During analog testing, the tester's driver/receiver boards connect the device under test (DUT) to the backplane's 6-wire analog bus. The ATB connects to the backplane and therefore has access to the DUT for stimulus, measurement, and guard/reference connections.

A switch matrix on the programmed driver/receiver boards connects the EFG terminals to the board-under-test and its devices under test (nodes) by way of a fixture.

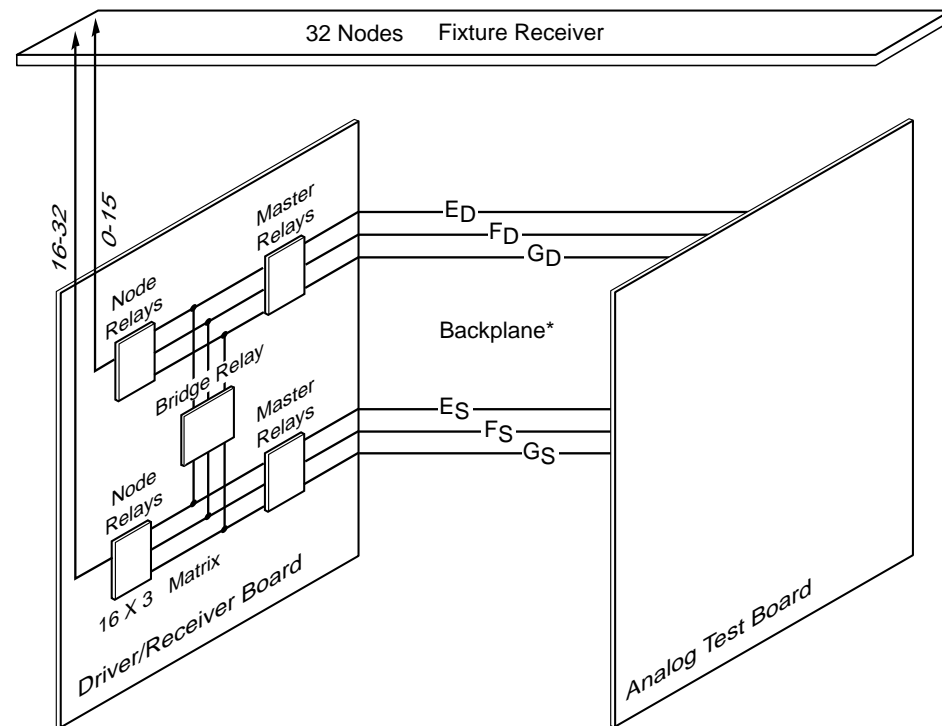
For each driver/receiver board, the switch matrix comprises two separate 16-channel by 3-pole arrays: drive and sense. Bridge relays connect nodes from the drive side to the sense side. In analog tests, the bridge relays connect the drive/sense pair at the driver board for better 3-wire accuracy. The master relays connect the poles to the backplane and are closed only when the poles are in use.

Each Z1800-series tester offers a specific number of independent channels.

Tester	No. of Channels
Z1800	640
Z1803-1	1024
Z1803-2	2048
Z1803 Plus-1	1024
Z1803 Plus-2	2048
Z1805-1	1024
Z1805-2	2048
Z1808-1	1024
Z1808-2	2048
Z1820	2048
Z1840	640
Z1850	1024
Z1860	2048
Z1860-NB	2048
Z1866	2048
Z1880-1	1024
Z1880-2	2048
Z1884	5119
Z1888-1	1024
Z1888-2	2048
Z1890	2048

Because every tester pin has access to all three analog poles, there are no multiplexer conflicts. A simplified ATB-to-driver/receiver illustration appears below.

Analog switching configuration

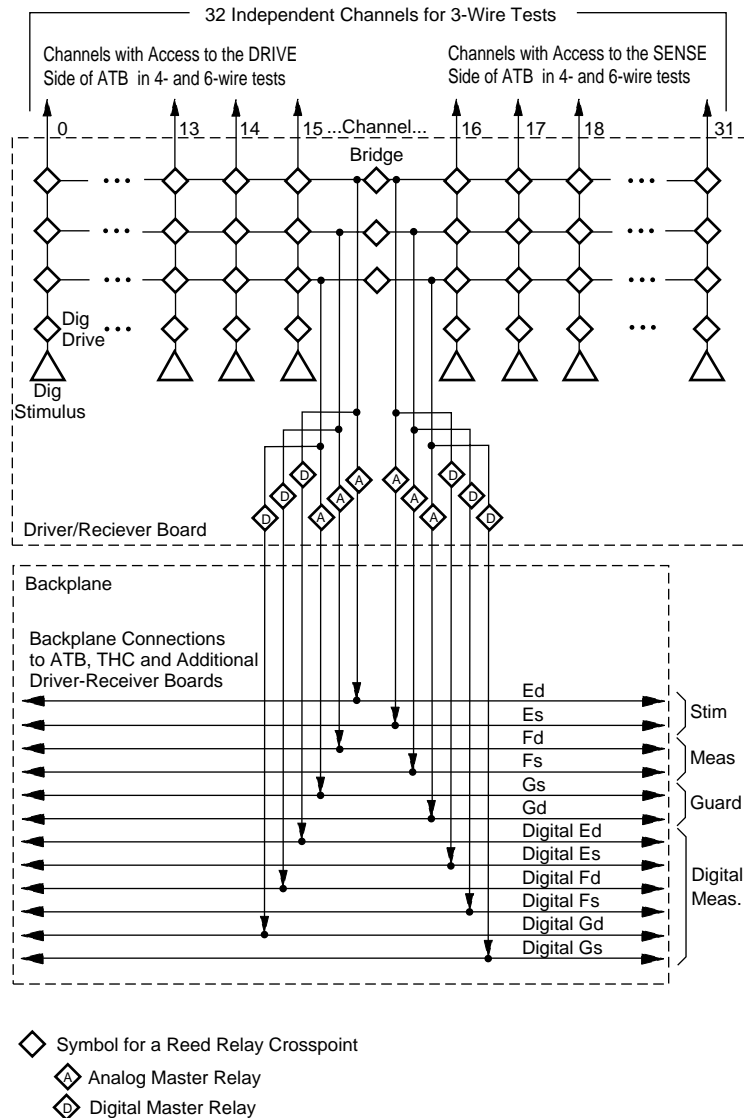


*Note: Digital E, F, and G lines in the backplane are not shown

The master relays determine whether the DUT connects to the analog or digital backplane buses. E, F, and G relays are used for analog as well as digital measurement. The D relays connect and isolate the system digital driver to and from the DUT. During analog testing, the D relay is open to prevent the driver circuit from loading down the DUT, causing false DUT failure reports. During digital testing, the D relays are closed and connect the digital drivers to the DUT to provide digital stimulus.

The analog and digital test switch matrix schematic is shown below. If you have a Vector Performance system, substitute the VP board for the THC in the following illustration.

Analog/digital switch matrix.



Note: The terms *channels* and *nodes* are interchangeable.

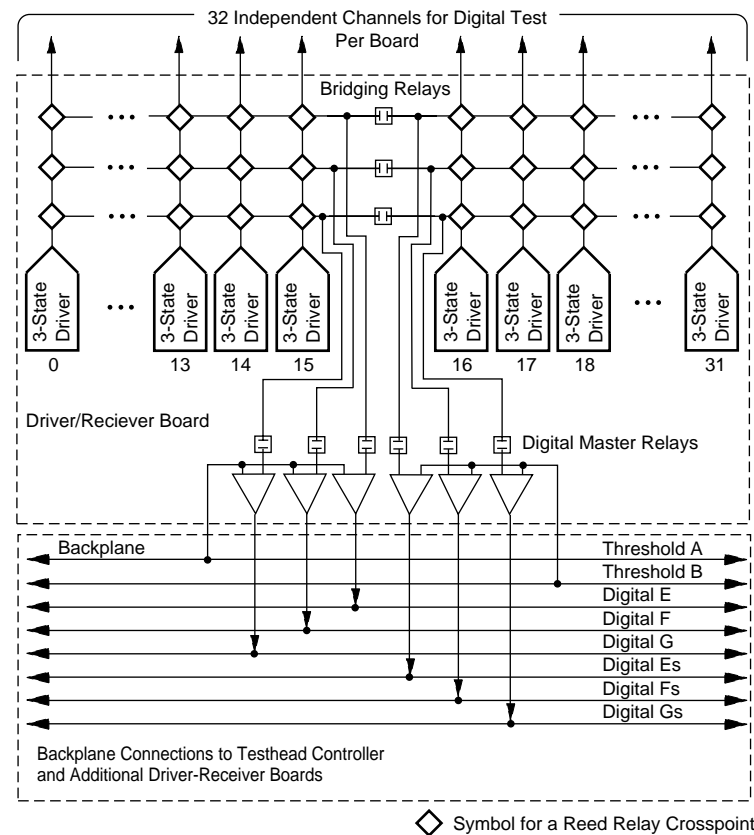
In the following illustration the digital measurement nodes are connected to threshold comparators via the E, F, or G node relays and the digital master (comparator isolation) relays. The threshold is a DC level specified in the test program and generated by the test head controller or vector processor in VP models.

When the comparator input relay closes, the comparator output drives its digital bus line which also connects to the test head controller board (or vector processor board on VP models) for measurement, capture, and analysis.

Each tester node has a dedicated digital driver and D relay for digital stimulus. Any pin can be a stimulator at any time, without having to share any resources with any other pins.

If you have a Vector Performance system, substitute the VP board for the THC in the following illustration.

Digital testing switch matrix



Each driver/receiver board supplies 32 channel drivers and six comparators (receivers). The comparators are arranged in two groups of three. One group obtains its threshold reference from the high threshold, and the other obtains its reference from the low threshold source. For single-threshold testing, the high and low threshold sources are set to the same voltage.

The driver/receiver system thus allows any number of stimulus pins in a single burst, and can analyze responses on up to six pins at one time. If a test requires more measurements than can be acquired in a single burst, the software system will automatically run as many bursts as needed without additional programming.

Relays affecting digital test which are not addressed in the analog test techniques section are explained below.

Relay Type	Function
Analog master relays	Connect the board's matrices to the six analog poles when terminators are used in digital tests.
Comparator Digital master relay	Open during analog testing to protect the comparator ICs from high voltages and also prevents the comparator's input bias currents from interfering with analog measurements. Closed during digital measurements.
Driver D relays	Connect the channel drivers to nodes in the board under test during digital test.
Bridging relays	Join two 3-pole by 16-channel matrices into a single 3-pole by 32-channel matrix. Used during dual-threshold testing to present each selected signal to both comparators simultaneously. Used in single-threshold tests to maximize the number of signals brought to the comparators in each burst.

Analog Test Techniques

This section describes

- Analog measurement theory
- Analog wire modes and measurement concepts
- ATB configurations

The program generator chooses the analog test configurations automatically for the majority of analog components on your board by using shorthand test types.

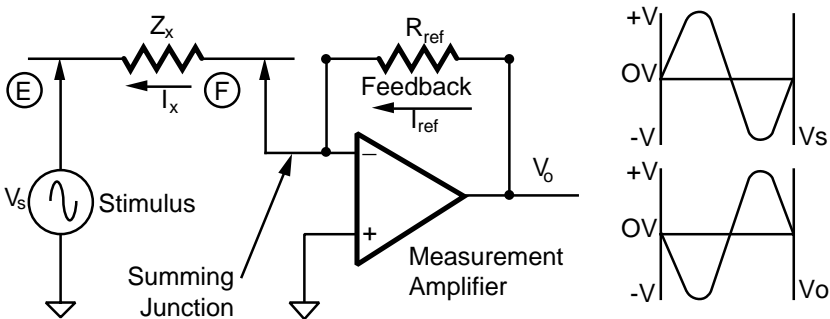
Analog test specifications for each type of analog component are contained in the **Z1800-Series Component Test Reference**, and in the **Z1800-Series Maintenance Reference**, chapter 5.

Analog Measurement Theory

Measurement Operational Amplifier

A key element to an understanding of analog in-circuit testing is comprehending the behavior of a measurement operational amplifier in an inverting configuration.

Simplified schematic of an ideal operational amplifier



The measurement operational amplifier (MOA) both stimulates and measures individual components. The ideal op amp concept is characterized as follows:

- The input impedance to the op amp is sufficiently high to prevent current from entering the (-) input. Therefore, all current flowing through the input path also flows through the feedback path.

$$I_x = I_{ref}$$

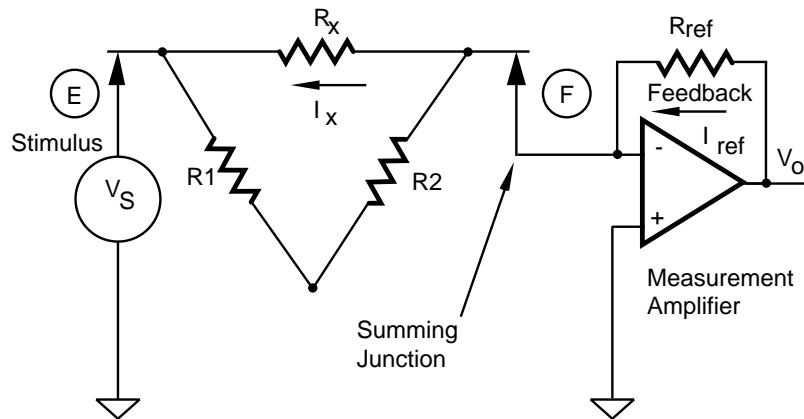
- The gain of the op amp is high enough to keep the inverting (-) input at the same potential as the non-inverting (+) input. Since the non-inverting input is grounded, the inverting input is also held at ground potential (termed “virtual ground” or summing junction).

Given these ideal conditions, the impedance of the device-under-test (Z_X) can be calculated:

$$Z_x = \frac{V_s}{I_{ref}}$$

The 2-wire test method is inadequate, however, for some circuit configurations. Shunting by parallel impedance paths of current around the DUT degrades measurements such that I_X no longer equals I_{ref} . Refer to the illustration of an unguarded configuration that follows.

Unguarded circuit

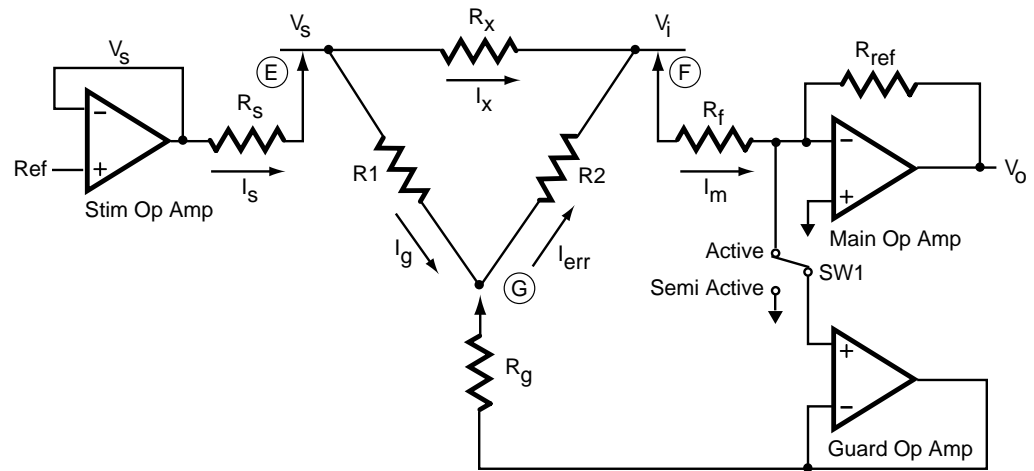


3-Wire Active Guarding

A third wire is necessary to isolate or guard the component from its surrounding circuit. The technique of masking parallel circuit paths by grounding is known as guarding. Use of the amplifier, combined with guarding, allows you to measure a component without influence from surrounding components.

In a 3-wire analog test guarding circuit, resistance paths R_1 and R_2 are connected to ground. Since the amplifier's feedback loop maintains point F at virtual ground, no current will flow through R_2 because both ends of the resistor are at ground.

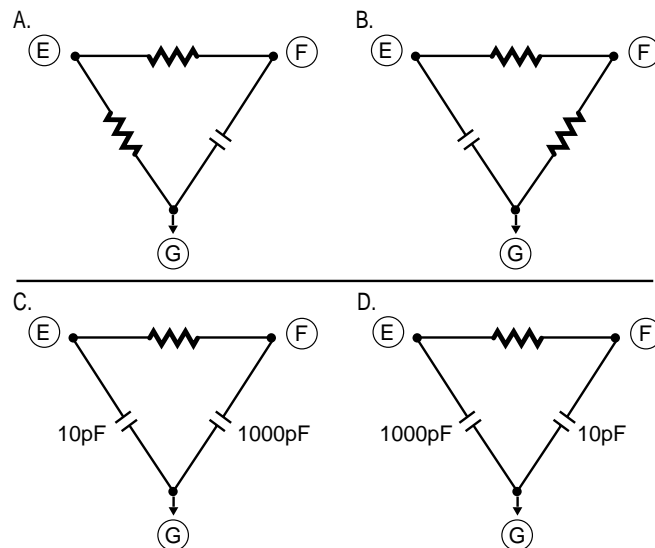
3-wire analog guarding circuit



Consequently, all current through R_x also flows through the computing resistor R_{ref} . Currents that would have flowed in the path parallel to the resistor under test are interrupted, and do not influence the measurement. In theory, analog guards have no bounds, but in practice there are a number of guard ratio limits:

- DC tests (resistors) are limited to ratios under 1000:1.
- AC tests (capacitors) are particularly sensitive to guarded capacitance on the F pole. If a capacitor must be guarded, it is better to connect the stimulus (E pole) to the end of R_x that goes to the capacitor. If capacitors go to both sides of R_x , connect the E pole to the larger capacitor. Refer to the illustration below.

Samples of guarding capacitors



When guarding capacitors, as in Diagrams A and C, the guard amplifier is placed in a potential positive feedback mode via the capacitor between the G and F poles and can become unstable. Oscillation sets in with values between 1 nF and a few μF . For stable readings, you should avoid active guarding in these cases.

3-Wire Semi-Active Guarding

If you use 3-wire semi-active guarding, the feedback path is broken, and the guard amplifier cannot oscillate because switch 1 is connected to analog ground. (See Fig. 5.6.) This guard mode provides slightly less guard ratio since offset errors from the MOA are not automatically compensated for. This mode is the preferred mode whenever capacitors are present between the G and F poles.

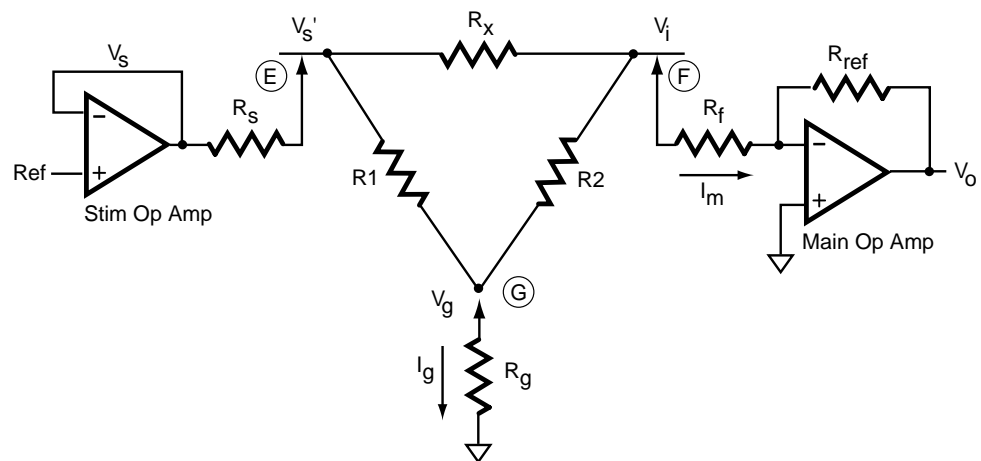
3-Wire Passive Guarding Error Sources

The 3-wire passive guarding setup does not use a guarding amplifier at all. The G pole is directly connected to the analog ground. This mode does not provide for 4 or 6 wire correction.

3-wire passive guarding is sensitive to several sources of error. The illustration below shows how these errors are potential causes for degradation in the measurement result.

Note that resistors R_s , R_g , and R_f are the residual resistances of the test system and fixture.

3-wire passive guarding error sources



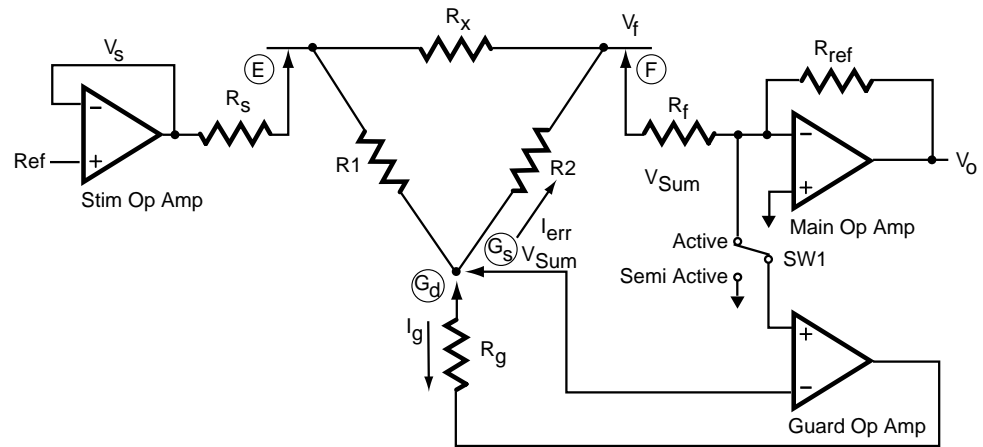
The following list contains the errors commonly encountered in this measurement configuration:

1. Stimulus error on the E-pole. The current drawn out of the stimulus creates a voltage drop across R_s , where $V_s' \neq V_s$.
2. Virtual ground error at the F-pole. The measured current, I_m causes a voltage drop across R_f and forces V_i from 0 volts.
3. Measurement amplifier offset. The amplifier has an inherent internal offset voltage between the - and + inputs.
4. Phase shift for AC tests. The measurement amplifier cannot deliver the requested gain in an AC configuration and the F pole is not held at virtual ground.
5. Guard Potential Errors at the G-pole. The guard current, I_g , causes a voltage drop across R_g and forces V_g from 0 volts.

4-Wire Correction. 4-wire mode uses a guard sense connection in addition to the basic 3-wire mode. The G-pole is split into G_d and G_s providing separate drive and sense paths from the guard amplifier to the guarded point. All of the guard current, I_g , is flowing through the G_d pole. The G_s pole serves as a feedback connection for the guard amplifier. This allows the guard amplifier to adjust for voltage drops across R_g and eliminates guard-resistance errors. The guard point is maintained at the same potential as the summing junction (V_{sum}) because no current flows in G_s (basic op amp principle #1), and $G_s = V_{sum}$ (basic op amp principle #2).

To use this method, two probes and two wires must be provided in the fixture: one on a drive node and one on a sense node. Both probes must contact the same electrical point on the board-under-test.

4-wire active guard



4-wire guarding is most useful when R_f is much less than R_x and R_1 , or R_2 is approximately R_g (1Ω). Under such conditions, $V_{sum} \cong V_f$ and $I_{err} \cong 0$. The 4-wire guard sense connection prevents I_g from contributing to I_{err} .

IMPORTANT: Four-wire mode is possible only when there is only one guard point necessary.

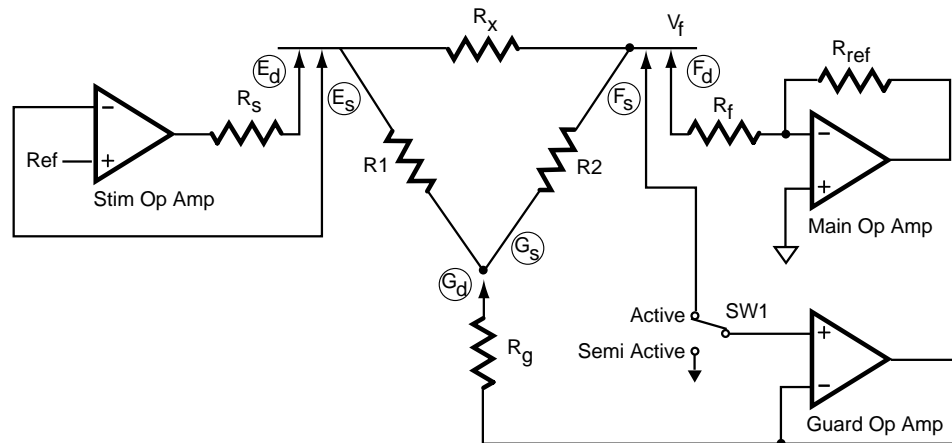
5-Wire Correction. In 5-wire mode the E- and F-poles are split into sense and drive poles. As with 4-wire mode, two probes and two wires must be provided for in the fixture for each split pole. The drive and sense probes for each pole must contact the same electrical node on the board-under-test. The sense probe should be closest to the DUT.

In 5-wire mode, the stimulus drive pole (E_d) and measurement drive pole (F_d) are sensed. The error caused by R_s is eliminated by sensing the actual E voltage and allowing the stimulus amplifier to adjust in response. Error currents flowing in R_2 are eliminated by sensing the actual V_f voltage and driving G_d to that voltage.

5-wire mode is most useful when the DUT impedance (R_x) approaches the system resistance (R_s and R_f), and when remote sensed-guarding is not required or practical. The guarding circumstances include:

- no guards needed, R_1 and R_2 much larger than R_g
- more than one node requires guarding, complicating the placement of a G_s probe.

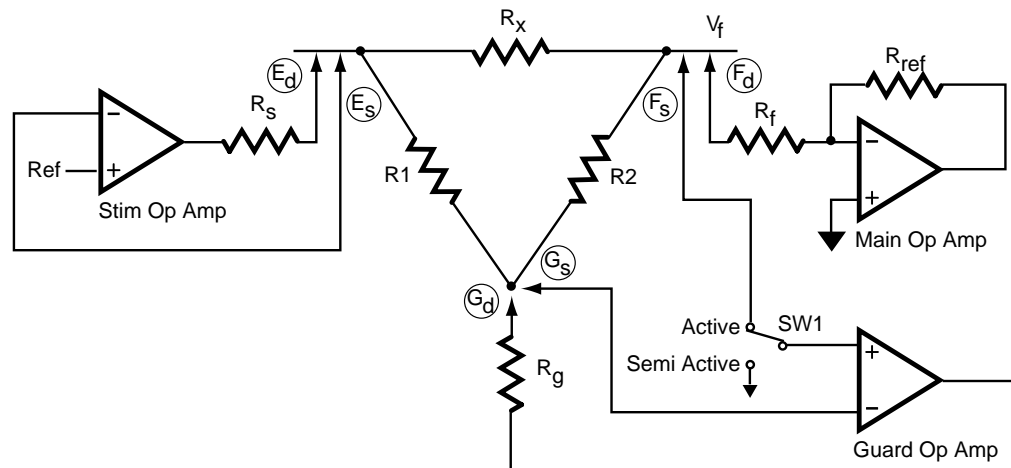
5-wire active guard



6-Wire Correction. In 6-wire mode, the E-, F-, and G-poles are split into sense and drive poles. As with 4-wire mode, you must provide two probes and two wires in the fixture for each split pole. The drive and sense probes for each pole must contact the same electrical node on the board under test. The sense probe should be closest to the DUT.

In 6-wire mode, which combines the benefits of 4- and 5-wire modes, the tester senses each of the three drive poles (E_d , F_d , and G_d) and compensates for the errors associated with each pole.

6-wire active guard



Precise Correction. Precise mode is a general analog measure concept designed to improve accuracy without the problems associated with multiwiring. Precise mode splits the drive and sense poles as far as to the driver/receiver boards and takes two measurements to correct for errors.

Precise mode uses two additional amplifiers, A1 and A2, to facilitate direct measurement across R_{ref} and R_x , eliminating errors caused by R_s and R_f .

The actual voltage across R_{ref} (a precision reference resistor) is measured by A1. The actual voltage on the F side of the DUT is sensed by both A2 and the guard amp. The guard amp keeps

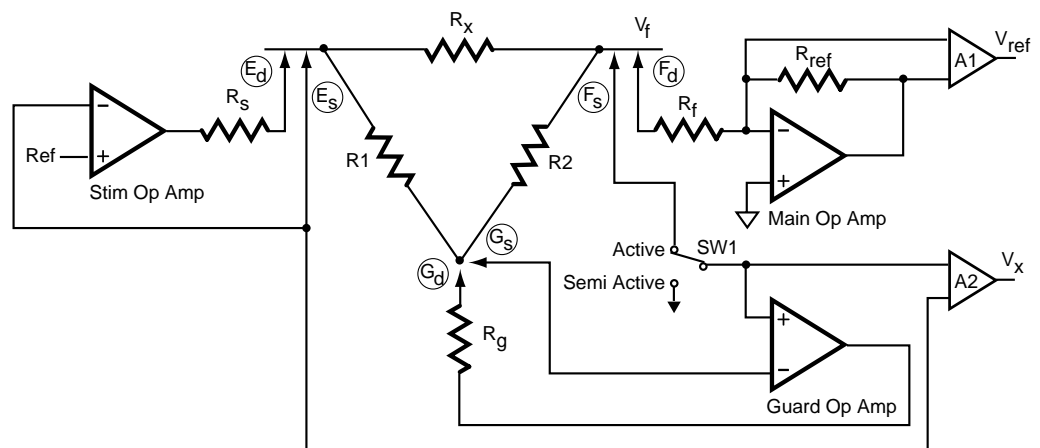
$G_d = G_s = F_s$, such that no current flows in R_2 (no guard error current). A_2 measures actual DUT voltage. Since there is no error current, the following is true:

$$I_{ref} = I_m = I_{dut}$$

$$I_{ref} = \frac{V_{ref}}{R_{ref}} = \frac{V_x}{R_x}$$

$$R_x = \frac{V_x}{V_{ref}} \times R_{ref}$$

6-wire active guard, precise



Precise can be specified for all wire modes in a component Step Worksheet. The illustration above is essentially correct for all wire modes. For example, with 3-wire precise, the only difference is where the sensing takes place. With 5- and 6-wire precise, the sense probes are in the fixture; with 3- and 4-wire, the sensing is at the driver/receiver board.

IMPORTANT: The best possible setup in terms of accuracy is in 5/6 wire mode with Precise turned On.

Higuard. Higuard is a general analog measurement technique for reducing the effects of thermal EMF (Electro Motor Force) in the relay connection leading from the ATB to the DUT. Because of dissimilar metals used in the contacts of the relays, each relay acts as a tiny thermocouple and introduces offset currents into the measurement. The magnitude of the offset depends on the temperature of the contact—the hotter it gets, the higher the voltage it generates. Higuard counteracts this tendency by taking an extra measurement while the stimulus is held at zero and reading the sum of the offset. Then the offset is subtracted from the final reading to yield a more correct answer.

Higuard relies for its initial measurement on a completely discharged DUT, which can be ensured by programming squelch. See also the discussion of Squelch on the following pages.

Higuard and AC measurement: Although Higuard is mainly designed for DC measurements, it also helps when doing AC type measurements as in capacitor or longhand testing with measure

types of PK and PK-PK. In these situations, Higuard works to increase the signal-to-noise ratio of a test. Since it takes an initial offset reading with the stimulus off, it measures background noise on the DUT and then subtracts that from the final reading, in effect, reducing the effects of system noise from the test.

Averaging. Averaging is a general measurement technique to increase stability of a given test. By taking multiple readings and then averaging them, you can obtain a more stable result. The Averaging field in analog Test Properties allows for the number of samples to range from 1 to 255. The more samples that are averaged, the more stable the result becomes at the expense of the time spent to take those samples. The times between individual samplings are not controllable. They are taken as fast as possible.

Wait and Squelch. Every analog measurement goes through the following sequence of internal phases when executing.

1. START—ATB reset, connection to DUT established.
2. SQUELCH—Delay to let the DUT discharge.
3. HIGUARD—No stimulus offset reading.
4. STIM—The stimulus is turned on.
5. WAIT—Delay to let the voltages stabilize.
6. MEASURE—The measurement takes place.

During the **squelch** phase, the DUT is discharged or “squelched” for a certain period of time. This is done after the connection to the DUT has been established and while the stimulus is still at 0 Volts. The Main op amp is set up for high current. This setup effectively discharges the DUT should there be any charges left across it from a previous test. The system provides default internal squelch times which can be extended through the Squelch field in Test Properties. Proper squelch times are essential for capacitor test as well as any tests using the Higuard feature since these test modes rely on a 0 volt starting point.

Wait is a delay which occurs after the ATB has been fully setup and the stimulus has been turned on. It stabilizes the voltage/current on the DUT before a reading is taken. The system provides default internal wait times which can be extended by the Wait field in Test Properties. Proper wait times are essential for components with parallel or surrounding capacitors.

When Wait times are used in conjunction with AC measurements such as in longhand tests using PK, PK-PK, or RMS measure types, they also define the time the measure detectors are open. Minimum gate times are always enforced. See also the longhand Test Types discussion in the **Z1800-Series Component Test Reference**.

Fast Mode. The Fast Mode option on some shorthand Step Worksheets is designed to eliminate the internal squelch and wait times in order to increase throughput. The internal delay times are not always needed depending on the topology of the board under test. The delay times for Squelch and Wait as defined on the Test Properties fields still apply and are the only source for the Squelch and Wait phases of the measurement sequence.

WARNING!

Fast Mode removes the noise measurement and signal averaging from the test result; therefore, Fast Mode test results may not be as stable as Normal Mode test results. We recommend you use Fast Mode when speed is an issue and testability has already been proven.

ATB Test Configurations

The Analog Test Board (ATB) is the tester's analog instrument comprising precision DC and AC stimulus sources, a voltage measurement section, and an amplifier integrated with

computer-controlled switching and timing circuits.

The ATB employs four primary techniques to test analog devices:

- Test V (TV): the ATB measures voltage only.
- Test V Stim V (TVSV): the ATB stimulates with voltage and measures it.
- Test V Stim I (TVSI): the ATB stimulates with current (SI) and measures or tests the voltage (TV).
- Test I Stim V (TISV): the ATB stimulates with voltage (SV) and measures or tests the current (TI).

The ATB can also be configured in Stim V for some digital Gray code and vector tests.

The ATB provides 3 drive and 3 sense terminals, for a total of 6 terminals for full Kelvin connections to nodes on the board under test:

Drive terminals: E_d , F_d , G_d

Sense terminals: E_s , F_s , G_s

In the most common 3-wire configuration, E_d is tied to E_s , F_d is tied to F_s , and G_d is tied to G_s . (A simple component test may use only two wires.) The EFG terminology is referred to in the software as Stim, Meas, and Guard/Ref, respectively.

Terminal or Pole	Definition
E_d and E_s	STIM (stimulus)
F_d and F_s	MEAS (measurement)
G_d and G_s	GUARD/REF (guard or reference)

Analog Wire Modes

There are four analog wire modes: 3, 4, 5, and 6. The driver/receiver switch matrix is configured differently for each wire mode.

Node positions 0 through position 15 on each driver/receiver board connect directly to drive poles. Nodes 16 through 31 connect directly to sense poles.

Driver/receiver bridge relays open and close in response to the selected wire mode, allowing drive and sense for any single pole to either split (open relays) or connect (close relays).

When a pole splits at the driver/receiver as a result of the wire mode selection, you or the software must program two nodes: one node for the sense pole and the other for the drive pole. In such a split-pole configuration, the bridge relays may be used to connect drive nodes to the sense pole or sense nodes to the drive pole.

For example, nodes 2 and 34 are specified as E-poles in a 6-wire test, and are therefore nodes in the drive range on their respective driver/receiver boards. Node 2 is located on the lower half of board 0 and node 34 on the lower half of board 1. However, because the nodes are on different driver/receiver boards, the bridge relay on the second board may route node 34 to the E-sense pole. Since bridge relays apply to a group of sixteen nodes, the final restriction prohibits having drive and sense nodes in the same half of the same driver/receiver board.

The wire modes and Precise attribute apply only to ATB test configurations Test V Stim I (TVSI) and Test I Stim V (TISV). ATB test configuration Test V Stim V (TVSV) is restricted to 3-wire Precise off.

Beta tests ignore wire mode settings and the Precise attribute. Consequently, beta always splits the E pole at both the ATB and the driver/receiver board, tying F and G poles in both places. Additionally, while using the high-volt stimulus, you cannot use the 6-wire differential amplifiers. A single measurement is therefore made at the MOA.

System resistance is subtracted when the Precise parameter is Off, and is not subtracted when Precise is On.

The tester uses Wire mode and Precise to perform analog tests as stated in the following sections.

3-Wire Mode

In 3-wire mode, you may assign any of the 32 channels per driver/receiver (D/R) board to any pole without restrictions.

3-Wire, Precise OFF. Both the ATB and D/R boards tie drive and sense together for all poles. A single measurement is taken at measurement operational amplifier (MOA). See also the longhand Test Types discussion in the **Z1800-Series Component Test Reference**.

Test I Stim V (TISV), 3-wire, Precise off

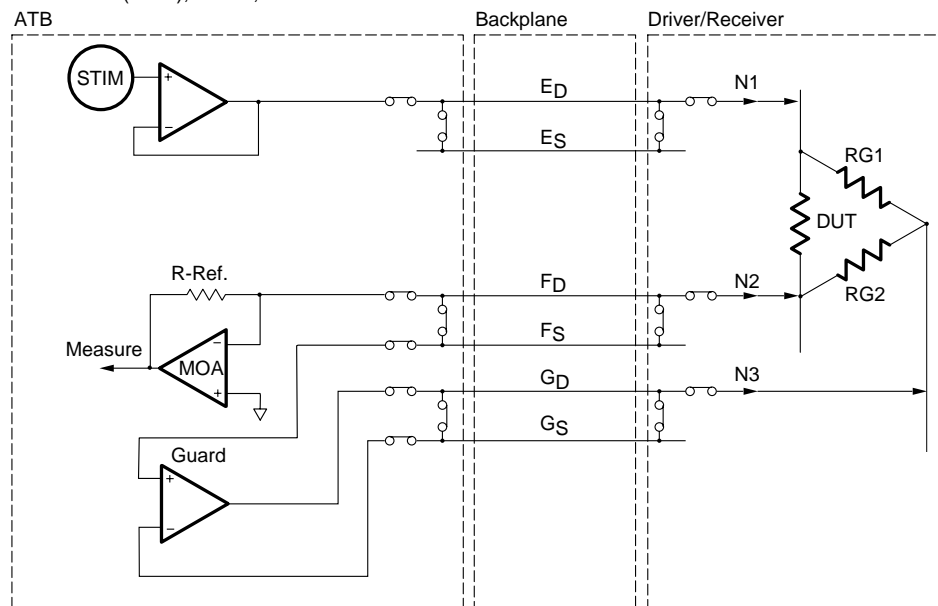
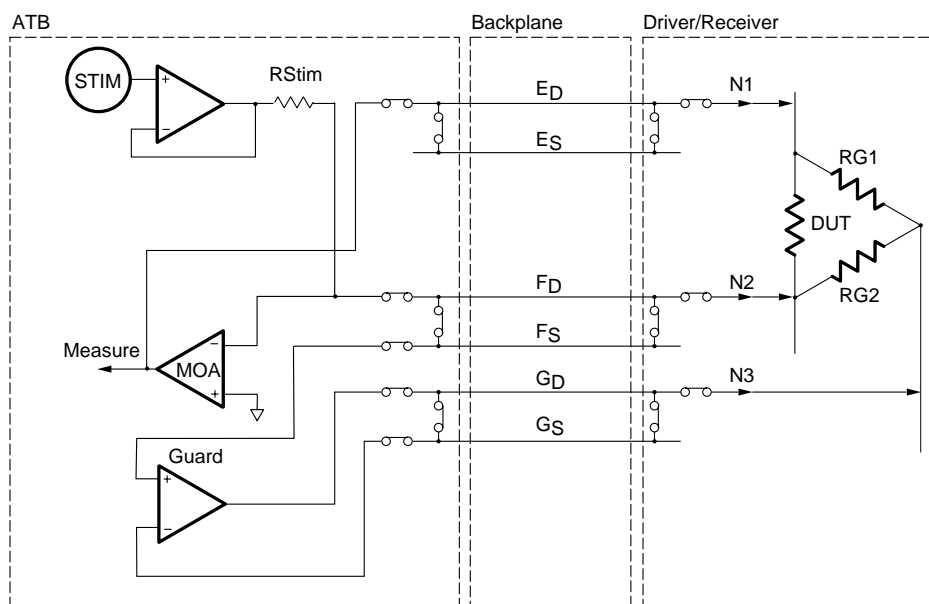
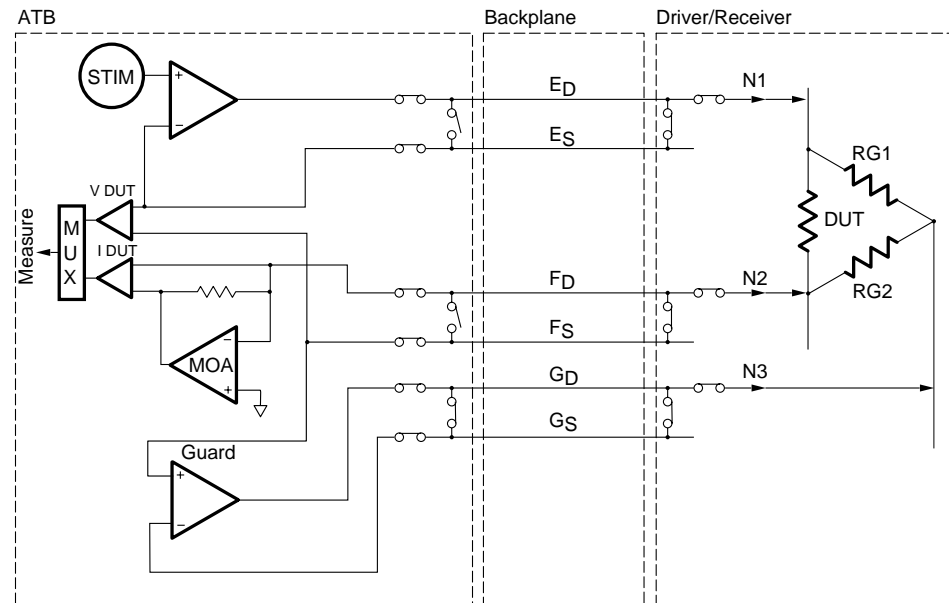


Figure 0.1 Test V Stim I (TVSI), 3-wire Precise off

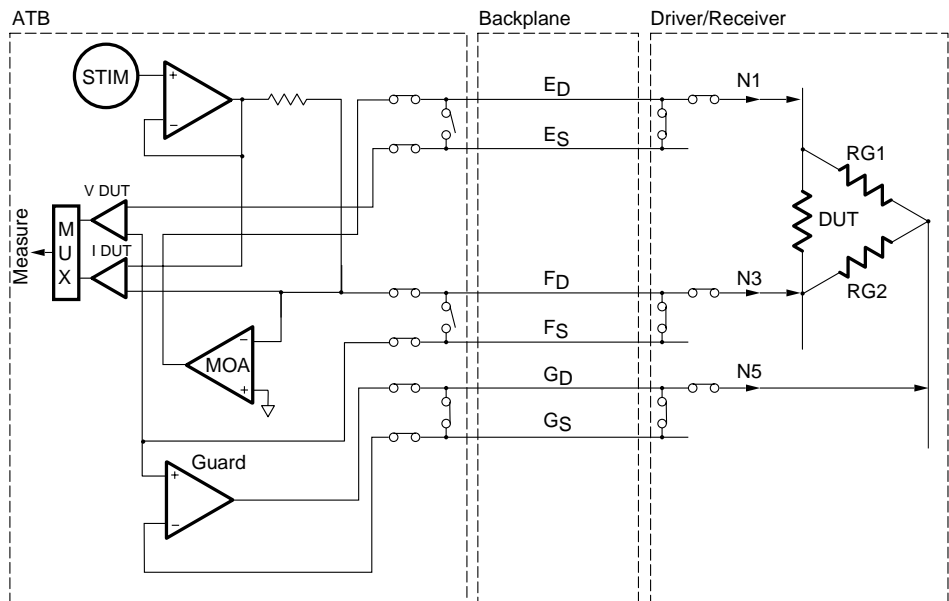


3-Wire, Precise ON. In 3-wire, Precise On mode, the ATB splits E and F poles, and ties the G pole drive and sense together. D/R boards tie drive and sense together for all poles. Two measurements are taken via the ATB's 6-wire MUX and differential amplifiers. One differential amplifier measures the voltage across the DUT and another measures the voltage across the computing resistor, providing 6-wire accuracy without requiring double noding of the E and F poles.

Test I Stim V (TISV), 3-wire, Precise on



Test V Stim I (TVSI), 3-wire, Precise on



4-Wire Mode

4-Wire, Precise OFF. Both ATB and D/R boards tie drive and sense together for all E and F poles. G pole is split at both ATB and D/R boards. Two G nodes are required. A single measurement is taken at the MOA.

Test I Stim V (TISV), 4-wire, Precise off

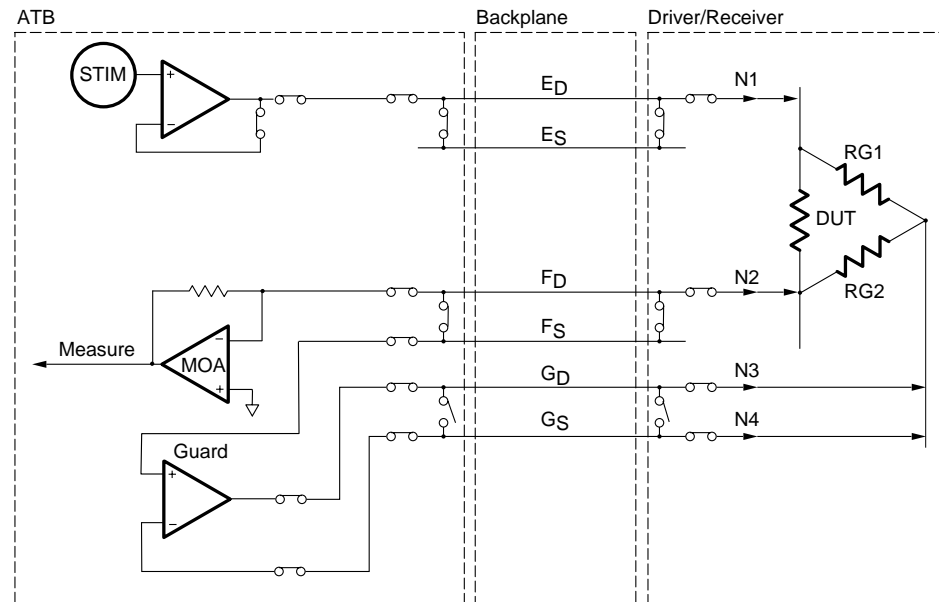
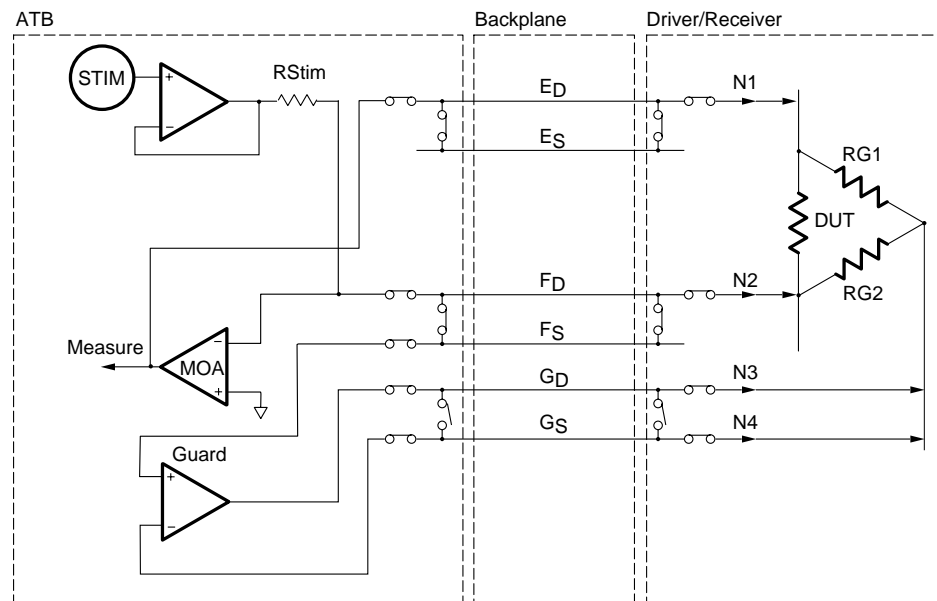
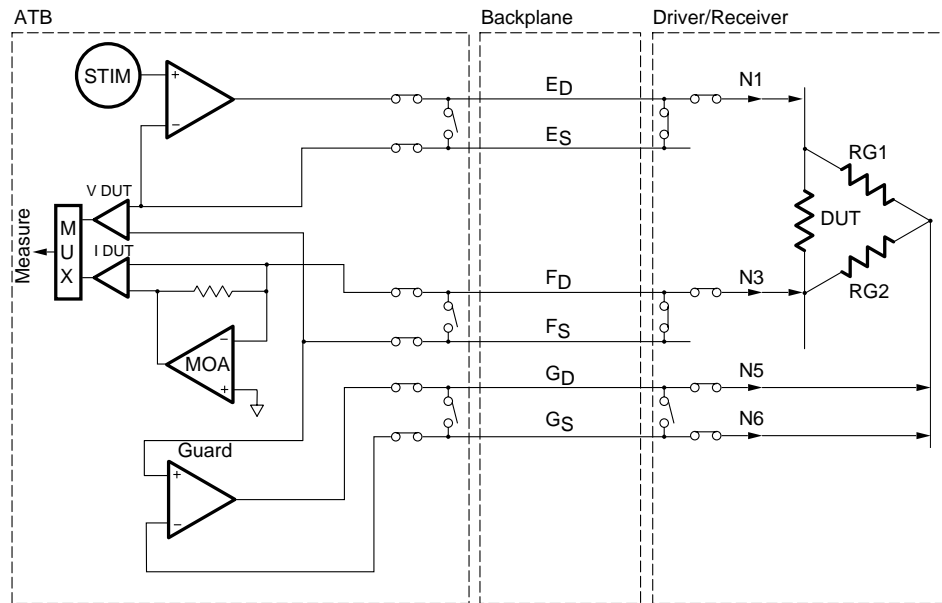


Figure 0.2 Test V Stim I (TVSI), 4-wire, Precise off f

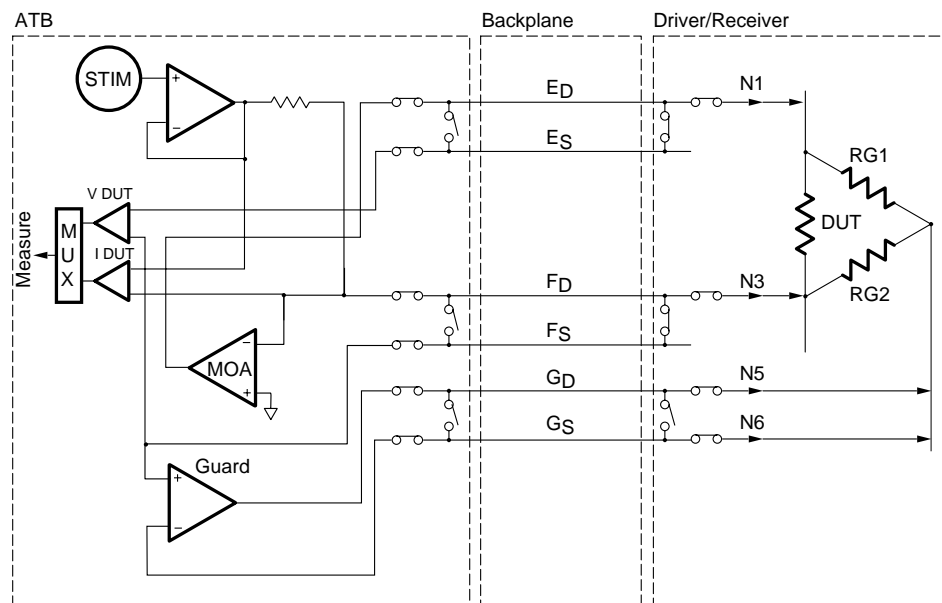


4-Wire, Precise ON. The ATB splits all poles. D/R boards tie drive and sense together for E and F poles. G pole is split at the D/R boards. Two G nodes are required. Two measurements are taken via the ATB's 6-wire MUX and differential amplifiers, providing 6-wire accuracy without requiring double nodding of the E and F poles.

Test I Stim V (TISV), 4-wire, Precise on



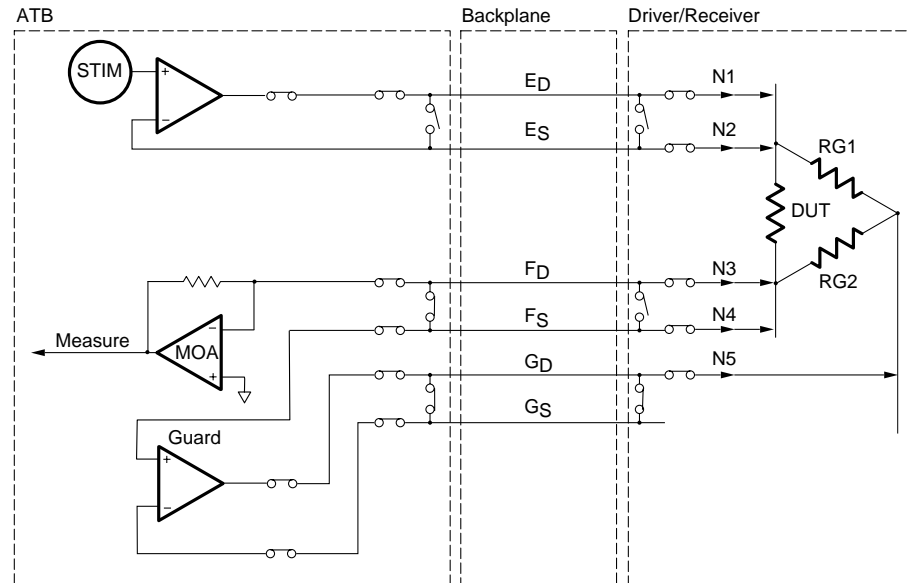
Test V Stim I (TVSI), 4-wire, Precise on



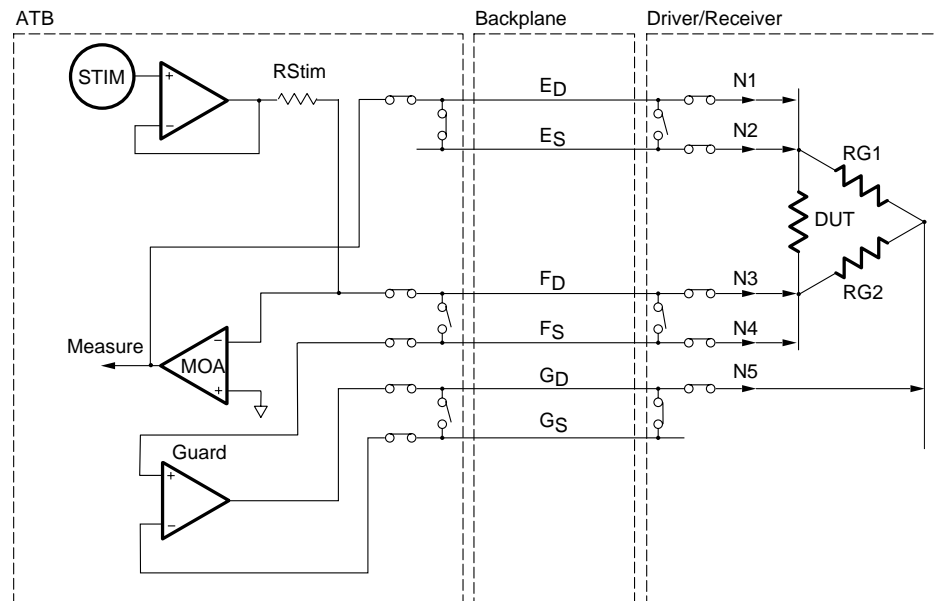
5-Wire Mode

5-Wire, Precise OFF. D/R boards split drive and sense for both E and F poles. ATB splits E pole in TISV, allowing remote sense of stimulus voltage. ATB ties E pole drive and sense for TVSI configuration. ATB ties F pole drive and sense for both configurations. Both ATB and D/R tie drive and sense together for the G pole. Two nodes are required for E pole and two for F. A single measurement is taken at the MOA.

Test I Stim V (TISV), 5-wire, Precise off

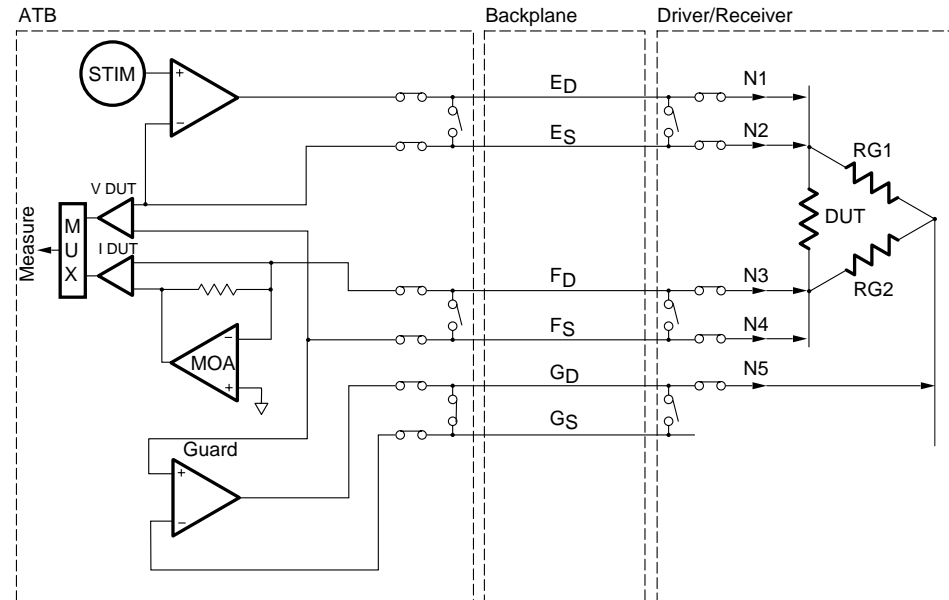


Test V Stim I (TVSI), 5-wire, Precise off

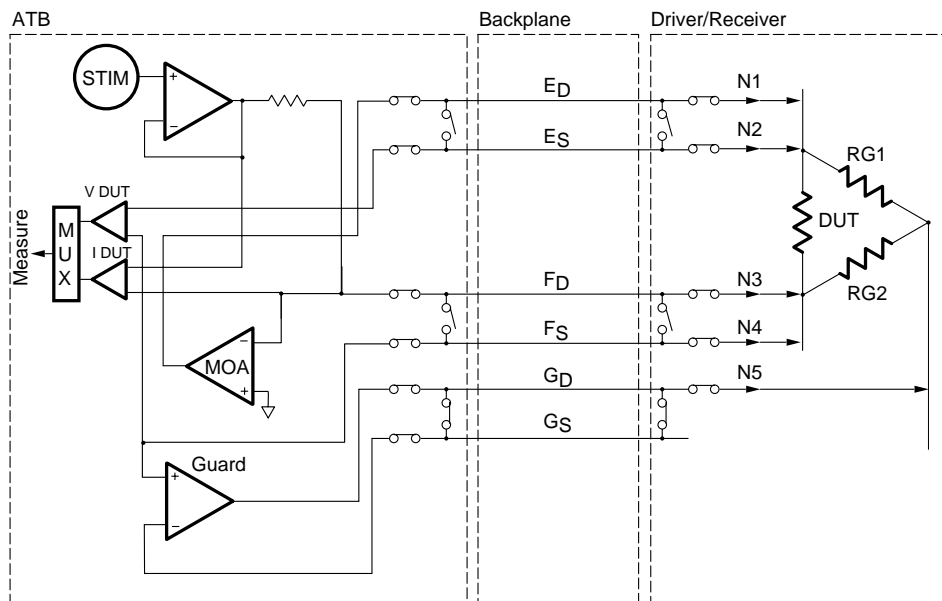


5-Wire, Precise ON. ATB splits E and F poles. D/R boards split drive and sense for E and F poles. Both ATB and D/R tie drive and sense together for G pole. Two nodes are required for E pole and two for F pole. Two measurements are taken via the ATB's 6-wire MUX and differential amplifiers. This configuration allows any number of G nodes.

Test I Stim V (TISV), 5-wire, Precise on



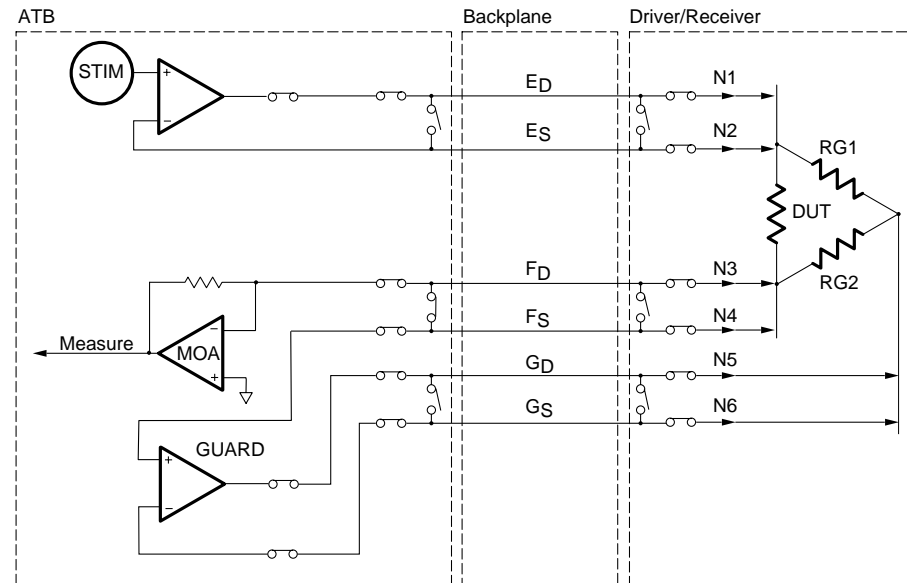
Test V Stim I (TVSI), 5-wire, Precise on



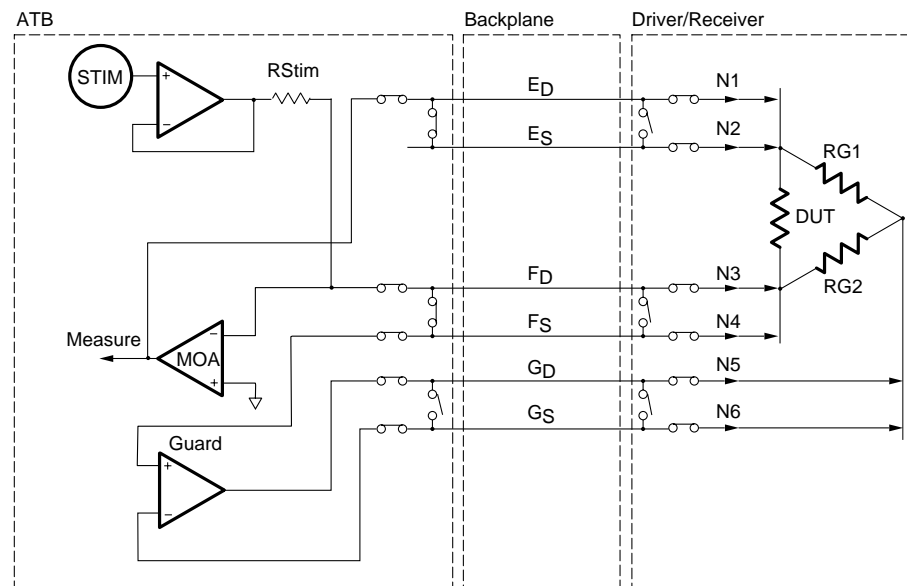
6-Wire Mode

6-Wire, Precise OFF. D/R boards split drive and sense for all poles. The ATB splits E pole in TISV, allowing remote sense of stimulus voltage, and ties E pole drive and sense for TVSI configuration. The ATB also ties F pole drive and sense for both configurations, and splits G pole. Two nodes are required for all poles. A single measurement is taken at the MOA.

Test I Stim V (TISV), 6-wire, Precise off

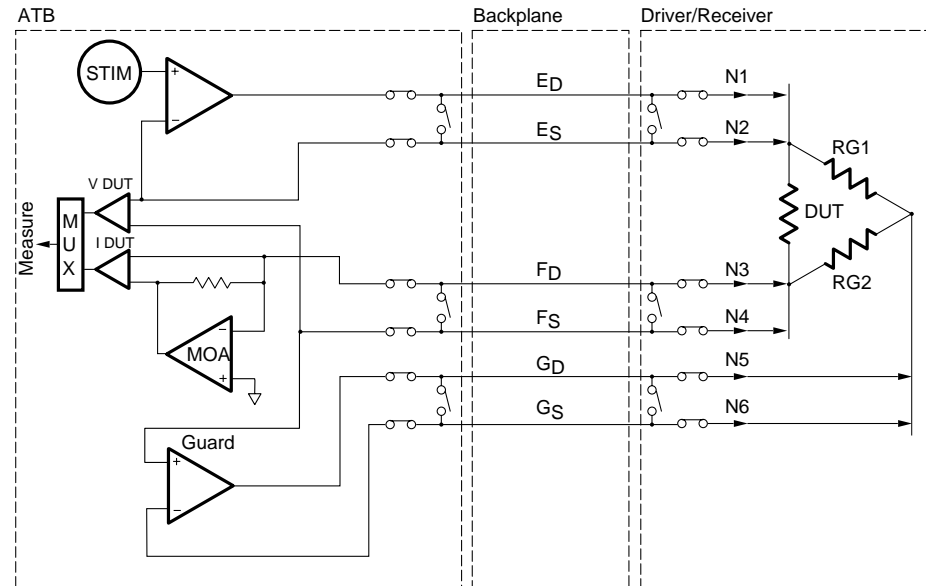


Test V Stim I (TVSI), 6-wire, Precise off

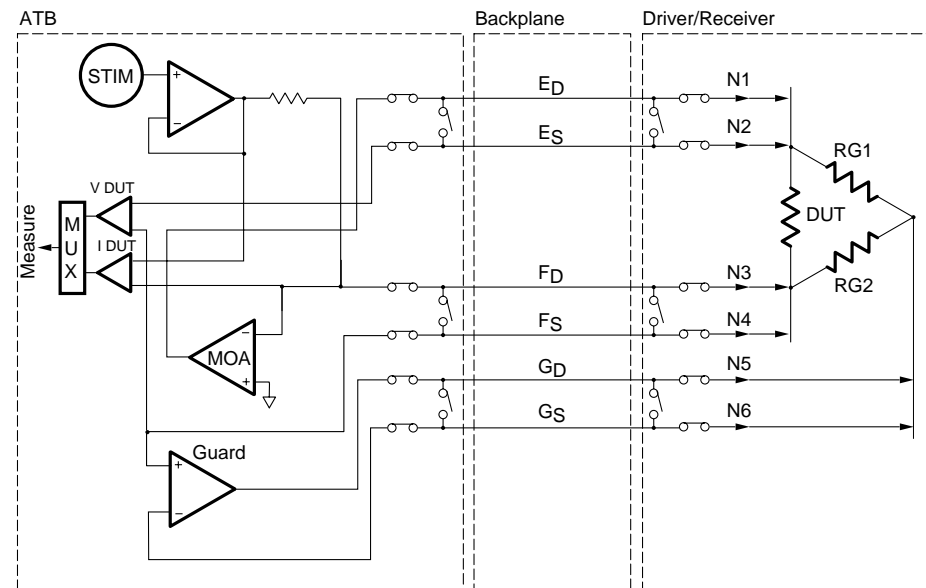


6-Wire, Precise ON. Both ATB and D/R boards split drive and sense for all poles. Two nodes are required for all poles. Two measurements are taken via the ATB's 6-wire MUX and differential amplifiers.

Test I Stim V (TISV), 6-wire, Precise on



Test V Stim I (TVSI), 6-wire, Precise on



Digital Test Techniques

Digital test is based on the technique of electrically isolating a device from upstream, driving elements so the device can be tested for function.

Isolation is achieved with brief bursts of low-impedance stimuli to the device-under-test (DUT), which, if necessary, override the outputs of the upstream elements. Testing is completed by checking for expected results at the DUT outputs. The results will verify whether or not the device is operating correctly, properly oriented and installed, and functional.

The best digital in-circuit test toggles as many of a device's input pins as possible and reads an expected result or signature on the outputs.

Z1800-series test systems are equipped with dual digital test subsystems. You can apply stimulus generation to toggle a device's inputs with either Gray code frequency generators or vector patterns. You can also use signature analysis circuits to measure the expected results (response data) of the Gray code stimulus. Gray code response data is either a four-digit hexadecimal signature (CRC) or a five-digit decimal integer (Count or High). Vector response sampling or measurement compares individual bits to known-good data.

Gray code tests and vector tests can both be used within a single board test program. Understanding the way that disables and guards interact in a program is the key to producing quality test programs efficiently.

Digital Guarding

The function of digital guarding is to add guard information from templated devices to other devices in the test program that require these guards. This process is executed automatically at the end of the Generate phase of Pgen and can also be initiated manually through a menu selection (Dig Guard).

Process

Currently the system software has two types of templates that support digital guard information. These are Gray Code and Vector, which are accessed in the input list by the IC and VEC tokens. During generation, template guard information is saved in a guard file (GFILE.DAT). At the end of generation, this guard file contains all of the digital guard information from the devices in the input list. The digital guarding algorithm is then run to apply this guard information to individual digital test steps.

Guard File

The guard file is a binary file having a time stamp at the top followed by guard records. A guard record corresponds to one digital guard. The record consists of two parts, a guard header followed by guard data. The guard header contains information such as

- the output node number that this guard data will drive
- the ID name of the component that this guard came from
- the number of stimuli that this guard contains
- a status flag that holds information such as the guard type (GC or VEC) and if the guarding algorithm has used this guard record yet

The guard data is a set of nodes and stimulus for those nodes. Each time new guard information is written to the guard file, the time stamp is updated. During program execution, a message may appear: "Warning: Digital guard data out-of-date, run Pgen/Dig Guard to correct this." The message indicates that guard information has been added to the guard file since the last time the guarding algorithm was run. To prevent this message, go to the Pgen menu and select Dig Guard and add new guards to the test program.

Guarding Algorithm

The guarding algorithm reads one guard record at a time from the guard file. For each guard record, it gets the node number that this record is going to guard. It then searches the input list database for all of the components that use this node, except of course the component that this guard came from.

Several checks are made to see if the guard can be applied to a test step.

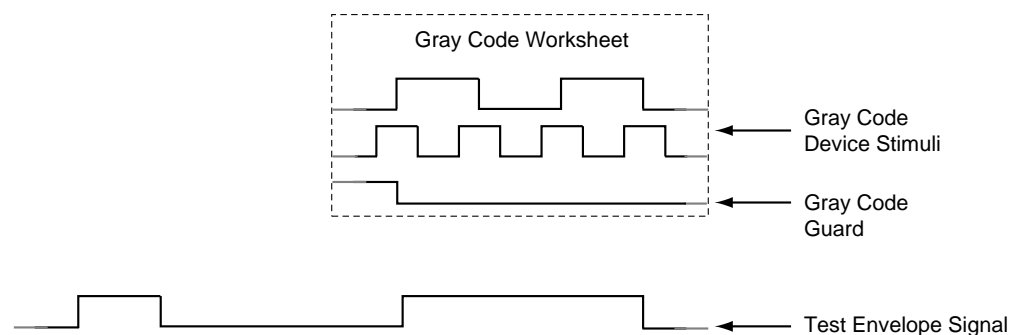
- First, the guard is rejected if any of its stimulus nodes are on the power bus.
- Next, if the test step that uses this guard node is not generated, the guard is not added. Each test step has a flag to indicate that guards have been added to it. If the flag in the test step shows that this step has been guarded and the flag in the guard record shows that this guard record has been applied to the test program, then guarding of the step for this node has already been done; otherwise the guard algorithm continues.
- The next test is to check for any stimulus node conflicts. A stimulus node conflict occurs when the guard node for which guard stimulus is to be applied is already used in the test step. Only those guard nodes from a guard record that is not in conflict is added to the test step. For example, if a guard record calls for four guards to be added to a step, and one of those guards is in conflict, then three guards are added, and the one in conflict is omitted.
- If no rejections occur during these checks, then the guard is added to the test step, the guard record in the guard file is marked as having been applied, and the flag in the test step is marked as being guarded.

Disables and Guards in Digital Tests

Gray code tests produce regular, coherent stimulus patterns. The frequencies used for stimulus provide high fault coverage for combinational devices such as SSI/MSI and memory devices. In addition to the stimulus pattern, you can add Gray code guards to better isolate the device (select Edit then Gray Code Guards from the Step Worksheet menu bar). If a device test includes Gray code guards, the guards run at the same time as the test burst.

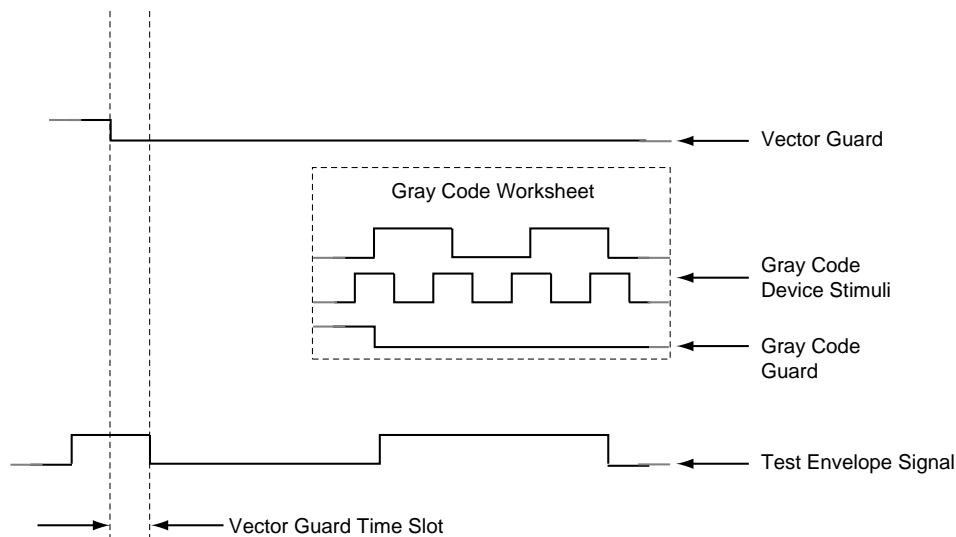
The signal test envelope below is a synchronization signal provided at the front panel of the tester for synchronizing an oscilloscope or logic analyzer.

Test envelope signal synchronized with Gray code stimuli and guard



In addition, if vector guards are used in the Gray code device test, they are bursted before the Gray code test with guards. The test burst synchronization pulse shows the timing of the leading vector guards.

Vector guard timing synchronized with test envelope signal

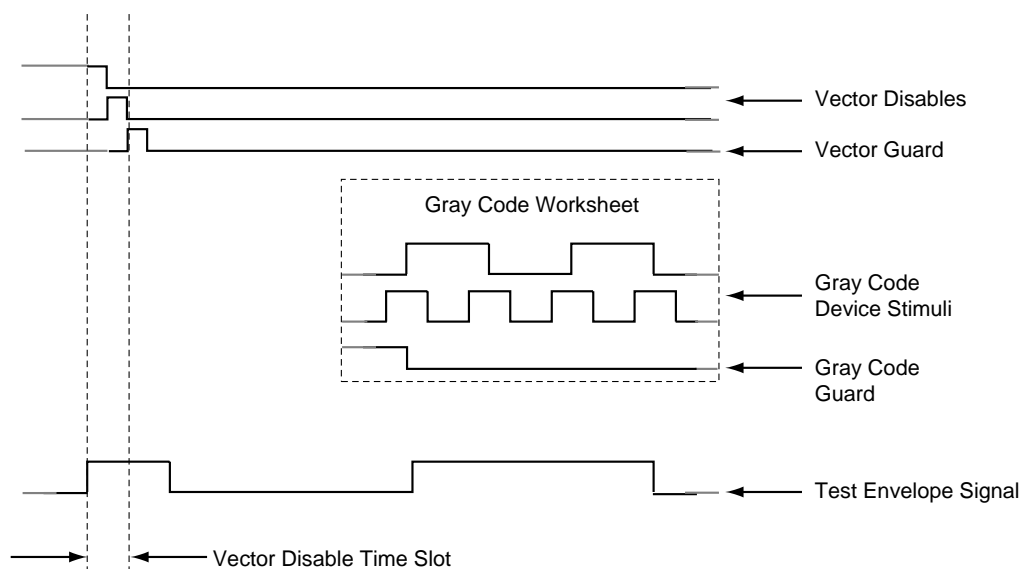


Disables, used to disable an entire board, can also run in conjunction with a Gray code device test. Use disables to put the entire board in a quiescent, or preferred state for testing.

Disable stimuli are similar to guards, but are specified one time at the beginning of digital testing and run with every digital device test.

There are two types of disables: Gray code disables and vector disables. The timing of a vector disable appears below:

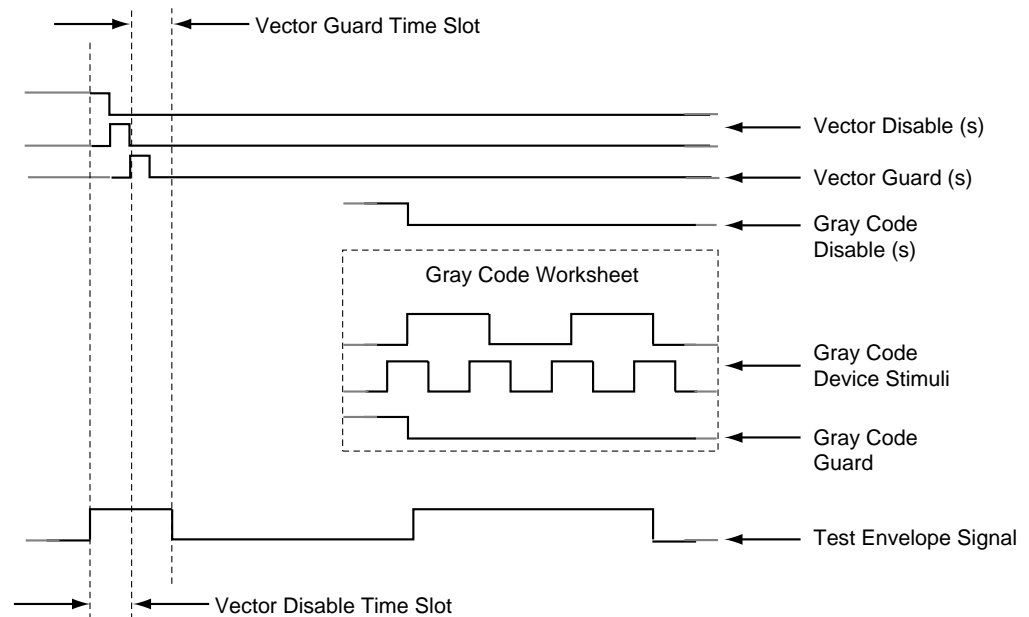
Vector disable timing



Both vector guards and vector disables can employ any sequence of highs and lows. These sequences run within the general timing periods shown in the previous drawing. The last state of the guard pin or disable pin is held through the device test burst. In summary, the order of execution is

1. Vector disables
2. Vector guards
3. Gray code device test, with simultaneous Gray code guards and disables. The overall timing appears below.

Relational display of timing for vector and Gray code disables and guards



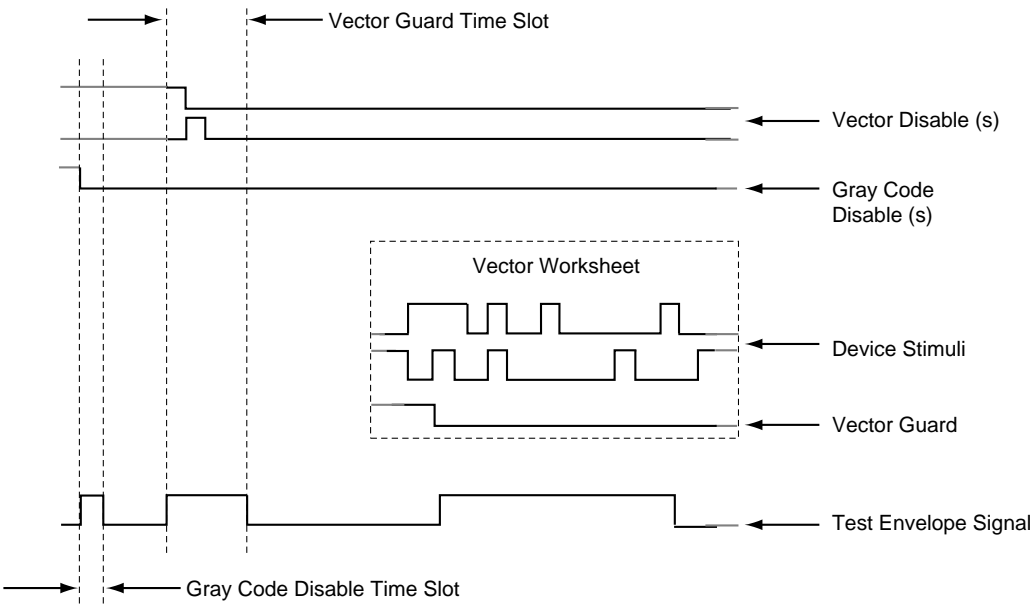
If both a Gray code disable and vector disable have been defined for the same pin, then only the Gray code disable will run if the current test is a Gray code test, and only the vector disable will run if the current test is a vector test. It is important to note that if a node is used both as a disable node (Gray code disable or vector disable) and it is also used in the specific device test, the disabling activity will be nulled with a static logic low level.

Disables and Guards in Vector Tests

Vector tests provide full control of the state of every pin on every pattern within a vector burst. Vector tests are best used for custom devices where CAE (Computer Aided Engineering) test vectors are already provided.

The overall timing of a vector-based device test sequence is shown in the following illustration.

Timing for vector -based device test sequence



If neither vector or Gray code disables are specified, the synchronization pulse for that disable group will not be present at the test envelope signal output. If you program both vector disable and Gray code disable on the same pin, then the Gray code disable will be nulled, as this is a vector test, and the vector disable would then become the preferred disable type.

Summary of Stimulus Hierarchy

If you combine stimuli, measures, guards, or disables on the same node, the following stimulus hierarchy determines the pin behavior on any particular test in a program.

Vector Tests	
Highest priority	Vector stimulus and/or measure and/or guard
	Vector disable
Lowest priority	Gray code disable
Gray Code Tests	
Highest priority	Gray code stimulus and/or measurement
	Gray code guard
	Gray code disable
	Vector guard
Lowest priority	Vector disable

See the **Z1800-Series Board Test Tutorial** for detailed information about adding disables to the test program.



6 PROGRAM GENERATOR INPUT LIST REFERENCE

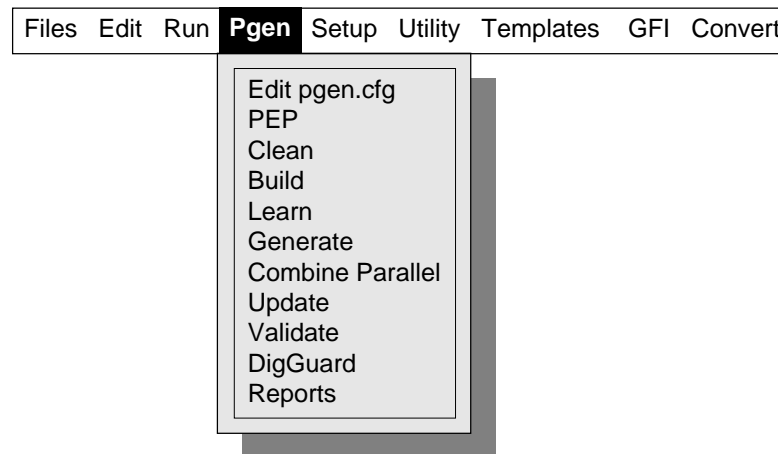
After a brief overview of the program generator menu, Chapter 6 discusses the 18xx input list structure and how you can modify the list to achieve desired results.

The program generator uses both an input list and test templates of analog and digital components from various libraries to create a test program.

You will also find information about how to create test programs with C.1 input lists.

Introducing the Program Generator Menu

To access the program generator, select Pgen from the Main menu.



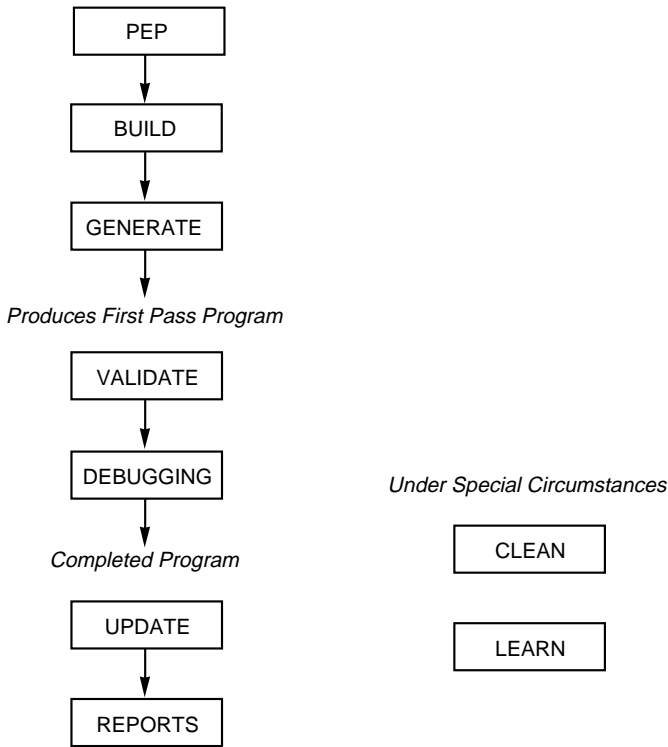
The Pgen selections are described below:

Menu Selection	Description
Edit PGEN.CFG	Provides access to local (default) or global PGEN.CFG. If neither exists, you can create default files.
PEP	(Programmer Efficiency Package) Examine input list; analyze testability of components. Use also to generate a report on recommendations and/or to automatically make changes to input list. See Programmer Efficiency Package User's Guide.
Clean	Removes old files before building a new component database from input list IPL.DAT. Do not use Clean with established program; Clean will destroy the created data. For more information see chapter 7, "Program Generator Tools."
Build	Creates the component database (IPL.DBF and various .NDX files) from the input list (IPL.DAT). Perform Generate after. For more information, see chapter 9, "Program Generator Tools."
Learn	Learns the interconnection topology of a board. For more information, see Z1800-Series Board Test Tutorial.
Generate	Automatically generates a test program (ICT.TST) from the component database. With existing programs, you perform Clean to delete faulty established data, then Build followed by Generate.
Combine Parallel	Creates parallel component tests from individual test steps. Combines various components which have the same nodes. See Z1800-Series Board Test Tutorial and chapter 7, "Program Generation Tools."
Update	Update the component database after program (ICT.TST) changes. Use frequently as you debug the program.
Validate	Improves stability and find guards for capacitors, resistors, and MultiScan tests. See chapter 9, "Test & Debug Tools."
Dig Guards	Finds guards for digital devices. Invoked automatically by Generate. Use frequently as you debug the program. See chapter 5, "Test Techniques & Strategies."
Reports	Prepare reports relating to nodes (NODE.LST), components (COMP.LST), topology (TOPOLOGY.LST) and input list (IPL.LST).

Overview of the Pgen Menu System

The following provides a general diagrammatic overview of the Pgen menu system, showing the commands and the files that they draw from, create, or remove. Since there are a number of ways to generate a test, the following chart is not prescriptive. For example, it is not necessary to invoke Clean each time you generate a program.

See the section, Using Clean and Build, in chapter 7 for additional information.



Understanding
Input List Structure

Program generation requires information from the input list file, IPL.DAT, a highly structured text file of board-under-test components data, buses and connections. This input list is a collection of records containing tokens, parameters, and nodes. Each board directory which generates from Pgen must have a separate IPL.DAT file.

Input lists also exist in another form, for instance, IPL.DAT is converted into a binary database IPL.DBF when you run Build.

See the section, Using Clean and Build, in chapter 9 for additional information.

IPL.DBF is used when you select Generate to create the ICT.TST (in-circuit test) file and for all internal references to board topology information such as APC, Shorts Locator, Validate, and so on.

IPL Records

A record is a section of the input list file IPL.DAT dedicated to a single component or flow of control test step. A summary of record syntax is included below.

Syntax	Explanation
Length	10240 characters maximum
Terminator	Newline (Carriage Return-Line Feed—CR-LF)
Escape from Newline character	, or \
Comment convention	/* anywhere */
Ignored characters	Blank, Tab, Form Feed
Lowercase/Uppercase	Converted to upper outside quoted strings
Field separators	,

Records consist of fields of up to 256 characters, separated by commas. Maximum characters per record is 10240.

IMPORTANT: You can exceed the limits, especially in Continuities (Cont) and Special Cases (SC) sections. In such cases, re-enter the token followed by the remaining node numbers.

Since component tests vary, the arrangement of fields varies from record to record. Some records have fields that specify the component’s name, its test parameters, and nodal connections. An example of a record from IPL.DAT is shown below.

<u>Token</u>	<u>ID</u>	<u>Description</u>	<u>Value</u>	<u>Tol 1</u>	<u>Tol 2</u>	<u>Pin 1</u>	<u>Pin 2</u>
C	, C8	,"Reg cap"	, 8.00nF	, 6.00%	, 9.00%	, 318	, 319

Other records may consist only of a flow of control token and a description such as

<u>Control Token</u>	<u>Description</u>
IPL	, A very long test string for an IPL is put in prog."

A record may contain up to 10240 characters and 2048 fields.

A record can wrap around to the next line by escaping the CR-LF with a slash (\) or comma (.). The backslash does not substitute for a comma. See the examples below. The wraparound must be followed by a numeric field.

The following are acceptable forms for escaping to a new line:

```
RP_SI, R33, , 4.7KO, 5%, 111, 112, 113,
114,115,
116,117,118,119,120
```

or

```
RP_SI, R33, , 4.7KO, 5%, 111, 112, 113,
114,115,
116,117,118,119\,
120
```

or

```
RP_SI, R33, , 4.7KO, 5%, 111, 112, 113,
114,115,
116,117,118,119,\
120
```

Records are separated by newlines, which in DOS is CR-LF.

Comments are strings which start with a slash star (/*) and end with a star slash (*). There is no limitation as to comment length or location. Comments cannot be nested.

The following characters are truncated from the text ipl: \r, \f, space, and tab. Space and tab are not affected if they occur within a quoted string.

The syntax is not case sensitive. All lowercase letters are converted to uppercase upon processing. Case conversion can be negated by placing the string in double quotes, as in “This sentence is mostly lowercase.” We suggest you use quoted strings only in ID and Description fields.

IPL Tokens

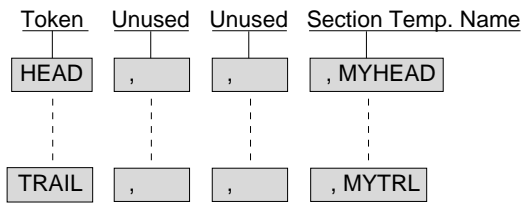
The IPL token or keyword is the first field in a record. All fields that follow are arguments to the token. The token specifies the number of parameters. Tokens may be of the following types:

- Control
- Component
- Template Component
- Special
- Link

These tokens are explained below in detail.

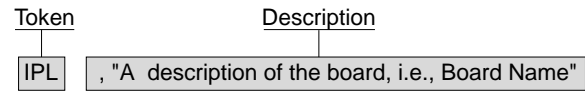
Control Tokens

Header/Trailer



A section template can be inserted into the program for the Header and Trailer sections respectively. The two unused fields are not currently used by the program generator and should be left blank. A section template can be created by modifying a Header (or Trailer) section and then saving it under a specific name by using the Component Select/Utility/Write function. The default sections are respectively Header and Trailer.

IPL



A description for the board can be placed into the program header with this token. Use a quoted string to retain lowercase. The string is limited to 64 characters. The default section is Header.

GND



The GND control token is used to list the nodes which are connected to the ground bus of the DUT and therefore have a potential of 0 volts when the DUT is connected to power. These nodes are stored in the program Header (PRGMVARS test step) and are used as ground reference nodes during power on analog tests. The node list is limited to 5 nodes. If you have more than 5 ground reference nodes on your board, list the lowest five and use these throughout your input list. The default section is Header.

PWR5

Token	ID	Description	Meas.	Ref.
PWR5	, PWR	, "5 Volt"	, 33	, 0

The PWR5 control token turns on the 5 volt power supply and performs a measurement on the listed nodes. The first node (33) is the measurement point; the second node is the reference point. The default section is Board Power.

IMPORTANT: The PROGSUPWS (programmable power supply worksheet) PGEN.CFG variable, which has values of Yes or No, controls the generation of Step Worksheets for programmable or fixed power supplies.

PWR5_5

Token	ID	Description	Value	Tol	Meas.	Ref.
PWR5_5	, PWR	, "3 Volt"	, 3V	, 8	, 23	, 0

The PWR5_5 control token turns on the 0–5.5 V power supply and performs a measurement on the listed nodes. The first node (23) is the measurement point; the second node is the reference point. The default section is Board Power.

PWRA and PWRB

Token	ID	Description	Value	Tol	Meas.	Ref.
PWRA	, PWR	, "10 Volt"	, 10 V	, 8	, 23	, 0

Token	ID	Description	Value	Tol	Meas.	Ref.
PWRB	, PWR	, "-8 Volt"	, -8V	, 8	, 11	, 0

IMPORTANT: The PROGSUPWS (programmable power supply worksheet) PGEN.CFG variable, which has values of Yes or No, controls the generation of Step Worksheets for programmable or fixed power supplies.

The PWRA and PWRB control tokens turn on the A or B adjustable power supplies (or programmable power supplies if you have the programmable power supply control board installed and PROGSUPWS is set to YES in PGEN.CFG). The value field holds the voltage that the power supply has been set to. It has to be specified in Volts. The Tolerance field holds the tolerance for the adjusting phase as well as the following measurement. The node list specifies the nodes for measuring the voltage on the DUT. The first node is the measure point, and the second node is the reference point. The default section is Board Power for both tokens.

IMPORTANT: Do not mix PWRA or PWRB (adjustable or programmable) tokens with PWR_A or PWR_B (fixed) tokens. Mixing these tokens in the input list produces an error message.

PWRA_F and PWRB_F

Token	ID	Description	Value	Tol	Meas.	Ref.
PWRA_F	, PWR	, "15 Volt"	, 15V	, 8	, 23	, 0
PWRB_F	, PWR	, "-15 Volt"	, -15V	, 8	, 11	, 0

These tokens are the same as the PWRA/PWRB above except that they adjust the power supply to 15 or 12 volts fixed. The value field has to be either ± 12 or ± 15 V; no other value is allowed. The default section is Board Power.

IMPORTANT: The PROGSUPWS (programmable power supply worksheet) pgen.cfg variable, which has values of Yes or No, controls the generation of Step Worksheets for programmable or fixed power supplies.

PB

Token	ID	Description	Value	Meas.	Ref.
PB	, PB1	, BUS1	, 78V	, 67	, [0]

PB creates a power bus voltage test in the board power section. The pin list can have 1 or 2 pins specified. If only one pin is given, the system measures against system ground rather than against the reference specified in pin 2. The test generates with a default tolerance of 5%. If 5% is not acceptable, use PBUS. The default section is Board Power.

PBUS

Token	ID	Description	Value	Tol	Meas.	Ref.
PBUS	, PB2	, BUS2	, -8V	, 5%	, 68	, 0

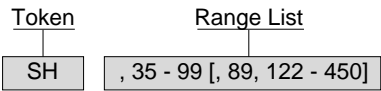
PBUS is the same as token PB but adds a tolerance field for the test. This token always needs a Measure (1) and reference pin (2). The default section is Board Power.

DCHG

Token	ID	Description	Pin 1	Pin 2
DCHG	, C5	, "Discharge"	, 56	, 89

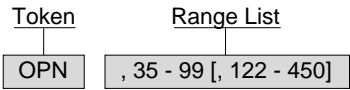
DCHG creates a Discharge statement for the nodes found in the pin list. The default threshold voltage is 5 mV with a maximum timeout of 25 seconds. The default section is Discharge. You can change the defaults with the DCTHRESH and DCTIMEOUT parameters in PGEN.CFG.

SH



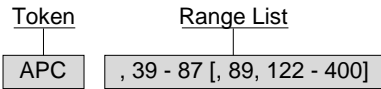
SH creates a shorts test. The range list can contain multiple ranges. The ranges have to be in ascending order. A single node is interpreted as a range to itself. The default section is Shorts.

OPN



OPN creates an opens test (fixture selftest with shorting plate). The range list syntax is the same as for shorts (SH). The default section is Opens.

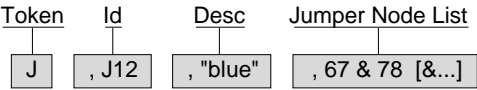
APC



APC (Auto Probe Check) creates a global APC test which runs each time the program executes. The range list is specified the same way as for shorts. The default section is Discharge.

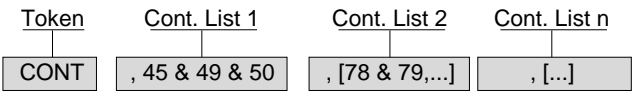
Component Tokens

J



J creates a jumper test on the specified nodes. Jumper can have multiple node groups. The default section is Jumpers.

CONT



Cont creates continuity tests with as many continuities as are specified. This example shows two continuities specified. The default section is Continuities.

SC

Token	Sc List 1	Sc List 2	Sc List n
SC	, 22 & 12 & 56	, [86 & 47,...]	, [...]

SC creates a special case/ignore statement with as many special cases/ignores as are specified. This example shows 2 entries. The default section is Ignores.

L

Token	ID	Description	Value	Tol	Pin 1	Pin 2
L	, L1	, "COIL"	, 100mH	, 20%	, 122 [&...]	, 123 [&...]

L creates an inductor test. The unit of the value field is in H (Henries) and the multipliers are m (milli), u (micro) and n (nano). The node list should contain the ampersand (&) if pins are double-noded. The maximum number of nodes per pin in the pin list is limited to 5 entries separated by &. The default section is Inductors.

C

Token	ID	Description	Value	Tol+	Tol-	Pin 1	Pin 2
C	, C1	, "cap"	, 10nF	, 10%	, 10%	, 82 [&...]	, 39 [&...]

Value is the value of the component in unit F (Farads) with allowable multipliers m (milli), u (micro), n (nano) and p (pico). The node list should contain & if pins are multinoded. The default section is Capacitors.

R

Token	ID	Description	Value	Tol+	Pin 1	Pin 2
R	, R1	, "Res"	, 1KO	, 7%	, 84 [& ...]	, 39 [& ...]

Value is in Unit O (Ohms) with possible multipliers of K (kilo) and M (mega). The node list should contain "&" if pins are multinoded. The default section is Resistors.

POT

Token	ID	Description	Value	Tol	Pin A	Pin B	Pin C
POT	, P1	,"pot"	, 5KO	, 12%	, 129	, 130	, 131

POT is the same as resistor in terms of value and tolerance except for the pins. Pin A and Pin C are the ends of the potentiometer whereas Pin B is the wiper. This creates a multipage test for potentiometers. Be sure to have an SC statement for each Pot. The default section is Switches/Pots.

REO

Token	ID	Description	Value	Tol	Pin 1	Pin 2
REO	, REO1	," "	, 5KO	, 12%	, 128	, 135

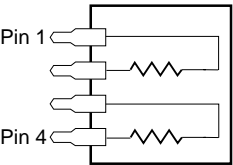
REO is the same as resistor but creates a specific rheostat test. The default section is Switches/Pots.

Resistor Packs

RP_SI

Token	ID	Description	Value	Tol	Pin 1	Pin 2	Pin n
RP_SI	, RP1	,"single inline"	, 5KO	, 5%	, 34	, 56	, ...

RP_SI indicates an Rpack with separate resistors in a single in-line package. The default section for the following resistor packs is Rpack.

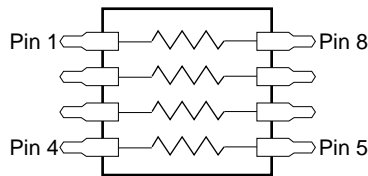


An example of an RP_SI IPL.DAT or input list entry:
RP_SI,RP1,"Single inline",5KO,5%,34,56,78,145

RP_DI

Token	ID	Description	Value	Tol	Pin 1	Pin 2	Pin n
RP_DI	, RPU4	,"Respack di"	, 2.20KO	, 7.00%	, 0	, 145	, ...

RP_DI indicates an Rpack with separate resistors in a dual in-line package.



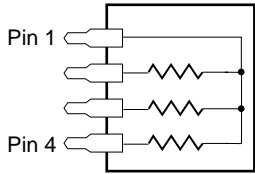
An example of an RP_DI IPL.DAT entry:

```
RP_DI,RPU4, "Respack di",2.20KO,7.00%,10,145,144,81,295,316,16,123
```

RP_SB

Token	ID	Description	Value	Tol	Pin 1	Pin 2	Pin n
RP_SB	, RPU2	,"Respack sb"	, 2.60KO	, 7.00%	, 0	, 245	, ...

RP_SB indicates an Rpack-based resistor in a single in-line package. Pin 1 is automatically the common node.



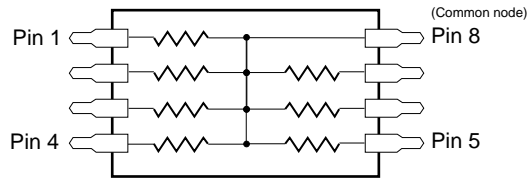
An example of an RP_SB IPL.DAT entry:

```
RP_SB,RPU2, "Respack sb",2.60KO,7.00%,0,245,144,81
```

RP_DB

Token	ID	Description	Value	Tol	Pin 1	Pin 2	Pin n
RP_DB	, RPU5	,"Respack db"	, 2.60KO	, 7.00%	, 1	, 2	, ...

RP_DB indicates an Rpack-based resistor in a dual in-line package.



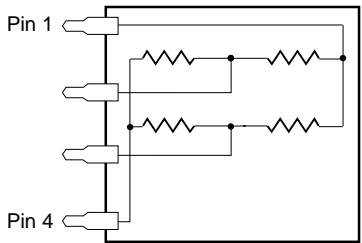
An example of an RP_DB IPL.DAT entry:

RP_DB,RPU5,"Respack db",2.60KO,7.00%,1,2,3,0,5,6,7,8

RP_ST

Token	ID	Description	Value 1	Value 2	Tol	Pin 1	Pin 2	Pin n
RP_ST	, RPU3	,"Respack db"	, 2.60KO	, 220.00KO	, 7.00%	, 0	, 145	, ...

RP_ST indicates an Rpack-terminated resistor in a single in-line package.



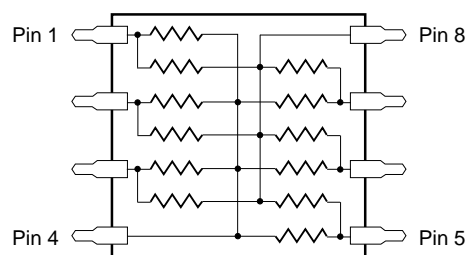
An example of an RP_ST IPL.DAT entry:

RP_ST,RPU3,"Respack st",2.60KO,220.00 O,7.00%,0,145,144,81

RP_DT

Token	ID	Description	Value 1	Value 2	Tol	Pin 1	Pin 2	Pin n
RP_DT	, RPU6	, "Respack dt"	, 4.60KO	, 120.00KO	, 7.00%	, 60	, 61	, ...

RP_DT indicates an Rpack-terminated resistor in a dual in-line package.



An example of an RP_DT IPL.DAT entry:

```
RP_DT,RPU6,"Respack dt",4.60KO,120.00KO,7.00%,60,61,62,0,64,65,66,5
```

D

Token	ID	Description	Pin Anode	Pin Cathode
D	, D1	, "Diode"	, 19	, 21

D creates a diode test. The first pin is always the anode; the second pin is the cathode. The default section is Diodes.

Z

Token	ID	Description	Value	Tol	Pin Anode	Pin Cathode
Z	, Z1	, "Z-Diode"	, 9.7V	, 15%	, 34	, 22

Z creates a Zener test. The unit in the value field has always to be expressed in V (Volts). Pin 1 is the anode; Pin 2 is the cathode. The default section is Zeners.

QN and QP

Token	ID	Description	Pin Base	Pin Emitter	Pin Collector
QN	, TR1	, "Transistor"	, 45	, 67	, 84
QP	, TR2	, "Transistor"	, 54	, 76	, 48

QN or QP create junction transistor tests. The first node is the Base; the second, the Emitter; the third, the Collector. The default section is Transistor.

GNPN and GPNP

Token	ID	Description	Min Gain	Ic	Pin Base	Pin Emitter	Pin Collector
GNPN	, TR3	, "	, 120Hfe	, 25 mA	, 45	, 67	, 84
GPNP	, TR4	, "	, 120Hfe	, 25 mA	, 54	, 76	, 48

GNPN or GPNP create beta tests. The Min Gain field specifies the threshold for a passing test. The Ic field specifies the collector current the beta is tested at. The default section is Trans.

U

Link Token	Token	ID	Description	Name	Pin 1	Pin n
... &	, U	, U1	, "unknown"	, CONN1	, 99	, ...

U—Unknown device—creates no test, just a component record for connector, CONN1, for example. The default section is determined by the link token.

Template Component Tokens

ATMPL

Link Token	Token	ID	Description	Name	Pin 1	Pin n
... &	, ATMPL	, U1	, "op_amp"	, LM342	, 101	, ...

ATMPL creates a test according to analog templates in the library. It uses the Name field to search the library. The default section is determined by the link token. See the Link Tokens table.

IC

Token	ID	Description	Name	Pin 1	Pin 2	Pin n
IC	, IC1	, " "	, 74ALS121A	, 56	, 57	, 59

IC indicates a Gray code template from GCTEMPL.LIB. The default section is Digital components.

DB

Token	ID	Description	Name	Pin 1	Pin 2	Pin n
DB	, IC2	, " "	, 74ALS121A	, 56	, 57	, 59

DB indicates a data bus. First the Gray code library is searched for a template of this name. If a Gray code template is not found, then the vector library is searched. The default section is Digital components.

VEC

Token	ID	Description	Name	Pin 1	Pin 2	Pin n
VEC	, IC3	, " "	, 74ALS121A	, 56	, 57	, 59

VEC indicates vector templates derived from an ASCII model. It accesses VRTEMPL.LIB. The default section is Digital components.

VIMG

Token	ID	Description	Name	Pin 1	Pin 2	Pin n
VIMG	, IC4	, " "	, 74ALS121A	, 56	, 57	, 59

VIMG indicates a vector image test derived from test developed with the Vector Editor or from IVL models such as those for Boundary Scan from VICTORY. It accesses VRSNAP.LIB.

IC, DB, VEC, and VIMG tokens create digital tests according to the library templates listed above. The pins cannot have multiple nodes concatenated with an &. The default section for the above tokens is Digital components.

VCLUST

Token	ID	Description	Name
VCLUST	, VC1	, "	, 5688VC

VCLUST creates a vector cluster (produced with SCAN, Victory, or Boundary Scan). A vector cluster consists of node numbers and with associated stimulus and measure. There are no nodes specified because vector clusters are hard-coded and cannot be modified. The default section is Digital.

WSCAN

Token	ID	Description	Name	<u>x</u> Inducer Number	Node List
W SCAN	,U1	,WaveScan test for pal	, PAL22V1Ø	, x 17	, 33, 17, 16, 15, 14

WSCAN indicates that the component is to be tested using WaveScan. The inducer number specifies which inducer to use for the component. You must type an x in front of the inducer number to distinguish the inducer number from the list of node numbers. The node list is a comma-separated list of nodes. The first node is for the first pin on the device; the second node is for the second pin, and so on up to the number of pins on the device. The length of the node list is up to 625 pins. If a pin is unconnected, then use the number 9999 to mark that pin. The default is 9999.

See the **MultiScan User's Guide** for additional information about WaveScan testing.

FSCAN

Token	ID	Description	Name	<u>x</u> Inducer Number	Node List
F SCAN	,U1	,FrameScan test for pal	, PAL22V1Ø	, x 17	, 33, 17, 16, 15, 14

FSCAN indicates that the component is to be tested using FrameScan. The inducer number specifies which inducer to use for the component. You must type an x in front of the inducer number to distinguish the inducer number from the list of node numbers. The node list is a comma-separated list of nodes. The first node is for the first pin on the device; the second node is for the second pin, and so on up to the number of pins on the device. The length of the node list is up to 625 pins. If a pin is unconnected, then use the number 9999 to mark that pin. The default is 9999.

See the **MultiScan User's Guide** for additional information about FrameScan testing.

FSPLUS

Token	ID	Description	Name	<u>x</u> Inducer Number	Node List
FS Plus	,fscanpls	,FrameScan Plus test	, P1	, x 10	, 231, 12, 30, 35, 23, 70, 56

FSPLUS indicates that the component is to be tested using FrameScan Plus. The sensor number specifies which sensor to use for the component. You must type an x in front of the sensor number to distinguish the sensor number from the list of node numbers. The node list is a comma-separated list of nodes. The first node is for the first pin on the device; the second node is for the second pin, and so on up to the number of pins on the device. The length of the node list is up to 625 pins. If a pin is unconnected, then use the number 9999 to mark that pin. The default is 9999.

See the **MultiScan User's Guide**, for additional information about FrameScan Plus test.

CSCAN

Token	ID	Description	Name	<u>x</u> Inducer Number	Node List
CSCAN	,cap_scan	,cap_scan test	, cap_scan	, x 101	, 1 & 447, 0 & 446

CSCAN indicates that the component is to be tested using CapScan. The sensor number specifies which sensor to use for the component. You must type an x in front of the sensor number to distinguish the sensor number from the list of node numbers. The node list is a comma-separated list of nodes. The first node is for the first pin on the device; the second node is for the second pin, and so on up to the number of pins on the device. The length of the node list is up to 625 pins. If a pin is unconnected, then use the number 9999 to mark that pin. The default is 9999.

See the **MultiScan User's Guide**, for additional information about CapScan test.

DSCAN

Token	ID	Description	Name	Node List
DSCAN	,U1	,DeltaScan test for pal	, PAL22V1Ø	, 33, 17, 16, 15, 14

DSCAN indicates that the component is to be tested using DeltaScan. The node list is a comma-separated list of nodes. The first node is for the first pin on the device; the second node is for the second pin, and so on up to the number of pins on the device. The length of the node list is 625 pins. If a pin is unconnected, then use the number 9999 to mark that pin. The default is 9999.

See the **MultiScan User's Guide**, for additional information about WaveScan testing.

DFP

Token	ID	Description	Name
DFP	,U1	,DFP for flash memory	, 28F256

DFP indicates that the component is to be tested using Digital Function Processor.

See the **Digital Function Processor User’s Guide** and the section, Digital Function Processor, in chapter 10 of this manual, for more information about DFP test.

Special Tokens

In addition to the previous tokens, there are two special tokens.

END terminates the IPL convert at the END token. It has the same effect as a comment to the end of the file.

Link Tokens

Link tokens consist of a keyword followed immediately by an ampersand, &. The ampersand links a component into a specific section rather than its default section. A regular token must follow a link token, as in

C&,R,r1,,4.7ko,5%,99,100

In this example, R1 is placed in the capacitor (C) section. Link tokens are explained in the following Link Tokens table.

Token	Link to Section
DCHG&	DISCHARGE
SW&	SWITCHES/POTS
L&	INDUCTORS
C&	CAPACITORS
R&	RESISTORS
POT&	SWITCHES/POTS
RP&	RPACKS
D&	DIODES
Q&	TRANSISTORS
Z&	ZENERS
ADOF&	ANALOG
ADON&	LINEAR
DIG&	DIGITAL
MISC&	MISCELLANEOUS

Key to Language Summary Table

The following amplifies and explains the following table, Summary of Input List Tokens and Fields. The IPL arguments constitute the language in the fields following the tokens.

Parameter abbreviations:

- ID—Component identification—R1 or C13, for example. The ID field is in the Component Properties portion of the Step Worksheet. Stored in the component ID in the ipl database. May be up to 8 characters.

- **DESC**—Description—any component-specific text of up to 64 characters, stored under Desc in the ipl database. The Desc field is in the Component Properties portion of the Step Worksheet. Should the parameter be an empty string, the Desc is filled with information from the VAL and TOL parameters, "Located at coord B-16," for example.
- **NAME**—Template Name. Becomes the Name field in the Component Properties portion of the Step Worksheet. Stored under Name in the IPL.DBF. May be up to 20 characters.
- **VAL1**—The first value, for example, 1.2KO
- **VAL2**—The second Value for components requiring two values, for example, 4700
- **TOL1**—The first tolerance in percent (Positive tolerance)
- **TOL2**—The second tolerance in percent for components requiring two tolerances. (Negative tolerance)

Unit Abbreviations, Modifiers, and Strings:

The unit abbreviation specifies the units allowed in the VAL1, VAL2 fields. The first character in the string represents the base unit as in V for Volt, A for Ampere, F for Farad, H for Henry or Hfe (as applicable), and O for Ohms. Unit modifiers are p for pico, n for nano, u for micro, m for milli and mega, and K for kilo. The strings in this column (Cont, for example) are used to fill Component Properties ID and Description fields in the absence of a supplied string through the ID or DESC parameters.

The & is used to group more than one node per pin as in the CnSC example below.

- **ShOp**—Short and opens, multiple pins, single nodes—10 - 112, 150 - 300
- **CnSc**—Continuities and Special cases, multiple pins, multiple nodes—12 & 13 & 16, 151 & 299
- **TWO**—Two pins, multiple nodes, 5,7 or 5 & 6, 7 & 8
- **TRI**—Three pins, multiple nodes—2,4,6 or 2&3, 4&5, 6&7
- **ATPL**—Analog template multiple pins multiple nodes—1,2,3,4,5,6,7,8,... or 1&2,3,4,5,6,7,8&9,..
- **DTPL**—Digital template multiple pins single nodes—1,2,3,4,5,6,7,8,9,10,...
- **NONE**—No pin information

Syntax example:

An example of syntax for a capacitor:

- token: C
- 5 parameters: ID,DESC,VAL1,TOL1,TOL2
- Default test section: CAPACITORS
- Allowable unit and multipliers: F (Farad), m (Milli), u (micro), n (Nano) and p (Pico).
- Capacitor token allows for two pins. Each pin can have multiple nodes: TWO
- Percent signs are optional.

C ID,DESC,VAL1,TOL1,TOL2 TWO munpF CAPS

A 4.7uF capacitor with a positive tolerance of 20% and a negative tolerance of 10% which is called C18 on the schematic and is located at coordinate D-5 on the board grid with the nodes 200 and 210 connected may therefore be represented as:

C,C18,"Located at coord D-5 (the little yellow one)",4.7uf,20%,10%,200,210

The following table lists the tokens and succeeding fields in the record.

Token	Parameters	Node Types	Pin Seq.	Units/String	Default Section
HEAD	UNUSED,UNUSED,NAME	NONE	—		HEADER
GND	UNUSED,UNUSED	ATPL	—		HEADER
IPL	DESC	NONE	—		HEADER
APC	NOPARAM	ShOp	—		DISCHR
DCHG	ID,DESC	TWO	P1–P2	" "	DISCHR
CONT	NOPARAM	CnSc	—		CONT
J	ID,DESC	CnSc	—		JUMP
SC	NOPARAM	CnSc	—		IGNORE
SH	NOPARAM	ShOp	—		SHORTS
OPN	NOPARAM	ShOp	—		OPN
L	ID,DESC,VAL1,TOL1	TWO	—	Hmun	INDUCT
C	ID,DESC,VAL1,TOL1,TOL2	TWO	—	Fmunp	CAPS
R	ID,DESC,VAL1,TOL1	TWO	—	KMO	RESIST
POT	ID,DESC,VAL1,TOL1	TRI	A,B,C*	KMO	POTS
REO	ID,DESC,VAL1,TOL1	TWO	—	KMO	POTS
RP_SI	ID,DESC,VAL1,TOL1	ATPL	P1–Pn	KMO	RPACK
RP_SB	ID,DESC,VAL1,TOL1	ATPL	P1–Pn	KMO	RPACK
RP_ST	ID,DESC,VAL1,VAL2,TOL1	ATPL	P1–Pn	KMO	RPACK
RP_DI	ID,DESC,VAL1,TOL1	ATPL	P1–Pn	KMO	RPACK
RP_DB	ID,DESC,VAL1,TOL1	ATPL	P1–Pn	KMO	RPACK
RP_DT	ID,DESC,VAL1,VAL2,TOL1	ATPL	P1–Pn	KMO	RPACK
D	ID,DESC	TWO	A,C		DIODES
QN	ID,DESC	TRI	B,E,C		TRANS
QP	ID,DESC	TRI	B,E,C		TRANS
GNPN	ID,DESC,VAL1,VAL2	TRI	B,E,C	munH/mumA	TRANS
GNPN	ID,DESC,VAL1,VAL2	TRI	B,E,C	munH/mumA	TRANS
Z	ID,DESC,VAL1,TOL1	TWO	(A,C)	V	ZENERS
PWR5†	ID,DESC,	TWO	(M,R)**	V	POWER
PWR5_5†	ID,DESC,VAL1,TOL1	ATPL	(M,R)*	V	POWER
PWRA†	ID,DESC,VAL1,TOL1	TWO	(M,R)**	V	POWER
PWRA_F†	ID,DESC,VAL1,TOL1	TWO	(M,R)**	V	POWER
PWRB†	ID,DESC,VAL1,TOL1	TWO	(M,R)**	V	POWER
PWRB_F†	ID,DESC,VAL1,TOL1	TWO	(M,R)**	V	POWER
PB	ID,DESC,VAL1	ATPL	P1–Pn	V	POWER
PBUS	ID,DESC,VAL1,TOL1	TWO	(M,R)*	V	POWER
IC	ID,DESC,NAME	DTPL	P1–Pn		DIGITAL
DB	ID,DESC,NAME	DTPL	P1–Pn		DIGITAL
VEC	ID,DESC,NAME	DTPL	P1–Pn		DIGITAL
VIMG	ID,DESC,NAME	DTPL	P1–Pn		DIGITAL
U	ID,DESC,NAME	ATPL	P1–Pn		PER LINK TOKEN
ATMPL	ID,DESC,NAME	ATPL	P1–Pn		PER LINK TOKEN
TRAIL	UNUSED,UNUSED,NAME	NONE	—		TRAILER
VCLUST	ID,DESC,NAME	NONE	—		DIGITAL
WSCAN	ID,DESC,NAME,INDUCER	DTPL	P1–Pn		FRAME/WAVESCAN
DSCAN	ID,DESC,NAME	DTPL	P1–Pn		DELTASCAN
FSCAN	ID,DESC,NAME,INDUCER	DTPL	P1–Pn		FRAME/WAVESCAN
CSCAN	ID,DESC,NAME,INDUCER	DTPL	P1–Pn		CAPSCAN
DFP	ID,DESC,NAME	NONE	P1–Pn		DIGITAL

*Pins A and C are the ends of the potentiometer; Pin B is the wiper.

**M = Measure; R = Reference

†The PGEN.CFG variable PS ... is significant to these tokens.

Sample Input List

The following is an example of an input list (IPL.LST) generated using the Text IPL List command from the Reports menu. The list includes every type of legal syntax and is accepted by Pgen with no errors.

```

HEAD,,,HEADER1
IPL,"GRAPHICS BOARD REV.1" /*gets put into HEADER/PGMVARS descr. field*/
GND,STEP1,0,0
DCHG,"vcc","",54,40
DCHG,"C 10 ","",1U 250V",32&40&54&62&68,76
DCHG,"C 11","",75,32&40&54&62&68
DCHG,"C 12","",72,32&40&54&62&68
APC,0-15,31-345
J,"S2",42,39
J,"R505",93,61
J,"R949",340,340
J,"R951",128,129
J,"R952",194,195
J,"R953",126,127
J,"R956",284,285
CONT,39&42&43,54&55&102&117&125&318,40, 61&93,66&69&71,67&227&229,68&70\
,113&114,115&116,126&127,128&129,194&195,226&228,230&231&354&355\
,284&285,319&321,320&322,325&326&352&353
SC,33&84&53,46&56&84,44&58&84,37&49&84,17&25&28&29&81&86&32&40&54&62&68&84
SH,20-355 /*should test as NO shorts within this group*/
OPN,300-322 /*should test as NO opens within this group*/
L,"T1","transformer",100.00uH,20.00%,71,69
L,"L2","2.7uh",2.70uH,10.00%,327,324
L,"T2","1R",100.00uH,40.00%,230,355
L,"fl1","tla471",4.00 U,0.00%,113,230
L,"fl4",".022uf + inductors",10.00 H,20.00%,54&125,67&112
L,"fl3",".022uf + inductors",10.00 H,20.00%,43,112
C,"C230","1U +20% -20%",1 uF,22%,22%,275,35
C,"C20",".47U +20% -20%",0.47 uF,22%,22%,371,131
C,"vcc/gnd","power capacitance",1.00mF,50.00%,0.00%,54&117&125,40
C,"C23","CAP",3.30uF,20.00%,20.00%,64,54&55&102&117&125
C,"C20","CAP",100.00nF,10.00%,10.00%,54&117&125,71
C,"C26","CAP",100.00nF,10.00%,10.00%,67,117&125
C,"C22","CAP",470.00nF,20.00%,20.00%,65,54&55&102&117&125
C,"C6","CAP",10.00uF,20.00%,20.00%,54&55&102&117&125,63
C,"C1028","CAP",4.70nF,20.00%,20.00%,54&117&125,72
C,"C8","100P +30% -30%",100 pF,44.3%,37.3%,47,46
C,"C1","680P +25% -25%",680 pF,30.6%,30.4%,54,29
C,"C5",".022U +20% -20%",0.022 uF,22%,22%,276,35
C,"C201",".01U +20% -20%",0.01 uF,23.1%,22%,56,131
C,"C23",".01U +20% -20%",0.01 uF,23.1%,22%,43,131
C,"C268",".001U +20% -20%",0.001 uF,25.4%,23.3%,43,131
C,"C212","560P +30% -30%",560 pF,35.8%,35.5%,45,35
C,"C13","2.5U +20% -20%",2.5 uF,22%,22%,39,35
C,"C248","2.5U +20% -20%",2.5 uF,22%,22%,38,35
C,"CFILTER","Check",1.2 uF,32%,32%,150,35
C,"CVCC","MANY TO CHECK",1 uF,32%,32%,147,131
C,"CVCCA","Check",7.2 uF,32%,32%,40,35
C,"CVCCB","Check",7.2 uF,32%,32%,41,35
C,"CVCC_BUF","C4,C216",7.4 uF,32%,32%,42,35

```

```

C, "CVCC_EAG", "C11,C202-C210,C224-C229,C233-C239", 1.1 uF, 32%, 32%, 21, \
131
C, "CVCC_ECH", "Check", 10.6 uF, 32%, 32%, 147, 131
C, "CVCC_FAL", "C16,C200,C217-C221,C231,C239-C242", 1.05 uF, 32%, 32%, 67, 35
C, "CVCC_FIL", "C9,C18,C19,C242,C257,C263", 380 nF, 32%, 32%, 36, 35
C, "CVCC_FPA", "C10,C244", 1 uF, 32%, 32%, 179, 131
C, "CVCC_SRC", "C22,C269", 650 nF, 32%, 32%, 120, 35
C, "CVCC_UNI", "C17,C256", 7.1 uF, 32%, 32%, 37, 35
C, "C_U9RET", "Check", 650 nF, 32%, 32%, 120, 35
C, "C 1b", "5 wire test", 1.00uF, 2.00%, 2.00%, 68&84, 73&89
R, "R961", "RES", 100.00 O, 10.00%, 114, 113
R, "R10", "RES", 10K, 8%, 40, 22
R, "R17", "RES", 10K, 8%, 40, 100
R, "R18", "RES", 10K, 8%, 40, 62
R, "R20", "RES", 10K, 8%, 40, 209
R, "R21", "RES", 10K, 8%, 100, 54&55&102&117&125
R, "R23", "RES", 10K, 8%, 40, 57
R, "R24", "RES", 10K, 8%, 40, 97
R, "R25", "RES", 10K, 8%, 40, 59
R, "R27", "RES", 10K, 8%, 40&211&327, 333
R, "R31", "RES", 10K, 8%, 40&211&327, 334
R, "R32", "RES", 10K, 8%, 40&211&327, 335
R, "R33", "RES", 10K, 8%, 40, 210
R, "R39", "RES", 10K, 8%, 40&211&327, 207
R, "R40", "RES", 10K, 8%, 54&55&102&117&125, 119
R, "R42", "RES", 10K, 8%, 54&55&102&117&125, 174
R, "R44", "RES", 10K, 8%, 54&55&102&117&125, 173
R, "R46", "RES", 10K, 8%, 40, 58
R, "R47", "RES", 10K, 8%, 40, 98
R, "R49", "RES", 10K, 8%, 54&55&102&117&125, 118
R, "R54", "RES", 10K, 8%, 177, 29
R, "R55", "RES", 10K, 8%, 29, 40&211&327
R, "R56", "RES", 10K, 8%, 30, 40&211&327
R, "R57", "RES", 10K, 8%, 17, 30
R, "R60", "RES", 10K, 8%, 40&211&327, 37
R, "R62", "RES", 10K, 8%, 40&211&327, 31
R, "R64", "RES", 10K, 8%, 40&211&327, 200
R, "R65", "RES", 10K, 8%, 40&211&327, 199
R, "R66", "RES", 10K, 8%, 40&211&327, 205
R, "R69", "RES", 10K, 8%, 40, 140
R, "R70", "RES", 10K, 8%, 40, 21
R, "R9", "RES", 10K, 8%, 54&55&102&117&125, 206
R, "R945", "RES", 10K, 8%, 40, 26
R, "R947", "RES", 10K, 8%, 327, 28
R, "R958", "RES", 10K, 8%, 119, 40
R, "R960", "RES", 10K, 8%, 40, 332
R, "R969", "RES", 10K, 8%, 40, 344
MISC&, R, "R222", "failed test", 1.00MO, 10.00%, 94, 68
RP&, R, "RPACK1", "Position A2", 1.00KO, 15.00%, 34, 45
POT&, R, "Pot1", "11k", 11.00KO, 10.00%, 68&84, 43&59
REO, "Rheo-1", "10k", 10.00KO, 10.00%, 41, 68
GNPN, "Q1b", "NPN gain test", 120.00 H, 0.50 A, 9, 1, 68
GPNP, "Q1b", "NPN gain test", 100.00 H, 0.50 A, 9, 1, 68
Z, "VR1", "1N4733A 5 Volts", 5.00 V, 20%, 84, 12
POT&, ATMP, "crt_msg1", " ", 84, 33, 245
POT&, ATMP, "SW1", "Closure pulls down stim voltage", switch, 109, 110, 111
MISC&, ATMP, "crt_msg", " ", 234, 435, 22
ADOF&, ATMP, "Zener1", "Zener in TVSI", TVSI Zener, 84, 12
ADOF&, ATMP, "Cap1", "RMS TISV capacitor", C 6 - TISV/RMS, 68, 74

```



```

ADON&,ATMPL,"Measure ","Retest Vcc at Node 21.",Node 21 test,21
RP_SI, "RP200","PACK",4700 0,6%,34,803,788,836,48,772,787,771,594
RP_SB, "RP201","PACK",4700 0,6%,34,803,788,836,48,772,787,771,594
RP_ST, "RP202","PACK",4700 0,3300 0,6%,34,803,788,836,48,772,787,771,594
RP_DI, "RP203","PACK",4700 0,6%,34,803,788,836,48,772,787,771,594
RP_DB, "RP204","PACK",4700 0,6%,34,803,788,836,48,772,787,771,594
RP_DT, "RP205","PACK",4700 0,3300 0,6%,34,803,788,836,48,772,787,771,594
D,"CR6","DIODE",29,125
D,"CR5","DIODE",30,54
QN,Q1,"Q1 NPN Junction Test",51,18,20
QN,Q3,"Q3 NPN Junction Test",32,35,53
ADOF&,ATMPL,K1,"RELAY",RELAY,67,29,243,28,9999,9999,30,244,31,53
R&,ATMPL,SW1,"SWITCH",SWITCH,131,131,131,131,644,659,660,675
L&,ATMPL,T1,"TRANS",TRANS,76,147,76,42
L&,ATMPL,T2,"TRANS",TRANS,147,147,21,179
L&,ATMPL,T3,"TRANS",TRANS,58,59,291,292
L&,ATMPL,T4,"TRANS",TRANS,259,260,38,39
L&,ATMPL,T5,"TRANS",TRANS,147,147,37,67
L&,ATMPL,T6,"TRANS",TRANS,37,37,41,40
L&,ATMPL,T7,"TRANS",TRANS,70,70,26,36
L&,ATMPL,T8,"TRANS",TRANS,157,156,72,71
L&,ATMPL,T9,"TRANS",TRANS,170,171,73,74
PWR5,"5V","5 VOLTS",40,22
PWR5_5,"5V","5 VOLTS",5.0 V,10%,40,22
PWRA,"15V","15 VOLTS",15.0 V,10%,32,22
PWRA_F,"12V B","12 VOLTS B",15.0 V,10%,52,22
PWRB,"15V","-15 VOLTS",15.0 V,10%,73,22
PWRB_F,"12V B","-12 VOLTS B",15.0 V,10%,62,22
ADOF&,U,J1,"CONN8",CONN8,74,71,72,73,19,19
VIMG,"U8","82C585",82C585,9999,9999,54,9999,327,64,65,216,63,213,48,155,201\
,324,51,323,101,50,72,49,37,227,31,214,40,9999,9999,9999,222,9999\
,9999,9999,9999,9999,9999,9999,9999,9999,9999,9999,9999,9999\
,9999,9999,9999,320,322,218,9999,217,212,69,71,9999,9999,215,\
,114,113,9999,229

IC,"U10","FF",74273,122,237,178,179,238,27,180,181,54,232,233,182,183,234\
,235,184,185,236,40
DSCAN,"U10-1","FF",74273,122,237,178,179,238,27,180,181,54,232,233,182,183\
,234,235,184,185,236,40
IC,"U14","",74f245,107,301,300,299,298,297,296,295,294,54,310,311\
,312,313,314,315,316,317,121,40
WSCAN,"U14-1","",74f245,X3,107,301,300,299,298,297,296,295,294,54,310,311\
,312,313,314,315,316,317,121,40
IC,"U15","",74f245,52,9999,147,146,145,144,143,142,141,54,148,149\
,150,151,152,153,154,9999,208,40
FSCAN,"U15-1","",74f245,X5,52,9999,147,146,145,144,143,142,141,54,148,149\
,150,151,152,153,154,9999,208,40
FSPLUS,fscanpls,"Framescan plus",P1,x10,231,12,30,35,23,89,70,56
CSCAN,cap_scan,"cap_scan",cap_scan,x101,1&447,0&446
IC,"U16","",74f245,52,251,250,249,248,247,246,245,244,54,264,265,266,267,268\
,269,270,271,208,40
IC,"U17","",74f245,52,259,258,257,256,255,254,253,252,54,272,273,274,275,276\
,277,278,279,208,40
IC,"U18","74f245",74f245,52,9999,263,262,261,260,54,280,281,282,283,9999\
,9999,9999,9999,208,40
IC,"U19","",74f245,107,293,292,291,290,289,288,287,286,54,302,303,304,305\
,306,307,308,309,208,40
IC,"U20","SERIAL PROM",9346,242,241,243,119,54,9999,9999,40
VEC,"U3","",V30_LG,75,74,73,92,40,40,54,54,54,91,90,89,88,9999,87,86,85,84\

```

```

,83,82,81,80,79,78,77,9999,96,95,93,54,9999,54,54,99,98,9999,57,58\
,59,62,9999,9999,23,97,94,100,76,9999
VEC,"U4","GAL",iowr_pal,9999,93,47,46,350,175,336,9999,9999,54,9999,38,7\
,347,341,9999,9999,9999,9999,9999,344,107,9999,40
VEC,"U2","ram",RAM,173,172,170,168,163,162,161,160,159,158,157,156,178,179\
,180,54,181,182,183,184,185,239,166,202,167,165,164,169,225,211,171\
,40
VEC,"U5","ram",RAM,173,172,170,168,163,162,161,160,159,158,157,156,186,187\
,188,54,189,190,191,192,193,240,166,202,167,165,164,169,225,211,171\
,40
VEC,"U7","NAND TREE",581,54,54,344,123,338,346,345,343,54,342,339\
,317,316,40,315,314,313,312,117,311,310,309,308,307,306,54,40,305,304\
,303,302,284,350,123,175,56,194,121,337,130,128,52,47,347,340,196,197\
,264,265,61,54,54,40,266,267,268,269,270,271,54,272,273,274,275,40\
,276,277,278,279,54,280,281,282,283,148,149,150,40,54,151,152,153,154\
,347,54,126,347,41,347,54,348,40,49,72,50,101,323,54,51,324,201,155\
,48,40,54,54,351,44,221,36,204,118,119,99,93,98,95,96,57,58,59,120,97\
,100,77,78,79,80,81,40,54,82,83,84,85,86,87,88,89,40,54,90,91,92,73\
,74,75,76,174,173,172,171,170,169,54,54,40,168,167,166,165,164,163\
,162,54,40,161,160,159,158,157,156,349,332,224,202,225,176,177,193,45\
,54,40,192,191,190,189,188,187,186,185,184,183,182,54,181,180,179,178\
,200,199,53,47,47,47,123,123,40,8
VEC,"U9","RAMPAL",RAMPAL,171,173,174,176,177,207,333,9999,54,335,240,28,9999\
,331,330,329,328,239,40
VEC,"U11","EEPAL",EEPAL,225,332,156,157,158,159,167,168,178,54,20,232,225\
,9999,241,242,243,9999,169,40
VEC,"U12","PAL",DMPAL,341,104,105,106,103,223,237,238,27,54,140,122,220,26\
,108,46,111,110,109,40,232
VEC,"U13","16L8",IRQPAL,22,22,123,22,22,21,233,234,235,54,236,134,139,133\
,137,132,136,138,135,40,232,122
VIMG,"SPARE2","",PARFXUB2,40,9999,9999,131,9999,9999,9999,54,9999,54,9999\
,38,9999,9999,9999,9999,9999,9999,130,9999,9999,40
VEC,"Y1",OSC,OSC,209,9999,9999,9999,24
VEC,"Y2",OSC,OSC,210,9999,9999,9999,25
VCLUST,"u1",,fun1
DFP,"U23",,fun2
TRAIL,, ,TRAILER1
END

```

Creating Test Programs with C.1 Input Lists

With the C1TODX function you can automatically transvert a C.1 input list to a DX format input list, retaining the ability to specify guard nodes, longhand tests, 4- or 6-wire tests, and the usage of E- and F-pole designators. The transvert process translates as well as converts your C.1 IPL.DAT to a DX IPL.DAT and then generates the test program, ICT.TST, if you so choose. A DX format input list can be used to create programs for DX or later revision system software.

To create a test program with a C.1 input list

- 1 Select an existing board directory (Files/Select).

It is assumed that you have one IPL.DAT per directory.

- 2 Exit to the DOS shell from the Utility menu.

- 3 At the prompt, call the C1 to DX transverter by typing

c1todx <FILENAME>

where the optional <FILENAME> is the name of the file you wish to transvert. If the file name is omitted, it is IPL.DAT by default.

After you have invoked C1TODX (one word, no spaces), if there are any analog templates

(ANnnnnnn.TPL) from prior runs in your local directory, you will be asked if you want to delete the analog templates prior to running the transverter. It is advisable to delete these templates because additional templates are created and put in the board directory each time you run the transverter, thus reducing your disk space.

4 To delete the templates type **y** and press Return.

5 If you want to skip deletion and continue with the transversion process, type **n** and press Return.

If you have files that contain guard nodes, longhand tests, 4- or 6-wire tests, or E- and F-pole designators, the transverter produces analog template output files in addition to the usual output files. These binary files take the form ANnnnnnn.TPL, where nnnnnn is a 6-digit random number.

After the conversion function has executed, the following prompt appears:

A new D1 IPL.DAT was created, do you want to run PGEN to build and generate a test program? [y or n]

IMPORTANT: If the file you selected for transversion was already in the Dx format, you will not see the message above.

6 After you have typed y or n, wait until program generation is finished and then type **exit** to return to the 18xx interface.

Adding Cnames to DX Programs. The C1TODX function also allows you to put Cname as part of the device ID, as long as you escape that string by enclosing the Cname with the following;

```
/*: */
```

C1TODX knows how to parse the CNAME correctly and not view it just as a comment.

Given this C1 token line,

```
C,c1/*: The Cname is: Cap1.00uF
:*/,1.00uF,5.00%,5.00%,30,31
```

the resulting D1 line is

```
C,c1, " The Cname is: Cap1.00uF
",1.00uF,5.00%,5.00%,30,31
```

Batch Files

The following are the batch files which drive the C1TODX process. They are provided for your information.

```
@echo off
rem
rem 10/2/92
rem This batch command DELTMPL.BAT is called from the program C1TODX
rem to delete all the local analog templates from previous run.
rem This batch also prints out the template names being deleted
rem for user's info.echo.
echo About to delete ALL local analog templates...
if exist an*.tpl goto yep
echo.
echo No analog templates exist!
echo.
goto end
```

```

:yep
echo.
for %%i in (an*.tpl) do echo Deleting %%i
del an*.tpl
echo.
echo ALL local analog templates DELETED.
echo.
:end
*****

@echo off
rem
rem 10/2/92
rem This batch command DOPG.BAT is called from the program C1TODX
rem to run PGEN -n , -b, and -g to generate test programs
rem for the newly created D1 IPL.DAT, result of running
rem the trnsverter program C1TODX through the C1 IPL file.
rem
echo.
echo 1.  Cleaning up (removing old files):18xx /P -n
18xx /P -n
echo.
echo 2.  Building component database:18xx /P -b
18xx /P -b
echo.
echo 3.  Generating test program:  18xx /P -g
18xx /P -g
echo.

```

Input List Examples

Below are examples of a C.1 input list before transversion and the same input list after transversion.

C.1 INPUTLIST

```
GND,FLAGS, 0
C,c1,1.00uF,5.00%,5.00%,30,31
C,c2,1.00uF,5.00%,5.00%,33,34,35
C,rc1,1.00uF,5.00%,5.00%,36,37,38
C:4,c3-4,1.00uF,5.00%,5.00%,40,60,41,61,42,62
C:6,c3-6,1.00uF,5.00%,5.00%,50,70,51,71,52,72
C&,R,rc1,10.00KO,5.00%,80,81,82
R,R1,1000.00 O,5.00%,90,91,92
R:4,R1-4,1000.00 O,5.00%,130,160,131,161
R:6,R1-6,1000.00 O,5.00%,230,260,231,261,232,262
C&,TVSI,TVSI,95.00mV,105.00mV,DC,10.00mA,DC,330,331
C&,TVSI,TVSI-G,95.00mV,105.00mV,DC,10.00mA,DC,430,431,432
C&,TISV,TISV,95.00mA,105.00mA,DC,10.00 V,DC,530,531
C&,TISV,TISV-G,95.00mA,105.00mA,DC,10.00 V,DC,630,631,632
```

Dx Inputlist after transversion

```
/* C1TODX - IPL-Version D2.0*/
/*-----*/
/* This is the output IPL.DAT after running*/
/* the transverter C1TODX. It's in Dx format.*/
/* The version date and time of this file is:*/
/*
   Mon Sep 28 14:51:36 1992
*/
/* If you don't see this comment header in other*/
/* IPL.DAT files, it's best to run them*/
/* through the C1TODX transverter.          */
/*-----*/
INPUTLIST
GND, FLAGS, , 0
C, C1, , 1.00UF,5.00%,5.00%,30,31
C&, ATMPL, C2, "1.00UF,5.00%,5.00%", AN000041.TPL ,33,34
C&, ATMPL, RC1, "1.00UF,5.00%,5.00%", AN018467.TPL ,36,37
C&, ATMPL, C3-4, "1.00UF,5.00%,5.00%", AN006334.TPL ,40,60
C&, ATMPL, C3-6, "1.00UF,5.00%,5.00%", AN026500.TPL ,50&70,51&71
C&, ATMPL, RC1, "10.00KO,5.00%", AN019169.TPL ,80,81
R&, ATMPL, R1, "1000.00 O,5.00%", AN015724.TPL ,90,91
R&, ATMPL, R1-4, "1000.00 O,5.00%", AN011478.TPL ,130,160
R&, ATMPL, R1-6, "1000.00 O,5.00%", AN029358.TPL ,230&260,231&261
C&, ATMPL, TVSI, "95.00MV,105.00MV", AN026962.TPL ,330,331
C&, ATMPL, TVSI-G, "95.00MV,105.00MV", AN024464.TPL ,430,431
C&, ATMPL, TISV, "95.00MA,105.00MA", AN005705.TPL ,530,531
C&, ATMPL, TISV-G, "95.00MA,105.00MA", AN028145.TPL ,630,631
```

▼▼▼

7 PROGRAM GENERATOR TOOLS

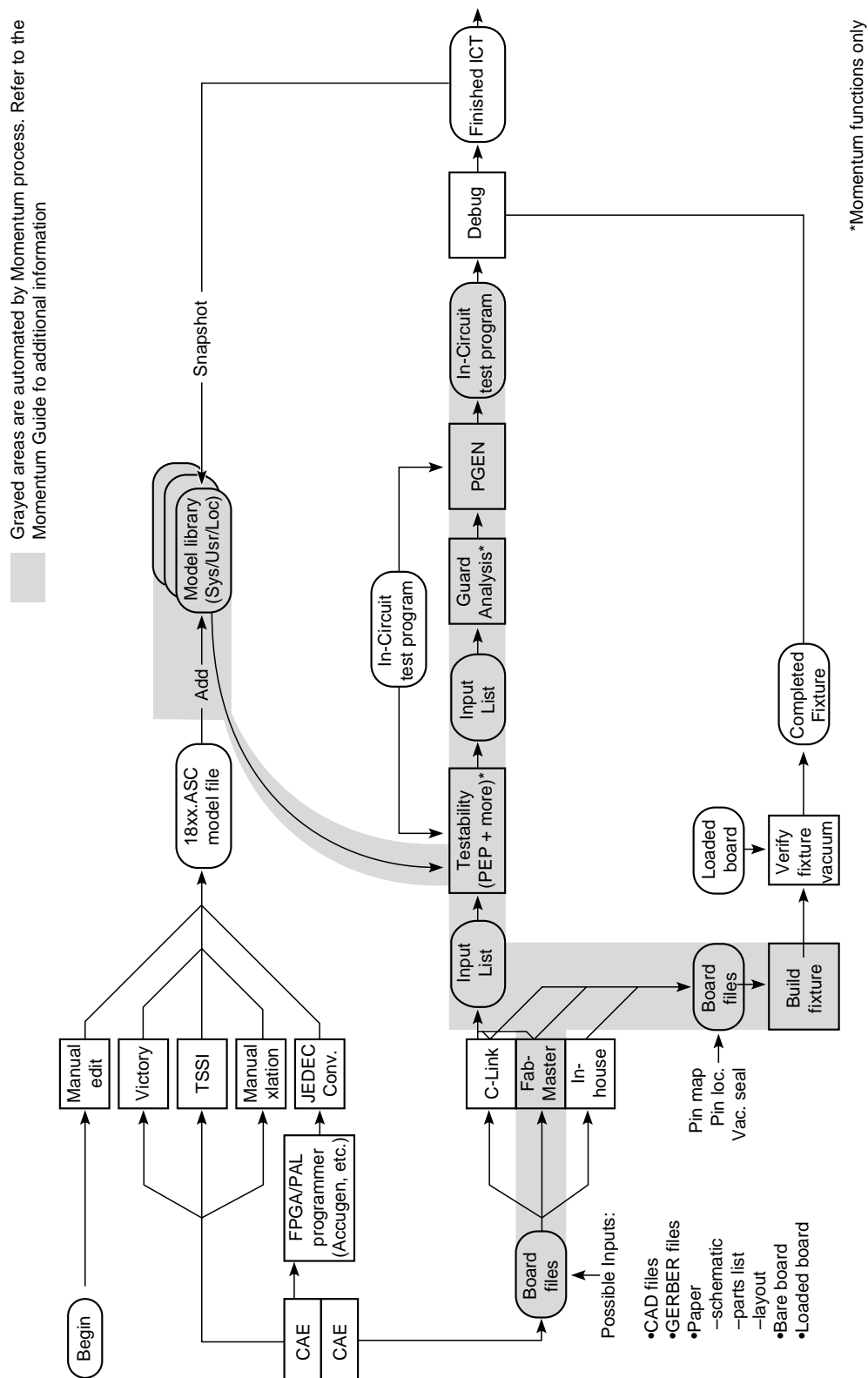
Chapter 7 discusses the various tools available in the Pgen menu.

The 18xx software system of Pgen (program generator) menus allows you to build and generate programs automatically, learn the board interconnects, and create a number of reports. The program generator uses both an input list and test templates of analog and digital components from various libraries to create a test program.

This chapter describes what the various Pgen functions are that affect program generation, development, and debug procedures. Depending on the inputs, program generation can be automatic or manual; you can add components in batch mode or individually.

See the **Z1800-Series Board Test Tutorial** for detailed information about generating a program manually.

For an overview of the entire program generation process which includes a number of board test approaches see the following page.



Using PGEN.CFG to Reassign Default Values

If your program has special requirements for test which do not fall within the system software's standard default parameters, you can modify these defaults before running Pgen. To make these changes, you must create or modify the PGEN.CFG files in the \PGT directory or board directory using the editor of your choice.

PGEN.CFG Hierarchy of Influence

The PGEN.CFG file can be in \PGT, the board directory, or both. The location of the PGEN.CFG file determines which parameters and values are applied in particular test situations. The hierarchy is as follows:

1. When Pgen is run on a program without a PGEN.CFG file, it uses the internally programmed defaults to assign parameters and values.
2. Then Pgen looks in \PGT for a PGEN.CFG file. If one exists, any values in it override the defaults.
3. Finally Pgen looks in the current board directory for a PGEN.CFG file. If one exists, any values there override both the \PGT and the defaults.

For example, if you want to test all diodes at .6 volts (the default is .7), set up that value in the PGEN.CFG file in \PGT to override the internally generated default. However, if you have one board on which you want to test diodes at .12 volts, set up that value in the PGEN.CFG file in the board directory where it will override both the system defaults and the value established in the file in \PGT.

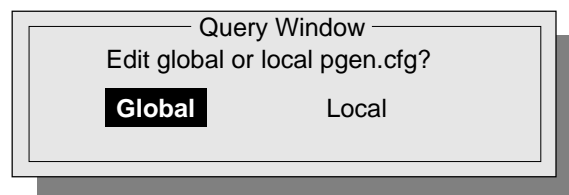
How to Edit PGEN.CFG

Use the PGEN.CFG worksheet, which takes the place of a text editor, to modify Pgen configuration variables.

Note that the text file, PGEN.CFG, still exists. If you need to directly edit the PGEN.CFG text file, you can. See the end of this section for information about modifying PGEN.CFG using a text editor.

To edit PGEN.CFG ,

1. Select Edit PGEN.CFG from the Pgen menu.
A query window appears.

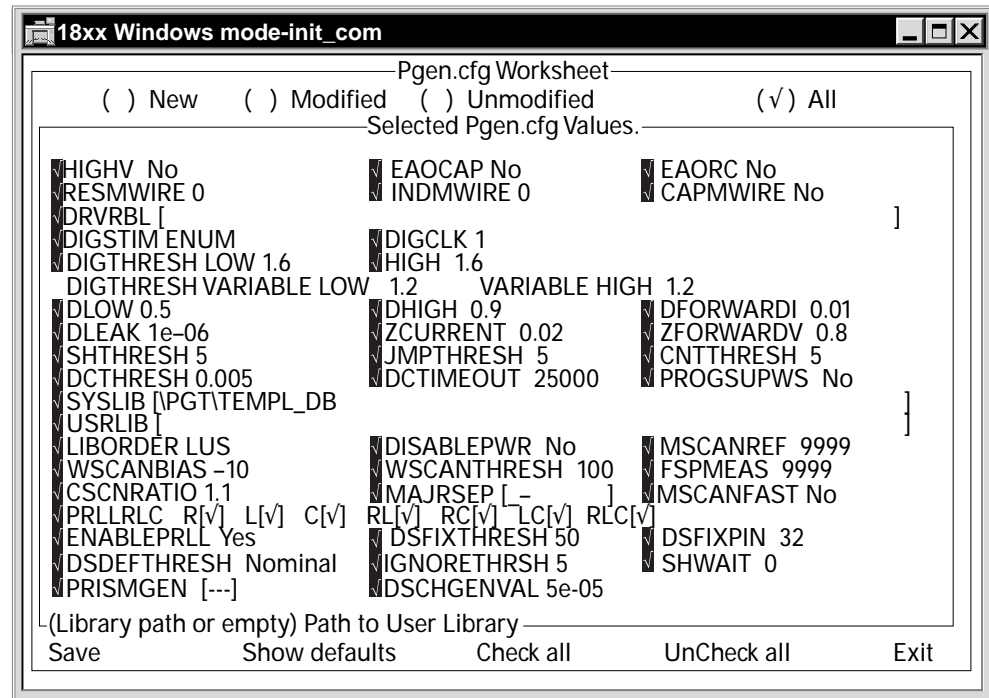


2. Select Local to edit the PGEN.CFG file in the current program directory or select Global to edit a PGEN.CFG in \PGT, and press Enter.

Once you make your selection, the PGEN.CFG worksheet appears. The changes you make in either Global or Local will affect the current program.

The worksheet reads data from the PGEN.CFG file, presents that data in the worksheet, allows you to edit that data, and writes the data back to the PGEN.CFG file when you choose Save.

The Pgen Config Worksheet contains three main parts: Viewing mode selections, Variable View window, and buttons.



Viewing Modes. The Viewing Modes determine which variables are displayed in the Variable View window. These choices are radio-type controls. To deselect the previously selected controls, click the check mark.

Selecting **New** displays all Pgen Config variables which are new to the 1800-Series system software since the last time the Pgen Config Editor was used and Save was selected. This functionality provides a mechanism that displays new variables and allows you to decide if you want to use them.

A token (CFGREV) in the PGEN.CFG file determines which variables are new. When you choose Save for the PGEN.CFG file you are editing, the CFGREV variable is updated to the current revision expected by the 1800-Series software. When the CFGREV in the PGEN.CFG file matches the revision expected by the software and you choose the New Viewing Mode, a message is displayed in the Variable View window, "No Data is selected." The message reports that no new variables have been found.

The **Modified** viewing mode lists in the Variable View window all the Pgen Config Variables which have been modified. A modified Pgen Config variable is one which is not at its default state.

The **Unmodified** Viewing Mode lists in the Variable View window all of the Pgen Config variables which have not been modified from their default state.

The **All** Viewing Mode lists in the Variable View window all of the Pgen Config variables, including New, Modified, and Unmodified.

Variable View Window. The Pgen Config variables are displayed and can be edited in Variable View. Each variable has a field just to the left of it. This field can either be checked or left blank. A checked field identifies a variable which is written to the PGEN.CFG file when Save is

selected. Variables without a check mark will not be written to the PGEN.CFG file.

Buttons. When you choose **Save**, all of the Pgen Config variables which are checked will be written to the PGEN.CFG file currently being edited—either the Global Pgen Config file or a Local one.

Selecting **Show Defaults** displays each Pgen Config variable and its default value in the Variable View window. The variables are not editable for this display. Show Defaults quickly displays the default value of any Pgen Config variable.

The **Check All** button applies to the Pgen Config variables which are currently displayed in the Variable View. For example, if you choose New as the Viewing Mode and Check All is selected, only the new variables are checked. If you choose All as the Viewing Mode and Check All is selected, then all Pgen Config variables are checked.

Just like Check All above, **UnCheck All** applies only to the Pgen Config variables which are currently displayed in the Variable View. For example, if you choose Modified as the Viewing Mode and UnCheck All is selected, the check marks for all of the unmodified Pgen Config variables are removed.

Choose **Exit** to leave the Pgen Config editor. If you have made edits but have not saved them, you will be asked if you want to save the changes.

Routing PGEN.CFG Changes to Except.lst

A record of the changes you make to PGEN.CFG can be routed to the EXCEPT.LST file. This record is a convenient way to see the changes you have made for the final file configuration; it displays final conditions when multiple PGEN.CFG files are used. It also enables you to check the system defaults.

IMPORTANT: The PGEN.CFG file always prints out in scientific notation, but you can enter values in other units. For example, if you enter 5 mV, the value would appear in the EXCEPT.LST file as 5e-003.

To route your changes to EXCEPT.LST, use Setup/Environment Variables—Record Pgen Configuration. The default is Off. Select On to route your changes.

PGEN.CFG Variables

The pgen.cfg variables listed below are grouped in the order that they appear on the worksheet.

High Voltage Option: HIGHV

Before editing the HIGHV variable, determine whether the target system has the High Voltage Option installed.

Examples:

HIGHV No	Target system does not have the High Voltage Option installed.
HIGHV Value =	High voltage option is present on system. Standard values are 55, 60, or 100 volts.

EAO Capacitors and RCs: EAOCAP, EAORC

EAOCAP indicates the upper limit value for generating a Cap Phase test. When the test is run, the internal settings are attempted. If the setups fail, normal generation occurs.

EAORC defines the use of the extended analog assembly for RCs.

Examples:

EAOCAP 50nf	100nf is highest value to use with EAO including hightol
-------------	----------------------------------------------------------

EAOCAP No	No EAO is present on target
EAORC Yes	Use EAO for RCs
EAORC No	Don't use EAO for RCs

Multiwire Threshold: RESMWIRE, INDMWIRE, CAPMWIRE

Multiwire threshold variables generate multiwire tests where appropriate. RESMWIRE and INDMWIRE generate multiwire tests if the value is less than the one specified and if the component is properly double noted. CAPMWIRE turns on multiwire tests for capacitors if the value is $\geq 1\mu\text{F}$ and $\leq 300\mu\text{F}$.

Examples:

RESMWIRE 120	Turn on multiwire for resistors less than 120 ohms.
INDMWIRE 1	Turns on multiwire for inductors less than 1 millihenry (mH).
CAPMWIRE Yes	Turns on multiwire for C if value is $\geq 1\mu\text{F}$ and $\leq 300\mu\text{F}$.

Driver/Receiver Configuration: DRVRBL

Determine the board configuration of your target system—that is, the tester on which the boards will be tested. It is possible that you may be generating values for a system different from the one you are working on.

DRVRBL indicates a driver/receiver board that works with a variable driver voltage. Using PGEN.CFG, you can describe a target system in the board directory while maintaining the main system in the \PGT.

The basic syntax for DRVRBL is composed of single nodes and/or node ranges. Syntax permits a space and a comma, but there must be no spaces or commas between a dash and a node number.

DRVRBL 5,12-18 48	Is permissible
DRVRBL 5,12 -18 48	Will not recognize nodes 13-18 because of spaces on either side of the dash.

The default for boards not mentioned is that they are considered FIXED supply boards.

Example:

DRVRBL 5-7

Default Gray Code Stim Value: DIGSTIM

DIGSTIM specifies default stimulus frequencies for Gray code tests. Selecting DIGSTIM displays the values ENUM, F1 through F14, and F0.

This default Gray code stimulus value is assigned when components are not found in the library. PWR/GND is group zero; all other pins default to group 1.

Examples:

DIGSTIM ENUM	Enumerates stims. Each nonpower/ground pin is assigned progressively lower frequencies; that is, the first pin gets F1, the next gets F2, the next gets F3, and so forth. After F14, the complement *F1, then *F2, and so forth, are assigned.
DIGSTIM F3	

Gray Code Clock Divisor: DIGCLK

The Gray code clock divisor sets the clock divisor on all Gray code components. When a board is slow, and you need to override the clock divisor in the templates, redefine the default using the following format.

Example:

DIGCLK 4

Digital Threshold: DIGTHRESH

The Digital Threshold variables allow you to set threshold values for all digital tests. If you give only one value when generating a digital test, the program generator applies that value to both high and low thresholds. To change the value for nodes on a variable board, that is, a driver/receiver 2 board, use DIGTHRESH VARIABLE LOW and VARIABLE HIGH.

Examples:

DIGTHRESH LOW 1.6 HIGH 1.6
DIGTHRESH VARIABLE LOW 1.2 VARIABLE HIGH 1.2

Diode Variables: DLOW, DHIGH, DFORWARDI, DLEAK

The diode variables set the values for all diodes, including diodes within transistors.

Examples:

DHIGH .5V	High knee value. Default value .9V
DLOW .3V	Low knee value. Default value .5V
DFORWARDI 15ma	Forward current. Default value 10ma
DLEAK 2ua	Leakage value. Default value 1µa
DLEAK 0	No leakage tests

Zener Variables: ZCURRENT, ZFORWARDV

The zener variables set the values for all zener tests to be generated.

Example:

ZCURRENT 15ma	Zener current threshold. Default is 0.02.
ZFORWARDV 0.9V	Zener voltage threshold. Default is 0.8.

Shorts Threshold: SHTHRESH

The shorts threshold variable sets the parameters for all shorts thresholds.

Example:

SHTHRESH 10	Shorts and Special Cases. Values under 10Ω indicate shorts.
-------------	-------------------------------------------------------------

Jumper Threshold: JMPTHRESH

The jumper threshold variable sets the parameters for all jumper thresholds.

Example:

JMPTHRESH 7	Values under 7Ω indicate a jumper. Default is 5Ω.
-------------	---------------------------------------------------

Continuity Threshold: CNTTHRESH

The continuity threshold variable sets the parameters for all continuity thresholds.

Example:

CNTTHRESH 8;	Values under 8Ω indicate continuities. Default is 5Ω.
--------------	-------------------------------------------------------

Discharge Parameters: DCTHRESH, DCTIMEOUT

The discharge threshold variable sets the parameters for all discharge tests.

Examples:

DCTHRESH 100	Default value is 5 mv.
DCTIMEOUT 15000	Default is 25000ms.

Power Supply Step Worksheet Control: PROGSUPWS

The target system has a programmable power supply controller if PROGSUPWS is Yes.

Example:

PROGSUPWS No	Generates fixed power worksheets
PROGSUPWS Yes	Generates programmable power worksheets

Library Control: SYSLIB, USRLIB, LIBORDER

Library variables set the paths and search order for libraries. Type in the paths in SYSLIB and USRLIB. Select one of six values for LIBORDER: LUS, LSU, SLU, SUL, ULS, USL. L = Local; U = User; S = System library.

Example:

SYSLIB [<library path>]	Default = "\\pgt\templ_db"
USRLIB [<library path>]	Default = "\\pgt\templ_db"
LIBORDER ULS	Search user library, then local library, then system library. Default is LUS

Power Step Worksheet Disable: DISABLEPWR

DISABLEPWR disables all power Step Worksheets so that they do not execute. WaveScan requires power bus information even on power-off testers. This variable provides such information to the generator and also provides an upgrade path for power-on testers.

DISABLEPWR No	Default for 1800-series testers with digital functionality
DISABLEPWR Yes	Default for analog-only test

MultiScan Reference Node: MSCANREF

The MSCANREF variable specifies the reference node for MultiScan tests. You may specify a particular node or use 9999.

MSCANREF 12	Default is 9999.
-------------	------------------

IMPORTANT: If your PGEN.CFG file has WSCANREF for the MultiScan Reference Node, it works just the same as MSCANREF.

WaveScan Bias Current: WSCANBIAS

The WSCANBIAS variable specifies the bias current used to bias the substrate diode for WaveScan test. The range is -20 ma up to but not including -2 ma and +20 ma down to but not including +2 ma.

WSCANBIAS -9	Default is -10.
--------------	-----------------

WaveScan Threshold: WSCANTHRESH

The WSCANTHRESH variable specifies the number of volts of threshold. The measured value must be higher than the threshold for a pin to be detected. The range is 1 to 65000.

WSCANTHRESH 50	Default is 100
----------------	----------------

FrameScan also uses the WSCANTHRESH variable to control the default threshold.

FrameScan Plus Measure: FSPMEAS

The FSPMEAS variable specifies the number of the node that will be used as the measurement node for all FrameScan Plus and CapScan tests. Default is 9999; a value other than 9999 sets the FrameScan Plus measure node in PRGMVARS.

FSPMEAS 1500 Default is 9999.

CapScan Ratio: CSCNRATIO

The CSCNRATIO variable sets the ratio of the strong and weak signals for a CapScan test. Default is 1.1.

CSCNRATIO 1.2 Default is 1.1.

Combine Parallel: PRLLRLC, ENABLEPRLL

Combine Parallel enables you to combine inductors, capacitors and resistors while preserving topology. It has two modes, Manual and Automatic. For additional information about Combine Parallel, refer to the section, Combining Parallel Components, in this chapter.

The PRLLRLC variable is used with Combine Parallel Manual or Automatic mode. PRLLRLC lists all possible component combinations—R, L, C, RL, RC, LC, RLC. You can prevent a component from being combined by selecting it from the list and then simultaneously pressing the Shift and Delete keys to remove it.

PRLLRLC R,L,C,RC,RLC Default is R,L,C,RL,RC,LC,RLC

To add a deleted component, position the cursor in the list and type the component.

PRLLRLC R,L,C,RC,LC,RLC;

The ENABLEPRLL variable is used with Automatic mode and performs a combine parallel on global Generate when set to its default Yes. If set to No, you will be required to combine components using the Manual mode.

ENABLEPRLL No Default is Yes.

Major Separators: MJRSEP

The MAJRSEP variable enables you to attach an additional identifier, up to four characters, to a component's major ID, before the dash (-) or the underscore (_).

Example:

MAJRSEP x- Identifies the major component ID U1x gray or U1x vec, distinguishing it from other similar components.

MultiScan Fast: MSCANFAST

MSCANFAST specifies whether to use fast mode for WaveScan or FrameScan tests. The default is No. If Yes, the number of sample measurements is reduced in order to increase throughput.

MSCANFAST Yes Default is No.

DeltaScan Parameters: DSFIXTHRESH, DSFIXPIN, DSDEFTHRESH,

The DSFIXTHRESH variable determines the fixed threshold value number for newly generated DeltaScan tests.

The DSFIXPIN variable specifies the fixed number of pins on a chip. If a chip has fewer pins than specified by DSFIXPIN, you must use the fixed threshold variable, DSFIXTHRESH.

The DSDEFTHRESH variable determines the DeltaScan default test threshold at Low, Nominal, High, or Fixed. Default is Nominal.

Examples:

DSFIXTHRESH 30 Default is 50.

DSFIXPIN 72 Default is 32.

DSDEFTHRESH Low

Ignore Threshold: IGNORETHRS

The ignore threshold variable sets the parameters for all ignore thresholds.

Example:

IGNORETHRS 5 (2-50) Threshold for newly generated interconnect Ignore steps

Shorts Wait: SHWAIT

The SHWAIT variable defines the wait time (in ms) of generated and learned short steps.

SHWAIT 30 Default is 0.

PRISM-Z Test Generation: PRISMGEN

The PRISMGEN variable defines which components, if any, are to be tested using the PRISM-Z board.

- Test all components with the ATB
- C Test capacitors with PRISM-Z
- L— Test inductors with PRISM-Z
- LC Test inductors and capacitors with PRISM-Z
- R— Test resistors with PRISM-Z
- R—C Test resistors and capacitors with PRISM-Z
- RL— Test resistors and inductors with PRISM-Z
- RLC Test resistors, inductors and capacitors with PRISM-Z

Discharge Generate Value: DISCHGENVAL

During Generate, discharge steps are automatically generated for capacitors greater than the value defined in DISCHGENVAL. Discharge steps are not generated for capacitor values smaller than those defined in DISCHGENVAL. This variable speeds up test execution by reducing the number of discharge steps performed.

DISCHGENVAL accepts values expressed in scientific notation.

Example:

DISCHGENVAL 5e-05 Default is 5e-05.

Editing the PGEN.CFG Text File

If you need to edit the text file or have one of your programs edit the PGEN.CFG file,

1. Look at the PGEN.CFG variables in the PGEN.CFG worksheet for variable names and ranges.
2. Look in the PGEN.CFG file for actual syntax.

Note that:

- The PGEN.CFG files are not case-sensitive.
- Comments take the same form as for Vector ASCII: an apostrophe or a pound sign anywhere at the beginning of a line of text or the backslash and asterisk at either end of one or more lines of text (`/* ... */`) all delimit comments.
- Annotations are preceded by hyphen (–) whereas the default PGEN.CFG are preceded by a pound sign (#).

Setting Up PEP for Input List Analysis

The Programmer Efficiency Package (PEP) automatically examines your input list and analyzes the testability of the components on your board. PEP issues a report on recommendations for the input list and, if you choose, makes changes to your input list to improve test efficiency. The function is accessible from the 18xx system software's interface, DOS, and Windows.

For detailed information about PEP, see the **Programmer Efficiency Package User's Guide**.

To set up PEP from the 18xx environment be sure that you are in the correct board directory; then

- 1 Select PEP from the Pgen menu.

The PEP Setup interface appears. The defaults are displayed.

Programmer Efficiency Package Options	
Update IPL	: No
Backup IPL.Dat to	: IPL.SAV
Add System Tolerance	: No
Generate Report	: No
Analyze IPL File: IPL.DAT	
Report Filename	: IPL.RPT
<div>OK Cancel</div>	

- 2 Click the Update IPL field to specify whether you want to update the input list.

A pop-up appears allowing you to select Yes or No. The default is No. When you select Yes, the Backup IPL.DAT field changes color and becomes active. You can then type in the name of the file to which you want the original input list saved if it is different from the default.

IMPORTANT: Program generation recognizes only the file name IPL.DAT for generating a test. It is important to identify your various IPL.DAT files. Be sure that you back up the original IPL.DAT file and change its name so you can identify it later. If you wish to generate a test from it later, after having created an IPL.DAT with PEP, change the names of the files accordingly.

- 3 Click the Generate Report field to specify whether you want to generate a report.

A pop-up appears allowing you to select Yes or No. The default is No. When you select Yes, the Report File Name field changes color and becomes active. You can then type in the file name of for the report if it is different from the default.

- 4 Enter the name of the file you want PEP to analyze in the Analyze IPL File field.

IPL.DAT is the default. If you have an input list with another name, enter that name in the field.

- 5 Click OK to confirm your selections and save them or Cancel to terminate the process.

Using Clean and Build

It is important to understand the relationship between Clean and Build.

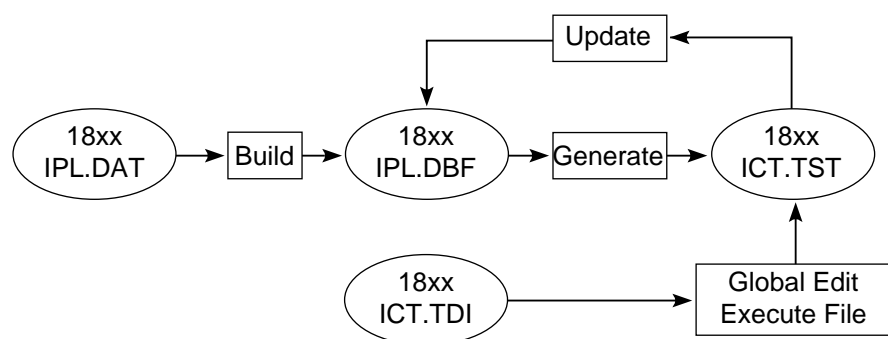
Clean

Clean removes existing test program files from the current board test directory. After you have used Clean, the only test-related files will be the input list, IPL.DAT, and a skeleton in-circuit test file, ICT.TST. If a local library was created in the board directory, then the files relating to the library will still exist.

Build

Build creates new entries in the IPL.DBF for each component found in the user-supplied IPL.DAT file. If you “build” twice or more on the same input list without using the Clean function, a second set of identical tests will be added to your test program.

The following diagram shows the program development process using Test Data Input (TDI)



See chapter 9, “Test and Debug Tools,” for more information about this function.

Learning Interconnect Information

If interconnects are not included in the input list, use Learn to insert the interconnect test steps in the program.

Before you begin the Learn Interconnect process, be sure the Board Name field shows ICT instead of FST.

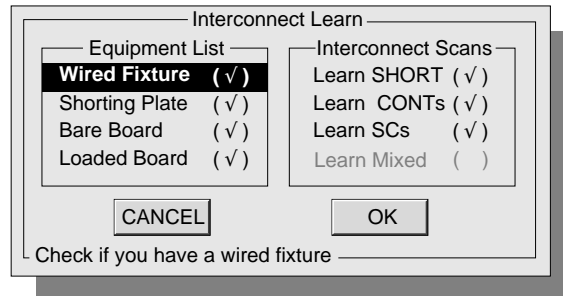
Use the Interconnect Learn window to

- Check Shorting Plate to learn the range of shorts.
- Check Loaded Board to learn special cases.
- Check Bare Board to learn continuities.

The following brief tutorial explains how to learn interconnects,

- 1 Select Pgen/Learn.

The ICT Interconnect Learn window appears with Wired Fixture, Shorting Plate, Bare Board, and Loaded Board selected by default. Because all items in the Equipment list are selected, Learn SHORT, Learn Cont's, and Learn SC's in the Interconnect scans list are also selected.



- 2 Deselect Shorting Plate in the Equipment list, since this example does not use a shorting plate.

Learn SHORT is automatically deselected in the Interconnected scans list.

- 3 Select OK.

The threshold window appears. The CNT threshold is 5 ohms by default. The Short range field shows 0 to 0.

- 4 Type 0 to 639 in the Short range field.

- 5 Select Execute.

A message box asks you to place the bare board on the fixture and apply vacuum.

- 6 Place the selftest fixture on the interface.

Remember—this tutorial uses the selftest fixture as both bare board and loaded board.

- 7 Before you apply vacuum, connect the vacuum hose to the selftest fixture.

- 8 Set the Fix Vac 1 switch to manual to apply vacuum.

- 9 Press any key to continue.

From the bare board, the tester learns the continuities it will test. A message “No continuities found” may appear briefly on the screen, followed by a message asking you to place the loaded board on the fixture and apply vacuum.

Since the tutorial uses the fixture self-test as the loaded board, leave the self-test fixture in place.

- 10 Press any key to continue.

From the loaded board, the tester learns of the low-impedance paths on the loaded board that will interfere with its Shorts test. It places this information in the Ignore table. You can see the Ignore table by selecting Edit from the Main menu, then selecting Intc/Ignores. Ignores may be due to potentiometers, jumpers, fuses, relays, transformer coils.

A message “No continuities found” may appear briefly on the screen, followed by a message **Working** which indicates the software is learning the low resistance paths.

Next, the software displays a message “Packing User File” which indicates the software is writing the interconnect information it has learned to the ICT.TST file.

Your program now has a complete interconnect section, including jumper entries, continuities entries, and shorts range. After the Learn program has run with the loaded board, it returns you to the Main menu.

Generating the Program

Using an input list to generate a program automatically is the most common method used to develop programs. The process begins with complete component information in the form of an input list. The input list either is derived from CAD data files or is created manually with an editor.

Automatic Program Generation

The basic procedure for automatically generating a program is outlined below.

- 1 Create a board program directory.
See the section, Creating a New Board Program Directory, in the **Z1800-Series Board Test Tutorial**.
- 2 Copy an input list into the new board directory.
The input list may be derived from CAD/CAE data or may be manually created.
- 3 From the Main menu select Pgen.
- 4 Select Clean from the Pgen pulldown menu.
A Query Window appears asking "Remove existing files?"
- 5 Select Yes.
The screen displays messages as the process occurs. When the process is complete, you are in the Main menu.
- 6 Select Build from the Pgen pulldown menu.
The program builds the component database files IPL.DBF, IPL_NOD.NDX, and IPL_TOK.NDX.
- 7 Press any key.
The CRT window disappears and you are in the Main menu.
- 8 Select Pgen from the Main menu.
- 9 Select Generate.
The Program Generator window appears and displays messages as it generates the program ICT.TST and its test files. When generation is complete, you are in the Main menu.

Incremental Program Generation

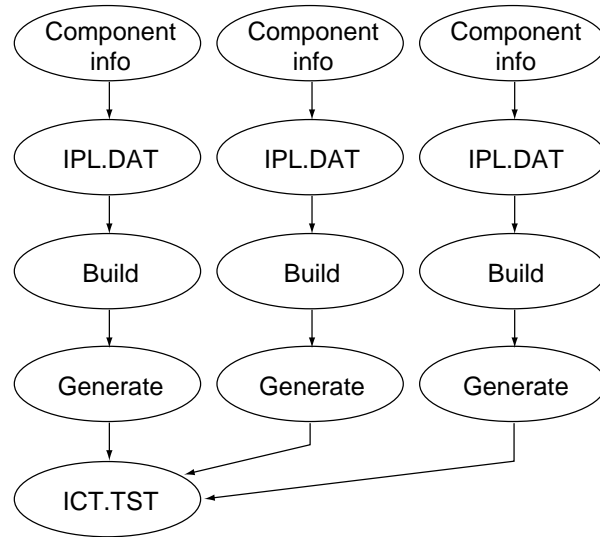
It is also possible to build your test program in incremental steps using an input list.

For example, if you have a board design with two independent sections, one section might be completed and assembled and ready for test while the other part is still a bare board. In such a situation, you might create an IPL.DAT for the finished portion and later add the remaining devices to the test program using the Build and Generate functions.

Build creates new entries in the IPL.DBF for each component found in the user-supplied IPL.DAT file. If you build twice or more on the same input list without using the Clean function, a second set of identical tests will be added to your test program. Therefore, be sure your incremental input list contains only new components data and does not repeat information from the original input list.

IMPORTANT: No error message alerts you to the addition of the second set of test steps. The extra steps will increase test time.

The following diagram shows the incremental generation process.



Combining Parallel Components

The combine parallel components function combines inductors, capacitors, and resistors while preserving topology. The 18xx system software has two modes for combining these types of components which share two nodes— manual mode, invoked by the Combine Parallel selection in the Pgen pull-down menu, and automatic mode performed as part of Pgen/Generate.

If the parallel devices are all the same type, such as two or more capacitors in parallel, then a single test page is all that will be necessary to test the equivalent component value. If the components are of different types, such as a resistor and a capacitor, then separate test pages will be generated for each component. If one of the parallel components is an inductor, then only the inductor is tested since the DC resistance of the inductor will cancel out any parallel component impedance.

The table below shows where the parallel tests are located.

Parallel Type	Goes in Section
L	L
RLC	L
RL	L
LC	L
C	C
RC	C
R	R

PRLLRLC

You can specify which components will be combined, whether you use Manual or Automatic Mode, by editing PRLLRLC in the PGEN.PGF file. PRLLRLC lists all possible component combinations; you can prevent a component type from being combined by omitting that type from the list: R, L, C, RL, RC, LC, RLC.

Refer to the Section, Using PGEN.CFG to Reassign Default Values, earlier in this chapter for information about editing PGEN.CFG.

Manual Mode

The manual mode of combining parallel components is useful in editing an existing program which has been changed. You may make changes in nodes and values to the existing test steps, both individual and parallel. However, you should avoid creating additional parallel test steps prior to running Combine Parallel.

If you create your own parallel test steps, 18xx sees them as additional physical devices in parallel with the individual components. Combine Parallel then creates another parallel test step. If you need to add more components to the test, just create a test for the component(s). Combine Parallel then correctly interprets the topology and creates a parallel component step, if necessary.

Using Combine Parallel

To use Combine Parallel, select Combine Parallel from the Pgen pull-down menu.

A query window appears asking if you want to combine the components. If you select Yes, parallel component steps will be created and the source components will be disabled.

Viewing Results of Combine Parallel. You can see the results of Combine Parallel actions on the whole test program in the EXCEPT.LST file and on a test-by-test basis in the Component Select windows and Step Worksheets.

The message in EXCEPT.LST tells you what components it found in parallel (and disabled their tests) and the parallel test step that it created. For example, if R1 and C32 are in parallel, the software writes a message stating that it found R1 and C32 in parallel, disabled their individual test steps, and created a parallel component step with the ID field R1,C32.

In the 18xx interface, parallel components are indicated in the Component Select window by three dots following the component name, R1, ... for example. Components that have been disabled because they are being tested in parallel are indicated by an asterisk.

IMPORTANT: Not all components marked with an asterisk have been disabled because of the combine parallel function.

To find out what other component is tested in parallel with the resistor, for example, select I/O Control from the Pre-Test Options of the resistor test page. The Comments field of the Pre-Test Options window displays a message similar to the following:

Parallel of - R 1,C32

Automatic Mode

The actions performed by Combine Parallel in the Pgen menu are automatically performed as part of Pgen/Generate. Thus, if R1 and C32 are listed in the input list for a test board, Pgen/Generate determines that they are in parallel and creates individual test steps for both devices in the appropriate sections. Then the software creates a parallel test step for R1,C32 in the Capacitor section and disables the individual test steps. The individual test steps, although disabled, are still used in the board topology, but the new parallel component test step is NOT included in the topology to prevent duplicate counting of components in board analysis.

The automatic mode performs a combine parallel on global Generate when ENABLEPRLL, in PGEN.CFG is set to its default, "Yes." When set to "No," you are required to combine components using the manual mode.

If, while generating the test, the program generator finds parallel components that it previously created, a query window appears asking if you want to delete the original parallel steps. Note that Parallel Components created using manual mode will not be recognized at this time.

IMPORTANT: If you have a program in which you manually disable a source component and then manually add a parallel component, when you run combine parallel, pgen will consider your manually created component as another source and combine it with the old sources.

Disable: *R1*R2
 Add: R1,R2
 Combine parallel
 results (1/2 final value): R R2 R1,R2

You must delete the manually created parallel component step if you want to use combine parallel.

Updating the Test Program

Pgen/Update corrects the component database to account for component tests which you have added, deleted, or edited. Correcting the component database files ensures that the component database, IPL.DBF, agrees with your test program, ICT.TST.

To update your files, select Pgen/Update from the Main menu. The screen displays messages about the progress of the update.

Validating the Test Program

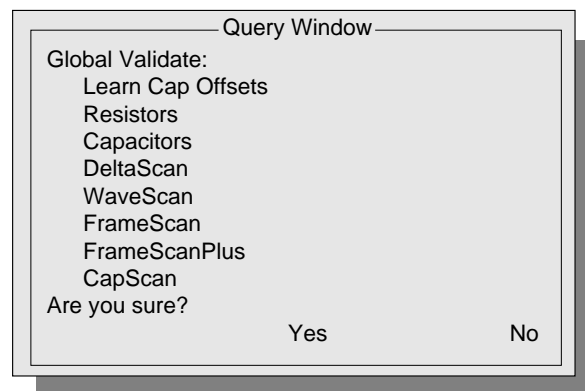
Validate improves test measurements by finding guard points of resistors and capacitors and those components tested using DeltaScan, WaveScan, FrameScan, FrameScanPlus, CapPhase, and CapScan techniques. It determines the correct guards, wait time, squelch time, and polarity of stimulus/measure.

See the **MultiScan User's Guide** for additional information about validation.

To perform global Validate,

- 1 Be sure the fixture and board-under-test are connected to the tester.
- 2 Enter the appropriate values and parameters in the Setup/Validate Configuration window if you have not already done so.
 See chapter 2, "System Setup," for information about setting up Validate.
- 3 Be sure that you have updated the component database to make it current.
- 4 Select Pgen from the Main menu.
- 5 Select Validate to run global Validate.

A query window appears asking if you are sure that you want to run Validate.



- 6 Select Yes.

The CRT window appears displaying the message
 Press START to test a board

7 Select Start.

Validate tries test parameters on the components to determine what works best with the values requested on the Step Worksheet.

During validation, messages appear in the CRT window and in the status area at the bottom of the screen, informing you of progress.

Global validation takes a minimum of an hour or two to run on a board with about 300 nodes with 20 to 30 ICs, 50 resistors, 70 capacitors, and other resistor packs and analog components. To stop the Validate process, press F10 or Cancel. Validation stops after the current component is complete.

Generating Reports in Pgen

When you select Pgen/Reports/Report All, program generation results in several outputs:

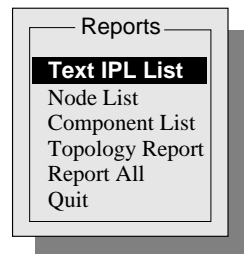
- IPL.LST
- NOD.LST
- COMPLST
- TOPOLOGY.LST

The IPL.LST, NOD.LST, TOPOLOGY.LST, and COMPLST files are generated when you select the individual report or Report All.

Reports Menu

Reports allows you to generate a number of program generator reports concerning the nodes, components, input list, and topology for a board test. The reports apply to the active board test directory.

When you select Reports from the Pgen menu, the screen below appears.



Text IPL List generates a text file documenting the contents of the binary component database file, IPL.DBF. This file, IPL.LST, is derived from the component database and may differ from the original text input list due to program changes. Update your component database to reflect the latest program changes before generating an IPL.LST file.

IMPORTANT: Do not use the IPL.LST file as an input for Build to regenerate a program. If you attempt to regenerate a complete test program from this file by renaming IPL.LST to IPL.DAT and doing a Build, you will lose all of your test data such as guard points, test modes, wait times, Pre- and Post-Test option settings.

Node List generates a nodes-to-components reference table which describes the board topology sorted by node number and lists all components connected to the nodes.

Component List generates a components-to-nodes reference table which describes the board's topology sorted by component and component pin, listing all connected nodes.

Topology Report generates two lists:

- components with matching IDs but different nodes
- components with different IDs but matching nodes

See the section, Managing Topology, below for detailed information about Topology Report.

Report All generates all four lists. These files are located in the board directory.

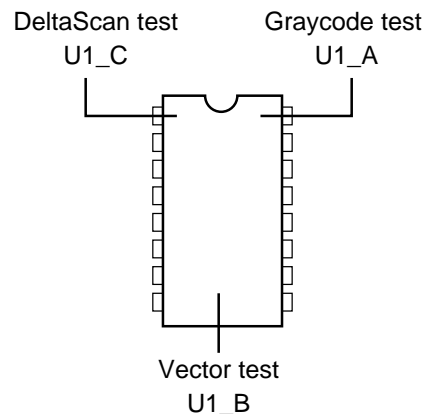
Managing Topology

Some test methods require accurate topology information during test creation. The lack of accurate topology causes tests to pass (or, less frequently, fail) erroneously. The Topology Report provides you with this information.

Topology in the 18xx system software is based on the concept of major and minor IDs. A component's major ID is the part of the ID that comes before a dash (-) or an underscore (_). A component's minor ID is the part of the ID that comes after a major ID. For example, in U1_A, U1 is the major ID, and _A is the minor ID.

Two IDs are identical if they have the same major ID. For example, 18xx system software recognizes U1_A and U1_B as the same physical component.

Test methods that have more than one test associated with the same component require that you use the same major ID in each test. For example, U1 may be tested using a Gray code test, a vector test, and a DeltaScan test. Each test should have the same major ID, such as, U1_gray, U1_vec, and U1_delta.

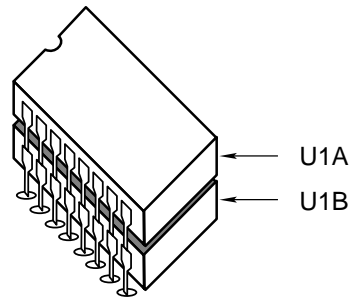


It is possible to distinguish a particular component by using the MAJRSEP command in PGEN.CFG. For additional information about MAJRSEP, refer to Recording PGEN.CFG Changes in this chapter.

IMPORTANT: When you are developing Multipanel board tests, do not use the dash or underscore in naming the same components in sequential panel tests, as in R_1 for a resistor test on panel one and R_2 for a resistor test on panel_ 2.

If your board test includes overlapping parts, the two components must have different IDs, such as U1A and U1B. Overlapping parts are those parts that share all the same nodes but different

major IDs. Even one unique pin on each component means that they do not overlap.



Analog components overlapping digital and digital components overlapping analog will not be reported.

To ensure that accurate topology information exists for your board test, you should check the Topology Report, which is accessed under Pgen/Reports, just after a Build but before a Generate. The report has two sections. Section 1, “Reporting—Components with matching IDs but different nodes” should be empty. Section 2, “Reporting—Components with different IDs but matching nodes” should contain only truly overlapping parts.

To update the Topology,

- either edit the IPL.DAT and regenerate using Pgen/Generate OR Clean and Build, or
- edit the Component Properties portion of the Step Worksheet and update the IPL.DBF using Pgen/Update
- rerun Pgen/Report/Topology to view results, then fix any conflicts that were reported

Topology Report Before Editing

A topology report before editing your board test looks similar to the following.

```
# The following is a topology report.
# =====
# Topology is based on the concept of Major and Minor Id's.
# A component's Major Id is the part of the Id that comes before
# a dash '-', or an underscore '_'.
# A component's Minor Id is the part of the Id that comes after
# a Major Id.
# Example: in 'U1_A', 'U1' is the Major Id and '_A' the Minor Id.
# Two Id's are identical if they have the same Major Id.
# Example: 'U1_A' and 'U1_B' should refer to the same component.
#
# Test methods that need accurate topology information can be
# supported by ensuring that all components that have more than
# one test associated share the SAME Major Id in each test step.
# Example: U1 is tested as a Grey code test, AND a vector test
AND
#           a DeltaScan test. Each test should have the same Major
#           Id. (i.e. 'U1_gray', 'U1_vec', and 'U1_delta')

# It is recommended that all components that share Major Id's be
# given DIFFERENT Minor Id's
# Note: Analog components overlapping Digital and Digital
# components overlapping Analog will not be reported.

# Reporting - components with matching Id's but different nodes.
# =====
Check group -
  Id = 'FL1'      Section = Jumpers      Token = J
  Id = 'FL1'      Section = DeltaScan    Token = DSCAN
Check group -
  Id = 'FL3'      Section = DeltaScan    Token = DSCAN
  Id = 'FL3'      Section = Jumpers      Token = J
Check group -
  Id = 'FL4'      Section = DeltaScan    Token = DSCAN
  Id = 'FL4'      Section = Jumpers      Token = J
Check group -
  Id = 'J5'       Section = Jumpers      Token = J
  Id = 'J5'       Section = Jumpers      Token = J
Check group -
  Id = 'J7'       Section = Jumpers      Token = J
  Id = 'J7'       Section = Jumpers      Token = J
Check group -
  Id = 'L2'       Section = Jumpers      Token = J
  Id = 'L2'       Section = Inductors    Token = L
Check group -
  Id = 'R36'      Section = Resistors    Token = R
  Id = 'R36'      Section = Jumpers      Token = J
Check group -
  Id = 'R43'      Section = Resistors    Token = R
```

```

        Id = 'R43' Section = JumpersToken = J
Check group -
        Id = 'S2' Section = JumpersToken = J
        Id = 'S2' Section = AnalogToken = ATMPL
Check group -
        Id = 'T1' Section = AnalogToken = ATMPL
        Id = 'T1' Section = JumpersToken = J
Check group -
        Id = 'T2' Section = JumpersToken = J
        Id = 'T2' Section = AnalogToken = ATMPL

# Reporting - components with different Id's but matching nodes.
# =====
Check group -
        Id = '5V' Section = Board Power    Token = PWR5
        Id = 'C10,....' Section = Capacitors    Token = C
        Id = 'C1019' Section = Capacitors    Token = C
        Id = 'C15,...,' Section = Capacitors    Token = C
        Id = 'R42' Section = Resistors    Token = R
Check group -
        Id = 'C3/R964' Section = Capacitors    Token = C
        Id = 'R964' Section = Resistors    Token = R
Check group -
        Id = 'CR5' Section = Diodes    Token = D
        Id = 'J4' Section = Jumpers    Token = J
Check group -
        Id = 'CR6' Section = Diodes    Token = D
        Id = 'J3' Section = Jumpers    Token = J
Check group -
        Id = 'FL3' Section = Jumpers    Token = J
        Id = 'S2' Section = Jumpers    Token = J
Check group -
        Id = 'FL4' Section = Jumpers    Token = J
        Id = 'J5' Section = Jumpers    Token = J
        Id = 'J7' Section = Jumpers    Token = J
Check group -
        Id = 'L2' Section = Jumpers    Token = J
        Id = 'R37' Section = Jumpers    Token = J
        Id = 'R5' Section = Jumpers    Token = J
Check group -
        Id = 'R3' Section = Resistors    Token = R
        Id = 'R965' Section = Resistors    Token = R
Check group -
        Id = 'R35' Section = Resistors    Token = R
        Id = 'R36' Section = Resistors    Token = R
Check group -
        Id = 'R43' Section = Resistors    Token = R
        Id = 'R50' Section = Resistors    Token = R
Check group -
        Id = 'R7' Section = Resistors    Token = R
        Id = 'R8' Section = Resistors    Token = R
# End of Topology Report

```

Topology Report After Editing

A Topology Report after editing should look similar to the following.

```
# Reporting - components with matching Id's but different nodes.
# =====

# Reporting - components with different Id's but matching nodes.
# =====

Check group -
  Id = '5V' Section = Board Power Token = PWR5
  Id = 'C10,....' Section = Capacitors Token = C
  Id = 'C1019' Section = Capacitors Token = C
  Id = 'C15,...,' Section = Capacitors Token = C
  Id = 'R42' Section = Resistors Token = R
Check group -
  Id = 'C3/R964' Section = Capacitors Token = C
  Id = 'R964' Section = Resistors Token = R
Check group -
  Id = 'CR5' Section = Diodes Token = D
  Id = 'J4' Section = Jumpers Token = J
Check group -
  Id = 'CR6' Section = Diodes Token = D
  Id = 'J3' Section = Jumpers Token = J
Check group -
  Id = 'FL3' Section = Jumpers Token = J
  Id = 'S2' Section = Jumpers Token = J
Check group -
  Id = 'FL4' Section = Jumpers Token = J
  Id = 'J5' Section = Jumpers Token = J
  Id = 'J7' Section = Jumpers Token = J
Check group -
  Id = 'L2' Section = Jumpers Token = J
  Id = 'R37' Section = Jumpers Token = J
  Id = 'R5' Section = Jumpers Token = J
Check group -
  Id = 'R3' Section = Resistors Token = R
  Id = 'R965' Section = Resistors Token = R
Check group -
  Id = 'R35' Section = Resistors Token = R
  Id = 'R36' Section = Resistors Token = R
Check group -
  Id = 'R43' Section = Resistors Token = R
  Id = 'R50' Section = Resistors Token = R
Check group -
  Id = 'R7' Section = Resistors Token = R
  Id = 'R8' Section = Resistors Token = R
# End of Topology Report
```

Section 2 of this edited report displays RCs, parallel low impedance components, and parallel resistors, all of which have the proper topology.

▼▼▼

8 MANAGING TEMPLATE LIBRARIES

Chapter 8 discusses library database management and template creation.

The system software's hierarchal library structure consists of System, User, and Local libraries. You can also create additional libraries as you need them. Analog and Gray code templates are included in the Standard Template Library, and vector templates are in the VLSI Binary Library.

You can perform a variety of editing operations on templates, including the creation of analog, Gray code and vector templates.

Information about the ASCII vector language is provided to enable you to modify translated JEDEC or TSSI input files, for example, to create ASCII vector files usable by the system software as vector templates.

You can also create your own templates. The program generator uses input from the template library to create all test steps for basic analog components, interconnects, and digital components.

Managing the Template Library

In the 18xx system software, templates contain the information necessary to test a device. The various types include analog, Gray code, and vector templates which are stored in several possible database files called template libraries.

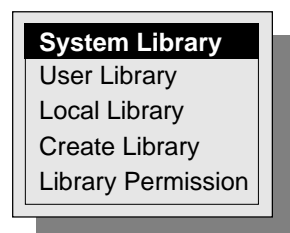
Three levels of libraries provide flexibility in the management of your template databases: System Library, User Library, and Local Library.

For detailed information about using the PGEN.CFG file to configure library paths, see chapter 7, "Program Generator Tools."

The default system library path is \PGT\TEMPL_DB. There is no default user path. The local user path is always the current board directory, that is a dot (.)

The Templates menu, located in the Main menu, provides access to these libraries.

When you select Templates, the menu appears. It presents you with a hierarchal system of libraries.



The System Library is the Teradyne Library. The User Library contains all of the common, modified templates. Local Library refers to a library in the currently selected program directory.

Create Library enables you to create new System, User, or Local libraries.

The Library Permission function, available only to someone with supervisor permissions, enables you to set up whether or not a write permission is required for the various libraries.

You can modify templates using the Template edit menu or DOS commands and their arguments.

Setting Up Library Permissions

Library permissions consist of Read Only or Read and Write permissions. The type of library permissions you have determines the number of template edit functions available to you in the System Library, User Library, and Local Library. If you have Read and Write permissions, then you have full access to all template edit functions: Add, Delete, Undelete, Extract, List, Pack, Jedec, Utility, and Quit. If you have Read Only permissions, then only the Extract, List, and Quit functions are available.

Only the supervisor may set up library permissions through both the Logins in the Setup menu and Library Permission in the Templates menu.

The following matrix explains the relationship between permissions set using the Logins menu and the Library Permission function.

		Templates Library Permission	
Login Permissions		Write Permission required	Write Permission not required
	No write Permission	Can only read library database	Can read and write to library database
	Has write Permission	Can read and write to library database	Can read and write to library database

To establish permissions using the Logins menu see chapter 1, “Introduction to System Software.” After having setup the desired permissions in the Setup menu, specify the Library Permissions.

To use the Library Permission

- 1 Select Library Permission from the Templates menu.
A dialog box appears.
- 2 Select a library type and click OK.
Another dialog box appears requesting the library path name.
- 3 Specify the library path name.
- 4 Specify whether a write permission is required for this library path—[Yes/No]— and click OK.

Creating a New Library

Your 18xx system software comes with an extensive System Library. To customize the library to your needs, you can create a User Library or a Local Library. Creating a new library requires that you establish the path name as well as your new library or directory. The Create Library selection in the Templates menu provides dialog boxes for setting them up. When you have made your specifications, you will have a skeleton directory consisting of ALIAS.LNK, DEVICE.LST, and TEMP.IND or TEMPL.NDX.

To create the library

- 1 Select Create Library from the Templates menu.
A pop-up window appears requesting that you select the library type: System Library, User Library, or Local Library.

- 2 Select the library type.
A dialog box appears requesting the path name for the library.
- 3 Enter the path name and select OK.
A dialog box appears asking if you want to create a new library or directory.
- 4 Select Yes.

DOS Command Equivalent

The DOS command line equivalent for creating a library (-c) and setting up permissions (-i) is

```
18xx /T -m [-c -i]
```

The -l<lus> -m arguments alone invoke the Template Utilities menu.

**Using Templates Menu
Edit Functions**

When you select System Library, User Library, or Local Library, the following edit menu appears.

Add	Delete	Undelete	Extract	List	Pack	Jedec	Utility	Quit
-----	--------	----------	---------	------	------	-------	---------	------

The primary functions you can perform with the edit menu are the following:

Command(s)	Function
Add	Add templates to the library. Also updates templates
Delete, Undelete	Delete/undelete templates from the library.
Extract	Extract templates to TTM files or ASCII vector files.
List	List aliases and templates including specific characteristics of templates.
Pack	Eliminate unused data from template database.
Jedec	Translate Jedec vectors into ASCII format.
Utility	Revision: Access template revisions other than the currently active one. Import Aliases: read contents from a text file into the template database. Export Aliases: write the contents of the template database to a text file.

Using DOS Commands for Template Menu Functions

In addition to using the pulldown commands to modify the template library database, you can use the DOS command, 18xx /T and its arguments. The DOS equivalents for the pulldown functions are listed at the end of each section discussing the menu commands.

Except for the DOS library creation function, when using 18xx /T, you must use the -l library argument. This argument along with one of l, u, or s specifies respectively the local, user, or system library, as in

```
18xx /T -l<lus> -a <ttm name>.
```

When specifying the local library, you must be in the directory where the local library resides.

Using the Add Function

As you develop new templates to suit your special needs, you need to add them to the template library database. Using the Add function, you select the template you want to put into the library, and Add automatically places it there.

The Add menu has the following commands:

Template	Takes a template in .TTM format (see Extract) and adds it to an existing library
Alias	Adds an alias to an existing template
ASCII Vector	Adds a vector in ASCII format to the VEC library.
IVL Vector	Adds an IVL vector to the VIMG library
IVL Cluster	Takes an IVL file with tester node numbers and creates a vector test for a cluster of devices. Used with Boundary Scan devices.

The template library function can be very useful when managing libraries, for example, if you wish to transfer templates from one machine to another. Extract creates an exact copy of an existing template on one machine, and Template adds it to the new machine's library in exactly the same image.

Adding Templates to Library

To add a new template in .TTM format to the library database

- 1 Click the Add menu.
The menu expands.
- 2 Select Template.
The selection window appears listing all the .TTM file in the .\TTM subdirectory of the library. If the .TTM file for the device you wish to add is not listed, or the .\TTM directory is empty, move the .TTM file of interest into this directory via DOS.
- 3 Select the template you want to add to the library.
If the template already exists in the database, a message appears to alert you. Press any key to return to the .\TTM selection window.
If an older version of the same template already exists in the database, a query window appears asking if you want to update the template. Click Yes to update that template.

DOS Command Equivalent

```
18xx /T -l<lus> -a <ttm name>
```

Adding Aliases to Library

To add a new alias to an existing device in the template library

- 1 Select Alias from the Add menu.
The Add New Alias window appears.
- 2 Enter the name of the existing device for which you want to create an alias, and press Enter.
- 3 Enter the new alias name, and press Enter.
- 4 Click OK or Cancel.
If a template does not exist in the database for the name entered for the existing database or if the new alias name already exists, an error message will appear.

DOS Command Equivalent

```
18xx /T -l<lus> -al <dev name> <alias name>
```


Adding ASCII Vectors

IMPORTANT: If you have an analog-only test system, information regarding digital test is not pertinent to your test situation. The analog-only functionality prevents you from generating, editing, or running digital tests.

To add an ASCII vector in .ASC file format to the library

- 1 Select Add ASCII Vector from the Add menu.
The ASCII vector selection window appears.
- 2 Select the ASCII vector you wish to add to the library.
If the template already exists, a query window appears asking you if you want to update the vector template. Click Yes or No. If you click Yes, the program parses the vector, and adds it to the library. If you click No, the process aborts. when complete, the function returns you to the ASCII vector selection window.

DOS Command Equivalent

```
18xx /T -l<lus> -av <asc name>
```

To add to a local library, invoke 18xx /T -ll -av in the directory where the library resides and specify the path as shown below:

```
18xx /T -ll -av .\TTM\68040.asc
```

Adding IVL Vectors

To add an IVL vector in .ivl format

- 1 Select IVL Vector from the Add menu.
The Add IVL Vector selection window appears, showing the .IVL files that are in the .\TTM directory.
- 2 Select the IVL vector you wish to add to the library.
If the template already exists, a query window appears asking if you want to update the vector template. Click Yes or No. There is no DOS command to add IVL vectors.

Adding IVL Clusters

IVL Clusters are added to Local or User libraries. An error message appears if an attempt is made to add one to the System library. To generate an IVL Cluster, use Pgen’s VCLUST token in the input list.

To add an IVL Cluster

- 1 Select IVL Cluster from the Add menu.
A selection window appears showing the .IVL files that are in the current board directory.
- 2 Select the .IVL file from which you wish to produce a .VEC file.

The following is a summary of digital and analog template types, their corresponding libraries, input list tokens and device types. Further information about their use and creation follows.

DOS File Type	Library Name	Input List Token	Device Type (In Comp ID)
---	GCTEMPL.LIB	IC	Gray Code
xxxx.ASC	VRTEMPL.LIB	VEC	Vector Template
xxxx.IVL	VRSNAP.LIB	VIMG	Vector Image
----	ANTEMPL.LIB	ATMPL	Analog Template

Extracting Templates

Extract in the Templates menu has three functions:

- **Extract Device:** To copy a device model (template) for a device into a TTM file so that you can transport it;
- **Extract Modified:** To extract (copy) all user-modified templates to individual TTM files.
- **Extract ASCII Vector:** To extract a specified vector template from the VRTEMPL.LIB file, translate it from binary to ASCII format for editing so that you can restore the edited template to the library.

To extract one template from the library database

- 1 Select Extract from the Templates menu.

The menu expands to list the Extract Device and Extract Modified commands.

- 2 Select Extract Device.

The Extract Device window appears.

- 3 Enter the name of the device you wish to extract, and press Enter.

- 4 Click OK or Cancel.

If you click OK and the device does not exist in the database, an error message appears to alert you. If the device exists, the extracted template is placed in a TTM file in the current library's directory.

If a TTM file already exists in the current library's directory having the same name as the device being extracted, a query window appears with a message similar to the following:

`74245.TTM already exists. Can this file be removed?`

- 5 Click Yes if it is all right to remove the existing file, or click No if you do not want to remove the existing file.

IMPORTANT: The template Name can be 16 characters with no embedded spaces. However, extracting a template truncates the name to the first 8 characters plus a .TTM extension.

Template library functions such as Add, Update, Add Alias, and Delete mark templates as being modified. All such modified device records can be extracted at one time by using the Extract Modified function.

To extract all modified templates

- 1 Select Extract Modified from the Extract menu.

All modified devices are written to TTM files in the directory .\TTM.

If a TTM file already exists in the .\TTM directory having the same name as the device being extracted, a query window similar to the one in step 5 above appears.

- 2 Click Yes to remove the existing file. Click No if you do not want to remove the file.

DOS Command Equivalent

```
18xx /T -l<lus> -x
```

To convert the binary format of a vector template to an ASCII format for editing

- 1 Select Extract ASCII Vector from the Extract menu.

A window appears allowing you to specify the template you want to edit.

- 2 Enter the name of the template in the Device Name field and select OK to extract it or Cancel to stop the operation.

Selecting OK removes the template from the library, translates it to ASCII format, and makes a file of that name in the TTM directory of the library that you are working in. Once the template is extracted, you can modify it.

IMPORTANT: Comments are not saved to the template library, therefore, the extracted template will not contain comments.

Managing Database Size

The Delete, Undelete, and Pack functions provide you with tools which affect the size of the library database. A packed template database is generally smaller than an unpacked one and improves template access time. Device templates or device aliases can be marked for deletion using the Delete function. Marking a template or alias as deleted retains the data for this device until a pack of the database is executed. Note that once a template or alias is marked as deleted, it cannot be used to generate tests (or add or update) unless the device is undeleted.

The Delete menu offers you the choice of marking templates or aliases. Marking a template for deletion schedules all device models and all device aliases for removal by the next pack operation. Marking an alias for deletion schedules only that one alias for removal by the next pack operation.

The Pack menu controls deletions from the library database with the following commands:

- Do pack: To remove devices marked for deletion and reduce the size of the database
- Restore old db: To restore previous version of database
- Remove back-up: To remove the back-up performed automatically when packing database with Save Back-Up Files selected
- Options: To save back-up files, restore on failure, and check packing process

Be aware that Pack requires several megabytes of memory, particularly if the old library is to be retained; therefore, you should use Pack while you still have adequate memory available.

To determine what templates and aliases will be removed during the pack operation, you can generate a list specifying that information. Refer to the List/Options section below.

IMPORTANT: Pack deletes all revisions of templates except the currently active one and the Teradyne revision.

To mark a template or alias as deleted

- 1 Select Delete from the Template menu to mark a device for deletion.
A pop-up window asks whether you want to delete a template or an alias.
- 2 Select Template to delete a template and all of its aliases or Alias to delete a single alias.
Another window appears asking you to enter the device name.
- 3 Enter the name of the device and click OK.

To pack the template library database

- 1 When you have marked all of your choices, select Options from the Pack menu.
The Select option(s) for packing window appears listing the following options:
Save Back-up Files
Restore on Failure
Pack Checking

All options are selected by default. If you do not want the program to perform any or all these procedures click the appropriate check mark(s) to toggle them off. Then click OK or Cancel.

IMPORTANT: It is important to check Options because, for example, if the Save Back-up files option has not been selected, the old database will not be saved, and should you want to restore it, you will not be able to after the pack operation is complete.

- 2 Select Do pack from the Pack menu.

The Database Pack Status window appears to apprise you of the progress of the packing procedure and notify you that a copy of the pre-packed database is being backed up if Save Back-up Files is selected in the Pack Options menu. The packing process takes several minutes. When the process is complete, press any key to return to the Template menu. To abort the process press ESC. When the pre-packing database is restored, you can press any key to return to the Template menu

DOS Command Equivalents for Pack Options

```
18xx /T -l<lus> -p[brc]
```

where b represents backup, r represents restore, and c represents check.

Restoring Library Database

You can restore a library database to the prepacked version using Restore old db from the Pack menu. The restore function does not work if you have deselected Save Back-up Files in the Pack/Options menu.

IMPORTANT: If, when you try to pack the database, an error message appears saying that the template database is old, you have probably tried to restore a D.X or earlier version of the library database which is incompatible with the E.0 or later database.

Undeleting Templates and Aliases

At any time prior to packing the library, the templates or aliases which have been marked for deletion can be “unmarked” by using the Undelete command.

To undelete a template or alias

- 1 Select Undelete from the Template menu to unmark a device for deletion.
A pop-up window asks whether you want to undelete a template or an alias.
- 2 Select Template to undelete a template, or select Alias to undelete an alias.
Another window appears asking you to enter the name of the device.
- 3 Enter the name of the device and click OK to confirm the undelete of the template or alias, or click Cancel to stop the procedure.

Accessing Template Revisions

Use Revision to access any of the template revisions in your database.

To select a template revision which is not currently active

- 1 Select Revision from the Template/Utility menu.

The Altering Template Revision Menu appears.

Altering Template Revision Menu

Enter Existing Device name:

Select one hi-lighted type of template:

- ☐ Gray Code Template
- ☐ Vector Template
- ☐ Vector Snap Shot Template
- ☐ Analog Template

OK Cancel

Enter Device Name and Press <Enter>

- 2 Type the name of the device whose template you want to access in the Enter Existing Device name field and press Enter.

All the types of templates existing for that device name become highlighted, and the topmost one has a check mark in the parentheses.

- 3 To select a highlighted template and deselect the one checked, click in the parentheses of the other template type and press OK.

Even if you do not want to change the selection, press OK to go to the next window.

A window appears listing the revisions and the dates they were created. A check indicates the active template.

Template Revision Listing

Device Name: 14000
DUAL 3-INPUT NOR GATE AND INVERTER

Template Revisions follow, select the desired active revision:

- ☐ Tue June 22 20:21:40 1998
TRANSLATED FROM BO.X1 TEMPLATE LIBRARY
- ☒ Thu May 05 14:10:27 1998
TRANSPORTED TO DATABASE
- ☐ Fri June 18 14:10:51 1998
TRANSPORTED TO DATABASE

OK Cancel

- 4 To select a different revision of the template, click in the parentheses next to it.

- 5 Press OK to accept the change or Cancel to ignore it.

When you generate a test, that revision is the one the program uses.

IMPORTANT: Packing a library eliminates these choices.

Modifying the Template Library Database

To modify the template library database, you must have access to the library database file. The database is in binary form. However, functions in the system software which also have a DOS command format enable you to convert pertinent data from the binary file, ALIAS.LNK, to an ASCII equivalent called ALIAS.CFG, located in the directory associated with whichever library you are using, (the System, User, or Local Library).

This ASCII equivalent, created by using the Export Aliases selection from the Templates/Utility menu or the tdb_menu -wa command in DOS, allows you to edit the template library database to facilitate

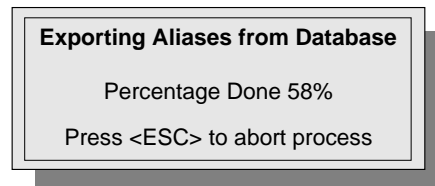
- calculations for backdrive analysis by expanding the ALIAS.LNK file thus allowing you to store more test information per alias, and
- modifying information in the template directives such as forcing voltage or bond wire type as may be required by driver/receiver 2 boards and the program generator.

Once you have created the ALIAS.CFG file and edited it using vi or another editor of your choice, you must update the binary file by using the software's Import Aliases or the DOS command tdb_menu -ra.

Accessing Aliases From 18xx Environment . The menu selections Export Aliases and Import Aliases enable you to convert (write) the ALIAS.LNK file to its ASCII equivalent and then to read the modified information back into the binary file.

To modify ALIAS.CFG

- 1 Select Template from the Main menu.
 - 2 Select the library or your choice: System, User, or Local.
 - 3 Select Export Aliases from the Utility menu.
- A statistics window appears.



The window indicates the progress of the conversion. When the conversion is complete, the window disappears.

- 4 Select Utility/DOS Shell from the Main menu.
- 5 At the DOS prompt, change to the directory where the chosen library is located.
- 6 Change directory to .\TTM.
- 7 Edit ALIAS.CFG using your favorite editor.
- 8 Save your edits and exit from DOS to the 18xx environment.
- 9 Select Import Aliases from Templates/Utility.

A statistics window appears indicating the progress of the update to ALIAS.LNK. When the window reads 100%, the template database has been updated with the modified information.

In addition a message window also appears above the statistics window to report the number of devices processed in from the database.

If there are errors reported in the file ALIAS.ERR, a message notifies you.

From DOS Shell. The ALIAS.CFG file is in <system, user, loc> library path.

IMPORTANT: When you are modifying the ALIAS.CFG file, the library defaults to the system library.

To execute the write function from the DOS shell,

- 1 Select Utility/DOS shell from the Main menu.
- 2 To execute the write function, at the DOS prompt type
18xx /T -l<lus> -wa

A statistics window appears indicating the progress of the conversion of the binary ALIAS.LNK to the ASCII ALIAS.CFG. When the conversion is complete, the window disappears.

- 3 Edit ALIAS.CFG using your favorite editor
- 4 Make and save your edits.
- 5 To execute the read function, type
18xx /T -l<lus> -ra
- 6 When the statistics window indicates that the database has been updated, exit from DOS to the 18xx environment.

Example of an ALIAS.CFG File

The following is a partial example of an ALIAS.CFG file. You can edit the Device, Bond wire, Average current (AVG I), Technology (TECHNO), Operating/Force Voltage (OP/FORCE VOLTAGE), and Oscillator (OSC) attributes. These attributes are of importance for customizing templates for backdrive analysis and for the driver/receiver 2 board.

The comments, indicated by the pound sign (#), detail the parameters of the attributes. The number in brackets indicates the number of alphanumeric characters you can type in each field.

```
#      Listing in ALIAS.CFG of all devices in 18xx Template Database
#
# Bondwire type: ALUM, GOLD, or PLAS
# [ALUM: aluminum bondwires in hollow packages
#  GOLD: gold bondwires in hollow packages
#  PLAS: gold bondwires in plastic packages.]
# If avg I is <= 0, a default value is assigned,
#   based on the technology.
# If the forcing volt is blank, it's = operating volt.
#
# Fields (num in [ ] is field width):
# Device[16]      Bond  AVG I  TECHNO[8] OP/FORCE  OSC[5]
#                wire  (mA)[6]          VOLTAGE
#                [5]                      [8]
#
1310              ALUM    790.0  ASTTL    5.0
23128             ALUM    790.0  ASTTL    5.0
23129             ALUM    790.0  ASTTL    5.0
23C128            ALUM    790.0  ASTTL    5.0
27128             ALUM    790.0  ASTTL    5.0
27C128            ALUM    790.0  ASTTL    5.0
```

```

613128          ALUM   790.0  ASTTL   5.0
7414            ALUM   790.0  ASTTL   5.0
92128           ALUM   790.0  ASTTL   5.0
U17             ALUM   790.0  ASTTL   5.0
#
# Some statistics:
# -----
#
# Total of 10 device names in the database; all are active.

```

Listing Aliases and Templates

If you want to see a complete list of devices in the database, select List from the Template menu. You can select

- All Aliases For list of all aliases
- All Templates For list of device commentaries
- One Template For one device commentary
- Options For customizing template commentaries

Note that all listings are displayed in a browser window. The data from the most recent listing is also retained in a file called PRINT.DAT in the current library's directory. This file can be sent to a line printer using DOS commands or viewed using any ASCII file editor.

All Aliases

For a list of all aliases select All Aliases from the List menu. A process window appears showing the running percentage until the compilation is complete. Then the list of all devices in the template database appears on screen. Use the Page Up, Page Down, Home and End keys to navigate through the list. Press ESC to exit from the list and return to the Templates menu.

All Templates

For a listing of the commentaries of all template groups in the Template database select All Templates from the List menu. A process window appears showing the running percentage until the compilation is complete. Then commentaries about each of the templates in the database are listed on screen. Use the Page Up, Page Down, Home and End keys to navigate through the list. Press ESC to exit from the list and return to the Templates menu.

The information listed is similar to that in the example below.

```

Desc: Octal Transceiver, Tri-State
Template Types: Digital
Template Modified: Yes
Template Deleted: No
Number of Pins - Vector Template: 20
Digital Template Model Revision Data:
Thu May 9 14:26:44 1996 TRANSPORTED TO DATABASE
Template Aliases:
74245 AM 74245

```

You can specify the type of information provided in the listing using the Options command. See below for information about the options available.

One Template

Select the One Template command to obtain commentary information for a single template. The List Device window appears. Enter the template name and click OK or Cancel. Press ESC to exit from the list and return to the Templates menu.

You can also specify the type of information provided in a listing for one template. See below for information about the options available.

Options

To customize the information provided in the commentary listing select Options. The Select Option(s) for Listing window appears offering you the following choices.

```
Show Template Deleted Flags
Show index number of each template
Show Device Descriptions
Show Types of Templates Available
Show Template Modified Flags
Show Number of Pins
Show Revision Date and Text
Show Templates Aliases
Show the Number of Template Groups
Show Status of Template Groups
```

Click in the parentheses to select or deselect an option.

The Show Template Deleted Flags option is useful to list deleted devices. All device aliases that have been marked for deletion are displayed within less than and greater than symbols. In the commentary shown above, if template 74245 is active but its alias AM74245 is marked as deleted, the display would appear as shown below.

```
Template Aliases:
74245  <AM74245>
```

To show all templates and aliases that are marked for deletion

- 1 Select the Options window from the List menu.
- 2 Turn on Show Template Deleted Flags and Show Template Aliases. Turn off all other options.
- 3 Select List All Templates.
- 4 View the resulting list, looking for deleted aliases and templates.

Translating Vectors

The system software enables you to translate both Jedec and TDS vectors to usable ASCII vector format. The Jedec function is accessible from the Templates menu. Access to the TDS translator is through the MOS directory and is run as a DOS utility program separate from the 18xx operating environment.

Translating Jedec Vectors

The Jedec menu offers two commands

- Translate Translates vectors from Jedec format to 18xx ASCII vector format.
- Options Enables you to add statistics and commentary to the file, and to customize the extension appended to the translated file.

To specify options and translate Jedec vectors which are in the current library's directory

- 1 Expand the Jedec menu and select Options.
The Select option(s) for JEDEC Xlate window appears offering the following choices:
 () Add Statistics to vector.
 () Add Comments to vector.
 (jdc) Extension for Jedec files.
- 2 Select the options you want to add. If you want to change the extension, enter a different three-letter combination.
- 3 Click OK.
- 4 Select Translate.
The TTM directory window appears showing all files in the TTM directory with the specified file extension.
- 5 Select the Jedec vector you want to translate.
Selecting the Jedec vector automatically translates it into 18xx ASCII vector format.
- 6 To add the translated Jedec vector to the template library, use the ASCII Vector command from Templates/Add

Translating TSSI TDS Vectors

The TDS18XX function translates TSSI TDS ASCII output vectors to 18xx ASCII format. The resulting .asc file can be then added to the Vector template library by using the Add/ASCII Vector function. The required input is a TSD vector file. The Pin Number Name file, which contains the Pin Number, Name and type for the input file, is optional.

To access the TDS translate menu

- 1 Go to the directory containing the TDS ASCII data file you want to translate.
- 2 Type
`TDS18xx <input file>`
 where <input file> is the name of the vector you wish to translate. To get a full syntax explanation, type only TDS18xx at the DOS prompt. See the end of this section for the syntax explanation.
- 3 A message similar to the following appears.

```
Device has at least 38 pins.
This total does not include unused pins such as ground and
power.
Enter total number of Pins on the device: 38
```
- 4 If the number displayed is correct, press Enter. If the number is not correct, enter the correct number and press Enter.
- 5 A window appears asking you if the pins are in numerical order.
 The default is Yes, the pins are in numerical order; No, if the pins are in logical groupings.
 If you press Enter to accept the Yes default a TDS Translate Table similar to the following appears.

Information in TDS files is not complete. TDS files contain the vector data, pin names and pin types. However there is no information on unused pins. And if pins are not in pin order, the pin numbers must also be corrected. Please fill in the pin name and number of the unused pins and check the pin numbers of all pins	Column	Pin Number	Pin Name	Pin Type
	0	1	CLKIN	I
	1	2	CLKOUT	O
	2	3	RESET	I
	3	4	DIRIN	I
	4	5	_DRIOUT	O
	5	6	SD7	IO
	6	7	SD6	IO
	7	8	SD5	IO
	8	9	SD4	IO
	9	10	SD3	IO
	10	11	SD2	IO
	11	12	SD1	IO
	12	13	SD0	IO
	13	14	_IOR	I
	14	15	_IOW	I
	15	16	_CS	I
	16	17	_PDOE	I
	17	18	SA1	I
	18	19	SA0	I
	19	20	PD7	I
Edit Save Convert Quit				

If the pins are in logical groupings and not in numerical order, type N (for No) and press Enter. A TDS Translate Table similar to the following appears.

Information in TDS files is not complete. TDS files contain the vector data, pin names and pin types. However there is no information on unused pins. And if pins are not in pin order, the pin numbers must also be corrected. Please fill in the pin name and number of the unused pins and check the pin numbers of all pins	Column	Pin Number	Pin Name	Pin Type
	0	6553	CLKIN	I
	1	6553	CLKOUT	O
	2	6553	RESET	I
	3	6553	DIRIN	I
	4	6553	_DRIOUT	O
	5	6553	SD7	IO
	6	6553	SD6	IO
	7	6553	SD5	IO
	8	6553	SD4	IO
	9	6553	SD3	IO
	10	6553	SD2	IO
	11	6553	SD1	IO
	12	6553	SD0	IO
	13	6553	_IOR	I
	14	6553	_IOW	I
	15	6553	_CS	I
	16	6553	_PDOE	I
	17	6553	SA1	I
	18	6553	SA0	I
	19	6553	PD7	I
Edit Save Convert Quit				

Using the TDS Translate Menu

The function menu on the left has four functions: Edit, Save, Convert, and Quit. To select a function use the left and right arrow keys or type the first letter of the function (e, s, c, or q) to highlight the desired function. Press Enter to invoke the function. The ESC key has the same function as Quit.

When the table is displayed, all fatal errors are displayed in red, cosmetic errors in gray, and valid entries in white. Pin numbers and types are essential in 18xx vectors. Errors in these columns are fatal and must be corrected if the translated 18xx vector is to be used in testing.

To make changes in the TDS Translate Table, select **Edit** from the menu and use the arrow keys to move in any direction. Use the Enter key to traverse the columns horizontally. The only columns that are active are Pin Number, Pin Name, and Pin Type.

Pin numbers must be unique and in the range 1 to 2048. With the exception of Gnd and Vcc, pin names are only symbolic and do not have to be unique, nor do you have to modify the names. The pin type must be one of the following: I (input), O (output), IO (input/output) or U (not used).

To modify pin numbers and names, move to the field and type in the correction. To modify the pin type move to the field and use the space bar to toggle through the predefined types. To return to the menu, press ESC.

The **Save** function saves the displayed data to a file with a .pnn extension.

To save data

- 1 Move the cursor to Save and press Enter.
A message similar to the following appears.
`Enter PNN filename: DATA.PNN`
- 2 To accept the file name, press Enter. To change it, enter a new file name and press Enter.

Convert uses data in the PNN table and the data in the vector data file to create the 18xx vector. The data is then stored to a file.

To convert a vector

- 1 Select Convert from the menu.
A message similar to the following appears.
`Enter vector filename: DATA.ASC`
- 2 If the name is acceptable, press Enter. If the name is not acceptable, type in a new file name and press Enter. The data is then saved to the specified file.

Selecting **Quit** terminates the program and returns you to the DOS operating system.

Adding the ASCII File to Library. To then add the ASCII file to the library

- 1 Move or copy the file to the \PGT\TTM directory using the appropriate DOS command.
- 2 In the 18xx environment, use the Templates/Add/ASCII Vector function as previously discussed.

Creating Templates

Templates stored in the template library—Sys, Usr or Local—provide essential information for the creation of test programs. Your template library database will include not only templates provided by Teradyne, but also your own custom templates. Templates you make may be analog, Gray code, vector model, or vector snapshots. The system software has a number of facilities for you to make templates or to convert templates from other formats to ASCII formats usable in the 18xx programming environment.

Briefly, the creation process for each of these template types is as follows:

Analog—Use Edit functions, then Tools/Create Template. Create Template puts the new template directly into the analog (ANTEMPL.LIB) library.

Gray Code—Same as analog, but adds to the Gray code (GCTEMPL.LIB) library.

Vector—Create an ASCII .ASC file by hand editing or by using a translator. Use the Templates/Add/ASCII Vector function to place it into the vector template (VRTEMPL.LIB) library.

Vector Snapshot—There are two ways to add vector snapshots.

- (1) Add as you add analog or Gray code templates.
- (2) Take an IVL vector file (.ivl format) and add it using Templates/Add/IVL Vector to the vector snapshot or vector image library (VRSNAP.LIB).

Analog Templates

An analog template is a test pattern of stimulus and measurement for an analog device that you save to the template library. Analog templates are available for passive analog devices and linear or power-on Step Worksheets.

Be aware that analog stimulus, measure, and guard points can be referenced by pin or tester node number. If a template is created for a Step Worksheet that references tester node numbers rather than component pins, then those node numbers become part of the template, and each time the template is used, it generates a test with those node numbers. Normally, templates should be created from Step Worksheets that reference component pins to allow each test generated from this template to use the node numbers specified in the Component Properties portion of the Step Worksheet.

IMPORTANT: The Z1800 software tool set is designed to restrict specific tests to certain sections—for example, board power tests to the Brd_Power section. However, some tools most notably Analog Templates, allow you to bypass this structure.

Teradyne does not assume liability if you import analog templates to inappropriate sections, for example, if you were to put an analog template for a power test in the Linear section.

To create an analog template:

- 1** Create and edit the component's Component Properties in the Step Worksheet.
Be sure to enter the appropriate component identifiers for your new analog template. Make sure the Name field in Component Properties is no more than 16 characters with no embedded spaces and contains the new name of the template. Put any useful comments in the Desc field since this field will also appear in the template listing. However, it is not mandatory to enter data in the Desc field.
- 2** Move the mouse to the Device Type field in Component Properties.
- 3** Click the Device Type field.
A pop-up window with a list of the available device types appears. Analog Template will be at the top of the list.

- 4 Click Analog Template.
- 5 Select the Tools/Create Template.
- 6 Answer Yes to the prompt to create the analog template.
The new template is added to the library.

To use an analog template:

- 1 In the IPL, make sure that the component line refers to the part as ATMPL token type and then the appropriate name (Name field).
- 2 Run Pgen as you would for normal parts in the library.

You can also perform an instant generation process within a Step Worksheet.

- 1 Type the template name into the Name field.
- 2 Select Analog Template from the Device Type field.
- 3 Select Get Template Pins from the Tools menu.
- 4 Select Tools/Edit Device Nodes and fill in the node numbers.
- 5 Select Generate Test from the Tools menu.

The process of creating a test from an analog test section is similar to creating a test from a digital template. When using an input list you must specify the test section, however, via one of the link tokens shown in Table 8.2.

Gray Code Templates

The process for making a Gray code template is similar to that for making analog templates. However, in editing the Step Worksheet for a new Gray code template, Gray code groups must have no tied connections and must be tested the same way. Only after you have created the new template, can you modify it by adding, for example, a tied configuration.

A group is a subset of nodes on an IC package used to test one logical element, such as a single NAND gate in a quadruple NAND gate package. Typically, a group describes all inputs and outputs of a single element. A pin can be assigned to a single group, to multiple groups, or to all groups.

The new revision of the template will carry all of the constraint handling capabilities of the base template. This is to say that the new version will be a superset of the base template. The new configuration will be in addition to the existing template capabilities.

When you use Create Template, you will notice that several new revision time stamps will appear when you list the template revision history (Main menu/Templates/Revision). This information also appears if you just do an ordinary List Template (Main menu/Templates/List). Multiple revision stamps appear because as the Create Template process occurs, the algorithm adds revisions for each burst group in the Gray code test as the new template version is being constructed. You will see this happening as the test system goes through the Create Template process and asks you whether or not to update each group one at a time. If a Gray code test has 4 burst groups, the system will query you 4 times asking whether or not to update each group. The first revision will have one group updated, the second revision will have two groups updated, the third revision will have three groups updated. When the Create Template process is complete, only the last of the 4 revisions shown will contain all of the revision information for all 4 burst groups. Usually, only the last version is of interest, since the others are just intermediate steps in the Create Template process.

Wiring constraints are defined by node numbers, not by Tied syntax in the Step Worksheet. The template library does not key anything off of Tied in the Step Worksheet. This means that without explicit node numbers, Create Template cannot work properly. Use 9999 if there are no valid node numbers for some pins, but do not use 9999 for all pins. You may use 9999 for all but one pin if you wish. Enter node numbers normally using the Tools/Edit Device Pins function.

New wiring configurations (defined via node numbers) do not prompt the operator, (“do you wish to update this group for this wiring constraint”) during Create Template. It assumes a new wiring constraint obviously must be added to the template. During Create Template, those groups which are setup in the Step Worksheet with wiring configurations that already match those existing in the template prompt the operator with (“do you wish to update this group for this wiring constraint”).

Ties between groups are disallowed during Create Template. Error messages will result if you attempt this. This must be handled manually at this time.

You must always create an unconstrained version (i.e. normal template) before adding a constrained version even if you only want to template a constrained configuration of the device. To meet this criteria, you must temporarily make all nodes unique then Create Template, then go back to your constrained test and Create Template again.

To create a Gray code template

- 1 Access a digital Step Worksheet from the Main menu by selecting the appropriate board program then Edit/Digital/Components. Select the digital ID from the Component Select window.
The Step Worksheet appears.
- 2 Fill in the Component Properties portion of the Step Worksheet.
 - a) Make sure the Name field contains the new name of the template. It can have no more than 16 characters with no embedded spaces. However, a dash “-” is allowed.
 - b) Put any useful comments in the Desc field since this field will also appear in the template listing. However, it is not mandatory to enter data in the Desc field.
- 3 Select the Tools/Get Template Pins function to verify that this is a new template or get the number of pins from an old template.
If the template does not exist, you will need to move the mouse to the Number of Pins field without clicking and type in the new number of pins.
- 4 Click the Number of Pins field.
A pop-up window appears listing the node entries for each pin. Make sure that each pin has a unique node number. In addition, make sure that there are no blank node entries. If you do not know the node number, insert 9999.
- 5 Click the Device Type field.
A pop-up window with a list of the available device type selections appears. Gray Code is at the top of the list.
- 6 Click Gray Code to select it.
The pop-up window closes as soon as you make your selection.
- 7 Make sure that each group has a unique group number.
For further information about editing the Step Worksheet see Gray Code Step Worksheet in chapter 3, “Program Editing” and the **Z1800-Series Component Test Reference**.
- 8 Select Tools/Create Template.
- 9 Answer Yes to the prompt to create the Gray code template.
The new template is added to the library.

Vector Image Templates

A vector image is an unconstrained vector added to the library. From the editor a vector test can be snapshot into the library using the same basic process as defined for analog and Gray code templates. Note the following differences:

- Select Vector Image from the Device Type pop-up window.
- You have to open and close the Test Properties portion of the Step Worksheet area using the toggle arrows.
- When a Vector Image is added to the library, any guards associated with it will be removed. Only the test patterns are saved.

Creating Models Using ASCII Vector File Structure

The ASCII vector file structure is a format that allows the software system to create template library models from, for example, JEDEC or TSSI input files (after translation). ASCII vector files are given .ASC extensions.

The file consists of a Pins section, optional Test Configuration section, and one or more of the following: Vector section, Guard section(s), and/or a Disable section.

All text including and following single quotes is treated as comments. However, comments are stripped out when devices are added to the library. White space and blank lines may be used freely since the parser ignores such characters.

Note that all text may be in upper- or lowercase.

NPins Section

```
<npins>-->
    NPins = <pin count>;
    <pin record> [<pin record>] ...

<pin count>  -->  <integer>

<pin record>-->  <pin number> ',' <pin name> ',' <pin type> ';'
<pin number>-->  <integer>
<pin name>-->    <string>
<pin type>-->    I | O | IO | OI | U | HO | HIO | AO | AI
```

After any comments, the file begins with the keyword NPins followed by an equal sign and the number of pins in the device. Next there should be a single line entered for the definition of each pin. The sequence in which the pins are listed indicates the column position where this pin will reside in the vector data section.

The format for each is <pin number>, <pin name>, <pin type>.

- Pin number is the actual device pin number.
- Pin name is a string of up to 9 characters. It is case sensitive.
- Pin type includes one or two ASCII characters representing the pin type.

Character	Description
I	Stimulus pin
O	Measure pin
IO	Stimulus/measure pin
OI	Stimulus/measure pin
U	Ground or power (unconnected signal pin)
HO	Homing loop measure pin
HIO	Homing loop stimulus/measure pin
AO	Analog measure pin for mixed mode testing
AI	Analog stimulus pin for mixed mode testing

Data for each pin except those marked with U is represented in the data section.

When analog stimulus/measure pins are present for a Mixed Mode test, additional information is required. When an analog Stim pin (AI) is used, then the STIMV test configuration statement is required. This statement allows the user to set the voltage value, stimulus type, resistor value, and wait time for the AI pin.

When an analog measure pin (AO) is used, additional data is required for each M state character used in the vector pattern data. The data for the AO pin(s) will be between the pattern information and the terminating semicolon for the state that the measure (M) is in. The analog measure data has the following format:

```
MEASV <pin # or name>, <voltage_range><meas_type>
```

where meas_type is DC, Peak, RMS, or Pk-Pk.

Both AI and AO pins have limitations. AI and AO pins cannot be used in disable or guard vectors. Only one AI and a maximum of 10 analog statements is allowed for each vector test step, that is, a maximum of one AI pin plus nine analog measure states (M) used on the AO pin are allowed.

See the Tabular/IVL Format section below for examples of IVL states for Mixed Mode test.

Test Configuration Section

```

<config> -->
    [MAXRATE <rng>]
    [MDELAY <del>]
    [THRESH <val> | THRESH LO <val> HI <val>]
    [TERM None | TERM 1K Pull-up | TERM 500 Pull-up |
     TERM 1K Pull-down |
     TERM 500 Pull-down | TERM 1K Term]

    [CLOCK <src>]
    [MAXHOMING <hcount>]
<src>          --> INTERNAL | EXTERNAL

<rng>          --> float {2.0Mhz - 1.221Khz}
<del>          --> integer { 100ns - 800us}
<val>          --> float  {-2.0 - 10.0}
<repeatcount> --> integer {1 - 32767}
<hcount>      --> integer {1 - 32767}

[<ana stim>]
<ana stim>     --> StimV <pin number | pin name>,<ana val>
                <stim volt scale><ana Stim type>
                res<resistor value>wait<wait value>ms

<ana val>      --> float {-1000.0 - 1000.0}
<stim volt scale> --> 'mV' | 'V'
<ana stim type> --> 'DC' | 'AC 1' | 'AC 2' | 'AC 3'
<resistor value> --> '0 O' | '10 O' | '100 O' | '1 K' |
'10 K' | '100 K' |
                '1 M' | '10 M'
<wait value>   --> integer {0 - 10000}

```

All test configuration statements are optional. If any of the statements are omitted, the variables assume their default values. The default values are

```

MAXRATE 2 MHz
MDELAY 500 ns
THRESH 1.6
TERM NONE
CLOCK INTERNAL
MAXHOMING 100

```

The **MAXRATE** statement is for the maximum clock rate, which can be a value in the range of 2 MHz to 1.221 KHz.

The **MDELAY** statement is for measurement delay, which can be a value in the range of 100 ns to 800 µs.

The **THRESH** statement is for threshold. For single threshold, the syntax is THRESH followed by value. For dual threshold the syntax is THRESH followed by LO with a value and HI with a value. The value can be in the range of -2.0 to 10.0 V.

The **TERM** statement is for terminators. TERM can be followed by one of the following: NONE, 1K Pull-up, 500 Pull-up, 1K Pull-down, 500 Pull-down, or 1K Term.

The **CLOCK** statement is for clock source which can be either INTERNAL or EXTERNAL.

The **MAXHOMING** statement defines the maximum number of times any one homing loop will repeat. If the homing loop does not reach its desired state by this count, the test will be aborted.

Data for the analog stimulus pin can be anywhere in the Test Configuration section.

Example: Stim V pin_2, 10.00 mv AC1 res 10 O wait 0 ms

The comma separating the <pin # or name> and the <value> is required because pin names can have spaces.

Vector Section

```
<vector> -->
    VECTOR ' ';

    <data>
    end vector;
```

The Vector section contains only test vector data.

The vector data section is made of constructs and optional constraints used to identify blocks of patterns that are combined to create a vector test.

The general format for the vector data section is

```
begin construct [constraint];
    (patterns)
end construct;
```

Rules for Accepting Vector Patterns During Generation

The vector generator accepts patterns from a vector template based on two criteria, constraint checking and pattern checking. Constraint checking is the more obvious of the two. It requires that for a given begin/end group of patterns to be accepted, the explicit constraint must be satisfied. Explicit constraints are CONNECTED, HIGH, LOW, INDEP, IRREL, UNAVAIL, AVAIL, and TRI.

See the section on Constraints for a definition of these constraints.

The constraints are evaluated at the line of the template which they were encountered, and the result of the constraint evaluation is either true or false. If the evaluation is false, this group of patterns is rejected. If the evaluation is true, the processing of patterns continues.

Pattern checking occurs for each pattern within the vector. The goal of the pattern checking is to verify that the tester will be able to achieve the vector patterns entered. It is called a pattern violation when the pattern checking finds a pattern that the tester cannot achieve. Possible pattern violations are

- Trying to measure a ground node High or stim High.
- Trying to measure a power node Low or stim Low.
- Trying to stim an unused node (9999) either Low or High.
- Trying to apply an analog stim to an unused, ground, or power node.
- Trying to do an analog measure on a ground or power node.
- Tied pin violations. All nodes of a tied group must be driven to or measured at the same state. For example, specifying an H and L on the same node constitutes a violation. Note that pattern checking ignores X's and Z's. For example, an H and a Z on the same node is acceptable.

When you use IVL patterns within a vector template, be aware that vectors keep their last state. For example, if pins 1 and 2 are tied, changing the state of pin 1 does not affect the state of pin 2. It is best to initialize the state of all tied pins for sequences which you want to be accepted for connected constraints.

Constructs

The constructs are

SET
 SEQUENCE (or SEQ)
 SELECTONE
 ENTRY
 SELECT
 BLOCK
 REPEAT
 HOMING

The following table shows the rules for nesting these constructs. Notice that all constructs must appear within a SET.

Construct	May contain these constructs
SET	SEQUENCE, SELECT, SELECTONE, HOMING
SEQUENCE	SEQUENCE, SELECT, SELECTONE, HOMING
SELECTONE	ENTRY
ENTRY	SEQUENCE, SELECT, SELECTONE, REPEAT
SELECT	BLOCK
BLOCK	SEQUENCE, SELECT, SELECTONE, REPEAT
REPEAT	SEQUENCE, SELECTONE

SETs are the highest level construct and will collapse if any pattern within the SET causes an abort. The word collapse means that all patterns that appear between the begin SET and its corresponding end SET construct will be ignored, and will therefore not appear in the vector test generated by this template. Groups of patterns can be separated using the begin/end SEQUENCE construct. If a SEQUENCE aborts, only those patterns within the begin SEQUENCE and end SEQUENCE will collapse. The rest of the SET is unaffected. This is a method for putting a fence around patterns.

SET follows the rules below.

- SETs cannot be nested.
- A SET cannot appear inside any other construct, nor can any other construct appear outside of a SET.

A **SEQUENCE (SEQ)** is used to isolate a group of patterns from the rest of the construct it is nested within. The SEQUENCE will collapse to nothing if anything within the SEQUENCE aborts. If a SEQUENCE aborts, it will not affect any constructs it is nested within. You may use the keywords SEQUENCE or SEQ.

The **SELECTONE** construct is used to choose a group of patterns from multiple groups. The first group which does not violate any constraints will be chosen. The groups of patterns are delimited by the begin/end ENTRY construct. An ENTRY construct is the only one allowed to

appear as the next level within a **SELECTONE**, but other legal constructs may appear within an **ENTRY**. Each **ENTRY** is evaluated in turn and the first one that does not violate any constraints is used. Therefore, the most desirable choice should be the first **ENTRY** and so on. If none of the **ENTRY**s are accepted, then the **SELECTONE** did not generate any patterns, and it will collapse to a **SEQUENCE** or **SET** boundary.

The **ENTRY** construct is used only to delimit pattern groups within a **SELECTONE**. Refer to the table above for constructs which can be nested within an **ENTRY**.

The **SELECT** construct is the most complex and powerful of all the constructs. From a syntactical point of view, **SELECT** looks exactly like **SELECTONE**. **SELECT** replaces **SELECTONE** in the syntax, and **BLOCK** replaces **ENTRY**. The processing of **BLOCKS** differentiates **SELECT** and **SELECTONE**. In **SELECTONE**, only one **ENTRY** is accepted and is inserted in a linear fashion to replace the entire **SELECTONE** construct. In **SELECT**, each **BLOCK** that is accepted is inserted in place of the **SELECT**, along with all the patterns both before the begin **SELECT** and after the end **SELECT**. This process holds true for all levels of nested **SELECT**s as well. As a result **SELECT** gives you the opportunity to create many patterns with just a few constructs. **SELECT** can be viewed as a mechanism which loops on **SET**s, each pass selecting a different **BLOCK** and combining it with the rest of the **SET** until all **BLOCKS** have been used.

See the example section for the use of **SELECT**.

The **BLOCK** construct is only used to delimit pattern groups within **SELECT**. Refer to the table for constructs which can be nested within **BLOCK**.

The **REPEAT** construct accepts a count which defines the number of times the patterns between the begin **REPEAT** and end **REPEAT** are to be duplicated. A syntax error is reported if a **SELECT** construct is used within a **REPEAT**. Constraints can follow the count. The count can be in the range of 1 to 32767.

The **HOMING** construct defines a homing loop. A homing loop is a set of patterns which are looped on at test execution time until all the measurements within the begin/end patterns pass or **MAXHOMING** count loops is reached. A syntax error is reported if a **SELECT** construct is used within a **HOMING** loop.

Constraints

Available constraints are

CONNECTED	The named pins must appear on the same node.
HIGH	The named pins are tied to one of the power nodes defined by the nodes associated with the GND, PWR5, PWRA, PWRA_F, PWRB, PWRB_F, PB, and PBUS input list tokens (device types).
LOW	The named pins are tied to one of the ground nodes defined by the nodes associated with the GND, PWR5, PWRA, PWRA_F, PWRB, PWRB_F, PB, and PBUS input list tokens (device types).
INDEP	The named pins are on a valid node but are otherwise unconstrained, i.e., not on a power bus and not tied.
IRREL	The named pins are not constraint-checked.
UNAVAIL	The node number for this pin is out of the range of the tester.
AVAIL	The named pins are on a valid node.
TRI	The named pins must be on a valid node where the output is tristatable.

Examples of constraint usages:

```
begin SET (CONNECTED (1,2,3));
    <patterns>
    begin SEQ (INDEP (5,6,7,14));
        <patterns>
    end SEQ;
end set;
```

Guard and Disable Sections

```
<guards> -->
    GUARD <guard pin> ';'

    Pins = <pin list>

    <data>
    end guard;

<disable> -->
    DISABLE ';'

    Pins = <pin list>

    <data>
    end disable;
```

The Guard and Disable sections contain a pin list followed by vector data. The pin list begins with the keyword PINS and is followed by an equal sign and a parentheses-delimited, comma-separated list of device pin numbers or names. (See the examples in the format specification section.) The pin numbers (or names) reference the pin data defined above in the pin record section. For Guards and Disables there is a one-to-one relationship between the pin numbers (or names) position in the pin list and its data in the Guard and Disable vector patterns. If pin number 1 is the first pin in the pin list, the data for this pin will be in column 0 of the following data.

The Guards section defines the inputs required to guard the individual outputs of this device. The syntax is the keyword GUARD followed by the device's output pin number, the list of stim pins that will be driven, and the patterns required to achieve this guard.

The Disable section contains the data patterns required to disable this device. Disables have a DISABLE keyword followed by a pin list and data.

Tabular/IVL Formats

Each pattern of the vector data in the Vector, Guard or Disable sections may be either in the tabular format or in IVL (Incremental Vector Language) format. In the tabular format, for each pattern there is a data character that defines the state for each active pin during that pattern. In the Vector section, the states for the pins in each pattern appear in the same order that the pins appear in the Pins section. In the Guard and Disable sections, the pin states appear in the same order as in the Pins list at the beginning of that section.

In the IVL format, only those pins that change state in any given pattern are mentioned. In this case, each state data character is followed in parentheses by a list of the pins that are to take on that state in the current pattern. Each pin may be identified either by its name or by its number (first field in the Pins section). Any pin not mentioned in the first pattern in a SET will take on a state of X until/unless it is mentioned in some later pattern.

In both the tabular and the IVL formats, each state/pattern of vector data is terminated with a semicolon (;). The set of valid data characters are L, H, Z, D, U, X, -. The first three (L, H, Z) are for stim pins, the next three (D, U, X) are for measure pins, and the last (-) is used in tabular

format only and retains the previous state. An asterisk (*) character after the semicolon (;) identifies that state/cycle as a global ignore, which implies that the measures will not be executed for that state.

The following is an example of an IVL state with a single analog measure such as you might encounter in a Mixed Mode test.

```
H(1,2,3,4,5,8)L(6,7,10)M(9) MeasV D1, 9.5 to 10.5 uv RMS;
```

An IVL state with multiple analog measures is shown below in table form.

```
HHHHHLLHMM MeasV pin_9, 9.5 to 10.5 uv RMS,  
MeasV pin_10, 4.5 to 5.5 uv pk-pk;
```

Format Specification

The following specification is in BNF format—a standard way to represent a language.

```
<npins>-->  
  NPins = <pin count>;  
  <pin record> [<pin record>] ...  
  
<config> -->  
  [MAXRATE <rng>]  
  [MDELAY <del>]  
  [THRESH <val> | THRESH LO <val> HI <val>]  
  [TERM None | TERM 1K Pull-up | TERM 500 Pull-up |  
    TERM 1K Pull-down |  
                                TERM 500 Pull-down | TERM 1K Term]  
  
  [CLOCK <src>]  
  [MAXHOMING <hcount>]  
  [<ana stim>]  
  
<vector> -->  
  VECTOR ';' '  
  
  <data>  
  end vector;  
  
<guards> -->  
  GUARD <guard pin> ';' '  
  
  Pins = <pin list>  
  
  <data>  
  end guard;  
  
<disable> -->  
  DISABLE ';' '  
  
  Pins = <pin list>  
  
  <data>  
  end disable;  
  
<data> -->  
  <set> [<set>...]
```

```

<pin count> --> <integer>

<pin list>--> '(' <pin number> | <pin name>
               [',' <pin number> | <pin name>...] ')' ';'

<pin record> --> <pin number> ',' <pin name> ',' <pin type> ';'
<pin number>--> <integer>
<pin name>--> <string>
<pin type>--> I | O | IO | OI | U | HO | HIO
<guard pin>--><integer>

<set>         --> begin SET [<constraints>] ';'
                  <constructs>
                  end SET ';'
<constraints>--> '(' <constr_type> <constr_pins>
                  [',' <constr_type> <constr_pins> ...] ')'

<constr_type>--> CONNECTED | HIGH | LOW | INDEP | IRREL | UNAVAIL
                  | AVAIL | TRI

<constr_pins>--> '(' <pin number> [',' <pin number>...] ')'

<constructs>--> <seq> | <selectone> | <select> | <patterns> | <repeat>

<seq>--> begin SEQ [<constraints>] ';'
          <seq> | <selectone> | <select> | <patterns>
          end SEQ ';'

<selectone>--> <entry> [<entry> ...]

<select>--> <block> [<block> ...]

<block>--> begin BLOCK [<constraints>] ';'
          <seq> | <selectone> | <select> | <patterns>
          end BLOCK ';'

<entry>--> begin ENTRY [<constraints>] ';'
          <seq> | <selectone> | <select> | <patterns>
          end ENTRY ';'

<repeat>--> begin REPEAT <repeatcount> [<constraints>] ';'
          <cycle>[<cycle>...]
          end REPEAT ';'

<homing>-->begin HOMING ';'
          <cycle>[<cycle>...]
          end HOMING ';'

<patterns>--> <cycle> [<constructs>...]
<cycle>--> [<stim_resp>] [<measV_data>] ';' [<global ignore>]
<stim_resp>--> <tabular_stim_resp> | <incr_stim_resp>

<tabular_stim_resp>--> <stim_resp_char> [<stim_resp_char>...]
<incr_stim_resp>--> <incr_pin_spec> [<incr_pin_spec>...]
<incr_pin_spec>--> <stim_resp_char> <incr_pin_list>
<incr_pin_list>--> '(' <pin> [ ',' <pin>...] ')'

```



```

<stim_resp_char>--> <stim> | <meas>
<stim>--> 'L' | 'H' | 'Z' | '-'
<meas>--> 'D' | 'U' | 'X' | '-'
<global ignore>--> '*'

<src>--> INTERNAL | EXTERNAL

<rng>--> float {2.0Mhz - 1.221Khz}
<del>--> integer { 100ns - 800us}
<val>--> float {-2.0 - 10.0}
<repeatcount>--> integer {1 - 32767}
<hcount>--> integer {1 - 32767}

<ana meas>--> MeasV <pin number | pin name>,<meas range>
               <meas volt scale><ana meas type>

<meas range>--> <low val> to <high val>
<low val>--> <ana val>
<high val>--> <ana val>
<meas volt scale>--> 'mV' | 'mV' | 'V'
<ana meas type>--> 'DC' | 'peak' | 'RMS' | 'pk-pk'

<ana stim>                --> StimV <pin number | pin name>,<ana val>
                           <stim volt scale><ana Stim type>
                           res<resistor value>wait<wait value>ms

<ana val>                  --> float {-1000.0 - 1000.0}
<stim volt scale>          --> 'mV' 'V'
<ana stim type>            --> 'DC' | 'AC 1' | 'AC 2' | 'AC 3'
<resistor value>           --> 'Ø 0' | '1Ø 0' | '1ØØ 0' | '1 K' |
'1Ø K' | '1ØØ K' |
                           '1 M' | '1Ø M'
<wait value>               --> integer {0 - 10000}

```

ASCII Vector Format Examples

Vector Example

NPINS = 20;

```

1,CP,I;
3,nRE,I;
4,D3,I;
5,D2,I;
6,D1,I;
7,D0,I;
9,nZRO,I;
10,S0,I;
11,S1,I;
12,Y0,O;
13,Y1,O;
14,Y2,O;
15,Y3,O;
16,nOE,I;
17,CN,I;
18,CN4,O;
19,nFE,I;
20,PUP,I;

```

```
2,VCC,U;
8,GND,U;
```

```
Maxrate 2 MHZ
Mdelay 500 NS
Thresh LO 1.6 HI 1.6
Term NONE
```

```
Disable;
PINS = (nOE);
Begin SET;
H;
End SET;
End Disable;
```

```
Vector;
Begin SET;
```

```

'-----
'BEGIN SET;
'- OE_ low.
'
'/ THIS SET IS FOR nOE TIED LOW
'
'Initialize chip
'Select direct input, all pins high
HHHHHHHHHHXXXXXHUHH; 'P: 1
'CONSTRAINTS
HHHHHHHHHHXXXXLHUHH; 'P: 2
'check direct inputs to outputs
'write 0101
HHLHLHHHHXXXXLHDHH; 'P: 3
'read 0101
HHLHLHHHHUDUDLHDHH; 'P: 4
'write 1010
HHHLHLHHHHXXXXLHDHH; 'P: 5
'read 1010
HHHLHLHHHDUDULHDHH; 'P: 6
'write and read 1111
HHHHHHHHHHUUUULHUHH; 'P: 7
'check nZRO
HHHHHHHLHDDDDLHDHH; 'P: 8
HHLLLLHLHDDDDLHDHH; 'P: 9
HHLLLLHHHDDDDLHDHH; 'P: 10
'CHECK INTERNAL ADDRESS REGISTERS
HHHLHLHHHHXXXXLHDHH; 'P: 11
'SELECT IAR
HHHLHLHHLXXXXLHXHH; 'P: 12
'ENABLE IAR
HLHLHLHHLXXXXLHXHH; 'P: 13
LLHLHLHHLXXXXLHXHH; 'P: 14
HLHLHLHHLUDUDLHDHH; 'P: 15
HLLHLHHHLUDUDLHDHH; 'P: 16
LLLHLHHHLUDUDLHDHH; 'P: 17
HLLHLHHHLUDUDLHDHH; 'P: 18
'CHECK nRE FOR DISABLE
HHLHLHHHLUDUDLHDHH; 'P: 19
HHHLHLHHLUDUDLHDHH; 'P: 20
LHHLHLHHLUDUDLHDHH; 'P: 21
```

```

HHHLHLHHLUDUDLHDHH; 'P: 22
HHHLHLHLLDUUDLHDHH; 'P: 23
Begin SEQ;
'if CN != LL & CN != LH
    'CHECK CN
HHHLHLHLLDUUDLLDHH; 'P: 24
LHHLHLHLLDUUDLLDHH; 'P: 25
HHHLHLHLLDUUDLLDHH; 'P: 26
HHHLHLHLLDUUDLHDHH; 'P: 27
LHHLHLHLLDUUDLHDHH; 'P: 28
HHHLHLHLLUUDLHDHH; 'P: 29
LHHLHLHLLUUDLHDHH; 'P: 30
HHHLHLHLLDDULHDHH; 'P: 31
LHHLHLHLLDDULHDHH; 'P: 32
HHHLHLHLLUDDULHDHH; 'P: 33
HHHLHLHLLUDDULHDLH; 'P: 34
LHHLHLHLLUDDULHDLH; 'P: 35
HHHLHLHLLDUDULHDLH; 'P: 36
LHHLHLHLLDUDULHDLH; 'P: 37
HHHLHLHLLUUDULHDLH; 'P: 38
LHHLHLHLLUUDULHDLH; 'P: 39
HHHLHLHLLDDUULHDLH; 'P: 40
LHHLHLHLLDDUULHDLH; 'P: 41
HHHLHLHLLUDUULHDLH; 'P: 42
HHHLHLHLLUDUULHDHH; 'P: 43
LHHLHLHLLUDUULHDHH; 'P: 44
HHHLHLHLLDUUULHDHH; 'P: 45
LHHLHLHLLDUUULHDHH; 'P: 46
HHHLHLHLLUUUULHUHH; 'P: 47
HHHLHLHLHDDUULHDHH; 'P: 48
HHHLHLHLHDDUULHDL; 'P: 49
LHHLHLHLHXXXXLHXLL; 'P: 50
HHHLHLHLHUUDULHXLL; 'P: 51
LHHLHLHLHUUDULHXLL; 'P: 52
HHHLHLHLHDUDULHXLL; 'P: 53
LHHLHLHLHDUDULHXLL; 'P: 54
HHHLHLHLHUDDULHXLL; 'P: 55
LHHLHLHLHUDDULHXLL; 'P: 56
HHHLHLHLHDDUULHXLL; 'P: 57
'endif
End SEQ;
Begin SEQ;
'if nOE != LL & nOE != LH
    '$comment ($all, 'Checking nOE pin. ');
    'SET checks nOE pin
    'Select Direct inputs, all pins high
HHHHHHHHHHXXXXLHUHH; 'P: 58
    'Check direct inputs with output
    'write 0101
HHLHLHHHHHHXXXXLHDHH; 'P: 59
    'Read 0101
HHLHLHHHHHUUDLHDHH; 'P: 60
    'CHECK nOE HIGH
HHLHLHHHHHHXXXXHHDHH; 'P: 61
HHLHLHHHHHHXXXXHHDHH; 'P: 62
HHLHLHHHHHHXXXXHHDHH; 'P: 63
HHLHLHHHHHHXXXXHHDHH; 'P: 64
HHLHLHHHHHHXXXXHHDHH; 'P: 65
HHLHLHHHHHHXXXXHHDHH; 'P: 66

```

```

HHLHLHHHZZZZHHDHH; 'P: 67
    'write and read 1010
HHHLHLHHHDUDULHDHH; 'P: 68
HHHLHLHHHXXXLHDHH; 'P: 69
End SEQ;
End SET;
End Vector;

```

Vector Example in IVL Format

IMPORTANT: To save space in long vectors, you can format an IVL pattern such as

```

HXLH;
;
;
;
;
x(77) H(reset) 1(2,3);

```

in the following way:

```
HXLH ; ;;; 1(2,3) x(77) H(reset) ;
```

The order of the IVL patterns is not important. However, what you gain in saving space, you lose in readability.

```

NPINS = 14; 'Pin record definitions, number of pins in
    'the device
1, pin_1 , IO; 'pin number one
2, pin_2 , O;
3, pin_3 , O;
4, pin_4 , O;
5, pin_5 , O;
6, pin_6 , O;
7, pin_7 , I;
8, pin_8 , I;
9, pin_9 , I;
10, pin_10 , I;
11, pin_11 , I;
12, pin_12 , U;
13, pin_13 , U;
14, pin_14 , I;

vector;
    begin set;
        Z(1,8,9,10,11,14) L(7);*   'P: 1
        L(7);*   'P: 2
        H(11) L(14);   'P: 3
        ;   'P: 4
        ;   'P: 5
        H(pin_1);   'P: 6
        ;   'P: 7
        ;   'P: 8
        ;   'P: 9
        ;   'P: 10
        ;   'P: 11

```

```

L(pin_1);    'P: 12
;           'P: 13
;           'P: 14
;           'P: 15
;           'P: 16
;           'P: 17
;           'P: 18
L(pin_7);    'P: 19
H(pin_7);    'P: 20
;           'P: 21
;           'P: 22
H(14);       'P: 23
D(2) H(14);  'P: 24
D(6);        'P: 25
X(6);        'P: 26
;*          'P: 27
;           'P: 28
U(6);        'P: 29
X(2) U(3,6); 'P: 30
;           'P: 31
D(5);        'P: 32
X(5);        'P: 33
;           'P: 34
X(3);        'P: 35
U(5);        'P: 36
;           'P: 37
;           'P: 38
end set;
end vector;

'Guards should only be applied to stim or stim/meas
'types. The pins identify the stims used to guard this
'node. The pin numbers or names refer to the pins defined above
'in the pin record definition section.

guard 2;      'Guard for output pin 2
Pins = (pin_1, pin_7, pin_8, pin_9, pin_10, pin_11, pin_14);

begin set;
Z(pin_1, pin_7, pin_8, pin_9, pin_10, pin_11, pin_14); 'P: 1
;           'P: 2
H(pin_11) L(pin_14); 'P: 3
;           'P: 4
;           'P: 5
end set;
end guard;

guard 3;      'Guard for output pin 3
Pins = (1, 7, 8, 9, 10, 11, 14);

begin set;
Z(1, 7, 8, 9, 10, 11, 14); 'P: 1
;           'P: 2
H(11) L(14); 'P: 3
;           'P: 4
;           'P: 5
end set;
end guard;
guard 4;      'Guard for output pin 4

```

```

    Pins = (1, 7, 8, 9, 10, 11, 14);

    begin set;
    Z(1, 7, 8, 9, 10, 11, 14); 'P: 1
    ; 'P: 2
    H(11) L(14); 'P: 3
    ; 'P: 4
    ; 'P: 5
    end set;
end guard;

guard 5;          'Guard for output pin 5
    Pins = (1, 7, 8, 9, 10, 11, 14);

    begin set;
    Z(1, 7, 8, 9, 10, 11, 14); 'P: 1
    ; 'P: 2
    H(11) L(14); 'P: 3
    ; 'P: 4
    ; 'P: 5
    end set;
end guard;

guard 6;          'Guard for output pin 6
    Pins = (1, 7, 8, 9, 10, 11, 14);

    begin set;
    Z(1, 7, 8, 9, 10, 11, 14); 'P: 1
    ; 'P: 2
    H(11) L(14); 'P: 3
    ; 'P: 4
    ; 'P: 5
    end set;
end guard;

DISABLE;  'disable patterns
    Pins = (1, 7, 8, 9, 10, 11, 14);

    begin set;
    Z(1, 7, 8, 9, 10, 11, 14); 'P: 1
    ; 'P: 2
    H(11) L(14); 'P: 3
    ; 'P: 4
    ; 'P: 5
    end set;
end disable;

```

Selectone/Select Example

```

begin set;
    begin repeat 2;
    (SETUP)
    end repeat;

    begin selectone;
        begin entry;
        (INIT1);
        end entry;

```

```

        begin entry;
        (INIT2);
        end entry;

        begin entry;
        (INIT3);
        end entry;
    end selectone;

    begin select;

        begin block;
        (TEST1);
        end block;

        begin block;
        (TEST2);
        end block;

        begin block;
        (TEST3);
        end block;
    end select;

    begin seq;
    (COMMON PATTERNS);
    end seq;

end set;

```

The previous example produces the following output:

```

SETUP
SETUP
INIT1
TEST1
COMMON PATTERNS

SETUP
SETUP
INIT1
TEST2
COMMON PATTERNS
SETUP
SETUP
INIT1
TEST3
COMMON PATTERNS

```

Nested Select Example

```

begin set;

    (INIT PATTS);

    begin select;

```

```

        begin block;

            (NESTED PATTS);

            begin select;

                begin block;
                    (2TEST1);
                end block;

                begin block;
                    (2TEST2);
                end block;

                begin block;
                    (2TEST3);
                end block;

            end select;

        end block;

    begin block;
        (TEST2);
    end block;

    begin block;
        (TEST3);
    end block;

    end select;

    (END PATTS);

end set;

```

The previous example will produce the following output, assuming there is no conflict between patterns and device wiring constraints.

```

INIT PATTS
NESTED PATTS
2TEST1
END PATTS

```

```

INIT PATTS
NESTED PATTS
2TEST2
END PATTS

```

```

INIT PATTS
NESTED PATTS
2TEST3
END PATTS

```



```

INIT PATTS
TEST2
END PATTS

```

```

INIT PATTS
TEST3
END PATTS

```

Sequential Select Example

```

begin set;

    (INIT PATTS);

    begin select;

        begin block;
            (TEST1);
        end block;

        begin block;
            (TEST2);
        end block;

        begin block;
            (TEST3);
        end block;

    end select;

    begin select;

        begin block;
            (2TEST1);
        end block;

        begin block;
            (2TEST2);
        end block;

        begin block;
            (2TEST3);
        end block;

    end select;

    (END PATTS);

end set;

```

The previous example will produce the following output

```
INIT PATTS
TEST1
2TEST1
END PATTS
```

```
INIT PATTS
TEST1
2TEST2
END PATTS
```

```
INIT PATTS
TEST1
2TEST3
END PATTS
```

```
INIT PATTS
TEST2
2TEST1
END PATTS
```

```
INIT PATTS
TEST2
2TEST2
END PATTS
```

```
INIT PATTS
TEST2
2TEST3
END PATTS
```

```
INIT PATTS
TEST3
2TEST1
END PATTS
```

```
INIT PATTS
TEST3
2TEST2
END PATTS
```

```
INIT PATTS
TEST3
2TEST3
END PATTS
```

Homing Loops Example

```

'Component : homing

NPINS = 6;

1,pin1,I;
2,pin2,I;
3,pin3,H0;
4,pin4,H0;
5,pin5,0;
6,pin6,0;

Maxrate 2 MHZ
Mdelay 500 NS
Thresh LO 1.6 HI 1.6
Term 5000 Pull-down
Clock Internal
Maxhoming 500

Vector;
  Begin SET;
    HHUXXX;
    LLXXXX;
    HHXUXX;
    LLXXXX;
    HHXXXX;
    Begin Homing;
      LLDDXX;
      HHUXXX;
      LLDDxx;
      HHUXXX;
      LLDDXX;
      HHUXXX;
    End Homing;
    LLXXXX;
    HHXXXX;
    LLXXDU;
    HHXXXX;
    LLXXXX;
    Begin Homing;
      HHXXXX;
      LLXDDX;
      HHXUXX;
      LLXXXD;
      HHXXXX;
      LLXXXX;
    End Homing;
    HHXXXX;
    LLXDDX;
    HHUXXX;
    LLUXXX;
    HHXXXX;
    LLXXXX;
    HHXXXX;
    LLXXXX;
  End Set;
End Vector;

```

```
End SET;
End Vector;
```

Mixed Vector Example

```
'Component : homing
```

```
NPINS = 6;
```

```
1,pin1,AI;
2,pin2,AO;
3,pin3,O;
4,pin4,O;
5,pin5,I;
6,pin6,O;
```

```
Maxrate 2 MHZ
Mdelay 500 NS
Thresh LO 1.6 HI 1.6
Term 500 Pull-down
Clock Internal
StimV pin1, 5.0V DC res 1K wait 100ms
```

```
Vector;
  Begin SET;
    aaUDHU;
    aaXDLU;
    aMDDLX measv pin2,4.5 to 5.5v DC;
    aaXDLX;
    aaUDHU;
    aaXDHU;
    aaXDHU;
  End Set;
End Vector;
```

▼▼▼

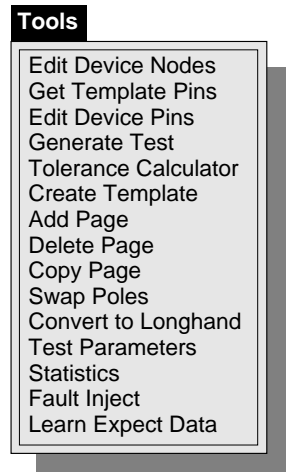
9 TEST AND DEBUG TOOLS

Chapter 9 is a reference for the tester's integrated software and hardware test and debugging tools. The various sections discuss

- using the Edit/Tools menu in test development,
- using test jacks for monitoring signals and identifying measurement intervals and intervals when stimuli are applied,
- setting up Digital Tracer to find opens,
- using Fault Inject to evaluate digital test quality and coverage,
- using the Utility menu for test development and debugging,
- setting up Documenter to create readable files documenting the test program, and
- setting up Validate to improve capacitor and resistor shorthand tests and MultiScan tests without the intervention of a programmer.

Test Development Tools

The Tools menu is located in the menu bar at the top of the component Step Worksheet.



With this menu you can

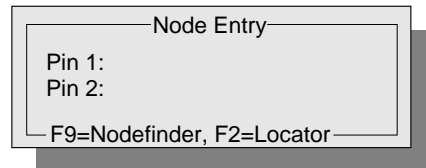
- edit the nodes of a device
- obtain the correct number of template pins
- generate a test step with the program generator
- calculate device tolerance
- create a template from the test step
- add, delete, and copy Test Properties pages
- swap stimulus and measurement poles
- convert test steps to a longhand format.

Edit Device Nodes

To edit the nodes of a device

- select Edit Device Nodes from the Tools menu, or
- click the Number of Pins field in the Component Properties portion of the Step Worksheet.

The Node Entry window shown below appears.



You can type node numbers into the fields or find and insert them automatically with Nodefinder. Click the field to Edit Device Pins.

See the section, Software Utilities, in this chapter for more information.

Get Template Pins

Get Template Pins obtains the component name and device type from the Component Properties portion of the Step Worksheet. Get Template Pins also fills in the correct number of pins in the Test Properties portion of the Step Worksheet. For a digital device type, the number of pins is taken from the template library. For most analog device types, the number of pins is the default number. The Resistor test type, for example, always has 2 pins.

If the Test Properties for the current Step Worksheet already exists, a query window asks if you want to overwrite the device test that already exists.

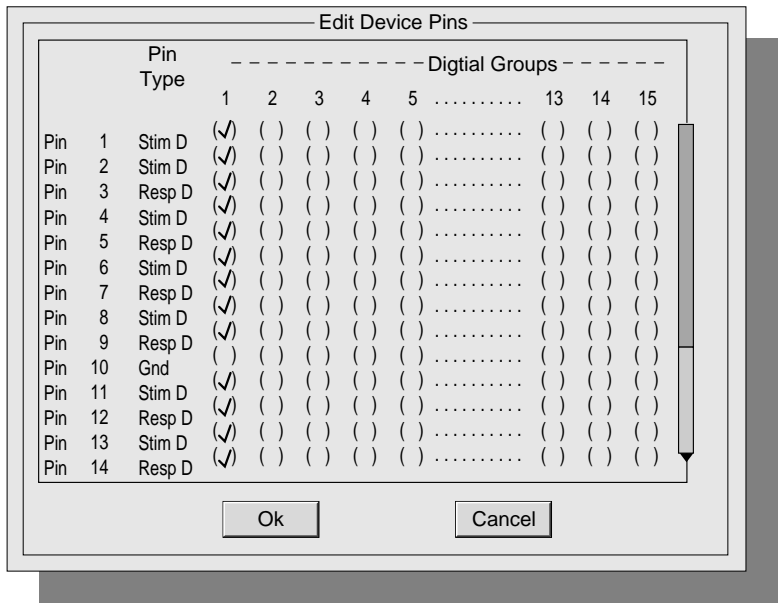
Edit Device Pins

With Edit Device Pins you can edit Gray code groups and pin types for all of the devices in a single window. This capability allows you to easily edit these groups and types for new templates. The global view of the pin settings makes Edit Device Pins a useful reference tool.

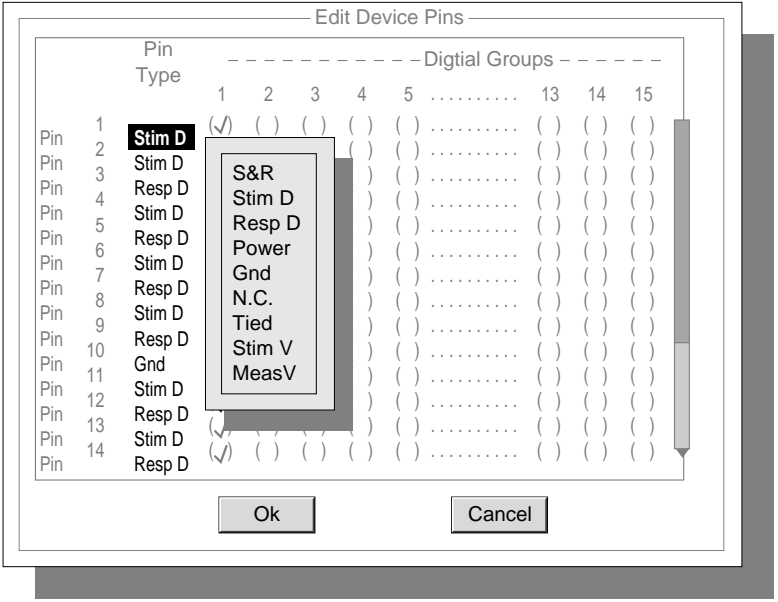
Edit Device Pins is available only for Gray code devices.

When you select Edit Device Pins from the Tools menu, the following window appears.

Use the scroll bar to the right to see the entire pin list for the device.



Click any field below Pin Type to bring up a pop-up window as shown below.
Click the pin type to select it.



Generate Test

Generate Test is different from the Generate command located in the Pgen menu. This command allows you to generate the test step as you debug. If the Test Properties for the current Step Worksheet already exists, a query window asks you to confirm test step generation.

Add, Delete, and Copy Page

To add a page to a Test Properties portion, select Add Page from the Tools menu.

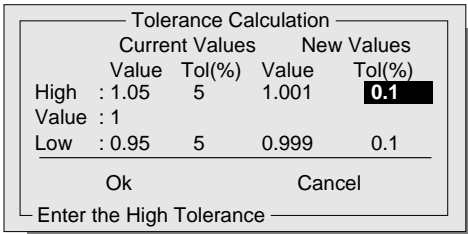
Select Delete Page to delete the page currently showing in the Test Properties portion of the Step Worksheet. The page is not fully deleted until you save the Step Worksheet. You can un-delete a page before saving by selecting Revert.

To add a page that is identical to the current one displayed in Test Properties, select Copy Page. If you copy or add, the new page is appended to the multipage series. For example, if you copy page 3 of 5, it becomes page 6 of 6. After you add or copy a page, return to the menu by pressing F10. Save the test before you add or copy additional pages. When you add a page, values default to the lowest possible value.

Tolerance Calculator

The Tolerance Calculator simplifies the task of calculating high and low tolerance limits that are based on the percentage of the tested value.

You can activate the Tolerance Calculator by double clicking the Value, High, or Low fields in analog worksheets as well as by selecting Tolerance Calculator from the Tools menu. The tolerance calculator applies only to tests which have a nominal value.



Tolerance Calculation				
Current Values			New Values	
	Value	Tol(%)	Value	Tol(%)
High	: 1.05	5	1.001	0.1
Value	: 1			
Low	: 0.95	5	0.999	0.1

Ok Cancel

Enter the High Tolerance _____

Current Values displays the current worksheet values and the current worksheet tolerance. Use the New Values area of the Tolerance Calculator to modify the high and low tolerances.

Choose OK to accept the new values or Cancel to abort. Each time you open a test step, the tolerance calculator's New Tolerance is initialized to the tolerance specified in the Component Properties worksheet. As new values are entered in the New Tolerance fields, those values will "stick" until the test step is exited. This means that if a tolerance of 12% is entered in the Tolerance Calculator, and the Tolerance Calculator is closed (with OK or Cancel), the next time the Tolerance Calculator is brought up for this step, the tolerance of 12% will be retained.

The tolerance calculator is not available if Test Properties does not have a value field.

If you are operating under Windows 95, you can also use the Windows calculator and copy and paste the values into the appropriate 18xx fields.

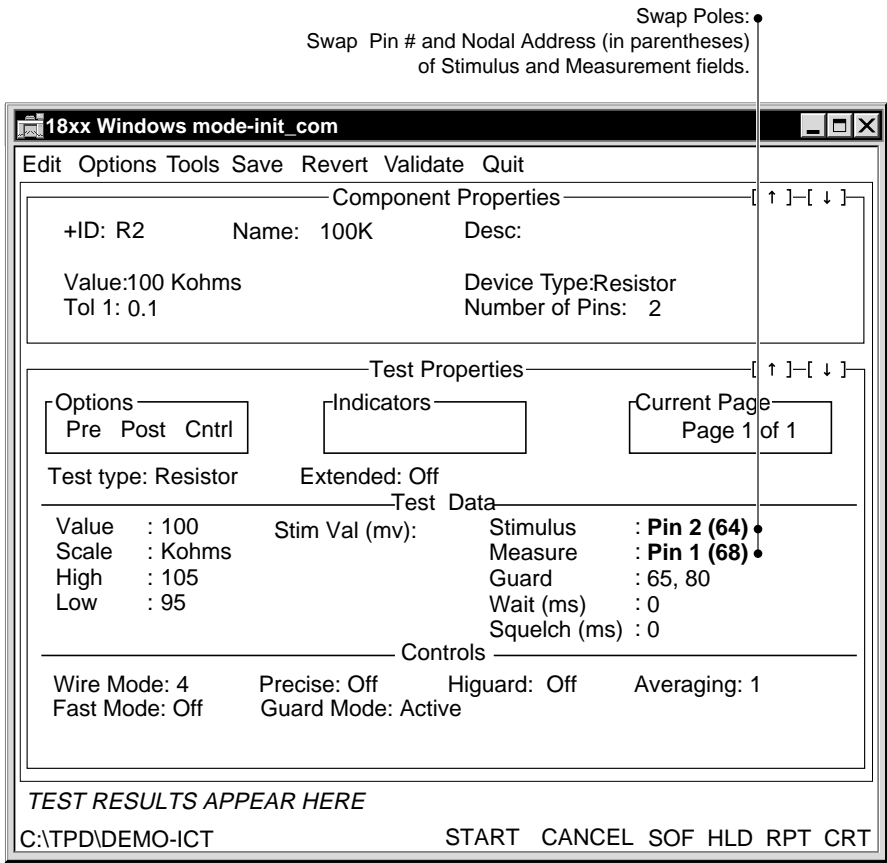
Create Template

Create Template allows you to take a snapshot of the current Step Worksheet. There are only a few Device Types that templates can be created from. The device type is specified by the Component Properties' Device Type field. The valid types are

- Analog Template—goes to ATMPL library
- Gray Code—goes to IC library
- Vector Image—goes to VIMG library.

Swap Poles

Swap Poles is an analog debugging tool you can use to reverse the stimulus and measurement pin/node numbers. It is sometimes necessary to swap poles to improve measurement stability. Poles are swapped in relation to the Pin/Node numbers stated in Test Properties—the E-pole remains stimulus and the F pole measurement. When you swap poles, no other aspect of the measurement is altered.



Convert to Longhand

Convert to Longhand converts the contents of a shorthand resistor or capacitor test into a roughly equivalent longhand test.

Shorthand resistor test:

18xx Windows mode-init_com

Edit Options Tools Save Revert Validate Quit

Component Properties [↑][↓]

+ID: R2 Name: 100K0 Desc: Res, delta with 1M, 1K

Value: 100.00 Kohms Device Type: Resistor

Tol 1: 0.1 Number of Pins: 2

Test Properties [↑][↓]

Options: Pre Post Cntrl Indicators: Current Page: Page 1 of 1

Test type: Resistor Extended: Off

Test Data

Value : 100	Stim Val (mv):	Stimulus : Pin 1 (68)
Scale : Kohms		Measure : Pin 2 (64)
High : 102.10		Guard : 65, 80
Low : 97.900		Wait (ms) : 100
		Squelch (ms) : 0

Controls

Wire Mode: 6 Precise: On Higuard: On Averaging: 1

Fast Mode: Off Guard Mode: Semi-Active

C:\TPD\DEMO-ICT START CANCEL SOF HLD RPT CRT

Shorthand resistor test converted to equivalent longhand test:

18xx Windows mode-init_com

Edit Options Tools Save Revert Validate Quit

Component Properties [↑][↓]

+ID: R2 Name: 100K0 Desc: Res, delta with 1M, 1K

Value: 100.00 Kohms Device Type: Resistor

Tol 1: 0.1 Number of Pins: 2

Test Properties [↑][↓]

Options: Pre Post Cntrl Indicators: Current Page: Page 1 of 1

Test type: Resistor Extended: Off

Test Data

Scale : ua	Stim Val (mv) : 200.00	Stimulus : Pin 1 (68)
High : -1.958	Scale : mv	Measure : Pin 2 (64)
Low : -2.042	Stim Type : DC	Reference : 65, 80
Measure Type: DC		Wait (ms) : 120
		Squelch (ms) : 2

Controls

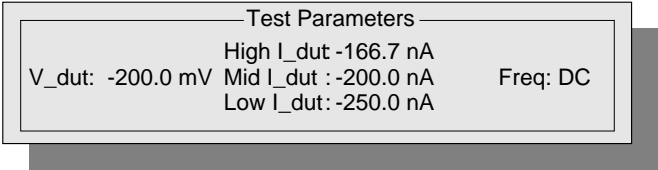
Wire Mode: 6 Precise: On Higuard: On Averaging: 1

Guard Mode: Semi-Active

C:\TPD\DEMO-ICT START CANCEL SOF HLD RPT CRT

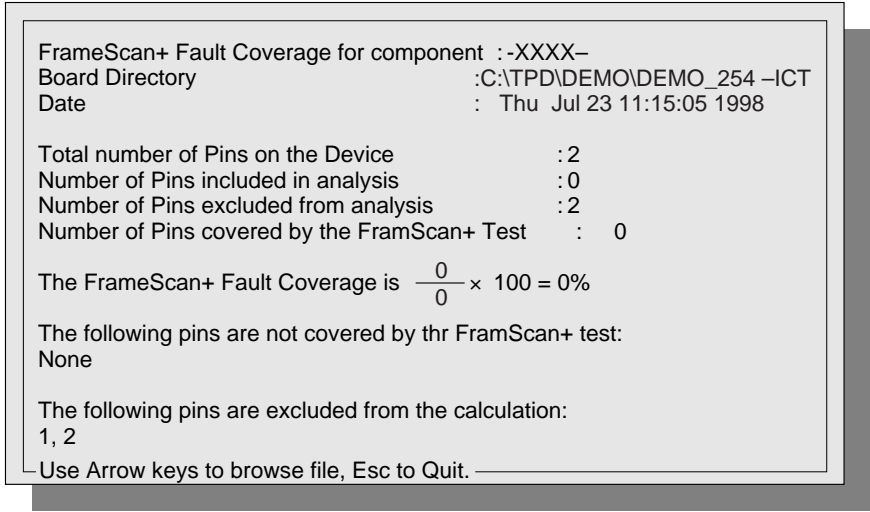
Test Parameters

Use Test Parameters to see the actual parameters that are used for the current test. When you select Test Parameters from the Tools menu, a window similar to the one below appears showing the actual voltages, currents, and frequency used in the test.



Statistics

The Statistics report is a function of WaveScan, FrameScan, and FrameScan Plus, and as such is accessible only for these tests. The report indicates the fault coverage for the device-under-test.



See the **MultiScan User’s Guide** for more information.

Using the Test Jack Panel in Debug

Test jacks, located on the test jack panel (or the control panel on the Z1820/00), connect to the instrument during debugging. Enable or disable test jacks in the Setup/Environment menu.

The Test Jack panel, similar to the one shown below, provides access via six test points labeled E_d, F_d, G_d, E_s, F_s and G_s to the analog/digital measure and stimulus signals as present on the system’s backplane poles (EFG). The panel also has two synchronization signals for oscilloscope or logic-analyzer work called the “Listen Window” and the “Test Envelope.”

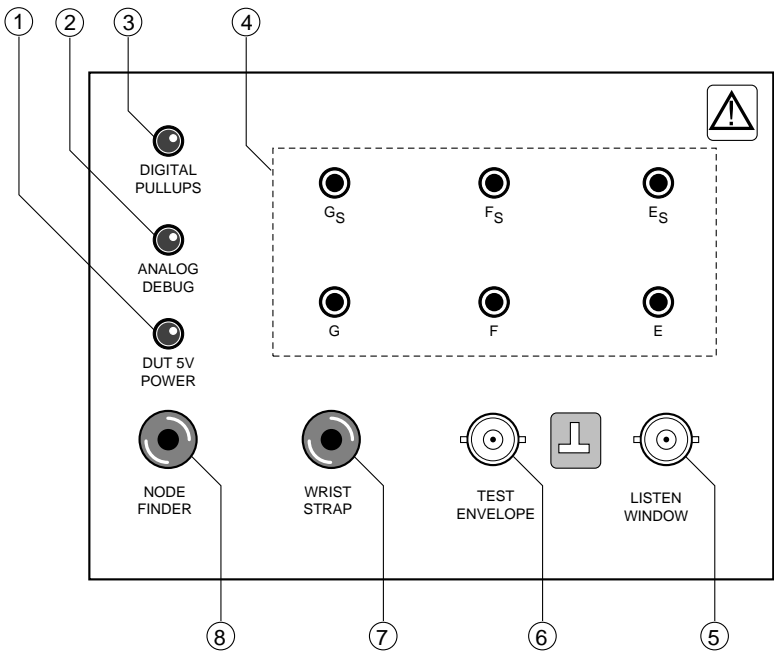
The six test points—E_d through G_s—are available only in the Step Worksheet debug mode, when the Test Jack Panel is set to Poles or Poles & Triggers in the Setup/System Variables menu. Section Run and Production Run never connect the Test Jack Poles regardless of settings in Setup/System Variables. For analog test the synchronization signals are provided only when the Test Jack Triggers or Test Jack Poles & Triggers are enabled, for digital tests, they are always present.

Note: the System Variables/Test Jack Panel variables are

Off	
Poles	$E_d - G_s$
Triggers	Test Envelope and Listen Window
Poles & Triggers	$E_d - G_s$ and Test Envelope and Listen Window

Refer to the table below for an explanation of each notation on the following figure.

1880/1890 Test Jack panel



Index	Name	Function
1	DUT 5 V Power LED	Green LED lights up when DUT 5 V Power is in use.
2	Analog debug	Green LED lights up during analog debugging, indicating possible high voltage on E, F, and G test jacks.
3	Digital pullups	Red LED lights up when built-in termination resistors in use.
4	E, F, G Poles (Tip jacks)	E, E_s (E_{sense}), F, F_s (F_{sense}), G, and G_s (G_{sense}) allow the monitoring of backplane signals. The E, F, and G analog drive and sense buses from the backplane are routed to these six jacks during analog debugging. The jacks are connected to the six digital response buses during digital debugging.
5	Listen window	Identifies the measurement interval.
6	Test envelope	Identifies time interval when stimuli are applied.
7	Wrist strap (Yellow banana jack)	Provides protection against electrostatic discharge. Connects ESD wrist strap to ground via a 10 M Ω resistor.
8	Node Finder (Black banana jack)	Connects Node Probe to tester ground.

When Poles is selected for Test Jack Panel, the test jacks add 10 to 20 pF (picofarads) of capacitance to the analog measurement buses. If you do not disable the Test Jack Poles, values of very small capacitors will read higher at production test than they do at debug. Debug small capacitors with the test jacks turned off.

The test jack panel functions similarly on all tester models. Refer to the System Reference for your tester for illustrations of test jack panel locations.

WARNING!

When the analog debug light is on, lethal voltages may be present on the test jack panel.

Test Envelope and Listen Window

The Test Envelope and Listen Window signals are useful for triggering an oscilloscope or logic analyzer. These are referred to as Triggers in the Setup/System variables menu.

IMPORTANT: Test Envelope is active high for the vector processor and active low for the THC. For digital tests using a THC, for example, the Test Envelope goes low when backdrive signals are applied and returns to high after the last clock in the burst.

In Analog Tests. For analog test, use the Test Envelope and Listen Window to trigger on actions within a measurement. With the THC, the Test Envelope goes low immediately after squelch is released and prior to the stimulus. The Test Envelope returns to high after this measurement is complete. The Listen Window goes high during the conversion cycle of the ADC (during the measurement sample), and returns to low afterwards. If multiple measurement samples are taken, there will be a Listen pulse for each sample.

In Digital Gray Code Tests. For digital tests, the Listen Window is the period in the test burst during which the signature hardware is listening to the signals coming from the device-under-test. The Listen Window is normally low but goes high only while signatures are being recorded within the Test Envelope.

Default actions are provided by the software to cover most of the cases you encounter. For example, the listen window opens immediately after the leading edge of F1 as a default and closes immediately after the last clock in the stimulus cycle. The last clock in a burst occurs at the first edge of the Gray code two levels longer. For example, if the longest Gray code stimulus used in a test is F5, then the leading edge of F7 will close the window. For the Default case, the Test Envelope ends at the same time as the Listen Window.

The phrase DEF (default) in all From, To, and While fields means that the default situation described in the previous paragraph applies. You can change the window to mask unwanted or unreliable data.

Changing the From field, delays the opening of the window. Instead of opening at the first edge of F1, the window will not open until the first edge of one of the other Gray code frequencies.

To change the closing of the window, change the To field, which also closes the Test Envelope. You may close the window earlier than the default with a smaller F number, or later with a larger F number. It is not meaningful to close the Listen Window before it has been opened, for example, opening the window at F10 and closing it at F3.

Changing the specification of the To field changes the length of the stimulus burst as well. The Test Envelope always closes at the To frequency.

There are two While fields which can be used to further control the Listen Window. Selecting a Gray code frequency for a While field will close the window whenever the frequency is low. Specifying both While fields results in a window that is the logical AND of the two selected frequencies. The two While windows are programmed independently. The default for While is Continuous. When both While functions are set to DEF, the window is fully open between the From and To events.

When Repeat is on in Gray code tests, an individual burst keeps repeating until you press Start. To correlate the six measurement poles to the CRC, COUNT, or HIGH results shown on the display line, refer to the following example of display line layout.

Edrive	Fdrive	Gdrive	Esense	Fsense	Gsense
---------------	---------------	---------------	---------------	---------------	---------------

Vector Test. The Test Envelope in vector tests (active high) extends from the first state of the vector through the last state. The Listen Window is active (high) whenever an Up or Down is expected on the pin assigned to the E Pole, by default. The only way to have the Listen Window assigned to some other pin is by selecting the Trigger function in the Pin Record Menu of the desired pin.

The Pole assignment for vector tests can be obtained from the pin-record of a particular measure pin. The Trigger ON/OFF field controls the Test Envelope and Listen Windows signal. When none of the pins has Trigger On, the Test Envelope is on for all bursts and the Listen Window follows the listen for the pin measured on the Edrive pole. When a measure pin has Trigger set to ON, the Test Envelope signal is only generated for the burst in which the particular pin is measured. The Listen Window is controlled by the listen of that particular pin.

Using Test Jacks in Digital Debug

If Test Jacks is On in the Setup/Environment menu, you can investigate a burst further by examining signals at the test jack panel. Trigger your oscilloscope or logic analyzer using the Test Envelope and/or Listen Window jacks. With a THC, the Test Envelope is normally high, and goes low during the digital burst. Listen Window goes high when the tester samples the results.

Synchronization signals are always present when executing a digital test, regardless of the state of Test Jacks Setup/System Variables. The Test Envelope spans the duration of a digital burst. The Listen Window is active whenever data is being sent to the CRC, HIGH, or COUNT register for Gray code tests or sampled during a vector test.

During digital test debugging, the six EFG test points are connected to the digital equivalent of the six analog poles. The signal from the DUT is routed through the Driver Receiver cards where thresholds are compared and the signals are buffered.

The buffered signals are then distributed onto the digital EFG poles on the backplane, where the Test Head Controller or Vector processor board reads them. It is these buffered signals which are routed to the EFG test points during digital debug.

Up to six measurements can be taken simultaneously, depending on the switch matrix allocation and the use of single or dual thresholds. The results are displayed in the six digital results fields of the display line. (See an example of the display line layout under Gray Code Tests.)

For a device with more than six measure pins, the stimulus sequence is repeated with a different set of measure nodes. The “re-bursting” is repeated until all measure points are collected. When making dual threshold measurements, only three measurements are possible per burst.

The drive pole carries the signal from the low threshold discriminator and the sense pole carries the signal from the high threshold device. The two signals are identical; however, when the signal lies between the thresholds the time sampling is inhibited.

Displaying Test Results

If a test requires more than one burst, you will see only the results of the last burst. If you press Repeat, F7, before choosing Start, the tester repeats endlessly on the first burst, allowing you to evaluate the test for stability. You can also trigger an oscilloscope with the test jacks on to view the test signals. To advance to the next burst when Repeat is active, press Start, F3. Repeat will show the sequence the tester follows during production testing. You can also use Hold, F6, to single-step through the bursts where F3 advances to the next burst.

You can use SOF (Stop On Fail) and Repeat to get the tester to stop and display a result that occasionally fails.

Using Test Jacks in Analog Debug

When executing analog tests, the six test points labeled E_s , E_d , F_s , F_d , G_s , and G_d are connected to the appropriate analog backplane signals of the same name. To indicate possibly high voltage levels, the green LED is turned on as long as the connection remains.

The Test Jack Poles are helpful during debugging because they facilitate access to the test signals, but they can adversely influence the test result. When active, Test Jack Poles can add 10 to 20 pF to the analog measurement poles, thereby changing small capacitor readings. Test Jack Poles should be Off in the Setup/System Variables menu when debugging small capacitors.

You can use test points to make sure that your measurement results are actually caused by the DUT and not by guarding errors of the system. To do this, connect a device of the same type and value as the DUT to the E and F poles and see the result change by 50% due to the paralleling of the device with the DUT.

The synchronization signals, Test Envelope and Listen Window, have the following relationship to the measurement sequence:

- Test Envelope is active during the entire time the measurement is being performed. It goes high before the stimulus turns on and extends until after the measurement(s) have been taken, encompassing all squelch and wait times.

When executing in “repeat mode” the Test Envelope is shorter since the stimulus is already established and wait and squelch times are usually not executed again. The off phase of the Test Envelope signal while in repeat mode, marks the time spent displaying the result and setting up for a measurement again. You can see the full measurement cycle, by pressing the START key on your keyboard to repeat the test.

- The Listen Window appears within the Test Envelope signal and indicates the point where the analog to digital conversion takes place. The Listen signal is usually very short, in the neighborhood of 1 μ S, except for Peak and Peak to Peak measurements when the Listen Window also includes the gate time for the Peak detectors.

You can connect an oscilloscope input to the E (stimulus) jack if Poles & Triggers are selected in the Test Jack Panel fields of the Setup/System Variables window. Trigger the oscilloscope from the Test Envelope jack, and observe also the Listen Window signal. The Test Envelope signal, normally high (THC), goes low for the duration of the test sequence. The normally low Listen Window pulse marks the measurement.

If Poles & Triggers are selected in the Test Jack Panel field, you can also connect from E to F a decade box of the same type of component you are debugging. If you are debugging a resistor test, use a resistance decade box. If you set the decade box to 10 times the impedance value of the component you are measuring, the tester should report a 9% lower impedance value. For example, a 100 kohm resistor added in parallel with a “perfect” 10 kohm resistor under test should cause the tester to read 9.091 kohm. A 1000-picofarad capacitor added in parallel with a perfect 10-nanofarad capacitor under test should read 11 nanofarads.

Using Digital Tracer to Test Connectivity

Digital Tracer is a digital testing tool which allows you to test the connectivity of digital component leads to the board-under-test via a hand held probe. Once enabled in the software, it stops a test upon locating a failing device to allow the operator to probe the pins. Digital Tracer displays a graphic representation of the chip on the screen and visually traces the progress of the probing. When contact is made with each pin or group of pins, indicating connectivity, the pin graphic changes color. At the end of the probing, pin graphics remaining the same color show where opens exist, and that information is sent to output devices specified in your program.

IMPORTANT: Because the imaging ability of Digital Tracer is limited to 34 pins per side, chips with pins exceeding that number can have more than one pin represented by the same pin graphic. The Pin Contact Indicator shows when pin contact is made with a target pin.

Digital Tracer can be set up in the program in three ways:

- Globally, in PRGMVARS
- By TRACER.DEF entries from a C.1 program which override global settings on matching components

Individually in the Step Worksheet description field entries which override both global and TRACER.DEF entries for that one component

In a production situation, the operator can override all previous Digital Tracer setups for an individual device by using the menu on the screen.

See the **Z1800-Series Operator's Guide** for information about operator use.

Setting Up Tracer in PRGMVARS

If all the chips on a board-under-test are of the same type and oriented in the same way, the setup in PRGMVARS is the only setup required. To set up global control of Digital Tracer, select Edit/Header/PRGMVARS. The following window appears.

18xx Windows mode-init_com

Edit Save Revert Quit

Header [↑] [↓]

(√) General Variables () Report Variables

General Variables

Vacuum Select	: Vacuum 1	Vacuum Delay (mS)	: 100
Auto Probe Check (APC)	: Off	Stop On Fail (SOF)	: Off
Digital HiCheck	: Off	Skip On Fail	: Off
Digital Tracer	: Off	Backdrive Timeout (mS)	: 0.0
Section Abort	: On	Duty Cycle (%)	: 0
Abort On Fail Count	: 0	Learn Cap Phase Offsets	: Off
Discharge Method	: PwrNodes	DR1 Emulation	: Off
MultiScan Discharge	: Off	DFP Configuration	: Off
		External Power Down	: Off

Ground Reference Nodes (7) () () () ()

MultiScan Reference Node () FrameScan Plus Measure ()

DeltaScan, Short Pwr/Gnd via G-pole: Off

DeltaScan, Verify Pwr/Gnd Shorting: Off

DeltaScan, Fixed Threshold: 50

Board Description [Self-Test Block Assembly (043-179 or 043-203)]

Dscan retry [5] Dscan retry delay [1] ms

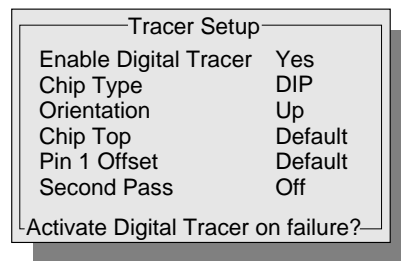
Dscan Run ALL Pin Pairs[Off] Dscan Hard Fixed Thresh [Off]

Select the desired Vacuum control

C:/TPD/DEMO/DEMO_172 - ICT START CANCEL SOF HLD RPT CRT

- 1 Click the Digital Tracer field.

The Tracer Setup window appears.



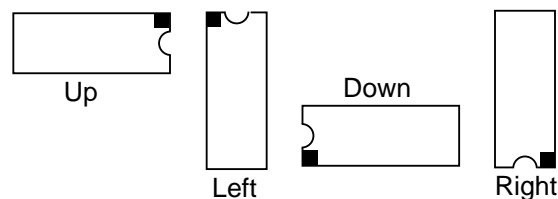
- 2 To enable Digital Tracer, click the Enable Digital Tracer field, and select Yes from the pop-up window.
- 3 To specify your chip type, click the Chip Type field and select DIP or QUAD from the pop-up window.

- Choose DIP if you have pins on 2 sides.
- Choose Quad if you have pins on 4 sides.

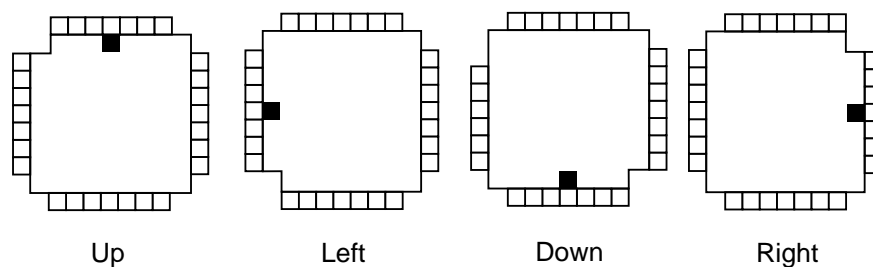
The default is DIP.

- 4 To specify the direction that pin 1 faces, click the Orientation field.

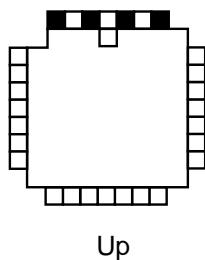
Device orientation corresponds to its physical placement on the board when mounted on the tester, as you face the front of the tester. The illustration below shows the orientation of a DIP chip and the location of Pin 1, which is indicated by the black square (pin graphic) in the corner. The default is Up.



The following shows the orientation of a QUAD chip and the location of pin 1.



Selecting Quad as the Chip Type, enables the Chip Top selection so that you can specify the number of pins on the top of the device. The device shown below has 7 pins across the top.



- 5 To specify that number, click the Chip Top field and select Top Pin = from the pop-up window.

IMPORTANT: Choosing 0 pins on top is the same as the default.

Top Pin = will appear in the Tracer Setup window.

- 6 Highlight the number field and type in the appropriate number (7, in this case).

Tracer Setup	
Enable Digital Tracer	Yes
Chip Type	DIP
Orientation	Up
Chip Top	Default= 7
Pin 1 Offset	Default
Second Pass	Off
Number of Pins on top of Quad.	

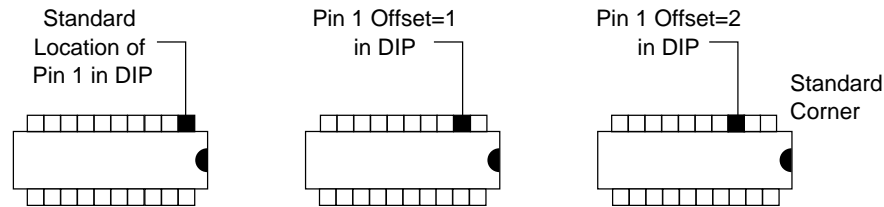
- 7 If Pin 1 is not in the standard location (see below for standard locations), click the Pin 1 Offset field and select Offset =. When Offset = appears in the Tracer Setup window, type in the offset number as shown below.

The offset number indicates how many pins away from the standard corner pin 1 on the chip(s) in question is located. Offset = 2 indicates, for example, that pin 1 is two pins away, counterclockwise, from the standard.

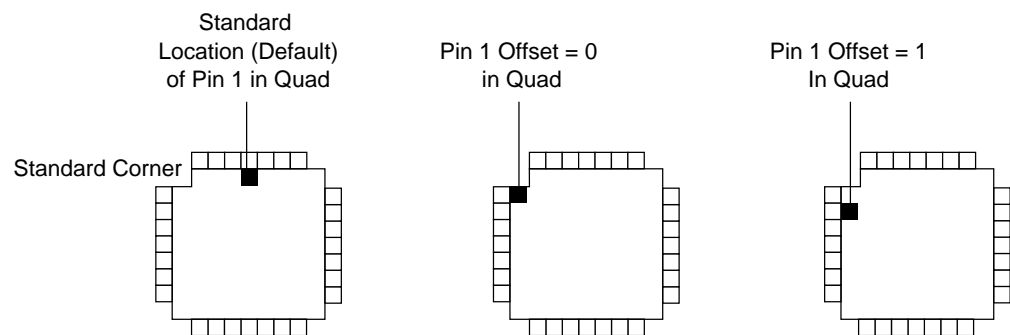
Tracer Setup	
Enable Digital Tracer	Yes
Chip Type	DIP
Orientation	Up
Chip Top	Default= 7
Pin 1 Offset	Default= 2
Second Pass	Off
Pin 1 offset from upper left corner.	

The placement of the Pin 1 offset differs between the DIP and Quad chip type.

Pin 1 offset on a DIP is relative to the location counterclockwise to where Pin 1 is normally found as shown below.



For greater ease in counting, Pin 1 offset on a Quad is relative to the corner counterclockwise from the standard Pin 1 location as shown below.



- 8 If some pins in the device are tied together, you may want to enable Second Pass to be able to test individual pin contacts after the initial pass. Click the Second Pass field and select On from the pop-up window to enable Second Pass.

IMPORTANT: Using the Second Pass function is strongly recommended for tied pins. During first pass, probing one tied pin finds all pins tied to that pin. Second Pass allows you to examine pin contact on previously unchecked pins.

- 9 Save your setup.

The Nodefinder threshold specified in Setup/Environment is used for probing. If power is on when Digital Tracer is called, the poweroff section will be run just before Digital Tracer, and power-on will be run just after Digital Tracer.

If you cannot find all pins on a device, try changing the Nodefinder threshold to a higher value before assuming that the pin is actually open.

Using the C.1 TRACER.DEF File

If you have programs which were generated with the C.1 software revision, the settings in PRGMVARS will be automatically overridden by a pre-existing TRACER.DEF file. For example, if the TRACER.DEF file has entries only for 16-pin chips, all 16-pin chips on your board will take the orientation specified in the TRACER.DEF file. Any other chips will not be affected by the TRACER.DEF file.

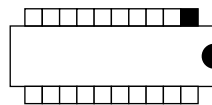
The TRACER.DEF file function is available in D.2 and later versions of the operating system. However, PRGMVARS and Component Properties entries provide all necessary programming control over Digital Tracer setup without using the TRACER.DEF function.

The format for the TRACER.DEF file is

number of pins number of top row pins side up pin 1 offset

Format	Function
Number of pins	The total pin count for a given device
Number of top row pins	The number of pins in the top row of the device. This number varies: a 16-pin DIP has 8 top pins if positioned horizontally, and 0 top pins when positioned vertically. A QUAD chip usually has one quarter of its pins on top.
Side up	Side up (1=Pin 1 side up; 2=Pin 1 side left; 3=Pin 1 side down; 4=Pin 1 side right. Note that this refers to the standard pin 1 location.
Pin 1 offset	Pin count counterclockwise on pin 1 side to the pin 1 position. Note that TRACER.DEF uses the C.1 offsets in QUAD chips, that is, offsets are not from a corner, but are from the standard pin 1 location.

Thus, a 16-pin device, positioned Up horizontally with no pin offset would be described in the TRACER.DEF as 16, 8, 1, 0 and would appear on your board as shown below.

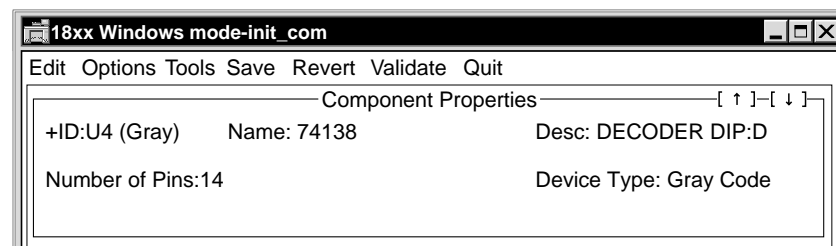


Setting Up in Component Properties

The TRACER.DEF (if it exists) and the global default in the header are both overridden by keywords in Component Properties' description field. To make such changes you must add at least two key pieces of information to the Description field or add the package type and orientation to Test Properties.

To override defaults

- 1 Edit a digital component.
- 2 Add the key words anywhere in the Desc. field: QUAD or DIP followed by a colon (with no spaces), then U, L, R, or D (specifying Up, Left, Right, or Down).



In addition, you can also change the position of pin 1 by adding another colon and a pin offset from the upper left corner of an Up-oriented chip. For example, if you want to specify a pin offset of 4, the Description field in the illustration above would read

```
DECODER DIP:D:4
or
DECODER QUAD:L:0
```

IMPORTANT: There must be no spaces between the keywords and colons. The keywords are not case sensitive. For example, you can type QuAd:R:8.

Evaluating Test Quality with Fault Inject

The 18xx system software provides several techniques to examine and report on fault coverage depending on the types of components. Gray code and vector tests use fault inject. WaveScan, FrameScan, FrameScan Plus, and DeltaScan use the analysis from the Statistics report. Other tests are analyzed based on whether they have been enabled or disabled. The Utility menu's Board Fault coverage selection generates a Fault Coverage Report that combines all the reports.

See the **MultiScan User's Guide** for additional information.

Digital Test Quality

Note, if you have an analog-only test system, information regarding digital test is not pertinent to your test situation. The analog-only functionality prevents you from generating, editing, or running digital tests.

Fault Inject, accessed through the Gray code or vector editor's Tools menu, helps you to evaluate digital test quality by examining state fault coverage and pin fault coverage. Fault Inject is intended to be used after a test has been debugged and is passing. While in debug mode, you can have Fault Inject evaluate the quality of the test by testing the ability of the Vector or Gray code test to detect different faults. Fault Inject generates a report of the results.

Vector Fault Inject

IMPORTANT: The Vector Statistics function also helps to evaluate the output test quality for vector tests. See chapter 7, "Component Test Reference," for more information about the Statistics function.

Vector Measure Pins. Vector Fault Inject helps you assess the quality of a Vector test by analyzing expected states on output pins and simulating stuck high and stuck low faults on input pins. Fault Inject evaluates the output state fault coverage by looking at expect states for each output. If an expect high state is found, the test will be able to detect a stuck at low fault on the pin. An expect low state indicates that a stuck at high fault can be detected.

Vector Stim Pins. Vector Fault Inject simulates a stuck at low condition by having the tester drive the input to a logic low state. Similarly, a stuck at high state is simulated by driving the input to a logic high state. During Fault Inject, inputs are evaluated one at a time by altering its stimulus to simulate a stuck at fault while stimulus to the other inputs is unchanged. If a failure is detected at any of the outputs, then the fault state being simulated is detectable by the test pattern. Fault Inject applies a simulated stuck at high and a simulated stuck at low to each input pin during fault coverage evaluation.

Vector Stim/Meas Pins. For bidirectional pins, a combination of the measure and stimulus pin analysis is used. The measure data is analyzed in the same manner as a measure only pin, and it determines if the pin can detect an output stuck high or an output stuck low condition.

For inputs stuck high and inputs stuck low, you must modify the vector so that all the stim patterns for the pin are changed to either stim logic high or logic low (depending on which type of stuck at condition you are testing for), and the measure patterns remain unchanged. Any failure that occurs when the vector test is executed will detect the stuck at condition.

If the vector test has a large number of patterns, the vector test step can be slow.

Gray Code Fault Inject

Gray Code Meas Pins. Gray code Fault Inject helps you determine if a pin will detect stuck at low and stuck at high conditions by analyzing the measure pin's CRC signature. A CRC of 0000 indicates that a measure pin will not detect a stuck at low condition. If a measure pin's expected CRC is reported as a HiCheck CRC, the measure pin will not detect a stuck at high condition.

Gray Code Stim Pins. Gray code Fault Inject follows the same procedure as Vector Fault Inject described above.

Gray Code Stim/Meas Pins. For the evaluation of bidirectional Gray code pins, the combination of fault coverage for Gray code Stim pins and Measure pins are analyzed using the methods described above.

IMPORTANT: If a Gray code Stim/Meas pin measures its own stimulus, the pin will fail during fault inject and will be reported as not covered.

Fault Inject Algorithm. Between each simulated stuck-at-fault that is injected, the unmodified test is rerun to ensure that the fault injection did not place the device in an undesirable state. Some devices such as sequential PALs, GALs, or ASICs may fail fault inject because of this unless the test vector properly initializes the part.

The Fault Inject function uses two measurements of fault coverage:

- Pin fault coverage
- State fault coverage

IMPORTANT: Mixed Mode pins are not covered by Vector and Gray code Fault Inject.

Pin Fault Coverage

Many 1800-series users may be more interested in pin fault coverage than in state fault coverage since the 1800-series tester is primarily used to detect manufacturing faults such as open pins in a production environment.

There are three types of pins on a component—stimulus, measure, and stimulus/measure. To assess the pin fault coverage of a component, one of the following tests is run for each pin depending on pin type.

For stimulus pins, Fault Inject simulates stuck at conditions for each pin. The test pattern being evaluated must be able to detect a stuck at low and a stuck at high in order to have pin fault coverage.

For each measure pins, the test pattern must expect both a high state and a low state before it can be declared to have pin fault coverage.

For stimulus/measure pins, the pin is considered to have pin fault coverage if the test can detect both an Input stuck at low state and an Input stuck at high state, or an Output stuck at low and an Output stuck at high.

Fault Inject produces the following type of display as the pin coverage analysis of 12 pins in a 20-pin device:

Pin Fault Coverage:

```

-----
Total number of Pin faults possible      : 12
Total number of Pin faults included      : 12
Total number of Pin faults excluded      : 0
Total number of Pin faults detected      : 7
Total number of Pin faults not detected: 5

```

$$\text{Pin Fault Coverage} = \frac{7}{12} \times 100 = 58.3 \%$$

State Fault Coverage

State fault coverage reports the number of logic levels covered in the process of injecting faults into the test. Each input as well as output has two possible logic level faults for inputs (H and L) and for outputs (U and D). Bidirectional pins have five possible logic states (H, L, U, D, and Z). Since the system software does not attempt to measure for the Z state, Z states are not part of any Fault Inject reports.

Fault Inject produces the following type of display as the state fault coverage report for a 20-pin device:

State Fault Coverage:

```

-----
Total number of State faults possible    : 30
Total number of State faults included    : 30
Total number of State faults excluded    : 0
Total number of State faults detected    : 23
Total number of State faults not detected: 7

```

$$\text{State Fault Coverage} = \frac{23}{30} \times 100 = 76.7 \%$$

Using Fault Inject

The Fault Inject interface consists of the Fault Inject window listing all the pins you are allowed to inject faults on, their types, and names. You can specify whether you want to hold/analyze the pin high, low or both high and low.

When you execute Fault Inject, a results window appears showing the fault coverage and detailed results.

To use Fault Inject

- 1 Select Fault Inject from the Tools menu in either the vector or Gray code Step Worksheet.

The Fault Inject worksheet—vector

Fault Inject

Current Pin fault Coverage Data: Pins tested = 0
 [Clear Coverage Data] Pin in analysis = 0
 Fault Coverage = 0%

Report Type: **Pin Fault Coverage**

pin	type	name	test
(✓) 1	Stim	pin_1	H&L
(✓) 2	Stim	pin_2	H&L
(✓) 3	Stim	pin_3	H&L
(✓) 4	Stim	pin_4	H&L
(✓) 5	Stim	pin_5	H&L
(✓) 6	Stim	pin_6	H&L
(✓) 7	Stim	pin_7	H&L
(✓) 8	Stim	pin_8	H&L
(✓) 9	Stim	pin_9	H&L
(✓) 10	Stim	pin_10	H&L
(✓) 15	Stim	pin_15	H&L
(✓) 16	Stim	pin_16	H&L
(✓) 17	Stim	pin_17	H&L
(✓) 18	Stim	pin_18	H&L
(✓) 19	Stim	pin_19	H&L
(✓) 20	Stim	pin_20	H&L

Clear All Execute Cancel

Report only Pin fault coverage or both Pin and State fault coverage

Fault Inject worksheet—Gray code

Fault Inject

Current Pin fault Coverage Data: Pins tested = 0
[Clear Coverage Data] Pin in analysis = 0
 Fault Coverage = 0%

Report Type: Pin Fault Coverage

pin	type	test	pin	type	test
(✓) 1	Stim	H&L	(✓) 11	Stim	H&L
(✓) 2	Stim	H&L	(✓) 12	Meas	H&L
(✓) 3	Meas	H&L	(✓) 13	Stim	H&L
(✓) 4	Stim	H&L	(✓) 14	Meas	H&L
(✓) 5	Meas	H&L	(✓) 15	Stim	H&L
(✓) 6	Stim	H&L	(✓) 16	Meas	H&L
(✓) 7	Meas	H&L	(✓) 17	Stim	H&L
(✓) 8	Stim	H&L	(✓) 18	Meas	H&L
(✓) 9	Meas	H&L	(✓) 19	Stim	H&L

Select All Clear All Execute Cancel

Clear the fault coverage totals

- 2 Select the Report Type, either pin fault coverage or pin and state fault coverage.
- 3 Select the pins on which you wish to inject faults.

All pins are selected by default. You can deselect individual pins by clicking the check marks within the parentheses. The Select All and Clear All options either select all or clear all pins.

- 4 Specify whether you want to check a selected pin for high (H), low (L), or both high and low (H&L) by clicking in the test column.
A pop-up window appears from which to select. The default is H&L.
- 5 Be sure that the fixture and board are in place, that vacuum is on, and that power has been applied to the board.
- 6 Click Execute to confirm your selections and execute the test, or click Cancel to terminate the procedure.

Vector Fault Inject results similar to the following will be displayed in a window. Results for Gray code Fault Inject will be similar.

Vector Fault Inject Results

```
Component: U2-ram
Test Type           : Vector
Board Directory      : C:\TPD\UB90ALL - ICT
Date                 : Mon Jul 27 16:09:30 1998
```

```
WARNING: Test is unstable! A good test (ie. Fault Inject was off) failed.
Results for stimulus pins 29 to 32 were not recorded.
The following is the results of the good test that failed:
NODE: 313 PIN: 14 STATES: 3,4,8,9,13...
NODE: 314 PIN: 15 STATES: 3,4,8,9,13...
NODE: 304 PIN: 18 STATES: 3,4,8,9,13...
NODE: 305 PIN: 19 STATES: 3,4,8,9,13...
NODE: 322 PIN: 21 STATES: 3,4,8,9,13...
```

***** Summary *****

```
Total number of pins on the device : 32
Number of pins in analysis          : 30
```

Pin Fault Coverage:

```
-----
Total number of faults possible      : 30
Total number of faults included      : 30
Total number of faults excluded      : 0
Total number of faults detected      : 8
Total number of faults not detected: 22
```

```

                        8
Pin Fault Coverage = ---- X 100 = 26.7 %
                        30
```

***** Detailed Results *****

```

x : fault excluded
- : fault detected
~det : fault not detected
<blank> : invalid state for pin type
```

Pin Name	Type	Input Stuck		Output Stuck		Pin Covered*
		High	Low	High	Low	
1 a17	Stim	-	-			Yes
2 a16	Stim	-	~det			No
3 a14	Stim	-	~det			No
4 a12	Stim	-	~det			No
5 a7	Stim	-	~det			No
6 a6	Stim	-	~det			No
7 a5	Stim	-	~det			No
8 a4	Stim	-	~det			No
9 a3	Stim	-	~det			No
10 a2	Stim	-	~det			No
11 a1	Stim	~det	~det			No
12 a0	Stim	~det	-			No
13 d0	Stim/Meas	-	~det	-	~det	No
14 d1	Stim/Meas	~det	~det	-	-	Yes
15 d2	Stim/Meas	~det	~det	-	-	Yes
17 d3	Stim/Meas	~det	-	~det	-	No
18 d4	Stim/Meas	~det	~det	-	-	Yes
19 d5	Stim/Meas	~det	~det	-	-	Yes
20 d6	Stim/Meas	~det	~det	-	-	Yes
21 d7	Stim/Meas	~det	~det	-	-	Yes
22 sc1	Stim	-	-			Yes
23 a10	Stim	-	~det			No
24 oe	Stim	-	~det			No
25 a11	Stim	-	~det			No
26 a9	Stim	-	~det			No
27 a8	Stim	-	~det			No
28 a13	Stim	~det	-			No
29 we	Stim	~det	-			No
30 cs2	Stim	~det	~det			No
31 a15	Stim	~det	~det			No

* Pin Coverage is calculated as follows:

If both Input Stuck High and Input Stuck Low states are detected,
or both Output Stuck High and Output Stuck Low states are detected,
then the pin is covered.

Pins excluded (de-selected) by User:

Pin Name	Type

Other pins not in Fault Inject analysis.

Pin Name	Type

16 gnd	Unused
32 vcc	Unused

***** Detailed Results *****

- : fault detected
x : fault excluded
~det : fault not detected
<blank> : invalid state for pin type

Pin Name	Type	Input Stuck		Output Stuck		Pin Covered*
		High	Low	High	Low	
1 n1g	Stim	-	~det			No
2 1a1	Stim	-	-			Yes
3 2y4	Meas			~det	-	No
5 2y3	Meas			~det	-	No
7 2y2	Meas			-	-	Yes
9 2y1	Meas			-	-	Yes
12 1y4	Meas			-	-	Yes
14 1y3	Meas			-	-	Yes
15 2a3	Stim/Meas	~det	-	-	-	Yes
17 2a4	Stim/Meas	-	-	-	-	Yes
18 1y1	Meas			-	~det	No
19 n2g	Stim/Meas	~det	-	~det	-	No

* Pin Coverage is calculate as follows:

If both Input Stuck High and Input Stuck Low states are detected,
Or both Output Stuck High and Output Stuck Low states are
detected,
then the pin is covered.

Pins excluded (de-selected) by User:

Pin Name	Type

Other Pins not in Fault Inject analysis.

Pin Name	Type

4 1a2	Unused
6 1a3	Unused
8 1a4	Unused
10 Gnd	Unused
11 2a1	Unused
13 2a2	Unused
16 1y2	Unused
20 Vcc	Unused

The report window is too large to fit in one screen. To navigate in the displayed report, you can use the Home, End, Page Up, Page Down and arrow keys.

The fault detected dashes in the High and Low columns indicate that the tests on those pins failed with a fault injected and are, therefore, able to detect that fault with the test. The “~ det” message indicates that the injected fault was not detected and, therefore, the test to detect that fault is bad. The X indicates that the programmer chose not to inject a fault at the stuck high or low state.

Saving Fault Inject Report. When you press ESC to quit, a query window appears with a message similar to the following

Save Fault Inject results to: 74240.fij?

To save the file select Yes. If you do not want to save the results to a file, select No. The resulting file is a DOS file and resides in the test program directory.

Evaluating Fault Coverage at Board Level

Board Fault Coverage in the Utility menu enables you to run Fault Inject on specified components and generate a report on fault coverage at the board level.

The Board Fault Coverage utility uses two sources of data for fault coverage analysis: data derived from 18xx and data derived from sources such as LabWindows functional tests or Victory VIT tests.

Board Fault Coverage provides a component-oriented report. That is, when a single component is tested with multiple test steps, the coverage of all test steps is combined for the overall fault coverage report for that component.

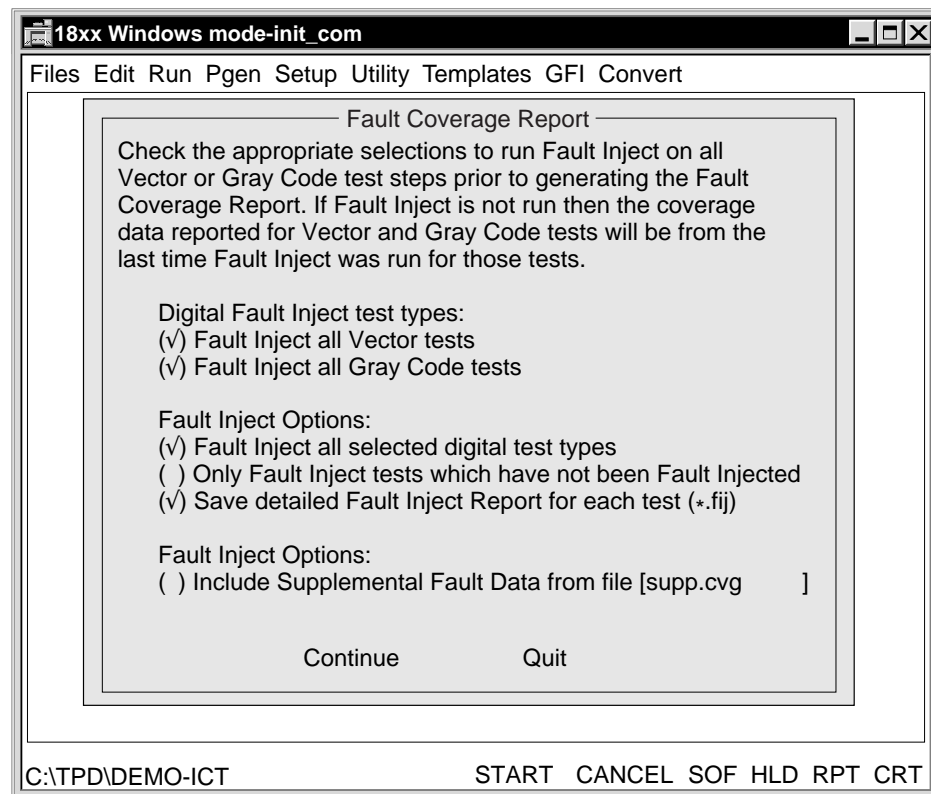
Board Fault Coverage reports on parallel components as well as single components. Parallel components, of course, are not actual components on the board. The parallel component's value is the electrical equivalent of the parallel circuit. The actual components on the board which are combined to make this parallel component are called "source components." When the program generator creates a parallel component, the source components which make up this parallel component are disabled.

For example if R1, R2, and R3 are all parallel and R1,2,3 is the test step which tests this parallel circuit (the Parallel Component), then each of R1, R2, and R3 is listed in the fault coverage report with the coverage that R1,2,3 has. The parallel Component named R1,2,3 is excluded from fault coverage calculations because it is not an actual component on the board under test.

To generate a board-level fault coverage report

- 1 Select Board Fault Coverage from the Utility menu.

The Fault Coverage Report window appears.



- 2 Click in the parentheses to toggle Fault Inject off or on.

If you do not run Fault Inject, the coverage data reported for Vector and Gray code tests will be from the last time Fault Inject was run for those tests. Running Fault Inject can be a time consuming process. If you merely want a current fault coverage report based on the last fault inject data gathered, toggle the Fault Inject options off. Components that have not had fault inject run on them will report zero coverage.

- 3 To include data derived from other sources in the report, check Include Supplemental Fault Data from File, and type the name of the file in the brackets.

See the Supplemental Fault Coverage section later in this chapter for more information.

- 4 Select Continue to proceed with the generation of the report or Quit.

The board level fault coverage report then appears.

Interpreting the Fault Coverage Report

The board level fault coverage report combines all the various fault reports into one large report. A fault coverage report similar to the version shown below appears.

Fault Coverage Report

Coverage for Board: UB90ALL

Board Directory : C:\TPD\UB90ALL - ICT

Date : Mon Jul 27 15:33:13 1998

Overall Board Coverage:

Total number of pins covered = 731

Total number of pins in analysis = 1030

Total Board Level Fault Coverage = 70.97%

Device Pins NOT Covered:

ID	Name	Pins(s)

J5	PINS 1 & 2	1
R5		1
R505		1
R950		1
R952		1
R953		1
R949		1
R967		1
R968		1
R970		1
R36	INDUCTOR	1
R43	INDUCTOR	1
C3	39 PF	1,2
R3	287.000	1,2
R38	10.00KO	1,2
R45	10.00KO	1,2
R506	33.000	1,2
R54	10.00KO	1,2
R8	150.000	1,2
R945	10.00KO	1,2
R964	100.000	1,2
R965	100.000	1,2
CR83		1,2
CR84		1,2
CR85		1,2
CR86		1,2
CR87		1,2
CR88		1,2

CR89		1,2
FL1	TLA471	1,2,6,7,9,11,12,14,15,16,17,18
U10	74FCT273	9
U11	EEPAL	13,14
U12	DMAPAL	5
U13	IRQPAL	2,4,5
U19	74F245	5,17
U2	RAM	1,2,3,4,5,6,7,8,13,14,15,17,18,19,20,21,31
U3	V30_LG	10,15,30
U4	20V8Q	2,10,17,18,27
U5	RAM	1,2,3,4,5,7,8,9,11,12,13,14,15,17,18,19,20,31
U7	581	23,34,41,50,84,87,89,115,203,204,205,206,207
U8	82C585	28,29,30,32,68
P3	24PINS	18
init	Z18xx, Counter	1,2
U9	RAMPAL	1,2,3,4,5,6,7,8,9,11,13,15,16
U7	581	3,4,5,6,7,8,10,11,12,13,15,16,17,18,20,21,22,23,24,25,28,29,30,31,32,33,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,54,55,56,57,58,59,61,62,63,64,66,67,68,69,71,72,73,74,75,76,77,80,81,82,83,86,88,91,93,94,95,96,97,99,100,101,102,103,109,110,111,112,113,114,115,116,117,118,121,124,125,126,127,128,129,132,133,134,135,136,137,138,139,142,143,144,145,146,147,148,149,150,151,152,153,154,158,159,160,161,162,163,164,167,168,169,170,171,172,174,175,176,177,178,179,180,184,185,186,187,188,189,190,191,192,193,194,196,197,198,199,200,201,202

Device Fault Coverage Data:

Device		Fault Coverage	Pins	Pins	Fault
ID	Name	Technique	Analyzed	Tested	Coverage
S2		Enabled	1	1	100.00%
J5	PINS 1 & 2	Disabled	1	0	0.00%
J7	PINS 1 & 2	Enabled	1	1	100.00%
R5		Disabled	1	0	0.00%
R37		Enabled	1	1	100.00%
R505		Disabled	1	0	0.00%
R951		Enabled	1	1	100.00%
R950		Disabled	1	0	0.00%
R951		Enabled	1	1	100.00%
R952		Disabled	1	0	0.00%
R953		Disabled	1	0	0.00%
R956		Enabled	1	1	100.00%
R949		Disabled	1	0	0.00%
R967		Disabled	1	0	0.00%
R968		Disabled	1	0	0.00%
R970		Disabled	1	0	0.00%
T1	Transformer	Enabled	6	6	100.00%
T2	Transformer	Enabled	4	4	100.00%
FL1	FILTER	Enabled	4	4	100.00%
FL3	FILTER	Enabled	1	1	100.00%
FL4	FILTER	Enabled	1	1	100.00%
L2	INDUCTOR	Enabled	1	1	100.00%

R36	INDUCTOR	Disabled	1	0	0.00%
R43	INDUCTOR	Disabled	1	0	0.00%
FL4	filter	Multiple	3	3	100.00%
FL3	filter	Multiple	3	3	100.00%
FL1	FILTER NETWORK	Enabled	18	18	100.00%
T1	100uh	Enabled	16	16	100.00%
T2	PT3983	Enabled	8	8	100.00%
L2	2.7 UH INDUCTOR	Enabled	2	2	100.00%
R36	2.2 UH INDUCTOR	Enabled	2	2	100.00%
R43	2.2 UH INDUCTOR	Enabled	2	2	100.00%
C3	39 PF	Disabled	2	0	0.00%
C6	10UF	Enabled	2	2	100.00%
C22	.47UF	Enabled	2	2	100.00%
C23	3.3UF	Enabled	2	2	100.00%
C1028-5W	4700UF	Enabled	2	2	100.00%
R10	10.00KO	Enabled	2	2	100.00%
R17	10.00KO	Enabled	2	2	100.00%
R18	10.00KO	Enabled	2	2	100.00%
R20	10.00KO	Enabled	2	2	100.00%
R21	10.00KO	Enabled	2	2	100.00%
R22	10.00KO	Enabled	2	2	100.00%
R23	10.00KO	Enabled	2	2	100.00%
R24	10.00KO	Enabled	2	2	100.00%
R25	10.00KO	Enabled	2	2	100.00%
R27	10.00KO	Enabled	2	2	100.00%
R28	33.000	Enabled	2	2	100.00%
R3	287.000	Disabled	2	0	0.00%
R31	10.00KO	Enabled	2	2	100.00%
R32	10.00KO	Enabled	2	2	100.00%
R33	10.00KO	Enabled	2	2	100.00%
R34	34.80KO	Enabled	2	2	100.00%
R35	39.200	Enabled	2	2	100.00%
R38	10.00KO	Disabled	2	0	0.00%
R39	10.00KO	Enabled	2	2	100.00%
R4	34.80KO	Enabled	2	2	100.00%
R40	10.00KO	Enabled	2	2	100.00%
R41	33.000	Enabled	2	2	100.00%
R42	10.00KO	Enabled	2	2	100.00%
R44	10.00KO	Enabled	2	2	100.00%
R45	10.00KO	Disabled	2	0	0.00%
R46	10.00KO	Enabled	2	2	100.00%
R47	10.00KO	Enabled	2	2	100.00%
R49	10.00KO	Enabled	2	2	100.00%
R50	39.200	Enabled	2	2	100.00%
R506	33.000	Disabled	2	0	0.00%
R51_6W	61.900	Enabled	2	2	100.00%
R52_5W	287.000	Enabled	2	2	100.00%
R53	10.00KO	Enabled	2	2	100.00%
R54	10.00KO	Disabled	2	0	0.00%
R55	10.00KO	Enabled	2	2	100.00%
R56	10.00KO	Enabled	2	2	100.00%
R57	10.00KO	Enabled	2	2	100.00%
R58	61.900	Enabled	2	2	100.00%
R59	287.000	Enabled	2	2	100.00%
R6	29.40KO	Enabled	2	2	100.00%
R60	10.00KO	Enabled	2	2	100.00%
R61	100.000	Enabled	2	2	100.00%
R62	10.00KO	Enabled	2	2	100.00%
R63	34.80KO	Enabled	2	2	100.00%

R64	10.00KO	Enabled	2	2	100.00%
R65	10.00KO	Enabled	2	2	100.00%
R66	10.00KO	Enabled	2	2	100.00%
R67	34.80KO	Enabled	2	2	100.00%
R68	1.00KO	Enabled	2	2	100.00%
R69	10.00KO	Enabled	2	2	100.00%
R7	150.000	Enabled	2	2	100.00%
R70	10.00KO	Enabled	2	2	100.00%
R73	100.000	Enabled	2	2	100.00%
R75	100.000	Enabled	2	2	100.00%
R8	150.000	Disabled	2	0	0.00%
R9	10.00KO	Enabled	2	2	100.00%
R945	10.00KO	Disabled	2	0	0.00%
R946	100.000	Enabled	2	2	100.00%
R947	10.00KO	Enabled	2	2	100.00%
R948	100.000	Enabled	2	2	100.00%
R954	33.000	Enabled	2	2	100.00%
R955	33.000	Enabled	2	2	100.00%
R958	10.00KO	Multiple	2	2	100.00%
R960	10.00KO	Multiple	2	2	100.00%
ratio		Enabled	2	2	100.00%
R961	270.000	Enabled	2	2	100.00%
R963	100.000	Enabled	2	2	100.00%
R964	100.000	Disabled	2	0	0.00%
R965	100.000	Disabled	2	0	0.00%
R966	2K	Enabled	2	2	100.00%
R969	10.00KO	Enabled	2	2	100.00%
CR5		Enabled	2	2	100.00%
CR6		Enabled	2	2	100.00%
CR83		Disabled	2	0	0.00%
CR84		Disabled	2	0	0.00%
CR85		Disabled	2	0	0.00%
CR86		Disabled	2	0	0.00%
CR87		Disabled	2	0	0.00%
CR88		Disabled	2	0	0.00%
CR89		Disabled	2	0	0.00%
Q1-BIAS		Multiple	3	3	100.00%
FL1	TLA471	DeltaScan	12	0	0.00%
SPARE2	20R4	Multiple	4	4	100.00%
U10	74FCT273	Multiple	18	17	94.44%
U11	EEPAL	Multiple	18	16	88.89%
U12	DMAPAL	Multiple	18	17	94.44%
U13	IRQPAL	Multiple	18	15	83.33%
U14	74F245	Multiple	18	18	100.00%
U15	74F245	Multiple	16	16	100.00%
U16	74F245	Multiple	18	18	100.00%
U17	74F245	Multiple	18	18	100.00%
U18	74F245	Multiple	10	10	100.00%
U19	74F245	Multiple	18	16	88.89%
U2	RAM	Multiple	29	12	41.38%
U20	9346	Multiple	4	4	100.00%
U3	V30_LG	Multiple	33	30	90.91%
U4	20V8Q	Multiple	18	13	72.22%
U5	RAM	Multiple	29	11	37.93%
U7	581	Multiple	165	152	92.12%
U8	82C585	Multiple	35	30	85.71%
U9	RAMPAL	Multiple	17	17	100.00%
P1	30PINS	FrameScan	25	25	100.00%
P3	24PINS	FrameScan	21	20	95.24%

P5	28PINS	FrameScan	25	25	100.00%
5V	VCC	Enabled	2	2	100.00%
init	Z18xx, Counter	Disabled	2	0	0.00%
20MhzOsc	STABILITY TEST	Enabled	2	2	100.00%
32MhzOsc	STABILITY TEST	Enabled	2	2	100.00%
U9	RAMPAL	Vector	17	4	23.53%
U7	581	Vector	156	1	0.64%

Devices tested with multiple techniques:

Device ID	Name	Fault Coverage Technique	Pins Analyzed	Pins Tested	Fault Coverage
FL4	filter	Enabled	3	3	100.00%
FL4	FILTER	Enabled	3	3	100.00%
FL4	FILTER	DeltaScan	1	0	0.00%
FL3	FILTER	DeltaScan	3	2	66.67%
FL3	filter	Enabled	3	3	100.00%
FL3	FILTER	Enabled	3	3	100.00%
R958_R	10.00KO	Enabled	2	2	100.00%
R958	10.00KO	Enabled	2	2	100.00%
R960_R	10.00KO	Enabled	2	2	100.00%
R960	10.00KO	Enabled	2	2	100.00%
Q1-BETA		Enabled	3	3	100.00%
Q1-BIAS		Enabled	3	3	100.00%
SPARE2	20R4	DeltaScan	4	0	0.00%
SPARE2	20R4	WaveScan	4	4	100.00%
U10	74273	Gray Code	17	16	94.12%
U10	74FCT273	DeltaScan	17	17	100.00%
U10_HOME	74273	Vector	17	6	35.29%
U10	74FCT273	WaveScan	17	17	100.00%
U11	EEPAL	DeltaScan	16	14	87.50%
U11	EEPAL	WaveScan	16	15	93.75%
U11	EEPAL	Vector	17	1	5.88%
U12	DMAPAL	DeltaScan	17	16	94.12%
U12	DMAPAL	Vector	18	2	11.11%
U12	DMAPAL	WaveScan	18	17	94.44%
U13	IRQPAL	DeltaScan	18	14	77.78%
U13	IRQPAL	WaveScan	18	15	83.33%
U14	74F245	DeltaScan	18	18	100.00%
U14_MIX	74f245	Vector	16	0	0.00%
U14	74f245	Gray Code	18	17	94.44%
U14	74F245	WaveScan	18	18	100.00%
U15	74f245	Gray Code	16	15	93.75%
U15	74F245	WaveScan	16	16	100.00%
U15	74F245	DeltaScan	16	16	100.00%
U16	74f245	Gray Code	18	17	94.44%
U16	74F245	WaveScan	18	18	100.00%
U16	74F245	DeltaScan	18	18	100.00%
U17	74F245	WaveScan	18	18	100.00%
U17	74f245	Gray Code	18	0	0.00%
U17	74F245	DeltaScan	18	18	100.00%
U18	74F245	WaveScan	10	10	100.00%
U18	74F245	DeltaScan	10	10	100.00%
U18	74f245	Gray Code	10	9	90.00%
U19	74F245	WaveScan	18	15	83.33%
U19	74F245	DeltaScan	18	16	88.89%
U2	ram	Disabled	32	0	0.00%
U2	RAM	DeltaScan	29	12	41.38%

U2	RAM	WaveScan	29	3	10.34%
U20	9346	WaveScan	4	4	100.00%
U20	9346	DeltaScan	4	4	100.00%
U20	9346	Gray Code	4	0	0.00%
U3	V30_LG	WaveScan	33	29	87.88%
U3	V30_LG	DeltaScan	33	15	45.45%
U4	20V8Q	DeltaScan	11	4	36.36%
U4	iowr_pal	Vector	14	4	28.57%
U4	20V8Q	WaveScan	11	8	72.73%
U5	RAM	DeltaScan	29	8	27.59%
U5	RAM	WaveScan	29	7	24.14%
U5	RAM	Disabled	32	0	0.00%
U7	82C581	WaveScan	165	145	87.88%
U7	581	DeltaScan	164	148	90.24%
U8	82C585	DeltaScan	35	19	54.29%
U8	82C585	Disabled	68	0	0.00%
U8	82C585	WaveScan	35	29	82.86%
U9	RAMPAL	WaveScan	17	17	100.00%
U9	RAMPAL	DeltaScan	16	16	100.00%

Test Steps not included in the Fault Coverage Analysis:

ID	Name	Section

vcc	vcc	Discharge
Cont.	Cont.	Continuities
Ignores		Special Case/Ignores
All SC		Special Case/Merge
Shorts	Shorts	Shorts
vcc/gnd	vcc-gnd	Capacitors

The fault coverage report is divided into 5 parts: Report Header, Device Pins NOT Covered, Device Fault Coverage, Devices Tested with Multiple Techniques, and Test Steps not included in the Fault Coverage Analysis.

Report Header. The Report Header section of the report identifies the Board Name, the directory path to the board, the date and time the report was generated, and the Overall Board Fault Coverage results.

Device Pins NOT Covered. After the board's total fault coverage is displayed, the device pins not covered are presented. This listing contains the ID and name for each device not 100% covered by the ICT program. For those devices not covered, the actual pins which the ICT program does not test are identified. This report appears at the beginning of the Fault Coverage Report to highlight areas of your board which are not being tested and which need work to improve coverage.

Device Fault Coverage. The Device Fault Coverage lists fault coverage for every component in the ICT program. Listed next to each component is the test technique used to calculate the coverage, the number of pins covered, the number of pins analyzed, and the fault coverage percentage.

Devices Tested with Multiple Techniques. If the fault coverage technique from the Device Fault Coverage section of the report is Multiple, then that component has entries in this section. This section provides a break down of components tested with multiple techniques. For Example if U43 is included twice in the ICT program, once as a DeltaScan Test and again as a Vector Test, it will have Multiple listed as its test technique in the Device Fault Coverage section. In this section, U43 will have two entries, one for the actual coverage obtained by the DeltaScan Test and another for the coverage obtained by the Vector Test.

Test Steps Not Included in the Fault Coverage Analysis. The final section of the Fault Coverage Report indicates all of the test steps which are not included in the Fault Coverage calculations. This list includes Discharge Steps, Interconnect Steps, No Test Steps, and Parallel Components.

Fault Coverage Techniques

The following table shows the fault coverage techniques for each 18xx test type.

18xx Test Type	Fault Coverage Technique
APC	None
Beta	Enabled or Disabled
Capacitor	Enabled or Disabled
Cap Phase	Enabled or Disabled
CapScan	Enabled or Disabled
Continuity	None
DeltaScan	DeltaScan
DigFuncProc	None
Diode	Enabled or Disabled
Discharge	None
External Program	Enabled or Disabled
FrameScan	FrameScan
FrameScan2	FrameScan
Gray Code	Gray Code
Gray Code Disable	None
Header Data	None
Header/Trailer	None
IEEE	Enabled or Disabled
Ignores	None
Inductor	Enabled or Disabled
Jumper	Enabled or Disabled
LabWindows	Enabled or Disabled
No Test	None
Opens	None
Potentiometer	Enabled or Disabled
Power	Enabled or Disabled
Power 5V	Enabled or Disabled
Power 5.5V	Enabled or Disabled
Power Prog A	Enabled or Disabled
Power Prog B	Enabled or Disabled
Power Prog Slaved	Enabled or Disabled
Resistor	Enabled or Disabled
Shorts	None
Special Case	None
Stim V	Enabled or Disabled
Test Digital Stim V	None
Test I Stim V	Enabled or Disabled
Test I Stim V Stim V	Enabled or Disabled
Test V	Enabled or Disabled

18xx Test Type	Fault Coverage Technique
Test V Stim Digital	None
Test V Stim I	Enabled or Disabled
Test V Stim I Stim V	Enabled or Disabled
Test V Stim V	Enabled or Disabled
Test V Stim V Stim V	Enabled or Disabled
Transistor	Enabled or Disabled
Vector	Vector
Vector Disable	None
WaveScan	WaveScan
Zener	Enabled or Disabled

Test Techniques: MultiScan

You can do an analysis on WaveScan, FrameScan, FrameScan Plus, and DeltaScan to determine fault coverage.

WaveScan, FrameScan, and FrameScan Plus use the pin level fault coverage from the Statistics report using the following formula.

$$\text{Formula: } \frac{\text{Total Wave/FrameScan pins Tested}}{\text{Total Wave/FrameScan pins included in analysis}} \times 100$$

The following pin types are not included in the analysis: Power, Ground and N/C. Normal pins and Normal Tied pins are recorded as covered; Tied and Not Tested pins are recorded as not covered.

DeltaScan tests always contain an up-to-date fault coverage percentage which is used in the board-level fault coverage report.

DeltaScan uses the following formula to determine coverage:

$$\text{Formula: } \frac{\text{Total DeltaScan pins Tested}}{\text{Total DeltaScan pins included in analysis}} \times 100$$

The following pin types are not included in the analysis: Pwr, Gnd, and N/C.

See the **MultiScan User's Guide** for additional information about MultiScan Fault Coverage.

Test Techniques: Enabled and Disabled

Any test may be enabled or disabled. An enabled test is 100% covered, and a disabled test is 0% covered.

$$\text{Formula: } \frac{\text{Total pins tested}}{\text{Total pins included in analysis}} \times 100$$

The total pins included in the analysis is equal to the number of pins on the device. The total number of pins tested is equal to the number of pins on the device if the test is enabled and is equal to 0 if the test is disabled. If any pages of the test are enabled, then all of the pins of the test step are said to be tested. If all pages of the step are disabled or the entire test step is disabled, then none of the pins are tested.

Test Techniques: Gray Code

Pin types not included: Power, Gnd, NC, Tied, StimV, MeasV

$$\text{Formula: } \frac{\text{Pins Tested}}{\text{Pins included in analysis}} \times 100$$

Test Techniques: Vector

Pin types not included: Unused, Monitor, StimV, MeasV

$$\text{Formula: } \frac{\text{Pins Tested}}{\text{Pins included in analysis}} \times 100$$

Test Techniques: None

Tests identified as None have no affect on board level fault coverage and so are not included in board-level fault coverage.

Supplemental Fault Coverage

The board fault coverage report can be supplemented by data imported from a text file. The Supplemental Fault Coverage option in Fault Coverage Report provides a means to import such data created outside the 18xx system software.

By default, Supplemental fault coverage is disabled. When enabled, you can specify the name of the file which contains the supplemental fault coverage data. The default name of the file is "supp.cvg." This file must reside in the board directory. If the supplemental fault coverage is enabled, and the file is missing or contains syntax errors, the error messages are displayed, and the Board Fault Coverage report will not be generated.

The file format is as follows:

```
<component entry> [<component entry>...]

<component entry> = '[' <id> [<name>] <technique> <npins>
                   <covered> <not covered> <not included> ']'

<id>= ID: <id string>
<name>= NAME: <name string>
<technique>= TECHNIQUE: <tech string>
<npins>= NPINS: <integer>
<covered>= PINS_COVERED: <pins list>
<not covered>= PINS_NOTCOVERED: <pins list>
<not included>= PINS_NOTINCLUDED: <pins list>

<pins list>= <pin list entry> [,<pin list entry> ...]
<pin list entry>= <pin number> | <pin range>
<pin range>= <pin number> - <pin number>
<pin number>= <integer>
```

Example:

```
[
ID: U33
NAME: 7400
TECHNIQUE: Supplemental
NPINS: 14
PINS_COVERED: 1, 2, 3, 4, 5, 6, 8 - 13
PINS_NOTCOVERED:
PINS_NOTINCLUDED: 7, 14
]
```

* A line in which the first character (excluding white space) is the “#” is treated as a comment and ignored by the parser.

** Blank lines and white space can be inserted freely.

Reporting Supplemental Fault Coverage

A component listed in a supplemental fault coverage file is considered to be the same component as a component tested by an 18xx ICT program if the Component ID and number of pins are the same.

As each 18xx test step’s coverage is reported, the supplemental coverage file is searched to determine if the component test by the current 18xx test step has supplemental coverage as well. If an 18xx test step has supplemental coverage, then it will be listed as a component with “Multiple” coverage techniques. Each 18xx test step which tests a particular component and each of the entries in the supplemental coverage file for this same component are listed in the ‘Devices tested with multiple techniques:’ section of the report.

Overall coverage for this component is determined by combining the coverage from both the 18xx test(s) and supplemental test(s) and reported in the “Device Fault Coverage Data” section of the report. As components from the supplemental are reported they are marked so that they cannot be reported twice.

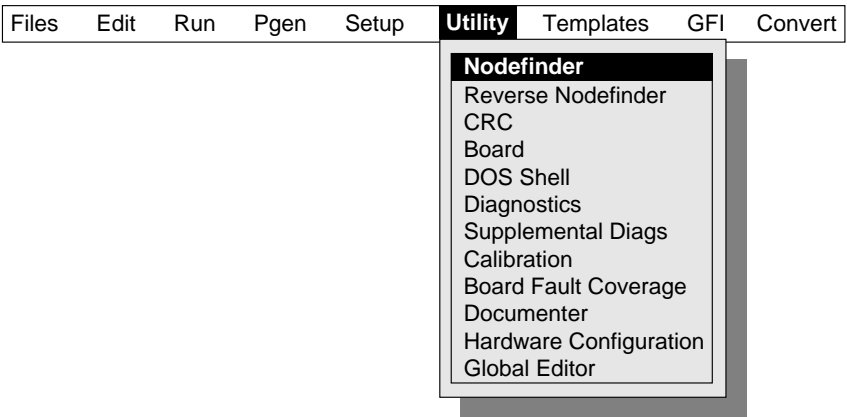
After all of the 18xx test steps and the entries in the supplemental coverage for these components have been reported the fault coverage report generator scans the supplemental coverage file for any entries not yet reported. Entries left in the supplemental coverage file identify components which are not represented in the 18xx ICT program. Each of the remaining entries in the supplemental coverage file is then reported. There may be components listed twice in the supplemental coverage file, in this case they will be listed individually in the “Devices tested with multiple techniques” section of the report. Then, as with all components tested with multiple techniques, the total coverage is calculated and the coverage is reported in the “Device Fault Coverage Data” section of the report.

All other entries in the supplemental coverage file which have not been reported have their coverage reported to the “Device Fault Coverage Data” section of the report.

As with 18xx test steps, if a component tested partially or completely with supplemental techniques is not 100% covered, then those pins which are not covered are reported to the “Device Pins NOT Covered” section of the report.

Software Utilities

Select the Main menu and then the Utility menu to access a wide variety of program development and debugging tools.



Utility	Description
Nodefinder	Nodefinder probe to learn the channel number of a node in the board under test.
Reverse Nodefinder	Reverse Nodefinder probe to learn the physical location of a specific node number.
CRC	Calculate the expected CRC signature value of a known bit sequence.
Board	Determine the driver/receiver board for a known node.
DOS Shell	Enter the DOS environment. Type exit to return.
Diagnostics	Perform diagnostic tests on system. See Z1800-Series Maintenance Reference for more information.
Supplemental Diags	Run existing or add new subprograms. See Z1800-Series Maintenance Reference for more information.
Calibration	Access software to calibrate ATB and clock.
Board Fault Coverage	Generate a report on fault coverage at the board level and run Fault Inject on specified components.
Documenter	Provide a readable, hard copy document of how a board is tested.
Hardware Configuration	Verify tester's hardware configuration matches test program configuration file in board directory.
Global Editor	Modify ITC files and use as a mechanism for adding guards and changing test limits.

Learning
Component Lead
Node

Nodefinder identifies the node number of a component lead on the board-under-test, a useful tool to help compile an input list. There are two ways to use Nodefinder—from the Main menu or from a Step Worksheet.

IMPORTANT: You can also access Nodefinder from the DOS command line. Type p18xx after the DOS prompt.

- 1 To use Nodefinder from the Main menu, select Nodefinder from the Utility menu.

The following message will appear on your screen:

Nodefinder -- Any key to terminate, Cancel to Abort

- 2 To account for Nodefinder resistance, set the Nodefinder probe impedance in Setup's Environment menu.

To install the Nodefinder probe, insert the end of the jack into the Nodefinder receptacle (Node Probe on a Z1860/40) located on the test jack panel.

WARNING!

Do not probe a component when power is applied to a board-under-test. Damage to the board and the test instrumentation may result. The software does provide built-in safety measures since Nodefinder automatically turns off power. When probing power-on sections like digital test sections, the software turns power off during the probe and restores it afterward.

Embedded Step Worksheet Nodefinder

You can use Nodefinder to find nodes in both analog and digital tests. You must first have the board under test engaged in its fixture, and the fixture mounted in the fixture receiver before you can use Nodefinder.

The general procedures for using Nodefinder are as follows:

- Access Nodefinder from within Step Worksheets by selecting Edit Device Nodes from the Tools menu. Nodefinder is active only in the Node entry window of the Component Properties' Number of Pins field.
- Press F9 or double click with the left mouse button on a node field to activate Nodefinder. (If you selected Nodefinder Display Off in the Setup/Environment menu, the display does not change.) You may track the progress of nodes found in the Nodes Entry pop-up window.

To find nodes, touch a component's pins with the Nodefinder probe. Nodefinder will issue a confirmation sound, update the current node field, and advance to the next pin. Press any key to skip pins you do not want to probe. Press ESC or Cancel to terminate node finding.

If you set Nodefinder Display On, the standard Nodefinder display will appear so you can check the nodes found before accepting them into the Node Entry fields. To confirm each node and transfer them into the Node Entry pop-up window, press any key.

You can build a partial input list containing an entry for each component on the board and null node (9999) entries instead of actual node numbers. The partial input list may then be used to generate a dummy program. (Without actual node numbers the program generator is unable to perform functions like paralleling bypass capacitors, generating RC combinations, and intelligent discharge statements.) You can then learn the node numbers for each component with Nodefinder and F9.

Identifying Analog Nodes. To identify the analog nodes of Passive components

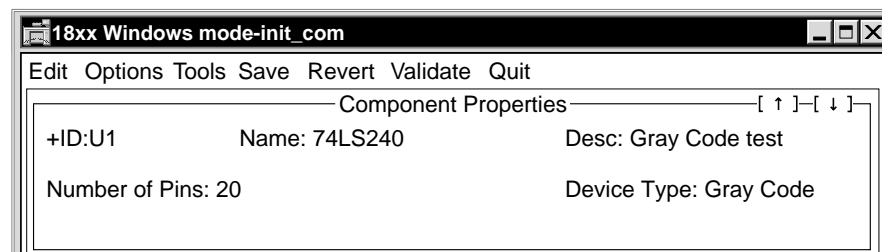
- 1 Select the Edit menu from Main menu bar.
- 2 Select Passive.
- 3 Select either Inductors, Capacitors or Resistors.
Each of these choices takes you to the Component ID menu.
- 4 Select Add to get a blank Component Properties portion of the Step Worksheet.
- 5 Fill in Component Properties.
 - Type in an appropriate ID
 - Fill in the Name field with the component name—R1 for resistor, L1 for inductor, and C1 for capacitor, for example.
 - Fill in the Desc. field with component descriptor information.

- Fill in the Value and Tolerance fields.
- 6 Select the Number of Pins field. The Node entry window appears.
 - 7 Place the cursor on Pin 1 and press F9 or single click to activate Nodefinder.
 - 8 Probe the pins of the device.
Nodefinder automatically advances to the next pin when a node is found if the Setup/Environment variable Nodefinder Display is set to Off. If the Environment variable is On, a confirmation window appears which requires you to press any key to advance to the next pin.
 - 9 Select Tools/Generate Test to create Test Properties for this device.
 - 10 Fill in the Test Properties.
Enter the name, parameters, and connections for one component.
 - 11 Press F10 to return to the Individual Component menu.
 - 12 Choose Save to save the information just entered.
 - 13 Choose Quit to enter the Component ID menu.
 - 14 Choose Add to enter the next component, or if the next component has the same value and tolerance, choose Copy to copy the data from the first component into a new Test Properties.
Copy is convenient when you're working with a parts list, as parts lists tend to be sorted into groups of similar components.
 - 15 Choose ID and change the Name field to the name of the new component.
 - 16 Choose Number of Pins field and probe the node numbers with F9 for the new component.
 - 17 Press F10 to return to the Individual Component menu.
 - 18 Choose Save.

Identifying Digital Nodes. Digital IC tests are generated from a library of templates. A template is a test without the node numbers filled in.

To identify node numbers

- 1 Choose Digital/Components from the Edit menu to get to the Component ID menu.
- 2 Choose Add.
A blank Component Properties portion of the Step Worksheet appears.
- 3 Choose ID.
Enter the Component's Name and ID. The component's name must be the same as its catalog name in the Z1800-series template library. Below is a sample Component Properties for a 74LS240 chip.



- 4 Select Get Template Pins from the Tools menu.
If the device type is in the library, the correct number of pins for this device will be entered. If it is not in the library, you can type this number. Enter the number (between 1 and 128).
- 5 Begin probing.
At this point, you can select the Number of Pins field by clicking it or pressing Enter and begin probing to learn a node number for the first IC pin. Place the highlight on the node number field next to pin 1.
Press F9. Touch the probe tip to the pin in question. If more than one node number is found, the first one found will be selected.
If the probed pin is not connected to the tester, no node number is found. To end a fruitless probing cycle, press any key to move to the next field. Leave the unfilled node number field blank. It will show up in the Input List as 9999, and in the final program as N/C.
- 6 When you have probed all the pins, select Tools/Generate Test to obtain the component test from the library.
- 7 Press F10 to return to the menu.
- 8 Choose Save.
- 9 Choose Quit to enter the Component ID menu.
- 10 Choose Add (or Copy if the same type of IC) to add the next test.
- 11 Choose Update from the PGen menu to update the component database.
The component database is created from the information stored in the in-circuit program.
The screen displays the following messages:
Recreating Component Database
Creating Node Index

Your database is now complete. You can add components not yet included into the program during debugging.

Locating a Node Number on a Board

Reverse Nodefinder identifies the physical location (the component lead) of a known node number.

Select Reverse Nodefinder from the Utility menu to access Reverse Nodefinder. The following information appears on your screen:

```
Node Locator
Search for node:0
Range (0 - 2047)
```

When you type the node number in the field and press Enter, the following message appears:

```
Nodefinder -- Any key to terminate, Cancel to abort
Locating Node: 6
```

Probe with the Nodefinder probe. The software beeps when you touch the correct component lead. The tone is different for right and wrong nodes.

Learning a CRC

The CRC utility calculates and reports the cyclic redundancy check (CRC) signature of any string of binary digits (1s and 0s) multiplied by a repetition factor between 1 and 32766.

When you place the cursor on various fields, the following messages appear on the bottom line:

Generate	Generate the Crc Signature
Clear	Clear the Data Register
Repetition Factor	Repetition Factor (1-32766)

To generate a CRC signature, select CRC from the Utility menu. The following screen appears.

Crc Generator

Generate Clear

Data:
 Repetition Factor: 1
 Crc Signature: 0000

Generate the Crc signature.

Highlight the Data field. Type in the binary data into the Data field and press Enter. Type in the Repetition Factor and select Generate. the CRC code will appear in the CRC field as shown below.

Crc Generator

Generate Clear

Data: 111000111000111
 Repetition Factor: 1024
 Crc Signature: 6CD1

Generate the Crc signature.

Locating a Driver/Receiver Board

The Board utility, also available from the Utility menu, allows you to determine the number of the driver/receiver board for a given node. The range of boards is 0 through 159 to cover the possible 160 board slots. Board also indicates the pin or node position or channel number between 0 and 31 on the specified board.

To determine the driver/receiver board for a node

- 1 Select Board from the Utility menu.

The following screen will appear.

Board

Enter node number and press
 enter to find out which board
 this node is on.

Node Number: 0

Enter the Node Number

- 2 Enter the node number in question.

The tester responds:

Node 656 is on board: 20position: 16

Remember that both node and board numbering start at 0, not 1.

DOS Shell

The DOS Shell function enables you to exit from the 18xx environment to perform a wide variety of tasks. When you have completed your operations in DOS, typing exit puts you directly back into 18xx again. Typical operations associated with test development include

- editing CONFIG.SYS and AUTOEXEC.BAT files
- editing PGEN.CFG
- listing files within a test program directory
- copying individual files between directories or changing file extensions
- copying files or directories from hard disk to floppy diskettes for backup or to transfer between machines
- printing files (such as from Documenter or from Pgen/Reports)
- viewing ASCII files
- editing ASCII models for the vector processor
- editing or creating input lists for use with Pgen
- utilizing communication protocols such as telnet or ethernet

Diagnostics

Running Diagnostics at regular intervals verifies the proper operation of your tester; to run, select Diagnostics from the Utility menu. The Diagnostics window will appear enabling you to specify the tests and the method in which you wish to run them—chained or individually.

For further information about running the test, or about the tests themselves, refer to chapter 4 of the **Z1800-Series Maintenance Reference**.

Supplemental Diagnostics

The Supplemental Diagnostics menu selection enables you to run a custom diagnostics program from the 18xx environment. The 18xx system software handles custom diagnostics programs as it does other subprograms: you must edit your custom diagnostics program to chain it to the main program or to add other test steps. You can chain up to three subprograms.

If you install options having supplemental diagnostics, the diagnostics will be installed in subdirectories of \TPD\((DIAGS).

To run Supplemental Diagnostics, select Supplemental Diagnostics from the Utility menu. The run-time screen appears, and the board name field at the bottom of the screen displays

D:\TPD\((DIAGS) - ICT

Calibration

The calibration utility provides access to the software that enables you to calibrate the ATB and the clock. Select Calibration from the Utility menu. The Calibration window appears allowing you to select the ATB or clock.

For calibration support, call 1.800.HLP.TEAM (1.800.457.8326) in North America or your local support center elsewhere.

Documenter

Documenter can be used to provide an ASCII text record of the contents of a specific test program. Since Documenter does not run concurrently with test execution, its output does not contain a record of test results. This listing, especially designed to be readable by persons unfamiliar with programming language, maps an 18xx Step Worksheet (ICT.TST file) program into an ASCII file.

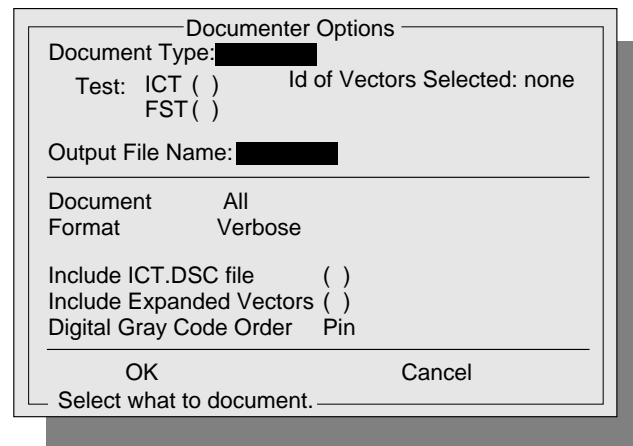
Depending on the size of your test program and whether you wish to output expanded vectors, the file size generated by Documenter can be considerable. The following table shows the relative

file size for the three output modes.

Mode	Complete Program Documentation	File Size
Verbose	Yes	Large file
Don't Print Default Options	Yes	Large file, but smaller than verbose
Terse	No	Smallest file

It should be noted that a normal program produces a file which, if printed, may consume large amounts of paper. For example, a typical verbose file may contain about 500 kilobytes of text, and if expanded vectors are included in the file, it can run to a megabyte or two.

When you select Utility/Documenter, the following window appears.



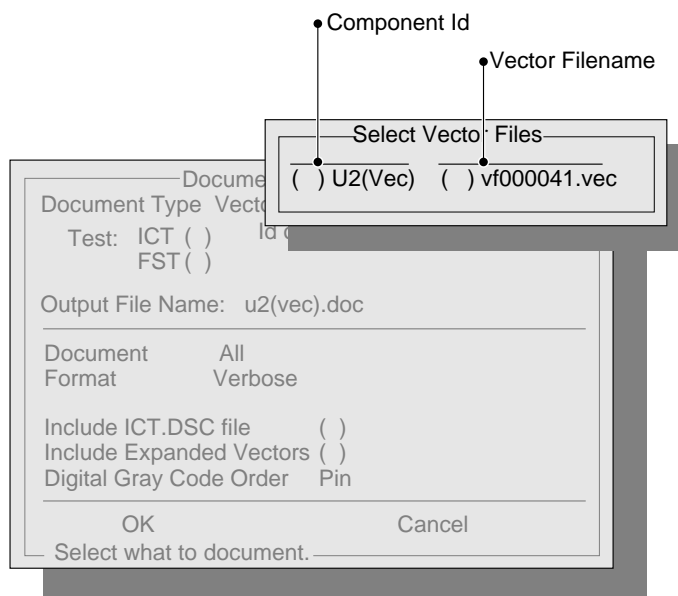
The window's second section, which includes the Document, Format, Include Expanded Vectors and Digital Gray Code Order fields, is enabled only when you select Test from Document Type.

To set up Documenter

- 1 Click the Document Type field, and select either Test to document a test program or Vector for a single vector file.
- 2 Select what type of test you wish to document.

If you have selected Test, the Test: ICT/FST fields are active. You may choose to document the in-circuit test program, the fixture selftest program, or both.

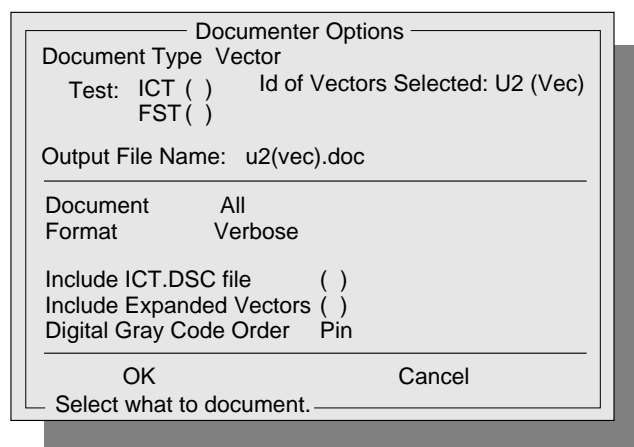
If you have selected Vector, the Id of Vectors selected field pops up a menu containing a list of vector file names with the associated component identifiers.



You may then select one or more vectors to be documented. When you exit this menu, the Output File Name field changes to reflect your choice of vector(s).

- If you do not select a vector, the Output File Name is reset to the original file name.
- If you select one vector, the Output File Name is the ID field of the test step with the “.DOC” extension.
- If you select more than one vector, the Output File Name is set to VECFILES.DOC.

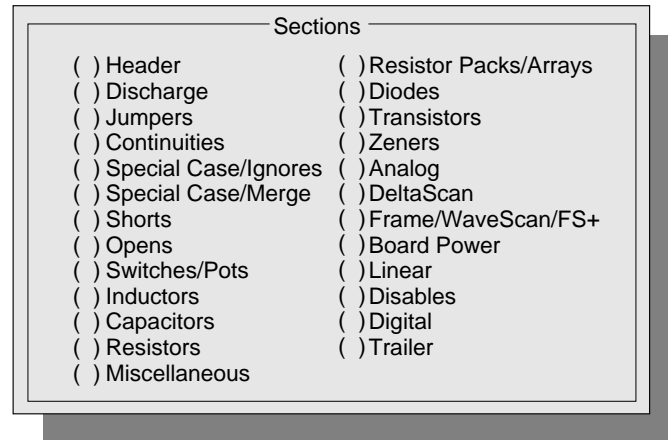
The following shows the Documenter window after selecting one vector.



- 3 Type in the name of the text file for Documenter output in the Output File Name field, or use the default provided.
- 4 If you selected Vector from Document Type, click OK now to document the test or Cancel to cancel without documenting the test.
- 5 If you selected Test, continue to step 6 to make further specifications in the second section of the Documenter window.

- 6** Specify in the Document field whether you want to document all sections of the test program or only certain ones.

All is the default. If you want to specify certain sections, click the Document field and select Sections from the pop-up window. The Sections window will appear as shown below.



- 7** Select the sections for documentation by clicking between the parentheses.
- 8** From the Format field select the type of output you want:
- | | |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Verbose | Everything is printed |
| Don't Print Default Options | The Pre- and Post-Test Options window and Control Options window are printed only if one or more variables in the window are different from the default. |
| Terse | Only the test data required to describe the test is printed. |
- 9** Click the Include ICT.DSC file field if you want to include the description file for this board (if it exists).
- This file will be output after the Title page and before the Setup/Environment Data screens. See the discussion on the F11 key for more information about the ICT.DSC file.
- 10** To include expanded vectors in the test program, click in the parentheses in the Include Expanded Vectors field.
- Otherwise the translation of each vector in a digital component test step is output to a file with the name of the test step's ID with .DOC appended.
- 11** To specify the order in which the Gray code tests are printed—by pins or burst group order—click Digital Gray Code Order and select Pin or Group from the pop-up window.
- 12** Click OK or Cancel.

IMPORTANT: Pressing the ESCape key terminates Documenter.

To represent each test step, fields are collated into groups based on functionality and which subwindow they appear in. For example, Component Properties usually contains the fields—ID, Name, Desc, Value, Number of Pins, and so forth. These groupings are referred to as Statement windows.

The overall format for the documented output will display the program's name and the header information on the first pages. Following this, at the top of the next page, the Header section will begin. Each subsequent section will begin at the top of a page with the section name. If the section contains no test steps, this fact will be noted at the top of the file, and that section will be skipped in the body of the text.

Global Editor

The Global Editor mechanism for 18xx system software allows data in a predefined ASCII format to be imported into a program. Use Global Editor to

- modify in-circuit test (ICT) files
- add guards and change test limits.

Access Global Editor from the Utility menu or directly from the 18xx command line. Refer to Using Global Editor and Global Editor Command line later in this section.

Overview of the Global Editor Structure

An input file for Global Editor is a linear list of agents called an Agent Input file. An agent is a statement that causes change; it is made up of a list of zero or more qualifiers followed by the agent type and a list of one or more actions. The list of qualifiers identifies the test step(s) and/or page(s) that the list of actions will be applied to. If no qualifiers are present, then the list of actions will be applied to every page of every test step.

The list of actions starts with an agent type. The agent types are

- component disable
- component enable
- component delete
- convert to ATB
- convert to PRISM
- modify

Only the agent type :modify, requires a list of actions to specify the fields to change and their new values.

The six formats for an agent are:

```
<list-of-qualifiers> : comp_enable;
<list-of-qualifiers> : comp_disable;
<list-of-qualifiers> : comp_delete;
<list-of-qualifiers> : conv_to_ATB;
<list-of-qualifiers> : conv_to_PRISM;
<list-of-qualifiers> : mod <list-of-actions>;
```

Note that a colon separates the qualifiers from the actions, and a semi-colon terminates the agent.

Qualifiers act to narrow the number of test steps that can be modified. Actions affect only the selected, or qualified, test steps. Each action will be applied to all qualified test steps.

Qualifiers and actions have very similar formats called "Attribute-Value pair." Every Attribute-Value pair consists of an attribute (a field identifier), an operator (such as == or +=), and a value. If the Attribute-Value pair is a qualifier, then the format is:

```
<attribute> <relational-operator> <value>
```


If the Attribute-Value pair is an action, then the format is:

`<attribute> <assignment-operator> <value>`

The only difference between a qualifier Attribute-Value and an action Attribute-Value is the type of operator.

White space can be placed anywhere within the file and will be ignored. You can comment your code using either of the C-style comments: `//` or `/* ... */`.

Agent Input File Format Synopsis

File:<agent> ; [<agent> ; ...]

agent:<list-of-qualifiers> : COMP_ENABLE|

<list-of-qualifiers> : COMP_DISABLE|

<list-of-qualifiers> : COMP_DELETE|

<list-of-qualifiers> : CONV_TO_ATB|

<list-of-qualifiers> : CONV_TO_PRISM |

<list-of-qualifiers> : MODIFY <list-of-actions>

list-of-qualifiers: [<qualifier> [, <qualifier> ...]]

list-of-actions: <action> [, <action> ...]

qualifier: <attribute> <relational-op> <value>

action: <Attribute> <assignment-op> <value>

relational-op: < | > | <= | >= | == | !=

assignment-op: = | += | -=

Below is a listing of all Attribute-Value pairs supported. The <op> is either a relational operator or assignment operator depending on whether this pair is used as a Qualifier or an Action.

The following are component-related Attribute-Value pairs:

ALL_NODES	<op> <node_mod_list>
COMP_NODES	<op> <node_list> <node_mod_list>
COMP_VALUE	<op> float
COMP_VALUE1	<op> float
COMP_VALUE2	<op> float
DESC	<op> <string>
DEVICE_TYPE	<op> < device_type>
ID	<op> <string >
NAME	<op> <string >
SECTION	<op> <section_string >
TOL	<op> <tol_val>
TOL1	<op> <tol_val>
TOL2	<op> <tol_val>

The following are test Step Worksheet-related Attribute-Value pairs:

AUTO_RANGE	<op> ON OFF
AVERAGING	<op> <int>
BW_LIMIT	<op> ON OFF
DIG_CLOCK_DIV_GC	<op> <int>
DIG_CLOCK_DIV_VEC	<op> <int>
DIG_CLOCK_SOURCE	<op> INTERNAL EXTERNAL
DIG_HIGH_THRESH	<op> <float>
DIG_LOW_THRESH	<op> <float>
DIG_LOGIC_LEVEL	<op> 3 VOLTS 5 VOLTS
DIG_MEAS-DELAY	<op> <int>
DIG_TERMINATOR	<op> NONE PWR5 1K UP
EXTENDED_MODE	<op> ON OFF
FAST_MODE	<op> ON OFF
GUARD_MODE	<op> PASSIVE SEMI_ACTIVE ACTIVE
GUARD_NODES	<op> <node_list>
HIGH_VALUE	<op> <test_val>
HIGUARD	<op> ON OFF
LINE_REJECT	<op> ON OFF
LOW_VALUE	<op> <test_val>
MEAS_NODES	<op> <node_list>
MEAS_PIN	<op> <int> <pin_type>
PAGE	<op> <int>
PINS	<op> (pin_num [, pin_num,...]) (pin_type [, pin_type, ...])
PRE-CYCLE_VACUUM	<op> YES NO
PRECISE	<op> ON OFF
PRE_REPEAT_COUNT	<op> <int>
PRE_REPEAT_DELAY	<op> <int>
PRE_REPEAT_MODE	<op> CONTINUOUS WHILE FAILING
PRE_REPEAT_PAGE	<op> ON OFF
REVERSE_TEST	<op> YES NO
SAMPLES	<op> <int>
SQUELCH	<op> <int>
STIM_NODES	<op> <node_list>
STIM_PIN	<op> <int> <pin_type>
TEST_NODES	<op> <node_list> <node_mod_list>
TEST_TYPE	<op> <test_type>
TEST_VALUE	<op> <float>

WAIT	<op>	<int>	
WIRE	<op>	3 4 5 6 NONE	
pin_num:		<int>	
pin_type:		BASE	
		EMITTER	
		COLLECTOR	
		WIPER	
		LEAD1	
		LEAD2	
		ANODE	
		CATHODE	

Pin Types can be used with PINS, STIM_PIN and MEAS_Pin attributes. Pin Types are limited to a specific set of devices. Using pin types on other devices will result in a warning message.

The allowed set of devices are dependent on the pin type:

Pin Type	Legal Device Types
Base, Emitter, Collector	Transistor-NPN, Transistor-PNP, Beta-NPN, Beta-PNP
Lead1, Wiper, Lead2	Potentiometer, Rheostat
Anode, Cathode	Diode, Zener

node_list: (<node> [,<node>])
node_mod_list: (<node_mod> [,<node_mod>])
node_mod: <node> TO <node>
node: <int>
tol_val: <float> '%'
test_val: <float> | <float> '%'
string:

A string can include spaces, tabs and any printable character except the delimiters: comma, semi-colon, colon, and slash (, ; : /). The beginning and end of the string starts (and ends) with the first (last) non-white space character. To include a delimiter, the string must be quoted. To include a quote in a quoted string, the quote must be preceded with a backslash.

Examples:

```
Id == CR 1
Id == "quote : \"
Name == "1/2 clk"
```

When using the attributes Id, Name, or Desc as qualifiers, you can use “*” to match Ø or more characters. To turn off the special meaning of “*”, precede the “*” with a backslash.

Additionally, you can ignore case and/or white space when performing string (Id, Name, Desc) searches. See the following information about Options in Global Explorer.

float:

Floats are required to have the following format:

<digits>.<digits>
 <digits>.<digits><exponent>
 <digits><exponent>

where <exponent> is defined to be “e” followed by an optional sign, followed by one or more digits.

section_string:	Header	
	Discharge	
	Jumpers	
	Continuities	
	Ignores	
	Merge_SC	
	Shorts	
	Opens	
	Switches_Pots	
	Inductors	
	Capacitors	
	Resistors	
	Miscellaneous	
	Rpacks	
	Diodes	
	Transistors	
	Zeners	
	Analog	
	DeltaScan	
	Frame_WaveScan_FS+	
	Brd_Power	
	Linear	
	Disables	
	Components	
	Trailer	
device_type:	APC	
	Adjustable_PS_A	
	Adj_PS_A	
	Adjustable_PS_B	
	Adj_PS_B	
	Analog_Template	
	APC	
	Beta-NPN	
	Beta-PNP	
	CapScan	
	Capacitor	

Continuity	
Data_Bus	
DB	
DeltaScan	
DigFuncProc	
DFP	
Diode	
Discharge	
Fixed_Slaved_PS_A	
Fix_PS_A	
Fixed_Slaved_PS_B	
Fix_PS_B	
FrameScan	
FS	
FrameScan+	
FS+	
Gray_Code	
GC	
Ground	
Header	
Ignore	
Inductor	
Jumper	
Linear_Template	
Opens	
Potentiometer	
Power_5.5V	
Power_5V	
Power_Bus	
PB	
Resistor	
Rheostat	
Rpack-DB	
Rpack-DI	
Rpack-DT	
Rpack-SB	
Rpack-SI	
Rpack-ST	
Shorts	
Trailer	
Transistor-NPN	
Transistor-PNP	
Unknown	

	Vector_Cluster	
	Vector_Image	
	Vector_Template	
	WaveScan	
	Zener	
test_type:	APC	
	Beta	
	Cap_Phase	
	Capacitor	
	Continuity	
	DeltaScan	
	DigFuncProc	
	DFP	
	Diode	
	Discharge	
	External_Program	
	Ext_Prog	
	FrameScan	
	FS	
	FrameScan+	
	FS+	
	Gray_Code	
	GC	
	Gray_Code_Disable	
	GC_Disable	
	IEEE	
	Ignores	
	Inductor	
	Jumper	
	LabWindows	
	No_Test	
	Opens	
	Potentiometer	
	Power_5V	
	Power_Prog_5.5V	
	Power_Prog_A	
	Power_Prog_B	
	Power_Prog_Slaved	
	Power	
	Prism_Capictor	
	Prism_Inductor	
	Prism_Resistor	

Prism_Test_I_Stim_V	
Prism__TISV	
Prism_Test_V	
Prism_TV	
Prism_Test_V_Stim_I	
Prism_TVSI	
Prism_Test_V_Stim_V	
Prism_TVSV	
Resistor	
Shorts	
Special_Case	
Test_I_Stim_V	
TISV	
Test_I_Stim_V_Stim_V	
TISVSV	
Test_V	
TV	
Test_V_Stim_I	
TVSI	
Test_V_Stim_I_Stim_V	
TVSISV	
Test_V_Stim_V	
TVSV	
Test_V_Stim_V_Stim_V	
TVSVSV	
Transistor	
Vector_Disable	
Vector	
WaveScan	
Zener	

Attribute-Value Pair Specifics

ALL_NODES **<op> <node_mod_list>**

Use this Attribute-Value only as an action. The ALL_NODES action helps you change all occurrences of node X to Y, no matter where the node is found (Component Properties data or Test Properties data) and no matter if the node is active or not. Allowed operators are: =. Note: an empty <node_mod_list> is not allowed.

Note that ALL_NODES affects the component nodes (as well as the Test Properties nodes) which affect every page; therefore, the qualifier attributes PAGE and PINS are ignored when used with ALL_NODES in the same agent.

Examples:

```
: All_nodes = (62 to 98)    // Changes node 62 to 98 in every test step of every section.
```

```
Section == Resistors : All_nodes = (62 to 98);
```

```
                // Changes node 62 to 98 in every test
                //step in the Resistor section.
```

AUTO_RANGE	<op> ON/OFF
BW_LIMIT	<op> ON/OFF
FAST_MODE	<op> ON/OFF
LINE_REJECT	<op> ON/OFF

Use these Attribute-Values as qualifiers or actions. As qualifiers, they allow you to select a test page by the AutoRange, BW Limit, Fast Mode or Line Reject fields. Allowed operators are: == or !=. When you use them as actions, you can change the AutoRange, BW Limit, Fast Mode or Line Reject fields of a test page. Allowed operators are: =.

AVERAGING	<op> int
SAMPLES	<op> int
WAIT	<op> int
SQUELCH	<op> int

You can use these Attribute-Values as qualifiers or actions. As a qualifiers, they allow you to select a test page by the averaging, samples, wait, or squelch fields. Allowed operators are: ==, !=, >, <, >= or <=. When you use them as actions, you can change the averaging, samples, wait or squelch fields of a test page. Allowed operators are: =, += or -=.

The WAIT attribute does not apply to the Wait 1 field in the Test Properties portion of the IEEE Step Worksheet.

COMP_NODES	<op> <node_list> <node_mod_list>
-------------------	-------------------------------------------------------------

You can use this Attribute-Value as a qualifier or action. As a qualifier, it allows you to select a test step by matching any node in the “Number of Pins” field using the <node_list> format. Allowed operators are: == or !=. As an action, it allows you to change any node in the “Number of Pins” field using the <node_mod_list> format. Allowed operators are: =. Note: an empty <node_list> or <node_mod_list> is illegal.

When you click the Number of Pins field in the 18xx interface, a pop-up displays the list of nodes for each pin. This is the topology information for the component. The following Agent would change node 5 to node 8 in the topology information everywhere.

```
: Mod Comp_nodes = (5 to 8);
```

COMP_VALUE	<op> <float>
COMP_VALUE1	<op> <float>
COMP_VALUE2	<op> <float>

You can use these Attribute-Values as qualifiers or as actions. When you use them as qualifiers, they allow you to select a test step by the Value of the component. Allowed operators are: ==, !=, >, <, >= or <=. When you use them as actions, you can change the Value of the component. Allowed operators are: =, += or -=.

Some tests, such as NPN and PNP tests, have two value fields: “Value 1” and “Value 2”. When this is the case, use COMP_VALUE1 or COMP_VALUE2 to select the specific value field.

We do not recommend that you set the fields to a value outside the test or device types range; for additional information see HIGH_VALUE.

DESC <op> <string>

You can use this Attribute-Value as a qualifier or an action. As a qualifier, it allows you to select a test step by matching the component Description Field. Allowed operators are: == or !=. When you use DESC as an action, you can change the contents of the description field. Allowed operators are: =.

DEVICE_TYPE <op> < device_type>

You can use this Attribute-Value only as a qualifier. It allows you to select test step by Component Device type. Allowed operators are: == or !=.

DIG_CLOCK_DIV_GC <op> <int>

You can use DIG_CLOCK_DIV_GC as a qualifier or action. As a qualifier, DIG_CLOCK_DIV_GC allows you select a Gray code test page by the Clock Divisor field. Allowed operators are: ==, !=, <=, >=, >, or <.

As an action, DIG_CLOCK_DIV_GC allows you to modify the Clock Divisor field of a Gray code test Page. Allowed operators are: =, +=, or -=.

DIG_CLOCK_DIV_VEC <op> <int>

You can use DIG_CLOCK_DIV_VEC as a qualifier or action. As a qualifier, DIG_CLOCK_DIV_VEC allows you select a vector test page by the Clock Divisor field. Allowed operators are: ==, !=, <=, >=, >, or <.

As an action, DIG_CLOCK_DIV_GC allows you to modify the Clock Divisor field of a vector test page. Allowed operators are: =, +=, or -=.

DIG_CLOCK_SOURCE <op> INTERNAL | EXTERNAL

Use these Attribute-Values as a qualifier or action. As a qualifier, DIG_CLOCK_SOURCE allows you to select a vector test page by the Clock Source field. Allowed operators are: == or !=.

As an action, DIG_CLOCK_SOURCE allows you to modify the Clock Source field of a vector test page. The allowed operator is: =.

DIG_HIGH_THRESH	<op> <float>
DIG_LOW_THRESH	<op> <float>
DIG_LOGIC_LEVEL	<op> 3V ...
DIG_TERMINATOR	<op> NONE ...

Use these Attribute-Values as qualifiers or actions. As qualifiers, the attributes allow you to select a vector or Gray code test page by the Low Threshold, High Threshold, Logic Level, or Terminator field. Allowed operators for DIG_HIGH_THRESH and DIG_LOW_THRESH are: ==, !=, <=, >=, <, or >. Allowed operators for DIG_LOGIC_LEVEL and DIG_TERMINATOR are: == or !=.

As actions, these attributes allow you to modify the Low Threshold, High Threshold, Logic Level, or Terminator fields of a Gray code or vector test page. Allowed operators for DIG_HIGH_THRESH and DIG_LOW_THRESH are: =, +=, -=. Allowed operators for DIG_LOGIC_LEVEL and DIG_TERMINATOR are: =.

DIG_MEAS_DELAY <op> <int>

Use this Attribute-Value as a qualifier or action. As a qualifier, DIG_MEAS_DELAY allows you to select a vector test page by the Measure Delay field. Allowed operators are: ==, !=, <=, >=, <, or >.

EXTENDED_MODE **<op> ON | OFF**

You can use this Attribute-Value as a qualifier or an action. As a qualifier, it allows you to select a test page by the extended mode. Since only Capacitors, Resistors, and Potentiometers have extended mode, the EXTENDED_MODE agent applies only to test pages with these device types. Allowed operators are == or !=. When you use it as an action, you can change the extended mode of a test page. The allowed operator is =.

GUARD_MODE **<op> PASSIVE | SEMI_ACTIVE | ACTIVE**

You can use this Attribute-Value as a qualifier or an action. As a qualifier, it allows you to select a test page by the guard mode. Allowed operators are: == or !=. When you use it as an action, you can change the guard mode of a test page. The allowed operator is =.

GUARD_NODES **<op> <node_list>**

See STIM_NODES.

HIGH_VALUE **<op> <test_val>****LOW_VALUE** **<op> <test_val>**

You can use these Attribute-Values as qualifiers or actions. As qualifiers, HIGH_VALUE and LOW_VALUE allow you to select the Test Properties page with the specified high and/or low value fields on the worksheet. Allowed operators are: ==, !=, >, <, >= or <=. When you use these as actions, you can change the high and/or low value field of a Test Properties page. Allowed operators are: =, += or -=.

The <test_val> operand has two formats: float or percentage. The float format is the actual value for the field. If you want the high value of a resistor test to equal 3.5 K ohms, the Agent would be High_value = 3.5e3, or High_value = 3500.0. If you use the percentage format, then the high/low value is calculated as a percentage of the Test Properties Value field.

Only Step Worksheets that have a Value field (such as Capacitors and Resistors) can use the percentage format.

For example, if the Test Properties value equals 10 ohms, then the agent, High_value = 5.0%, sets the high value field to 10.5 ohms.

The percentage must be written as a float followed by the percent sign (that is, 5.0% is the correct syntax; if entered as 5%, a compile error will result). Using += or -= with the percentage format can produce unexpected results. For example, the action, Low_value += 10.0 %, will not increase the low tolerance by 10, but will increase the low value by 10% of the value field. For example, if a resistor test has Value = 10 ohms and Low = 8 ohms, the low tolerance is 20%. Then the action, Low_value += 10.0% will increase the Low field to 9 ohms or 10% (not 30%).

Note that it is possible to set the field to a value outside the test or device types range when using the COMP_VALUE, COMP_VALUE1, COMP_VALUE2, TEST_VALUE, LOW_VALUE, or HIGH_VALUE attributes. Example: Resistor tests are scaled from ohms to Mohms. If you try to modify the LOW_VALUE of a resistor test to 0.0006 (that is, Low_value = 0.0006) the value displayed on the resistor Test Properties portion of the Step Worksheet will be rounded up to 0.001. However, the value stored is 0.0006! Similarly, the action, Low_value = 0.0004, will be truncated and the display will show 0.000. Since the real value (in the first case) is 0.0006, but the display shows 0.001, the qualifier Low_value ==0.001 will NOT select this test! To select this test not knowing what the real value is, we recommend that you to search for a range of values.

Example: The following agent will select this test and set the value to 0.001:

```
Low_value<    0.001,
Low_value>    0.0001
:modify
Low_value=    0.001;
```

HIGUARD <op> ON | OFF

See PRECISE.

ID <op> < string >

You can use this Attribute-Value as a qualifier or an action. As a qualifier, it allows you to select a test step by matching the Component ID field. Allowed operators are: == or !=. When you use it as an action, you can change the ID of a test step. Allowed operators are: =.

LOW_VALUE <op> <test_val>

See HIGH_VALUE.

MEAS_NODES <op> <node_list>

See STIM_NODES.

MEAS_PIN <op> int | pin_type

You can use this Attribute-Value as a qualifier or an action. As a qualifier, it allows you to determine if the device Pin Number is the active node data for the Measure and if this is true, which pin i is. You can use pin numbers (see STIM_PIN example for swapping poles), or you can use pin types such as Wiper, Lead1, Lead2, etc. Allowed operators are: =.

NAME <op> < string >

You can use this Attribute-Value as a qualifier or an action. When you use it as a qualifier, you can select the test step by matching the Component Name field. Allowed operators are: == or !=. When you use it as an action, you can change the NAME of a test step. Allowed operator is: =.

PAGE <op> <int>

You can use this Attribute-Value only as a qualifier. PAGE allows you to specify the page of the test steps to be modified. Allowed operators are: ==, !=, >, <, >= or <=.

Note that the PAGE and PINS qualifiers cannot be used with the ALL_NODES action. See ALL_NODES for more detail.

PINS <op>(pin_num [, pin_num,...] |
(pin_type [, pin_type, ...])

You can use the PINS Attribute-Value only as a qualifier. The PINS qualifier helps you select a specific page of a multipage test. You can use pin numbers or pin types. The following is a list of recognized pin types: Wiper, Lead1, Lead2, Anode, Cathode, B (Base), E (Emitter), and C (Collector). Allowed operators are == or !=.

Note that pin numbers and pin types cannot be mixed; there must be at least one pin in the list. (That is, Pins == () would result in a syntax error.)

The PAGE and PINS qualifiers cannot be used with the ALL_NODES action. See ALL_NODES for more detail.

PRECISE <op> ON | OFF

HIGUARD <op> ON | OFF

FAST_MODE <op> ON | OFF

You can use these Attribute-Values as qualifiers or actions. As qualifiers, they allow you to select a test page with the precise, higuard or fast mode field. Allowed operators are: == or !=. When you use them as actions, you can change the precise, higuard, or fast mode field of a test page. The allowed operator is =.

PRE_CYCLE_VACUUM <op> YES | NO

You can use this Attribute-Value as a qualifier or an action. As a qualifier, it allows you to test the Pre-Test Option Variable: Cycle Vacuum. Allowed operators are: == or !=.

As an action, PRE_CYCLE_VACUUM sets Cycle Vacuum to YES or NO. Be aware, that setting Cycle Vacuum to YES will cause the master variable: Repeat Page to be set to ON. Setting Cycle Vacuum to NO will have no effect on the master variable: Repeat Page. Refer to PRE_REPEAT_PAGE for additional details. Allowed operator is: =.

PRE_REPEAT_COUNT <op> <int>

You can use this Attribute-Value as a qualifier or an action. As a qualifier, it allows you to test the Pre-Test Option Variable: Repeat Count. Allowed operators are: ==, !=, <=, >=, < or >.

As an action, PRE_REPEAT_COUNT allows you to set the number of time to repeat the page. Be aware that Modifying Repeat Count will also cause the master variable: Repeat Page, to be set to ON. Resetting Repeat Count to zero (the default), will have no effect on the master variable: Repeat Page. Refer to PRE_REPEAT_PAGE for additional details. Allowed operators are: =, += or -=.

PRE_REPEAT_DELAY <op> <int>

You can use this Attribute-Value as a qualifier or an action. As a qualifier, it allows you to test the Pre-Test Option Variable, Repeat Delay. Allowed operators are: ==, != <=, >=, < or >.

As an action, PRE_REPEAT_DELAY allows you to set the number of milli-seconds Repeat Delay will be effective. Be aware that Modifying Repeat Delay will also cause the master variable: Repeat Page, to be set to on. Resetting Repeat Delay to zero (the default), will have no effect on the master variable: Repeat Page. Refer to PRE_REPEAT_PAGE for additional details. Allowed operators are: =, += or -=.

PRE_REPEAT_MODE <op> CONTINUOUS | WHILE FAILING

You can use this Attribute-Value as a qualifier or an action. As a qualifier, it allows you to test the Pre-Test Option Variable, Repeat Delay. Allowed operators are: == or !=.

As an action, PRE_REPEAT_MODE allows you to set Repeat Mode to CONTINUOUS or WHILE_FAILING. Be aware that setting Repeat Mode to while failing will also cause the master variable: Repeat Page, to be set to ON. Setting Repeat Mode to CONTINUOUS (the default), will have no effect on the master variable, Repeat Page. Refer to PRE_REPEAT_PAGE for additional details. Allowed operator is: =.

PRE_REPEAT_PAGE <op> ON | OFF

This Attribute-Value effects the Pre-Test Option variable, Repeat Page and its associated variables, Repeat Delay, Repeat Mode, Cycle Vacuum and Repeat Count. Repeat Page is the master variable. Repeat Delay, Repeat Mode, Cycle Vacuum and Repeat Count are the subordinate variables.

You can use this Attribute-Value as a qualifier to test if the master variable: Repeat Page, is ON or OFF. Allowed operators are == or !=.

As an action, turning PRE_REPEAT_PAGE off causes all the Repeat Page subordinate variables to be reset to their default values. Turning Repeat page on, simply sets the Repeat Page on. Allowed operator is: =.

SECTION <op> <section_string >

You can use this Attribute-Value only as a qualifier. SECTION allows you to specify the section (ex. Resistors, Capacitors ...) of the test steps to be modified. Allowed operators are: ==, !=, > , < ,

>= or <=.

SQUELCH <op> <int>

See AVERAGING.

STIM_NODES <op> <node_list>

MEAS_NODES <op> <node_list>

GUARD_NODES <op> <node_list>

You can use these Attribute-Values as qualifiers or actions. As qualifiers, they allow you to select a test page by the stimulus, measure or guard nodes used. Allowed operators are == or !=. When you use them as actions, you can change the stimulus, measure or guard nodes of a test page. Allowed operators are =, += or -=.

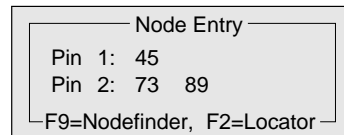
It is legal to have an empty node list. Thus the qualifier, Guard_nodes != (), selects a test page that has at least one guard. The action, Stim_nodes = (), sets the stimulus nodes for the test page to none.

IMPORTANT: The node data for STIM_NODES, MEAS_NODES and GUARD_NODES can be stored two places. If a check mark is on Device Pin Number field, then the nodes are really stored in the component field Number of Pins. Otherwise, the check mark is on the Node Numbers field, and the nodes are stored with the Test Properties data.

The field (Device Pin Number or Node Numbers) with the check mark is considered to contain the active nodes for the test. If the active nodes are in the Node Numbers field, then the agent: Stim_nodes == (45) : Mod Stim_nodes += (46,47); would simply add nodes 46 and 47 to the Node Numbers field.

However, a problem occurs when the active nodes are in the Device Pin Number field, (which really are stored in the component field, Number of Pins).

For example, if the Stimulus field in the Test Properties portion of the Step Worksheet displays Pin 1 (45), then the stimulus node (45) is stored in the Number of Pins field in the component data section of the test page. (That is, when you click the Number of Pins field, a 45 is displayed next to pin 1.)

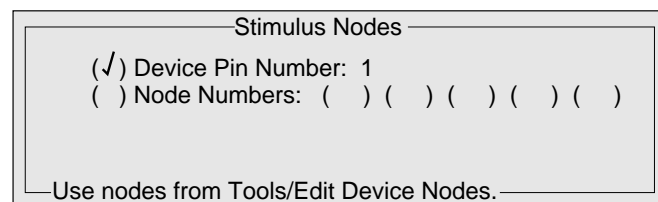


Node Entry

Pin 1: 45

Pin 2: 73 89

F9=Nodefinder, F2=Locator



Stimulus Nodes

☒ Device Pin Number: 1

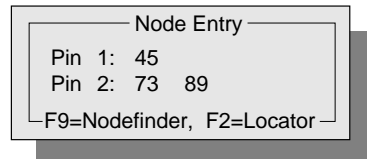
☐ Node Numbers: () () () () ()

Use nodes from Tools/Edit Device Nodes.

Thus the agent, Stim_nodes == (45) : Mod Stim_nodes += (46,47); would add nodes 46 and 47 to the component node data. (Now when you click the Number of Pins field, 45, 46 and 47 is

displayed next to pin 1.) This change would affect every page that uses Pin 1.

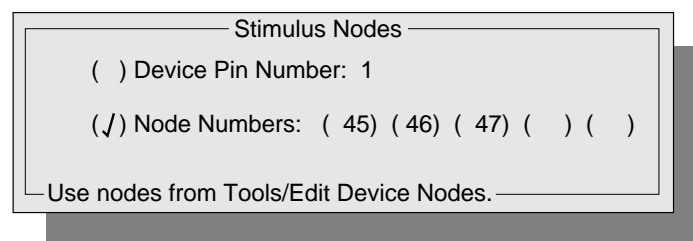
Since it is NOT acceptable to have a Test Properties action modify the Component Properties data, the Component Properties data will be copied to the Node Numbers field (overwriting any data that may already be there), and the Nodes Numbers field will contain the active node data. Thus, the agent would first copy the 45 to the Node Numbers field and add nodes 46 and 47.



Node Entry

Pin 1: 45
Pin 2: 73 89

F9=Nodefinder, F2=Locator



Stimulus Nodes

() Device Pin Number: 1

(√) Node Numbers: (45) (46) (47) () ()

Use nodes from Tools/Edit Device Nodes.

STIM_PIN

<op> <int> | pin_type

You can use this Attribute-Value as a qualifier or an action. As a qualifier, it allows you to determine if the “Device Pin Number” is the active node data for the Stimulus and, if this is true, which pin it is. You can use pin numbers (see Example below). Or you can use pin type such as Wiper, Lead1, Lead2, etc. Allowed operators are: == or !=.

Example: swapping poles

```
Id == name, //Only one of these agents will
Meas_pin == 1, //qualify and cause the poles to
Stim_pin == 2 //be swapped.
: modify
Meas_pin = 2,
Stim_pin = 1;
```

```
Id == name-mod,
Meas_pin == 2,
Stim_pin == 1
: modify
Meas_pin = 1,
Stim_pin = 2;
```

```
Id == name-mod, //Restores the Id field if the first
: modify //agent was the agent that
Id = name; //swapped the poles
```

TEST_NODES

<op> <node_list> | <node_mod_list>

You can use this Attribute-Value as a qualifier or an action. As a qualifier, it allows you to select a test page by matching the node(s) used in Test Properties (such as Stimulus, Measure, Guard, Reference or...). Thus the agent, Test_nodes == (7), would match any page that used node 7 in any Test Properties node field.

Note: the Stimulus, Measure, Guard, Reference... fields, all have two ways to store node information. When you click one of these fields, a pop-up displays Device Pin Number and Node Numbers fields.

Stimulus Nodes

(√) Device Pin Number: 1
 () Node Numbers: () () () () ()

Use nodes from Tools/Edit Device Nodes. _____

If Device Pin Number is selected, then the nodes are stored in the component pin data (which can be viewed from the Number of Pins field). If the Node Numbers field is selected, then the nodes are stored in the list immediately to the right of this field. Only one Device Pin Number, and the Node Numbers field will have a check mark next to it. These nodes are the active nodes. If the node 7 is in the active list, then the qualifier, Test_nodes == (7), qualifies the page. Allowed operators for Test_nodes qualifiers are: == or !=.

Be alert to the fact that when you use the “!=” operator in a Test_nodes qualifier, an unexpected number of multipage test pages would qualify.

For example, in the DEMO program, test step RPACK1 has the following pin/node data:

Pin 1: 84
 Pin 2: 66
 Pin 3: 35
 Pin 4: 34
 Pin 5: 67
 Page one uses the node data for pins 1 and 2.
 Page two uses the node data for pins 1 and 3.
 Page three uses the node data for pins 1 and 4.
 Page four uses the node data for pins 1 and 5.

The agent, Test_nodes != (35) : Mod Guard_nodes += 77; would add node 77 to the guards for pages one, three and four!

When you use TEST_NODES as an action, you can change any node in Test Properties by using the <node_mod_list> format. Allowed operators are: =.

Note that the Test_nodes action is the equivalent of using the Stim_nodes, Meas_nodes, and Guard_nodes actions all at the same time. Test_nodes just like Stim, Meas and Guard_nodes has a problem when the active nodes are really stored in the Component Properties data. See the note for STIM_NODES, MEAS_NODES, and GUARD_NODES.

Example: The following agent would change node 7 to node 11 in Test Properties of every test page that used node 7. Note that node 7 would be changed to 11 only in the active node data.

```
Test_nodes == (7) : Mod Test_nodes = (7 to 11);
```

An empty <node_list> or <node_mod_list> is not allowed.

TEST_TYPE <op> <test_type>

You can use this Attribute-Value only as a qualifier. As a qualifier, it allows you to select the test page by Test type. Allowed operators are: == or !=.

TEST_VALUE <op> <float>

You can use this Attribute-Value as a qualifier or an action. As a qualifier, it allows you to select the test page with the Value field in Test Properties. Allowed operators are: ==, !=, >, <, >= or <=. When you use it as an action, you can change the Value field of a test page. Allowed operators are: =, += or -=.

We do not recommend that you set the fields to a value outside the test or device types range; for additional information see HIGH_VALUE.

TOL <op> <tol_val>

TOL1 <op> <tol_val>

TOL2 <op> <tol_val>

You can use these Attribute-Values as a qualifier or an action. As qualifiers, they allow you to select a test step by the Tolerance of the component. Allowed operators are: ==, !=, >, <, >= or <=. As actions, they allow you to change the Tolerance of the component. Allowed operators are: =, += or -=.

Some tests, such as Capacitor tests, have two value fields: “Tol 1” and “Tol 2”. When this is the case, then TOL1 or TOL2 must be used to select the specific tolerance field.

Note that TOL, TOL1 and TOL2 change the tolerance value only in the component data. Do not use them to try to modify the test values in Test Properties.

WAIT <op> <int>

See AVERAGING.

WIRE <op> 3 | 4 | 5 | 6 | NONE

This Attribute-Value can be used as a qualifier or action. As a qualifier, it allows the User to select a test page by the wire mode. Allowed operators are: ==, !=, >, <, >= or <=. When you use it as an action, you can change the wire mode of a test page. The allowed operator is =.

Examples of Global Editor Format

```
/* This increases the symmetrical limits of component POT2 by 5 % */
Id ==      POT2
: modify
High_value+=5%,
Low_value-=5%;

/* This adds guard nodes 33 and 44 to component R1 */
Id ==      R1
: modify
Guard_nodes+=( 33, 44 );

/* This turns fast mode on for all components in the resistor section. */
Section==  Resistors
: modify
Fast_mode= On;

/* For all resistor tests in sections: Capacitors to Resistors,
*turn Extended mode on.
*/
Section<=  Capacitors,
```



```

Section>= Resistors,
Test_type==Resistor
:modify
Extended_mode = on;

```

Using Global Editor

There are three ways you can use Global Editor.

- Create a new Agent Input file and run it from the Editor (create/execute new file) window.
- Run an existing Agent Input file from the Execute existing edit file window.
- Run an existing Agent Input file using the Command Line function 18xx /G.

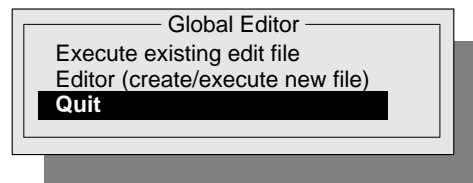
Navigating the Global Editor window: The instructions for creating new Agent Input files assume you are using a mouse. If your computer does not have a mouse, you can create the files using your keyboard and Global Editor's Window feature.

- Type W to select Window from the menu bar.
- Use the up or down arrow keys to choose from the pop-up menu.
- Use the up or down arrow keys to navigate within the Create Attribute-Values window.
Use the plus key (+) on the keyboard's number pad to add a Qualifier or an Action.
- Use the space bar to select a modifier (Modify, Comp_Enable, Comp_Disable, or Comp_Delete).
- In the Constructed Agent Display section of the window, use the plus key (+) on the keyboard's number pad to select a line.
- Use F10 to return to the menu bar.

Create a new Agent Input file. The Editor (create/execute new file) feature has pop-up menus and shortcuts to help you create a new Agent Input file. This feature is useful if you are unfamiliar with Attribute-Value syntax.

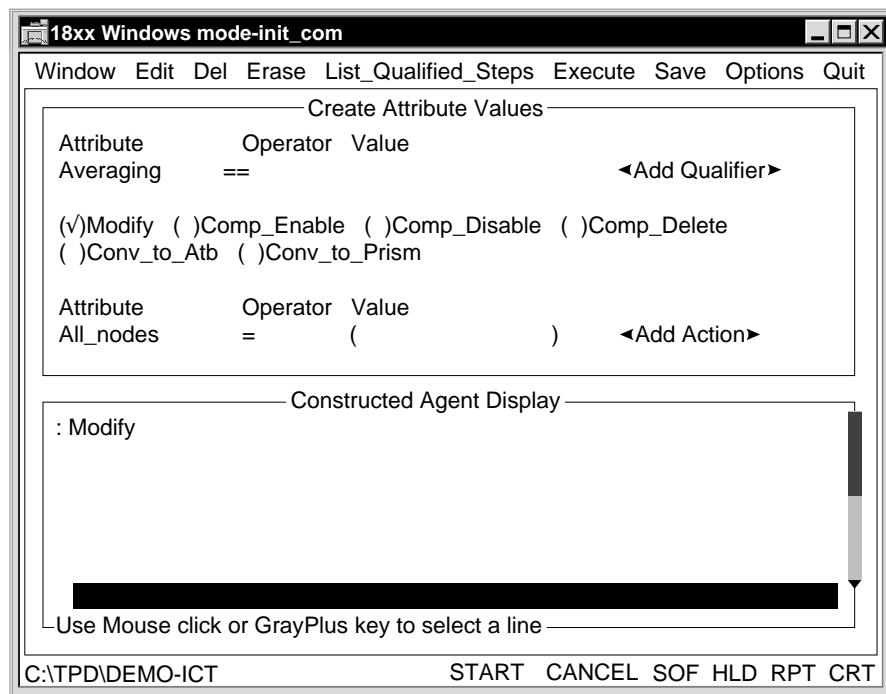
- 1 Select Utility from the Main menu.
- 2 Select Global Editor from the Utility menu.

The Global Editor window appears.



- 3 Select Editor (create/execute new file) from the Global Editor menu.

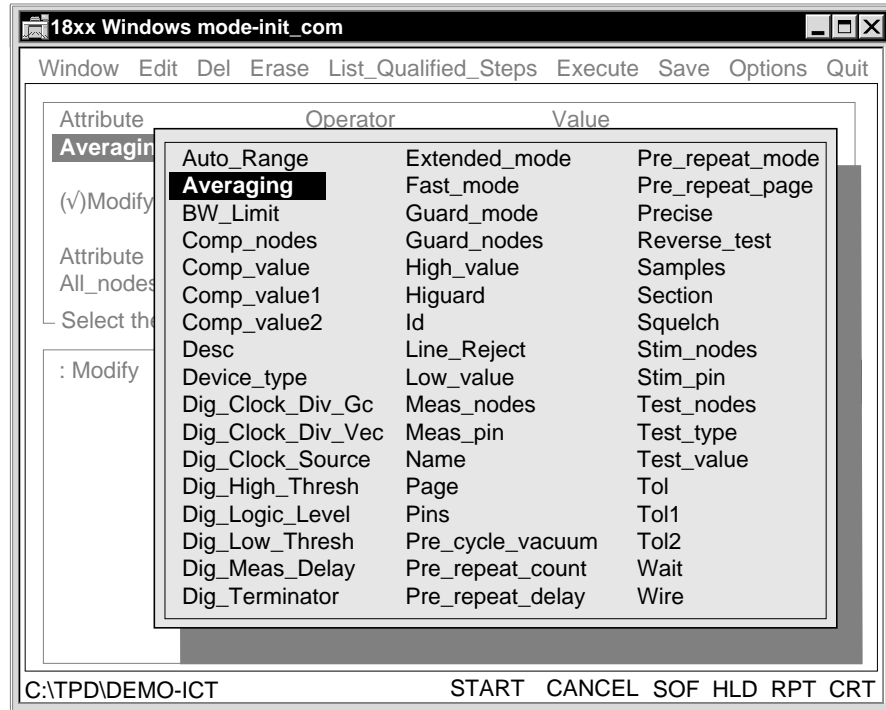
The Create Attribute Values window appears.



The upper section of the Agent Editor Window is used to create Qualifier and Action Attribute-Values.

- 1 Click the Qualifier Attribute field.

A pop-up appears listing the available Qualifier Attributes.



Select the appropriate attribute.

2 Click the Qualifier Operator field.

A pop-up appears listing the Operators available for the chosen Attribute. Select the appropriate Operator.

3 Enter a specific Value based on the chosen Attribute.

If you choose Section, Device_type, or Test_type as the Attribute, you can use certain shortcuts to enter the value:

Type the first two or three letters of the Value string, press enter, and the full name of the value string is filled in. Value strings that are similar, that is, Rpack-SI or Rpack-SB, require you to type additional letters to distinguish the specific value string.

Certain value strings have allowed acronyms that you can type. (Note that when using an allowed acronym, you can also “shortcut” by typing the first two or three letters).

4 Click unit (not available for all Attributes).

The Unit option is displayed when you choose an Attribute that has an integer or float value. A pop-up appears allowing you to choose the unit or multiplier.

5 Click the Add Qualifier button.

The information you entered is displayed in the Constructed Agent Display area of the window.

6 Click an Agent Type.

Modify The only agent type that allows Action Attribute-Value(s).

Comp_Enable Temporarily enables the selected test step. Does not allow Action Attribute-Value(s).

Comp_Disable Temporarily disables the selected test step. Does not allow Action Attribute-Value(s).

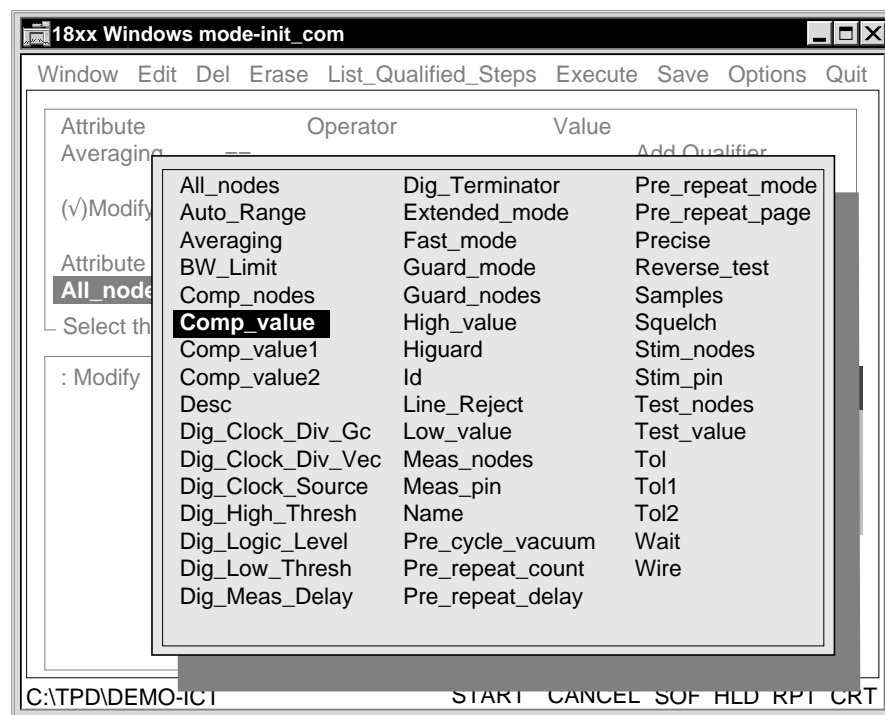
Comp_Delete	Permanently removes the test step from the ICT.TST program file.
Conv_to_ATB	Converts from tests executed using PRISM to tests executed using ATB.
Conv_to_PRISM	Converts from tests executed using ATB to tests executed using PRISM.

If you choose the Agent Type “Modify,” continue to step 1 below to create an Action. You create an Action agent very much the same way you created the Qualifier agent.

If you choose Comp_Enable, _Disable, or _Delete, proceed to step 5 below which describes how to modify the agents you constructed.

- 1 Click the Action Attribute field.

A pop-up appears listing the available Action Attributes.



- 2 Click the Action Operator field.

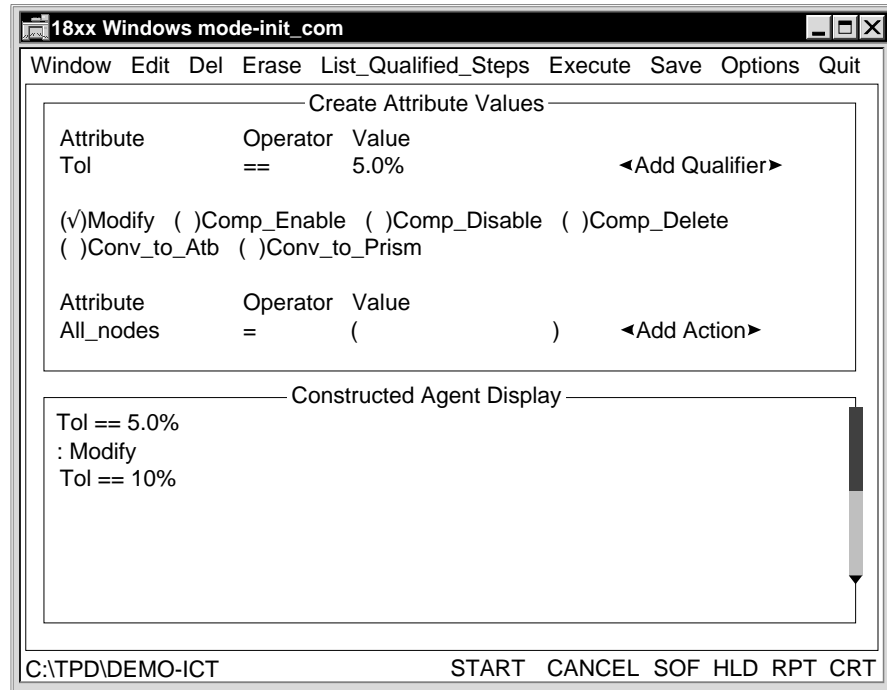
A pop-up appears listing the Operators available for the chosen Attribute. Select the appropriate Operator.

- 3 Enter a specific Value based on the chosen Attribute.

Refer to comments about Values and unit in the Qualifier steps above. After you construct the Action Attribute-Value

- 4 Click the Add Action button.

The information you entered is displayed in the lower section of the window, the Constructed Agent Display area.

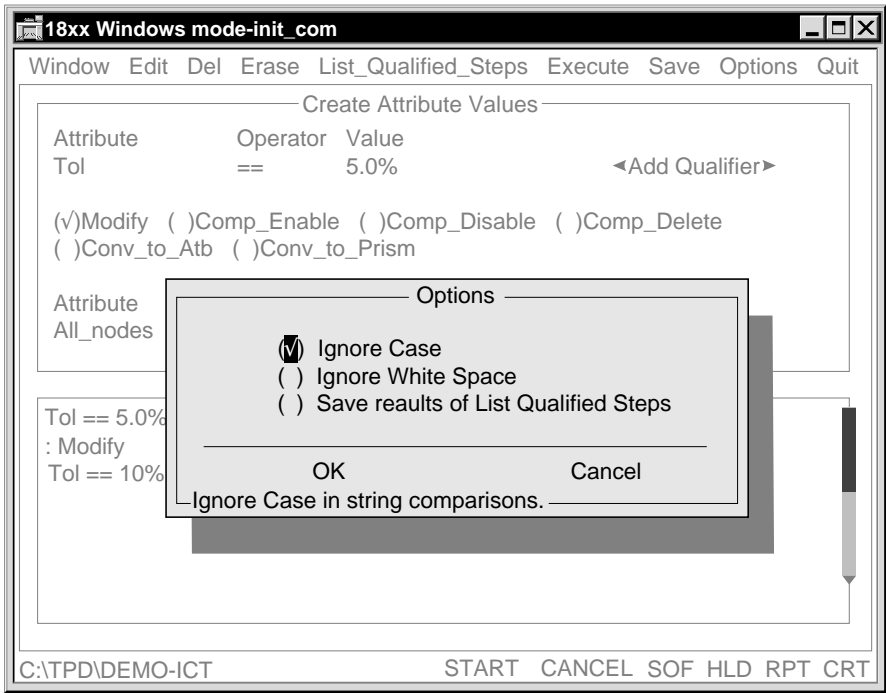


You can add up to 20 lines of Qualifier, and if appropriate, Action agents in the Constructed Agent Display area by following the previous steps.

Using the Options Menu

The Options Menu facilitates the search for Id, Name, and Desc attributes which accept strings as their values. The Ignore Case, and Ignore White Space options, used as qualifiers, allows you to ignore case and/or white space in searches on Id, Name, or Desc fields.

In addition, The “Save results of List Qualified Steps” turns on or off a pop-up allowing you to save the results to a file window after each execution of List Qualified Steps.

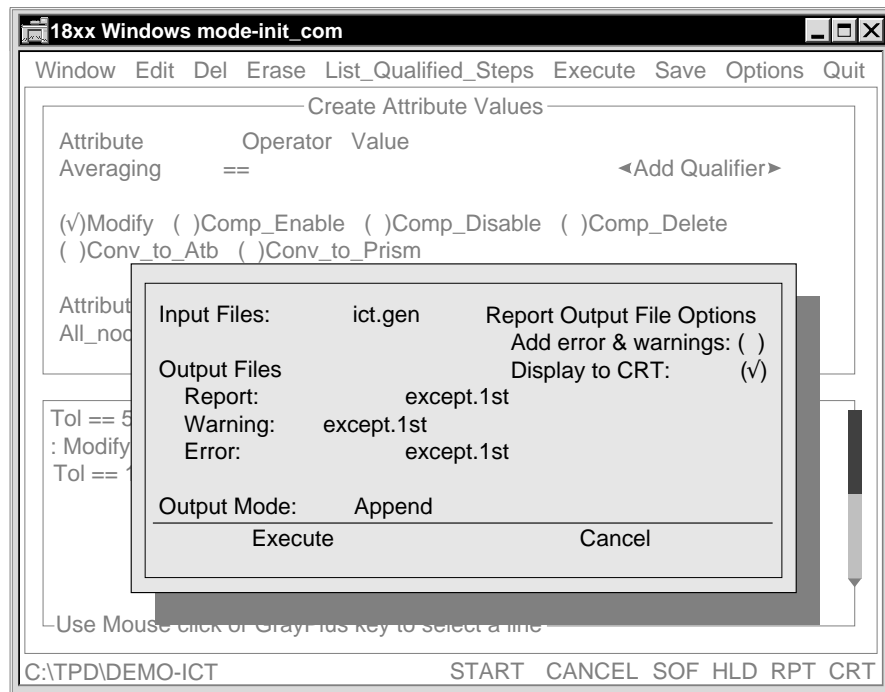


- 5 Modify the constructed Qualifier and/or Action agents.
Use the menu bar functions as listed in the table below to modify the displayed agents.

Function	Action
Window	Create files using your keyboard (for users who do not use a mouse).
Edit	Copies selected Attribute-Value to upper section of window where it can be modified. After modification, the Add button returns the Attribute-Value to the CAD.
Del	Deletes selected Attribute-Value
Erase	Clears all Attribute-Values except Modification type from the CAD.
List_Qualified_Steps	Displays a list of test steps that meet all qualifier specification.
Execute	Runs the test based on the constructed agents. The Report file is saved to the default EXCEPT.LST or any valid specified DOS file. See Execute window below.
Save	Appends or Overwrites a specified file with the agents listed in the Constructed Agent Display area.
Options	Allows ignoring of case and/or white space in string comparisons. Also enables Save after running list of qualified steps.
Quit	Exits Global Editor without running test(s) and returns to Main Menu.

- 6 Click Execute in the Menu Bar to run the Agent Input file.

The Execute window appears.



Refer to the information in the following table about modifying fields in the Execute window.

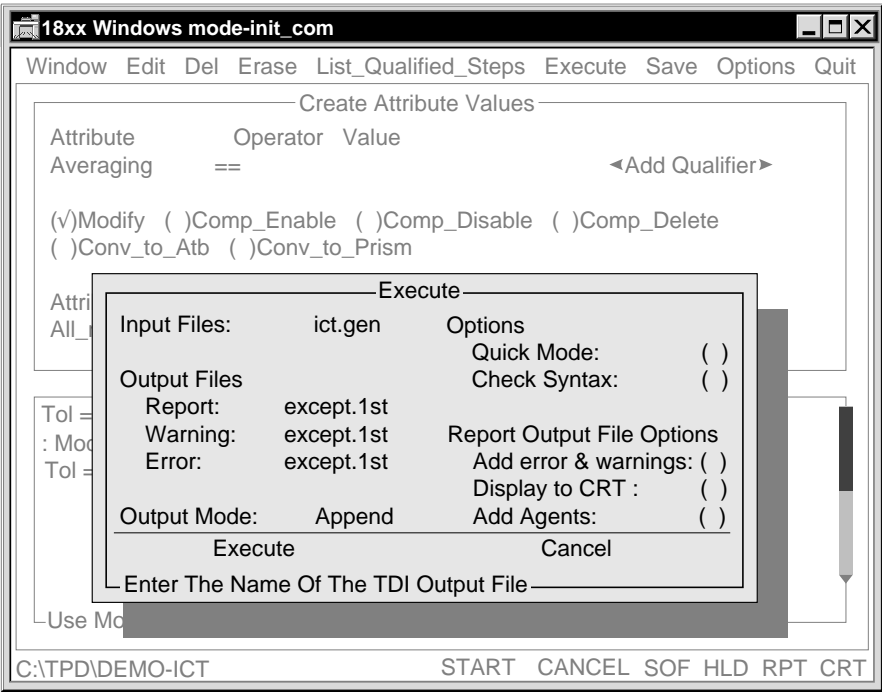
Field	Description
Output Files	Default is EXCEPT.LST or enter name(s) of any file that is in the current board directory.
Report	Contains listing of the modified test steps.
Warning	Contains possible warnings, including messages regarding any test steps that qualified but were not modified. Saving Warnings to a separate file may be helpful.
Error	Contains possible errors that occurred during execution.
Output Mode	Append or Overwrite Output files.
Report Output File Options	Add error and warnings (default is Off). Display to CRT (default is On).
Execute	Runs constructed agent input file.
Cancel	Exits Execute window without running the agent and returns to the Global Editor window.

- 7 Click Execute to run the agent input file.
- Click Cancel if you decide not to run the file at this time.

Run an Existing Agent Input File—Execute. To run an agent input file that already exists, use the Execute an existing edit file feature.

- 1 Select Utility from the Main menu.
- 2 Select Global Editor from the Utility menu.

- 3 Select Execute an existing edit file from the Global Editor menu.
The Execute window appears.



The following table explains the function and purpose of the Execute window fields.

Field	Description
Input File	Enter name of file containing the agent to execute. File must be in current directory.
Output Files	Default is EXCEPT.LST or enter name(s) of any file that is in the current board directory.
Report	Contains listing of the modified test steps.
Warning	Contains possible warnings, including messages regarding any test steps that qualified but were not modified. Saving Warnings to a separate file may be helpful.
Error	Contains possible errors that occurred during execution.
Output Mode	Append or Overwrite Output files.
Options	Click in parentheses to enable.
Quick Mode	Useful if input file contains more than one agent.
Check Syntax	Checks input file for syntax errors.
Report Output File Options	Add error and warnings (default is Off). Display to CRT (default is On). Add agents (default is Off).
List Qualified Steps	Lists steps that can be modified by one or more agents in the input file.
Execute	Runs constructed agent input file.
Cancel	Exits Execute window without running test(s) and returns to the Main Menu.

- 4 Click Execute to run the agent input file.

Click Cancel if you decide not to run the file at this time.

Run an Existing Agent Input File—Command Line. You can also run an existing agent input file using 18xx command line arguments. The command must be called from your current board directory within the 18xx test program directory (TPD).

At the prompt type

```
18xx /G.
```

Global Editor runs the Agent Input file, lists the steps modified, and then returns to the prompt.

Refer to Global Editor Command Line Interface for additional information.

Output File Management. If you execute an agent that does not have qualifiers, all test steps will qualify. This will result in warning messages similar to the following:

```
Step NNN Qualified, but no changes resulted!
```

To isolate the list of modified test steps from warning messages, send the warning output to a different file. To prevent warning and error messages from being included in the Report file, make sure the “Add error and warnings” option is not selected.

Finding Test Steps that use a Specific Node. The All_Node attribute is allowed only as an action; to list all test steps that have a specific node, use the agent

```
:modify
All_nodes = (NNNN to NNNN)
```

(where NNNN = specific node)

Send the report to a separate file and deselect the “Add error and warnings” option.

Searching Multiple Board Directories

You can use Global Editor to search for boards that share a common feature.

For example, if you want to know which programs use the DFP, you can access Global Editor from the Utilities menu. Create the following agent and save it to the file SEARCH.OUT in the directory C:\TDI:

```
Device_type      == DigFuncProc
:modify
All_nodes        = (1 to 1); //Dummy action
```

Create a batch file that changes to each board directory and calls p18xx /G to invoke Global Editor.

Example—Global Editor Batch File

```
set IFILES=-i c:\tdi\search.ged
set OFILES=-o c:\tdi\search.out -e c:\tdi\junk -w c:\tdi\junk
set ARGS=-m append -1
```

```
cd c:\tdp\demo
echo. > c:\tdi\search.out
echo.>> c:\tdi\search.out
echo c:\tdp\demo >> c:\tdi\search.out
18xx /G %IFILES% %OFILES% %ARGS%
```

```
cd c:\tdp\28f020
echo. > c:\tdi\search.out
```

```

echo.>> c:\tdi\search.out
echo c:\tpd\28f020 >> c:\tdi\search.out
18xx /G %IFILES% %OFILES% %ARGS%

set IFILES=
set OFILES=
set ARGS=

```

Global Editor Command Line Interface

The Global Editor module can be accessed through 18xx system software using command line arguments. You invoke 18xx by using the /G switch which instructs 18xx to run the Global Editor module and then exit. Following the /G switch are the arguments for Global Editor.

The following is a description of Global Editor command line usage and arguments. Note that when you use Global Editor embedded in 18xx, the command must be called from the target 18xx test program directory, just as Pgen is currently used. You can also enter the complete path of the input and output files with the -i -o -e and -w switches.

Global Editor Command line:

```
18xx /G [[-i | -o | -e | -w] file name] [-h] [-?] [-l] [-p] [-q] [-a]
```

Global Editor options:

-i <file name>	Specifies the input file name. The default input file name is ICT.GED.
-o <file name>	Specifies the report output file name. The default is EXCEPT.LST.
-e <file name>	Specifies the error output file name. The default is EXCEPT.LST.
-w <file name>	Specifies the warning output file name. The default is EXCEPT.LST.
-d	Turns Off the CRT display of the report activity. The default is On.
-h or -?	Displays Help message. Outputs a synopsis of Global Editor usage and options.
-l	Lists qualified test steps. Causes Global Editor to ignore any actions and lists the test steps that can be modified by the agents in the input file. Note that a test step will be listed more than once if more than one agent can modify it. Note that even though a test step qualifies, it does not mean the associated actions will actually change the step. The actions may not apply to the step or they may produce errors.
-m <mode>	Turns On Output Mode, where mode equals Append or Overwrite. The default is Append.
-p	Parses the input file and exits. Checks that the input file is syntactically correct.
-q	Turns On Quick Mode. The default is Off. When On, each section's agent list is scanned,

and all agents that modify test steps in that section are processed. (This saves time since the information for each section is read only once.) When Off, the agent list is processed in order. (This is a longer process since each section must be opened and scanned to see if the agent modifies any test steps. This is also repeated for each agent.)

- r Reports all activity to the report output file
Reports all test steps that qualified, whether or not they were modified. Note that -r is needed only when the -e file name or -w file name options are used.
- a Turns on Verbose.
Echo the input file to the report output file.

Hardware Configuration

The Hardware Configuration function scans the test head cage for the location of different types of system hardware, driver/receiver cards, and the functional interface board (FIB).

IMPORTANT: Do not scan cages with the fixture and board on the fixture receiver.

The driver/receiver cards have a variety of capabilities including basic analog, 5 volt only, and 3 to 5 volt. The scan ensures that the components on your board are noded properly by issuing warning messages when a current configuration fails to match an expected configuration, which you have specified.

The Hardware Configuration function

- Stores/displays the actual system hardware and DR configuration of the tester
- Creates the “expected” configuration file
- Displays the expected configuration/actual configuration screen

The scan is triggered automatically whenever tester power is cycled or the PC is rebooted (that is, the pcio TSR is removed). You can also run a scan by selecting Hardware Configuration from the Utility menu. When the software detects an incompatibility between the actual and the expected configuration files, it displays a warning message.

The actual tester configuration is used by many functions within the 18xx operating system, a few of these are:

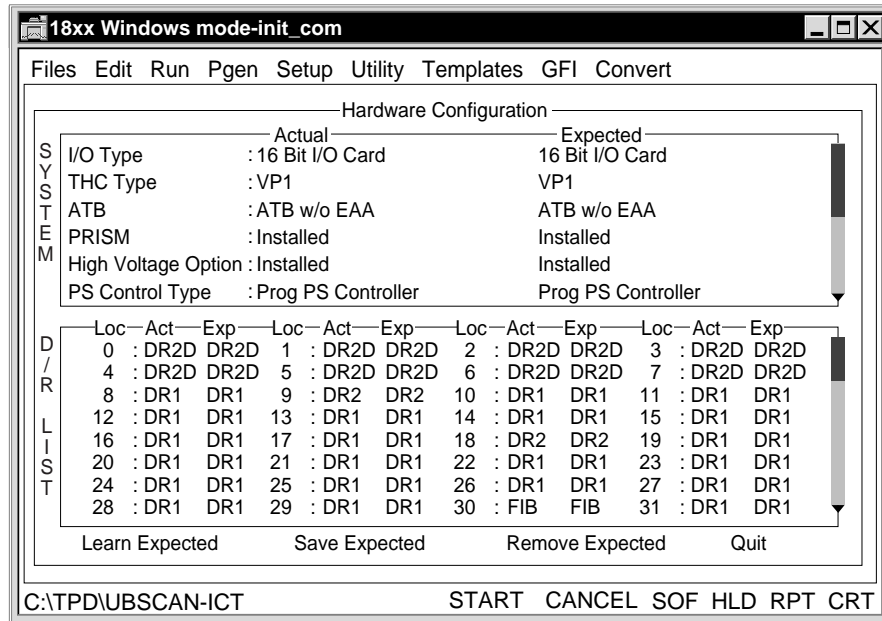
- Any addressing of a node for which no driver/receiver card exists generates a warning. (This warning can be suppressed with the -a flag on the PCIO command line.)
- When any node is to be used in a digital context, and the particular driver/receiver slot contains a DR2A, analog-only card, a warning message is generated. The software scans all digital sections of a program, including called subprograms, before the program executes. To disable this scan, select Analog DR Check/Off from the Setup/Environment menu.
- Just before power is about to be applied to the board-under-test during program execution, the configuration of the tester in the configuration file is compared to the requirements in the node3v file. If any discrepancies are detected, the test is stopped, and a message is displayed indicating any board slots which require a DR2 board but contain a DR1 board.

Setting Up and Using the Hardware Scan

After the initial software installation, set up the hardware configuration scan as directed below. You must have “Setup” permission to perform this procedure.

- 1 Select Hardware Configuration from the Utility menu.

The Hardware Configuration window appears listing the actual and expected hardware in the cage, and the location (Loc), actual configuration (Act), and the expected configuration (Exp) of the DR cards in the cage. When the window first appears, the Exp columns are empty.



2 Click the Learn Expected button.

This selection learns the current (actual) test head cage configuration and displays it in the System and DR List expected columns.

Use the scroll bar in the upper System window to view the status of all the hardware scanned: I/O Type, THC Type, ATB, Prism, High Volt Option, PS Control Type, DeltaScan, and WaveScan

3 Click the Save Expected button.

Once you have learned the expected configuration, the Save Expected button allows you to save it to the CAGE_SCN.DAT file. The contents of this file are compared to the actual configuration on each subsequent hardware scan.

In future scans, where the actual hardware and expected hardware type do not match, the actual type is highlighted. If there is no expected hardware configuration file available, the expected column is empty.

4 When you have finished your operations in the Hardware Configuration window, click Quit.

When the hardware is started up, (the pcio is installed or system power is turned off and back on), the hardware is re-scanned. If the results of this scan differ from the contents of the scan file, the following warning message appears:

"Cage Configuration file vs. Installed Hardware mismatch detected. Run Utility-Hardware configuration."

To remove an expected configuration file, click the Remove Expected button.

Scanning for DR2 3-Volt Cards

The Hardware Configuration function as it relates to 3 volt test is a precaution to help you ensure that your components have been noded correctly to prevent application of 5 V to 3 V devices.

When the Hardware Configuration function is selected, the test head cage is scanned. This information ensures that the tester does not apply 5 volts to logic parts that accept only 3 volts. DR1 boards are capable of delivering only 5 volts to the device under test, whereas DR2 boards can deliver 3 volts. Any nodes associated with 3 volt logic parts must be supported by a DR2 board.

The 3-volt requirements for the board-under-test are stored in a file called NODE3V in the board directory for the board-under-test. This information is obtained from the template database when the program is generated, or from information added by the user by editing the file by hand.

WARNING!

The system software does not prevent you from using the editor to put a 5 V node on a 3 V device and executing a test. Be sure that you do not remove nodes from the NODE3V file that are attached to 3 V logic devices.

Scanning With an FIB Installed in Slot 0

If you put an FIB in slot 0 and will be using digital test on your board, you need to disable Runtime Analog Check. When Runtime Analog Check finds an FIB in slot 0, it is assumed that you have an analog-only system, and digital functionality is not available.

To disable Runtime Analog Check, select **Setup** in the Main menu. Then select **Environment Variables**. Select **Off**.

Validate

The Validate function attempts to improve shorthand and MultiScan tests without the intervention of a programmer. It is based on a complex set of rules and restrictions. Its primary input is the component database, which supplies the process with the board's topology information. The secondary input includes the in-circuit program (ICT.TST) and interconnect section. Although finding guard points is a major part of the task, Validate also experiments with pole swapping, wait times, and digital averaging. It also allows you to accept as the nominal component value, a value which is derived from the test results (Automatic Accept).

You can invoke Validate from three different places in the menu.

- Global Validate called from the Pgen menu validates an entire program.
- Section Validate from the Component Select menu executes Validate on the current section.
- Step Validate from the Step Worksheet Options menu validates an individual step only.

Validate functions only for shorthand resistor and capacitor tests and MultiScan tests. You can disable Validate on a page-by-page basis by modifying the Validate flag in the Options/Test Step Controls menu for that page. Such disabling may be desirable for very specific manually created tests, where Validate should not modify anything.

As mentioned above, to find guard points, the Validate process relies heavily on information from the component database and passing interconnect section. Because it is important that the component database reflect the board as accurately as possible, it is generally a good idea to run Update from the Pgen menu before using Validate.

See chapter 2, "System Setup," or the **MultiScan User's Guide** for additional information about MultiScan validation.

Validate Process

In general, the Validate process consists of testing known-good components on a circuit board, using various combinations of stimulus, measure, and guard configurations, wire modes, and wait times, and building a test around the configuration that yields the test result which is closest to the nominal component value.

Validate attempts to find a good test for a component by testing to a narrower acceptance band than the component tolerance would dictate. The width of this acceptance band is selectable by the user in the Setup/Validate menu. If a component passes using the acceptance band, it is sure to pass when using the normal component tolerances. The narrower the acceptance band, the more processing that is done by Validate to find a good test configuration.

Validate begins by making an initial measurement of the component using all the parameters as specified in Test Properties. The test limits of the component are those dictated by the narrower user-selected acceptance band rather than the nominal component tolerance. If the component passes this test, no further refinement is necessary, and processing for this component is complete.

If this first test doesn't pass, Validate tests the component with the stimulus and measure nodes in the configuration specified in the component database, and again with these nodes swapped. This process gives two baseline test results. If either of these results is within the acceptance band for the component, the configuration of the better result is saved so that it can be used in the test program. If neither result is within the acceptance band, the next step is to try guarding the component to attain better test results.

Guarding is a means of electrically isolating a component from the effects of its neighbors so that the value of the component can be determined. In the case of resistors, parallel components makes the value of the tested resistor appear to be lower than it actually is, while in the case of a capacitor, parallel components makes its value appear higher. If a guard node is tried and found to make the measured component value closer to the expected value, the guard node is kept. If a guard node has no effect, or if it makes the test result worse, it is rejected and not included in further tests. Guard nodes that do help are moved to the top of the guard node list, while guard nodes that don't are immediately rejected.

Validate makes some intelligent decisions about where to try guarding. Guard nodes must be electrically close to the component under test. Nodes that are on the other side of power or ground buses from the component under test are not be included. To do so, would result in an extensive list of nodes which would have little effect on test results. Once a list of potential guard nodes is established, tests are made using these nodes one at a time to see if they improve test results or not.

Validate gets its input from three sources. One source is the in-circuit test program (ICT.TST). A second is the component database, which is a software representation of the board interconnections and component values. The third source is the board itself. Validate makes tests on the board to see if a particular test configuration improves test results or not. Therefore, it is extremely important that the component database be 100% accurate and that all the components on the board-under-test are known to be good. It is also important that Validate have information for low impedance components. This information can be obtained by running the Learn section of the Pgen menu or by including the nodes for low impedance components in the SC portion of the input list file IPL.DAT. Basically, you must have a passing Interconnects section.

Several measurements are taken for each guard attempt, and the stability of these measurements is taken into account when determining whether a particular guard configuration helps or not. Guarding is tried for both the straight and the swapped stimulus and measure node configurations. A separate list of guard nodes is kept for each configuration. At any time, if the guarding configuration brings the test results within the acceptance band, Validate accepts this as the final configuration and stops searching for more guard nodes.

Validate then makes two final measurements. The first is with the stimulus and measure nodes in the straight configuration with the straight guard set. The second is with the stimulus and measure nodes swapped, and with the swapped guard set. The better of the two results is chosen.

During the last measurement, extra processing may be done based on the Validate options selected in the Setup menu. These options are as follows:

Calculate Wait Times—As part of the final measurement, the time needed to produce stable test results is determined. If the Calculate Wait Times option is selected, this time is saved and used in the test program for this component.

Average Samples—If you select this option, different numbers of tests are tried and their results are averaged. The number that gives the most stable result is selected and put into the test program.

Automatic Accept—If you have selected Automatic Accept and if guarding and pole swapping don't yield test results that are within the acceptance band, Validate moves the component acceptance limits to values that cause the test to pass. This may be necessary for components with parallel paths that can't be guarded.

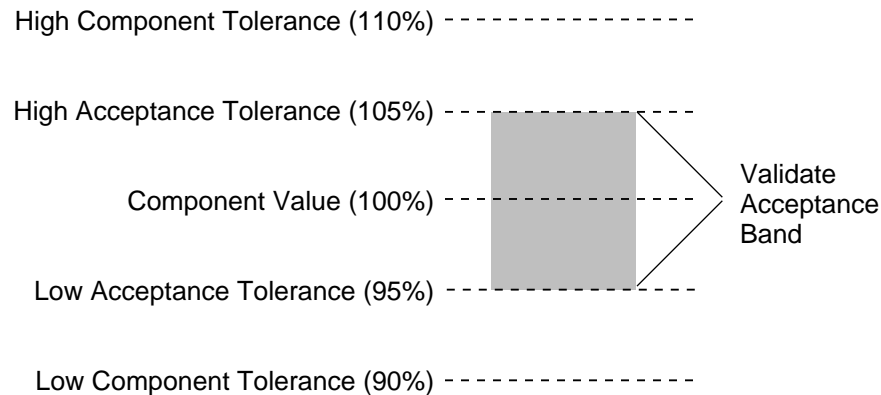
Automatic Exclude—If you select this option and Validate makes any modification to the test, this test is marked as having gone through the Validate process so that further Validation does not take place.

Establishing Guards

Acceptance Band

The acceptance band used for these tests is a combination of the component value tolerance and a second value called the acceptance range. This acceptance range is a percentage selectable by the user in the Setup/Validate environment. It is an amount by which the component tolerance is tightened for the purpose of the Validate tests. The acceptance range can have any value between 10% and 100%. For example, if the component that is being tested has a tolerance of 10%, and the user has selected a value of 50% for the acceptance range, the test limits will be 50% of 10%, or in other words, 5%. The acceptance range can be thought of as a percentage of a percentage.

The following diagram illustrates the Validate acceptance band.



In the initial phase, two measurements are taken as base values for future readings in the absence of guard points. One reading is taken with the stimulus and measurement poles set as specified Test Properties, the other is taken with the stimulus and measure poles swapped. Should either of the results fall within the acceptance band, no guard points will be needed, and the process will proceed to find wait times and averaging constants if you have specifically selected those

functions. The stim-meas pole setting which gives the better result is kept. Should the results of the initial pass fall outside the acceptance band, Validate tries guarding the component to achieve better test results.

Factors Prompting Guarding

With resistors, any components that are in parallel with the resistor under test will lower the measured value. The basic assumption is made that if the value measured in a test is below the low acceptance tolerance, there are components that need to be guarded to eliminate their influence in the circuit.

With capacitors, parallel components raise the measured value of the capacitor being measured. Therefore, if the measured value of a capacitor is above the high goal tolerance, we attempt to find guards that will bring the measured value within the acceptance band.

The purpose of guarding is to eliminate other components from affecting the measured result of the component being tested. If these external effects are eliminated completely, the test result should be exactly the value of the tested component. Guarding should never produce a result for a high value resistor that is above its real value, or a value for a capacitor that is below its real value. If guarding attempts produce results like these, it is assumed that noise or other external effects are producing a false indication. In this case, the guard that produced the false readings is rejected.

However, low value resistors (resistors smaller than 150 ohms) may in fact measure above the goal tolerance because of system resistance. In these cases, Validate will try different wire modes if possible, to see if the result is within specifications. Validate will first see if there are enough nodes to do a 6-wire test. If not, 5-wire and 4-wire modes are checked. If any of these multiwire modes are possible, they are used to give better test results.

Establishing Potential Guard Points

The potential guard list should be a list of nodes that are electrically close to the component under test. These nodes are the ones that will have the most influence over test results.

Care is taken to exclude nodes from low impedance components such as jumpers, inductors, etc., as potential guards. Since guarding a node places it at a ground potential, guarding nodes that are too close to the stimulus pin, may reduce the stimulus voltage getting to the component under test, thus producing erroneous test results.

It is important that the information for low impedance traces is known before Validate is run. The low impedances must be made known to the test program either by running the Learn section of the Pgen menu, or by including these components by hand in the IPL.DAT input list file.

In building the potential guard node list, any nodes that you may have already suggested as guard nodes in Test Properties are put into the list of potential guard nodes first. These will be the first nodes tried in the testing phase to see how they affect test results. As part of this routine, Validate ensures that any nodes you have suggested are not connected to the component under test. If they are, they are not included in the potential guard list, and Validate will remove them from Test Properties.

Next, the test program file is searched, and lists of all nodes in common with the stimulus node and all nodes in common with the measurement node are made. In common means continuities or low impedance components. The ICT.TST file contains all the test information for the board-under-test. In this file is a section called the SC_Merge section which is a collection of all the continuities, jumpers, special cases, and low impedance components. From this section, all the nodes of this type that attach to the component under test are found.

Once the common nodes are found, Validate will fan out from each of these nodes and generate a list of nodes that are on the other side of components attached to these common nodes. Here, the

While adding nodes to the potential guard list from the B side of the component, the routine checks to see if the node to be added has already been added to the list from the component's A side expansion. If it has, the node is likely to be the best candidate for improving the test results. In this case, the node will be moved to the front of the potential guard node list directly behind any nodes that were added from Test Properties. In the case of the circuit shown above, node 37 will be found twice. The first time will be from the expansion of the A side of the component, and the second time will be from the expansion of the B side of the component. This node will be moved to the front of the potential guard list directly behind any user-supplied Test Properties guard nodes.

When Validate is finished with this process, the results are as follows:

Expanded list for side A: nodes 21, 22, 6

Expanded list for side B: nodes 41, 32, 35, 17

Fanout list for side A

R6

node 22	In expanded "A" list, so not a guard candidate
node 9	Add this node to potential guard list

R16

node 21	In expanded "A" list, so not a guard candidate
node 37	Add this node to potential guard list

C1

node 21	In expanded "A" list, so not a guard candidate
node 7	Add this node to potential guard list

AR1

node 6	In expanded "A" list, so not a guard candidate
node 13	Add this node to potential guard list
node 1	Add this node to potential guard list
node 0	Add this node to potential guard list
node 11	Add this node to potential guard list

Fanout list for side B

R18

node 32	In expanded "B" list, so not a guard candidate
node 37	Add this node to potential guard list

R27

node 41	In expanded "B" list, so not a guard candidate
node 19	Add this node to potential guard list

C9

node 41	In expanded "B" list, so not a guard candidate
node 45	Add this node to potential guard list

C25

node 17	In expanded "B" list, so not a guard candidate
node 20	Add this node to potential guard list

No attempt is made to guard node 4 because it is more than one component away and will not do anything that node 11 will not do.

From the above analysis the following nodes are determined as the potential guard node set that Validate will use when testing R1:

nodes 9, 37, 7, 13, 1, 0, 11, 19, 45, 20.

Algorithmic Analysis of Guard Nodes

After the list of potential guard nodes has been composed, a complex algorithm tries to reduce the number of guard nodes to the most effective points. Once again swapped and unswapped stimulus and measurement pole settings are evaluated. The result of this algorithm is a set of guard nodes and the preferred stim-meas pole setting.

The algorithm works as follows. The list of potential guard nodes is copied into a list for use with the stimulus and measure nodes as specified in Test Properties, and another list for use with the stimulus and measure nodes swapped. Starting with the poles as stated Test Properties, each potential guard node is tried one at a time until all have been tried. If a guard node does not improve the result, it is thrown out. If it improves the result by at least 1/2%, its value is remembered. At the end of this pass through all the potential guard nodes, the one that helped the most is moved to the top of the guard node list. If the resulting measurement brings us within the goal tolerance for the component, the process is ended, and this single guard found will be used. However, if the situation improves, but still isn't within the goal tolerance, Validate goes through the loop again trying the rest of the potential guard nodes one at a time, along with the guard node from the first pass. Thus, all the nodes that help are tried with all the other nodes that help in combinations that will give the best possible result. If at any time the goal tolerance for the component under test is met, Validate uses those guards and drops out of the loop.

When the possible combinations of guard nodes with the stimulus and measure nodes as specified in Test Properties have been exhausted, the guarding algorithm is tried again with the poles swapped. As a final step Validate makes one more measurement for each of the 2-pole configurations, taking into account the conditions of the Validate options. Validate uses the guards and pole swapping configuration that give the best result. It then decides which of the two configurations is better taking into consideration how much the signal bounced over several samples, and, if precise mode was turned on, how much the stimulus voltage was degraded. Based on these results, Validate copies the guard nodes found to the Test Properties guard node list.

Validate does whatever it can to bring the final test results within the acceptance tolerance which you have specified. If Automatic Accept is not turned on, and if Validate can't bring the test results within the acceptance tolerance, the Test Properties parameters for the component will not be altered. The component test will remain as it was before Validate was attempted, and a message will be written to the EXCEPT.LST file indicating that Validate couldn't bring the test results within the limits specified. If guards are found that will make the results better but not good enough to pass, the guards will not be used if Automatic Accept is not turned on.

Other Options

Automatic Accept

In some cases, getting the readings within limits with guarding alone is just not possible. In these cases, the measured value may be accepted as the nominal value for the component. The effects of adjacent components will always be factors in the measured value of the component under test, but this influence will be the same from board to board. In this case, if the Automatic Accept feature is enabled, the nominal value of the component will be shifted to a value matching the measured value. The component tolerance percentages will remain the same, so they will track the shift in nominal value. Components which have had their test parameters altered in this way will be listed in the EXCEPT.LST file.

Guards that were found to help are kept as part of the Auto Accepted test even though they did not improve the test results enough to make the test pass without shifting the acceptance limits.

If Validate can't achieve a passing test and AutoAccept is On, one of two things will occur.

1.If a stable shorthand test is not possible, the test is disabled, and a message is written to the EXCEPT.LST file indicating that the test has been disabled.

2.If a stable test is obtained, but the test results are outside the test limits,

- The test limits are shifted.
- The test is marked as validated.
- The test is marked as AutoAccepted
- A message is written to the EXCEPT.LST file saying that this component's test limits have been changed.

If Validate can't achieve a passing test, and AutoAccept is Off, the original test parameters as specified Test Properties are restored.

A message is added into the EXCEPT.LST file indicating that Validate can't bring test results within the goal tolerance. This allows you to run the section in Stop on Fail mode, and manually examine each failing test in the section to determine what action to take.

Calculate Wait

If you select Calculate Wait Times from the Setup/Validate menu, Validate determines if waiting a certain amount of time before taking a measurement will help make a more stable test. It does this by turning the stimulus off, applying a squelch for 100 ms, removing the squelch, turning on the stimulus, and watching the measured response to see how long it takes the signal to become stable. Either the larger of this settling time or the system default wait time is used as the wait time for this component.

Average Samples

If you select Average Samples in the Setup/Validate menu, the component is measured 255 times, and examined for how much the signal bounces. If the bounce is outside the goal tolerance, different sample sizes are tried starting with 10 and working up until a sample window size gives a bounce value which is inside the goal tolerance.

▼▼▼

10 SPECIAL AND ADVANCED APPLICATIONS

Chapter 10 contains information about the special and advanced applications possible on Z1800-series testers.

The level of information presented here assumes that you are familiar with the tester's hardware and software systems.

Measuring Frequencies

Frequency measurements expressed in Hertz or decimal multiples of Hertz are possible with the special application and interpretation of a Count signature. **Count** counts positive transitions of a measured signal while the Listen Window is open.

With the Listen Window open exactly one millisecond, the resulting Count signature is the same as the frequency in kilohertz. Decade-valued windows produce decimal-factored expressions of the measured frequency. For example, a 10-millisecond window results in Count signatures equal to the ten times the frequency in kilohertz.

None of the default Listen Windows are open for decade values of time. Default-programmed Count tests therefore report numbers proportional to frequency, but do not express frequency in decimal terms.

To program a Count signature as a decimal multiple of the true frequency, program an appropriate From-To pair and a Clock Divisor that result in the desired Listen Window length.

IMPORTANT: Measurable signals above 65535 counts will wrap around. For example, if 65538 positive transitions occur in the window, the result will be reported as 2. The tester does not report overflow for Count wrap-arounds.

Example 1—Higher Frequencies

To get a 10-millisecond Listen Window

- Set Clock to 26. This setting is useful for frequencies above 500 kHz.
- Measure From F9 TO F11.

The actual window is 9.984 milliseconds. The tester reads approximately 0.2% low.

The maximum display is 32767, corresponding to 3.2767 MHz, which exceeds the tester's speed. The upper speed limit is approximately 5 MHz, imposed by losses in fixture wiring and tester comparator circuitry.

Example 2—Middle Frequencies

To get a 100-millisecond Listen Window

- Set Clock to 49. This setting is useful for frequencies between 10 kHz and 500 kHz.
- Measure From F5 TO F13.

The actual window is 99.960 milliseconds. The tester reads approximately 0.04% low.

The maximum display of 32767 corresponds to 327.67 kHz.

Example 3—Lower Frequencies

To get a one-second Listen Window

- Set Clock to 62. This setting will be useful for frequencies between 10 Hz and 50 kHz.
- Measure From F10 TO F16.

The actual window is 999.936 milliseconds. The tester reads 0.01% low.

The maximum display of 65535 corresponds to 65.535 kHz.

Example 4—50,000 Hz In testing a 50,000 Hz signal to $\pm 1\%$ accuracy, in the one-second window, you would expect to see $50,000 * .999936 = 49996.8$ counts, nominal.

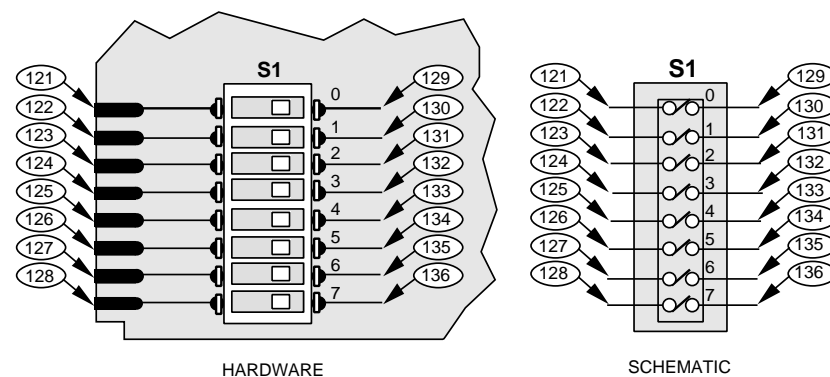
To determine tolerances for the test, add 0.1% for acceptable variation, 0.01% for tester tolerance, and $1/2$ count for quantization uncertainty. The upper count limit is 50052, and the lower count limit is 49941.

Testing Switches

Include switch contacts in the Ignore section of interconnects to prevent them from interfering with shorts testing and to verify that the switch contacts are not shorted to other nodes.

A simplified schematic of a DIP switch and nodal addresses is shown below. (Circled numbers are node number examples.)

Simplified illustration of DIP switch.



Using a programmed message under the Switches/Pots section, instruct the operator to set the switches to a known position for the remainder of the in-circuit test. For multiposition switches, generate a new set of operator instructions for each position.

For example, using the CRT output in the Pre-Test window, instruct the operator to

```
SET ALL SWITCHES IN S1 TO ON (DOWN)
THEN PRESS START TO CONTINUE.
```

Program the Test Properties portion of the Step Worksheet to perform the closed switch test(s). For the open switch test, use the Pre-Test window to instruct the operator to

```
SET ALL S1 SWITCHES TO OFF (UP) .
THEN PRESS START TO CONTINUE
```

Test Properties acceptance limits, stimulus, and computing resistor values are programmed at the beginning of the test sequence since they are the same for both sections of the test. The switch test examines individual switches in the “on” position first and then in the “off” position. The switches are then left off for the remainder of the in-circuit test. This test can also be performed in the opposite order depending upon which position affects fewer components while the program is executed.

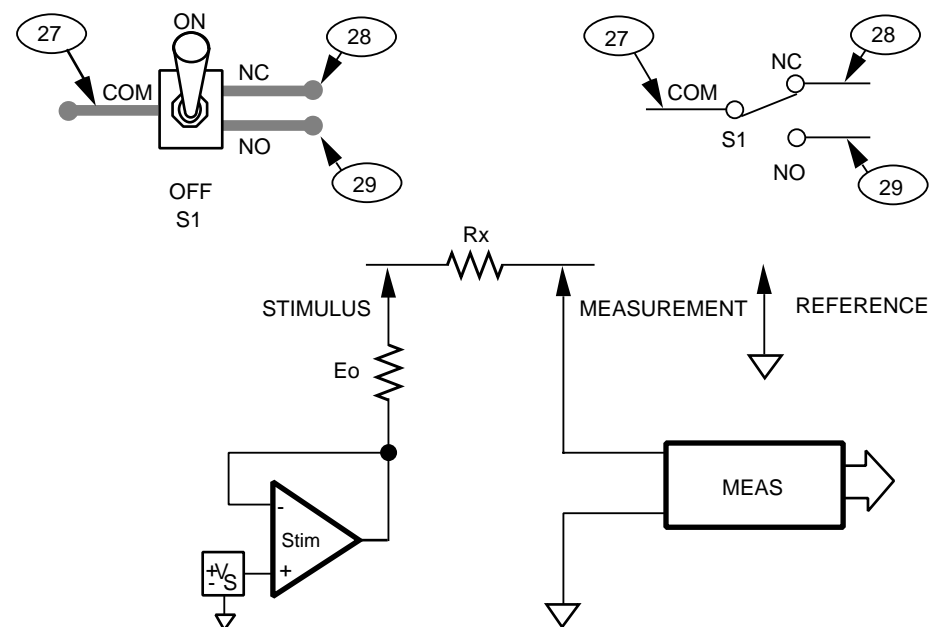
Notice that the first part of the test sequence ensures that the switches close when on. The second part of the test connects the stimulus and measurement buses together, then connects the Reference to the other side of the switch. The second part of the test ensures that both partial and catastrophic failures are located when the switches are off. Any leakage current flowing through the Reference node directly affects the measured response and will cause the test to fail.

Test V Stim V mode splits the analog test instrument into two sections: a programmable voltage stimulus source and a $3\frac{1}{2}$ -digit DVM. The computing resistor serves as a programmable current limiter for the excitation circuits.

The default Switch test measures the resistance of a closed or open switch. The default test type is Resistor; however, if the switch resistance is unknown, use the Test V Stim V test type.

If the impedance of the device-under-test is unknown, the largest computing resistor should be programmed during initial program generation. The correct value for that resistor can be found during subsequent program debugging. The node selection entry defines the nodes connected to the E bus (stimulus), F bus (measurement), and G bus (reference). The Test V Stim V configuration is illustrated below.

Simplified Test V Stim V circuit configuration and noded switch schematics



Testing Relays

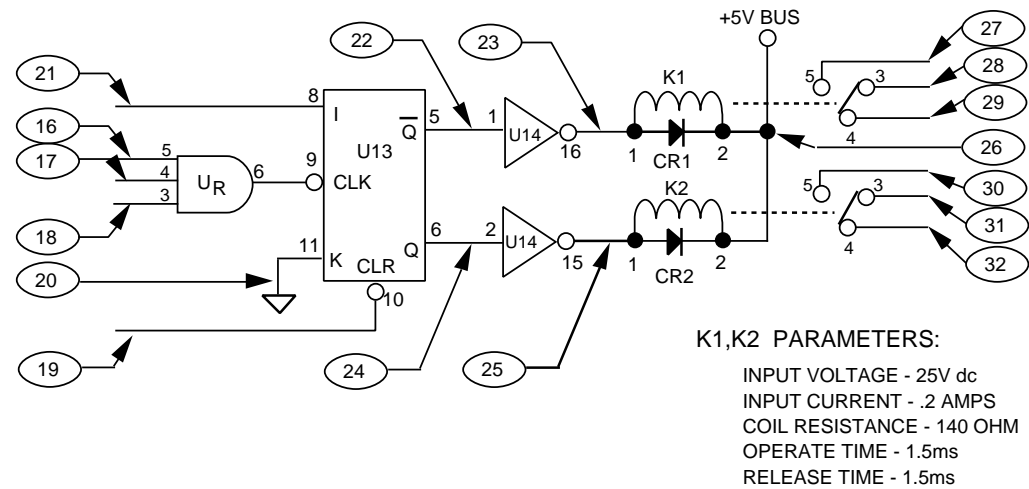
Relays have two positions, the passive or de-energized position, and the active or energized position. A complete relay test must be performed in two parts.

The first part of a relay test is a passive test that examines the relay contacts and coil resistance in a static or de-energized state. Include the coil and contacts in the Ignore section to prevent a defective relay from invalidating other subsequent measurements.

After all the power-off analog measurements have been performed, the relays can be energized and the relay contacts again examined to ensure that the relay is properly energized.

The following figure illustrates a typical relay circuit.

Typical relay circuit and node assignment example



The first step in a relay test is to ensure that the relay coil is neither internally shorted nor open by measuring the DC coil resistance. By examining the relay coil first, you identify and replace defective coils before running the entire program. After examining the relay coil, check the relay contacts with the relay in the off or de-energized state.

You can use two formats to develop a passive relay test. One format uses information concerning the relay coil resistance extracted from the relay data sheet for a shorthand resistance test. The other format applies to relays where the coil resistance is not known.

The typical test program examines the coil resistance of relay K1. If the coil resistance test fails, the program should inform the operator as follows:

K1 COIL FAILED

Use Before and After control to perform a conditional test. If the coil proves to be good, then examine the relay contacts in their de-energized state. Notice that when the normally open contacts are examined, both the E and F poles are connected to the common contact and the G pole is connected to the normally open contact. This method effectively tests for both partial shorts and catastrophic failures. Any current leakage through the normally open contacts directly affects the measured response.

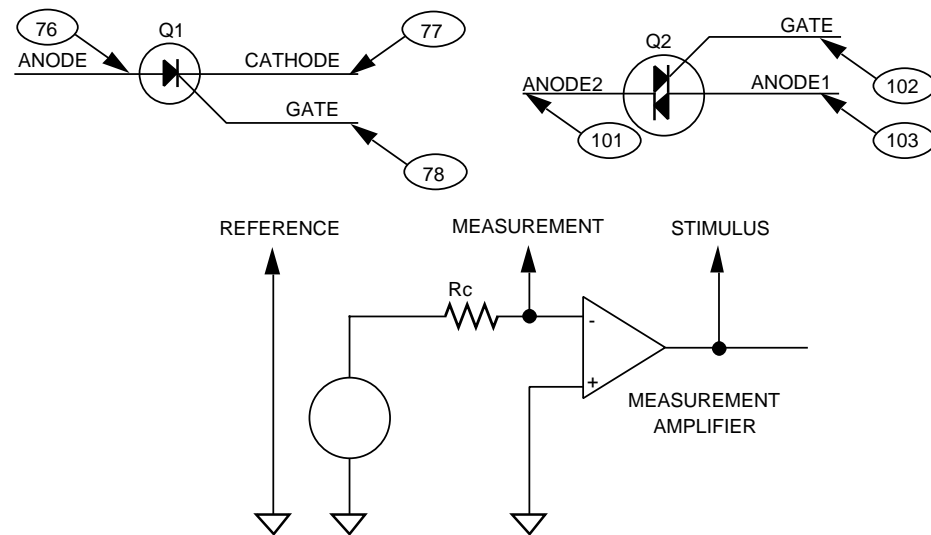
Testing SCRs/Triacs

SCRs (silicon controlled rectifiers) are semiconductor devices that block voltages applied from either direction.

A Triac is essentially two SCRs in parallel. Triacs are bidirectional and can be triggered into either forward or reverse conduction by a pulse applied to its Gate electrode. Triacs usually function as electrically controlled switches for AC loads.

The figure below shows typical SCR and Triac circuits. (Sample node numbers are circled.)

Typical SCR and Triac circuits and test configuration



The test for SCR on voltage is similar to the forward-bias voltage test for a diode. The analog measurement circuits are configured for Test V Stim I. The Anode is connected to the measurement node, the Cathode is connected to the stimulus node, and the Gate is guarded. The Gate electrode is at the same potential as the measurement node, with the Cathode negative in respect to both. Applying a stimulus current to this device essentially turns it on. Gate control is usually lost once this happens.

After the SCR turns on, the normal cathode/anode forward voltage drop can be measured. If the forward voltage drop passes the examination, examine the reversed-biased characteristics.

Like the diode, the SCR off current is examined by applying a reverse-biased voltage across the device and examining the resultant current in Test I Stim V mode. As with the SCR on test, the Gate electrode is guarded. The same switch setup is used for the reverse-bias test as was used for the forward-biased test.

For most SCRs, the forward voltage drop is approximately 1.0 volts with a stimulus current of 100 mA, while the reverse leakage current of 1.0 mA or less with a 10 V stimulus is usually acceptable. However, the reverse-biased test uses a negative stimulus and measures the leakage current in Test I Stim V mode. The same node arrangement used for the forward-biased measurement is kept for the reverse test.

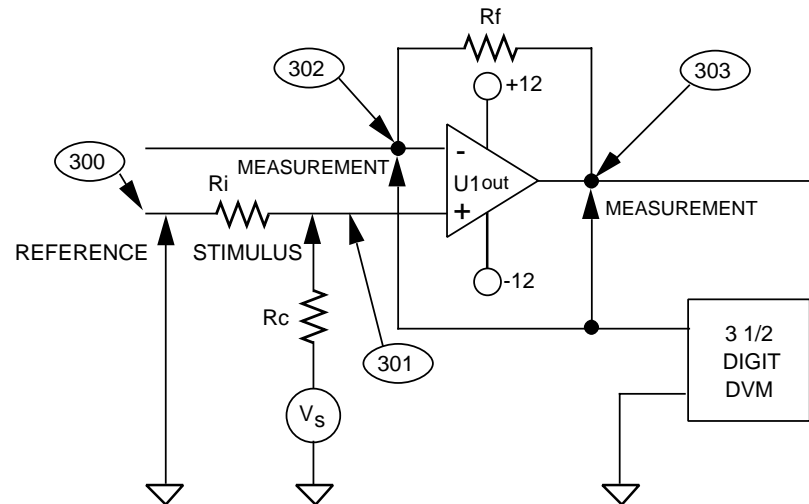
The SCR test can use the Analog menu and the Test V Stim I test type Test Properties on the first page, with a tolerance of 10%, for example. The acceptance range is 0.9-1.1 V (-100 MA). Page two uses the Test I Stim V test type with a range of 0-1MA (-9.99)V. Specify appropriate nodes in Test Properties Nodes/Timing column.

Since Triacs pass current in both directions, the off condition is impossible to measure. The Triac on voltage is testable in both directions. Use the same basic parameters for the SCR on measurement for the Triac's forward voltage drop in one direction, Anode 1 to Anode 2, then in the other direction, Anode 2 to Anode 1. In both cases, the Gate electrode held is at ground potential through the guard connection.

Testing Operational Amplifiers

Op amps are tested in a voltage-follower configuration, as illustrated in the figure below. (Sample node numbers are circled.)

Operational amplifier circuit example



The requirements for op amp testing are that

- Board Power must be previously applied and verified.
- The op amp input resistor R_i should have been tested and found to be within tolerance.

The tester allows multiple nodes to be connected to any of the three buses (Stim, Meas, and Reference). To configure the op amp as a voltage follower, connect the Measurement bus to both the output and the negative input of the op amp. Place the reference on the side of the input resistor away from the op amp input. Connect the Reference to the printed circuit assembly's ground reference—make sure that this node is also in the header Reference Node. Apply the stimulus directly to the non-inverting input of the op amp. Use the Test V Stim V Test Type in Analog Test Properties to apply the stimulus and measure the response.

The expected output of the op amp can be computed using the formula:

$$V_{out} = (R_i / (R_i + R_c)) (V_s)$$

where:

V_{out} is the output of the op amp under test

R_i is the value of the input resistor

R_c is the value of the computing resistor

V_s is the value of the voltage stimulus

Generally, different stimulus voltages are used to check the linearity of the op amp. Assume that the op amp illustrated in the previous figure is an LM741 with a 27 kohm input resistor. Assume the op amp operates between +12 and -12 volts referenced to ground. In order to test the op amp over as wide a range as possible, apply stimulus voltages of 1 volt and 9.99 volts DC through a 10 k Ω resistor. Assume a 10% tolerance.

The output voltages are computed as

$$\begin{aligned} V_{L_{out}} &= ((27 \times 10^3) / ((27 \times 10^3) + (10 \times 10^3)))(1) \\ &= 7.3 \times 10^{-1} \\ &= .73 \text{ volts} \end{aligned}$$

$$\begin{aligned} V_{H_{out}} &= ((27 \times 10^3) / ((27 \times 10^3) + (10 \times 10^3)))(9.99) \\ &= 7.3 \text{ volts} \end{aligned}$$

Therefore, the acceptance limits are

$$\text{low} \quad 73 \times \pm 10\% = .66 - .80 \text{ volts acceptable}$$

$$\text{high} \quad 7.3 \times \pm 10\% = 6.6 - 8.0 \text{ volts acceptable}$$

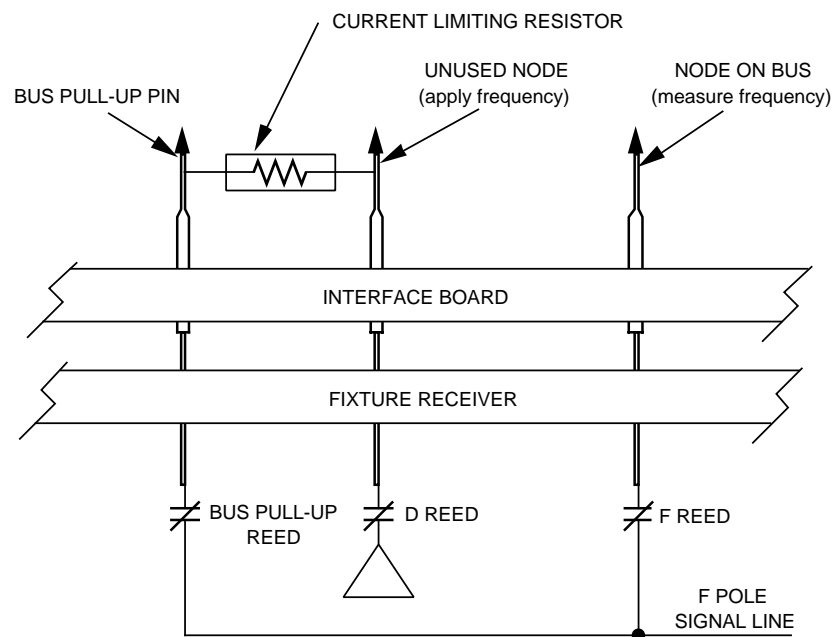
Enter the acceptance limits and all parameters into Test Properties. Use a different Test Properties page for each stimulus.

Developing a Bus Test

Complex digital circuits transfer information between devices across data buses. Data buses are usually connected in parallel to many devices, especially memory circuits. If one device fails, all the devices connected to the data bus fail.

Examine the data buses at the beginning of a digital test program to ensure they are not artificially held to a logic state.

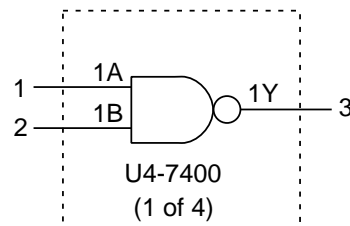
Bus pull-up resistor configuration diagram



Testing Gates

Complete truth-table testing of AND, NAND, OR, and NOR gates is relatively easy because of the way in which the stimulus frequencies are developed. Although the stimulus frequencies are synchronized, none of the frequencies transit (go high or low) during the same time interval. Therefore, it is possible to perform a complete truth-table test using a different stimulus frequency for each input. The data bit stream output then represents the entire truth-table of the gate-under-test. Since gates have a known beginning logic state, you can program a CRC measurement to translate the data bit stream into a four-bit hexadecimal signature. The 2-Input Positive NAND Gate (7400) is a simplified example of a testable gate.

Simplified test schematic of 2-Input NAND gate



After drawing a simplified test schematic and labeling the input/output pins, you can develop a prototype test module. Since each of the four NAND gates in this device is identical, it is only necessary to develop a test for one gate, then use the test as a template for the remaining gates. The stimulus frequencies supply all possible logic state combinations during the test sequence. The two stimulus frequencies chosen for this test are Gray codes F2 and F3.

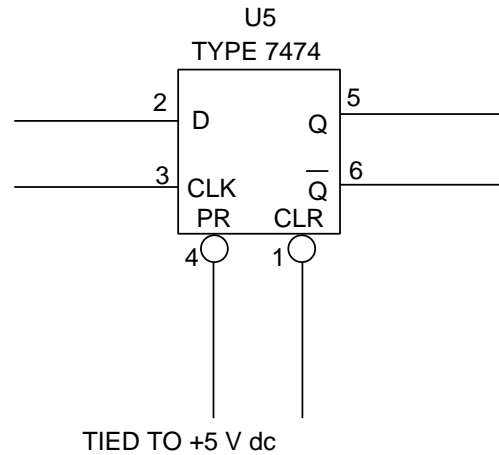
To reduce overall program execution time, it is important to choose the fastest stimuli that produce a stable signature. Since the beginning input/output state of this device is known, you would use a CRC-type signature to characterize the output.

Testing Flip-Flops

Many flip-flop devices have asynchronous preset and enable inputs that can be used to set the device to a condition independent of data and clock inputs. One such device is the 7474 D-Type flip-flop, U5, illustrated in the figure below.

The key to effective testing of flip-flops or other triggerable devices is to place the highest stimulus frequency (F1) on the clock input and lower stimulus frequencies on other inputs. Intermediate frequencies can be used for the data inputs.

Simplified test schematic of D-type flip-flop



The stimulus chosen for the D-Input in the prototype test module is F2. There are two control inputs to this device, PR (preset) and CLR (clear); however, since the PR input is tied directly to +5V DC, only the CLR input can be used to control the beginning state. A PL, or Preset Low pulse, is applied to the input to clear the device before the beginning of the test sequence. Since the device can be forced to a known condition, a CRC measurement can be programmed to examine the output.

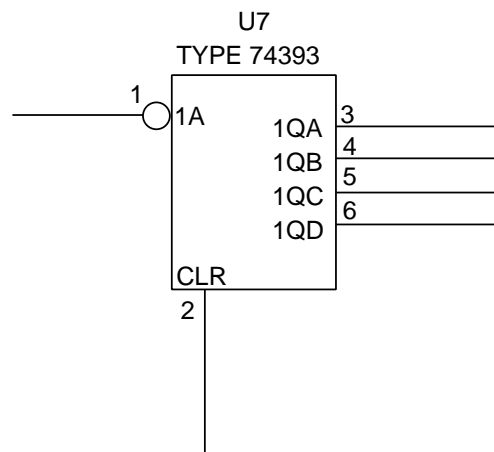
Testing Counters

A counter is a circuit designed to count input pulses. The counter output is capable of changing between a sequence of distinguishable states when it receives a discrete input signal. The sequence is incremented or decremented at a binary or decade rate. More complex counters have control inputs that allow the counter output to be set to a predetermined value before the counting of the input pulses begins. The counter used for the programming example (U7) is a DUAL 4-Bit Binary Counter with a clear input (74393) as shown below. Although only one-half of the counter is shown, it can be used as a template for both halves.

As with other devices that have clock inputs, the highest stimulus frequency (F1) is applied to the clock input. However, the counter used in the programming example has an active low clock input which means that the counter is incremented by a negative-going clock pulse. You would then use a complement stimulus (F1*) as the clock input.

The example counter also has a clear (CLR) input that forces the device to a known beginning state before counting begins, and permits a CRC measurement on the outputs. Four measurements are made with each test execution cycle with the measurement window designated TO F8, to ensure that enough clock pulses activate the output channels.

Simplified test schematic of binary counter

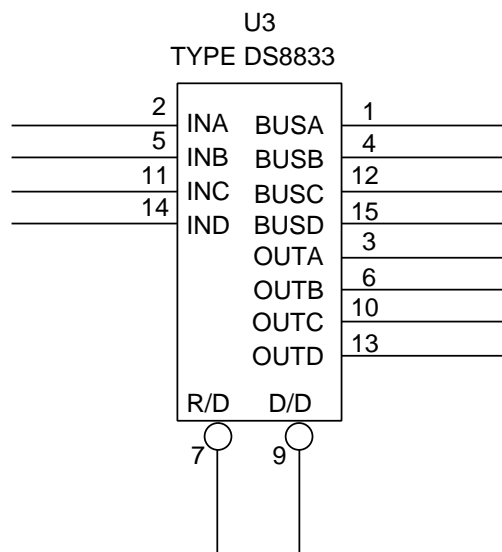


Testing Line Transceivers

A line transceiver is a device that can both transmit and receive data signals. Examples of the use of line transceivers are input/output terminals for memory devices or data bus transceivers.

The line transceiver (U3) illustrated below has four bidirectional input/output lines, and two control inputs.

Simplified test schematic of line transceiver



There are two main problems encountered in developing a test for this device:

- data flow is bidirectional and must be controlled in order to test the device thoroughly
- the data output channels must be forced to a known logic state in each direction of data flow.

The two control inputs, RD and DD, are used to control the direction of data flow. These two active low inputs enable either the data receive circuits (RD) or the data transmit circuits (DD). Both the bus drivers and the output drivers should be tristated when disabled. When RD is low and DD is high, the bus drivers are tristate and information presented to the BUSA through BUSD inputs is coupled to the data outputs, OUTA through OUTD.

When RD is high and DD is low, the bus drivers are enabled and the data receivers are tristated. Information presented at the data inputs, INA through IND, is coupled through the bus drivers to the bus outputs, BUSA through BUSD. To test this device thoroughly, each individual data channel should be examined in each operational mode by applying two slower stimulus frequencies to the control inputs and faster stimuli to the data inputs.

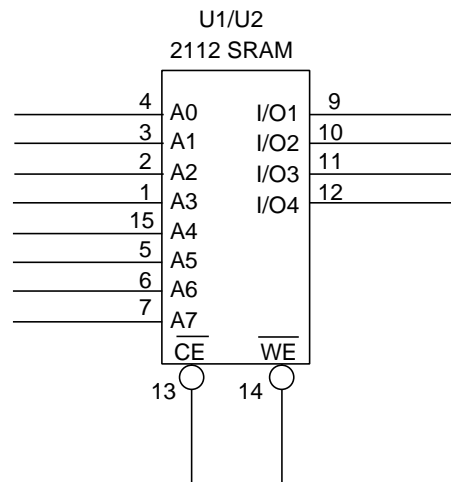
The control stimuli for this test are F9 for the DD or Driver Disable input, and F6 for the RD or Receiver Disable input. Stimulus frequencies F1 through F4 are applied to the data inputs, INA through IND.

Since the highest stimulus frequency used is F9 and a measurement window is not specified, the default measurement window is the first transition of F1 through the first transition of F11. At the beginning of each test execution cycle, both the DD and RD inputs are enabled. Any stimuli presented to the data inputs are coupled to both the BUS outputs and the data outputs. When F6 goes true (high), the RD input is disabled and no more information is coupled through to the data outputs. However, the BUS outputs are still enabled until the stimulus applied to the DD input, F9, goes true, using one stimulus at the data inputs to check both the data and bus outputs.

Testing Memory Devices

The memory devices examined in digital testing include most ROMs and RAMs. Since ROMs have data already stored in different memory locations, the test sequence needs only to examine those memory addresses and compare the measurement signature with that from a known good device. RAMs, however, require more complex tests. Typically, RAMs have combined input/output lines, memory location address inputs, and control inputs. The input/output lines are used to write data into the memory device and store the data in the location defined by the address inputs. The same input/output lines are also used to read data from the memory location defined by the address inputs. The control inputs determine if the device is in a read or write mode. Two of these devices (U1 and U2) are illustrated below.

Simplified test schematic for 256x4 RAM memory



These devices have eight address inputs (A0 through A7), four input/output data lines (I/O1 through I/O4), and two control inputs (CE and R/W*). The two control inputs are used to:

- clock the data in or out of the device (CE), and
- select either the write or read mode of operation (R/W*).

Since the four input/output data lines are bidirectional, data must be read in and out of the same line during a test execution sequence with the stimulus connect/disconnect functions. The measurement window is open only when data is being read out of the memory.

In order to examine all memory locations during both the memory write and memory read cycles, stimulus frequencies F2 through F9 are applied to the address inputs. The CE*, Chip Enable, input is used to clock data into or out of the memory and is an active low input. Following the stimulus rule for clock inputs, stimulus frequency F1* is applied to the CE* input. The stimulus applied to the data input/output lines must be between the clock stimulus and the highest address stimulus. The stimulus also must be disconnected when the measurement is taken during the memory read cycle.

Using the While function to designate when the measurement is to be taken restricts the measurement window to take place only while stimulus F10 is true. A Clock is also added prior to measurement execution to slow the test execution sequence, accounting for the slow cycle time of the RAM. A CRC measurement is used to control the data entered into and read out of the device.

Burst Time and Duty Cycle Supervision for Backdrive Protection

When the tester asserts a logic level on a node of the board under test, it may disagree with and therefore drive against the IC output which normally determines the state of that node. In such a case, the IC output is said to be “back-driven.” An IC dissipates more power when being backdriven than it does in normal operation. The amount of added power is determined more by the IC than by the tester, i.e. an IC of a logic family with low output impedance will consume more backdrive current than one with high output impedance, and therefore dissipate more added power.

The added power of backdrive is converted to heat in many ways within an IC. Some heat is produced by recombination of electrons and holes at impurity junctions within the crystalline silicon. Some is produced as ohmic heat, either in the silicon or in the resistances of the bond wires. Heating of any part of the IC during normal in-circuit test is small, because (1) during any given test step, most outputs are not involved, so are not being backdriven, (2) test steps consist of short bursts, with most bursts being only a few tens of microseconds long, and (3) on the average, any given signal at any given point in a test burst is as likely to be in agreement with the tester driver as it is to be in disagreement.

The Z1800-series is equipped with a supervision system for control of burst time and duty cycle. This supervision system is under direct software control; the user may set it according to any favored rule. It is programmed in the tester's Setup/Environment menu, and its setting governs the operation of all programs that run on the tester. See chapter 2, “System Setup,” for information about programming backdrive timeout and duty cycle.

If a test burst is longer than the maximum specified in this menu, the test will reject, and the following types of messages appear depending on the mode you are in:

- Production mode—“No Measure—Step Error” (datalogged)
- Debug mode—“Backdrive time exceeded” (error message)

If you see one of these messages, it means that one of the tests in your program has a burst time longer than the maximum you specified in the Setup/Environment menu or the PRGMVARS window. Choose one of the following remedies that is most harmonious with your testing situation:

1. Increase the value of the maximum burst time in the Setup/Environment menu.
2. Shorten the test burst.
 - 2a. Use a smaller CLOCK divisor.
 - 2b. Remove vectors from the end of a VECTOR test.
3. Exempt this one test from burst time supervision: in the digital Step Worksheet, pull down the Options menu from the menu bar, and select Controls. The Test Step Controls window appears. Change Backdrive Timeout control from Enable to Disable.
4. Remove this test from your test program.

The tester controls burst application time and duty cycle. It does not control the number of channels involved in a test: indeed, a vector test can be constructed that will legally involve all the channels in the tester. The observation is sometimes made that if all outputs of a multiple-output device were simultaneously backdriven against their normal state, and the backdrive were maintained continuously for a long period of time, excessive heat might raise a bondwire to a temperature high enough to cause damage to the backdriven IC. The damage would occur in a ceramic package through recrystallization of the bondwire metal or, in a plastic package, through melting or charring of package material surrounding the bond wire.

The bondwire in question is one of the IC's power supply bondwires, which must carry the combined backdrive currents of all the backdriven outputs. As noted above, this extreme backdrive event rarely, if ever, happens in practice. ICs are very forgiving of backdrive, and can endure almost anything the tester is capable of applying for the following reasons:

- Real test bursts are short, not long.
- Driving all the outputs of a low-impedance IC to a constant high level for a long period of time contributes no test information, so it's never done in a real test program.
- IC ground and Vcc bondwires in low-output-impedance logic families are usually doubled, cutting the ohmic dissipation to 25% of that of a single bondwire, and doubling the rate at which that heat can escape.

IMPORTANT: If chip damage is suspected, please contact Teradyne immediately.

The safety of your products is of the utmost importance to Teradyne. Backdrive may not be the cause of damage to IC devices. If you have observed chip damage, consider the following possibilities.

- Is the problem caused by latchup instead of by backdrive overstress? History, in many thousands of installations, tells us that damage from excessive backdrive never occurs, but damage from latchup sometimes does. Latchup can be eliminated by improving the ground and Vcc wiring in the test fixture. (See the **Fixturing Manual** for more information.) Poor ground and power wiring in fixtures allows power supply rails on the DUT to shift with respect to power rails in the tester during backdriving cycles. This pulling of DUT nodes outside of the DUT power supply rails stimulates latchup. If proper fixture power wiring rules have been followed, latchup is unlikely. Improving the wiring in an inadequately-grounded fixture will end the problem. Specific troubleshooting for latchup can be done with a current probe measuring the DUT supply current. If large spikes are present, attempt to correlate them with a specific stimulus in a specific test. The stimulus in question may not even be connected to the IC being latched up.
- Is there a large capacitor on your board charged to a voltage greater than the logic supply voltage? If it is not discharged before the board is removed from the tester, and the board is then placed on a conductive surface, the voltage of the capacitor may be shorted to a logic node and damage an IC. The Discharge section, which is rerun upon execution of the Power down in the Trailer, can be programmed to discharge such capacitors.
- Is the burst time /duty cycle supervision system operating? This supervision system will regulate backdrive application, but will not control latchup or accidental damage due to handling of boards with charged capacitors. Choose the Setup menu and then choose Environment to see what numbers are in the Backdrive Timeout and Duty Cycle fields. Zeroes indicate "no supervision."
- Are there a lot of stimuli in the Disable page of the digital test program? Normally, only a few are needed. These are signals which will be applied on every burst in the program, that is, these stimuli are repeated many times more than stimuli in an individual digital test page. Do any of the disable stimuli drive outputs of the device that is being damaged? Are more than three or four outputs of that IC simultaneously being backdriven high when they would otherwise be low? Is this the IC that is being damaged? Can you apply additional stimuli that will precondition the outputs to agree with the Disable stimuli so that the desired disabled state will occur, but backdriving will not actually occur? This is a more gentle way to do the disable function.

Effects of THC and VP on Backdrive

The type of test head controller installed in your tester affects the backdrive timeout function.

If your tester is equipped with the THC or VP prior to the E.0 revision of the system software and the VP3 (PN 051-048-00), backdrive timeout behaves differently for Gray code and vector tests. For Gray code tests, the backdrive timeout is calculated prior to running the test. If it is determined that the run time will exceed the timeout, the test does not start.

For vector tests, there is no backdrive timeout, and the tests run without protection.

If your tester is equipped with the VP3 (PN 051-048-00) and you have E.0 (or later) revision software, both Gray code and vector tests run until the timeout specified in Setup/Environment or Header/PRGMVARS.

Accessing External Programs

The External Program function enables you to call a user-specified program such as an existing DOS program to run at a particular point in the in-circuit test program. After the external program has run to completion, your in-circuit test program returns to this point.

IMPORTANT: During an 18xx power down, you can turn off external power supplies which may be used with external programs, by enabling External Power Down in PRGMVARS/General Variables. See chapter 5 for more information.

You can access External Program either through Test Type or the Pre- or Post-Test Options menus in the Test Properties portion of the Step Worksheet, or Header/Trailer message steps.

Using Test Type

To set up External Program using Test Type,

- 1 Select Test Type, and then select External Program from the Test Type list.
The External Program Test Properties section replaces the Test Properties section of the original worksheet.

The screenshot shows a window titled "18xx Windows mode-init_com" with a menu bar (Edit, Options, Tools, Save, Revert, Validate, Quit). The window is divided into several sections:

- Component Properties:** Contains fields for +ID: L1, Name: 1 mH, Desc: Ind., 6 wire mode, Value: 1.000 mh, Tol: 15, Device Type: Inductor, and Number of Pins: 2.
- Test Properties:** Contains sub-sections for Options (Pre, Post, Cntrl), Indicators, and Current Page (Page 1 of 1).
- Test Type:** Set to External Program.
- Test Data:** Contains fields for Scale: milli, High: 1.020, Low: 0.800, Execute Mode: Direct, 18xx Hotkeys: Disable, Environment Refresh: Off, Screen Refresh: Off, Flush COM Ports: No, E Pole: Pin 2 (75), and F Pole: Pin 1 (68).
- Controls:** Contains a field for Wire Mode: 6.

At the bottom of the window, there is a status bar with the text "C:\TPD\DEMO-ICT" and a row of buttons: START, CANCEL, SOF, HLD, RPT, CRT.

2 Set the Execute Mode.

The Execute Mode can be Direct or Command.com. If you select Direct, the external program cannot be a batch file, nor can it include internal commands, such as “copy” or “cd.” However, it can return a non-zero value. If you select Command.com, you can run batch files and use internal commands, but the return value is always zero unless an error occurs.

3 Set Hotkeys, Environment Refresh, and/or Screen Refresh to configure the working environment of the external program and whether the environment and screen parameters are refreshed upon return from the external program.

If the 18xx hotkeys (F1–F12) are disabled, the external program can redefine them for its own purposes. If the 18xx hotkeys are enabled, then they keep their 18xx definitions.

If Environment Refresh is on, upon return from the external program, 18xx system software copies the current state of the TSR environment into the image. If off, any changes made to the TSR environment by the external program are discarded.

If Screen Refresh is on, the screen is refreshed upon return from the external program. If off, the screen is not refreshed.

4 Set Flush COM Ports.

If your external program uses a COM port, you can enable Flush COM Ports to ensure that no data is in the buffers when the external program starts.

5 Set up the external program call by typing the path to the external program in the Command Line.

For example, to run the DOS check disk program type

```
c:\dos\chkdsk
```

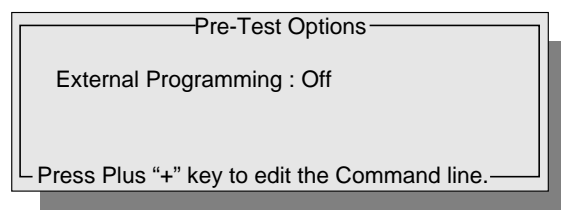
If the program requires an assembler or compiler other than C language, the command line must include the assembler or compiler. For example, for a QuickBasic language program, the command line would be “qbasic [pathname to executable program]”.

Using Pre-Test or Post-Test Options Menu

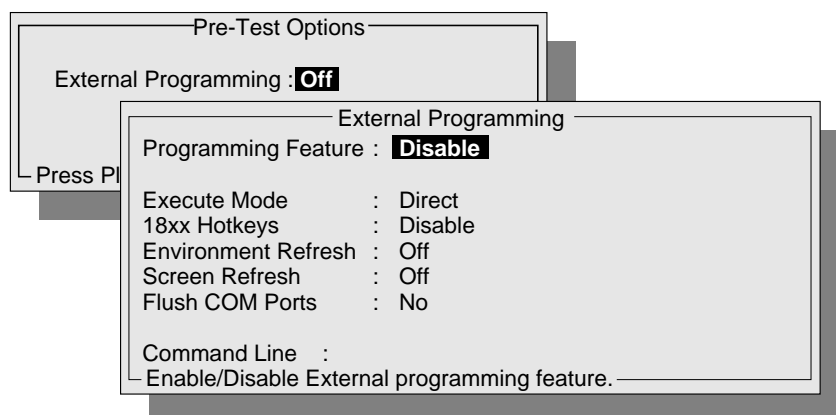
To access External Program through the Pre- or Post-Test Options menu

- 1** Select Pre-Test Options (or Post-Test Options) from the Edit/Options menu.
- 2** Select External Program.

The following window appears:



- 3 Click Off to display the External Programming window.



- 4 Select the Programming Feature field, and then select Enable to call the program named in the Command Line.

Programming Feature enables or disables the calling of the program named in the command line. Even if the other fields are enabled, they will be ignored if Programming Feature is not enabled.

- 5 Set up Execute Mode, 18xx Hotkeys, Environment Refresh, Screen Refresh, and Flush COM Ports as outlined above.
- 6 Type the path to the program in the Command line, for example

c:\dos\chkdsk

Now your program is set up to call an external program when the worksheet's Pre-Test (or Post-Test) Option is executed.

Using IEEE Instrumentation

The IEEE test type allows you to perform tests with optional IEEE-488 bus peripheral equipment. The test outputs data to an IEEE instrument and captures the results made by the instruments. Obtain hardware access to the IEEE instrument through the PC's serial port and a serial/IEEE converter module. You may connect measurement signals through an optional Relay Array board. See the **Z1800-Series IEEE Guide** for detailed information about performing test using IEEE instrumentation.

When you select IEEE as the Test Type from Intc/Discharge, Passive, Semi, PwrOff, Brd_Power, and Linear Test Properties, a Test Properties page similar to the following appears.

The screenshot shows a window titled "18xx Windows mode-init_com" with a menu bar (Edit, Options, Tools, Save, Revert, Validate, Quit) and a scrollable content area. The content is organized into several sections:

- Component Properties** (with up/down arrows):
 - +ID: C11
 - Name: 1uF
 - Desc: Cap, 3 wire test w/guard
 - Value: 1.000 uf
 - Tol 1 : 5
 - Tol 2: 5
 - Device Type: Capacitor
 - Number of Pins: 2
- Test Properties** (with up/down arrows):
 - Options**: Pre, Post, Cntrl
 - Indicators**: (empty box)
 - Current Page**: Page 1 of 1
 - Test Type: IEEE
 - IO Device: IEEE
- Test Data**:

Scale: micro	Format Skip : 0	E Pole : P1 (68)
High : 1.050	Format Use : 0	F Pole : P2 (75)
Low : 0.950	Format Reverse : No	G Pole :
	Format Length : Exact	Wait1 (ms):0
		Wait2 (ms):0
- IEEE String**: (empty text field)
- Controls**:
 - Wire Mode: 3
 - Keep Stim: No
 - Keep Nodes: No

At the bottom of the window, the status bar displays: C:\TPD\DEMO\DEMO_179 - ICT START CANCEL SOF HLD RPT CRT

The following table lists the fields in the IEEE Test Properties portion of the Step Worksheet and provides a description of their functions and ranges.

Field	Description
IO Device	Indicates which device this page should output to/input from.
Scale	Power unit modifier—pico, nano, micro, milli, unit, Kilo, Mega.
High	High measurement limit.
Low	Low measurement limit.
Format Skip	Number of incoming characters to ignore. Range = 0 to 64.
Format Use	Number of characters to compare to high and low limits. Range = 0 to 64.
Format Reverse	YES or NO. YES= start with least significant bit. NO= start with most significant.
Format Length	Exact or Maximum. Specifies string length defined by Format Use and Format Skip. Default is Exact.
E Pole	Pin/node number(s) to which stimulus applied. 5 nodes max.
F Pole	Pin/node number(s) at which measurement is taken. 5 nodes max.
G Pole	Pin/node number(s) of the analog guard point(s). 10 nodes max.
Wait 1 (ms)	0 to 65534 ms. Wait state after relays set but before IEEE programming.
Wait 2 (ms)	0 to 65534 ms. Wait state after IEEE programming but before IEEE read.
IEEE String	I/O register to send commands to IEEE devices. Input operation required.
Keep Stim	If yes, this test retains the analog stimulus from the previous page/step.
Keep Nodes	If yes, this test retains the “nodes connected” from the previous page/step.

To edit **Scale**, click the Scale field. A pop-up window appears listing the following scales.

pico
nano
micro
milli
unit
Kilo
Mega

Click the appropriate value to select it.

To edit the **High** and **Low** fields, highlight each one and type in the appropriate value.

The **Format Skip**, **Format Use**, and **Format Reverse** fields format and compare IEEE data. In the Format Skip field, you may discard 0 to 64 characters of the incoming stream. In the Format Use field, you may use 0 to 64 characters of the incoming stream to compare to High and Low.

To edit the Format Skip and Format Use fields, highlight each one and type in the appropriate value.

IMPORTANT: If the Format Use field is 0, this page will not wait for IEEE input. It is IEEE output only.

Format Reverse reverses the incoming data stream. You must set it to either Yes or No. When Yes, the incoming stream starts with the least significant bit first. Set at No, the incoming stream starts with the most significant bit.

To edit Format Reverse, click the either Yes or No to bring up the toggle box. Click the desired word.

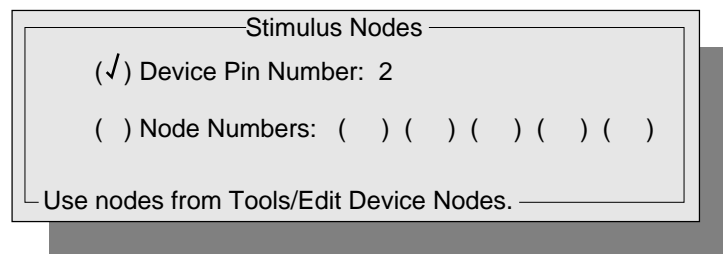
Format Length can be either Exact or Maximum. Exact is the default and causes the IEEE test to expect a string of exactly the number of bytes defined by the Format Use and Format Skip. Maximum causes the IEEE test to accept a variable length string where the maximum length is defined by Format Use and Format Skip.

For IEEE tests, the measurement node is the F-pole (measurement pole). 3-wire testing is the standard test mode using the **E, F, and G poles**.

To edit the E, F, and G pole fields, click their respective fields to bring up the Stimulus Nodes, Measure Nodes, or Guard Nodes window.

For IEEE test types, the stimulus node is the E-pole (excitation pole). 3-wire testing is the standard test mode using the E, F, and G poles. Edit the E-Pole field in the Stimulus Nodes window.

For example, when you select the E-pole field, the Stimulus Nodes window appears.



For most component tests, the stimulus node is on the end of the device having the most components. (The measurement pole generally resides on the more isolated end of the component.) Placing stimulus and measurement on the ends facilitates guarding when required, and reduces the effects of noise.

Wait1 and **Wait2** fields apply to the IEEE test type. Wait 1 and 2 function in IEEE tests as follows:

1. Set relays (including those on optional user relay array).
2. Wait 1
3. Program IEEE instrument.
4. Wait 2
5. Read data from the IEEE port.

The acceptable range for Wait1 and Wait2 is 0 to 65,534 milliseconds (ms).

IEEE String allows you to send and receive IEEE device commands. You must include the syntax for both an input and an output operation. You may not send output-only (setup) commands. (Use the Pre and Post-Test I/O Control menus for output only.) Test Properties processes the data it receives from the IEEE instrument through the Format fields, and compares the result to the High and Low fields.

The **Controls** section of the IEEE worksheet allows you to control the Wire Mode, Keep Stim, and Keep Nodes.

Digital Function Processor

The optional Digital Function Processor (DFP) is a flexible platform for performing value-added test and product functional test. DFP applications program non-volatile memories such as EEPROM and Flash ROM.

The 18xx interface provides setup and access to the DFP software. The Header variable, DFP,

- enables DFP
- specifies the source directory path
- specifies communications channels AUX 1, 2, or 3

The DigFuncProc Device Type available in the Component Properties portion of the Step Worksheet (except Interconnect) enables you to generate the DFP Test Properties page.

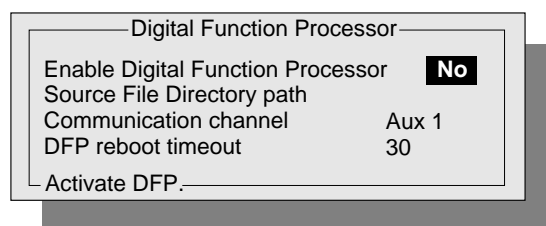
IMPORTANT: You can have vector guards only in a DFP test that is in a digital section.

Setting Up DFP in PRGMVARS

To setup DFP in the Header/PRGMVARS

- 1 Click the DFP Configuration field in PRGMVARS.

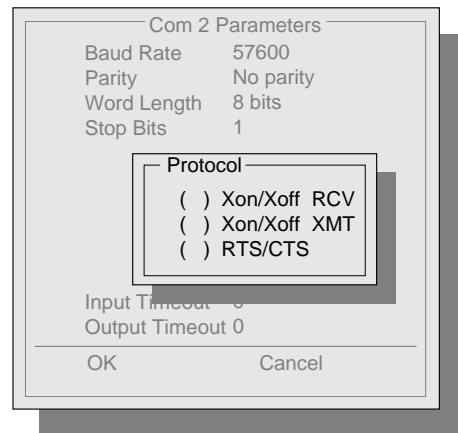
The following window appears.



- 2 Click No in the Enable Digital Function Processor field to bring up the pop-up box, and select Yes to activate DFP.
- 3 In the Source File Directory path field, enter the complete path to the main DFP directory.
- 4 In the Communication channel field, select the Auxiliary channel which contains the communications port for the DFP channel.

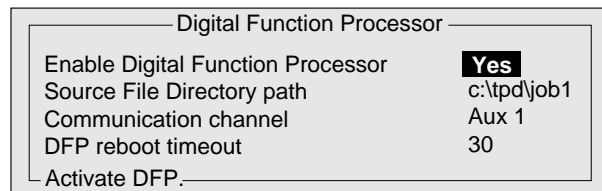
This should have been specified in the Setup/Device & Channel Data window.

IMPORTANT: When you installed the optional DFP, you set up communications port parameters for DFP in the Com Parameters window. Those settings, shown in the illustration below, never change.



- 5** In the DFP reboot timeout field, enter the maximum number of seconds to allow the DFP to reboot. You can enter a value from 30 (default) to 360.

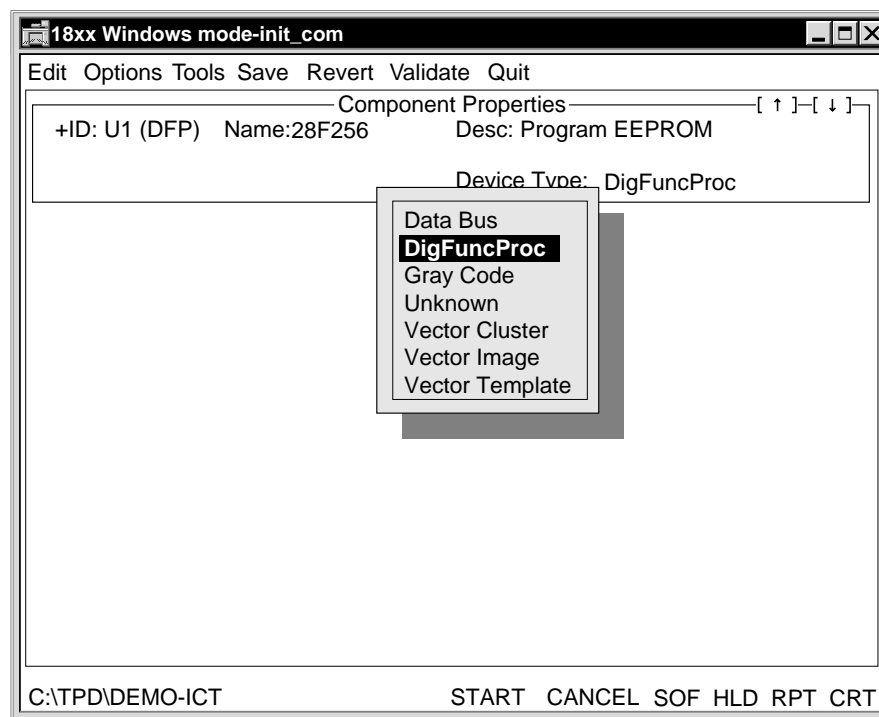
When you have finished setting up DFP, the DFP PRGMVARS window should look similar to the following:



Generating DFP Test Properties

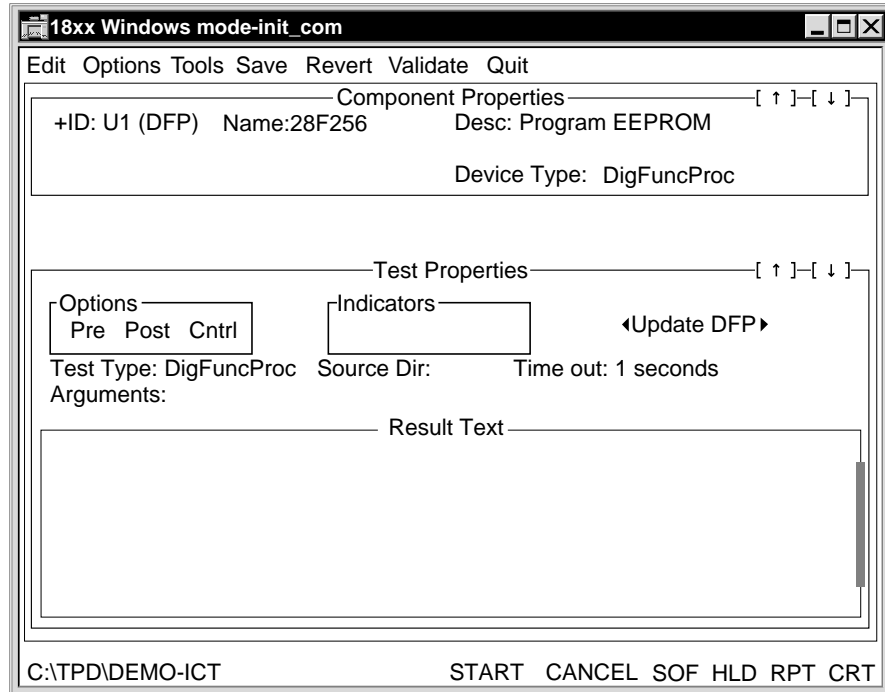
To generate DFP Test Properties

- 1 In the test page's Component Properties, click the Device Type field.
A window appears listing the device types available.



- 2 Select DigFuncProc.
- 3 From the Tools menu select Generate Test.

The DFP Test Properties appears in the lower portion of the Step Worksheet.



Test Properties Fields

DFP Test Properties fields enable you to manage DFP operation. The Result Text area displays messages generated by PTPROG.EXE.

When you click Update DFP, the most recent version of any file in the directory specified by the Source Dir field will overwrite identically named files on both computers.

With DigFuncProc as the Device Type, the only Test Type available is DigFuncProc.

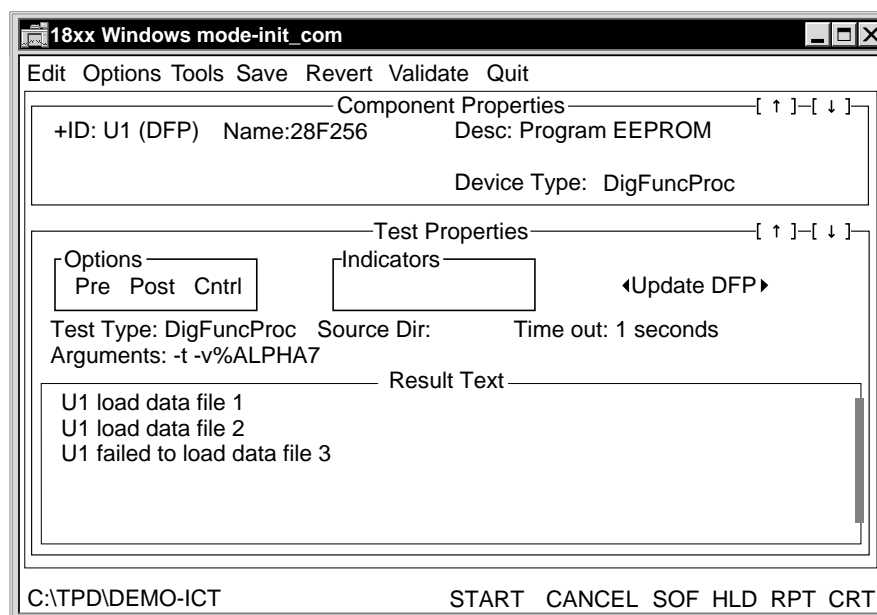
Source Dir refers to the subdirectory of the directory specified in the Header/PRGMVARS. The field takes a DOS directory name of eight characters. This directory contains the source files for the DFP test to be run from this Test Properties page. These files include PTPROG.EXE, PT2.INI, and any other files associated with this DFP test.

Time out enables you to specify the Com port timeout. If there is no activity on the Com port for the specified period of time, an error message is generated, and you will get a test failure.

The Arguments field enables you to send arguments to PTPROG.EXE.

After you have executed the DFP test, resulting messages from PTPROG.EXE appear in the Result Text box.

After your DFP test runs, DFP Test Properties look similar to the following



For detailed information about DFP, see the **Digital Function Processor User's Guide**.

External Synchronization & VP Tests

An external signal connected to the EXT SYNC pin on the VP via the fixture receiver can be used for one of the following functions:

- Replacing the internal VP clock with the external VP clock
- Pausing vector test execution with VP Hold

As the same fixture/receiver pin is used for VP synchronization and VP hold, the two functions are mutually exclusive: VP Hold is available only when the internal VP clock is selected.

IMPORTANT: U44 (P/N 45220) on the VP board has to be Rev. B.0 or later to use VP Hold or VP External Clock.

VP External Clock

VP External Clock allows the 18xx system software to run a vector pattern off the on-board clocks that cannot be overdriven by the tester. In this case, the EXT SYNC pin is edge sensitive.

To use this function, first, go to the vector Step Worksheet and set the Clock to External in the Test_Config menu or put a Clock External statement into the VP ASCII template file.

The VP External Clock is used to generate stim clocks for Vectors. Measure clocks are generated after a Delay after the stim clock, in the same way as when the Internal clock is used.

The falling edge of the VP External Clock signal on the EXT SYNC pin generates the stim clock.

Synchronization with an External Clock does not mean vector states will change at every clock edge. The serial nature of the VP still holds true. States will change in response to external clock edges, but some states will take multiple clock cycles. The external clock merely replaces the Internal VP clock as the synchronization source for vector tests. To get each vector state change at every external clock edge, the frequency of the external clock must be slower than the longest time required for priming a vector state in the vector set.

Hardware Specifications

Refer to the fixture receiver wiring diagram for your system:

21D48010	1888-1, 1880-1, 1808-1, 1805-1, 1803-1,
21D46816	1890-2, 1880-2, 1808-2, 1805-2, 1802-2
21D48031	1888-2
21E48493	1884
21D48213	1866
21D46306	1860
21D46006	1840
21D45372	1820
21D45257	1800

The VP connects its VP External clock/VP Hold signal to the fixture receiver pin labeled EXT SYN 2. This signal is an input to the vector processor. The VP also provides three output signals that appear at the fixture receiver interface:

- VP signal BUFSTMCLK* (the stimulus clock) appears at EXT SYN 1,
- VP signal BUFSTMSYNC* (the free-running internal VP clock) appears at EXT CLK 2,
- VP signal BUFMCLK* (the measurement clock) appears at EXT CLK 1.

All of the outputs from the VP are TTL compatible signals.

The incoming VP sync/hold signal must also be TTL compatible.

The signal is pulled high if it is not driven.

Timing Specifications for VP External Clock Signal

Minimum Pulse Width*: 50 ns

Maximum Frequency**: 10 MHz

Delay from VP sync falling edge to Stim_clk† : 50 ns

(Assuming all VP Primes are done)

*Minimum pulse width means the minimum amount of time the signal must be high in order for the VP to reliably recognize the signal's rising edge and the minimum of time the signal must be low in order for the VP to reliably recognize the signal's falling edge.

**Although the VP can synchronize to a signal up to 10 MHz, in practice it should run at a speed not faster than the VP to process any single vector state (minimum is 350 ns).

† The measurement delay must be less than or equal to the external clock period or measurement will not occur.

VP Hold

The VP Hold feature is available whenever the Internal VP clock is used. In this case, the EXT SYNC pin is level sensitive.

The EXT SYNC pin is normally pulled high. If it is driven low, then the Internal VP clock will not generate stim-measure clocks until it is again brought high.

Timing Specifications

Minimum Pulse Width*: 50 ns
 Hold Time after Stim_sync : None
 Setup Time before Stim_sync : 50 ns

Alternatively:

Delay from Stim_Clock active to VP Hold active when default clock divisor (20) is used:
 Max = 450 ns, Min = 0 Ns

*Minimum pulse width means the minimum amount of time the signal must be high in order for the VP to reliably recognize the signal's rising edge and the minimum of time the signal must be low in order for the VP to reliably recognize the signal's falling edge.

Multipanel Testing

Multipanel is a “push-button” solution for testing several boards in a multipanel assembly. When you select Multipanel from the Files menu, a series of windows leads you through the process of creating a multipanel program consisting of master program and its subprograms.

The process involves

- Selecting a debugged source panel program
- Entering parameters in Master Program Data
- Entering parameters in Panel Program Data
- Verifying parameters in other panels as necessary
- Optionally modifying PANEL.DAT, the exceptions file
- Initiating the building of the multipanel programs

For information about using Multipanel for copying a program for uses other than multipanel testing, see the section, Copying a Program Using Multipanel in Chapter 1, “Introduction to System Software.”

Entering Master Program Data

Start with a thoroughly debugged program as your source. Then fill in the Master Program Data and Panel Program Data sections of the worksheet.

To setup the master program data

- 1 Select Multipanel from the Files menu.

The Multipanel Copy window appears with the name of the current program already in **Program to copy as panel one**. (This field is not editable.)

Multipanel Copy

Program to copy as panel one: DEMO_172

Master Program Data:

Make Master Program : Yes

Master Program Name : MASTER

Chain from : Header

Make Master Read Only : No

Panel Program Data

Number of panels : 2 (Including source)

Panel NAME Prefix : DEMO_172

Panel Node Offset : 0

Change Report Prefix : No

Add Panel to Bd Desc : No

Board Description : Self-Test Block Assembly (043-179

Exclude Sections : No

Exclude Nodes : No

Make all panels read only: No

OK Cancel

☐ Do you want to create a master program?

The default for the Make Master Program is Yes.

Select No from the pop-up in the Make Master Program field if you do not want to make a Master Program. For example, you may merely want a copy of an old program to use as a base for creating a test on a new board.

- 2 Type in a name for the Master Program in the Master Program name field.

The new program will be at a directory level parallel to your source panel (used as panel #1).

- 3 Select Header or Trailer from the Chain from field pop-up to specify where the program copies are to chain from.

The default is Header. Calls are placed at the end of the Header or the beginning of the Trailer depending on which is chosen.

- 4 If you want to make the Master Program Read Only, select Yes from the pop-up.

The default is No.

Entering Panel Program Data

The panel parameters entered in the Panel Program Data area on Multipanel Copy setup screen are used to initialize the data for all subsequent panel screens. After you have filled in all of the panel data fields and selected OK, the code creates data structures for the number of panels specified in the Number of Panels field.

- 1 Enter the number of panels for your program, including the source program, in the Number of Panels field.

If one or fewer panels are specified, only a master program is made. If you enter 2 (or more up to 300), new panels are created from your original. Calls to your source panel and new panels are automatically entered into your master program.

2 Enter the panel name prefix.

The name entered in the Panel Name Prefix field is copied to each panel with the actual panel number appended to it. Note, the first panel program is the program you selected when you chose Multipanel from the Files menu; therefore, the name of the first panel displayed in the Program Copy setup screen will be the panel 2 of N where N is the Number of Panels.

3 Enter a value in the Node Offset field to enable the Exclude Section and Exclude Nodes fields.

The value entered in the Panel Node Offset indicates the node offset between panels. The offset added to each node of each panel will be (panel number - 1) x Panel Node Offset.

The data in the rest of the fields in the Panel Program Data section is simply copied to each panel.

4 To include the board description macro in the report prefix, select Yes from Change Report Prefix.

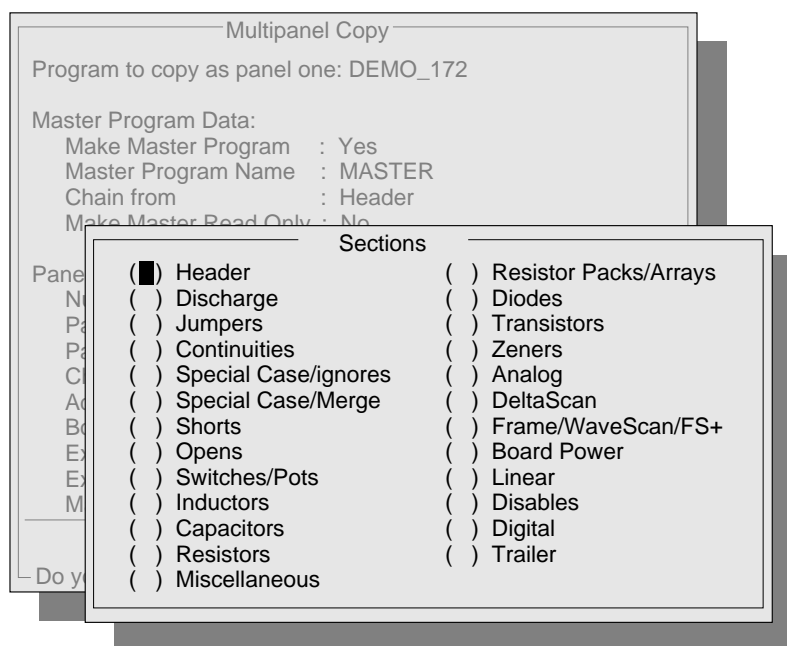
Selecting Yes from the pop-up window creates a new Report Prefix string for the new program that includes the %BOARD macro.

5 To prepend the panel name to the board description string, select Yes from Add Panel to Bd Desc.**6** Enter a board description for each panel.

The board description is a string that can be embedded in any other string using the %BOARD macro. This function is useful in tailoring failure messages for different copies of a program. The %BOARD can be included in the Report Prefix string by using the field above or by adding it to a board failure message in the trailer.

7 To exclude node offsets from being added to particular sections of the panel program, select Exclude sections.

The Sections selection window appears listing all of the test program sections.



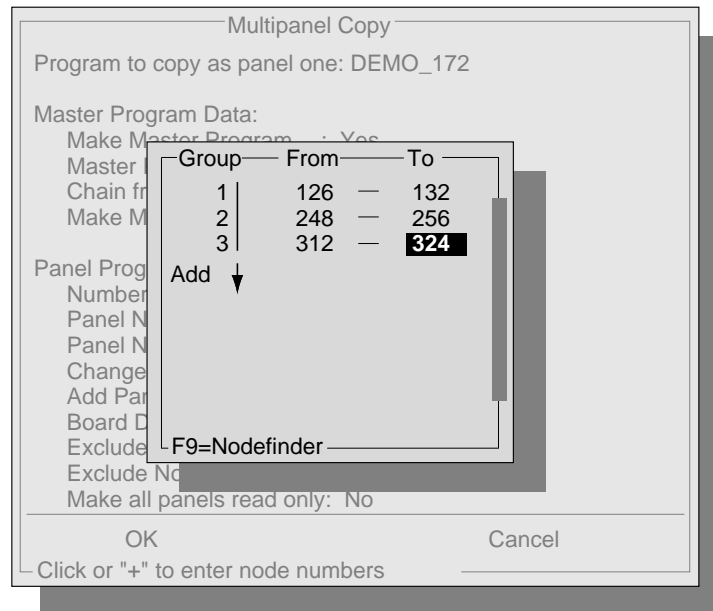
8 Click the sections to exclude.

This feature is useful, for example, if all of the boards use the same power nodes. In such a case you can specify that the Brd_Power section not have the offset added and also indicate any other power or ground nodes to exclude by using Exclude Nodes.

When you have made your selection(s) and return to the Multipanel Copy window, “Yes” appears in the Exclude Sections field.

9 To exclude node offsets from being added to particular nodes of the panel program, select Exclude Nodes.

The Exclude nodes window appears.



10 Specify the desired range of nodes within a group.

- Click “Add” to generate Group 1,
- Change 9999 to the desired numbers or a blank space.

If there is a single node to exclude, just enter that node number in the From field and leave the To blank.

- Press Enter.

For more than one group of nodes, press Enter again. Group 2 appears. Repeat the node entry process.

11 Accept the default No, or select Yes from the pop-up in Make all panels read only.

12 When you have finished setting the Multipanel Copy parameters, click OK (or Cancel to quit Multipanel Copy and return to the Main menu).

Clicking OK copies the parameters in the Panel Program Data section to each panel. This mechanism for initializing panel data makes it easy to create programs with large numbers of panels that all have the same fixed offset and naming convention.

Clicking OK also brings up the first of the Program Copy pages. For example, if you entered the number 3 in the Number of Panels field, Panel 2 of 3 appears.

Verifying Panel Programs

When you have entered a number of panels greater than 1 and have clicked on OK in the Multipanel Copy window, Multipanel presents you with the specified number of panels. It is important to verify the parameters in each program copy.

To verify program copy,

- 1 Examine the displayed parameters.

The name of the source panel appears in the first field. The source panel is always used as subpanel number one.

- 2 Make changes on the individual panel, if necessary.

In each individual panel you can

- change the report prefix,
- change the board description,
- change the node offset,
- specify whether the file for that particular panel is to be Read Only or not.

See the instructions above for entering panel program data.

Program Copy Panel 2 of 2

Source Panel#1 is named: DEMO_172
 Current Panel Name is : **Copy A**

Change Report Prefix: No
 Board Description : Self-Test Block Assembly (043)

Node Offset : 50
 Exclude Sections: Yes
 Exclude Nodes : No
 Make file read Only : No

Next Prev Edit Save Cancel Done

☐ Change prefix to use macro for Board Description?

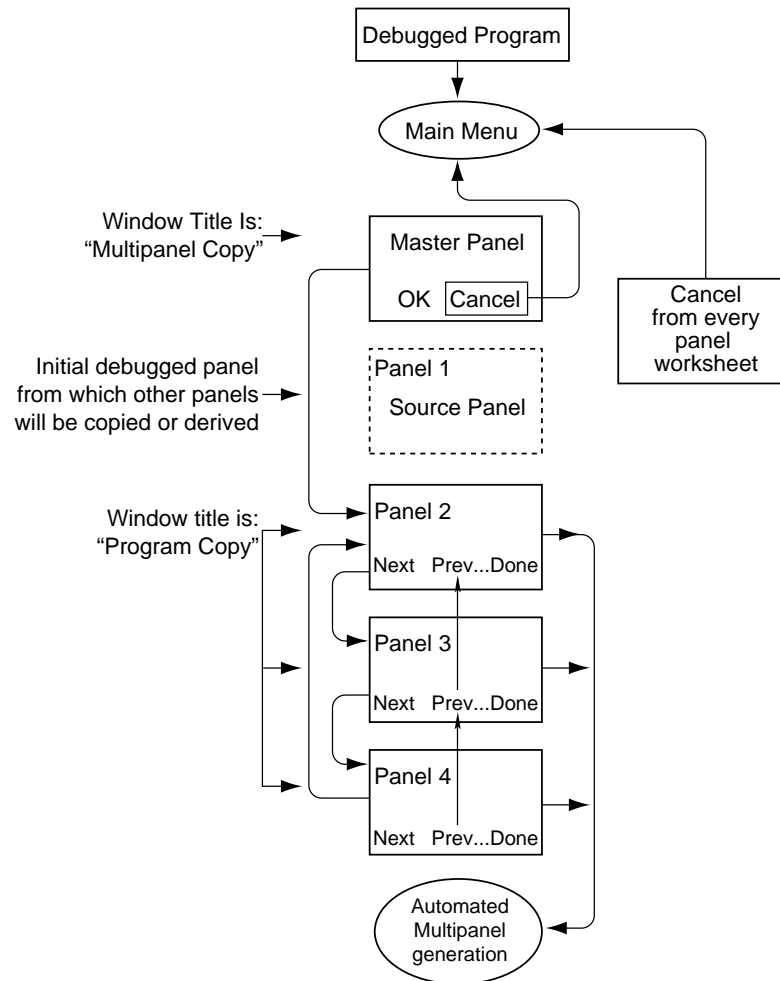
- 3 Select Next to move to the next panel of the series.

Clicking Next advances you through the number of panels you have specified, and when you have reached the end, return you to the second panel again. To access the Master Program from the second panel, select Prev at the bottom of the panel window. For more information see the section on editing the PANEL.DAT file.

- 4 When you have finished filling in the required information for all of the panels, select Done to start the multipanel creation process you have defined.

The Done button instructs the code to create the panel board directories and programs specified by the Multipanel setup screens. After the panel programs have been created, the software gives you the option of saving the panel configuration to the PANEL.CFG file. If you go back to the master panel and change the number of panels, increasing the number of panels appends to the end of the list of panels and does not affect previously edited panel Test Properties. The delete function removes panel Test Properties from the end of the list.

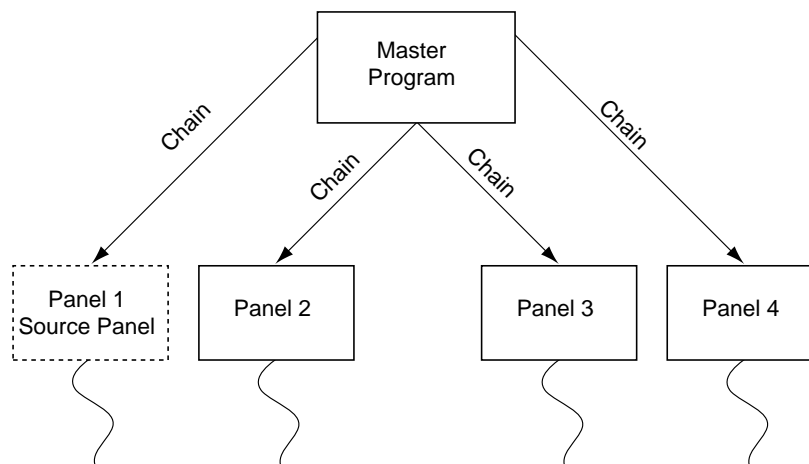
The following diagram represents the development sequence for Multipanel Copy. Note that you must start with a thoroughly debugged program for the first panel.



Following the previous development sequence has two possible results:

- The creation of a master program that chains sequentially through your source panel and all additional panels
- A program copy if Make Master Program in Master Program Data is set to No.

The following diagram illustrates the final program flow of execution for a master program with chained panels.



Multipanel's Program Copy setup screens contain a Save button which saves the entire panel setup, all the data in all of the Program Copy screens plus the data in the Multipanel Copy screen, to a file named PANEL.CFG. This PANEL.CFG file is stored in the source panel's board directory—the currently selected board directory of which all panel programs are copies.

The next time the Multipanel command is selected for this source panel, the PANEL.CFG file is read and the data used to initialize the Multipanel setup screens. The net effect is that the Multipanel setup screens will appear just as they were the last time Multipanel was used.

Editing PANEL.DAT

The PANEL.DAT file is a text file residing in the board directory of the source panel that contains information about nonlinear node numbers and power relay control. To edit this file, click Edit to access the editor of your choice.

IMPORTANT: You can choose the active editor in the Setup/Environment/DOS Editor menu choice.

After you have made your edits and quit from the editor, you are returned to the 18xx programming environment.

The format for PANEL.DAT entries is as follows:

(The pound sign [#] and apostrophe ['] indicate comments.)

```

<panelname> NODE <orignode#> = <newnode#>[;]
#for specific node exceptions
<panel> RELAY [Set|Clear] <relay#> = [Set|Clear] <relay#>

```

Examples

```

pn11 NODE 5 = 205;#any occurrence of node 5
    # becomes 205, overrides node
    #range in panel worksheet
pn12 RELAY SET 1 = 5;#any set of relay 1 becomes 5
pn13 RELAY CLEAR 2 = 6;#any clear of relay 2 becomes 6
pn14 RELAY 3 = 7;#any set or clear of relay 3
    #becomes relay 7

```

Vector Performance Strategy Options

IMPORTANT: If you have an analog-only test system, information regarding digital test is not pertinent to your test situation. The analog-only functionality prevents you from generating, editing, or running digital tests.

A number of test strategies are available with the Vector Performance option. Your strategy will depend upon the inputs available to you, completion schedule, and the coverage requirements.

This section provides guidelines for users who have no in-house process to handle ASIC, PAL, and VLSI test strategies.

You may enter vector patterns into the tester in 4 ways:

1. ASCII format
2. PAL JEDEC format
3. Standard template library, Teradyne-supplied
4. Victory Boundary Scan output

ASCII format:

- ASIC/VLSI patterns from CAE Simulation
- ASIC/VLSI patterns from another tester
- ASIC/VLSI patterns of unknown suitability
- ASIC/VLSI/PAL patterns from functional descriptions
- ASIC/VLSI/PAL patterns from board schematics

PAL JEDEC format:

- PAL patterns from JEDEC test pattern
- PAL patterns from PAL equations

Teradyne-supplied library:

- VLSI device exists in library
- VLSI device model resembles the library model

Victory BSCAN output:

- Boundary Scan patterns from Teradyne Victory toolset

A summary of the option categories, expected coverage, level of effort, and rate of success follows. Subsequent sections discuss the individual categories.

ASCII/VLSI/PAL Strategy Option Summary

Category	Coverage	Effort	Success
1. Fits Lib Model	Highest	Fastest	Highest
2. Resembles Model	High	Fast	High
3. CAE Pattern	Highest	Fastest	Highest
4. From Another Tester	High	Medium	Medium
5. Unknown	Unknown	Slow	Low
6. Functional Desc	High	Slowest	High
7. Schematic Only	Lowest	Fastest	Low
8. PAL JEDEC Pattern	Highest	Fastest	Highest
9. PAL EQN	Highest	Fastest	Highest
10. Boundary Scan Pattern	Highest	Fastest	Highest

VLSI Model Exists in Library

The approach described in this section assumes that the VLSI device under test is available in either the Z1800-series VLSI library distribution or in the user vector snapshot library. The VLSI model library has a target pin level fault coverage of >90% and strives to accommodate at least two hookup constraints. The constraints of the device under test must be covered by the VLSI model. You can determine whether hookup constraints are covered by looking the model up in the **Z1800-Series Template Library Manual** or by reading the Header information in the ASCII version of the specific model. Device models that embody constraint handling as described are referred to as vector templates, and are found in the VLSI library, and use the IPL token VEC.

The other pertinent device model library is referred to as the user's vector snapshot library. These models are “snapshot” images of a single unconstrained version of the device test, developed using the 18xx editor's snapshot tool “Create Template.” The vector snapshot library is a collection of these single configuration device tests a user has developed and uses the IPL token “VIMG.”

The test strategy for devices in this category does not differ from the strategy for developing the test program for SSI digital devices from the IPL. IPL fully specifies the board topology, including constraints and the device model names. The program generator creates the first pass test program, including the test for this VLSI device.

First pass pin-level fault coverage will likely be excellent, with little programming effort, similar to the effort required for an SSI device. The recommended test process is as follows:

- 1 Create Z1800-series VP board-level test.
 - Create Input List, specifying the device model name using the VEC or VIMG token, as appropriate.
 - Generate the test program.
 - Debug board-level test.
- 2 In Vector Edit, Run Tools/Generate Test, as required to save the “snapshot” to the VIMG library. To save your changes, choose from the following techniques.
 - Modify the ASCII vector, which is physically the .ASC file. Modifications can be made directly to this file, and then added back into the library using Template-Add (if new) or Template-Update (if already existing). You would use this technique if you need to have the model modified for future re-use on another board under test. Since the model is updated, it can be reused later.

- Take a snapshot, which takes the “image” of what is currently modified Test Properties and saves it in the library. To reuse this test, you must change the reference to the part in the IPL from VEC to VIMG (hence VIMG library). This is known as the “snapshot” technique. You would use this technique if this test is a “one-time-only” situation, and you do not need to use it again.
- 3 Update the IPL to reflect the new reference to the device test in the VIMG library instead of the VEC library. To save your changes, choose from the following techniques:
- Modify the ASCII vector, which is physically the .ASC file. Modifications can be made directly to this file, and then added back into the library using Template-Add (if new) or Template-Update (if already existing). You would use this technique if you need to have the model modified for future re-use on another board under test. Since the model is updated, it can be reused later.
 - Take a snapshot, which takes the image of what is currently modified in Test Properties and saves it in the library. To reuse this test, you must change the reference to the part in the IPL from VEC to VIMG (hence VIMG library). This is now known as the snapshot technique. You would use this technique if this test is a one-timeonly situation, and you do not need to use it again.

VLSI Model Resembles the Library Model

In this category, the device under test resembles the device model found in either the Teradyne template library or the user's snapshot library.

Resembles in this context means:

- Same device but different constraints, e.g. pins tied high, unused pins, pins tied together, pins tied low.
- Same device but different pinout due to different package.
- Different device, but embodies some functionality of the device in the library, such as the differing types of 68HC11 with and without ADC's.
- Different device, but the handshake is similar, as are some useful functions such as reset, memory write, instruction fetch, or I/O.

Two strategies suggested for this category of device test development:

Strategy 1: If the similarity is high (same device) the existing device model can be used as-is and the resultant test hand modified during debug to learn the new configuration. The modified test can then be saved to the snapshot library for reuse. This strategy requires the least programmer effort and creates a template that handles a specific constraint.

Strategy 2: If the similarity is low (different device) and you want to reuse the model on future boards, the existing device test should serve as a guide for creating a new device test model. The new model is written in the Z1800-series VP ASCII format and added to the library with the Template-Add ASCII Vector function. This strategy provides a new model that handles constraints for future reuse.

First-pass pin-level fault coverage is likely to be high, as is the likelihood of success. The amount of effort required is a function of how close the device-under-test is and the inherent complexity of the device.

Strategy 1—Use the Existing Model As-Is

The recommended development strategy is as follows:

- 1 Create Z1800-series VP board-level test.
 - Create Input List, specifying the “close” device model name using the VEC or VIMG token as appropriate.
 - Generate the test program.
 - Debug board-level test.
- 2 In Edit, change the device type to “Vector Image” and save the snapshot using Tools/Create Template. You may also want to modify the device name before saving, in order to distinguish it from the “parent” model.
- 3 Save the Step Worksheet changes and update the IPL database to reflect the new reference to the device. If you expect to use your initial input list (IPL.DAT) again, modify it to reflect any changes in the token type or device name.

Strategy 2—Create a New Model from an Existing Model

The recommended development strategy is as follows:

- 1 Edit source listing of the “close” device model (DEVICE_NAME.ASC), creating a new user device model library element in ASCII format.
Name the new ASCII vector format NEW_DEVICENAME.ASC, where NEW_DEVICENAME is the actual device name.
- 2 Run Template-Add/Add ASCII Vector, adding NEW_DEVICENAME.ASC to the VLSI library.
- 3 Create Z1800-series VP single device test.
 - Create Input List, specifying the device program library model
 - Create your IPL according to normal practices for any other component using ChipChecker* (a fixture or jig of your choice to hold a single device).
 - Run Pgen.
- 4 Debug device test on ChipChecker, modifying NEW_DEVICENAME.ASC as required.
- 5 Run Template-Add/Add ASCII Vector on NEW_DEVICENAME.ASC, to store new model in library
- 6 Create Z1800-series VP board-level test
 - Create Input List, specifying the device model name in the library
 - Run Pgen
 - Debug board-level test

*A ChipChecker is a Teradyne product that interfaces a single device to the Z1800-Series VP tester. It is basically a fixture designed to accommodate a single device. There are a number of device adapters to accommodate differing package types (i.e., DIP, pin grid array, SMD, etc.)

The ChipChecker is sold out of Teradyne Programming Services in Richardson, Texas. Please contact your Teradyne salesman for pricing and delivery information.

ASIC/VLSI Patterns from CAE Simulation

This category is the “ideal” scenario for the development of ASIC test patterns for in-circuit use on the Z1800-series VP, with the best possible outcome (highest pin level fault coverage) and the minimal investment of labor and the shortest schedule. The outcome, however, is largely influenced by the engineer's ability to adequately use the CAE system to produce test patterns ideally suited to the Z1800-series VP. With the TSSI TDS* tool, the following CAE systems can be accommodated (IN converters supported);

Vendor	Simulator	Files Used
HHB	CADAT	.DPO, .DSL
GenRad	HILO	HILO.CAP
Teradyne	LASAR	EVENT.TAP, PINAMES.TAP, PONAMES.TAP
Daisy	Logician	VALIF
Mentor	Quicksim	LOG
SimuCad	Silos	OUTPUT
Calma	TEGAS	SAVEFILE
Calma	TEXOUT	SAVEFILE NAMETREE
Calma	TESSIM	SAVEFILE NAMETREE
Valid	ValidSIM	TABULAR I/O OR PRIMARY I/O
ZyCad	ZILOS	EVENT OR .TABLE
HP EDS	HILO	.DPO, .DSL
Gateway	Verilog	TSSI ASCII
Lattice	Logic Expert	TSSI ASCII
Logic Model Sys		TSSI ASCII
Silicon Comp Sys	LSIM	TSSI ASCII
Silvar-Lisco Logix	Helix	TSSI ASCII
Tektronix CAE Sys		TSSI ASCII
Viewlogic	ViewSim	TSSI ASCII

*TDS is a required accessory product for your Z1800-series VP. TDS is required to make ASCII vector translation from CAE possible.

First pass pin level fault coverage will likely be excellent. The effort required is a function of the number of pins and inherent complexity of the device under test. The typical range of effort in this category is from 4 to 32 hours.

The recommended development strategy is as follows:

- 1** Transfer CAE simulation files to TDS host.
- 2** Run TSSI's TDS.
 - Run IN converter for appropriate CAE format
 - Run SIMPLIFY conditioner
 - Run ASCII OUT converter
- 3** Transfer TSSI ASCII vector files to Z1800-series VP.
- 4** From the DOS shell, run TDS18XX, creating a device model in the VLSI library.
- 5** Create Z1800-series VP single device test program.
 - Create Input List, specifying the device model name. Create your IPL according to normal practices for any other component (if using Teradyne ChipChecker).
 - Run Pgen
- 6** Debug device test on ChipChecker.
- 7** Create Z1800-series VP board-level test.
 - Create Input List, specifying the device model name using the VIMG or VEC token
 - Run Pgen
- 8** Debug board-level test.

ASIC/VLSI Patterns from Another Tester

In this category, the available test patterns were originally targeted for another component or in-circuit tester. As such, they may have capabilities nonexistent on the Z1800-series VP, such as pattern timing requirements, minimum pulse widths, formatting, timesets, phases, windows, Clocks-Per-Pattern, looping, subroutining, branching, pattern depth, or terminations. However, when the test pattern for the “other” tester utilizes what is available on the Z1800-series VP, pattern translation is probable.

“Other” testers in this context may include:

- Teradyne L2,L3 in-circuit testers
- Teradyne Z8 in-circuit testers
- GenRad 22XX in-circuit testers
- Factron 3XX in-circuit testers
- Teradyne J941, J971 component testers
- Sentry 20 component testers

First-pass pin-level fault coverage is typically high, with coverage fallout usually due to the test requiring functionality not available on the Z1800-series VP.

The effort required in this process depends on how much functionality is embodied into the “other” test vector that must be condensed to fit onto the Z1800-series VP. The likelihood of success varies from poor to very good.

Teradyne L2, L3 In-Circuit Testers

The key functions of the L2 or L3 in-circuit testers that must be addressed in translating device test patterns to the Z1800-series VP include:

- formats—NRET only
- stimulus phases—single
- measurement windows—limited
- terminations—limited selection
- pattern rate—variable timing
- looping—none
- clocks per pattern—CPP = 0.5

The likelihood of success is high, with the exception of the PATC loop, which is not translatable to the Z1800-series VP.

Teradyne provides translation as a turn-key service for our customers. Contact your Teradyne representative for more information.

Teradyne Z8000-Series In-Circuit Tester: Protocol Test

Key Z8000-series in-circuit tester functions to deal with in translating device test patterns to the Z1800-series VP include:

- formats—NRET only
- timesets—single
- timing generators—single
- pattern rate—variable timing
- external clock—none
- looping—none

The likelihood of success is good. Missing external clock functions can prevent successful program translation.

Teradyne provides translation as a turn-key service for our customers. Contact your Teradyne representative for more information.

Teradyne Z8000-Series Vector Test

Teradyne provides translation as a turn-key service for our customers. Contact your Teradyne representative for more information.

GENRAD 22XX In-Circuit Testers

Key GR22XX in-circuit tester functions to deal with in translating device test patterns to the Z1800-series VP include:

- terminations—limited selection
- pattern rate—variable timing
- looping—none
- clock pins—none

The fault coverage is typically very close to that of the original Genrad program. The likelihood of success is very good. The main exception is that looping is not translatable.

The recommended development strategy is as follows:

- 1 Transfer GR test pattern to Z1800-series VP.
- 2 Run GRZVEC on Z1800-series VP. (GRZVEC is a translation tool to get vector tests from a Genrad to 1800-series system software. GRZVEC is provided free upon request.)
- 3 Run Template/Add/Add ASCII Vector, adding model to the VEC library.
- 4 Create Z1800-series VP single device test program.
 - Create Input List, specifying the device model name in the VEC library. Create your IPL according to normal practices for any other component (if using Teradyne ChipChecker).
 - Run Pgen
- 5 Debug the device test on ChipChecker.
- 6 In Edit, change the device type to Vector Image and save the snapshot using Tools/Create Template.

You may also want to modify the device name before saving, in order to distinguish it from the “parent” model.
- 7 Create Z1800-series VP board-level test using GRZIPL.

(GRZIPL, which is provided free upon request, is a tool to get GR input files to the INPUTLIST format.)

 - Create Input List, specifying the device model name in the VEC or VIMG library as appropriate
 - Run Pgen
- 8 Debug board-level test.

Factron 3XX In-Circuit Testers

The key functions of the Factron 3XX in-circuit testers that must be dealt with in translating device test patterns to the Z1800-series VP include:

- terminations—limited selection
- pattern rate—variable timing

The fault coverage is typically very close to that of the original Factron test. The likelihood of success is very good.

The recommended development strategy is as follows:

- 1 Transfer Factron test pattern to Z1800-series VP.
- 2 Manually edit the test pattern into the Z1800-series VP ASCII format.
- 3 Run Template/Add/Add ASCII Vector, adding model to the VEC library.
- 4 Create Z1800-series VP single device test program.
 - Create Input List, specifying the device model name in the VEC library. Create your IPL according to normal practices for any other component (if using Teradyne ChipChecker).
 - Run Pgen
- 5 Debug the device test on a ChipChecker.
- 6 In Edit, change the device type to Vector Image and save the snapshot using Tools/ Create Template. You may also want to modify the device name before saving, in order to distinguish it from the “parent” model.
- 7 Create Z1800-series VP board-level test program.
 - Create Input List, specifying the device model name in the VEC or VIMG library as appropriate.
 - Run Pgen
- 8 Debug the board-level test.

Teradyne J941, J971 Component Testers

The key functions of the J941 and J971 component testers that must be dealt with in translating device test patterns to the Z1800-series VP include:

- pattern depth limit of approximately 100K patterns.
- formats—NRET only
- stimulus phases—single
- measurement windows—limited
- terminations—limited selection
- pattern rate—variable timing
- looping—none

The likelihood of success is typically poor to average. The component test will likely be very large and use much of the timing capability of the component tester. As a result, huge pattern expansions often result from the “flattening” of formats and timesets.

The recommended development strategy is as follows:

- 1 Transfer J941 or J971 files to TDS host.
- 2 Run TSSI's TDS.
 - Run J941 or J971 IN converter
 - Run SIMPLIFY conditioner
 - Run ASCII OUT converter
- 3 Transfer ASCII vector files to Z1800-series VP.
- 4 From the DOS shell, run TDS18XX, creating a device model in the VEC library.
- 5 Create Z1800-series VP single device test.
 - Create input list, specifying the device model name in the VEC library. Create your IPL according to normal practices for any other component (if using Teradyne ChipChecker).
 - Run Pgen

- 6 Debug device test on ChipChecker.
- 7 Modify the ASCII input.
- 8 Create Z1800-series VP board-level test.
 - Create input list, specifying the device model name in the VIMG or VEC library as appropriate.
 - Run Pgen
- 9 Debug the board-level test.

Sentry 20 Component Testers

The key functions of the Sentry 20 component testers that must be dealt with in translating device test patterns to the Z1800-series VP include:

- pattern depth limit of approximately 100K patterns.
- formats—NRET only
- timing generator per pin—single
- terminations—limited selection
- pattern rate—variable timing
- looping—none
- subroutines—none

The likelihood of success is similar to the J941, J971 case. The recommended development strategy is as follows:

- 1 Transfer Sentry files to TDS host.
- 2 Run TSSI's TDS.
 - Run J941 or J971 IN converter
 - Run SIMPLIFY conditioner
 - Run ASCII OUT converter
- 3 Transfer ASCII vector files to Z1800-series VP.
- 4 From the DOS shell, run TDS18XX, creating a device model in the VEC library.
- 5 Create Z1800-series VP single device test program.
 - Create input list, specifying the device model name in the VEC library. Create your IPL according to normal practices for any other component (if using Teradyne ChipChecker).
 - Run Pgen
- 6 Debug device test on ChipChecker.
- 7 Edit ASCII and add again.
- 8 Create Z1800-series VP board-level test program.
 - Create input list, specifying the device model name in the VEC or VIMG library as appropriate
 - Run Pgen
- 9 Debug the board-level test.

**ASIC/VLSI/PAL
Functional
Descriptions Available**

In this category, create a new device test model from “scratch.” The programmer must understand the functionality of the device-under-test and generate the test pattern for high fault coverage.

First-pass pin-level fault coverage is typically high.

The recommended development strategy is as follows:

- 1 Design RESET/INITIALIZE test patterns in the DEVICE_NAME.ASC file.
 - Identify RESET pin, and pulse appropriately
 - Identify RESET program sequence
 - Create bus cycle for instruction fetch
 - Create pattern for RESET sequence
- 2 Design MEMORY READ/WRITE test patterns as required.
 - Create bus cycle for memory WRITE
 - Create data to WRITE to memory
 - Create bus cycle for memory READ
 - Create data to READ from memory
- 3 Design INPUT/OUTPUT bus test patterns as required.
 - Create bus cycle for INPUT function
 - Create data to INPUT
 - Create bus cycle for OUTPUT function
 - Create data for OUTPUT
- 4 Design INTERRUPT test patterns as required.
 - Create bus cycle for INTERRUPT function
 - Create instructions for device interrupt handling
 - Create data for INTERRUPT test
- 5 Design test patterns for OTHER device functions as required.
 - Create bus cycle for OTHER function
 - Create instructions for device OTHER function
 - Create data for OTHER function
- 6 Run Template/Add/Add ASCII Vector on the DEVICE_NAME.ASC file, adding the device model to the VEC library.
- 7 Create Z1800-series VP single device test program.
 - Create input list, specifying the device model name in the VEC library
 - Run Pgen
- 8 Debug the device test on ChipChecker.
- 9 Save the debugged snapshot or modify original model.
 - Edit original DEVICE_NAME.ASC file as appropriate and run Template/Add/Add ASCII Vector to create a device model in the VEC library
- 10 Create Z1800-series VP board-level test.
 - Create input list, specifying the device model name in the VEC or VIMG library as appropriate
 - Run Pgen
- 11 Debug the board-level test.

ASIC/VLSI/PAL Board Schematic Only

For this category, the functionality of the device under test is only known to a “black box” level. With that, the probable test will cover the basic function of reset. Reset is an important test for downstream devices, and the minimum test recommended for any device on the board.

In addition to the Reset function test, Gray codes at the device may yield a unique “presence” test, providing a degree (although unpredictable) of additional pin level fault coverage over the Reset. The most difficult part of this test is determining when outputs are valid (bidirectional lines). The test works best on devices with fixed inputs and outputs.

First-pass pin-level fault coverage is typically very low. The range of effort is 2 to 24 hours. The upper bound is reached as the limited combinations of gray codes are exhausted by trial and error. The likelihood of success is high for Reset only and low for additional Gray code coverage.

The Reset Test

The recommended development strategy is as follows:

- 1 Edit RESET/INITIALIZE test patterns in DEVICE_NAME.ASC file. Identify RESET pin, and pulse it appropriately.
- 2 Run Template/Add/Add ASCII Vector, adding model to the VEC library.
- 3 Create Z1800-series VP single device test program.
 - Create input list, specifying the device model name in the VEC library
 - Run Pgen
- 4 Debug the device test on ChipChecker.
- 5 Modify ASCII input.
- 6 Create the Z1800-series VP board-level test.
 - Create input list, specifying the device model name in the VEC or VIMG library as appropriate
 - Run Pgen
- 7 Debug the board-level test.

The Gray Code Test

The recommended development strategy is as follows:

- 1 Create first-pass board test without the device, running Pgen.
- 2 Using the Gray code test editor, create and execute a new test.
- 3 Apply Gray codes to all the inputs.
 - Apply the fastest frequencies (F1,F2) to the control strobes
 - Apply the next fastest frequencies (F3,F4,F5) to the data lines
 - Apply the slowest frequencies (F13,F14) to the address lines
 - Apply static logic levels for the static control lines
- 4 Measure the outputs.
 - Select a measurement window (WHILE, for example) appropriate for the valid output pin cycles
 - Use CRCs when the result data is stable and deterministic
 - Use COUNT or HIGH when the result data is asynchronous
- 5 In Edit/Digital, save the Step Worksheet. If you want to make a template, “Create Template” is required.

PAL JEDEC Test Pattern Available

In this category, the PAL test pattern is available in the JEDEC format generated on a PAL development station such as “ABEL”, DATA I/O's PAL tool or Accugen. Remember to consider board constraints when developing the test patterns.

First-pass pin-level fault coverage is excellent. The effort required is very low. The likelihood of success is excellent.

The recommended development strategy is as follows:

- 1 Transfer JEDEC files to Z1800-series VP.
- 2 From the Templates/Jedec menus, run PALVEC to create a device model .ASC file in the Z1800-series VP ASCII format. Alternatively, you can create a device model .ASC file with Jed18xx from the DOS shell.
- 3 Run Template/Add/Add ASCII Vector, adding the model to the VEC library.
- 4 Create Z1800-series VP board-level test program.
 - Create input list, specifying the device model name in the VEC library
 - Run Pgen
- 5 Debug the board-level test.
- 6 Edit ASCII.

PAL Equation Available

In this category, the PAL equation is provided. Equations are in Boolean equation form, with documentation on the PAL type. Accugen is the recommended tool to generate PAL test patterns for equations.

First-pass pin-level fault coverage is excellent. The range of efforts is 2 to 8 hours. The likelihood of success is excellent.

Accugen

The recommended development strategy is as follows:

- 1 Enter PAL equations into Accugen.
- 2 Run Accugen automated pattern generator.
- 3 Transfer JEDEC test vector file to Z1800-series VP.
- 4 From the Templates/Jedec menus, run PALVEC to create a device model .ASC file in the Z1800-series VP ASCII format. Alternatively, you can create a device model .ASC file with Jed18xx from the DOS shell.
- 5 Run Template/Add/Add ASCII Vector, adding model to the VEC library.
- 6 Create the Z1800-series VP board-level test.
 - Create input list, specifying the device model name in the VEC library
 - Run Pgen
- 7 Debug the board-level test.
- 8 Edit ASCII.

LASAR Circuitbreaker

Teradyne provides translation as a turn-key service for our customers. Contact your Teradyne representative for more information.

Z8000-Series APG

Teradyne provides translation as a turn-key service for our customers. Contact your Teradyne representative for more information.

Victory Boundary Scan Test Vector

The Teradyne Victory Boundary Scan tools take Boundary Scan Description Language (BSDL) information to produce IEEE 1149.1 Boundary Scan tests for Teradyne testers, including the Z1800-series VP.

The Victory tools produce three types of tests: Boundary In-Circuit Tests (BICT), Virtual Interconnect Tests (VIT), and Test Access Port Interconnect Tests (TAPIT). Each of these tests for the Z1800-series VP are embodied in the ASCII Incremental Vector Language (IVL) format.

The .IVL tests created from the Victory tools are placed in the VIMG library on the Z1800-series VP. Tests are treated similarly to a single device model of the VIMG form. Board level tests including these Victory Boundary Scan tests are generated by referring to the VIMG tests, similar to the way digital vector tests are included.

First-pass pin-level fault coverage is 100%, as provided by the IEEE 1149.1 Boundary Scan test technique. The effort required ranges from 2 to 8 hours, depending on the BSCAN device and the board's complexity. The likelihood of success is excellent.

The recommended development strategy is as follows:

- 1 Enter BSDL into Victory toolset. (Victory toolset is a Teradyne product.)
- 2 Run Victory toolset, creating the Z1800-series VP IVL files BICT.IVL, VIT.IVL, and TAPIT.IVL, as required.
- 3 Transfer IVL files to Z1800-series VP.
- 4 Run Template/Add/Add IVL Template, adding model to the VIMG library.
- 5 Create Z1800-series VP board-level test.
 - Create input list, specifying the device model name in the VIMG library
 - Run Pgen
- 6 Debug board-level test.
- 7 In Vector Edit, Run Tools/Template Test, as required, saving the snapshot to the VIMG library.

ASIC Test Pattern Design Criteria

The following are guidelines for designing test patterns for the Vector Processor.

Disable Requirements Output Tristate

The optimum disable vector will place all of a device's outputs into a tristate condition so that downstream devices can be driven without any overdrive requirements. The possibility of overdrive damage to the ASIC is increased if disable vectors are not used and large numbers of overdriving nodes test the downstream device. The predominant failure mechanism is excessive current in the ground bond wire(s), caused by the sum of all of the overdriven outputs routing through the ground bond wire(s). As the pin count on ASICs grows, the likelihood of bond wire overcurrent failure increases. Thus the use of disables for downstream device tests is a very important precautionary programming practice. In some devices, the output disable is controlled by a single input, making the programmer's job easy. Usually, ASIC disabling requires a vector pattern.

Preferred Overdrive State

When it is not possible to put all of the ASIC outputs in the tristate mode, the next best alternative is to put them into the preferred overdrive state. The preferred overdrive state is that which requires the least amount of overdrive current to overcome. For TTL logic, this is logic high.

Test Pattern Restrictions

Pattern Size

The maximum number of vector patterns (rows) is 100,000. The maximum number of vector channels (columns) is 2048 on the Z1860/20 and 640 on the Z1840/00. Small test pattern size is important to reduce debug effort and provide the highest test throughput.

Formats

Non-delayed non-return (NRET) is the standard format. For situations involving Clocks Per Pattern (L3 programs), the Z1800-series VP is restricted to CPP=0.5. The current set of vector translation tools can handle either linearized (truth table layout with columns = pins and rows = cycles) or incremental change formats (L3 or Victory Boundary Scan formats with time and events). The current set of tools cannot accommodate hardware subroutines such as Sentry 50 subroutines, for example. The current set of tools (i.e. TSSI) “flatten” formats such as RZ, RC, R1, or R0 into expanded vector cycles. For example, an H cycle on an R0 format gets “flattened” into 3 cycles: L, H, L. The TSSI TDS can “flatten” with functions like SNAP, WINDOW, and SIMPLIFY.

TSSI TDS is an externally purchased companion product to your tester.

Minimum Pulse Width

The minimum pulse width is 500 ns. There is no software upper limit, but a hardware limit exists in the overdrive timer.

Measurement Timing

The measurement is level triggered, where the measured signal must remain valid for the duration of the measurement cycle. The measurement cycle is programmable from a minimum of 100 ns after the beginning of the cycle to the end of the cycle. Relay is programmable in 25 ns increments. The default measurement timing is a delay of 400 ns from the beginning of the cycle to the end of the cycle.

Non-Synchronous Clocking

The timing between pattern cycles on the Z1800-series VP will vary with the number of channel transitions during that cycle. The fastest pattern cycle (500 ns) occurs when two or fewer channel transitions occur on any of the channels during a single pattern cycle. The pattern cycle rate with the clock rate set to 2 MHz is computed by the following equation:

$$\text{Cycle period} = [(\# \text{ of pin changes} - 2) \times 150\text{ns}] + 500\text{ns}$$

The cycle period is rounded up to the nearest 500 ns interval. The pin changes for that cycle are any stimuli or expect state changes.

Synchronous Clocking

The Z1800-series VP can be run in a synchronous clocking mode, whereby all of the pattern cycle intervals are equal. Synchronous clocking mode is possible by selecting a clock speed for which all transitions in all patterns can be achieved in a single cycle. In other words, you are slowing the rate down to the slowest cycle.

Tristate Measurement

The tristate test is recommended as an independent test outside of the main test vector pattern. The Z1800-series VP detection capability provides the states of logic high (above the high threshold), logic low (below the low threshold), and between state (above the low threshold and below the high threshold) which results in a threshold error. The threshold error will force the test burst to fail. Hence, any known between state conditions must either not be measured (expect set to X) or else terminated to a valid logic high or logic low.

Pattern Coverage Guidelines

Board Hookup Constraints

Board hookup constraints include

1. pins tied to fixed levels (Vcc or Gnd, for example) that cannot be overdriven to other logic levels.
2. pins tied together such that they cannot be driven to opposing logic levels.
3. unused pins where there is no tester access.
4. bidirectional I/O pins that require terminations.

The test pattern must take these constraints into account at the time the patterns are developed. Post-processing of the original test pattern to correct these oversights is often time consuming.

Pin Coverage

A test pattern's first goal regarding in-circuit test is to verify correct assembly of the device on the board by toggling each input and output. Such pin-level fault coverage (100%) means that the test pattern must toggle all inputs and make sure that they result in corresponding changes to specific outputs. The ultimate goal is to achieve the shortest possible pattern depth. Minimal pattern depth translates into ease of test development, minimal test development/debug time, and maximum test throughput. Minimal pattern depth is not necessarily the goal of device testers which have different functional coverage goals. Often, simulations performed by designers focus on the functionality of the part leading to large numbers of test pattern bursts, large pattern depths, and less than 100% pin-level fault coverage. 100% pin-level fault coverage must be proactively embodied into the test pattern development process for in-circuit testers such as the Z1800-series testers. Post-processing of test vectors to improve the pin-level fault coverage is a complex endeavor.

Self Initialization

Each test pattern must be self initializing, that is, the test burst must not require an outside pattern or mechanism to achieve the initial condition. Without self initialization, the test pattern will not repeat reliably in production testing.

Developing an ASIC Test

This section explains the process of developing an ASIC vector test from simulation to working in-circuit board test program.

Overview

This process is based upon a custom ASIC used as a parallel printer port interface for a PC-compatible motherboard, a 40-pin device using standard cell logic. The simulations were done on a Verilog system.

The patterns developed for the Z1800-series VP are intended to

1. provide disable information for the device under test such that downstream devices are testable.
2. represent the constraints posed by the board under test's circuit configuration.
3. adapt the device test to the tester's timing.
4. achieve 100% pin-level fault coverage.
5. produce patterns to simplify debug and maximize test throughput.

ASIC Test Development Process

The process for developing ASIC test patterns for the Z1800-series VP from CAE simulation data should be documented into a formal process. With practice and fine tuning, total process time can be reduced.

The recommended ASIC test development process is as follows:

1. Simulate.
2. Transfer CAE simulation files to TDS host.
3. Run TSSI'S TDS.
4. Transfer ASCII vector files to Z1800-series VP.
5. Run TDS Conversion.
6. Create Z1800-series VP single device test.
7. Debug device test on chip checker.
8. Create Z1800-series VP board-level test.
9. Debug board-level test.

Step 1. Simulate

The process begins with device test CAE simulation embodied with the requirements of board test. The simulation requirements are contained in the ASIC Design Criteria section.

The goal of simulation is to create a test vector (and disable vector if required) that achieves 100% pin-level fault coverage in the most efficient way possible.

One must determine the quality of the ASIC test at this point, since improving upon test pattern coverage or efficiency is very difficult if not impossible at later stages. Determining quality is typically an interdepartmental effort involving both design and test engineering.

Simulation itself works best when implemented by well-understood and repeatable processes; it will not run smoothly if run on an exception basis. Thus, the initial setup of the design-to-test interface process that guarantees accurate, complete, and quality test patterns is of primary importance.

The output of Step 1 is a CAE simulation file that contains the test pattern. The test pattern must achieve the desired goal for test coverage. A disable simulation file may also be appropriate. Often the disable vector is a simple static vector created manually as was the case with this process.

Step 2. Transfer CAE simulation files to TDS host

The TDS (Test Development Series) by TSSI (Test System Strategies, Inc.) is a set of software tools used to convert simulation data from various sources into an internal data representation (SEF format). The SEF format is then processed into test vector patterns for various target testers.

TSSI offers a set of tools that are used to fit the test pattern to the target tester as well. For the Z1800-series VP conversion process, the minimum requirement is the TDS MiniMaster configuration. This configuration includes a single simulation IN converter, ASCII IN converter, ASCII OUT converter, all utilities, and a limited set of conditioners. The TDS Master goes beyond the Z1800-series VP requirements and can produce test patterns with timing considerations taken into account that produce test vectors and timing sets for testers like the Teradyne L3. The TDS Master configuration adds the full set of conditioners, simulation rules checkers, and tester output generation (P Bridge).

Design test patterns, disable patterns, and device documentation are delivered to test engineering when complete.

Step 3. Run TSSI'S TDS

Running the TSSI's TDS tool to produce a first pass device test vector for the device under test requires the following three steps:

1. Run the TDS IN converter.
2. Run the TDS Conditioner SIMPLIFY.
3. Run the TDS OUT converter.

To run TDS IN Converter:

- 1 Start the TDS IN converter by responding IN to the command prompt.
- 2 Use the menu system to select the input simulator (Verilog or ASCII in this case).
- 3 Enter the appropriate file names for the events .file and the SIGNAL.SDF file.
- 4 Choose a name for the SEF destination file.
- 5 Enter the state mapping used in the Verilog simulation file (H is drive logic high, U is expect logic high, for example).
- 6 Select the RUN option to run the IN converter.
- 7 Use the TDRAW utility to view the simulation pattern.

To run SIMPLIFY:

The SIMPLIFY conditioner changes the time data in the SEF file between adjacent events to a uniform, user-specified interval, creating a new destination SEF file in the process. Alternatively, the WINDOW conditioner will provide the equivalent results as they relate to the Z1800-series VP, although with far more processing steps and programmer attention than with SIMPLIFY.

- 1 Start the TDS conditioner by answering CONDITION to the command prompt.
- 2 Use the menu system to select the SIMPLIFY conditioner type.
- 3 Use the TDS menu, select the combine mode and a time increment of 1µsec.
- 4 Enter the source and destination SEF file names as required.
- 5 Select the RUN option to run the SIMPLIFY conditioner.

The ASCII output converter creates an ASCII pattern file from the SEF file. This file format is then used by the Z1800-series VP to create a device test.

To run TDS OUT Converter

- 1 Start the TDS OUT converter by answering OUT to the command prompt.
- 2 Enter the source SEF file name (resulting from the SIMPLIFY conditioner) and the destination ASCII file name.
- 3 Select the RUN option to run the OUT converter.

The result thus far is an ASCII output file with the ASIC's test vector ready to transfer to the Z1800-series VP. If the device has a complex disable function requiring a simulation pattern, repeat the TDS process steps for the disable vector to create a second ASCII output file.

Step 4. Transfer ASCII vector files to Z1800-series VP

This step transfers the ASCII output file from the TDS host to the Z1800-series VP host PC.

Owing to the large size of these files, both efficiency and reliability of the transfer process are important considerations. For today's ASICs, the typical files size can easily exceed a megabyte, with the largest approaching six megabytes. Ethernet for these transfers is highly recommended.

Once all steps are completed, the ASCII pattern file resides on the Z1800-series VP ready for processing into a Z1800-series VP device test.

Step 5. Run TDS Conversion

The TDS Conversion program will convert the TSSI ASCII format vector file into a Z1800-series VP device test model into the vector format. The new file can then be processed by the program generator into a Z1800-series VP device test Step Worksheet.

- 1 Enter 18xxts to run the TDS Conversion program.
- 2 Then enter the TDS Conversion program through DOS.
- 3 Enter the appropriate file names for the ASCII format vector file and the device name.
You will be prompted to enter the pin numbers and names corresponding to the signal requirements found in the ASCII format vector file. In addition, you will be asked to provide the power, ground, and unused pins so that the complete mapping of total pins to functions is known. At the end, you will be prompted whether to continue on with the processing of a Z1800-series VP device test model.

Once all steps are complete you will have a Z1800-series VP device test model that can be used by the program generator to create a device test Step Worksheet.

Step 6. Create Z1800-series VP Single Device Test

You should debug the ASIC patterns individually in free air on a single-component fixture known as a Chip Checker before attempting to debug the ASIC pattern at the board-level. We highly recommend debugging in this manner for customers starting out with ASIC test pattern translation. The extra step decreases overall board test development time, as debug at the board-level is considerably more time consuming than at the device level. As you refine the design-to-test integration process and build confidence and productivity, you can omit the Chip Checker debugging step.

- 1 Create the input list for the node assignments of the device in the ChipChecker.
The input list token must also refer to the ASCII device model name just created. You can insert the reference with the editor, or with Generate's Probing function or Nodefinder.
- 2 Enter the Pgen menu from the Z1800-series Main menu.
- 3 Run Generate to create the device test Step Worksheet which will include power on, setup, etc., and the test itself.

Step 7. Debug the Device Test on the Chip Checker

A ChipChecker is a fixture for a single device comprising two parts: a vacuum fixture and interface pin board assembly, and a family of PC board adapters. The adapters are available in various device package sizes and types (DIP, pin grid arrays, PLCC, etc.). The ASIC under test is loaded into the ZIF socket of the adapter board and then power and ground lines hooked up with slide-on jumpers.

- 1 Run the device test Step Worksheet and debug via the editor.
First verify the disable signals (or test pattern if appropriate) for correct operation. Next, verify that the test vector pattern passes good parts, that pin level fault coverage is complete, that the test is repeatable, and that failing parts fail.
- 2 Once the vector test pattern passes and repeats the test coverage goals, either modify the ASCII vector template file to reflect the necessary channels or take a snapshot of the image as the debugged test model for integration into the entire board program.

Step 8. Create the Z1800-Series VP Board-Level Test

The board-level input list plus all required device model library elements are required at this stage to create the test Step Worksheet for the entire board. You can create the input list

- manually
- with the C-Link software tool
- with input list translators from CAD layout systems
- with the probing process available in Generate

The device model library for the ASIC created in the prior step must be referenced by name in the input list. Run the program generator to produce a first-pass board-level test Step Worksheet.

Step 9. Debug the Board-Level Test

Load the board under test onto the fixture to run and debug the entire board test.

Transporting Z875/Z850 Test Applications

A test application is a fixture and program used to test a board. Test applications designed for Z875 and Z850 testers can be used on Z1800-series testers.

To prepare a Z875 or Z850 test application for the Z1800-series tester, follow the procedures outlined in the paragraphs below.

Examine the Fixture

5120-node fixtures fit on the Z1884 tester. 2048-node fixtures fit on Z1890, Z1888-2, Z1880-2, Z1860, Z1820, and Z1803-2 testers. 1024-node fixtures fit the Z1888-1, Z1880-1, Z1860, Z1820, Z1805, Z1803-1, and Z850. The Z1866 tester accommodates twin 1024-node fixtures. 320 and 640 node fixtures fit Z1840, Z1800, Z1860, Z1820, Z850, and Z875.

Check to see that the fixture's power wiring scheme will work with the Z1800-series' power capabilities and pinouts. Do not remove the two backplane + and – wires. Make sure the power supplies are compatible. Use (optional) programmable relays as needed to insure that the fixture will work on both kinds of testers. Debug the power wiring before transferring the program.

Transfer and Edit the Input List

Load the Z850/875 program's Producer 2 input list into the Z1800-series tester using a file transfer program. For most components, the Z18XX input list format is similar to that of Producer 2.

Rename the input list IPL.DAT in the DOS environment. Create a subdirectory in the Test Program Directory (\TPD) and copy IPL.DAT into it.

Edit the input list to conform to the IPL format specification contained in chapter 6, "Program Generator Input List Reference."

Generate the Program

- Enter the 18xx programming environment's Main menu.
- Select the proper directory/board program. (The subdirectory created earlier.)
- Select Pgen.
- Select Clean. Clean will remove the test program (ICT.TST), the ipl database (IPL.DBF), token and node index files, component list, node list, and exceptions list. Clean will put a copy of the skeleton file in the ICT.TST. Clean will not remove IPL.DAT.
- Run Build to build the new input list. Note errors and correct with an editor.
- Select Generate to generate the program. Afterwards, look at the exceptions report to see which Producer 2 lines were not used by the Z1800-series tester.

Debug the Program

Debug the new program through the Step Worksheets and the editor. Refer to the Z850/875 program to check guard points and disable tables.

▼▼▼

A SUMMARY OF 18XX ERROR MESSAGES

The first section of this appendix lists the error messages which may appear while you are using the Z1800-series system software. The error messages are alphabetized. After each error message, you will find a brief explanation of the error message. This explanation may be helpful in completing the operation or process you are attempting.

The second section of this appendix lists the C-Scape messages.

18XX Error Messages

<file> does not contain any template type data.

This template transport file is not a valid 18XX template.

<file name> file cannot be opened.

If you are logged in as supervisor, go to DOS_Commands and use the Unlock function to make this file both read and write. Note that the Unlock function works on files, not directories.

<ID>: Illegal node separator (<separator>) found in IPL.DAT.

Valid node separators are "-" or "&".

<ID>: Invalid number of pins.

A device requires an even number of pins, but you have specified an odd number of pins.

<ID> is a deleted device. Undelete before continuing.

Undelete this device before proceeding.

<ID>, <name>: <#> pins specified in the template and <#> pins in the IPL.

The template and the IPL entry have a different number of pins.

<ID>, <name>: Analog template is not at the current ICT revision, <rev>. Use the program ATMPLCON.EXE to convert the library.

The program ATMPLCON.EXE was installed in the \MOS directory. It is used to convert an analog template library from D.2.3/E.2 or E.3/E.4 to F.0 file format. Type atmplcon at the command line and a usage message will appear.

<ID>, <name>: Disable vector not generated.

Disable was not generated possibly due to a constraint failure.

<ID>, <name>: Guard for pin <#> not generated.

Guard was not generated possibly due to a constraint failure.

<ID>, <name>: Scan vector <#> not found.

The 18XX system software cannot find the <name> file in the current board directory. Choose the Templates menu; select Add, and then Create IVL Cluster.

<ID>, <name>: Template not found.

The template was not found when you attempted to get a vector or Gray code test from the library.

<ID>, <name>: Template type does not match token type.

In generating a vector, the 18XX system software checks the template type first to verify that it matches with the ipl token.

1800-Series Login Incorrect. Please Try Again.

The login name or password you have entered is incorrect. Try again by typing your correct login name and password. Ask your supervisor if you need further help.

A**ADCread: ADC timed out.**

The analog/digital converter on the analog test board (ATB) did not supply a measurement result within the expected time. Run the system self-test. A problem may exist with the PC/IO, the test head controller (THC), or the ATB. The error message may also appear if power to the tester chassis has been interrupted. Quit the tester's software and restart from the Z1800 login.

Alias <alias name> not found in the template database.

When you are adding new aliases, this error message appears if the old alias name no longer exists in the DEVICE.LST file. You will not be able to add any new aliases to the database.

All Analog Pages are used for this Mixed Mode Test.

All of the mixed-mode analog pages have been allocated. The maximum number of analog pages allowed for a mixed-mode test is 10.

An error occurred when adding digital guard data.

When adding digital guard information from the guard file (GFILE.DAT) to the ICT.TST file, a read/write error occurred. A corrupted ICT program or a corrupted guard file may be possible causes.

Analog template library data requested but <path> is not readable.

When the 18XX system software was printing the header data for a template group, it was unable to read the ANTEMPL.LIB file. Check to see if the file exists.

APC threshold reassigned to <value> na.

The Auto Probe Check (APC) threshold specified in the program header has an invalid value. To execute the test, its value has been reassigned to the default value. Modify the APC threshold in the program header if required.

Asynchronous port already in use.

You should assign only one active device to a PC COM port at a time. You can assign both BoardWatch and Line Printer to the same port in the menus, but an error will occur when you run the test.

B

Backup file <name> does not exist.

The 18XX system software could not find the <name> file when it was attempting to restore a backup version of the template library.

Bad data for Sys/Opt/Usr flag

ICT.TST has corrupted data Sys/Opt/Usr flag data for this test step.

Bad stim/meas value (in DSRD[])

ICT.TST has corrupted data in Gray code test.

bad stim/resp.

A bad digital test record was encountered during normal read/write processing. The file is in error. Restore the file prior to backup. Both an existing template alias and a new alias name must be provided. In adding a new alias to the template database, you have failed to provide existing and new template aliases.

Both an existing template alias and a new alias name must be provided.

In adding a new alias to the template database, you have failed to provide the existing and new template aliases.

Bridge conflict between Emitter and Coll. node <node>.

It was not possible to route the desired emitter and collector nodes to the analog test board (ATB). Move either the emitter or collector node to a different driver/receiver board.

Button "Type: Vector", will not work!

Could not create a temporary file in which to pass the list of selected vectors to document. Exit the 18xx and try again.

C

Can only have 10 mix-mode analog pages per digital test.

The digital test buffer can hold only 10 mixed-mode analog test pages. The maximum number of pages cannot be altered.

Can't add discharge test, section overflow.

You cannot add a discharge step because the discharge section already contains the maximum number of steps.

Can't add test step for capacitor discharge.

An error occurred while adding a step to the ICT. A disk space problem may exist.

Can't add test step for component <ID>.

An error occurred while adding a step to the ICT. A disk space problem may exist.

Can't resolve <token> statement: conflict with previous PWR statement.

The A/B supplies must both be in either fixed/slaved or adjustable/programmable.

Can't test large Caps in Six-wire mode!

Capacitors that are approximately 300 uF or greater are tested in DC charge-up mode. Six and four-wire testing are not compatible with this mode. Six-wire testing is unnecessary in DC charge-up mode as the virtual impedance of the capacitor under test is held at roughly 120 ohms.

Can't validate in 4 or 6 -wire mode, assigned 3 wire mode.

Four and six-wire tests cannot be validated. The tester will set the wire mode to three-wire and attempt to validate the test. Check the resulting test to determine if it is necessary to switch back to the six-wire mode. When you are satisfied with the test, set the Disable Validate flag to prevent further attempts to validate this test step.

Cannot add new template to deleted device. Undelete device before adding.

In adding template updates to the database, you have specified a deleted device name.

Cannot add this Device Type to the library.

This error message appears when the system tries to write a snapshot of the test to the template directory. The only device types that can be added to the library are Vector, Vector Image, Gray Code, and Analog Template.

Cannot append, exceeds maximum number of steps.

When you were trying to append steps to the current section, the maximum number of steps per section has been exceeded.

Cannot change directory to <path>

The 18xx system software was unable to change directory to the identified path. Make sure that the path is valid and does exist.

Cannot delete all pins of a vector.

A vector test must have at least one pin.

Cannot execute DFP test: Arguments, when expanded, exceed <number> characters.

The argument list, after expansion of %ALPHA, %INT, and other 18xx macros exceeded <number> characters! Reduce argument list size and try again.

Cannot execute Sp/Ignore: no Short range given.

To execute the Special Case/Ignore section, you must specify a short range in the Short section of the program. Only nodes from the short range will be activated in the Special Case/Ignore test.

Cannot Get Pins for this Device Type.

This error message appears when you attempt a "get template pins" in Step Worksheet editing. You can only use "Get Pins" from Template Library for Vector, Vector Image, Gray Code, and Analog Template Device Types.

Cannot make all pins unused.

A vector must have at least one active pin.

Cannot pack database to leave <#> active aliases. Undelete at least 1 alias.

The number of active aliases is less than one. The packing of database is not complete.

Cannot re-open file <file name> before it is closed.

An attempt was made to open a file that is already open. Exit the 18xx and re-enter.

Cannot wait <number> ms!

The wait in the StimV mixed mode vector statement cannot exceed 10,000 ms.

Can't modify master program.

An error occurred while adding the calls to chain programs to the master program. These calls are in the form of Header or Trailer steps.

Chain depth exhausted: Cannot chain more then <#> levels.

Modify your program so that it does not chain as deep or change the "chain depth" variable in the Setup/Environment menu.

ch_in can only be read from a COMM device.

The Input Port setting must be a COMM port. Verify that the setting is correct in the Setup menus, and edit if necessary.

clock_rate: maximum clock division (200) exceeded.

Use the editor to lower the clock divisor.

close_apl: file close error.

An error occurred while closing the ICT test program.

comp type not processed yet.

While reading or writing a test page, the 18XX system software has found an invalid component type. The test data is corrupted.

Component <name>, <ID> exceeds maximum backdrive time allowed.

The time required to execute this digital test burst exceeds the maximum backdrive time allowed for the Backdrive Timeout variable in the Environment Setup menu. Reduce the backdrive time or increase the Backdrive Timeout in the Environment Setup menu.

Component Database doesn't match software revision level!

The component database, which has been written under a previous version of the operating system, is not compatible with the current file structure. Run Update to recreate the component database from the test program for the current revision.

Conflicting disable table entry for node <node number>.

During the digital test step generation process, a new and conflicting disable stimulus type was found for an existing node in the disable table. Check the state of the disable for the given node.

copy_file(): read file error: <message>.

<message> occurred while you were attempting a file copy.

Could not add aliases to <DEVICE.LST>.

The attempt to add an alias to the DEVICE.LST file failed.

Could not add guards.

The 18XX system software was unable to add the specified guards to the vector test.

Could not initialize memory for vector.

The 18XX system software was unable to allocate temporary memory for the vector data.

Could not locate device <device name> in <DEVICE.LST>.

The 18XX system software cannot perform “get template pin”, “get template”, “get revision, list”, “delete” or “update” because it cannot find the specified device.

Could not open <file name> file in the <read/write> mode.

The wb mode allows you to create a new binary file in write mode.

The rb+ mode allows you to open a binary file with read and write mode.

Could not open <file name> file in the wb+ mode.

The <file name> file was not opened. Check your DOS environment files variable.

Could not patch edits to vector.

An error occurred when the software was saving edits to vector file.

Could not remove existing <file name> file.

You may not have the proper permission to remove the file.

Could not reopen new <file name> file.

A problem may exist with the <file name> file, or you may not have the proper permission to open the file.

D**DFP Software not found!**

Could not find DFPPROM.EXE or DFPVER.EXE. Reinstall DFP software and try again.

delete_step: Test already deleted.

You cannot mark a deleted step for deletion.

Deletion of all pages is not allowed.

To delete all the pages, you should delete the test step.

Different number of pins specified in template and IPL record.

The number of pins in the template and IPL for this device must match.

DIG_GUARD_USED

Node <decimal> is already used on Pin <decimal>.

Digital device file format error.

The 18XX system software has found invalid data in the file for this test step.

Digital: power must be applied prior to digital testing.

Power has not been turned on. The Brd_Power section turns board power on, and the section must be executed prior to executing a digital test. Go to the Brd_Power section and turn on the 5-volt supply.

Digital Test Buffer is empty.

You cannot use the Get_Signature function on an empty digital page. Edit Test Properties by filling in the stimulus for all stimulus pins which affect the pin for which you want to get a signature.

Digital Testing not supported with this version O.S.

Your system does not support digital testing. The 1805 system software prohibits the editing, generating, or running of digital tests.

dig_thresh: digital threshold out of range (-2.0 to 9.95).

Use the editor to enter a valid threshold value.

Directory is empty.

The 18XX system software has failed to find any files of the requested type in the directory.

Disable count limit reached.

A limit of 128 entries is allowed in the digital disable table. Do not attempt to enter more than 128 disables.

Discharge method in Header/PRGMVARS not compatible with power worksheets. See HELP topic POWER SUPPLY CONTROL for more info.

You cannot use the PwrNodes Discharge method with the programmable power worksheets. Either change the Discharge method or change the power worksheets.

DISetup: discharge limit < 0 volts.

A discharge below 0.0 volts was requested. Check and correct the discharge limit for this test.

DUT supply set to different range.

You have tried to set a supply that is already set to a range other than the one requested. Review your test strategy and reset if required.

E**EMPTY template database <path>. Quitting operation.**

An operation is being performed on an empty Template Library.

End of pattern delimiter ';' not found, state <#>.

A syntax error exists in the 18XX IVL file format.

Error adding EOPIN mark to node array.

When the 18XX system software was building the component node array, an error occurred in adding the end of a pin marker. A memory allocation error exists.

Error adding EOPIN_MARK to node list. pin <#>.

When the 18XX system software was building the component node array, an error occurred in adding the end of a pin marker. A memory allocation error exists.

Error adding node <#> to node list. pin <#>.

When the 18XX system software was building the node list for this component, an error occurred in adding the node. A memory allocation error exists.

Error adding nodes to node array.

When the 18XX system software was building the node list for this component, an error occurred in adding the node. A memory allocation error exists.

Error: cannot use %ENODE, %FNODE, %GNODE for this test type.

This test type does not have E, F, or G nodes. Therefore, it is an error to use %ENODE, %FNODE, or %GNODE macros.

Error in input string, <format_string>, format.

The format string is not in the proper format. The format string must strictly adhere to the following format:

```
<digit(s)><char><digit(s)><char>[<digit(s)> <char>]
```

where digit(s) is a numeric value in the range of 0 - 9 and char consists of the alpha characters X, N, or R.

Error opening files.

Check your DOS files statement. You may be out of files, buffers, or memory.

Error while inputting string from input port.

The PC is not receiving data at its serial port. Verify that setup variables are correct and that the serial cables are properly installed.

Error while writing to file <file name>!

Verify that your hard disk is not full.

Error writing an index to file <file name>.

Verify that your hard disk is not full.

Error writing to <file name>.

Verify that your hard disk is not full.

Error writing to free format file.

Verify that your hard disk is not full.

error reading template section main.

The data is not correct and may be a previous software version of section snapshot.

error reading template test steps.

The data is not correct and may be a previous software version of section snapshot.

error writing test step.

An error occurred when the 18XX system software was writing a test step to a file. Check the available disk space.

ERROR: Can't update teststep for component <ID>.

An error occurred when the 18XX system software was writing a test step to a file. Check the available disk space.

Error: Cannot create limits from a value of 0.

The 18XX system software cannot calculate a percentage high and percentage low limit from a value of 0.

Error: Select nested within a Repeat.

An invalid template exists. You cannot nest selects within repeats.

Error: Unable to read timestamp from component database.

A problem exists with the component database and associated index files. Recreate the component database by performing an Update.

Exceeded Maximum <number> analog statements.

A maximum of 10 vector mixed mode statements (MeasV and StimV) are allowed. Note that of the 10 statements, only one may be a StimV.

Execution aborted, unable to learn Cap Offsets for read-only files.

Execution is aborted because the system is unable to learn Cap Offsets for read-only files. The learn Cap offsets feature can be used only with read-write files.

Execution aborted, unable to successfully learn Cap Offsets.

Execution is aborted because the system is unable to successfully learn Cap Offsets. This occurs because the capacitor could not be measured. Verify that the test step for this capacitor can be run from the worksheet.

Expanded List data of <number> entries too large for 18xx pointer data structure.

While the vector was generating, the number of pin changes has exceeded the amount that the data structure can hold. During the Expand phase of vector generation, 4 megabytes of pin changes is the maximum.

F**Failing states > <number failures>, exceeded PC space available.**

The 18XX system has run out of conventional memory and cannot store the failure data.

File is out of sync.

The in-circuit or fixture self-test program is not in the proper format. Restore the file with the latest backup file.

File not in 1800 format.

The user file is not in the correct format. The program is probably the wrong revision level. This error is usually accompanied by another error specifying where the file was found to be in the correct format. Refer to this message or messages for corrective action. If the file was destroyed, copy the backup file (ICT.BCK) to the user file (ICT.TST) using DOS_Commands.

File revision/type not correct for <file name>.

A mismatch exists in the software revision level or in the file type.

File is READ-ONLY.

The file you are trying to open is locked and is specified as "Read-Only." If you are logged in as supervisor, go to DOS_Commands and use the Unlock function to make this file both read and write. Note that the Unlock function works on files, not directories.

First page must be generated. Cannot add.

You cannot add a page to an ungenerated test. Generate the test first, and then add pages.

Fixes have been lost. Restoring user file to last backup.

The file structure was corrupted during the file packing process when modifications were being merged into the test program. The system has reloaded the prior backup file.

G**General WaveScan failure, possibilities are: Open Ground Lead, Un-socketed Device, Backwards Device, or Wrong Device.**

This debug only message for WaveScan tests lists the possible causes of the failure.

Graycode Burst timed out.

The digital Gray code test was not completed within the time limit specified for VP Timeout in the Setup/Environment menu. Increase this time or disable VP Timeout in the Options/Control menu in Test Properties. If the problem continues, there may be a hardware error in communicating with the testhead controller board.

Guard count limit reached.

A maximum of 20 guards is allowed for each digital test step. You will have to determine the best combination of twenty or less digital guards which will give the most reliable test result.

H**Header-PRGMVARS-WaveScan reference node required.**

To execute WaveScan test, the Header section PRGMVARS test step must contain a valid tester node number in the WaveScan reference field.

I

Illegal combining of standard AND programmable power supply worksheets!
Please see HELP topic POWER SUPPLY CONTROL for more info.

If the first power test step executed is a new power Step Worksheet, then the power supply controller is set to the new mode, and this message is displayed if an old power Step Worksheet is executed. If the first power test step is an old power Step Worksheet, then the power supply controller is set to the old mode and new power Step Worksheets will cause this error. Note that this message is possible only when the programmable power supply controller is installed.

Illegal DOS file name: "<string>".

Chosen output file name is not a legal DOS file name. Choose another file name and try again.

Illegal pin number <number>; number must be between 1 and <number>.

In vector mixed mode statements, StimV or MeasV, the pin number exceeds NPINS—the number of pins declared.

Improper voltage range!

In the vector mixed mode statement, MeasV, the voltage range should be expressed as <low voltage> to <high voltage>. For example, MeasV pinxxx, 9.5 to 10.5 V Peak.

INDmeas: measured DC resistance >= impedance.

The analog test board (ATB) cannot measure this inductor. The ATB reads a higher DC resistance than AC impedance, which is impossible. Convert this test to a resistive measurement using RES or longhand test mode.

INDsetup: programmed inductance <= 0.

The requested measure value was less than or equal to 0.0 5h for this component. Check and modify the requested inductor value in the Editor.

Inputlist input line too long. Buffer overflow!

The text inputlist (IPL.DAT) is longer than 1023 characters, and its contents have been discarded. An inputlist statement must be contained within 1023 characters and terminated with a carriage return linefeed. Check your inputlist (IPL.DAT) with a text editor to find the faulty line, which is likely to occur with long lists of continuities. Breaking the list into multiple continuity statements will alleviate the problem.

Interconnect threshold reassigned to <value> ohms.

The interconnect threshold specified for this test has an invalid value. To execute the test, the 18XX system software has reassigned its value to the default value of 5.0 ohms. Check the threshold values in the Interconnect section and correct faulty values.

Invalid BusScan Test Type.

Invalid BusScan syntax is specified in the Vector template. Valid BusScan test types are STUCK_BUS, STUCK_DEV, STUCK_OPN, STUCK_SHORT.

Invalid node number given, try <low>—<high>!

You have entered an invalid node number when requested for lowest or highest node number. Enter again within the range given by the error message. If the <high> field does not represent the highest node in the system, go to the Setup menu and correct the entry for the last driver/receiver node.

Invalid discharge setup.

The Discharge test is not properly set up. Go to the Step Worksheet and verify that the discharge limit is not too low.

Invalid setup.

The test cannot run with the values specified. Go to the Step Worksheet and verify that the expected value and the tolerances are correct.

Invalid template name characters, "|<>".

A DOS file is created from the template name. These characters cannot be used to create a DOS file. Therefore, they are invalid as characters in a template name.

L

Linear: power must be applied prior to linear testing.

Power has not been turned on. The Brd_Power section turns board power on, and must be executed prior to executing a Linear test. Go to the Brd_Power section and execute the power on steps.

M

Maximum input value string length is 21.

The value string is composed of a number of characters specified either before the N (normal parse) or R (reverse parse) format character. These are the characters that are collected and converted into the floating point value.

Maximum number of digital measurements exceeded.

Sixty four or more measurements have been generated for this test. Go to Step Worksheet and verify the number of measurements required. If a large number of measurements is required, consider dividing your test into two tests.

Maximum number of measurement sequences exceeded.

There is a limit of 32 digital measurement sequences per test. A sequence is a complete measurement burst. The worst case allows for 32 measurements for a device and occurs when the Digital Groups flag is set to separate. The best case would allow for all 84 pins.

Measurement pins only belong to one group.

Each measurement pin can only be assigned to one group. Scan through the Step Worksheet and verify that no Meas V or Resp pin has more than one group assigned to it.

Measurement out of range.

When the system was restoring a previously set power supply, the measured value was not within the tolerance range specified by the test that originally set this supply. Verify that the supply is on and functional.

Mismatch—Password unchanged.

You must verify the new password before it is accepted. When you entered the new password the second time, it did not match the first entry. Try again, but use the same spelling for both password entries.

Missing ';' following last 'MeasV' statement.

The last MeasV statement must be terminated by a semicolon.

Missing comma between MeasV statements.

When more than one MeasV statement occurs on a state, all MeasV statements except the last one must be followed by a comma.

Missing pin name or number.

The vector mixed mode statement, StimV and MeasV, require a pin name or the pin number to identify the pin the stimulus or measure will be performed on.

Missing right parenthesis.

The matching right parenthesis of an IVL vector state is missing. For example, L(1,2,3 H(4,5,6);
The comma is missing between the 3 and the H.

Mix-Mode analog data is not allowed in guard or disable vectors.

The mixed mode vector statement, MeasV, is allowed only in the Vector section.

Mix-mode analog meas is outside the digital bursts execution window.

The wait time specified in mixed-mode Meas V test page forces the analog measure to take place after the digital test has been completed, i.e., after the digital stimuli have been removed. The tester will attempt to correct this problem before this message appears by increasing the “to time” of this burst incrementally up to F16. Increasing the “to time” lengthens the digital test time and may bring the analog measure within its bounds. If the tester cannot successfully manipulate this test, the error message appears, and the test is allowed to continue with a “to time” of F16.

Mix-mode analog meas wait time exceeds maximum digital test window.

The mixed-mode analog measurement is outside the digital test window and the tester cannot compensate for this error. Even if the “to time” and clock divisor are at their maximum values, F16 and 200 respectively, the measurement will still fall outside of the digital test window.

Mix-mode analog measurement error.

The Analog/Digital Converter on the analog test board (ATB) did not supply a measurement result within the expected time. Run the system self-test. There may be a problem with the PC_IO, the test head controller (THC), or the ATB. The error message may also appear if power to the tester chassis has been interrupted. Quit the tester's software and restart from the Z1800 login.

Mix-mode analog setup error.

An error occurred while setting up the mixed-mode analog test structures. Check to see that you have an analog page for the assigned mixed-mode pin.

N**Need 2 or more nodes on <string> pole for remote sensed setup.**

The requested wire mode requires two or more nodes on the pole identified by <string>.

New alias <alias name> already exists in the template database.

The alias name you have given is a duplicate of what already exists in the database.

New Path is too long...

The new path you've created is too long for DOS. DOS has a file path limitation of 64 characters. Consult your DOS manuals for detailed information.

No Analog Measure data for pin <number>.

There is no corresponding MeasV for the M in the vector state. For example, L(1,2,3) H(4,5,6) M(7,8) MeasV7, 9.5 to 10.5 MV peak;

The MeasV statement for pin 8 is missing.

No component Types Selected For Global Validation.

The Setup/Validate menu allows you to choose which component types to validate. No components have been selected.

No digital device in the Test Buffer.

You cannot save an empty test. Edit the test by filling in the test information for each pin in the device. Then use the Save function.

No DRIVE node specified on <string> pole.

No drive node is specified on the E, F, or G pole specified by <string>.

No Driver/Receiver boards found in the system.

Prior to running a test program, the system checks the driver/receiver boards to see if full initialization is necessary. The error message appears when the system does not detect the driver/receiver boards. You may also have a problem with the test head controller (THC) or analog test board (ATB).

No interconnects left.

You have reached the maximum number of interconnects which can be added to the test program.

No Memory Available for the Asynchronous Buffers.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

No memory for the Component Names.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

No new guardable points found.

Either no new digital tests have been added since the last digital guard analysis, or no usable guard points were found in the templates used.

No non-zero group or valid node measurements specified.

A test has been created that does not have any groups specified other than the zero group. Go to the Step Worksheet, scroll through the group fields, and verify that separate tests are indicated by different group numbers.

No SENSE node specified on <string> pole.

One of the nodes being used on the E, F, or G pole specified by <string> must be a SENSE node for the requested wire mode.

No template name specified.

The template name field has been left blank. A name is required before the template can be created.

No valid signature exists for this pin.

You cannot use the <F9> key to obtain a signature before the test has been executed. Use the Go function first, then use the Get_Signature function while in the Step Worksheet on the pin for which you want to obtain a signature.

Node conflict, node <#> - used on pin <#> and <#>.

Two pins of this device are connected to the same tester node and are the same pin type or one pin is a digital stimulus while the other is an analog stimulus.

Not able to extract modified device <device name>.

You cannot extract to the .TTM file because of invalid permission or insufficient disk space.

Not able to find link list head for <name> in <file name>.

The 18XX system software is unable to find a link list head for the alias group. The database has probably been corrupted.

Not able to make file: <file name>.

You cannot perform this function because of invalid permission or insufficient disk space.

Not able to show header data requested.

The error occurs when the list function prints the header data for a template group to the PRINT.DAT file.

Not enough memory!

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Not enough memory for capacitor bypass list.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Not enough memory for component count array.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Not enough memory for interconnect merge.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Not enough memory for ipl record.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Not enough memory for node index array.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Not enough memory for node list.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Not enough memory for Pin Array.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Not enough memory for RC combination list.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Not enough memory for token index array.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Not enough memory for Step Worksheet data buffer.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Not enough memory to create browser search window.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Not enough memory to create browser window for Fault Inject results.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Not enough memory to create the bob.

Bob stands for Basic Object.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs that are not loaded high reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Not enough memory to create the menu.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Not enough memory to create the window.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Not enough memory to execute DFP Software.

Reduce number of loaded programs, drivers and/or devices, and try again.

Number of active aliases (<#>) exceeds maximum (#). Delete <# of aliases> before pack.

Too many aliases exist.

Number of aliases for <device number> in old template database less than 1.

Too few aliases exist.

Number of pins in pin section exceeds NPINS <number>.

NPINS must equal the number of pins listed in the pin section.

**One analog stimulus per digital group.**

Due to hardware configurations, each digital group can have only one mixed-mode analog stimulus assigned to that particular group.

Only one 'Stim V' statement allowed.

Only one StimV statement is allowed per vector template. See “Only one Analog Stim allowed per test” below.

Only one Analog Stim allowed per test.

Mixed-mode tests may have multiple analog measurements in which each analog measure is in a separate burst. Since all stimuli must be present in all bursts, only one analog stimulus per test is allowed.

P**Page count limit reached.**

You have already added 32 pages, which is the maximum number of pages allowed per test step. If you need more than 32 pages, you must use more than one test step to represent the specified component test.

Page disabled.

Setting the Test Page Execution flag to Disable in the Flags Control window of this test has disabled the execution of this test page. If you want this test page to execute, go to the Flags Control window (of this page) and enable Test Page Execution.

Panel config read error.

An error occurred while reading from the Multipanel configuration file.

Panel config write error.

An error occurred while writing to the Multipanel configuration file.

Password file is nonexistent or unreadable.

The password file has been corrupted. Restore the password file from your last backup or reinstall the tester's software.

Password unchanged.

A zero-length password is not allowed. Use a password with at least six characters.

Pattern has MeasV statement(s), but no associated 'M's!

There is no corresponding M for the MeasV statement in this pattern. For example, L(1,2,3) H(4,5,6) M(7) Z(8) MeasV 7, 9.5 to 10.5 peak, MeasV 8, 4.5 to 5.5 pk-pk;

In the above pattern, pin 8 should be an M, not a Z.

Pause: timing error.

No timers were available on the IO board while it was attempting to discharge the board-under-test.

Permission denied.

The current login name does not give you permission to execute this function or access this menu. Log in using a login name with the proper permission to execute the function. Refer to chapter 1 of this manual for detailed information.

Pin <pin number> already used with another Analog Page.

The specified pin has already been assigned to a mixed-mode analog page. It may not be reassigned until the analog page is released by changing the specified pin's stimulus/response to something other than Meas V or Stim V.

Pin number greater than device pin count.

In the pin list section, the pin number of this pin exceeds NPINS. (NPINS is the total number of pins on the device).

Pin <string> (<number>) does not have an analog Measure ('M') on it!

There is no corresponding MeasV for the M in the vector state. For example, L(1,2,3) H(4,5,6) M(7,8) MeasV7, 9.5 to 10.5 MV peak;

The MeasV statement for pin 8 is missing.

Pin <string> (<number>) is not an Analog <string> pin!

This pin was not declared as an analog pin (AI or AO) in the pin list section.

Pin table contains data for only <number A> pins, expecting pin data for <number B> pins.

The pin table is missing data for pin <number A> to <number B>.

Power bus is not discharged.

The Power Off routine attempts to fully discharge the board on all power buses if power was applied. This process prevents arcing when the board is removed from the fixture. The maximum discharge time is set at 5 seconds per bus. If the system does not fully discharge the board this error message is displayed.

Power not set correctly.

The tester is unable to measure the requested value from this supply. Verify that the requested value is reasonable and that the supply is on and functional.

Power restore not successful! Relay /power actions after BRD_POWER section not restored.

The power restore executes test steps from the power section only. If you have used Relay or Power Control from the Pre-/Post-Tests of test steps from sections other than the Power section, then power restore may not restore the tester to the same state as when power-down was executed.

Printer <name> is not plugged in.

The specified printer is not plugged in. Verify that the cable is plugged in to both the printer and the computer.

Printer <name> is off line.

The specified printer is off line. Verify that the printer is on line.

Printer <name> is out of paper.

The specified printer is out of paper. Load the printer with paper.

Printer <name> is powered down.

The specified printer does not have power. Verify that the printer is plugged in and on line.

Process aborted by Cancel.

The keyboard or tabletop Cancel button was activated.

Program revision level does not match system revision level.

The revision level of the program you are trying to run is not the same as that of the tester's software in use. Run the Convert program located in the \MOS directory on your PC. Select the program that initiated the revision error and convert it.

Punctuation syntax error.

An unexpected comma or semicolon in a vector mixed mode statement, StimV or MeasV, displays this message.

R

rd_option: Not enough memory for pre/post buffer.

The system does not have enough memory resources for internal data structures. Go back to the DOS level and free all possible memory. Note that all Terminate Stay Resident programs reduce your available memory size. Check the available memory by executing CHKDSK from DOS. You must have nearly 600 kB available to operate the tester.

Rebooting the VP sub-system.

The VP did not respond during the execution of a vector test.

Requested node <#> not available.

An attempt to run Interconnect has failed because the number of system nodes, which was specified in the Environment Setup menu, has been exceeded. Reset the system nodes or check the test program.

S

Screen data file, WORK.BIN, wrong revision.

The screen data file found by the operating system is the wrong revision for this operating system. This file should be in the \MOS directory. Make sure your DOS PATH is correct and the Setup-Data/1800 Operating System Drive is correct.

Sections: Jumper, Continuity and Ignore have no test steps, therefore the All SC test step contains no nodes.

This message occurs when you are trying to edit the SC Merge test step and all the jumper, continuity, and ignore steps are either marked as deleted or disabled.

Sorry. Passwords do not match.

You entered the password incorrectly. You must know the current password before you can change it.

Specified template is not in the Template Library.

No template exists for the specified device in the Template Library. Check your spelling of the template name using the List option and try again, or enter the component test information using the editor.

stimulus current > 25 mA (100 V Opt).

The editor will limit the zener Rev I current to 250 mA. For zeners with voltage drops greater than 10 volts, the current is limited to 25 mA. This error message is displayed only during Run or Go.

'<String>' does not match keyword '<keyword>'.

There is a bad format in the vector mixed mode statement, StimV or MeasV.

'<String>' does not match one of '<list>'.

There is a bad format in the vector mixed mode statement, StimV or MeasV. One of the elements out of the <list> was expected.

T

Template file <name> does not exist.

You cannot delete or undelete a template which does not exist. Check your spelling of the template name using the List option and try deleting or undeleting again.

Test failed before Fault Inject.

Before all Fault Inject vector bursts are executed, that is, the vector test is run with one input pin held high or low, the unmodified vector test is executed. If this test fails, the device has not been initialized completely and the Fault Inject results will not be reliable.

Test Head Controller not available.

A hardware error was discovered by the system.

Test Type/Device Type mismatch, cannot create template.

You can create a template only from a Step Worksheet for Analog Template, Vector Image, and Gray Code Device Types. If the Device Type is Vector Image, the Test Type must be Vector. If the Device Type is Gray Code, the Test Type must be Gray Code.

Text Inputlist Syntax Error: Wraparound not followed by node-number.

The indicated input list line was not formatted correctly. If a statement needs to be continued on the second line, a number must be the first entry on the continued line. Correct the text inputlist file.

The chained program <file name> does not exist.

The 18XX system software attempted to run the named program as a chained program, and the program name was not found. Return to the main select menu and look for the program name. An error could have been made in the Header/Trailer step that called the program's file name. Edit this step to call the correct program.

The power worksheets used require a programmable power supply controller. See HELP topic POWER SUPPLY CONTROL for more info.

To execute the programmable power worksheets, the programmable power supply controller is required.

There are no Disables to Delete!

You attempted to delete a disable when no disables exist. Use the Delete function only when there are disables to delete.

There are no Guards to Delete!

You attempted to delete a guard when no guards exist. Use the Delete function only when there are guards to delete.

This directory contains Read Only Files - Remove Aborted.

“Read-Only” files cannot be removed using the Remove command. “Read-Only” files may be changed to read/write files by using the Unlock command in the Utilities MS-DOS_Commands menu. To completely eradicate a subdirectory, all files must be deleted prior to removing the subdirectory.

This directory contains sub-directories - Remove Aborted.

To completely eradicate a subdirectory, all files must be deleted prior to removing the subdirectory. Remove all files within the subdirectory (if desired), then proceed.

This directory contains System Files - Remove Aborted.

The Remove command cannot remove system files. To completely eradicate a subdirectory, all files must be deleted prior to removing the subdirectory. If you wish to remove system files from your hard disk, verify a recent backup copy exists before proceeding.

This occurred after a <name> fault was injected on pin <#>.

This message, which follows the “Test failed before Fault Inject” message, gives specific information about the previous Fault Inject burst's uninitializable condition.

This template has too many pins.

The requested template has more than 84 pins. The maximum pin count available for a digital device is 84 pins. Develop the test using the editor. Limit the number of pins in the test to 84. Use the Save_Test function to obtain a snapshot of the test configuration which may be added to the library.

U

Unable to allocate test step ID array.

The system was unable to allocate memory. This could be a sign of a memory leak. Exit the 18xx and restart. If it was a memory leak, the error should no longer appear.

Unable to allocate test step index array.

The system was unable to allocate memory. This could be a sign of a memory leak. Exit the 18xx and restart. If it was a memory leak, the error should no longer appear.

Unable to change back to the working directory.

A DOS system error occurred while you were attempting to change back to the working directory after verifying the existence of the working directory. Reboot the PC and continue. If the problem persists, consult your DOS manuals.

Unable to change back to the working drive.

A DOS system error occurred while you were attempting to change back to the working directory after verifying the existence of the working drive. Reboot the PC and continue. If the problem persists, consult your DOS manuals.

Unable to close the user file.

An error occurred while you were trying to close the user file. Lack of memory or a closed file may have initiated this error. Verify that free memory is available for storage.

Unable to close user file.

An error occurred while you were trying to close the user file. Lack of memory or a closed file may have initiated this error. Verify that free memory is available for storage.

Unable to copy the Header Flags Table.

Only one Header Flags Table may exist in the Header Test section; it cannot be copied.

Unable to delete the Header Flags Table.

Only one Header Flags Table may exist in the Header Test section; it cannot be deleted.

Unable to find matching Begin.

Too many end statements produce this error message.

Unable to find WaveScan Inducer.

An attempt to read the Inducer Finder signal failed. This occurrence is probably due to a WaveScan hardware failure.

Unable to move the Header Flags Table.

The Header Flags Table must always be the first component in the Header Test section; you cannot move it to a different location in the Component Index.

Unable to open file <file name>.

The specified file could not be opened for appending. Verify that the <file name> file exists.

Unable to open file: <file name>.

The requested file could not be opened. The file may be missing or the number of files may be restricted in CONFIG.SYS, FILES. The setting should be Files = 28.

Unable to open files for Short Locator.

The Shorts Locator function uses the component database files to display additional information for each failed node. The component database may be corrupted or missing. Recreate the component database using Update in the Generate menu.

Unable to open template library index file.

The requested file (TLIST.IND of /PGT) could not be opened. The file may be missing, in which case you could reload it from a backup or release disk, or too few files are specified in CONFIG.SYS, FILES. The setting should be Files = 28.

Unable to open the BoardWatch Port.

The specified port could not be opened. Verify that the port is connected.

Unable to open the Datalog File.

The specified port could not be opened. Verify that the port is connected.

Unable to open the Diagnostic Printer Port.

The specified port could not be opened. Verify that the port is connected.

Unable to open the Format String Port.

The specified port could not be opened. Verify that the port is connected.

Unable to open the guard file.

The requested file (GFILE.DAT in the currently selected directory) could not be opened. The file may be missing, in which case you could reload it from a backup or release disk. Too few files may also have been specified in CONFIG.SYS, FILES. The setting should be Files = 28.

Unable to open the temp file.

The requested file could not be opened or created. It is possible that too few files are specified in CONFIG.SYS, FILES. The setting should be Files = 28.

Unable to open the Test Program.

The requested test program file could not be opened or it may not be in the right format. If the file is in a format that is invalid, you can restore the file from a backup file. Also, too few files may have been specified in CONFIG.SYS, FILES. The setting should be Files = 28.

Unable to process input string, no pre-test specified.

An input string test cannot be executed if the test does not have a pre-test option buffer specified with the input string option set. Verify this condition and edit where necessary.

Unable to process input string, pre-test buffer is empty.

An input string test cannot be executed if the test does not have the input string option specified in the pre-test option buffer. Check the input string and edit where necessary.

Unable to read WaveScan results from the <file name>.

This debug message indicates that the operating system was unable to read the Wave-Scan results from the named file.

Unable to remove '<path>': <DOS error message>.

A DOS error occurred while you were trying to remove this file. Please refer to the DOS error message and your DOS manuals for further explanation.

Unable to remove directory '<path>': <DOS error message>.

A DOS error occurred while you were trying to remove this file. Please refer to the DOS error message and your DOS manuals for further explanation.

Unable to restore power.

During certain activities in the development of a board test, the tester sometimes removes board power, as when node probing is specified in digital test pages. Later, when power is required (for Go in digital tests), the tester will attempt to reapply board power. If reapplication of power fails (measured value outside of limit), the test will display this message and halt all further testing on this board.

Unable to set up the BoardWatch Port.

The Environment configuration file has been corrupted or erased. Use the Setup option from the Main menu to edit the Environment and Serial configuration parameters. Be sure to save the new setup which will create a valid Environment configuration file.

Unable to set up the Datalog File Name.

The Environment configuration file has been corrupted or erased. Use the Setup option from the Main menu to edit the Environment and Serial configuration parameters. Be sure to save the new setup which will create a valid Environment configuration file.

Unable to set up the Diagnostic Printer Port.

The Environment configuration file has been corrupted or erased. Use the Setup option from the Main menu to edit the Environment and Serial configuration parameters. Be sure to save the new setup which will create a valid Environment configuration file.

Unable to set up the Format String Port.

The PC cannot find the serial port hardware as indicated in the Input Port field of the Setup menu. The address and/or the interrupt setting of the serial port hardware may be incorrect. Verify this condition and edit where necessary.

Unable to validate WaveScan Test.

WaveScan Validate was not successful. This occurrence could be due to a file error with the WaveScan results.

Unable to write step, program is read-only.

The file you are trying to write to is locked and specified as “Read-Only.”

Unable to write WaveScan parameters to file <file name>.

This is a general file write error. It may be the result of insufficient disk space or an insufficient number of file handlers.

Unexpected 'end set', 'end block' or 'end entry' statement.

The End and Begin statements do not match up at this point.

Unexpected semicolon.

The program encountered a semicolon when expecting a MeasV statement.

Unknown pin name: <string>.

The pin name in the MeasV or StimV statement does not match any pin name in the pin list section.

Use clock divisor <number> to lengthen the digital test window.

If the digital test time cannot be sufficiently lengthened by increasing the “to time” up to a maximum of F16, the tester will then calculate the minimum clock divisor that will permit the analog measurement to be made within its bounds. This number will be reported to the user. The calculated clock divisor will not be automatically entered into the test page. The user must insert the new value (if desired).

V**Validate: Can't determine wait time!**

Validate was not able to determine a reasonable wait time for the test. Measurements that are severely unstable can produce this result. If possible, loosen the tolerance or develop a longhand test.

Validate: Can't measure final configuration!

Validate cannot measure the final configuration. There are a variety of situations that will produce this result, however, the resulting test step is almost always a bad test.

Validate: Can't measure guarded configuration!

Validate was not able to measure a guarded configuration and Validate was aborted at this point. The autoranging or overranging features of the analog test board (ATB) may produce this result.

Validate: Can't settle initial No_guard measurement!

The initial phase of Validate was unsuccessful in obtaining a reasonably stable measurement from the unguarded situation. Validate was aborted.

Validation disabled for this Page.

An attempt was made to run Validate on a test page where Validate was disabled in the Flags menu of the test step. No modification was made to the test. If you wish to run Validate on this test, enable the page for Validate.

Vector Burst timed out.

The digital vector test was not completed within the time limit specified in VP Timeout in the Setup/Environment menu. Increase this time or disable VP Timeout in the Options/Control menu Test Properties. If the problem continues, there may be a hardware error in communicating with the Vector Processor board.

Vector file <"string"> does not exist!

The ICT program refers to a vector file of name <string> that does not exist in the program directory. This test will need to be regenerated.

VP: Backdrive Timeout.

With a VP2, the digital test took longer to execute than the Backdrive time specified in the Setup/Environment or the Edit/Header/PRGMVARS menu.

VP: Bad Memory Request.

A request was made for zero or less bytes of memory on the VP.

VP: Heap Error.

The memory allocator on the VP is corrupted.

VP: Out of Memory.

The VP cannot satisfy a memory request even after freeing the unused vector tests.

VP List Data of <size> entries exceeds VP capabilities.

The vector could not be compressed. The VP has 1 megabyte of memory; this vector's pin change data is greater than 1 megabyte.

W**Warning: Component <Name><ID> exceeds maximum cool down time of 65 seconds.**

The cool down time is calculated from Duty Cycle, which is an Environment Setup variable, and the test time of this burst exceeds the maximum range of hardware timers. The cool down time will be set to its maximum, 65 seconds, and will permit the test to continue.

Warning: could not remove tmp file: <string>!

System could not remove the file: <string>. This is a temp file and may be removed by the user at any time.

Warning: Digital guard data out of date, run Pgen/Dig Guard to add new guards.

Changes have been made in the test program that may not be reflected in the digital guard file. Go to the Program Generate menu and run Dig_Guard to update the guards in the test program using the digital guard file.

If you encounter this message while running an FST program, it is likely that digital guards need to be applied to your ICT program. To correct the problem, return to the ICT program, use Pgen/Update Dig Guards, and then return to FST.

Warning: No component database.

In the Program Header section, the Shorts ID Depth setting is greater than zero. This setting instructs the tester to compare the date the component database was last modified to the date the test program was last modified. By comparing dates, the software could not find a component database file. Select Update from the Program Generator menu to create a component database from the test program.

Warning: parsed string truncated to <number> characters!

Expansion of 18xx macros (such as %ALPHA, %INT, and so forth) exceeded <number> characters.

WaveScan Calibrate failed.

The calibrate function may have failed because of a WaveScan hardware failure or WaveScan's inability to find a frequency with a sufficient signal-to-noise ratio for reliable testing.

WaveScan Parameter file revision mismatch.

All WaveScan parameter files contain a revision. Either you are using incompatible versions of WaveScan and system software, or the file has become corrupt.

WaveScan Validate Parameter file error.

General file read error of the Validate Parameter file. This occurrence may be caused by an incomplete or corrupt file.

C-SCAPE ERROR
MESSAGES

ERROR

DESCRIPTION

D

disp_Clear 1011:

Passed bad display manager.
The device interface has not been initialized.

disp_GetCurrent 1013:

Passed bad display manager.
The device interface has not been initialized.

disp_GetFont 1001:

Passed bad display manager.
The device interface has not been initialized.

disp_GetHeight 1009:

Passed bad display manager.
The device interface has not been initialized.

disp_GetWidth 1010:

Passed bad display manager.
The device interface has not been initialized.

disp_Repaint 1012:

Passed bad display manager.
The device interface has not been initialized.

disp_SetCurrent 1014:

Passed bad display buffer.

draw_field 20:

Illegal call. This is an internal error.

F

field_Close 1:

Passed bad field.

field_FirstChar 2:

Passed bad field.

field_GetData 13:

Passed bad field.

field_GetMerge 3:

Passed bad field.

field_GetRecord 4:

Passed bad field.

field_LastChar 6:

Passed bad field.

field_NextChar 7:

Passed bad field.

field_NextChar 8:

Passed bad record position.

field_PrevChar 9:

Passed bad record position.

field_SetChar 10:

Bad field position or field contains no write-
able positions.

field_SetChar 11:

Passed bad field.

field_SetData 14:

Passed bad field.

field_SetString 12:

Passed bad field.

K

kb_Idle 1008:	Passed bad display manager. The device interface has not been initialized.
kb_Record 1007:	Passed bad display manager. The device interface has not been initialized.

M

menu_DeleteField 82:	Passed bad menu.
menu_DeleteField 83:	Passed bad field number.
menu_DeleteRows 84:	Passed bad menu.
menu_DeleteRows 85:	Passed bad arguments.
menu_Destroy 40:	Passed bad menu.
menu_Flush 41:	Passed bad menu.
menu_GetDownField 42:	Passed out-of-range field number.
menu_GetDownField 43:	Passed bad menu.
menu_GetLeftField 44:	Passed out-of-range field number.
menu_GetLeftField 45:	Passed bad menu.
menu_GetRightField 46:	Passed bad menu.
menu_GetUpField 47:	Passed out-of-range field number.
menu_GetUpField 51:	Passed out-of-range field number.
menu_GetUpField 52:	Passed bad menu.
menu_InsertRows 86:	Passed bad menu.
menu_InsertRows 87:	Passed bad arguments.
menu_MoveField 80:	Passed bad menu.
menu_MoveField 81:	Passed bad field number.
menu_OK 54:	Menu has been closed. An attempt to use a menu object that has already been closed has occurred. Either do not close the menu, or open a new menu.
menu_Printf (percent)(menu.c) 56:	Unanticipated conversion character. The percent (%) character has been used in a format string without a recognizable conversion character (such as "%z").
menu_Printf (repeat) 59:	Bad repeat arguments. The repeat syntax has been given bad arguments (either no comma was found or the repeat string was of zero length).

menu_Printf 53:	Percent buffer overflow. A percent (%) expansion within a format string larger than 511 characters has occurred. These must be less than 511 characters.
menu_Printf 55:	@c arguments too long. The number of characters within the brackets of an @c command has exceeded 127.
menu_Printf 58:	Unexpected end of format string. The end of the format string has been reached by the parser before it was expected. Usually this occurs when the closing ']' of a command is missing. This can also occur when an '@' is found with no command or quoted character following it. Quoting the '@' with another '@' will usually solve the problem.
menu_Printf 60:	Field too long. A field longer than 1000 (MAX_FIELD_LEN) characters was encountered.
menu_Printf 61:	Passed bad menu.
menu_Printf 62:	@p arguments too long. The number of characters within the brackets of an @p command has exceeded 127.
menu_Printf 63:	Repeat arguments too long. The number of characters within the brackets of a repeat '@[' command has exceeded 127.
menu_Printf 64:	Out-of-range repeat count. The repeat count is either negative or greater than 127.
menu_Printf 65:	Unanticipated state. This is an internal error in the menu_Printf parser.
menu_Printf 70:	Missing '[' in @a command.
menu_Printf 71:	@a arguments too long. The number of characters within the brackets of an @a command has exceeded 127.
menu_Printf 74:	Bad '@' command. An invalid character was found following '@' in the format string.
menu_Printf 75:	Missing '[' in @p command.
menu_Printf 76:	Missing '[' in @c command.
menu_Printf 77:	Missing '[' in @f command.
menu_Printf 79:	Field name too long.
menu_SetTB 88:	Passed bad menu.
menu_SetTB 89:	Passed bad text.
menu_SwapFields 90:	Passed bad menu.

menu_SwapFields 91:	Passed bad arguments.
O	
obj_Open: OBJM_GETDATSIZE 1020:	Uninitialized od. An object class has not set its od_size.
obj_Open: OBJM_GETDATSIZE 1021:	Uninitialized xd. An object class has not set its xd_size.
obj_Open: OBJM_GETDATSIZE 1022:	Uninitialized id. An object class has not set its id.
OGL 1100:	Call to uninitialized OGL routine. An ogl_routine has been called without calling ogl_Init.
S	
scroll_adjust 136:	Passed bad sed. scroll_adjust is an internal C-scape routine called during movement between fields.
sd_goto_char 22:	Passed negative field position. (sd_goto_char is an internal C-scape routine used to move the cursor within a field.)
sd_goto_char 23:	Passed bad sed. (sd_goto_char is an internal C-scape routine used to move the cursor within a field).
sd_overwrite 24:	Passed bad sed. (overwrite is an internal C-scape routine used to write characters to a field.)
sd_refresh_field 27:	Passed out-of-range field number.
sd_refresh_field 28:	No fields in sed.
sd_scroll 30:	Passed bad sed.
sed_Clear 100:	Passed bad sed.
sed_Close 101:	Passed bad sed.
sed_DecChar 102:	Passed bad sed.
sed_DecField 103:	Passed bad sed.
sed_Draw 29:	Passed bad sed. (sed_Draw is an internal routine used by sed_Repaint, sed_RepaintFields, sed_Update, and sed_Update Fields.)
sed_GetColors 104:	Passed bad pointers. One or more of the pointers passed is equal to NULL.
sed_GetColors 105:	Passed bad sed.

sed_GetFieldColors 142:	Passed bad pointers. One or more of the pointers passed are equal to NULL.
sed_GetFieldColors 143:	Passed bad sed.
sed_GetPosition 109:	Passed bad sed.
sed_Go 108:	Passed bad sed.
sed_Goto 115:	No fields in sed.
sed_Goto 116:	Passed negative field number.
sed_Goto 117:	Passed bad sed.
sed_GotoChar 110:	Passed bad field position.
sed_GotoChar 111:	Passed bad sed.
sed_GotoChar 112:	Passed bad sed.
sed_GotoField 113:	Passed bad field number.
sed_GotoField 114:	Passed bad sed.
sed_gotoFirstField 106:	Passed bad sed.
sed_GotoLastField 107:	Passed bad sed.
sed_IncChar 118:	Passed bad sed.
sed_IncField 119:	Passed bad sed.
sed_MarkField 120:	Passed bad sed.
sed_MoveGField 121:	Passed bad direction. This is an internal error.
sed_MoveGField 122:	Passed bad sed. sed_MoveGField is the function that handles grid movement between fields (UpField, DownField, Etc.).
sed_OK 124:	Passed a closed sed. An attempt has been made to use a sed that was closed.
sed_Overwrite 123:	Passed bad sed.
sed_Page Up 128:	Passed bad sed.
sed_PageDown 126:	Passed bad sed.
sed_PageLeft 127:	Passed bad sed.
sed_PageRight 125:	Passed bad sed.
sed_Pop 25:	Passed bad sed.
sed_PullLeft 131:	Passed bad sed.
sed_PullRight 132:	Passed bad sed.
sed_PushLeft 129:	Passed bad sed.

sed_PushRight 130:	Passed bad sed.
sed_RepaintField 133:	Passed bad sed.
sed_ScrollDown 138:	Passed bad sed.
sed_ScrollLeft 139:	Passed bad sed.
sed_ScrollRight 137:	Passed bad sed.
sed_ScrollUp 140:	Passed bad sed.
sed_SetBorder 134:	Passed bad sed.
sed_SetColors 141:	Passed bad sed.
sed_SetHeight 144:	Passed bad sed.
sed_SetHeight 145:	Passed bad height.
sed_SetPosition 146:	Passed bad sed.
sed_SetPosition 148:	Passed negative position.
sed_SetRecord 147:	Passed bad sed.
sed_SetWidth 149:	Passed bad sed.
sed_SetWidth 150:	Passed bad width.
set_field (menu.c) 69:	Passed bad menu.
sfile_Close 191:	Passed bad screen file.
sfile_LoadSed 190:	Passed bad screen file. The screen file has not been opened correctly. Check the return value of sfile_Open for NULL.

T

tb_Close 170:	Bad text buffer.
tb_DrawLines 172:	Passed bad text buffer.
tb_Puts 175:	Passed out-of-range coordinates. tb_Puts is called by menu_Printf whenever it adds text to the text buffer. This error indicates that an attempt was made to place text at negative coordinates.
tb_Puts 176:	Passed bad text buffer.
tb_SetData 178:	Passed bad text buffer.
ted 179:	No cursor move method set. A ted cursor movement routine was used without first calling ted_StartWorking.

V

varsize 180:

Illegal use of field function. A field function whose varsize element is VAR_INVALID has been used with asked ir screen file.

W

win_Class:WINM_CLOSE 1019

Window has bad parent.

win_Close 1018:

Passed bad window.

win_MakeCurrent 1015:

Found bad current window.

win_MakeCurrent 1016:

Passed bad window.

win_MakeCurrent 1017:

Called within a mouse handler.

wmgr_CreatWin 1004:

Passed bad display manager. The device interface has not been initialized.

wmgr_ExposePixBox 1005:

Passed bad display manager. The device interface has not been initialized.

wmgr_ExposePixBox 1006:

Bad tiler data pointer. The window manager could not allocate memory.

wmgr_Init 1002:

Passed bad display manager. The device interface has not been initialized.

wmgr_Wrapup 1003:

Passed bad display manager. The device interface has not been initialized.

▼▼▼

B COMMAND LINE FUNCTIONS

This section of the appendix lists the command line options that help you to access certain functions and processes directly from the system prompt at start-up. You can enter a command line option as an attribute after typing “18xx” at the system prompt.

IMPORTANT: Command line options are case sensitive.

Command Line Options

Command line options are used as a shortcut method to take you directly from the start-up system prompt to a specific function or process without having to step through a menu interface. Not all functions or processes can be accessed in this way.

To use the command line function, type the appropriate option and, if applicable, its suboption immediately following 18xx. Note that command line options are case sensitive.

Usage: C:\>18xx [/v /h /o /r /P /N /C /T /I /TPD]

/v

Displays current OS Version.

/h

Displays the help menu, listing available command line options, and returns to the system prompt.

The menu is displayed for reference only; you must type 18XX at the system prompt, followed by the appropriate option.

/o

Overrides any GFI start-up processes.

/r

Goes directly to program run.

/P

Runs the Program Generator Process. The Program Generator Usage menu is displayed listing available suboptions. Usage: 18xx /P [-n -b -g -p -u -li -lc -ln -lt -d -g]

The menu is displayed for reference only. “Press Key to continue” returns to the system prompt.

- n New (cleans out old files).
- b Builds a component database from IPL.DAT file.
- g Generates the ict program.
- p Combines parallel components (ex. RCs).
- u Updates component database from ict program.
- li Creates the IPL list file.
- ln Creates the node list file.
- lc Creates the component list file.

- lt Creates the topology list file.
- d Runs digital guarding algorithm.
- q Turns on quiet mode for batch processing.

/N

Runs NodeFinder.

/C

Runs Program Convert.

/T

Runs the Template Library Menu's The Template Tools Usage menu is displayed listing the available suboptions. Usage: 18xx /T -l [-m -p -a -al -av -x -ivl -ra -wa]

The menu is displayed for reference only. "Press Key to continue" returns to the system prompt.

- l<lus> One of these <local, user, system> must be specified.
- m : Invokes the Template Utilities Menu.
- m [-c -i]: Does not specify -l if using -c (create), -i (permission).
- p [brc]: Pack Template Library, [backup, restore, check].
- a <ttm name>: Adds Template.
- al <dev name>

<alias name>Adds New Alias.
- av <asc name> Adds Vector Template.
- x Extracts Modified Devices.
- ibl <ivl name> Adds IVL to Library.
- ra Reads file ALIAS.CFG to update the template database.
- wa Writes to file ALIAS.CFG from the template database.

/G

Runs the Global Editor's "Execute a file" process. Modifies ICT.TEST as specified by the agents in the input file.

Usage: 18xx /G [[-i | -o | -e | -w] file name] [-h] [-?] [-l] [-p] [-q] [-v]

(suboption -? or -h displays a menu for reference only. Use the Page-Up or the Page-Down key to view additional menu text; press the ESC key to return to the system prompt.)

- d Turns Off the CRT display of the report activity. The default is On.
- e <file name> Specifies the error output file name. The default is EXCEPT_LST.
- ? or -h Displays Help message. Outputs a synopsis of Global Editor usage and options.
- i <file name> Specifies the input file name. The default input file name is ICT.GED.
- l Lists qualified test steps. Causes Global Editor to ignore any actions and lists the test steps that can be modified by the agents in the input file.

Note that a test step will be listed more than once if more than one agent can modify it.

- m <mode> Turns on Output Mode, where mode equals Append or Overwrite. The default is Append.
- p Parses the input file and exits.
Checks that the input file is syntactically correct.
- q Turns On quick mode.
The default is Off. When quick mode is On, each section's agent list is scanned, and all agents that modify test steps in that section are processed. (This save time since the information for each section is read only once.) When quick mode is Off, then the agent list is processed in order. (In this case, each section must be opened and scanned to see if the agent modifies any test steps. This step must be repeated for each agent.)
- r Reports all activity to the report output file and to the CRT (if -d is not used).
Reports all test steps that qualified, whether or not they were modified.
- v Turns on Verbose. Echoes the input file to the report output file.
- w <file name> Specifies the warning output file name. The default is EXCEPT_LST.

/TDP <path>

Sets the path for the selected board. Use this option with other command line options, such as Pgen or Convert, to specify the test program to operate on.

▼▼▼

Symbols

%ALPHAn*, output macro 3-20
 %BOARD*, output macro 3-20
 %CLEAR_ALPHAn, output macro 3-20
 %CLEAR_ELAPSED, output macro 3-20
 %CLEAR_INTn, output macro 3-20
 %DESC*, output macro 3-20
 %ELAPSED, output macro 3-20
 %ENODE, output macro 3-20
 %FAILURES, output macro 3-20
 %FNODE, output macro 3-20
 %GNODE, output macro 3-20
 %HIGH, output macro 3-20
 %ID*, output macro 3-20
 %IEEE*, output macro 3-20
 %INTn, output macro 3-20
 %LOW, output macro 3-20
 %MEAS, output macro 3-20
 %NAME*, output macro 3-20
 %OPTn, output macro 3-20
 %STAT, output macro 3-20
 %TIME, output macro 3-20

Numerics

1800 Operating System Drive
 Path Data setup field 2-2
 18xx.CGF 2-2
 18START.BAT, file description 1-8
 18xx directory system 1-6
 18xx file system 1-6
 18XX.CFG, file description 1-8
 18XX.ERR, file description 1-8
 18XX.EXE, file description 1-7
 18XX.ICO, file description 1-8

A

Abort Count Reached, System flag 4-7
 Abort on Fail Count
 specifying in PRGMVARS 3-15
 test flow control 4-5
 Abort on Fail Count, variable function 3-15
 absence of component, test for 3-48
 Action Attributes
 Global Editor pop-up, example 9-64
 Agent Input file
 creating in Global Editor 9-61
 ALCONV.EXE, file description 1-7
 ALIAS.CFG
 ASCII equivalent of ALIAS.LNK 8-10
 example of file 8-11
 modifying 8-10
 modifying from DOS 8-11
 ALIAS.CFG, file description 1-9
 ALIAS.LNK, file description 1-9
 aliases
 marking for deletion 8-7
 undeleting 8-8
 ALL_NODES, used in Global Editor 9-51

Allprint

blocking output 3-18
 effects on Tokenlog content 2-8
 enabling Allprint string for failures 3-17
 Exclude Allprint Device(s) 3-18
 Allprint Message description 3-17
 Altering Template Revision menu 8-9
 Alternate Program Directory
 changes affecting 2-3
 Path Data setup field 2-2
 analog debug LED 9-8
 Analog Failure, System flag 4-7
 analog guarding
 3-wire active guard correction 5-21
 3-wire passive guarding 5-20
 3-wire, sources of error 5-20
 4-wire correction 5-22
 5-wire mode 5-22
 6-wire mode 5-23
 active guard correction 5-21
 active guarding limitations 5-20
 default mode (3-wire) 5-21
 guard ratio limits 5-19
 Precise mode 5-23
 reducing error currents 5-21
 requirements for 4-wire mode 5-22
 requirements for 6-wire mode 5-23
 theory 5-18
 when to use 5-wire mode 5-22
 when to use Precise mode 5-23
 analog measurement
 averaging to increase test stability 5-25
 theory 5-17
 when to use Higuard 5-24
 analog switching 5-13
 analog templates
 creating 8-17, 9-5
 importing to inappropriate sections 8-17
 using 8-18
 See also templates
 analog test
 3-wire guarding 5-18
 analog guarding, theory 5-17
 debugging using test jack signals 9-11
 measurement theory 5-17
 techniques 5-4
 using squelch times function 5-25
 using wait times 5-25
 Analog Test Board. *See* ATB
 analog test properties
 Averaging, for measurements 3-32
 Current Page of worksheet 3-32
 Guard Mode, type 3-32
 Guard, pin or node 3-32
 High, calculated tolerance 3-32
 Higuard, reduce thermal EMF 3-32
 Indicators, page deletion 3-32
 Low, calculated tolerance 3-32

- analog test properties *continued*
 - Measure, pin or node 3-32
 - Options 3-32
 - Precise, for wire modes 3-32
 - Scale, Value modifier 3-32
 - Squelch, ms 3-32
 - Stimulus, pin or node 3-32
 - Test Type, worksheet test configuration 3-32
 - Value, expected 3-32
 - Wait, ms added to default 3-32
 - Wire Mode, test configuration 3-32
- analog wire modes
 - 3-wire, Precise Off 5-28
 - 3-wire, Precise On 5-29
 - 4-wire mode 5-30
 - 4-wire Precise Off 5-30
 - 4-wire, Precise On 5-31
 - 5-wire mode 5-32
 - 5-wire, Precise On 5-33
 - 6-wire mode 5-34
 - 6-wire, Precise On 5-35
 - configurations 5-27
- ANALOG.HLP, file description 1-8
- ANALYZE.BAT, file description 1-8
- ANTEMPL.LIB, file description 1-9
- APC (Auto Probe Check)
 - description 3-14
 - setup PRGMVARS, execution control 3-14
 - System Failure flag 4-7
 - test step control option 3-48
- Append (Utilities menu)
 - duplicating test steps 3-22
- application strategy, developing 5-1
- ASCII vector file structure 8-20
 - constraints 8-25
 - AVAIL 8-25
 - CONNECTED 8-25
 - HIGH 8-25
 - INDEP 8-25
 - IRREL 8-25
 - LOW 8-25
 - TRI 8-25
 - UNAVAIL 8-25
 - constructs 8-24
 - BLOCK 8-25
 - ENTRY 8-25
 - REPEAT 8-25
 - SELECT 8-25
 - SELECTONE 8-24
 - SEQUENCE 8-24
 - SET 8-24
 - Disable section 8-26
 - file sequence 8-20
 - format examples 8-29
 - homing loops 8-39
 - mixed vector 8-40
 - Nested Select 8-35
 - Selectone/Select 8-34
 - Sequential Select 8-37
 - vector 8-29
 - vector in IVL format 8-32
 - format specification 8-27
 - Guard section 8-26
 - IVL format 8-26
 - nesting constructs 8-24
 - saving space in long vectors 8-32
 - test configuration statements 8-22
 - CLOCK 8-23
 - MAXRATE 8-22
 - MDELAY 8-22
 - TERM 8-22
 - THRESH 8-22
 - vector data section 8-23
 - Vector section 8-23
- ASIC test
 - board hookup constraints 10-48
 - development strategy 10-38
 - disable requirements 10-46
 - measurement timing 10-47
 - minimum pulse width 10-47
 - non-synchronous clocking 10-47
 - pattern coverage guidelines 10-48
 - pattern design criteria 10-46
 - pattern restrictions 10-47
 - pin coverage 10-48
 - preferred overdrive state 10-46
 - self initialization 10-48
 - standard format 10-47
 - synchronous clocking 10-47
 - tristate measurement 10-47
- ASYNCSYS
 - enabling com ports 2-20
 - file description 1-8
- ASYNCSYS.TXT, file description 1-8
- ATB
 - drive terminals 5-26
 - sense terminals 5-26
 - test technique
 - Test I Stim V 5-26
 - Test V 5-26
 - Test V Stim I 5-26
 - Test V Stim V 5-26
 - test techniques 5-26
- ATBCAL.EXE, file description 1-7
- ATMPLCON.EXE, file description 1-7
- attribute-value pairs, used in Global Editor 9-51
- Auto Probe Check. *See* APC
- AUTO_RANGE, used in Global Editor 9-52
- Auto-Convert
 - default permissions 1-17
 - login permissions 1-16
- AUTOEXEC.BAT file, editing 1-6
- Automatic Program Conversion, permission 1-26
- Auxiliary ports, Device & Channel Data setup 2-3
- AVERAGING, Global Editor attribute-value 9-52

B

- backdrive
 - effects of test head controller 10-15
 - theory 10-13
 - timeout 10-13
- Backdrive Timeout 2-10
 - description 3-14
 - specifying in PRGMVARS 3-14
 - test step control 3-48
- BD.VEC, file description 1-8
- bidirectional pins
 - Fault Inject 9-18
 - Gray code 9-18
 - vector 9-17
- board damage prevention
 - powering logic ICs vii
 - Section Abort in debug mode vii
 - testing at 5V instead of 3V vii
- Board Description, variable function 3-15
- Board Fault Coverage 9-24
- Board Name field, location on worksheet 3-5
- board power
 - power-down sequence 5-7
 - power-up sequence 5-7
 - Test Type restrictions 5-6
- board power, relationship to Test Types 3-8
- board program
 - name syntax 1-22
 - See also* test program
- board-under-test
 - evaluating test strategy 5-5
 - power warning 9-36
 - specifying voltage requirements 9-73
- Boundary Scan Intelligent Diagnostics. *See* BSID
- branching
 - setting up 4-14
 - specifying conditions 4-8
- Brd_Power. *See* board power
- BSID
 - BICT software 3-49
 - Failure Reporting
 - setting up 3-49
 - test step control 3-48
 - VIT software 3-49
- Build, creating new entries in IPL.DBF 7-12
- bus test, developing 10-7
- BusScan, enabling/disabling 3-48
- BW_LIMIT, used in Global Editor 9-52

C

- C, input list token 6-10
- C.1 system software
 - C.1 input list after transversion 6-28
 - C1TODX, input list transversion 6-26
 - creating program with C.1 input list 6-25
- CAD data, in program and fixture development 5-1
- CAGE_SCN.DAT, file description 1-8
- calculator. *See* Tolerance Calculator

- Calibration
 - default permissions 1-17
 - login permissions 1-16
- Calibration utility 9-40
- Cancel Abort, System flag 4-6
- Cancel key (F4) 4-13
- Cancel, stopping tests 4-12
- Cap Phase test
 - enabling Validate 2-18
 - General Variables, Learn Cap Phase Offsets 3-15
- capacitance. *See* system capacitance
- capacitor testing
 - 3-wire analog test mode 3-36
 - 4-wire analog test mode 3-36
 - 5-wire analog test mode 3-36
 - 6-wire analog test mode 3-36
 - converting to longhand 9-5
 - discharging capacitors 3-35
 - editing Component Properties 3-26
 - editing Test Properties 3-30
 - enabling sampling in Validate 2-18
 - generating guard nodes with Validate 3-35
 - getting stable test results 3-35
 - Guard Node field 3-35
 - guarding 5-19
 - improving stability with Validate 2-16
 - Measure field 3-35
 - squench times 3-35
 - stimulus and measurement 3-34
 - Wire Mode field 3-36
- CapScan Ratio, PGEN.CFG 7-9
- CapScan testing
 - enabling Validate 2-20
 - overview 5-4
- card cage scan. *See* Hardware Configuration
- categories 3-8
- category. *See* component category
- Chain Depth, setting up 4-4
- chaining. *See* program chaining
- Change Permissions, logging in 1-16
- channels, number available 5-13
- chip damage 10-14
- Clean
 - preventing duplicate test steps 7-14
 - removing existing test program files 7-12
- Clear test page set up 2-10
- CLKCAL.EXE, file description 1-7
- CMPHDR.TXT, file description 1-8
- CNTTHRESH, syntax 7-7
- Com Parameters
 - baud rate 2-21
 - input timeout 2-21
 - output timeout 2-21
 - parity 2-21
 - protocols 2-21
 - RTS/CTS 2-21
 - stop bits 2-21
 - word length 2-21

- Combine Parallel command 7-15
 - using 7-16
 - viewing results 7-16
- Combine Parallel, PGEN.CFG syntax 7-9
- combining parallel components 7-15
 - automatic mode 7-16
 - editing existing program 7-16
 - globally 7-16
 - manual mode 7-16
 - manually disabling source component 7-17
 - specifying components to combine 7-15
- COMMAND.COM, use of 2-14
- communications channels
 - configuring 2-4
 - disabling 2-4
 - setting up 2-3, 2-4
 - structure 2-4
- communications ports
 - input timeout 2-22
 - parameters window 2-21
 - setting baud rate 2-21
 - setting input timeout 2-21
 - setting output timeout 2-21
 - setting parity 2-21
 - setting Stop Bits 2-21
 - setting word length 2-21
- COMP.LST, file type and description 1-10
- COMP_NODES, Global Editor attribute-value 9-52
- COMP_VALUE, Global Editor attribute-value 9-52
- component category
 - Brd_Power (Board Power) 3-7
 - Digital 3-7
 - execution sequence 4-1
 - Header 3-7
 - Intc (Interconnect) 3-7
 - Linear 3-7
 - Passive 3-7
 - PwrOff 3-7
 - Quit 3-7
 - selecting from Component Select menu 3-6
 - Semi 3-7
 - Trailer 3-7
- component pin types, fault coverage 9-18
- Component Properties
 - creation 3-25
 - data known about a component 3-25
 - Desc field 3-28
 - Device Type field 3-28
 - ID field 3-27
 - Name field 3-27
 - navigation 3-26
 - Number of Pins field 3-29
 - Tolerance field 3-29
 - Value field 3-28
- component search function 3-10
- Component Select menus
 - Add 3-11
 - Copy 3-11
 - Delete 3-11
 - Disable 3-11
 - Edit 3-11
 - Move 3-11
 - Utilities 3-11
 - Validate 3-11
- Component Select window
 - Header example 3-9
 - selecting components 3-8
- component test step, deleting 3-11
- component token
 - D (Diode) 6-14
 - GNPN (Beta test) 6-15
 - GNPN (Beta test) 6-15
 - QN (Junction Transistor) 6-15
 - QP (Junction Transistor) 6-15
 - RP_DB (Rpack-based resistor, dual) 6-13
 - RP_DI (Rpack, dual in-line) 6-12
 - RP_DT (Rpack-terminated resistor, dual) 6-14
 - RP_SB (Rpack-based resistor, single) 6-12
 - RP_SI (Rpack, single in-line) 6-11
 - RP_ST (Rpack-terminated, single) 6-13
 - U (Unknown) 6-15
 - Z (Zener) 6-14
- conditional I/O, setting of flags in execution 4-2
- CONFIG.SYS file, editing 1-6
- Constructed Agent Display, in Global Editor 9-65
- CONT, input list token 6-9
- continuities, Interconnect Learn 7-12
- continuity threshold, PGEN.CFG syntax 7-7
- control characters
 - a (alarm) 3-21
 - authorized 3-21
 - cnnn (color) 3-21
 - f (form feed) 3-21
 - in system variables 3-20
 - n (newline) 3-21
 - nnn (ASCII numeral) 3-21
 - t (tab) 3-21
- controlling Gray code bursts 3-49
- controlling test execution with Option flags 4-14
- CONVD2.EXE, file description 1-7
- converting programs 1-14
 - pre-F.0 DeltaScan database 1-26
 - unconverting to earlier software version 1-26
 - with corrupted files 1-26
- copying component sections 3-11
- Count
 - measuring frequencies 10-1
 - See also* Digital Test
- counter tests, developing 10-10
- CRC (Cyclic Redundancy Check), calculating 9-38
- CRT channel, configuring 2-4
- CRT key (F8) 4-13
- CSCAN, input list token 6-18
- CSNRATIO syntax 7-9

- current directory
 - finding name of 1-12
 - Path Data setup field 2-2
- custom diagnostics, running 9-40
- D**
 - D, input list token 6-14
 - D18XX.EXE, file description 1-7
 - Datalog
 - assigning channel to file name 2-7
 - customizing Begin and Valid messages 2-7
 - description 3-17
 - enabling in PRGMVARS 2-7
 - master program/subprogram datalogging 4-4
 - report tag 2-7
 - setting path 2-7
 - setup example 2-7
 - DATALOG.DAT
 - Device & Channel Data field 2-3
 - file type and description 1-11
 - DB, input list token 6-16
 - DCHG, input list token 6-8
 - DCTHRESH, syntax 7-8
 - DCTIMEOUT, syntax 7-8
 - debugging tests
 - resistor test 9-11
 - small capacitors 9-9
 - using Repeat in digital test 9-11
 - using Section Run mode 4-12
 - using SOF 4-12, 9-11
 - using test jack panel 9-7
 - using test jacks in digital debug 9-10
 - Z850/Z875 tests in 18xx 10-52
 - Default Program Directory, Environment Data 2-2
 - defaults
 - ASCII vector file, test configuration 8-22
 - Averaging in analog tests 3-37
 - C1TODX filename 6-25
 - capacitor squelch 3-36
 - communications parameters 2-21
 - Datalog channel filename 2-7
 - default login password 1-15
 - default permissions for users 1-17
 - DeltaScan pin number 6-18
 - Digital Tracer chip type 9-13
 - Digital Tracer device orientation 9-13
 - discharge threshold voltage 6-8
 - Docuenter, sections to be documented 9-43
 - effect of adding pages to Test Properties 9-4
 - Fault Inject, single pin state 9-21
 - FrameScan pin number 6-17
 - GFI Refresh option 2-15
 - Guard Mode 3-32
 - IPL token default section
 - APC 6-9
 - ATMPL 6-15
 - C 6-10
 - CONT 6-9
 - D 6-14
 - DB 6-16
 - DCHG 6-8
 - GND 6-6
 - GNPN 6-15
 - GNPN 6-15
 - Header 6-6
 - IC 6-16
 - IPL 6-6
 - J 6-9
 - L 6-10
 - OPN 6-9
 - PB 6-8
 - PBUS 6-8
 - POT 6-11
 - PWR5 6-7
 - PWRA 6-7
 - PWRA_F 6-8
 - PWRB 6-7
 - PWRB_F 6-8
 - QN 6-15
 - QP 6-15
 - R 6-10
 - REO 6-11
 - RP_DB 6-13
 - RP_DI 6-12
 - RP_DT 6-14
 - RP_SB 6-12
 - RP_SI 6-11
 - RP_ST 6-13
 - SC 6-10
 - SH 6-9
 - Trailer 6-6
 - U 6-15
 - VCLUST 6-17
 - VEC 6-16
 - VIMG 6-16
 - Z 6-14
 - keyboard selection letter color 1-2
 - line transceiver measurement window 10-11
 - listen windows 10-1
 - location of default passwords 1-18
 - logging in, startup user Id 1-19
 - measurement timing, ASIC test patterns 10-47
 - MultiScan Discharge 3-15
 - output devices for test failure reports 2-6
 - overriding defaults with PGEN.CFG 7-3
 - Pack menu Options 8-7
 - PEP update 7-11
 - PRGMVARS 3-14
 - Report All Vector Failures 3-51
 - resistor averaging 3-32
 - resistor wait 3-32
 - switch test type 10-3
 - system library path 8-1
 - vacuum in PRGMVARS 3-14
 - Validate acceptance range 2-17
 - Validate averaging 2-18

defaults *continued*

- Validate FrameScan minimum threshold 2-20
- Validate WaveScan minimum threshold 2-20
- Validate/Samples 2-18
- WaveScan pin number 6-17
- While 9-9

DeltaScan parameters, PGEN.CFG 7-10

DeltaScan testing

- database conversion 1-25
- Dscan Hard Fixed Thresh variable function 3-16
- enabling Validate 2-19
- fault coverage 9-32
- Fixed Threshold variable function 3-15
- overview 5-3
- retrying pin pairs 3-16
- running all possible pin pairs 3-16
- setting delays 3-16
- Short Power/Ground via G-pole 3-15
- specifying reference node 7-8
- topology 7-19
- Verify Pwr/Gnd Shorting variable function 3-15

DESC, Global Editor attribute-value 9-53

device damage, troubleshooting 10-14

Device Type

- editing in Component Properties 3-29
- influence on Step Worksheet generation 3-29
- relationship to Test Type 3-7

DEVICE.LST, file description 1-9

DEVICE_TYPE, Global Editor attribute-value 9-53

DFP

- accessing from 18xx interface 10-21
- Communications channel setup 10-21
- communications port parameters 10-22
- generating Test Properties 10-23
- overwriting files 10-24
- PRGMVARS Communication channel field 10-21
- PRGMVARS Source Directory path 10-21
- reboot delay 2-11
- Result Text box 10-24
- sending arguments to ptprog.exe 10-24
- setting up in PRGMVARS 10-21
- specifying Com port timeout 10-24
- specifying in PRGMVARS 3-15
- specifying source directory 10-24
- test properties 10-24
- updating DFP 10-24

DFP Configuration, variable function 3-15

DFP, input list token 6-19

Diagnostics

- default permissions 1-17
- Device & Channel Data field 2-3
- login permissions 1-16
- specifying method 9-40

DIAPRISM.IMG, file description 1-7

Dig Guard command 5-36

DIG_CLOCK_DIV_GC, attribute-value 9-53

DIG_CLOCK_DIV_VEC, attribute-value 9-53

DIG_CLOCK_SOURCE, attribute-value 9-53

DIG_HIGH_THRESH, attribute-value 9-53

DIG_LOGIC_LEVEL, attribute-value 9-53

DIG_LOW_THRESH, attribute-value 9-53

DIG_MEAS_DELAY, Global Editor attribute-value 9-53

DIG_TERMINATOR, attribute-value 9-53

DIGCLK, syntax 7-7

digital bursts

- burst time control 10-13
- exceeding maximum limits 10-13
- viewing results 4-12

digital component, relationship to test structure 3-8

Digital Function Processor. *See* DFP

digital guarding

- Dig Guard command 5-36
- digital guard file 5-36
- Gray code test 5-37
- guard header data 5-36
- guard record 5-36
- guarding algorithm 5-37
- purpose 5-36
- synchronizing stimulus with guard 5-37
- templates supporting guard information 5-36
- testing for conflicts 5-37
- testing for stimulus node conflicts 5-37
- time stamp 5-36
- vector guards in Gray code test 5-38

Digital HiCheck

- setup in PRGMVARS 3-14

Digital HiCheck, description 3-14

digital pullups LED 9-8

digital stimulus hierarchy 5-40

digital test

- debugging using test jack signals 9-10
- displaying results of bursts 9-10
- effect of relays 5-17
- evaluating test quality 9-17
- isolating devices (guarding) 5-36
- pin behavior/stimulus hierarchy 5-40
- techniques 5-4
- theory 5-36
- using Nodefinder 9-36

digital threshold, PGEN.CFG 7-7

Digital Tracer

- description 3-14
- device orientation 9-13
- global set up/PRGMVARS 9-12
- imaging ability 9-12
- overriding global default 9-16
- Second Pass function 9-15
- setting up in PRGMVARS 3-14, 9-12
- Setup window 9-13
- specifying chip type 9-13
- test step control 3-48
- testing connectivity of leads 9-11
- using with C.1 programs 9-15
- variable function 3-14

DIGITAL.HLP, file description 1-8

DIGSTIM ENUM, syntax 7-6

- DIGTHRESH syntax 7-7
 - diode variable, PGEN.CFG 7-7
 - directories
 - changing 1-22
 - changing using Toggle TPD 1-22
 - MOS 1-6
 - PGT 1-6
 - program directory variables 2-3
 - TPD, test program directory 1-6
 - updating program directories 2-3
 - DISABLE.IVL, file type and description 1-10
 - DISABLE.VEC, file type and description 1-10
 - DISABLEPWR syntax 7-8
 - disables
 - in Gray code tests 5-38, 5-40
 - in vector tests 5-40
 - putting board in preferred state 5-38
 - vector disable timing 5-38
 - disables and guards, order of execution 5-39
 - disabling
 - globally 9-63
 - power step worksheets 7-8
 - test step 3-11
 - to improve fault coverage, MultiScan test 9-32
 - with Qualifier attribute 9-63
 - Discharge Failure, System flag 4-6
 - Discharge Method
 - Discharge Section Rerun (DschrSect) 5-7
 - None 5-8
 - Power Nodes (PwrNodes) 5-7
 - PwrNodes 3-15
 - specifying in PRGMVARS 3-15
 - Discharge Method, variable function 3-15
 - discharge parameters
 - PGEN.CFG 7-8
 - DISCHGENVAL variable syntax 7-10
 - DISCHGENVALvariable, PGEN.CFG 7-10
 - Documenter
 - format 9-44
 - Options window 9-41
 - setting up 9-41
 - specifying output type 9-43
 - DOS
 - exiting to the DOS environment 1-5, 1-21
 - listing directory files 1-22
 - using for test development 9-40
 - DOS command line key (F11) 4-13
 - DOS EDITOR, Path Data setup field 2-2
 - DOT.RST, file type and description 1-11
 - DR1 Emulation Mode, variable function 3-15
 - DR2d boards, specifying power source 3-15
 - DRFIXED
 - syntax 7-6
 - driver/receiver
 - architecture 5-16
 - Hardware Configuration 9-73
 - master relays 5-14
 - switch matrix 5-13
 - DRVRBL
 - syntax 7-6
 - DS_CONV.EXE, file description 1-7
 - DSCAN, input list token 6-18
 - DSCAN.BAK, file type and description 1-11
 - DSCAN.DB, file type and description 1-11
 - DSCAN.EXE, file description 1-7
 - DSDEFTHRESH syntax 7-10
 - DSFIXPIN syntax 7-10
 - DSFIXTHRESH syntax 7-10
 - DTRAN.EXE, file description 1-7
 - DUT 5 V Power LED 9-8
 - duty cycle 2-10
 - control 10-13
 - Duty Cycle, variable function 3-15
 - specifying in PRGMVARS 3-15
- ## E
- E-, F-, G-pole jacks 9-8
 - EAOCAP
 - syntax 7-5
 - ECO revision, executing test page 4-10
 - Edit Device Pins window 9-3
 - Edit menu
 - component categories 3-7
 - overview 1-12, 3-6
 - sections 3-7
 - test steps 3-7
 - Edit PGEN.CFG command 7-3
 - Editor
 - default permissions 1-17
 - login permissions 1-16
 - EFG test terminals 5-26
 - 1800 Operating System Drive
 - Path Data setup field 2-2
 - 18xx.CGF 2-2
 - EMF, reducing effects of thermal EMF 3-37
 - ENABLEPRLL syntax 7-9
 - Enter (Return) key, keyboard selection 1-2
 - environment controls, setting up 2-1
 - Environment Data
 - refreshing in GFI 2-14
 - setup 2-2
 - Environment Data fields
 - Alternate Program Directory 2-2
 - Currently Selected Directory 2-2
 - environment variables
 - DFP reboot delay 2-11
 - display redraw 2-11
 - record Pgen configuration 2-11
 - runtime DR check 2-11
 - type of power down at cancel 2-11
 - chain depth 2-11
 - disable auto update 2-11
 - Nodefinder display 2-11
 - vector fail limit 2-11
 - ESC key, quitting menu 1-6

EXCEPT.LST, location of validated tests 2-18
EXCEPT.LST, file type and description 1-10
Exclude Allprint Device(s), description 3-18
Execute Step on Condition Flags, message step 3-19
exiting. *See* quitting
EXTENDED_MODE, Global Editor attribute-value 9-54
External Power Down variable 5-9
External Power Down, variable function 3-15
external power supplies, powering down 5-9
External Program 3-48

- accessing through Pre- or Post Test Options 10-16
- accessing through worksheet block 10-15
- functionality explained 10-15
- message step field 3-19

F

F1 key for Help vi
F10 key, to quit menu 1-5
F11 Command, Path Data setup field 2-2
failure reporting

- adding analog stim node data 3-18
- Allprint 3-17
- appending messages to Datalog device 3-17
- BSID 3-49
- controlling in PRGMVARS 3-17
- CRC signatures (Digital HiCheck) 3-14
- customizing standard tags 3-21
- measurement node number report 3-17
- preventing false failure messages 3-14
- Report Meas Node 3-17
- Report Prefix 3-17
- Report Values 3-17
- Section Abort 3-14
- Shorts Locator 3-17
- specifying failing values in PRGMVARS 3-17
- specifying Report Output Device 3-17
- vector test 3-51

failures

- identifying intermittent 4-12
- skipping test step 3-14
- stopping test step (SOF) 3-14

Fast Mode, eliminating internal wait, squelch 5-25
FAST_MODE

- Global Editor attribute-value 9-55
- used in Global Editor 9-52

Fault Coverage

- board level report 9-25
- board-level analysis 9-24
- MultiScan 9-32
- pin level formula, MultiScan 9-32
- report, generating 9-24

fault coverage report

- Device Fault Coverage 9-30
- Device Pins Not Covered 9-30
- Devices Tested with Multiple Techniques 9-30
- Report Header 9-30
- supplementing with imported data 9-33
- Test Steps Not Included 9-31

fault coverage test techniques 9-31

- Gray code 9-33
- MultiScan 9-32
- None 9-33
- vector 9-33

Fault Inject

- accessing 9-17
- algorithm 9-18
- effect on test time 9-25
- examining state fault coverage 9-17
- Fault Inject menu 9-20
- pin fault coverage 9-18
- report 9-21
- saving report 9-23
- setting up 9-19
- state fault coverage 9-19
- vector, example of results 9-21

FD.VEC, file description 1-8
FIB 3-46
FIB HB cluster, setting relays 3-47
FIB HC cluster, setting relays 3-47
FIB, using in slot 0 9-73
File Manipulation

- default permissions 1-17
- login permissions 1-16

files

- 18xx.CGF 2-2
- ALIAS.CFG 8-10
- ALIAS.LNK 8-2
- ASYNCR.DOC 2-21
- AUTOEXEC.BAT 1-6
- CONFIG.SYS 1-6
- DATALOG.DAT 2-7
- DEVICE.LST 8-2
- DSCAN.DB 1-26
- DSCAN.E4 1-26
- GFILE.DAT 5-36
- IPL.DAT 6-4
- IPL.DBF 6-4
- IPL.DFB 7-18
- NODE3V 9-73
- PANEL.DAT 10-33
- PGEN.CFG 1-6, 6-7, 7-3
- TEMP.IND 8-2
- TRACER.DEF 9-15

Files menu

- Exit 1-21
- Exit to DOS shell 1-25
- functions 1-12, 1-20
- ICT/FST 1-21, 1-22
- managing programs 1-20
- Multipanel 1-21
- New 1-21
- Remove 1-21
- Select 1-21
- Toggle TPD 1-21

fixture, building strategy 5-2

flags
 Modify Flags field 4-6
 specifying conditions 4-7
 specifying flags in message step 3-19
 See also System flags. Option flags. User flags.

flip-flop tests, developing 10-9

flow control

 branching 4-14
 Repeat Count 4-10
 Repeat Delay 4-10
 Repeat functions 4-10
 Repeat Mode 4-10
 Repeat Page 4-10
 Test Page Execution 4-10

Flow Control options 4-6

 Branching field 4-8
 Branching/Condition 4-8
 Branching/Destination 4-8
 Branching/Test Flags 4-8
 Comments field 4-6
 editing 4-6
 Modify Flags field 4-6
 Modify Relays 3-43

FrameScan Plus testing

 enabling or disabling Validate 2-19
 Measure Node variable function 3-15
 Measure Node, description 3-15
 overview 5-3
 specifying measurement node 7-9
 Statistics report 9-7

FrameScan testing

 enabling or disabling Validate 2-19
 minimum threshold 2-19
 overview 5-3
 specifying fast mode 7-9
 specifying inducer number in IPL syntax 6-17
 specifying reference node 7-8
 specifying threshold 7-9
 Statistics report 9-7

frequencies, measuring 10-1

FSCAN, input list token 6-17

FSPLUS, input list token 6-18

FSPMEAS syntax 7-9

FST.B0, file type and description 1-10

function keys 4-13, 4-14

functional interface board. *See* FIB

G

gate test, developing 10-8

GCIO.DLM, file description 1-7

GCTEMPL.LIB, file description 1-9

GCTHC.DLM, file description 1-7

General Factory Interface. *See* GFI

General Failure, System flag 4-6

General Variables 3-13

 Abort on Fail Count 3-15
 Backdrive Timeout 3-14
 Board Description 3-15

DeltaScan Fixed Threshold 3-15

Digital HiCheck 3-14

Digital Tracer 3-14

Discharge Method 3-15

DR1 Emulation Mode 3-15

Dscan Hard Fixed Thresh 3-16

Dscan Retry 3-16

Dscan Retry Delay 3-16

Dscan Run all Pin Pairs 3-16

Duty Cycle 3-15

External Power Down 3-15

FrameScan Plus Measure Node 3-15

Ground Reference Nodes 3-15

Learn Cap Phase Offsets 3-15

MultiScan Discharge 3-15

MultiScan Reference Node 3-15

Section Abort 3-14

Short Power/Ground via G-pole variable function
 3-15

Skip On Fail 3-14

Stop On Fail (SOF) 3-14

Vacuum Delay 3-14

Vacuum Select 3-14

GFI

 adding second page to test program 2-14

 changing execution order 2-14

 default permissions 1-17

 deleting applications 2-15

 Direct execution mode 2-14

 disabling application 2-14

 disabling hotkeys 2-14

 Execute Mode 2-13

 login permissions 1-16

 menu functions 1-14

 refreshing environment data 2-14

 refreshing output device files 2-15

 refreshing the screen 2-14

 selecting from Setup menu 2-12

 setting up interface 2-13

 use of Tokenlog file for test results 2-15

 window 2-12

GFILE.DAT

 digital guard file 5-36

GFILE.DAT, file type and description 1-10

Global Editor 9-63

 accessing 9-44

 Action Attributes pop-up, example 9-64

 actions format 9-44

 adding guards 9-44

 agent formats 9-44

 agent input file format synopsis 9-45

 agent type 9-44

 attribute-value pair specifics 9-51

 batch file, example 9-69

 changing test limits 9-44

 command line interface 9-70

 command line interface options 9-70

 Constructed Agent Display area 9-65

Global Editor *continued*

- converting from ATB to PRISM tests 9-64
 - converting from PRISM to ATB tests 9-64
 - creating new Agent Input file 9-61
 - enabling test step 9-63
 - Execute window example 9-67
 - finding specific test steps 9-69
 - format, examples 9-60
 - managing contents of output file 9-69
 - modifying constructed agents 9-66
 - modifying in-circuit test files 9-44
 - navigating window 9-61
 - Options menu 9-65
 - overview of structure 9-44
 - Qualifier Attributes 9-63
 - qualifiers format 9-44
 - running an existing agent input file
 - command line 9-69
 - execute 9-67
 - searching for component ID attributes 9-65
 - searching multiple board directories 9-69
 - ways of using 9-61
 - window, example 9-62
- global Validate, performing 7-17
- GND, input list token 6-6
- GNPN, input list token 6-15
- GPNP, input list token 6-15
- G-pole, used with DeltaScan 3-15
- Gray code
 - clock divisor 7-7
 - Combine groups 3-49
 - measure pins 9-18
 - Separate groups 3-49
 - Step Worksheet, Options 3-39
 - stim pins 9-18
 - using Fault Inject with 9-18
- Gray code groups
 - editing 9-2
 - in Gray code templates 8-18
 - test step control 3-49
- Gray code Test Properties
 - Clock Divisor 3-39
 - High Threshold 3-39
 - Indicators 3-39
 - Logic Level 3-39
 - Low Threshold 3-39
 - Measure 3-39
 - Options 3-39
 - Pin 3-39
 - Stimulus 3-39
 - Terminator 3-39
- Gray code testing
 - backdrive timeout 10-15
 - Component Properties 3-38
 - configuring ATB 5-26
 - creating templates 8-19, 9-5
 - disables 5-39
 - guards 5-37

- listen window default actions 9-9
 - Step Worksheet 3-37
- Ground Reference Nodes
 - specifying in PRGMVARS 3-15
- Ground Reference Nodes, variable function 3-15
- GUARD_MODE, Global Editor attribute-value 9-54
- GUARD_NODES, Global Editor attribute-value 9-57
- guarding
 - active guard mode 3-37
 - capacitors 9-76
 - default mode (3-wire) 5-21
 - low-impedance DUT 5-21
 - passive guard mode 3-37
 - resistors 9-76
 - semi-active guard mode 3-37
 - stimulus and measurement placement 3-34
- Validate 9-73
- Validate with resistors 9-74
- vector guards in DFP tests 10-21
- vector tests 5-39
- See also* analog guarding *or* digital guarding

H

- Hardware Configuration 9-73
 - Actual/Expected configuration window 9-72
 - creating "expected configuration" file 9-71
 - driver/receiver 9-73
 - scanning for DR2 cards 9-73
 - scanning with FIB in slot 0 9-73
 - setting up card cage scan 9-71
 - Utility menu 9-71
- Header
 - adding custom message 3-21
 - controlling test flow 4-4
 - editing message step 3-18, 4-5
 - section template insertion 6-6
- Help, context-sensitive vi
- High Threshold, Gray code 3-39
- High Voltage Option variable, PGEN.CFG 7-5
- HIGH_VALUE, Global Editor attribute-value 9-54
- Higuard
 - analog measurement 5-24
 - reducing effects of thermal EMF 5-24
 - using for AC measurements 5-24
- HIGUARD, Global Editor attribute-value 9-55
- Hold (F6) 4-13
- hotkeys. *See* GFI

I

- I/O Control Options, operator communication 3-43
- IC, input list token 6-16
- ICT and FST, toggling between 1-22
- ICT.B0, file type and description 1-10
- ICT.BCK, file type and description 1-10
- ICT.D1, file type and description 1-10
- ICT.DO, file type and description 1-10
- ICT.DSC, file type and description 1-10
- ICT.E4, file type and description 1-10

- ICT.NDX, file type and description 1-10
- ICT.TST
 - after Clean 7-12
 - file type and description 1-9
- ICTD2.X2, file type and description 1-10
- ID, Global Editor attribute-value 9-55
- IEEE test
 - Controls section 10-20
 - Device & Channel Data field 2-3
 - E, F, and G Pole editing 10-20
 - Format Reverse 10-19
 - Format Skip field 10-19
 - Format Use field 10-19
 - IEEE String field 10-20
 - performing tests with IEEE instruments 10-18
 - Scale field 10-19
 - Test Properties 10-18
 - Test Type 10-18
 - Wait fields 10-20
- ignore threshold, PGEN.CFG syntax 7-10
- IGNORETHRSYNTAX 7-10
- impedance, calculating DUT impedance 5-18
- improving capacitor test measurements 7-17
- improving resistor test measurements 7-17
- in-circuit program. *See* test program
- INDEX.HLP, file description 1-8
- INIT_COM.EXE, file description 1-7
- input control options, editing 3-42
- input data, specifying variable to receive data 3-19
- input list
 - C.1 to current revision 6-25
 - C.1, adding Cname to current revision 6-26
 - comments 6-5
 - effect on Device Type 3-28
 - generating one component at a time 5-2
 - loading Producer 2 input list 10-52
 - structure 6-4
 - summary of record syntax 6-4
 - token 6-5
 - using Nodefinder to build partial 9-36
- input list component token
 - C (Capacitor) 6-10
 - CONT (Continuity) 6-9
 - J (Jumper) 6-9
 - L (Inductor) 6-10
 - POT (Potentiometer) 6-11
 - R (Resistor) 6-10
 - REO (Rheostat) 6-11
 - SC (Special Cases) 6-10
- input list control token
 - APC (global APC) 6-9
 - DCHG (Discharge) 6-8
 - GND 6-6
 - Header 6-6
 - IPL 6-6
 - OPN (Open) 6-9
 - PB (power bus) 6-8
 - PBUS 6-8
 - PWR5 6-7
 - PWR5.5 6-7
 - PWRA 6-7
 - PWRA_F 6-8
 - PWRB 6-7
 - PWRB_F 6-8
 - SH (Shorts) 6-9
 - Trailer 6-6
- input list language
 - DESC 6-20
 - ID 6-19
 - input list sample 6-22
 - input list token summary, table 6-20
 - NAME 6-20
 - strings 6-20
 - syntax example 6-20
 - TOL1 6-20
 - TOL2 6-20
 - unit abbreviations 6-20
 - unit modifiers 6-20
 - VAL1 6-20
 - VAL2 6-20
- input list special token
 - END 6-19
 - INPUTLIST 6-19
- input list token 6-9
- Input Type, message step field 3-19
- Intc. *See* interconnect testing
- Interconnect
 - relationship to Test Types 3-8
- Interconnect Learn 7-12
- interconnect testing
 - inserting in ICT 7-12
 - Learn window 7-13
 - System Failure flag 4-6
- Invert Pass/Fail, test for component absence 3-48
- IPEXCEP.LST, file type and description 1-9
- IPL.DAT
 - after Clean 7-12
 - input list file 6-4
 - record 6-4
 - use in PEP 7-11
- IPL.DAT, file type and description 1-9
- IPL.DBF
 - Build 7-12
 - input list file 6-4
- IPL.ID.NDX, file type and description 1-10
- IPL.LST
 - example of file 6-22
 - file type and description 1-10
- IPL_NDX, file type and description 1-11
- IPL_NOD.NDX, file type and description 1-10
- IPL_SPEC.TXT, file description 1-8
- IPL_TOK.NDX, file type and description 1-10
- IPLHDR.TXT, file description 1-8

J

- J, input list token 6-9
- Jedec vectors
 - adding statistics and commentary 8-13
 - creating library templates from Jedec files 8-20
 - translating to 18xx format 8-13
- JMPTHRESH, syntax 7-7
- JTOOL.EXE, file description 1-7
- jumper threshold
 - PGEN.CFG syntax 7-7

K

- keyboard
 - operation in Step Worksheets 3-1
 - viewing on-screen template 4-14

L

- L, input list token 6-10
- Learn Cap Phase Offsets
 - enabled with Bare Boards 3-15
 - enabled with Wired Fixture 3-15
- Learn Cap Phase Offsets, variable function 3-15
- learning
 - continuities 7-12
 - shorts range 7-12
 - special cases 7-12
- learning interconnects 7-13
- LEDs
 - analog debug 9-8
 - digital pullups 9-8
 - DUT 5 V Power 9-8
- LIBORDER syntax 7-8
- libraries
 - access to 8-1
 - adding ASCII vector 8-5
 - adding IVL clusters 8-5
 - adding IVL vector 8-5
 - adding new alias 8-4
 - adding new templates 8-4
 - adding TDS vectors 8-16
 - analog library 8-17
 - creating in DOS 8-3
 - creating new libraries 8-2
 - customizing commentary 8-13
 - digital and analog template types summary 8-5
 - edit menu 8-3
 - extracting modified templates 8-6
 - extracting templates 8-6
 - Gray code 8-17
 - Local Library 8-1
 - Local library 8-2
 - local user path 8-1
 - managing database size 8-7
 - modifying template database 8-10
 - overview 8-1
 - packing 8-7
 - packing database 8-7
 - restoring database 8-8

- setting up paths with PGEN.CFG 7-8
- setting up permissions 1-16, 8-2
- System Library 8-1
- system library path, default 8-1
- User Library 8-1
- User library 8-2
- vector image 8-17
- vector snapshot 8-17
- vector template 8-17
- library control, PGEN.CFG 7-8
- line printer
 - Device & Channel Data field 2-3
- line transceiver tests, developing 10-11
- LINE_REJECT, used in Global Editor 9-52
- linear component, relationship to test structure 3-8
- link tokens 6-19
- listen window 9-9
 - controlling opening with From 9-9
 - controlling with To field 9-9
 - controlling with While 9-9
 - Gray code test 9-9
 - in analog test 9-9
 - programming for higher frequencies 10-1
 - programming for lower frequencies 10-1
 - programming for middle frequencies 10-1
 - vector test 9-10
- Locator key (F2) 4-13
- logging in
 - as supervisor 1-15
 - Auto-Convert permission 1-16
 - Calibration permission 1-16
 - changing passwords 1-18
 - changing permissions 1-16, 1-17
 - default permissions 1-17
 - Diagnostics permission 1-16
 - Editor permissions 1-16
 - GFI Execute permission 1-16
 - Logins menu 1-14
 - Node Probe Tools permission 1-16
 - password card 1-18
 - passwords 1-15
 - Program Execute permission 1-16
 - Program Generator permission 1-16
 - Program Select permission 1-16
 - Revision Control permission 1-16
 - saving settings permanently 1-20
 - setting default login 1-19
 - setting logins 1-14
 - Setup permission 1-16
 - Startup User Id window 1-19
 - Stop On Fail permission 1-16
 - Template Lib Write permission 1-16
 - Template Library permission 1-16
 - Validate permission 1-16
- login permissions
 - File Manipulation 1-16
 - Template Lib Write Permissions 1-17
- longhand test mode, when to use 3-33

low threshold, Gray code 3-39
 LOW_VALUE, Global Editor attribute-value 9-54

M

Main menu
 access to 18xx programming functionality 1-11
 accessing 1-14
 Select command 1-21
 selection techniques 1-2
 major ID
 topology syntax 7-19
 See also topology
 Major Separators 7-9
 MAJRSEP syntax 7-9
 Master Datalog On, System flag 4-7
 master program/subprogram
 Cancel operation 4-4
 rules for shared variables 4-4
 MEAS_NODES, Global Editor attribute-value 9-57
 MEAS_PIN, Global Editor attribute-value 9-55
 measure pins
 analyzing CRC signature, Gray code 9-18
 Fault Inject 9-18
 measurement operational amplifier. *See* MOA
 measurement stability, swapping poles 9-5
 memory device tests, developing 10-12
 menus
 keypad cursor keys, navigating/selecting 1-2
 menu bar 1-1
 menu bar location on Step Worksheet 3-5
 navigation with arrow cursor keys 1-2
 selecting from 1-1
 merging. *See* program merging
 message step
 adding to Header or Trailer 3-21
 Chain Board Name 3-19
 customizing failure tags 3-21
 editing 3-18
 Header 4-5
 Trailer 4-5
 enabling board description 3-15
 enabling name string output 3-18
 Execute Step on Condition Flags 3-19
 External Programming 3-19
 Input Type 3-19
 Modify Flags 3-19
 Output Device 3-19
 Output String 3-19
 setting up program chaining 4-16
 Step Name 3-19
 Vacuum Control 3-19
 Variable 3-19
 window example 3-18
 minor ID
 topology syntax 7-19
 See also topology
 mixed mode testing, controlling timing 3-36
 MM.COM, file description 1-8

MOA
 ideal conditions 5-18
 inverting configuration 5-17
 with guarding for accurate measurement 5-18
 Modify Flags, message step field 3-19
 Momentum
 FABmaster 5-3
 option 5-3
 MOS files listed 1-7
 mouse operation in Step Worksheet 3-1
 MSCANFAST syntax 7-9
 MSCANREF syntax 7-8
 Multipanel
 copying panel program 10-31
 creation process 10-31
 development sequence 10-32
 entering board descriptions 10-29
 entering node offsets 10-29
 entering panel program data 10-28
 excluding node offsets 10-29
 Program Copy window 10-31
 using %BOARD output macro 10-29
 Multipanel functions 1-24
 Multipanel testing
 copying programs 1-25
 creating master program 10-27
 Multipanel Copy window 10-28
 process overview 10-27
 topology syntax 7-19
 multiple board testing. *See* Multipanel testing
 MultiScan Discharge, variable function 3-15
 MultiScan Failure, System flag 4-7
 MultiScan Fast
 PGEN.CFG 7-9
 MultiScan Reference Node, variable function 3-15
 MultiScan testing
 controlling discharge in Header 3-15
 fault coverage, disabling/enabling tests 9-32
 improving with Validate 9-73
 MultiScan Reference Node 3-15
 overview 5-3
 specifying reference node 3-15, 7-8
 Validate function 2-16

N

NAME, Global Editor attribute-value 9-55
 New Board Directory window 1-23
 Node Finder jack 9-8
 node fixtures, examining fit 10-52
 node numbers
 finding associated D/R board 9-39
 finding component leads 9-38
 null node (9999) 9-36
 use in creating templates 8-19
 using 9999 in template creation 8-19
 using Nodefinder from Main menu 9-35
 Node Probe Tools
 default permissions 1-17
 login permissions 1-16

node specification in Test Properties 3-34
NODE.LST, file type and description 1-10
NODE3V

- file type and description 1-11
- voltage requirements 9-73

Nodefinder

- accessing 9-36
- activating 9-36
- finding nodes with probe 9-36
- identifying analog nodes 9-36
- identifying digital nodes 9-37
- identifying node numbers 9-35
- impedance 2-10
- in Digital Tracer 9-15
- installing probe 9-36
- setting probe impedance 9-36
- using in analog test 9-36

NODHDR.TXT, file description 1-8

NOIO.DLM, file description 1-7

noise reduction, stimulus placement 3-34

NOTHC.DLM, file description 1-7

NUM_MOD.EXE, file description 1-7

Number of Pins, definition 3-30

O

online documentation key (F12) 4-13

on-screen template function key 4-13

op amp testing 10-6

- checking linearity 10-6
- computing expected output 10-6
- configuring as voltage follower 10-6
- requirements 10-6

operational amplifier. See op amp

OPN, input list token 6-9

Option flags

- controlling test execution in Run mode 4-14
- modifying 4-6
- operation 4-8
- window 4-7

output channels

- Output Devices window 2-5
- preventing Tokenlog routing 2-7
- program level setup 2-6
- setting up at system level 2-4
- test step level setup 2-6

output control options, editing 3-42

output data

- Datalog setup example 2-7
- enabling devices in program mode 2-5
- enabling systemwide 2-4
- hierarchy of control 2-4
- PEP 5-2
- routing conditional I/O 2-7
- Section Execute mode 2-5
- See also failure reporting
- Step Execute mode 2-5
- suppressing report generation 2-6
- Tokenlog setup example 2-8

- Tokenlog, machine-readable format 2-8
- valid settings for output devices 2-5

Output Device

- Environment Data window 2-5
- message step field 3-19
- refreshing files 2-15
- selecting in Environment Data window 2-5
- specifying in I/O Control Options 3-43

output macros, for custom messages 3-20

Output String, message step field 3-19

overlapping parts, topology 7-19

P

Pack command, memory requirements 8-7

page execute, mode of operation 2-4

PAGE, Global Editor attribute-value 9-55

PALAMOD.EXE, file description 1-7

PALAMOD.SET, file description 1-8

PANEL.CFG, file type and description 1-11

PANEL.DAT

- editing for Multipanel testing 10-33
- format 10-33

PANEL.DAT, file type and description 1-11

parallel component test step, inclusion in topology 7-16

parallel components, board fault coverage 9-24

passive components, relationship to test structure 3-8

passwords

- changing 1-15
- changing procedure 1-18
- location of user default 1-15
- syntax 1-19

path data setup 2-2

PB, input list token 6-8

PBUS, input list token 6-8

PCIO.EXE, file description 1-7

PCIO.TXT, file description 1-8

PEP

- accessing from 18xx interface 7-11
- analyzing component testability 5-2
- functions 6-2
- output data 5-2
- setup in 18xx environment 7-11

permissions, how to change 1-17

Pgen

- Build function 6-2
- Clean function 6-2, 7-14
- Combine Parallel function 6-2
- Dig Guards function 6-2
- Edit PGEN.CFG function 6-2
- Generate function 6-2
- Learn function 6-2
- overview of menu system 1-12, 6-3
- PEP function 6-2
- PEP functions 6-2
- Reports function 6-2
- Update function 6-2
- Validate function 6-2

- Pgen Config variables
 - saving to file 7-5
 - showing defaults 7-5
 - viewing all 7-4
 - viewing modified 7-4
 - viewing new 7-4
 - viewing unmodified 7-4
 - writing to a file 7-4
- PGEN.CFG
 - choosing variable view mode 7-4
 - diode variables 7-7
 - DISCHGENVAL variable 7-10
 - editing 7-3
 - editing text file 7-10
 - file syntax 7-11
 - file type and description 1-11
 - hierarchy of influence 7-3
 - High Voltage Option variable 7-5
 - making global changes 7-3
 - modifying current program 7-3
 - modifying for programmable p. s. 5-13
 - modifying test program 7-3
 - place in directory structure 1-9
 - PRISMGEN variable 7-10
 - recording changes 7-5
 - routing changes to EXCEPT.LST 7-5
 - SHTHRESH variable 7-7
 - viewing all Pgen Config variables 7-4
 - viewing modified Pgen Config variables 7-4
 - viewing new Pgen Config variables 7-4
 - viewing unmodified variables 7-4
 - worksheet 7-3
 - zener variable 7-7
- PGEN.CFG syntax
 - adding identifiers to components 7-9
 - CapScan Ratio 7-9
 - Combine Parallel 7-9
 - continuity threshold 7-7
 - defining wait time in short steps 7-10
 - DeltaScan parameters 7-10
 - digital threshold 7-7
 - discharge parameters 7-8
 - driver/receiver configuration 7-6
 - EAO capacitor 7-5
 - EAO RC 7-5
 - Gray code clock divisor 7-7
 - Gray code Stim Value 7-6
 - ignore threshold 7-10
 - jumper threshold 7-7
 - Major Separators 7-9
 - MultiScan Fast 7-9
 - multiwire threshold 7-6
 - power supply controller 7-8
 - shorts threshold 7-7
 - Shorts Wait 7-10
 - specifying measurement node for FrameScan
 - Plus test 7-9
- PGT files listed 1-9
- pin fault coverage 9-18
 - measure pins 9-18
 - stimulus pins 9-18
 - stimulus/measure pins 9-18
- Pin Number, definition 3-30
- pin types
 - editing 9-2
 - stimulus, measure, stimulus/measure 9-18
- pins, filling number in Test Properties 9-2
- PINS, Global Editor attribute-value 9-55
- pop-up window
 - changing size 1-4
 - description 1-4
 - invoking with keyboard 3-2
 - invoking with mouse 3-1
 - moving 1-4
- Post-Test Options
 - External Program 3-48
 - FIB 3-46
 - RAB 3-43
 - submenus 3-41
 - Variables
 - effect on execution 4-2
- Post-Test Options, setting and clearing relays 3-43
- POT, input list token 6-11
- potentiometer adjustment, Repeat Page 4-10
- Power
 - controlling board power 5-6
 - preventing application to defective boards 3-14
 - Test Properties 5-10
 - mode selections 5-11
 - programmable 5-10
 - See also* board power
- power down at cancel 2-11
- Power Failure, System flag 4-7
- Power Nodes, Discharge Methods 3-15
- Power Off, relationship to test structure 3-8
- power supply
 - ±12 volt 6-8
 - ±15 volt 6-8
 - control boards 5-6
 - controlling in Test Properties 5-9
 - fixed worksheet generation 6-7
 - listed in Test Properties 5-12
 - management 5-9
 - modifying with PROGSUPWS 7-8
 - On/Off sequence 5-11
 - programmable worksheet generation 6-7
 - relationship to system software revisions 5-9
 - relationship to Test Properties 5-11
 - system architecture 5-6
 - test program restrictions 5-6
 - updating D.X programs 5-13
- power supply control tokens, caution against mixing
 - fixed and programmable 6-7
- power supply controller
 - PGEN.CFG syntax 7-8

- Power Test Properties
 - adjustable 5-10
 - Disable function 5-12
 - effect of Pre-/Post Test Options 5-12
 - Enable function 5-12
 - Measure Internally 5-12
 - Measure Low/High 5-12
 - Off field 5-11
 - On field 5-11
 - Reprogram 5-12
 - restriction on mixing Test Properties 5-10
- Power Test Types, restrictions 5-6
- Power worksheets
 - controlling generation of fixed 6-7
 - controlling generation of programmable 6-7
- power-down
 - discharge sequence 5-7
 - express mode 5-7
 - orderly mode 5-7
 - restore process 5-8
 - suppressing discharge phase 5-8
- power-up sequence 5-7
- PRE_CYCLE_VACUUM attribute-value 9-56
- PRE_REPEAT_COUNT attribute-value 9-56
- PRE_REPEAT_MODE attribute-value 9-56
- PRE_REPEAT_PAGE attribute-value 9-56
- PRE_REPORT_DELAY attribute-value 9-56
- Precise mode, improving analog measure 5-23
- PRECISE, Global Editor attribute-value 9-55
- Pre-Test Options
 - External Program 3-48
 - FIB 3-46
 - Option Variables
 - DUT Power 3-47
 - Power Supply 3-47
 - RAB 3-43
 - setting and clearing relays 3-43
 - submenus 3-41
 - Variables 3-47
 - effect on execution 4-2
- PRGMVARS
 - Allprint 3-17
 - Allprint Message 3-17
 - APC (Auto Probe Check) 3-14
 - Datalog 3-17
 - DeltaScan, Verify Pwr/Grd Shorting 3-15
 - Exclude Allprint Device(s) 3-18
 - Report Analog Stim Nodes 3-18
 - Report Component Name 3-18
 - Report Description 3-17
 - Report Dscan Pin Pair Data 3-18
 - Report ID 3-17
 - Report Limits 3-18
 - Report Meas Nodes 3-17
 - Report Output Device 3-17
 - Report Prefix 3-17
 - Report Values 3-17
 - Revision Control 3-17
 - setting global variables 3-13
 - setting up 3-13
 - Shorts Locator 3-17
 - Status Display Line 3-17
 - test flow control 4-5
 - Tokenize Reports 3-18
 - windows 3-13
- Primary Program Directory
 - changing program directories 2-3
 - Path Data setup field 2-2
 - See also* directories
- PRINT.DAT, file description 1-9
- Printer Width, Device & Channel Data field 2-4
- Prism
 - backplane capacitance 2-10
 - line frequency 2-10
 - memory test level 2-10
 - system capacitance 2-10
 - system inductance 2-10
 - system resistance 2-10
 - trim interval 2-10
- PRISM.IMG, file description 1-7
- PRISMCAL.EXE, file description 1-7
- PRISMGEN variable
 - PGEN.CFG 7-10
- PRISM-Z
 - compatibility with ATB tests 5-4
 - defining components to test 7-10
- PRISM-Z, analog measurement subsystem 5-4
- PRLLRLC syntax 7-9
- Probe key (F9) 4-13
- program chaining
 - Chain Board Name 4-3
 - chain depth 4-3, 4-4
 - effect of Cancel on program execution 4-4
 - Multipanel testing 10-28
 - preventing execution of main program 4-17
 - setting up 4-16
 - setting up message steps 4-16
 - specifying program in message step 3-19
 - Start button operation 4-3
 - start setup 4-3
 - subprogram control 4-4
 - vacuum operation 4-4
- Program Execute channels, Device & Channel Data 2-3
- program execution
 - controlling in Run mode 4-14
 - default permissions 1-17
 - login permissions 1-16
 - mode of operation 2-4
 - sequence 4-1, 5-10
 - specifying power supply mode 5-6
- program flow tools
 - Abort on Fail Count 4-5
 - Chain Depth 4-4
 - Header and Trailer 4-4
 - Section Abort 4-5

- program generation
 - Component List 7-18
 - incremental process 7-14
 - Node List 7-18
 - output files 7-18
 - Reports menu 7-18
 - Test IPL List 7-18
 - Topology Report 7-19
 - Z850 10-52
 - Z875 10-52
 - program generator
 - accessing 6-1
 - default permissions 1-17
 - login permissions 1-16
 - See also* Pgen
 - program merging 3-21
 - overwriting files 3-22
 - using Append 3-22
 - writing out files 3-22
 - program name syntax 1-24
 - program revision, tracking 3-11
 - Program Select
 - default permissions 1-17
 - login permissions 1-16
 - program variables
 - General Variables 3-13
 - Report Variables 3-13
 - program voltage control 5-6
 - PROGRAM.DOC, file type and description 1-11
 - PROGRAM.HLP, file description 1-8
 - Programmer Efficiency Package. *See* PEP
 - PROGSUPWS, syntax 7-8
 - PWR5, input list token 6-7
 - PWR5.5, input list token 6-7
 - PWRA, input list token 6-7
 - PWRA_F, input list token 6-8
 - PWRB, input list token 6-7
 - PWRB_F, input list token 6-8
 - PwrNodes. *See* Power Nodes
 - PwrOff. *See* Power Off
- Q**
- QH.EXE, file description 1-7
 - QN, input list token 6-15
 - QP, input list token 6-15
 - Qualifier Attributes, Global Editor 9-63
 - quality control, auditing the program 5-6
 - Quick Help key (F1) 4-13
 - quitting
 - from one menu level to another 1-5
 - operations 1-5
 - Step Worksheet 1-5
 - test program with ESC key 1-6
 - to DOS 1-5
- R**
- R, input list token 6-10
 - RAB
 - clearing relays 3-46
 - setting and clearing relays 3-43
 - User Relays window 3-44
 - RAM testing 10-12
 - record syntax for IPL 6-4
 - relay array board. *See* RAB
 - relays
 - analog master 5-17
 - bridging 5-17
 - clearing 3-46
 - clearing with Cancel 4-12
 - controlling in Flow Control options 3-43
 - delay for settling 3-46
 - digital master (comparator) 5-17
 - driver D 5-17
 - during digital test 5-14
 - passive test formats 10-4
 - testing 10-3
 - typical circuit 10-3
 - RELNOTES.TXT, file description 1-8
 - Remove command, Files menu 1-23
 - REO, input list token 6-11
 - Repeat (F7) 4-13
 - Report All, generating reports 7-19
 - Report Analog Stim Nodes, description 3-18
 - Report Component Name, description 3-18
 - Report Description, PRGMVARS 3-17
 - Report Dscan Pin Pair Data, description 3-18
 - Report Output Device, indicator in PRGMVARS 3-17
 - report output, adding pass/fail limit of test 3-18
 - report tags, customizing for entire program 3-21
 - Report Variables 3-13
 - Allprint 3-17
 - Allprint Message 3-17
 - Exclude allprint Device(s) 3-18
 - Report Analog Stim Nodes 3-18
 - Report Component Name 3-18
 - Report Description 3-17
 - Report Dscan Pin Pair Data 3-18
 - Report ID 3-17
 - Report Limits, description 3-18
 - Report Meas Nodes 3-17
 - Report Nominal 3-18
 - Report Output Device 3-17
 - Report Prefix 3-17
 - Report Values 3-17
 - Revision Control 3-17
 - Shorts Locator 3-17
 - Status Display Line 3-17
 - Tokenize Reports 3-18
 - reports, generating with Report All 7-19
 - resistor testing
 - converting to longhand 9-5
 - debug using decade box 9-11
 - enabling sampling in Validate 2-18
 - improving stability with Validate 2-16
 - Return to menu control key F10 4-13
 - Reverse Nodefinder, for component leads 9-38

- Revert to last saved edits
 - Edit menu 1-5, 3-25
 - Setup menu 1-4, 2-1
- Revision Control 3-11
 - default permissions 1-17
 - edit log file 3-12
 - enabling 3-11
 - login permissions 1-16
 - permissions 3-11
- Revision Control, description 3-17
- ROM testing 10-12
- RP_DB, input list token 6-13
- RP_DI, input list token 6-12
- RP_DT, input list token 6-14
- RP_SB, input list token 6-12
- RP_SI, input list token 6-11
- RP_ST, input list token 6-13
- Run command 1-12
- Runtime Analog Check, disabling for cage scan 9-73

S

- safety
 - check fixture wiring vii
 - external power supplies vii
 - powering down vii
 - transformer testing vii
 - troubleshooting the fixture vii
- safety precautions vii
- saving
 - packing user file 1-5
 - program edits 1-5
 - save functions in 18xx software 1-4
 - setup changes 2-1
 - setup to memory and disk 1-4
 - temporary edit file 1-5
- SC, input list token 6-10
- SCR testing 10-4
 - off current 10-5
 - on voltage 10-5
 - typical circuit 10-4
- screen colors
 - changing with Color window 2-22
 - changing with Palamod 2-22
 - Color window example 2-23
- scroll arrows, location on Step Worksheet 3-5
- Section Abort
 - setup in PRGMVARS 3-14
 - test flow control 4-5
 - variable function 3-14
- Section Abort, variable function 3-14
- Section Run, Skip on Fail 3-14
- SECTION, Global Editor attribute-value 9-56
- selecting from menus, techniques 1-2
- self-test program, selecting 1-22
- semi component, relationship to test structure 3-8
- serial communications
 - setting up 2-20
 - see also* communications ports
- setting relays
 - FIB HB cluster 3-47
 - FIB HC cluster 3-47
- Setup
 - cancelling operations 2-2
 - Chain Depth 4-4
 - default permissions 1-17
 - login permissions 1-16
 - menu overview 1-13, 2-1
 - OK button operation 2-2
 - Save operation 2-2
 - setting up serial communications 2-20
- Setup/Data window. *See* Environment Data
- SH, input list token 6-9
- shared variables, rules 4-4
- shorthand test mode
 - improving with Validate 9-73
 - when to use 3-35
- Shorts Locator, description 3-17
- shorts range, Interconnect Learn 7-12
- shorts threshold, PGEN.CFG syntax 7-7
- Shorts Wait
 - PGEN.CFG syntax 7-10
- SHTHRESH variable, PGEN.CFG 7-7
- SHTHRESH, syntax 7-7
- SHWAIT syntax 7-10
- Skip On Fail, description 3-14
- snapshot
 - creating template with Step Worksheet 9-5
 - technique 10-36
 - See also* vector image templates
- spacebar, toggling fields in 18xx interface 3-3
- special cases, Interconnect Learn 7-12
- squelch times
 - defined 5-25
 - discharging capacitors 3-35
 - relation to Higuard use 5-25
 - relation to test time 3-35
 - squelch, wait, stimulus states 3-36
 - using Fast Mode to eliminate 5-25
- SQUELCH, Global Editor attribute-value 9-52
- Start key (F3) 4-13
- start-up screen, removing copyright message 1-11
- Statistics report, WaveScan, FrameScan, FrameScan Plus 9-7
- Step Worksheet
 - accessing 3-23
 - controlling tester power 5-10
 - editing 3-23
 - Gray code 3-37
 - introduction 3-5
 - menu bar 3-25
 - menu functions 3-25
 - navigation techniques 3-1
 - Options menu, Step Validate 9-73
 - Test Properties
 - adding pages 9-4
 - deleting pages 9-4
 - Validate 9-73

- Vector 3-40
 - viewing 3-4
- STIM_NODES, Global Editor attribute-value 9-57
- STIM_PIN, Global Editor attribute-value 9-58
- stimulus nodes, specifying in Test Properties 3-34
- stimulus pins, Fault Inject 9-18
- stimulus sources, ATB 5-25
- stimulus/measure pins, Fault Inject 9-18
- Stop On Fail
 - Default Permissions 1-17
 - login permissions 1-16
- Stop On Fail (F5) 4-13
- Stop On Fail, description 3-14
- subprogram control 4-4
- SubProgram, System flag 4-7
- SUPP.CVG, file type and description 1-11
- Supplemental Diagnostics, running 9-40
- supplemental fault coverage 9-33
 - file format 9-33
 - reporting 9-34
- switch matrix
 - analog/digital 5-15
 - digital testing 5-15
 - three-wire configuration 5-28
- switch testing 10-2
- SYSLIB syntax 7-8
- system capacitance, default value 2-9
- System flags
 - editing 4-6
 - functions 4-6
 - modifying 4-6
 - window 4-8
- system flags, changing 4-6
- system software
 - basic architecture 3-3
 - customizing interface color 2-22
 - directory structure 1-6
 - file structure 1-6
 - hierarchy of control 4-1
 - starting up 1-11
- system variables
 - ATB system capacitance 2-9
 - ATB system resistance 2-9
 - backdrive timeout 2-10
 - Clear test page set up at 2-10
 - duty cycle 2-10
 - highest system node 2-9
 - Nodefinder impedance 2-10
 - Prism backplane capacitance 2-10
 - Prism Line Frequency 2-10
 - Prism memory test level 2-10
 - Prism system capacitance 2-10
 - Prism System inductance 2-10
 - Prism system resistance 2-10
 - Prism Trim interval 2-10
 - setting up 2-8
 - test jack panel 2-10
 - VP timeout 2-10

T

- TDS Translate menu 8-16
 - Convert 8-16
 - Edit 8-16
 - Quit 8-16
 - Save 8-16
- TDS Translate Table 8-14
- TDS vectors
 - adding to library 8-16
 - ASCII vector translation 10-38
 - ASIC pattern development 10-37
 - creating templates from TSSI input files 8-20
 - tools for converting simulation data 10-49
 - translating 8-14
 - translating to 18xx format 8-13
- TDS18XX.EXE, file description 1-7
- TELLME.EXE, file description 1-7
- TEMPL.IND, file description 1-9
- template component token
 - ATMPL (analog template), input list token 6-15
 - CSCAN, input list token 6-18
 - DB (Data bus) 6-16
 - DFP 6-19
 - DSCAN 6-18
 - FSCAN 6-17
 - FSPLUS 6-18
 - IC (Gray code template) 6-16
 - VCLUST (vector cluster), input list token 6-17
 - VEC (vector template) 6-16
 - VIMG (vector image) 6-16
 - WSCAN 6-17
- template files subdirectory. *See* TTM subdirectory
- Template Lib Write, login permissions 1-16
- Template Library
 - default permissions 1-17
 - login permissions 1-16
- templates
 - accessing revisions 8-9
 - adding templates to library 8-3
 - analog template, instant generation 8-18
 - comments in extracted template 8-7
 - Create Template process 8-18
 - creating 8-17
 - creating analog templates 8-17
 - creating analog, overview 8-17
 - creating Gray code, overview 8-17
 - creating Gray code, procedure 8-19
 - creating vector snapshot, overview 8-17
 - creating vector, overview 8-17
 - deleting/undeleting 8-3
 - extracting templates 8-3
 - listing aliases 8-12
 - listing aliases and templates 8-3
 - listing All Aliases 8-12
 - listing All Templates 8-12
 - listing deleted devices 8-13
 - listing One Template 8-13
 - listing templates 8-12

templates *continued*

- marking for deletion 8-7
 - modifying with DOS commands 8-3
 - name syntax 8-6
 - packing database 8-3
 - revision time stamps, Gray code 8-18
 - translating Jedec vectors into ASCII 8-3
 - transporting 8-6
 - TTM subdirectory 1-6
 - undeleting 8-8
 - using analog templates 8-18
 - Utility/accessing template revisions 8-3
 - Utility/reading files into template database 8-3
 - Utility/writing database to text file 8-3
 - vector image creation 8-20
 - vector image templates 8-20
 - wiring constraints 8-19
- Templates menu 1-13, 8-1
- Create Library 8-1
 - Library Permission 8-1
- Templates/Add
- Alias 8-4
 - ASCII Vector 8-4
 - IVL Cluster 8-4
 - IVL Vector 8-4
 - Template command 8-4
- Templates/Extract
- copying user-modified templates 8-6
 - for editing vector templates 8-6
- Templates/Extract, copying for transport 8-6
- test application
- definition 5-1
 - preparing Z850 for Z1800-series 10-52
 - preparing Z875 for Z1800-series 10-52
- Test Control buttons 4-10
- Cancel 4-11, 4-12
 - CRT 4-11, 4-12
 - HLD 4-12
 - HLD (Hold) 4-11
 - location in test step window 4-11
 - location on Step Worksheet 3-5
 - RPT 4-12
 - RPT (Repeat) 4-11
 - SOF (Stop On Fail) 4-11, 4-12
 - Start 4-10, 4-11
- test development tools 9-1
- test envelope 9-9
- test evaluation 5-5
- test execution
- changing test step sequence 3-11
 - effects of Pre-/Post-Test variables 4-2
 - increasing speed/Status Display Line 3-17
 - program sequence 4-1
 - standard sequence 5-10
- test flow control
- message steps 4-5
 - PRGMVARS 4-5
- Test I Stim V (TISV), 4-wire, Precise off 5-30

Test I Stim V, ATB test technique 5-26

test jacks

- debugging tests 9-7
- enabling/disabling 9-7
- panel 2-10
- panel functions 9-9
- using in analog debug 9-11

Test Page Allprint, test step control 3-49

Test Parameters, used for current test 9-7

test points

- listen window 9-8
- test envelope 9-8

test program

- auditing quality 5-6
- building programs from existing sections 3-11
- categories 3-4
- Converting manually 1-26
- converting to different software version 1-25
- copying 1-25
- copying for Multipanel test 1-21
- creating new 1-23
- creating new program 1-21, 1-24
- developing test strategy 5-4
- development strategies 5-2
- directories

- changing to alternate directory 1-21
- current path 1-21
- list of files 1-9
- managing TPD files 1-20
- selecting ICT or FST 1-22
- toggling between directories 1-22

editing basics 3-5

effects of power supply architecture 5-6

how to run 4-12

injecting faults to check program quality 5-6

losing edits quitting with Esc 1-6

master/subprogram relationship 4-3

merging *See* program merging

modifying defaults with PGEN.CFG 7-3

order of execution 3-6, 4-1

quality control measures 5-6

recording test techniques 5-4

removing from directory 1-21, 1-22

routing output data 2-5

sections 3-3

selecting 1-21, 1-22

test documentation/Documenter 9-40

test steps 3-3

un-converting programs 1-14

updating D.X programs 5-13

updating files 7-17

validating 7-17

validating entire program 9-73

Test Properties

adding pages 9-4

Averaging field 3-37

controlling power supplies 5-9

Controls section 3-36

- copying pages 9-4
- Current Page Box 3-33
- editing analog Test Properties 3-30
- Gray code
 - compared to analog 3-39
 - editing 3-39
 - fields 3-39
- Guard Mode field 3-37
 - active 3-37
 - Passive 3-37
 - semi-active 3-37
- Guard Node field 3-33, 3-35
- high tolerance field 3-33
- Higuard field 3-37
- Indicators box 3-32
- interconnect test step pages 3-23
- low tolerance field 3-33
- Measure field 3-33, 3-35
- Mode, On/Off 5-11
- number of analog test step pages 3-23
- Options box 3-32
- post-test options 3-41
- Power 5-10
 - mode selections 5-11
 - programmable 5-10
- Precise Mode field 3-37
- pre-test options 3-41
- Scale field 3-33
- Squelch field 3-35
- Stimulus field 3-33, 3-34
- Test Data area 3-33
- Test Type field 3-33
- upper area fields 3-32
- Value field 3-33
- Vector fields 3-40
- Wait field 3-36
- Wire Mode field 3-36
- test results, setting up Status Display Line 3-17
- test run, viewing on screen 4-12
- test section
 - how to run 4-11
 - Section Abort variable 3-14
 - section execute 2-4
 - Section Execute channels 2-4
 - sections listed 3-8
 - Validate 9-73
- test step
 - Component ID, adding to worksheet 3-11
 - controlling execution 3-36
 - copy Step Worksheet 3-11
 - delete component test 3-11
 - disable component ID 3-11
 - disabling globally 9-63
 - editing 3-13, 3-23, 3-25
 - enabling globally 9-63
 - how to run 4-11
 - multiple analog pages 3-30
 - Options menu functions 3-25
 - pages, order of execution 4-2
 - Revert function 3-25
 - Save function 3-25
 - See flow control
 - setting Options 3-41
 - Step Execute channels 2-4
 - Tools menu functions 3-25
 - Validate 3-25
- test step controls 3-48
 - APC 3-48
 - Backdrive Timeout 3-48
 - BSID Failure Reporting 3-48
 - Digital Tracer 3-48
 - Gray Code Groups 3-49
 - Test Page Allprint 3-49
 - Test Page Execution 3-48
 - Validate 3-48
 - VP Timeout 3-48
 - window 3-48
- test step options
 - adding default options values 3-41
 - copying options set up 3-41
 - I/O Control Options
 - editing 3-42
 - Input Device field 3-43
 - Output Device field 3-43
 - Variable field 3-43
- test strategy
 - coverage 5-5
 - repeatability 5-5
 - throughput 5-5
 - transportability 5-5
 - See *also* board-under-test
- test techniques
 - analog 5-4
 - digital 5-4
- test time, increased by duplicate test steps 7-14
- Test Types
 - relationship to test step structure 3-7
- test types 3-8
- Test V Stim I, ATB test technique 5-26
- Test V Stim V, ATB test technique 5-26
- Test V, ATB test technique 5-26
- TEST_NODES, Global Editor attribute-value 9-58
- TEST_TYPE, Global Editor attribute-value 9-60
- TEST_VALUE, Global Editor attribute-value 9-60
- testability, analyzing with PEP 5-2
- thermal EMF, reducing 3-32
- This Page, used for ECO revisions 4-10
- This Page, using for ECOs 4-10
- three-wire mode, switch matrix 5-28
- Tokenize Reports, description 3-18
- Tokenlog
 - configuring 2-4
 - directing messages to file 2-8
 - effects of Allprint 2-8
 - local control 2-7
 - output effects 2-8

- Tokenlog *continued*
 - preventing data routing 2-7
 - program setup 2-8
 - use with GFI 2-8
- TOKENLOG.DAT, file type and description 1-10
- TOL (tolerance), Global Editor attribute-value 9-60
- Tolerance Calculator
 - calculating tolerance limits 9-4
 - using with Windows 95 9-4
- tolerance limits, calculating 9-4
- Tools menu 9-1
 - Add Page 9-4
 - Convert to Longhand 9-5
 - Copy Page 9-4
 - Create Template 9-5
 - Delete Page 9-4
 - Edit Device Nodes 9-2
 - Edit Device Pins 9-2
 - Generate Test 9-3
 - Get Template Pins 9-2
 - Swap Poles 9-5
 - Test Parameters 9-7
 - Tolerance Calculator 9-4
- TOOLS.HLP, file description 1-8
- topology
 - accessing Topology Report 7-20
 - example of report after editing 7-23
 - example of report before editing 7-21
 - overlapping parts 7-19
 - Topology Report 7-19
 - updating 7-20
- TOPOLOGY.LST, file type and description 1-11
- TPD. *See* test program directory 1-6
- TRACER.DEF file 9-15
- Trailer
 - adding custom message 3-21
 - controlling test flow 4-4
 - editing message step 3-18, 4-5
 - section template 6-6
 - subprogram control 4-4
- Triac testing
 - on voltage 10-5
 - typical circuit 10-4
 - See also* SCR testing 10-4
- trigger function, vector test 9-10
- TTM files, removing existing file 8-6
- U**
 - U, input list token 6-15
 - updating D.X programs 5-13
 - UPDT_MOD.EXE, file description 1-7
 - UPL.DBF, file type and description 1-9
 - User flags
 - modifying 4-6
 - operation 4-8
 - user interface
 - editing basics 3-5
 - keyboard controls 3-1
 - keyboard navigation
 - Component Properties 3-1
 - in Controls section 3-2
 - in Options fields 3-2
 - in Test Data section 3-2
 - keyboard selection techniques 3-1
 - keyboard window operation 3-3
 - mouse
 - selection in Component Properties 3-1
 - selection techniques 3-1
 - navigating through menus 1-2
 - pop-up windows 3-2
 - searching in Component Select window 3-9
 - selection in Component Select window 3-1
 - Status Line location on Step Worksheet 3-5
 - Step Worksheet
 - description 3-5
 - portions 3-23
 - types 3-13
 - Test Control buttons 3-9
 - USRLIB syntax 7-8
 - Utilities, development and debugging 9-35
 - Utility menu
 - Board
 - description 9-35
 - using 9-39
 - Board Fault Coverage
 - description 9-35
 - Calibration
 - description 9-35
 - using 9-40
 - CRC
 - description 9-35
 - CRC, using 9-38
 - Diagnostics
 - description 9-35
 - verifying operations of tester 9-40
 - Documenter
 - description 9-35
 - providing ASCII text record 9-40
 - terminating 9-43
 - DOS Shell
 - description 9-35
 - exiting from 18xx 9-40
 - functions 1-13
 - Global Editor
 - adding guards 9-44
 - description 9-35
 - modifying in-circuit test files 9-44
 - overview of structure 9-44
 - Hardware Configuration 9-71
 - description 9-35
 - Nodefinder
 - description 9-35
 - identifying node numbers 9-35
 - using in analog test 9-36

- Reverse Nodefinder
 - description 9-35
 - using 9-38
- Supplemental Diagnostics
 - description 9-35
 - running 9-40
- V**
- Vacuum Control
 - activation and release 3-19
 - message step field 3-19
 - selecting vacuum in PRGMVARS 3-14
- Vacuum Delay, description 3-14
- Vacuum Delay, specifying in PRGMVARS 3-14
- vacuum operation
 - setting up in program chaining 4-16
 - subprogram 4-4
- Vacuum Select, description 3-14
- Validate 9-73
 - acceptance range 2-17, 9-75
 - Automatic Accept 2-18, 9-75, 9-79
 - Automatic Exclude 2-18, 9-75
 - Average Samples 9-75, 9-80
 - averaging 2-18
 - Calculate Wait Times 2-17, 9-75, 9-80
 - configuration window 2-17
 - default permissions 1-17
 - disabling page-by-page 9-73
 - enabling for Cap Phase 2-18
 - enabling for CapScan 2-20
 - enabling for DeltaScan 2-19
 - enabling for FrameScan 2-19
 - enabling for FrameScan Plus 2-19
 - enabling for WaveScan 2-19
 - establishing guard points 9-76
 - finding most effective guard points 9-79
 - guarding 9-73
 - inputs 9-73
 - invoking 9-73
 - login permissions 1-16
 - Multiscan 9-73
 - process overview 9-74
 - requirements for running 2-16
 - setting up 2-16
 - shorthand test mode 9-73
 - specifying measurement samples 2-18
 - specifying threshold Frame/WaveScan 2-19
 - Step Worksheet 9-73
 - test stability for capacitors and resistors 2-16
 - test step controls 3-48
- Variable, message step field 3-19
- variables, rules for shared variables 4-4
- VEC, input list token 6-16
- VECFILES.DOC, file type and description 1-11
- vector image templates
 - adding to library 8-20
 - creating 9-5
- vector measure pins 9-17
- Vector Performance
 - See also* VP
- vector processor, synchronization 10-25
- vector stim pins 9-17
- vector stim/meas pins 9-17
- vector templates
 - constraint checking 8-23
 - pattern acceptance 8-23
 - pattern checking 8-23
 - tied pin violations 8-23
 - translating from binary to ASCII 8-6
 - See also* templates
- vector test
 - ASCII/VLSI/PAL strategy 10-35
 - backdrive timeout 10-15
 - disable and guard timing sequence 5-40
 - disables 5-39
 - guards 5-39
 - pausing 10-25
 - Step Worksheet 3-40
 - synchronization with external clock 10-25
 - test strategies 10-34
- vector Test Properties
 - Data Window field 3-40
 - expanded 3-41
 - Marker field 3-40
 - Pins field 3-40
 - Position field 3-40
 - Size field 3-40
- vector test strategy
 - ASIC test development 10-48
 - CAE Simulation 10-37
 - CAE simulation 10-49
 - create 18XX device test 10-51
 - create 18XX worksheet 10-51
 - debug device test 10-51
 - debug test 10-52
 - run TDS conversion 10-51
 - running TDS 10-50
 - transfer ASCII vector files to VP 10-50
 - transferring CAE simulation files 10-49
 - developing ASIC/VLSI patterns from
 - ASIC/VLSI/PAL descriptions 10-43
 - board schematic only 10-44
 - Factron 3XX testers 10-40
 - Genrad 22XX testers 10-40
 - other testers 10-39
 - Sentry 20 testers 10-42
 - Teradyne J941, J971 testers 10-41
 - Teradyne L2, L3 testers 10-39
 - Z8000-series testers 10-39
 - Gray code development 10-44
 - PAL equation available 10-45
 - using Accugen 10-45
 - using Victory test tools 10-46
 - VLSI device in library 10-35

- vector test strategy *continued*
 - VLSI model resembles library model 10-36
 - create new model 10-37
 - use model "As Is" 10-37
 - when PAL Jedec test pattern available 10-45
- VF123456.GRD, file type and description 1-11
- VF123456.TST, file type and description 1-10
- VF123456.VEC, file type and description 1-10
- VICTORY
 - enabling Quiet Mode 3-50
 - tests 3-49
- Victory Boundary Scan tools 10-46
- VIMG, input list token 6-16
- VP External Clock
 - specifications 10-26
 - synchronizing vector tests 10-25
 - timing specifications 10-26
- VP Hold 10-25, 10-26
 - timing specifications 10-27
- VP timeout 2-10
 - Test Step Controls 3-48
- VP_12.IMG, file description 1-8
- VP3.IMG, file description 1-8
- VPIO.DLM, file description 1-7
- VPTHC.DLM, file description 1-7
- VRSNAP.LIB, file description 1-9
- VRTEMPL.LIB, file description 1-9

W

- wait times
 - defined 3-36, 5-25
 - using Fast Mode to eliminate 5-25
 - Vacuum Delay 3-14
- WAIT, Global Editor attribute-value 9-52
- WaveScan testing
 - disabling power 7-8
 - enabling or disabling Validate 2-19
 - minimum threshold 2-19
 - overview 5-3
 - specifying bias current 7-8
 - specifying fast mode 7-9
 - specifying inducer number in IPL syntax 6-17
 - specifying reference node 7-8
 - specifying threshold 7-8
 - Statistics report 9-7
- windows 1-2
 - changing color configuration 2-22
 - dialog boxes 1-2
 - pop-up 1-2
 - process windows 1-2
- Windows 95 calculator, using with 18xx 9-4
- wire modes
 - attribute 5-27
 - See also* analog wire modes 5-27
- WIRE, Global Editor attribute-value 9-60
- wiring constraints, templates 8-19
- WORK.BIN, file description 1-8
- wrist strap 9-8

- WSCAN, input list token 6-17
- WSCANBIAS syntax 7-8
- WSCANREF syntax 7-8
- WSCANREF, working the same as MSCANREF 7-8
- WSCANTHRESH syntax 7-8

Z

- Z, input list token 6-14
- Z850 program generation 10-52
- Z875 program generation 10-52
- zener variable, PGEN.CFG 7-7