



Agilent Technologies

ADS 2008
January 2008
Using Circuit Simulators

Advanced Design System 2008

© Agilent Technologies, Inc. 2000-2008

395 Page Mill Road, Palo Alto, CA 94304 U.S.A.

No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Agilent Technologies, Inc. as governed by United States and international copyright laws.

Acknowledgments

Mentor Graphics is a trademark of Mentor Graphics Corporation in the U.S. and other countries. Microsoft®, Windows®, MS Windows®, Windows NT®, and MS-DOS® are U.S. registered trademarks of Microsoft Corporation. Pentium® is a U.S. registered trademark of Intel Corporation. PostScript® and Acrobat® are trademarks of Adobe Systems Incorporated. UNIX® is a registered trademark of the Open Group. Java™ is a U.S. trademark of Sun Microsystems, Inc. SystemC® is a registered trademark of Open SystemC Initiative, Inc. in the United States and other countries and is used with permission. MATLAB® is a U.S. registered trademark of The Math Works, Inc.. HiSIM2 source code, and all copyrights, trade secrets or other intellectual property rights in and to the source code in its entirety, is owned by Hiroshima University and STARC.

Errata The ADS product may contain references to "HP" or "HPEESOF" such as in file names and directory names. The business entity formerly known as "HP EEsof" is now part of Agilent Technologies and is known as "Agilent EEsof". To avoid broken functionality and to maintain backward compatibility for our customers, we did not change all the names and labels that contain "HP" or "HPEESOF" references.

Warranty The material contained in this document is provided "as is", and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license. Portions of this product include the SystemC software licensed under Open Source terms, which are available for download at <http://systemc.org/>. This software is redistributed by Agilent. The Contributors of the SystemC software provide this software "as is" and offer no warranty of any kind, express or implied, including without limitation warranties or conditions or title and non-infringement, and implied warranties or conditions merchantability and fitness for a particular purpose. Contributors shall not be liable for any damages of any kind including without limitation direct, indirect, special, incidental and consequential damages, such as lost profits. Any provisions that differ from this disclaimer are offered by Agilent only.

Restricted Rights Legend If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Contents

- Simulation Basics
- - Simulation Types
 - - About Licensing
 - Common Simulation Usage
 - Working with the Examples Directory
 - The Simulation Process
 - - Comparison of Schematic Wizard to Smart Simulation Wizard
 - Using the Schematic Wizard
 - - Accessing the Schematic Wizard
 - - Starting a New Project
 - - Opening a New Schematic Window from Main
 - - New Design
 - - Simulation Error
 - Schematic Wizard Start Page
 - Schematic Wizard Navigation
 - Creating a Circuit
 - - Circuit Setup Step
 - - Naming Ports Step
 - - Finish Circuit Creation Step
 - Creating a Simulation Schematic
 - - Application Selection Step
 - - Circuit Selection Step
 - - Simulation Setup Step
 - - Port Specification Step
 - - Schematic Completion Step
 - Correcting an S-Parameter Simulation Schematic
 - - Term Properties Specification Step
 - - Term Placement Step
 - - Schematic Completion Step
 - Correcting a Simulation Schematic with No Simulation Controller
 - - Template Setup Step
 - - Schematic Completion Step
 - Using the Smart Simulation Wizard
 - Selecting Simulation Controllers
 - Using the Simulator Options Component
 - - Setting Miscellaneous Simulation Options
 - - Setting Convergence Options
 - - - Default Preset Tolerance Values
 - - - Current Relative Tolerance, Current Absolute Tolerance
 - - - Voltage Relative Tolerance, Voltage Absolute Tolerance
 - - - Setting Advanced DC Convergence Options
 - - Setting Output Options
 - - Setting DC Solution Options
 - Sweeping Parameters
 - Optimizing a Design
 - Working with Expressions

Advanced Design System 2008

- - Expressions Examples
- Saving and Controlling Simulation Data
- - Automatically Displaying Simulation Data
 - Using a DisplayTemplate Component
 - Manually Displaying Simulation Data
 - Selectively Saving and Controlling Simulation Data
 - - Output Tab Fields for Simulation Controllers
- Displaying Simulation Parameters on the Schematic
- - DC Simulation Display Options
- Controlling a Simulation
- Simulating from a Layout
- Viewing DC Solutions
- - Viewing Device Operating Point Data
- Displaying Simulation Results
- - Tuning
- Reusing Simulation Solutions
- Analog/RF Simulation Computations and Convergence Criteria
- - - DC analysis
 - Transient analysis
 - Harmonic Balance (HB)
- Solving Nonlinear Algebraic Equations
- Common Circuit Simulation Methods
 - - Backward Euler
 - Trapezoidal Rule
 - Backward Difference Formulas (Gear's methods)
 - Truncation Error
 - Local Truncation Error (LTE)
 - Global Truncation Error (GTE)
- Convergence Criteria
 - - Residue criterion
 - Update criteria
- Using Continuation Methods
 - - Source Stepping
 - Gmin Stepping
 - Arc-Length Continuation
- Preventing Convergence Problems
 - - Clearing Highlights from Items Causing Simulation Errors
- Using Circuit Simulators for RF System Analysis
- ◦ Applicable Simulation Components
- Applicable Measurements
- Fundamentals of Using Circuit Simulators for System Analysis
- - - Block diagram of a typical RF system
- Example
 - - Example RF system
- Budget Analysis
- - Using the Perform Budget Simulation Option
- Using Budget Measurement Components
- Adding a Budget Path
- Using Budget Measurement Functions
- Using the BudLinearization Component

Advanced Design System 2008

- Budget Examples
 - Calculating Gain and Noise Figure
 - Using the AC simulator to obtain budget gain and noise figure
 - Calculating Spurious Signals and TOI
 - Spectral data for RF_SYS1.dsn
 - System TOI example
 - Obtaining Budget Incident Power and Gain
 - Using bud_gain, bud_snr and bud_pwr_inc
 - Obtaining Group Delay Data
 - Using the S-parameter simulator to obtain group delay data
 - Budget Analysis Capabilities
 - Budget Analysis calculations are based on both voltages and currents
 - Budget Analysis setup requires close attention
 - Budget measurements work with only AC or HB simulations
 - Budget measurements may not work with both AC and HB simulations
 - Different AC and HB usage of functions that can work with both simulations
 - HB Budget Analysis needs a frequency plan
 - Frequency plans are internally generated
 - Common issues about Budget Noise Figure
 - Budget Noise Figure is really a reformulated signal-to-noise ratio
 - Using the pinNumber argument in budget functions
 - Using the SrcIndx argument in budget functions
 - Budget Path - how to use it effectively
 - Mixer2 and MixerIMT2 components issues
 - Clipped values in bud_nf and bud_tn
 - Contextual meaning of input parameters: Beware of short syntax form
 - Using IMT-Based Mixer Models in Spurious Signal Analysis
 - Interpreting IMT Charts
 - O-type IMT data for second degree RF nonlinearity mixed with fifth degree LO nonlinearity at PRF = -10 dBm and PLO = +7 dBm
 - B-type IM table showing complex spur information for dual RF and single LO mixing
 - Interpolation of IMT Data
 - Modeling IMT with Conversion Gain
 - System Noise Analysis
- Parameter Sweeps and Sweep Plans
 - Conducting Sweeps
 - Using Options under the Simulator Sweep Tab
 - Using Sweep Controllers
 - Basic Procedures
 - Using ParamSweep to Sweep Two Parameters
 - Example setup for a hierarchical sweep
 - Define Inner and Outer Variables
 - Define the Inner Variable in the Simulation Controller
 - Establish a Sweep Plan for the Outer Variable
 - Launch the Simulation and Display Data
 - Nesting Parameter Sweeps with Multiple Items
 - Using SweepPlans to Perform Fine and Coarse Sweeps
 - Example setup for using two sweep plans
 - Define Frequencies
 - Establish Two Sweep Plans

Advanced Design System 2008

- Define Sweep Frequencies and Sweep Plan in the Simulation Controller
 - Launch the Simulation and Display Data
 - Additional Examples
- Recommendations and Tips
- - Ensuring that Sweep Results are Displayed Correctly
 - Controlling the Amount of Data Sent to the Dataset
 - Using Sweeps in Monte Carlo Analysis
- SweepPlan Controller
- Parameter Sweep Controller
- - Simulations Tab
 - Display Tab
- Working with Data Files
- - Supported Data Formats
 - Making a Data File
 - Saving a Data File
 - Using Data Files, Datasets, and Data Access Components
 - Reading and Writing Data Files
 - - Starting and Exiting the Data File Tool
 - Parts of the Data File Tool
 - - Data File Tool in read mode
 - Data File Tool in write mode
 - Reading a File
 - Writing to a File
 - Examples
 - Touchstone SnP Format
 - - Linking an .snp File to an SnP Component
 - Overview
 - Basic SnP File Format
 - - The Option Line
 - Default Option Line
 - Option Line Examples
 - Data Lines
 - - Data-line Formats
 - Adding Comments to Data Files
 - Adding Noise Parameters to an SnP File
 - - Example File Containing Noise Data
 - Applying the SnP Format, and Examples
 - - Guidelines
 - G-Parameter Files
 - H-Parameter Files
 - S-Parameter Files
 - Y- and Z-Parameter Files
 - - Y- (Z-) Parameter 1-Port MA and RI File Formats
 - Y- (Z-) Parameter 2-Port MA and RI File Formats
 - Y- (Z-) Parameter 3-Port MA and RI File Formats
 - Y- (Z-) Parameter 4-Port MA and RI File Formats
 - Y- (Z-) Parameter 3-Port File Example
 - Y- (Z-) Parameter 5- to 99-Port File Formats
 - - Y- (Z-) Parameter 10-Port File Example (at One Frequency)
 - Discrete Format

Advanced Design System 2008

- - Selecting a Row
 - Using the File with a DAC
 - Example
- Model MDIF Files
 - - Reading a Model MDIF file using a DataAccessComponent
- PDF Format
 - - Guidelines for .pdf
 - Example PDF File
 - Interpretation of PDF data
- S2PMDIF Format
 - - General File Structure
 - Guidelines
 - Example S2PMDIF File
 - Additional Examples: ACDATA and NDATA Blocks
- P2D Format
 - - Guidelines for .p2d
 - The ACDATA Block
 - ACDATA Block Examples
 - The NDATA Block
 - The IMTDATA Block
 - Example .p2d File
- S2D Format
 - - Guidelines for .s2d
 - The ACDATA Block
 - The NDATA Block
 - Understanding GCOMP Data
 - The GCOMP1 Block
 - The GCOMP2 Block
 - The GCOMP3 Block
 - The GCOMP4 Block
 - The GCOMP5 Block
 - The GCOMP6 Block
 - The GCOMP7 Block
 - Complete .s2d File Example
- IMT Format
 - - O-Type IMT Format
 - Example O-Type IM Table
 - A-Type IMT Format
 - Example A-Type IM Table
 - B-Type IMT Format
 - Example B-Type IM Table
- SPW Format
 - - Guidelines for .ascsig
 - Example .ascsig Files
- TIM Format
 - - The BINTIM Format
 - Guidelines for .tim Files
 - TIMEDATA Block
 - Example .tim Files
 - Time periodic voltage vs. Time with time period 32 μ sec.

Advanced Design System 2008

- Generic MDIF
- CITIfile Data Format
 - - Overview
 - Data Formats
 - File and Operating System Formats
 - CITIfile Definitions
 - - Package
 - - Header
 - - Data Array
 - - Keywords
 - CITIfile Examples
 - - Display Memory File
 - - Agilent 8510 Data File
 - - Agilent 8510 3-Term Frequency List Cal Set File
 - - 2-Port S-Parameter Data File
 - CITIfile Keyword Reference
 - - CITIfile Keywords and Definitions
 - CITIfile Guidelines
 - Converting Between Disk Formats
 - - HFS
 - - DOS
 - CITIfile Device-Specific Definitions
 - - Network Analyzer (#NA) Definitions
 - - - Network Analyzer Keyword Commands
 - Error Array Numbering
 - - Network Analyzer Error Coefficient Arrays
 - Disk Filename Requirements
 - Agilent 8510 Series CITIfile
 - - Agilent 8510 CITIfile Prefixes
 - Agilent 8700 Series CITIfile
 - - Agilent 8700 CITIfile Filenames
- ADS Simulator Input Syntax
 - Setting Environment Variables
 - - - Environment Variables Required for the ADS Simulator (ADSsim)
 - Platform-Specific Variables
 - Codewording and Security
 - Running a Simulation from the Command Line
 - General Syntax
 - - Typographic Conventions
 - The ADS Simulator Syntax
 - - Field Separators
 - Continuation Characters
 - Name Fields
 - Parameter Fields
 - Node Names
 - Lower/Upper Case
 - Units and Scale Factors
 - - - Fundamental Units in ADS
 - - Recognizing Scale Factors
 - - Predefined Scale Factor Words

Advanced Design System 2008

- Scale Factor Units
 - Scale Factor Prefixes
- Booleans
- Global Nodes
- Device Pin Names
- Comments
- Statement Order
- Naming Conventions
- Currents
 - State currents
- Integer Formats for Simulator Input
- Instance Statements
- Model Statements
- Subnetwork Definitions
- Expression Capability
- C-Preprocessor
 - File Inclusion
 - Library Inclusion
 - Macro Definitions
 - Conditional Inclusion
- Data Access Component
- Reserved Names and Name Spaces
 - ADS Simulator Namespaces and Associated Reserved Name Groups
 - Parser Reserved Names Group
 - Reserved Names Group
 - Predefined Expression Reserved Names Group
 - Predefined Variable Reserved Names Group
 - Predefined Function Reserved Names Group
- Preparing a Circuit for Simulation in ADS
- ◦ Using Current Probes
- Naming Nodes
- Using NodeSet and NodeSetByName Components
 - Using NodeSet or NodeSetByName to Facilitate a Simulation
 - NodeSet Fields
 - NodeSetByName Fields
- Highlighting Nodes
 - Clearing Highlights
- Using Constants, Variables, and Functions
 - Pre-Defined Constants
 - Pre-Defined Variables
 - Pre-Defined Functions
 - The Effect of Expressions on Units
- Applying Measurements
 - Quantities Measurements Can Reference
 - Example
 - Simulation Output Names
- Using Simulation Templates
 - Simulation Template Descriptions
- Using Simulation Instrument Components
- S-Parameter Test Labs and Sequencer

Advanced Design System 2008

- - Creating a Test Bench
 - - Port component
 - - Supported terminal components for S-parameter test lab
 - S-Parameter Test Labs
 - - Test Lab Usage Rules
 - - Configuring an S-Parameter Test Lab
 - Schematic window with design named using `_TL` S-parameter test lab naming convention
 - Test Lab S-parameter controller icon, SP Lab.
 - Completed S-parameter test lab
 - - Notes
 - - Data Display Naming Convention for Simulation Results
 - Standard Results Data
 - Measurement Equation Results Data
 - - Optimization and Statistical Analysis
 - Sequencer
 - - Creating a Sequence
 - - Notes
 - - Examples
 - - Usage Rules
 - Using Measurement Equations with a Test Lab or Sequencer
 - Improving Test Lab Simulation Efficiency
- RefNets
 - - - `RefNetTB` and `RefNetDesign`
 - - Notes:
 - RefNetTB Using an S-Parameter Test Lab
 - - Example amplifier circuit
 - - Procedure
 - Input subnetwork
 - Active device subnetwork
 - Output subnetwork
 - Desired effect of test bench `A_TB` (input)
 - Test bench `A_TB` completed
 - Desired effect of test bench `B_TB` (input) with `Term` and `RefNetTB` placed
 - Test bench `B_TB` completed
 - Desired effect of test bench `C_TB` (output) with `RefNetTB` and `Term` placed
 - Test bench `C_TB` completed
 - Completed S-parameter test lab incorporating `RefNetTB`
 - Test bench for entire circuit chain
 - S-parameter test lab incorporating sub-circuit and entire circuit, `ABC_TB`, test benches
 - RefNetDesign - File Based Termination
 - - Top-level one-port design file to read in S-parameter data file
 - - Top-level design that references a file-based one-port network to realize a file-based termination
 - An ADS Simulation Example
 - Placing Circuit Sources
 - - Ensuring Sources are Connected Properly
 - Specifying Points for Collecting Data
 - Selecting a Simulation Type
 - - Editing Simulation Parameters
 - Selecting a Sweep Type and Plan
 - Setting Simulation Options

- Modifying the Simulation Setup
- Starting the Simulation
- Displaying Simulation Data
 - Viewing the DC Solution
 - Viewing the Detailed Device Operating Point
 - Viewing the Brief Device Operating Point
 - Viewing More Results in the Data Display

Simulation Basics

This documentation provides information that applies to the Analog/RF simulation in ADS. It also contains general information about the various simulation controllers that are available in ADS. Before using this documentation, you should see [Advanced Design System Quick Start](#) and [Schematic Capture and Layout](#) to review the whole product.

For details about running an Analog/RF simulation, you should also see [Preparing a Circuit for Simulation in ADS](#) and [An ADS Simulation Example](#), then continue with other areas of the Simulation documentation that provide details about the simulation controllers.

Simulation Types

ADS provides simulators that enable you to simulate circuits and RF systems designed for specific objectives. The following table provides brief descriptions of the available simulation controllers. See the documentation for the Analog/RF simulation controllers for complete information about each one.

Simulator	Description
DC	Fundamental to all simulations, it performs a topology check and an analysis of the DC operating point of a circuit. See DC Simulation .
AC	Obtains small-signal transfer parameters, such as voltage gain, current gain, and linear noise voltage and currents. This simulator is useful in designing passive circuits and small-signal active circuits such as low-noise amplifiers (LNAs). See AC Simulation .
S-parameter	Provides linear S-parameters, linear noise parameters, transimpedance (Z_{ij}), and transadmittance (Y_{ij}), by linearizing the circuit about the DC operating point and performing a linear small-signal analysis that treats the circuit as a multiport. Each port is turned on

Advanced Design System 2008

	<p>sequentially. S-parameters can be converted to Y- and Z-parameters. This simulator can be used to achieve many of the same design goals as the AC simulator. See S-Parameter Simulation.</p>
Harmonic Balance	<p>Uses nonlinear harmonic-balance techniques to find the steady-state solution in the frequency domain. This simulator is useful in designing RF amplifiers, mixers, and oscillators. A Krylov subspace technique is available to reduce memory requirements and increase the speed of solution. This option is useful in designing large RF integrated circuits or RF/IF subsystems, where a large number of devices or large numbers of harmonics and intermodulation products are involved. See Harmonic Balance Simulation.</p>
Large-signal S-parameter (LSSP)	<p>A type of harmonic balance simulation, it performs large-signal S-parameter analyses to represent the nonlinear behavior of items such as power amplifiers. The accompanying P2D simulator available in ADS can be used to speed up subsequent analyses. See Large-Signal S-Parameter Simulation.</p>
P2D	<p>Generates a .p2d file that can be used to describe the behavior of a file-based component (such as the AmplifierP2D component, available in the System-Amps & Mixers library). See P2D Simulation.</p>
Gain Compression (XdB)	<p>Seeks a user-defined gain-compression point at which an actual power curve deviates from an idealized linear power curve. This is useful in power amplifier design. See Gain Compression Simulation.</p>
Circuit Envelope	<p>Uses a combination of frequency- and time-domain analysis techniques to yield a fast and complete analysis of complex signals such as digitally modulated RF signals. It represents input waveforms as RF carriers with modulation envelopes that are described in the time domain. This is useful in designing circuits and systems involving modulators/demodulators or complex modulated signals. See Circuit Envelope Simulation.</p>
Transient/Convolution	<p>Solves a nonlinear circuit in the time domain, and linear components can be simulated by means of convolution or a simplified equivalent-circuit model. See Transient and Convolution Simulation.</p>
RF System Budget Analysis	<p>Determines the linear and nonlinear characteristics of an RF system comprising a cascade of two-port linear or nonlinear components. The RF system may also include automatic gain control (AGC) loops to control gain and set power levels at specific points in the RF system. See</p>

[RF System Budget Analysis.](#)

About Licensing

These simulators require licenses to run a simulation. Confirm that the simulator of interest is included with your purchase. ADS allows you to create a circuit, but if you do not have the correct license you will not be able to simulate it.

Common Simulation Usage

The following table describes some common design objectives and the simulators that would be appropriate to each. The simulators are listed in the order they would generally be applied.

Simulator used for Various Design Types

Design	Simulator	Comments
Filter	DC	
	AC	
	S-parameter	
Mixer	DC	
	AC	Test for AC frequency conversion (also known as frequency-converting AC, or FCAC). Applies to system mixer models only.
	Harmonic Balance	Select nonlinear noise option to obtain noise figure.
	Transient	
	Envelope	
	XDB	

Power amplifier	DC	
	AC	
	S-parameter	
	Harmonic Balance	Test for load-pull characteristics.
	LSSP	Also use the P2D simulator to generate a .p2d file.
	XDB	Find gain-compression point.
	Transient	
	Envelope	Find ACPR (adjacent-channel power ratio).
Transceiver	DC	
	AC	Test for AC frequency conversion (FCAC).
	Harmonic Balance	
	Envelope	
	Budget	RF system must be a cascade of two-port components.
Oscillator	DC	
	S-parameter	
	Harmonic Balance	Check for power spectra and phase noise.
	Envelope	Check for startup switching.
Phase-locked loop	Envelope	Check for transient responses.

Working with the Examples Directory

ADS includes example projects containing designs that you can open and run.

Most of the designs discussed in this manual are available in the location where you installed ADS, typically `$HPEESOF_DIR/examples` directory. For detailed information about locating and opening ADS example projects, see [Schematic Capture and Layout](#). For documentation on ADS examples organized by application, see the Examples Documentation.

Briefly, here is how to get started using examples. ADS examples include projects and templates. On UNIX, these

projects are read-only directories. To work with an example project, you must first make a copy in a directory for which you have write permission. Windows users should also copy these examples to preserve the integrity of the examples. For convenience in keeping track of designs, you may want to create directory names that mirror those in the examples directory.

Do not copy projects by using your operating system alone. Use these methods:

- Use the Copy Project command in the Main window to copy entire projects to your local directory.
- Copy projects as part of the software installation procedure.

This ensures that all files that are part of the project are copied.

The Simulation Process

The following list shows the basic simulation process, and the sections to see for more information:

- Create your schematic, then add current probes and wire/pin labels to identify the nodes from which you want to collect data.
- Select a simulation method, specifying parameters as necessary. The parameters you specify are based on the type of simulation you choose, the simulation options you require, and whether you're sweeping parameters, using expressions, and optimizing your design.
[Selecting Simulation Controllers](#)
[Using the Simulator Options Component](#)
[Sweeping Parameters](#)
[Working with Expressions](#)
- Select a name for the dataset. This is where the simulation data will be saved.
[Saving and Controlling Simulation Data.](#)
- Run the simulation.
[Controlling a Simulation](#)
- View DC data by annotating the schematic with DC solutions and by viewing brief or detailed device operating point data.
[Viewing DC Solutions](#)
- Display additional results using the Data Display.
[Displaying Simulation Results](#)
- Optimize and tune a design.
[Displaying Simulation Results](#)
For complete information, see [Tuning, Optimization, and Statistical Design.](#)

These are the basic steps. It is possible to develop very detailed, complex simulations, but the process, for the most part, remains the same. The remainder of this chapter gives an overview of these steps. If you are new to using ADS, or you use it infrequently, two wizards are available to help you with several steps in the process:

- [Using the Schematic Wizard](#) follows the standard ADS use-model to help you with design creation and setting up

- a simulation.
- [Using the Smart Simulation Wizard](#) requires an existing design, and helps you sequence several simulations on the same device.

These wizards provide different features which may determine which one you prefer to use. The Schematic Wizard helps you develop a design and prepares it for a simulation as if you are working directly in the design environment. The Smart Simulation Wizard requires that you already have a design available and adds a Smart Simulation module for sequencing simulations to the design. The following table presents additional differences:

Comparison of Schematic Wizard to Smart Simulation Wizard

Schematic Wizard	Smart Simulation Wizard
Sets up the initial schematic, but does not include simulation sequencing, allowing support for more application types.	Has two major features: <ul style="list-style-type: none"> • Set up initial schematic. • Simulation sequencing on a common device under test.
Uses the standard ADS use-model for schematic development.	Simulation sequencing is a non-standard use-model in ADS.
Allows selection of application categories and applies to more application types.	Simulation sequencing is limited to certain types of devices under test, and is not generally extendable to all application categories.
Provides selection by application, and includes schematics by simulation-type.	Does not include selection of simulation-type.
Assists with some schematic corrections when simulation errors appear.	Does not include error correction.

Using the Schematic Wizard

The Schematic Wizard is provided to assist new ADS users or those who use it infrequently in performing the basic steps associated with schematic creation. Two options for schematic creation are available:

- Creating a design in the schematic that can be used as a component or subnetwork in another ADS design.
- Setting up a simulation based on a desired application or simulation type. The schematic can include an existing or sample test circuit, or simply provide a simulation schematic into which a test circuit can be placed.

The Schematic Wizard guides you through a sequence of steps gathering information from you about the type of schematic you want to create. Based on your inputs, the wizard automatically creates the specified schematic components. The wizard then provides you with instructions for completing the schematic manually, and for invoking the simulator when applicable. The simulations are set up to automatically display the results after successful simulations.

Accessing the Schematic Wizard

Access to the Schematic Wizard is controlled by the Schematic Wizard and Create Initial Schematic Window preference options. You can set these options in the Main Preference dialog (in the ADS Main window: Tools > Preferences). The Schematic Wizard automatically appears when you perform certain actions related to the Schematic window.



Note

The Schematic Wizard is not available for designs manipulated through any Layout window.

Starting a New Project

When you start a new project within ADS, the Schematic Wizard will appear if you have selected both preferences Schematic Wizard and Create Initial Schematic Window.

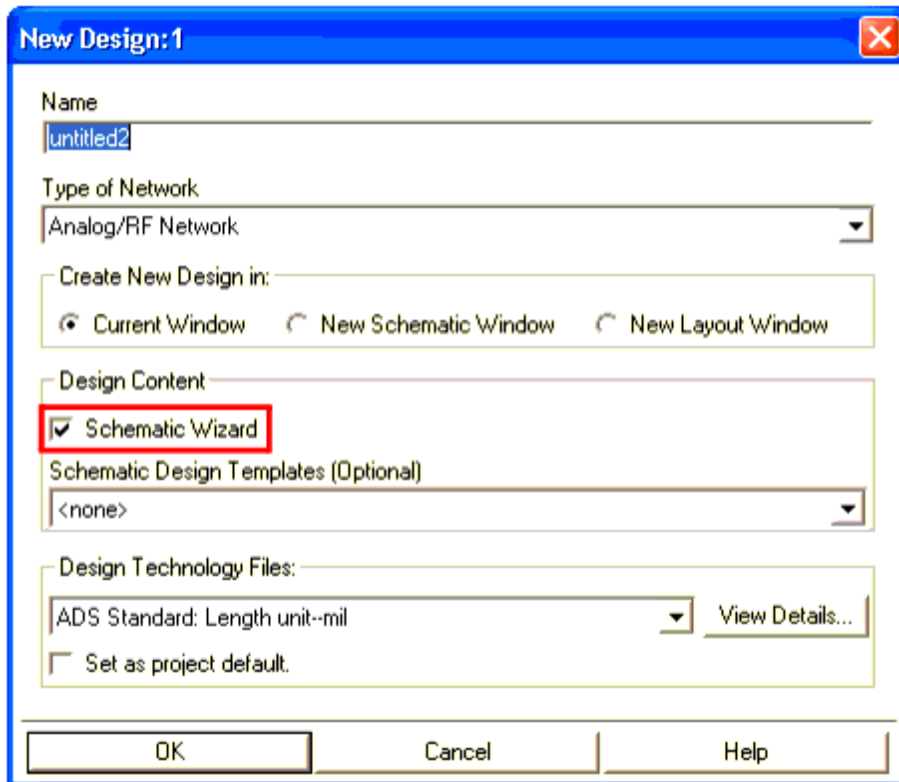
Opening a New Schematic Window from Main

When you open a new Schematic window from the ADS Main window (using the toolbar button or Window > New Schematic), the Schematic Wizard will appear if you have selected the Schematic Wizard preference. The wizard will not appear if the new Schematic window is requested from an existing Schematic or Layout window.

New Design

Requesting a new design from any ADS window opens the New Design dialog. This dialog also contains a Schematic Wizard option. (This option is not accessible when a new design is opened from any Layout window.) If the wizard option is selected, the Schematic Wizard will open after you click OK in the New Design dialog. The default setting for the Schematic Wizard option is controlled by its preference setting in the Main Preference dialog. If it is selected in the Main Preference dialog, it will be checked by default in the New Design dialog.

When setting the design content options, the Schematic Wizard and Schematic Design Templates cannot be used at the same time since they both place components on the Schematic. Therefore, selecting the Schematic Wizard option automatically clears any request for a Schematic Design Template. Similarly, requesting a Schematic Design Template automatically clears the Schematic Wizard option.



Simulation Error

If you run a simulation using a simulation schematic that is not properly configured, the simulator will terminate with errors. If the Schematic Wizard option is selected, the wizard will appear automatically after the simulation attempt has completed if the error occurs due to either of these reasons:

- A schematic with an S-parameter simulation controller does not contain any valid Term components.
- The simulation schematic does not contain a valid simulation controller.

You can use the Schematic Wizard to help you correct the schematic. See [Correcting an S-Parameter Simulation Schematic](#) and [Correcting a Simulation Schematic with No Simulation Controller](#).

Schematic Wizard Start Page

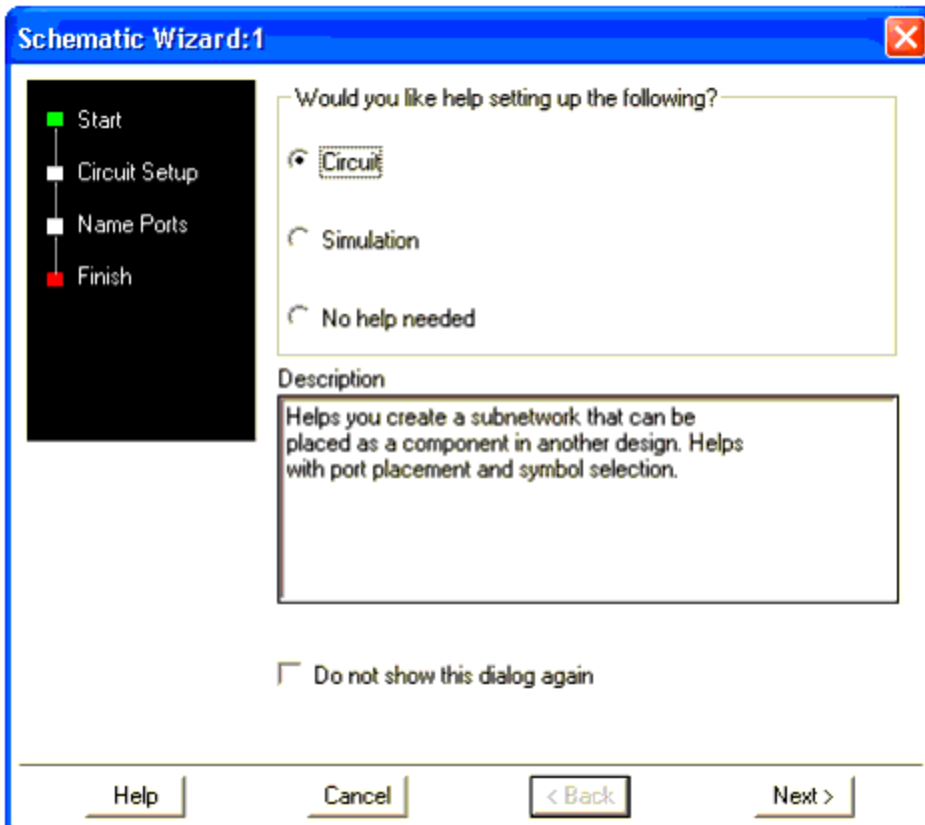
When the Schematic Wizard appears, the Start page presents you with the following choices for proceeding with the schematic creation:

- Circuit Helps you create a subnetwork that can be used as a component in another ADS design. See [Creating a](#)

[Circuit.](#)

- Simulation Helps you set up a simulation and place a circuit to be simulated. See [Creating a Simulation Schematic](#).
- No help needed Dismisses the wizard.

When the Schematic Wizard is accessed by starting a new project or opening a new schematic window, this page also offers the option Do not show this dialog again. Selecting this option will turn off the Schematic Wizard preference. You can select the Schematic Wizard preference option again in the Main Preferences dialog.



Schematic Wizard Navigation

The Schematic Wizard provides a navigation bar in the upper left corner of the window. This navigation bar indicates your progress through the steps required to complete the schematic setup, with a green box next to the current step. The steps on the navigation bar change depending on the options you select in the wizard.

Use the Back and Next buttons to move back and forth through the steps. The Back button is active for all steps except for Start. The Next button remains inactive for each step until you make a valid selection. When you have completed all required steps, use the Finish button to initiate schematic creation. The final step also provides an

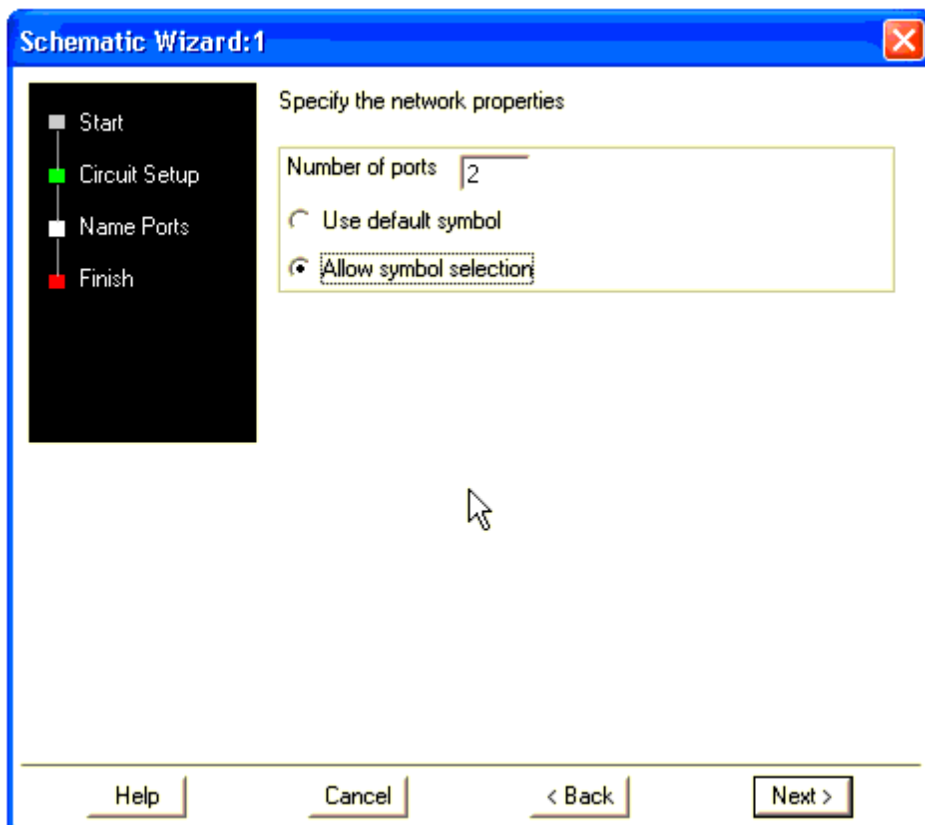
option to have instructions appear that assist you in the remainder of the schematic creation process. This option is persistent, so the setting for this option is the default the next time the wizard is used.

Creating a Circuit

Choosing the Circuit option from the Start page enables you to create a subnetwork that can be placed in another ADS design. Creating a subnetwork involves placing and naming ports, and selecting a symbol that will represent the circuit. The steps associated with this choice are:

- Circuit Setup
- Name Ports
- Finish

Circuit Setup Step



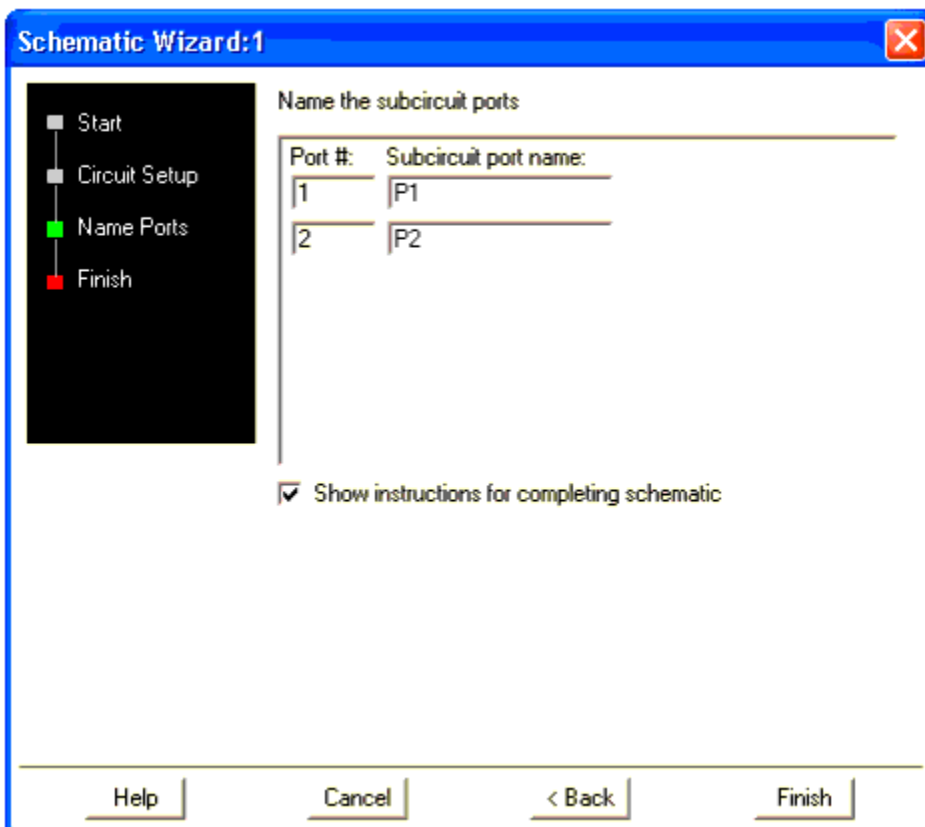
Network ports represent the connections of a circuit to the outside world. In this step of the design, you must specify how many ports you anticipate for your circuit. For example, if designing an amplifier from a transistor and passive components, the circuit might have an input, output, and bias connection. In this case, you should request three

network ports.

A symbol is used to represent a subnetwork when it is placed within another ADS design. Each connection point on the symbol will correspond to one of the subnetwork ports. ADS can automatically generate a symbol for you based on the number of ports specified, which is achieved using the Use default symbol option. However, if you want your symbol to be representative of the underlying subnetwork, use the Allow symbol selection option. You will be provided with a large set of possible symbols from which to choose.

Correctly specifying the number of ports at this stage of the process will ease the work in creating the subnetwork. However, if you later determine that you must add or remove a port from the circuit, this can be done manually. It is important, however, that the change is made to both the circuit design and the symbol in order for the subnetwork to function properly.

Naming Ports Step



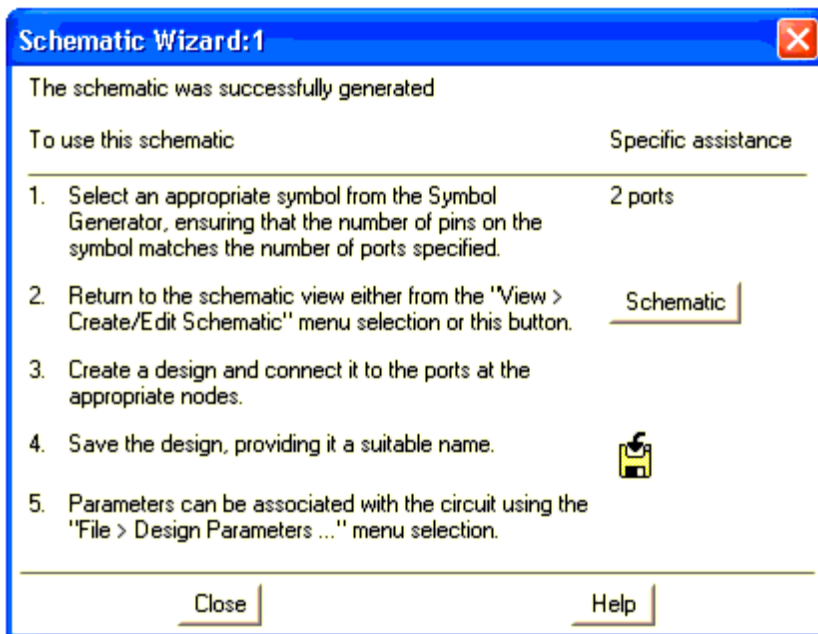
Ports created in ADS assume default names of P1, P2, etc. However, to make the port designations more physically meaningful, it is possible to specify alternate names for these ports. In this phase of the process, you may either use the default names provided or type in the desired names for each port.

If you chose the Use default symbol option in the Circuit Setup step, you are given the opportunity to determine

whether or not you would like instructions to appear after the wizard has completed the setup. However, if you chose the Allow symbol selection option, this option regarding supplemental instructions is not available. In this case, the instructions will be shown to help you create the custom symbol and return to the schematic view following symbol selection.

Finish Circuit Creation Step

Successful completion of the wizard leads to a starting design in which ports are placed. If you chose to allow ADS to create a default symbol for you, you will see the requested number of ports placed on the schematic. You can view the symbol that has been created for you using the View > Create/Edit Schematic Symbol menu selection. If you go to the symbol view, you can return to the schematic using the View > Create/Edit Schematic menu selection. If you chose to have supplemental instructions provided, a dialog will also appear, similar to the following figure, containing these instructions. You can move this dialog out of the way to interact with the schematic.



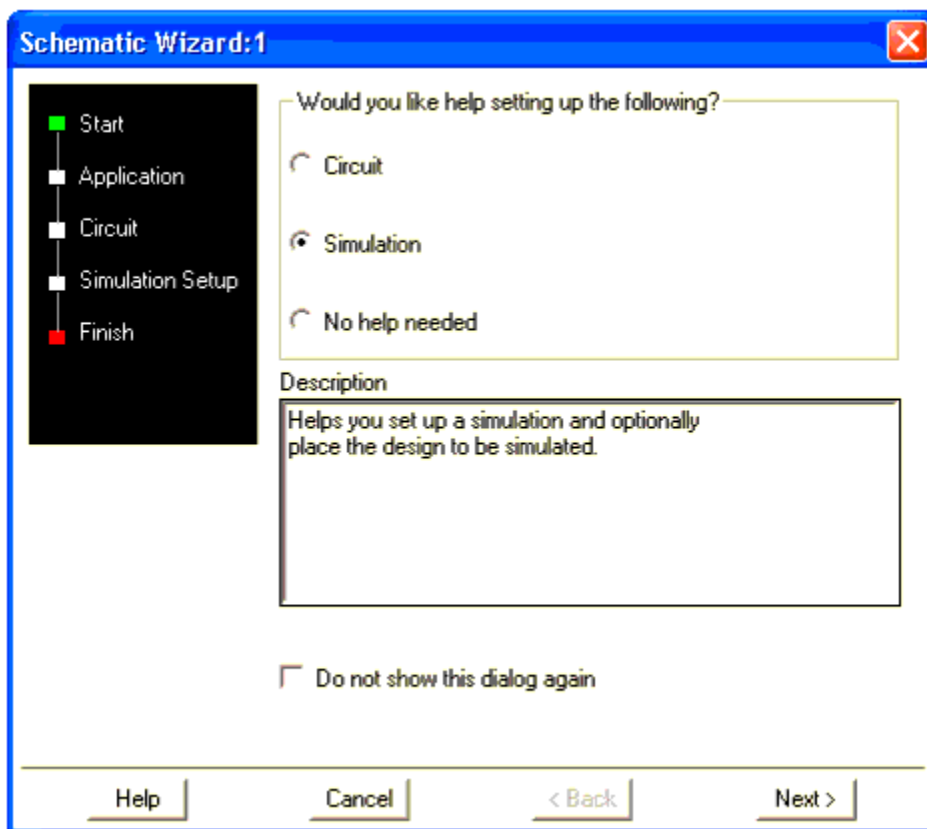
If you chose to create a custom symbol in the Circuit Setup step, the Symbol Generation dialog box appears with a selection of different symbols. You can scroll through these selections and choose a suitable symbol. Be sure, however, that the number of pins on the symbol matches the number of ports specified on the wizard. Once you have selected a symbol, you can return to the schematic view either using the Schematic button on the Schematic Wizard's instruction dialog or the View > Create/Edit Schematic menu selection. If you chose a symbol with a number of pins that does not match the specified number of ports, you will be warned of this problem when you return to the schematic view provided that the dialog containing the instructions is currently visible.

In either case, once you are in the schematic view, you can create the appropriate design and connect it to the ports at the proper nodes in the circuit. Once the design has been saved and provided a suitable name, it will be ready for placement in other designs.

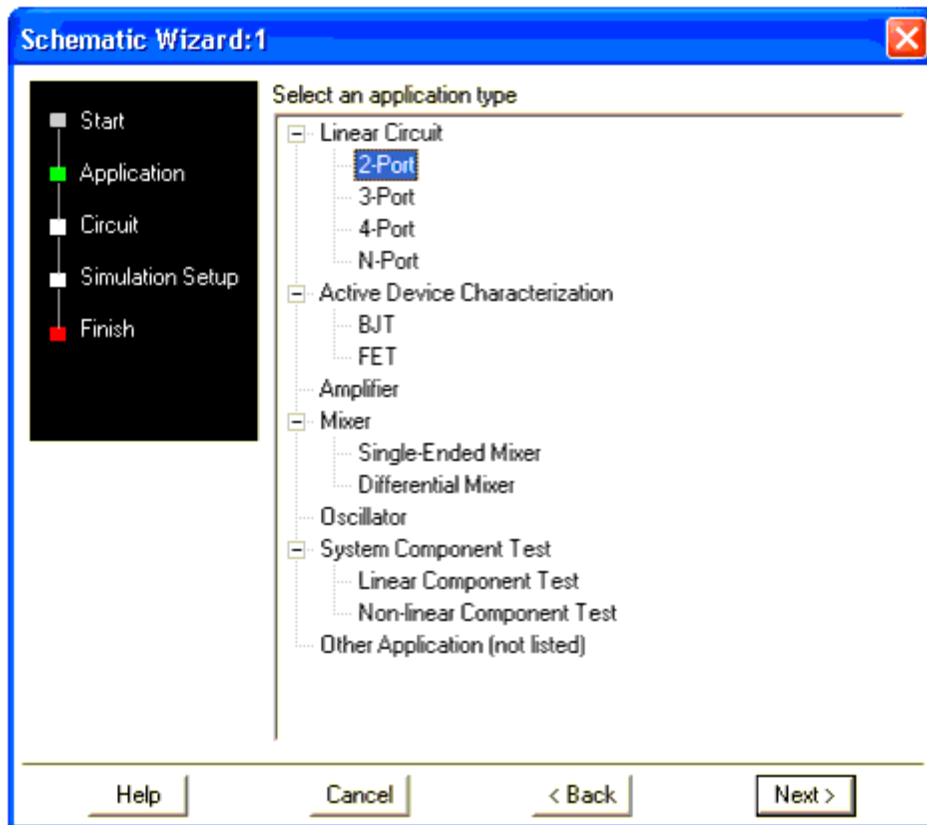
Creating a Simulation Schematic

Choosing the Simulation option from the Schematic Wizard Start page enables you to create a schematic that will simulate the behavior of a sample or user-created circuit. Creating this schematic involves choosing the desired application, specifying the test circuit, indicating the desired simulation type, and when appropriate, specifying how a circuit should be placed in the simulation schematic. The steps associated with creating a simulation schematic are:


- Application
- Circuit
- Simulation Setup
- Finish



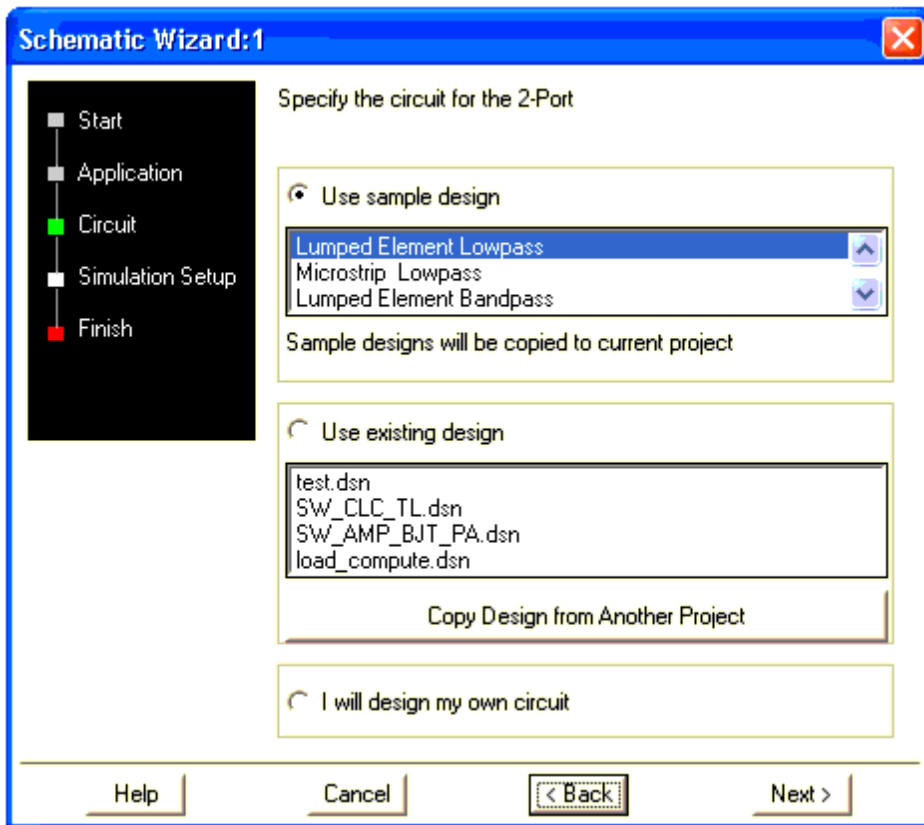
Application Selection Step



The first step in creating a simulation schematic is choosing the application type. A variety of different choices representing common applications are provided. If you find your intended application on the list, you can select it. If you do not see your application, the Schematic Wizard may still be able to provide assistance in creating your schematic. Simply choose the Other Application (not listed) option at the bottom of the tree.

-  **Hint**
The wizard will not allow you to proceed until you have made a valid selection from the list. Top-level items in the tree structure that have sub-items beneath them are not valid selections.

Circuit Selection Step



Once you have determined the application type, you are ready to specify the circuit that will be simulated within the schematic. Three options are provided relative to the test circuit:

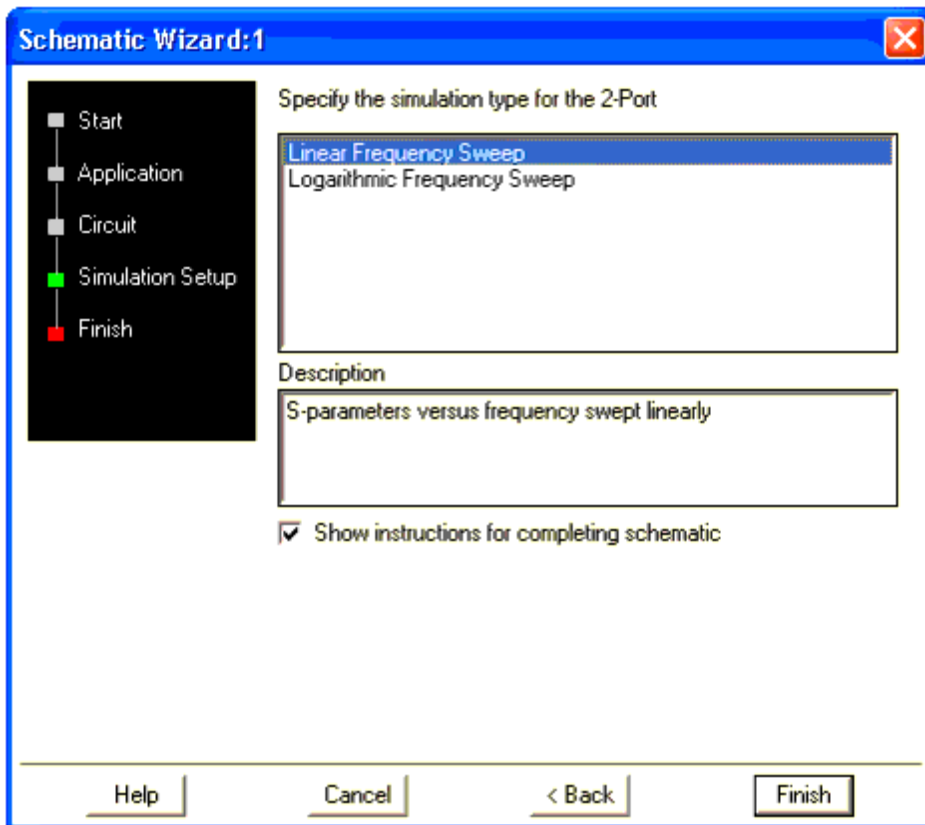
- **Use sample design** A sample circuit appropriate for the application is provided. This circuit will be copied into your project directory and connected into the simulation schematic.
- **Use existing design** Enables you to specify an existing ADS subnetwork (created, for example, using the Circuit option of the Schematic Wizard) for placement within the simulation schematic. All designs in the current project will be shown. However, if you select a design that has not been properly created for use as a subnetwork, a warning will be issued and the design will be deselected. If you have a design within another project that you would like to use, you can copy it into the current project using the Copy Design from Another Project button.
- **I will design my own circuit** No test circuit will be placed in the simulation schematic. It is assumed that you will design your own circuit and manually connect it into the schematic created by the wizard.

The availability of each option is dependent on the selection made at prior steps.



Hint

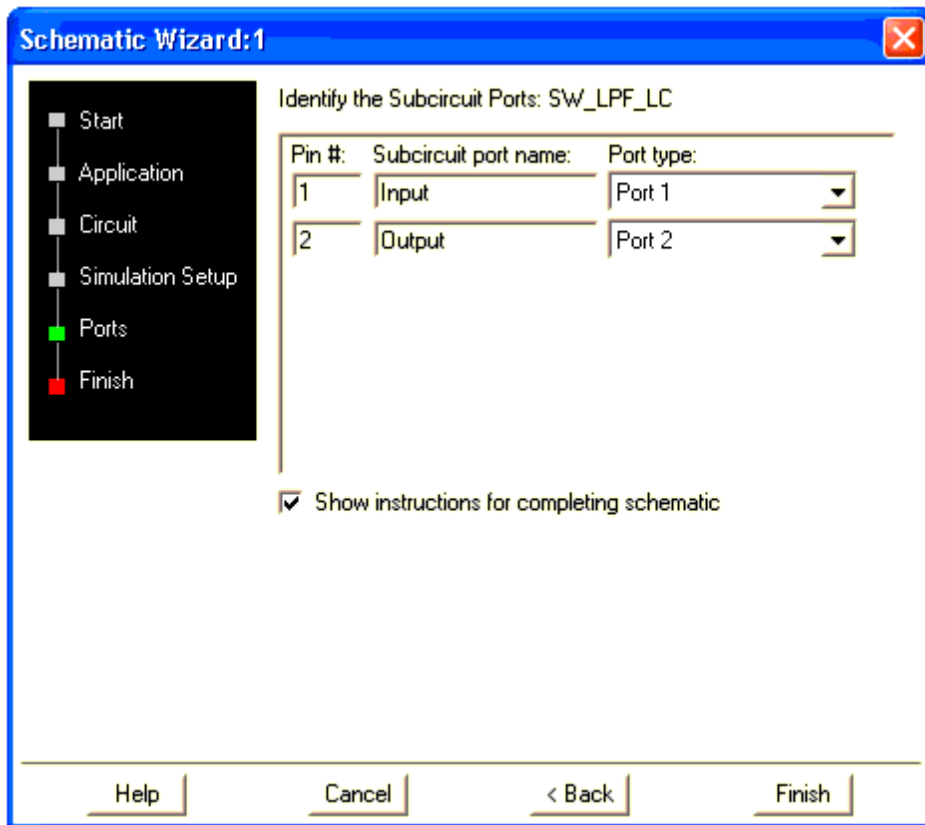
The wizard will not allow you to proceed until you have made a valid selection.



You are now prepared to specify the type of simulation that you would like to complete. Based on prior selections, a list of possible simulations is offered. If you could not find your desired application earlier (Other Application), then at this stage you are presented with a tree structure of common simulations as well as system and user-defined simulation templates. The Description area below the list of simulations helps you to choose from the different simulation options.

- ✔ **Hint**
 The wizard will not allow you to proceed until you have made a valid selection.

Port Specification Step



If you chose Use existing design in the Circuit Selection step, and you have selected a valid design to use as a test circuit within your simulation schematic, the Port Specification step is added. You must use this step to indicate what each of the pins on the component refers to within the subnetwork. Based upon application/simulation selections, you will be given a list of possible designations for each port. Using the pull-down list, specify the appropriate port type. If you do not see the port type listed, you can choose either to ground the port or leave it unconnected (open circuit termination). The circuit will be placed on the schematic at this point so that you can visually inspect it to assist in the port designation.



Hint

Using the Back button at this step will remove the placed circuit from the schematic.

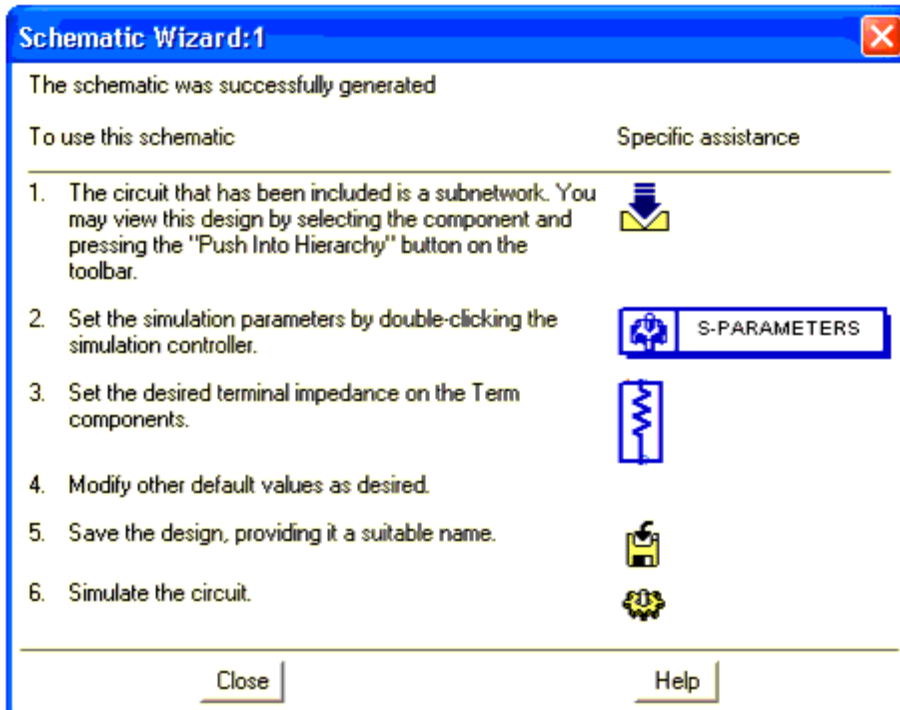
Schematic Completion Step

Successful completion of the wizard leads to a schematic that is nearly ready for simulation. If you requested that instructions be provided in the Simulation Setup step, a dialog will appear with information to assist you in performing the specific tasks associated with completing your design, simulating the circuit, and viewing the simulation results.



Important

Be sure to save the design if you want to preserve it.



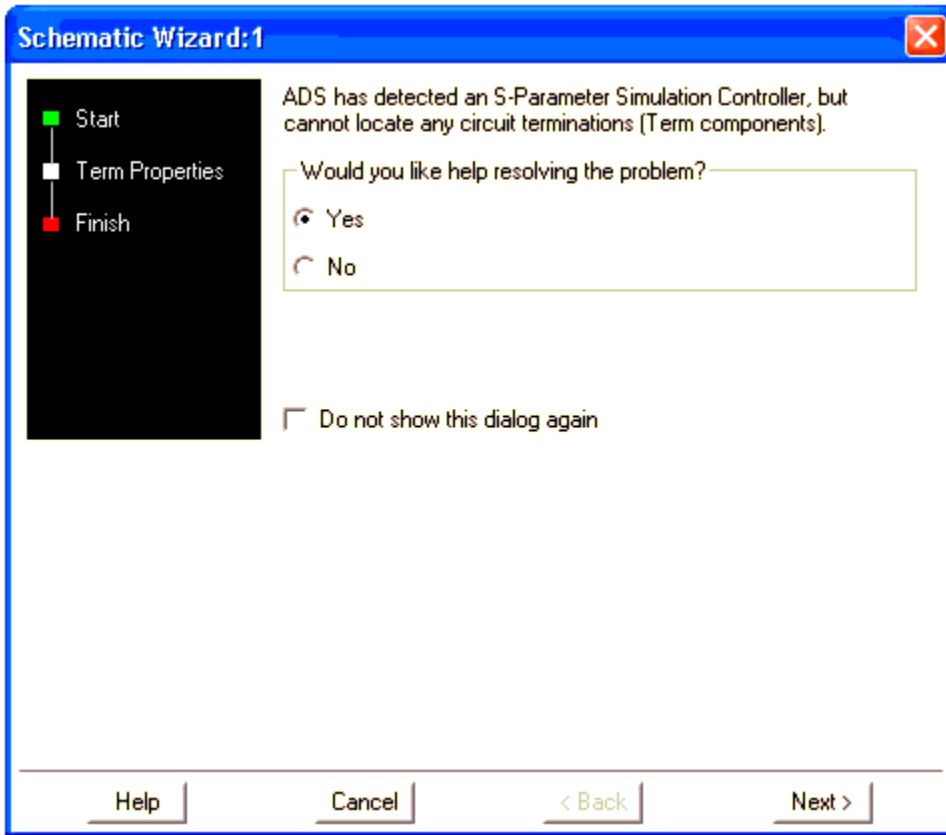
If you chose to use a simulation template (obtained using the Other Application path), you must manually connect the test circuit (if specified) into the simulation schematic. Some templates may already have a test circuit included, in which case you can either use the existing test circuit or delete it and put the specified test circuit placed by the wizard in its place. Furthermore, if you chose to create your own circuit, you must do so before meaningful results can be generated by the simulation.

Each simulation schematic is associated with a display template. Once you have completed the schematic and successfully simulated a design, a display window will appear showing the results of the simulation for your circuit.

Correcting an S-Parameter Simulation Schematic

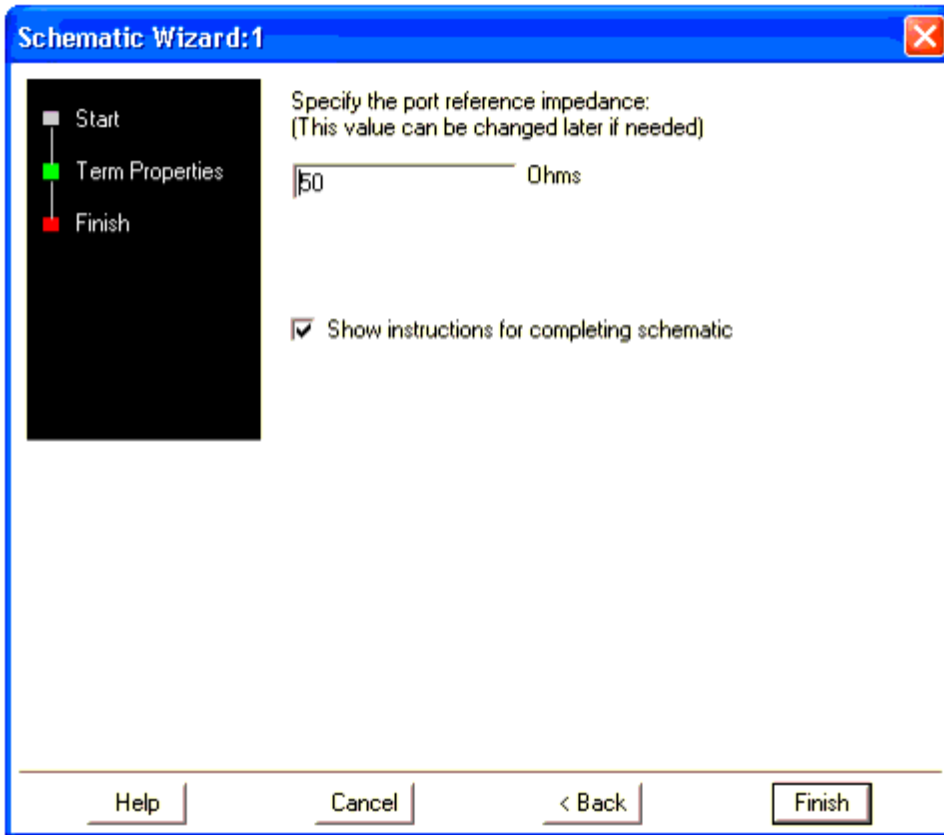
If a simulation schematic contains an S-Param simulation controller but does not include Term components, the Schematic Wizard will appear (if the preference is set). In this case, the Start page will indicate the error and give you the option of using the wizard to assist in correcting the schematic. The tasks associated with correcting an S-parameter simulation schematic are:

- Term Properties
- Finish




This page also offers the option Do not show this dialog again. Selecting this option will turn off the Schematic Wizard preference option. You can select the wizard option again in the ADS Main window: Tools > Preferences.

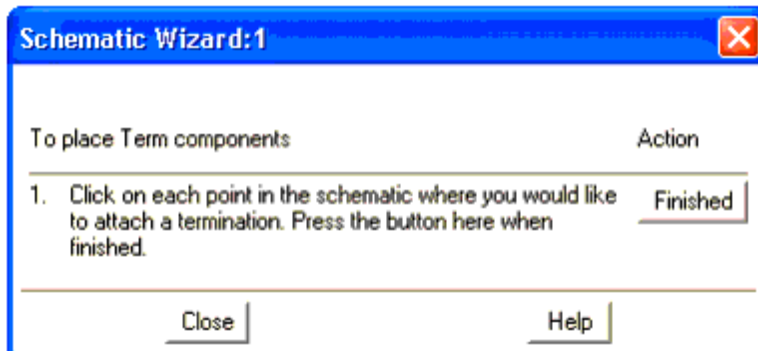
Term Properties Specification Step



The first step in correcting an S-parameter simulation schematic is specifying the reference impedance for the ports in the network. You can later change this value by editing the parameters of the Term components placed on the schematic.

 **Hint**
The wizard will not allow you to specify an invalid reference impedance.

Term Placement Step




You are now prepared to place your S-parameter network ports (Term components) on the schematic. Simply click

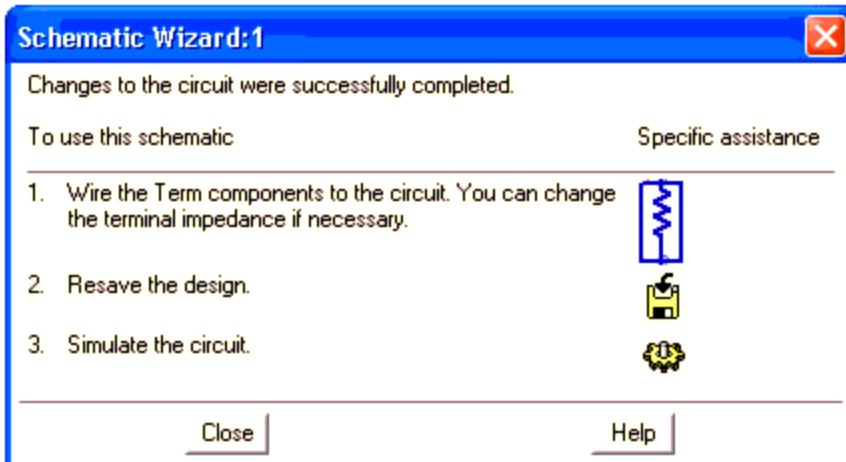
the mouse at the locations in the circuit where you would like to place a Term. The Schematic Wizard will automatically place the component for you with a ground, and the ports will be numbered in the order in which they are placed. Placing the Term components directly on a node of a circuit will result in their automatic connection to the circuit. Placing them elsewhere will require that you manually wire the Term components to the desired nodes after you have finished placing them. If the placement is not exactly what you had intended, you can manually move and rewire the Term components after completion of the placement step.

Once you have finished placing all desired terms, click Finished.

Schematic Completion Step

If you requested that instructions be provided in the Term Properties step, a dialog will appear with information to assist you in performing the specific tasks associated with completing your design and simulating the circuit. If you did not place the Term components directly on a circuit node, you will need to manually wire them to the intended nodes in the circuit.

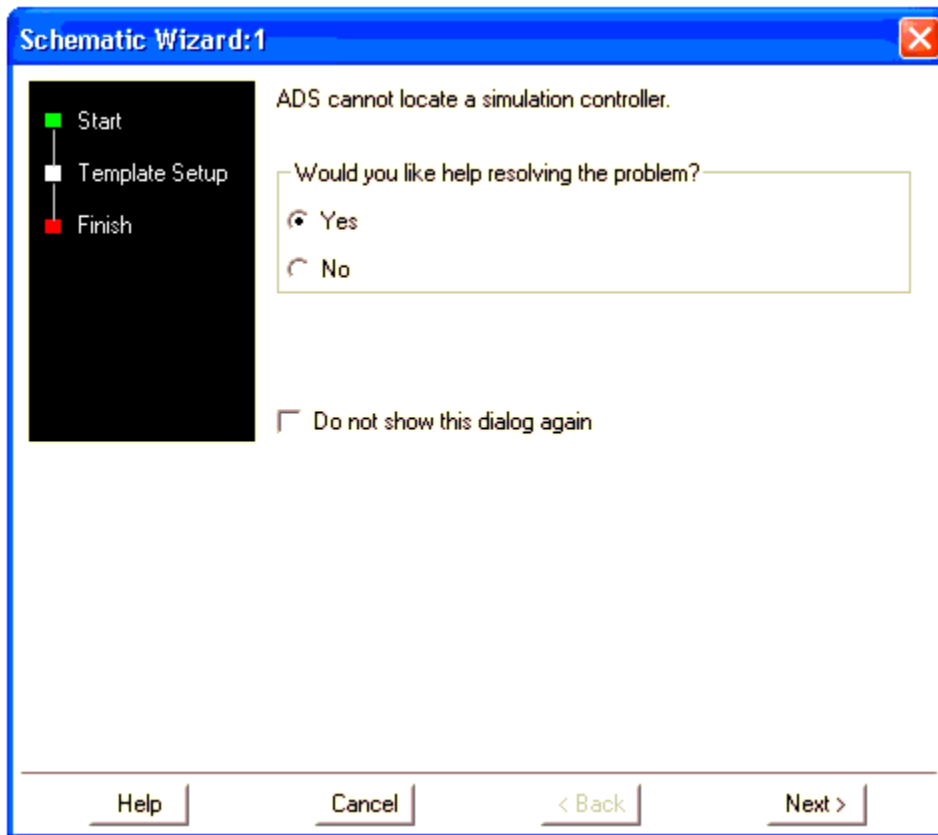
 **Important**
Be sure to save the design if you want to preserve the changes.



Correcting a Simulation Schematic with No Simulation Controller

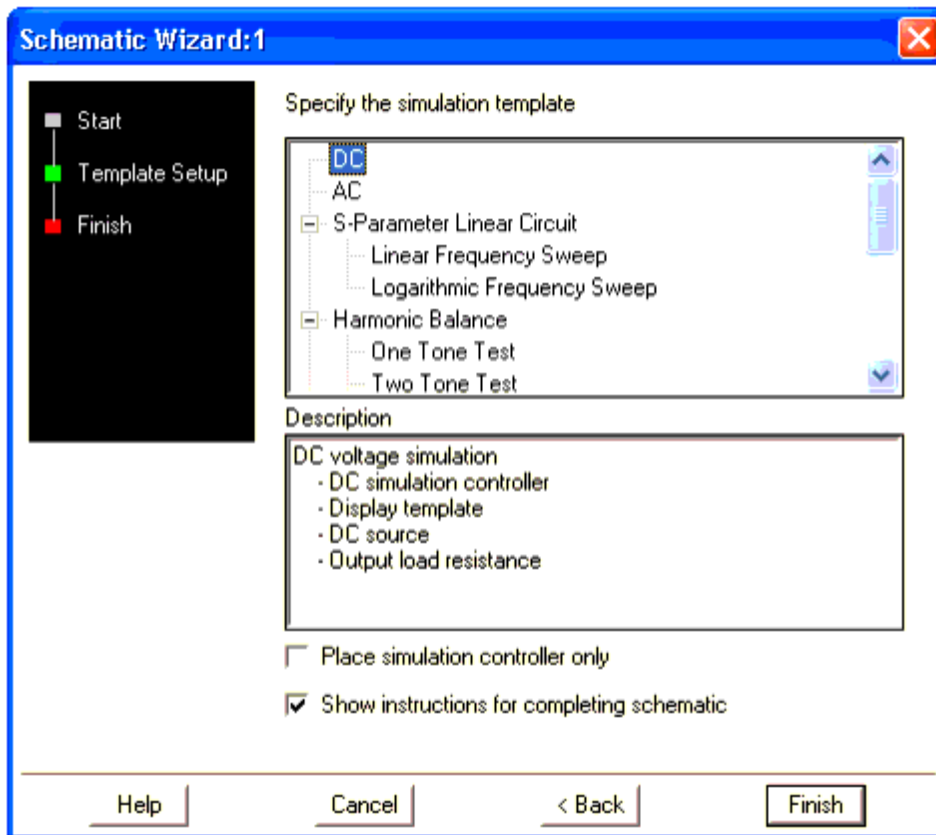
If a simulation schematic does not contain a simulation controller, the Schematic Wizard will appear (if the preference is set). In this case, the Start page will indicate the error and give you the option of using the wizard to assist in correcting the schematic. The tasks associated with correcting the simulation schematic are:

- Template Setup
- Finish




This page also offers the option Do not show this dialog again. Selecting this option will turn off the Schematic Wizard preference. You can select the wizard option again in the ADS Main window: Tools > Preferences.

Template Setup Step




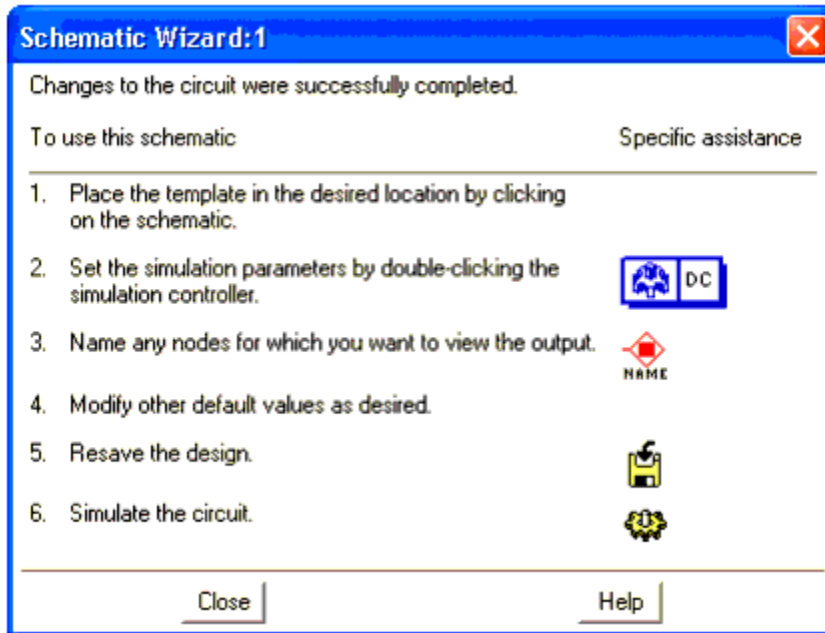
You can now specify the type of simulation that you would like to complete. You are presented with a tree structure of common simulations as well as system and user-defined simulation templates for the design type (Analog/RF or DSP). The Description area below the list of simulations helps you to choose from the different simulation options. If you do not wish to place an entire simulation template, you can choose to place only a simulation controller by selecting the Place simulation controller only option.

 **Hint**
 The wizard will not allow you to proceed until you have made a valid selection from the list. Top-level items in the tree structure that have sub-items beneath them are not valid selections.

Schematic Completion Step

If you requested that instructions be provided in the Template Setup step, a dialog will appear with information to assist you in performing the specific tasks associated with completing your design and simulating the circuit. For most templates, this will include placing the requested simulation template in the desired location on your schematic. If, however, you requested a full S-parameter simulation template, the main elements of the template will be placed at the top of your schematic and you will be given the opportunity to place the Term components on the schematic. The instructions will not appear until you have clicked Finished on the dialog that appears.

 **Important**
 Be sure to save the design if you want to preserve the changes.



Using the Smart Simulation Wizard

The Smart Simulation Wizard is provided to assist new ADS users, as well as those who use it infrequently, in setting up simulations for typical microwave/RF circuits. The wizard will guide you through the process of:

- Selecting an application-specific design (or your own design)
- Selecting predefined simulation setups
- Specifying simulation settings (frequency, bias, etc.)

The wizard then configures the sources and simulation controls and begins the simulation(s). When multiple simulations-requiring different configurations-are requested, the wizard automatically reconfigures the subnetwork for the appropriate sources, terminations, and simulation controls. When the simulation is finished, simply click to display the results. Note that although basic simulation setups are provided with the various simulator licenses, additional simulation setups require specific DesignGuide licenses. These differences are identified in the wizard.

To invoke the Smart Simulation Wizard :

From the Schematic window in the project of interest, choose Simulate > Smart Simulation Wizard.



Step 1 prompts you to select one of several different application types.

Advanced Design System 2008

Device Characterization	BJT Characterization FET Characterization MOSFET Characterization
Amplifier	Amplifier
Mixer	Single-Ended Mixer Differential Mixer
Linear Circuit	Linear 2-port Linear 4-port

Step 2 prompts you to select one of the following design types:

- A sample design provided by the Smart Simulation Wizard
- An existing ADS subnetwork design
- A new subnetwork design


Step 3 varies based on the choice made in Step 2. You are prompted to select an existing design, enter a name for a new design, or select one of the following application-specific designs.

Device Characterization		
	BJT Characterization	
		NPN BJT
		PNP BJT
	FET Characterization	
		GaAs MESFET Statz Model
		EEFET Model
		GaAs MESFET Model
		HEMT Model
		JFET Model
	MOSFET Characterization	
		NMOSFET Model
PMOSFET Model		
Amplifier		
	Amplifier	
		MOSFET Power Amplifier
		BJT Power Amplifier
	Behavioral Model Amplifier	

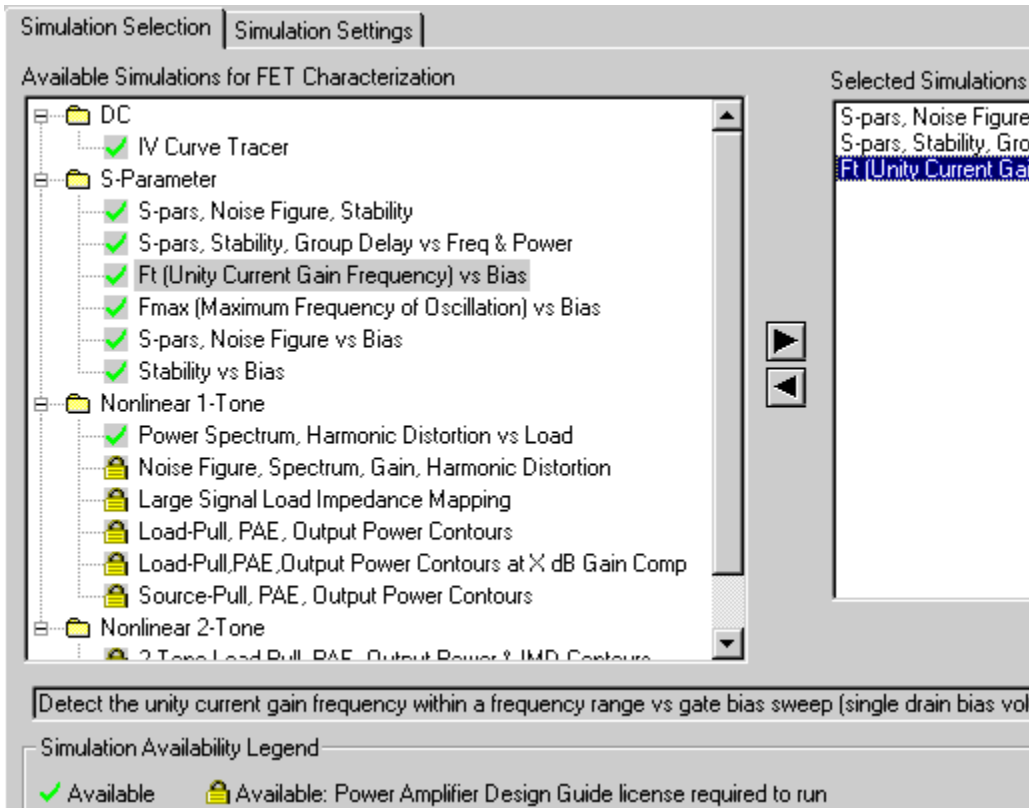
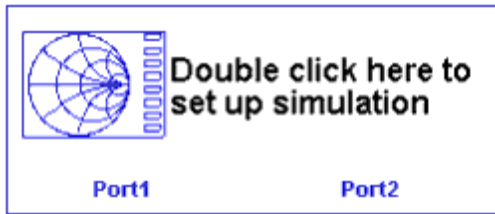
Mixer		
	Single-Ended Mixer	
		MESFET Gilbert Cell Mixer
		FET Mixer
		BJT Gilbert Cell Mixer
		Behavioral Model Mixer
	Differential Mixer	
		MOSFET Gilbert Cell Mixer
		FET Mixer
Linear Circuit		
	Linear 2-port	
		Simple Lowpass Filter
		Microstrip Bandpass Filter
		S-Parameter Data File
		Linear FET
	Linear 4-port	
		Linear FET Modeling

Step 4/Step 5 varies based on your previous choices. For an existing ADS design, you are prompted to identify the port type for each port in your design (input, output, base, collector, etc.). For all design types, the wizard then describes how to view the network associated with the schematic symbol and how to access the simulation setup portion of the wizard.



When you click Finish, the top-level design appears, and you will see that it consists of two main parts: a schematic symbol representing the subnetwork to be simulated and a simulation setup symbol.

 **Note**
 If working with a sample design, the top-level and subnetwork designs, as well as the related data displays, are copied to the current project. If you select an existing design from a different project (via an Included project), that design is copied to the current project.

- Schematic symbol -A schematic symbol representing the subnetwork to be simulated, is visually connected to the simulation setup symbol. Push into the symbol to view, edit, or create the subnetwork. When you push into most of the schematic symbols, you will notice that the subnetwork designs contain cautions against deleting or renumbering of ports.
- Simulation setup symbol -The simulation setup symbol is similar to the one shown next. Double-click (or right-click and select the first choice from the pop-up menu) to specify the simulation setup details.




Each simulation is marked with one of two icons, as shown next.

-  Only the appropriate simulator license is required for this simulation
-  A specific DesignGuide license is required, in addition to the appropriate simulator license, for this simulation

You can highlight any selected simulation (from the list box on the right) and click Show Schematic to view the design containing the simulation setup.

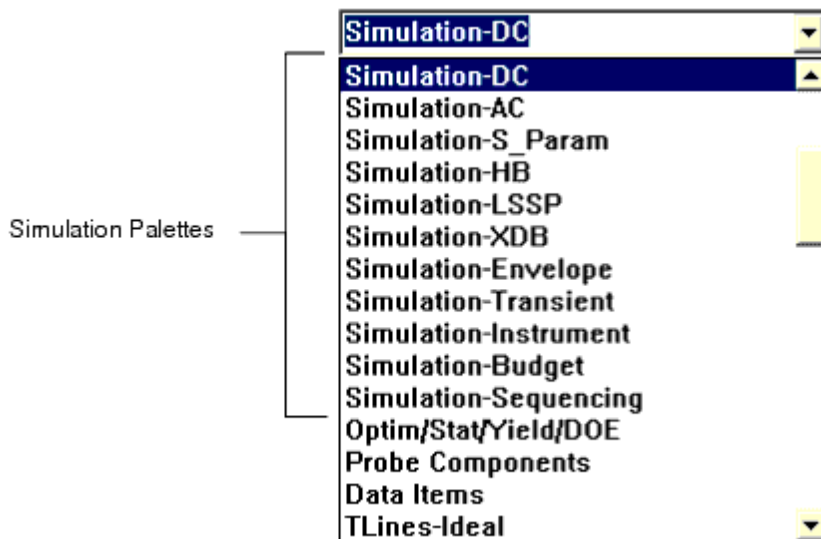
From the Simulation Settings tab you can specify the desired settings for the simulation parameters such as frequency, power, bias, etc. When you have selected all the desired simulations and specified the desired settings, click Simulate to proceed. The progress window appears and is dynamically updated to indicate which simulations have completed and which remain. When all simulations are complete click Display Results to view the data displays.

Note that the results for each simulation are displayed on separate pages, which can be accessed individually from the Page menu.

 **Hint**
After simulating a given design once, you can display the results from the previous simulation via the pop-up menu. Position the pointer over the simulation setup symbol, click right, and select Display Data from Last Simulation.

Selecting Simulation Controllers

Simulation controllers are grouped in a number of simulation palettes accessed from the Component Palette List.



Each palette contains the specific simulation controller, plus:

- The Options component
- Components for defining sweep plans and parameter sweeps
- Node set components
- Measurements
- Frequently-used components, such as ports and sources

To use a controller, select it from the palette, position the pointer in the drawing areas of the Schematic window and click to place it.

Using the Simulator Options Component


Advanced Design System 2008

This section discusses the details about the Options component in ADS. The Options component includes general simulation options such as convergence tolerances, warnings, and global noise temperature. An Options component can be used with any ADS simulation, and it is available from every simulation palette. The options cover the following areas:

Tab Name	Description	For details, see...
Misc	Miscellaneous options for simulation and model temperatures, topology checking, and linear and nonlinear devices.	Setting Miscellaneous Simulation Options
Convergence	Options related to voltage and current convergence tolerances.	Setting Convergence Options
Output	Sets warnings, and the saving of branch currents and node voltages.	Setting Output Options
DC Solutions	Saves DC solution to a file to re-use as an initial guess in further simulations.	Setting DC Solution Options
Display	Controls the visibility of simulation parameters on the Schematic.	Displaying Simulation Parameters on the Schematic

Setting Miscellaneous Simulation Options

Use the Misc options described in the following table to control simulation and model temperatures, topology checking, and set options for linear and nonlinear devices. In the table, names used in netlists and ADS schematics appear under Parameter Name.

 **Note**
 Simulator options are commonly used in nonlinear noise analyses. The IEEE standard temperature (T0) for noise figure measurement is 290 K (16.85 degrees Celsius). This can be set by editing Simulation temperature to that value (on Misc tab).

Miscellaneous Simulator Options

Setup Dialog Name	Parameter Name	Description
Temperature		
	Simulation temperature	Temp
		Sets the ambient temperature at which a simulation will be run. The default is 25 degrees Celsius. The predefined

Advanced Design System 2008

			variable temp is set to this value.
	Model temperature	Tnom	Sets the default value for the nominal temperature of models. The default is 25 degrees Celsius. The predefined variable tnom is set to this value.
Topology Checker			Sets topology checker mode and warning message formatting.
	Perform topology check and correction	TopologyCheck	Performs a topology check and corrects common topological problems before a simulation is run. Enabled by default. A summary of topological problems is reported in the Simulation/Synthesis Messages window. It is recommended that you perform topology checks for better simulation performance. †
	Format topology check warning messages	TopologyCheckMessages	Sets the mode for listing topology check messages to Summary or Verbose. By default a summary of the topological problems found is printed to the Simulation/Synthesis Messages window if TopologyCheck=yes. To see a list of all the nodes that have topological problems, set TopologyCheckMessages to Verbose. †
Linear Devices			
	Use S-parameters when possible	ForceS_Params	Causes the simulator to attempt an S-parameter simulation on linear devices.
Nonlinear Devices			

	P-N parallel conductance	Gmin	Specifies the minimum conductance added in parallel to the p-n junctions in the nonlinear devices. The default is 1e-12 siemens. Some of the models have the Gmin parameter. If it is specified in the nonlinear model, it takes precedence over the one in the options.
	Explosion current	Imax	Specifies the p-n junction explosion current used in the nonlinear devices. When p-n junction current exceeding this value, the junction is linearized. The Imax value specified in the device model parameter takes precedence over the one in the options. If Imax is not specified in the model parameter, the Imax given in the options will be used. If Imax is not specified in the options, the default Imax value from each nonlinear model will be used.
	Explosion current	Imelt	Specifies the p-n junction excessive explosion current used in the nonlinear devices.
	Mosfet BSIM3, 4 diode limiting current	Ijth	Similar to Imax, except that it is called Ijth in BSIM3 and Ijthd fwd, Ijthdrev Ijths fwd, Ijthsrev in BSIM4.

† For more information about topology checking see [DC Simulation](#).

Setting Convergence Options


The simulators work using an iterative method to solve the nonlinear equations. Given an initial guess x_0 , it


Advanced Design System 2008

computes a new guess x_{j+1} . From that, it computes x_{j+2} . This continues until convergence is reached. When x_j is close to x_{j-1} , it is considered converged, and the solution stops changing. Convergence is defined as follows:

```
if ( $x_j - x_{j-1} < \text{reltol} * x_j + \text{abstol}$ ) then
  converged
else
  keeps iterating
```

If the difference between the two iterations is less than the relative tolerance times the solution plus an absolute tolerance, the convergence is effective.

 **Note**
Advanced simulation parameters are accessible with this group. However, as a result of the improvements made to the DC simulation algorithm, it is extremely unlikely that the default values need to be modified. You are strongly encouraged to leave the advanced parameters set to their default values. If you encounter a circuit for which a DC analysis does not converge using the default values, or you find it necessary to change the value of any of these parameters, please contact Agilent EEs of Technical Support. See [Setting Advanced DC Convergence Options](#) for details about these parameters.

 **Caution**
Simulator parameters saved in design files in previous releases are supported in later releases. The advanced simulation parameters saved prior to and opened in ADS 2005A are recognized and populated in the simulation setup dialog box. However, due to the improvement in robustness and speed of the default DC simulation algorithm the user-defined values are disabled, and factory-defined default values are used. Changing these default values is not recommended. However, if you find it necessary to restore the original user-defined values, you must manually enable Advanced Settings to restore them.

Use the Convergence options described in the following table to select voltage and current convergence criteria (tolerances) which apply to all analysis types. In the table, names used in netlists and ADS schematics appear under Parameter Name.

Simulator Convergence Options

Setup Dialog Name	Parameter Name
Convergence Check - There are three tolerance presets to provide options for beginning users. For comparison of tolerance preset values, see the following table.	

	Relaxed	
	Intermediate	
	Strict	
	Custom	
	Analysis Defaults	
<p>Tolerances-these apply to all simulation types. For details about these parameters, see Current Relative Tolerance, Current Absolute Tolerance and Voltage Relative Tolerance, Voltage Absolute Tolerance.</p>		
	Voltage relative tolerance	V_RelTol
	Current relative tolerance	I_RelTol
	Voltage absolute tolerance	V_AbsTol
	Current absolute tolerance	I_AbsTol
	Frequency relative tolerance	FreqRelTol
	Frequency absolute tolerance	FreqAbsTol
Advanced...		

Default Preset Tolerance Values

	Relaxed	Intermediate	Strict (default)
V_RelTol	10^{-3}	3×10^{-5}	10^{-6}
I_RelTol	10^{-3}	3×10^{-5}	10^{-6}
V_AbsTol	10^{-4}	10^{-5}	10^{-6}
I_AbsTol	10^{-8}	10^{-10}	10^{-12}

Note
 If simulator options are not set, transient analysis uses a default value of 10^{-3} for V_RelTol, and I_RelTol, while all other analysis types use the default value of 10^{-6} . If simulator options are set, they apply to all analysis types.

Current Relative Tolerance, Current Absolute Tolerance

These tolerances are used to satisfy Kirchhoff's Current Law (KCL) in solving for the currents at each node in the circuit. The simulator attempts to find a solution that satisfies KCL, so that the sum of the currents entering (or

leaving) all circuit nodes is zero. At each iteration, it uses Current relative tolerance and Current absolute tolerance as a tolerance for the node currents. For convergence to be achieved, the currents must satisfy the following at each circuit node:

$$\sum |i_n| \leq \epsilon_{rel} \sum |i_n| + \epsilon_{abs}$$

where

i_n = Current in each branch connected to the node

ϵ_{rel} = Current relative tolerance

ϵ_{abs} = Current absolute tolerance

The default value for Current relative tolerance is 10^{-6} (0.0001 percent), and the default value for Current absolute tolerance is 10^{-12} (1 pA). For many problems, these tolerances are much tighter than they need to be. (The default value of Current relative tolerance in Berkeley SPICE 3e1 is 10^{-3} .) Relaxing these tolerances not only allows problem circuits to be solved, but it also allows them to be solved in less time.

Voltage Relative Tolerance, Voltage Absolute Tolerance

Once Kirchhoff's law is satisfied for all nodes, the simulator checks for unique solutions by calculating all node voltages. Sometimes, large changes in node voltages cause very little change in node currents. For instance, if two S-parameter blocks (that is, any 2-port, such as an amplifier or filter, for which there are measured S-parameters) are cascaded, and the reference node between the two components is not grounded, then the differential voltage between the two S-parameter blocks can have any value at all without changing the currents. The circuit then has multiple possible solutions. To find the correct solution for all node voltages, the simulator will use the Voltage relative tolerance and Voltage absolute tolerance parameters in a manner similar to the way it uses Current absolute tolerance and Current relative tolerance. For convergence, the following relationship must be satisfied for every node voltage in the circuit:

$$|\Delta V_n| \leq \epsilon_{rel} |V_n| + \epsilon_{abs}$$

where

ΔV_n = Change in the node voltage solution from the previous iteration

V_n = Node voltage found in this iteration of the solution

ϵ_{rel} = Voltage relative tolerance

ϵ_{abs} = Voltage absolute tolerance

The default value for both Voltage relative tolerance and Voltage absolute tolerance is 10^{-6} . Like Current absolute tolerance and Current relative tolerance, these tolerances can be loosened to help with simulation convergence and speed.

Setting Advanced DC Convergence Options

The stand-alone DC simulator's sole role is to do a DC analysis. All other simulators such as AC, S-parameter, transient, harmonic balance, and circuit envelope do an initial DC analysis as their first step. The Advanced DC Convergence options are used to control the initial DC analysis done by these simulators. For information about setting up stand-alone DC simulations, see [DC Simulation](#).

The robustness and speed of the default DC analysis algorithm has been significantly improved in ADS 2005A. All DC analyses with factory-default settings are expected to converge to the correct solution with near-optimal speed. This means that it is extremely unlikely that either of the following advanced simulation parameters must be altered:

DC_ConvMode
MaxDeltaV

Use the options in the following table to select Advanced DC convergence options. In the table, names used in netlists and ADS schematics appear under Parameter Name.


Advanced DC Convergence Options

Setup Dialog Name		Parameter Name
Advanced DC Convergence Settings		
	Max. Delta voltage	MaxDeltaV
	Mode	DC_ConvMode
	Auto sequence	0
	Newton-Raphson	3
	Forward source-level sweep	4
	Rshunt sweep	5
	Reverse source-level sweep	6
	Hybrid solver	7
	Pseudo transient	8

† For more information about setting MaxDeltaV and DC_ConvMode for DC simulations, see [DC Simulation](#).

Setting Output Options


Use the Output options described in the following table to select warnings options, as well as to determine whether branch currents and node voltages will be saved. In the table, names used in netlists and ADS schematics appear under Parameter Name.

 **Note**
 These Output options available in the Options component are not the same as the Output parameters used in other simulation setup dialog boxes, such as HB, AC, etc., which are described in the section [Selectively Saving and Controlling Simulation Data](#).

Simulator Output Options

Setup Dialog Name	Parameter Name	Description
<p>Warnings - If threshold limits are specified, the simulator will display the warning(s), in the Simulation/Synthesis Messages window, the first time they are exceeded during a dc, harmonic balance or transient simulation. For appropriate components, you may open the component dialog box to edit the component, then specify threshold values. Most of the parameter names will begin with "w" for warning, and some (but not all) will also include "max" in the name.</p>		

	Issue warnings	GiveAllWarnings	Causes warning messages to be reported.
	Maximum number of warnings	MaxWarnings	Sets the number of warnings desired.
	Ignore shorts	IgnoreShorts	Allows the simulation to proceed in the presence of shorts.
Output filters			
	Save branch currents	SaveBranchCurrents	Creates a record of branch currents found by a simulation.
	Save internal node voltages	OutputInternalNodes	Creates a record of internal node voltages found by a simulation.

 **Note**
 A resistor has threshold parameters for wP_{max} and wI_{max} , for maximum power and current dissipation, respectively (all such settings begin with "w," which signifies a warning will be issued in the Simulation/Synthesis Messages window). Some components also check voltages. A BJT has eight threshold settings. All diodes, transistors, FETs, resistors, capacitors, current probes, and shorts contain threshold parameters.

Setting DC Solution Options

You can save the complete DC solution to a file and then re-use it as an initial guess in further simulations. For large circuits or those with time-consuming DC simulations, this can save a significant amount of CPU time by avoiding the needless repetition of the same or similar simulations each time. This applies to any simulation that either performs or relies on a DC solution, which includes all simulations with nonlinear elements.

For example, once a DC solution is obtained by running an AC simulation, future AC simulations at different frequencies or linear noise simulations do not have to re-simulate to get the same DC solution again. If the circuit is changed, either via a parameter change or even a topology change that will change the DC solution, this saved DC solution can still be used as an initial guess for the new DC solution. If the circuit change was not too extensive, then having a reasonable initial guess usually will still reduce the total re-simulation time. If the circuit change is so extensive that the simulation cannot converge using the supplied initial guess, then the simulator will proceed with its normal DC simulation algorithm. In this case, it would save CPU time to disable the Use Initial Guess.

If the circuit topology has changed between the time the solution file was created and when it is used as an initial guess, the simulator will still attempt to use as much of the data as possible. It will also output various messages, if desired, noting what has changed between the two versions. While this feature does not check for parameter changes,

Advanced Design System 2008

it can be a useful tool for comparing the topology of two circuits, or to identify what has changed since the solution was last saved. Items checked include the total number of equations (nodes and branches), the total number of instances and their names, and most connectivity changes.

For information on initial guess and final solution options available in the Harmonic Balance simulation controller, see [Setting Up the Initial Guess](#).

Use the DC Solutions options in in the following table to select options for saving DC solutions. In the table, names used in netlists and ADS schematics appear under Parameter Name.

Simulator DC Solutions Options

Setup Dialog Name	Parameter Name	Description	
Initial Guess			
	Use initial guess	DC_ReadInitialGuess	Instructs the simulator to read the input file and use it as an initial guess for any DC solve. If a file name is not supplied (DC_InitialGuessFile), a file name is internally generated using the design name, followed by a .dcs suffix. If a file name is supplied, the suffix is neither appended nor required.
	File	DC_InitialGuessFile	File name for initial guess file.
	Annotate	InitialGuessAnnotation	Enables you to select a detailed record (2), a summary (1), or none (0).
Final Solution			
	Write final solution	DC_WriteFinalSolution	Instructs the simulator to write the final DC solution to the output file. If a file name is not supplied (DC_FinalSolutionFile), a file name is internally generated using the design name, followed by a .dcs suffix. If a file name is

			supplied, the suffix is neither appended nor required. If this box is checked, then the last DC solution is output to the specified file. If this is the same file as that used for the initial guess, this file is updated with the latest solution. If a swept analysis is being performed that changes the DC solution, you will either want to not write a final solution or use two different file names for the initial guess file and the final solution file.
	File	DC_FinalSolutionFile	File name for final solution file.

Sweeping Parameters

Most simulations are performed over a range of values instead of just a single point. You can sweep over time or frequency (depending upon the type of simulation) or you can elect to sweep over another parameter. For ADS, you can sweep one or more parameters using the following methods:

- You can set the sweep range of time, frequency, or a single other parameter (under the Sweep tab for a given simulator controller).
- You can use the ParamSweep and SweepPlan components for sweeping more than one parameter, or sweeping over more than one range of values. These components appear on all simulation palettes. For details about using or using these components, see [Parameter Sweeps and Sweep Plans](#).

If using the Load Sharing Facility (LSF) utility, you can break up a sweep and run the simulation on multiple machines, in parallel, by selecting Parallel Hosts as the Simulation Mode (Simulate > Simulation Setup). Individual sweep points are run on each machine and the results are combined into a single dataset on the local machine. For details on setting up remote and local machines for remote processing, see "Using Remote Simulation" in the installation documentation for your platform:

- [Windows Installation](#)
- [UNIX and Linux Installation](#)

Optimizing a Design

You can set up nominal optimizations or statistical yields as part of a simulation. These features require a separate license. For complete information, see the [Tuning, Optimization, and Statistical Design](#) documentation.

Working with Expressions

You can add variables, functions, and conditional statements to a schematic, making your designs more flexible and versatile.

You can use these items:

- In component parameter definitions. From the component parameter editing dialog box, select the parameter and, if available, click Equation Editor. You can write an expression that defines this parameter.
- In variables. Variables can be added to a schematic using the VAR (VarEqn) component, which can be found in the Data Items palette. Once a variable is defined, it can be used in expressions within the design.

You can also add measurements to a schematic. Measurements are predefined functions that process data so that it can be presented in the Data Display. There are numerous predefined measurements under the simulation palettes, but you can also create your own using the MeasEqn component.


For more information on how to use measurements, see [Measurement Expressions](#).

Expressions Examples

Many of the projects in the Examples directory use variables and measurements. One example that includes many variable definitions plus conditional statements is NADC_PA_Test.dsn in RF_Board/NADC_PA_prj.

Saving and Controlling Simulation Data


The results of the simulation are stored in a dataset. This dataset is then used by the Data Display for viewing results. You can select the name and location of the dataset you want to use for a simulation. The default name of the dataset is the name of the design.

 **Note**
If you accept the default dataset name and perform multiple simulations, the dataset will be overwritten each time. To collect separate datasets for each simulation, specify a unique name (for the dataset you are about to


create) prior to each simulation.

To specify a dataset name prior to simulating:

1. In the Schematic window, choose Simulate > Simulation Setup.
2. In the Dataset field, enter the name of the dataset where you want simulation data to be saved. Click Browse to view existing dataset names in the current project. Click Apply or, if you are ready to run the simulation, click Simulate.

 **Hint**
Datasets are stored in the /data subdirectory of a project. This means that you should provide unique names for the datasets that will be generated from the different designs in the same project directory.

3. Supply a name in the Data Display field, or accept the default name.
Data Display --The name you specify here will be the title of the Data Display window that is opened, and the default filename should you choose Save As (from the Data Display window). The default name shown is based on the current design name.
Open Data Display when simulation completes --If enabled, a Data Display window will open automatically when the simulation is complete. For details see [Automatically Displaying Simulation Data](#).
4. Select the desired Simulation Mode:
 - To simulate on a single machine, select Single Host. Select the Single tab and select the desired Simulation Host Type, Local or Remote. If you select Remote, you can then select Specify to choose a specific server by name, or select Find Fastest and let the Load Sharing Facility (LSF) choose the most suitable remote host.
 - To break up your sweep and simulate individual parts simultaneously on remote machines, select Parallel Hosts. In the Parallel tab, select the desired Sweep Variable from the drop-down list and set the desired Start, Stop and Step values. Individual sweep points are run on each machine and the results are combined into a single dataset on the local machine.
 - To break up a signal processing BER simulation over multiple hosts, select Parallel Hosts. In the Parallel tab, check the Parallel BER check box, and specify the number of hosts to be used in the Number of Partitions field. This feature is only available when BER simulation is using one of the berMC, berMC4, or BER_FER components. For more information, see the documentation for these components in Signal Processing Components documentation.

 **Note**
Taking advantage of the LSF utility requires the installation and configuration of that software on the necessary files/machines. For details on setting up your remote and local machines for remote processing, see "Using Remote Simulation" in the installation documentation for your platform:

- [Windows Installation](#)
- [UNIX and Linux Installation](#)

Automatically Displaying Simulation Data

When setting up your simulation (Simulate > Simulation Setup, in the Schematic window), you can enable an automatic display of your results.

Advanced Design System 2008

Select the Open Data Display when simulation completes option to force a Data Display window to open automatically when the simulation is complete. The data display that appears in that window depends on the simulation setup and the status of display templates:

- If you specify the name of an existing display to open, that display is opened.
- If you specify a new name, then a blank Data Display window opens.
- If one or more data display templates are associated with the current design, then the Data Display window opens and inserts each template on its own page. A data display template can be associated with a design in one of two ways:
 - By default, if your design includes a supplied schematic template, and a data display template is associated with that schematic template.
 - Explicitly, if you create your own data display template (File > Save As Template in the Data Display window) and associate that template with your design via the DisplayTemplate component. (See the next section, [Using a DisplayTemplate Component](#))
- If none of the above criteria are met, but you select the option, then a blank Data Display window opens.


Using a DisplayTemplate Component

The DisplayTemplate component (available from most simulation libraries) enables you to associate one or more data display templates with a given design. (If you include one of the supplied schematic templates in your design, it most likely has a data display template associated with it.)

The starting point of this procedure assumes you have already created a data display file for use as a template, by setting up the Data Display window as desired and choosing File > Save As Template.

To associate a data display template with the current design:

1. Place a DisplayTemplate component in the Schematic window.
2. Select String and Reference as the Parameter Entry Mode.
3. Enter the name of the template (the filename you supplied in the Data Display window) and click Add.

 **Hint**
You can specify multiple templates for the same data display and subsequently access them from the Page menu).

4. When you are through specifying display templates for the current design, click OK.

Manually Displaying Simulation Data

If you do not want the Data Display window to open automatically, disable the option Open Data Display when simulation completes in the Simulation Setup dialog box.

To open a new window for displaying and manipulating data:

Choose Window > New Data Display from the Main, Schematic, or Layout windows.

To save a graph for later viewing/manipulating:

Choose File > Save.

To save a graph for use as a template:

Choose File > Save As Template.


To open a previously saved data display:

1. Choose Window > Open Data Display from the Main, Schematic, or Layout windows. In the dialog box that appears, the path is automatically set to the current project directory and the filter displays all saved graphs (*.dds).
2. Double-click the graph you want to open or select it and click OK.

For details on working with simulation data, see [Data Display](#).

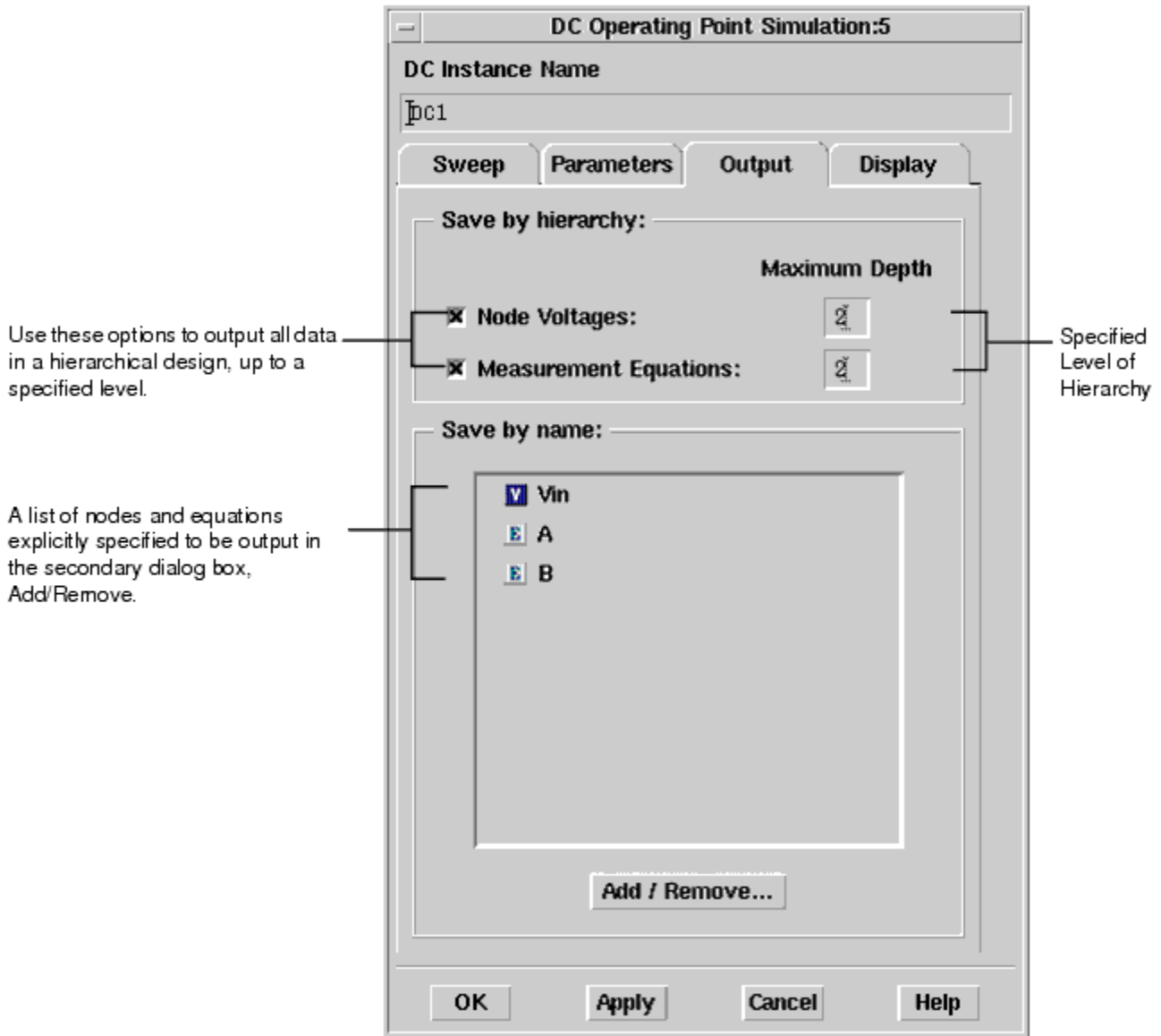
Selectively Saving and Controlling Simulation Data

The Output tab in all A/RF analysis components can be used to create an Output Plan, which controls the data that will be saved to the dataset. You can control output generated from named nodes, buses, measurement and VAR equations. By default, all named nodes up to two levels below the top level are saved. The data from all measurement equations and components with the parameter SaveCurrent set to yes is also saved. You can also control the saving of data using the hierarchy level for nodes and measurement equations or by explicitly specifying the output quantity.

 **Note**
The Output tab described in the section is used in the simulation dialog boxes, such as HB, AC, etc. It is not the same as the Output tab used in the Options component, which is described in the section [Setting Output Options](#).

To modify the default behavior of sending data to the dataset:

1. Edit the analysis controller item, and select the Output tab.



- To output all named nodes/measurement equations in the top-level design only, select the Node Voltages and/or Measurement Equations options and use zero as the Maximum Depth.
- To output all named nodes/measurement equations in the top-level design and one or more levels in the hierarchy, select the Node Voltages and/or Measurement Equations options and set the desired Maximum Depth.

Note
 When you use the Save by hierarchy options, all data from the specified levels is output-you cannot restrict it. However, you can add to it, selectively, from lower levels in the hierarchy using the Save by name section.

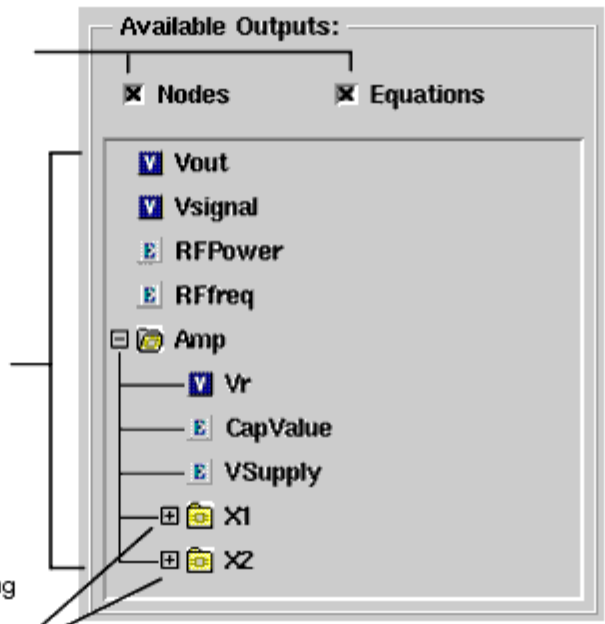
- To output named nodes/measurement equations selectively, irrespective of the hierarchy, disable the Node Voltages and/or Measurement Equations options in the Save by hierarchy section and use the Save by name section to select only those nodes/equations you want to output. (Click Add/Remove to open the Edit Output Plan dialog box.)
2. If using the Save by hierarchy method, select the desired level of hierarchy and click OK.
 3. If using the Save by name method (alone or in conjunction with a specified level of hierarchy), click

The Edit Output Plan dialog box appears with a list of available nodes and equations.

Use these to limit the list to one or the other to facilitate the selection process.

All named nodes, measurement and VAR equations, and buses that you can output.

Additional folders representing subnetworks that can be expanded to show additional data.



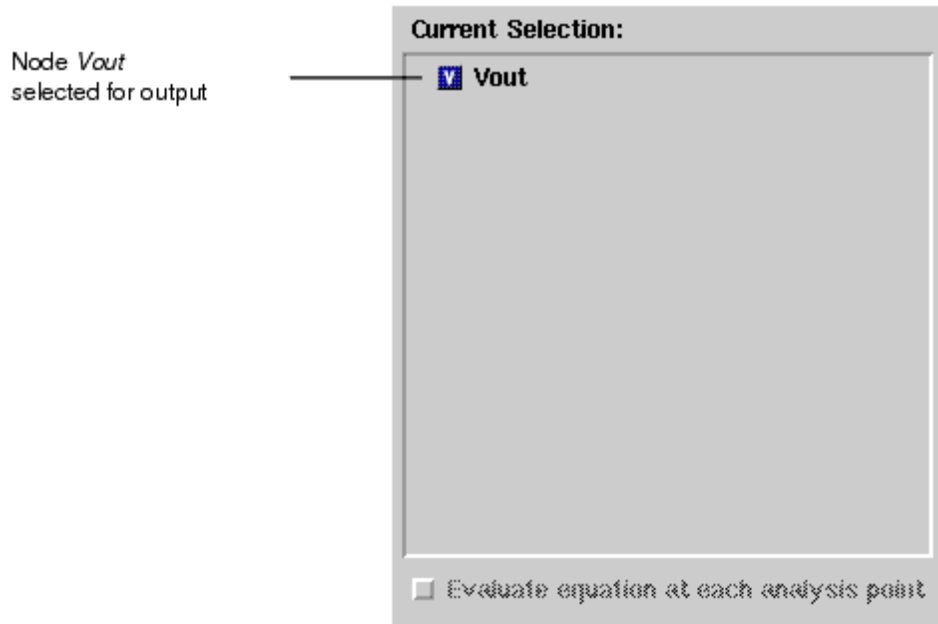
Node Voltages/Buses

Meas and VAR eqn.

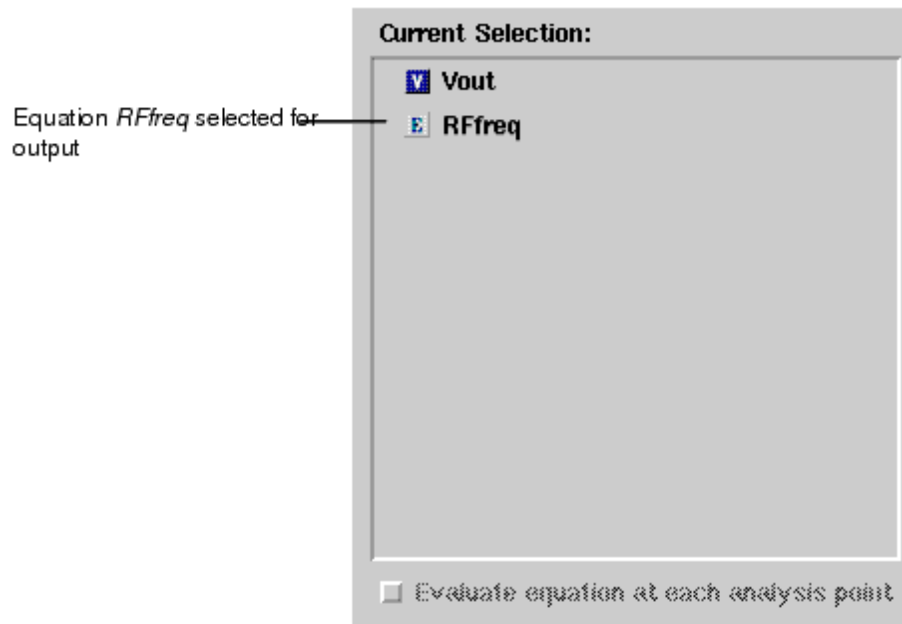


Note
 Buses appear in the list box as nodes. They do not contain any indices (as they do in the design environment). Individual components of a bus cannot be output.

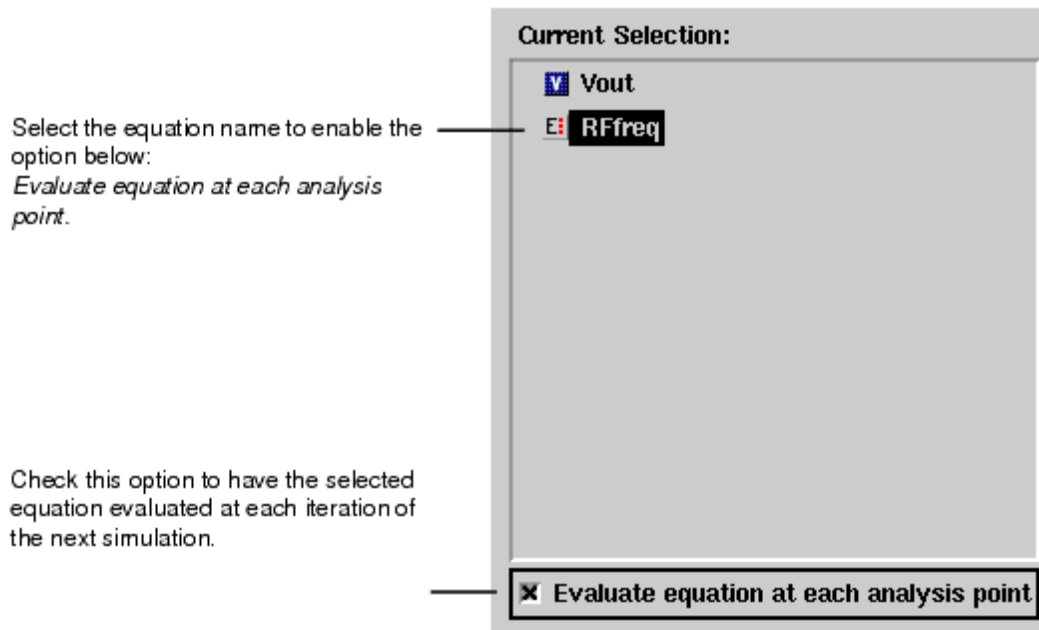
4. Select each individual node from the Available Outputs list that you want to send to the dataset and save for each iteration (time, frequency) for the next simulation. Click Add to move each to the Current Selection list box. Selected nodes will be stored in the parameter NodeName[i] on the schematic.



5. Select each individual equation you want to send to the dataset and save at the end of the next simulation. Click Add to move each to the Current Selection list box. Selected equations will be stored in the parameter SavedEquationName[i] on the schematic.



6. In addition, you can also select individual equations appearing in the Current Selection list box to be saved for each iteration (time, frequency) of the next simulation. Select an equation in the Current Selection list box and check the option labeled Evaluate equation at each analysis point. The option must be checked separately for each equation appearing in the list. The checked equation will be stored in both parameters SavedEquationName[i] and AttachedEquationName[i] on the schematic.



Note
A non-selected equation may still be output depending on the Maximum Depth setting, but all data will be processed at the end of all simulation iterations, requiring more memory.

If an equation is selected for any one simulation, it will not be output for any other simulation for which it is not explicitly selected.

- Click OK to accept all changes. The selected nodes and equations are then displayed in the Save by name section on the Output tab.
- Make any other desired changes for this analysis and click OK.

Output Tab Fields for Simulation Controllers

Following are more detailed explanations of the dialog box fields.

Save by hierarchy. Enables you to save the data in the active hierarchical designs. To output all named nodes/measurement equations in the top-level design only, select the Node Voltages (`UseNodeNestLevel`) and/or Measurement Equations (`UseSavedEquationNestLevel`) options and use zero as the Maximum Depth (`NodeNestLevel/SavedEquationNestLevel`). To output all named nodes/measurement equations in the top-level design and one or more levels in the hierarchy, select the Node Voltages and/or Measurement Equations options and set the desired Maximum Depth. When you use the Save by hierarchy options, all data from the specified levels is output-you cannot restrict it. However, you can add to it, selectively, from lower levels in the hierarchy using the Save by name section (see next paragraph).

Save by name. Identifies the names of individual nodes and equations that you want to save to a dataset. To output named nodes/measurement equations selectively, irrespective of the hierarchy, disable the Node Voltages and/or

Measurement Equations options in the Save by hierarchy section described in the preceding paragraph and use this section, along with the secondary Add/Remove dialog box (see next paragraph) to select only those nodes/equations you want to output.

Add/Remove. This button opens the Edit Output Plan dialog box, which corresponds to the node voltages and measurement equations included in the active design. In this dialog box, deselect Equations (`SavedEquationName[1]`, `AttachedEquationName[1]`) if you want to view a shorter list while adding selections. Choose the Add button to select equations that you want to include in the data to be saved. Choose the Remove button to take equations out of the Current Selection list. The selections in this list will be reflected in the Save by name box in the controller dialog box, and will be evaluated after your analysis is finished. If you would like the selected equation to be evaluated at each analysis point, select the Evaluate equation at each analysis point option.

Displaying Simulation Parameters on the Schematic

You can reduce screen clutter by displaying on the schematic only the parameters you are interested in. Whether a parameter is displayed or not does not affect its functionality. However, some parameters must be displayed to be used.

DC Simulation Display Options

Display

Display parameter on schematic-Enables you to set the visibility of simulation parameters on the schematic.

- Set All-Use this option to quickly select all parameters, and then deselect those you do not want to display.
- Clear All-Use this option to quickly deselect all parameters, and then select only those you want to display.

Controlling a Simulation

You can select how to start and end a simulation:

- To start a simulation, choose Simulate from the Simulate menu. You can also start one by clicking the Simulate button on the tool bar or by pressing F7.
- To end a simulation before it is finished, choose Stop and Release Simulator from the Simulate menu. This will release your simulation license. To end the simulation but keep the license, choose Simulation/Synthesis > Stop Simulation from the Simulation Message window.

You can add more than one simulation component to a schematic, and specify which simulation to run (only one simulator can be active at a time):

To disable simulations that are not desired, choose Edit > Component > Deactivate/Activate and click the appropriate simulation component.

For more information about deactivating simulation controllers, as well deactivating/activating other components in your design, see [Activating, Deactivating, and Shorting Components](#).

If using the Load Sharing Facility (LSF) utility, you can break up a sweep and run the simulation on multiple machines, in parallel, by selecting Parallel Hosts as the Simulation Mode (Simulate > Simulation Setup). Individual sweep points are run on each machine and the results are combined into a single dataset on the local machine. You can also use this utility to select the fastest available machine.

For details on setting up remote and local machines for remote processing, see "Using Remote Simulation" in the installation documentation for your platform:

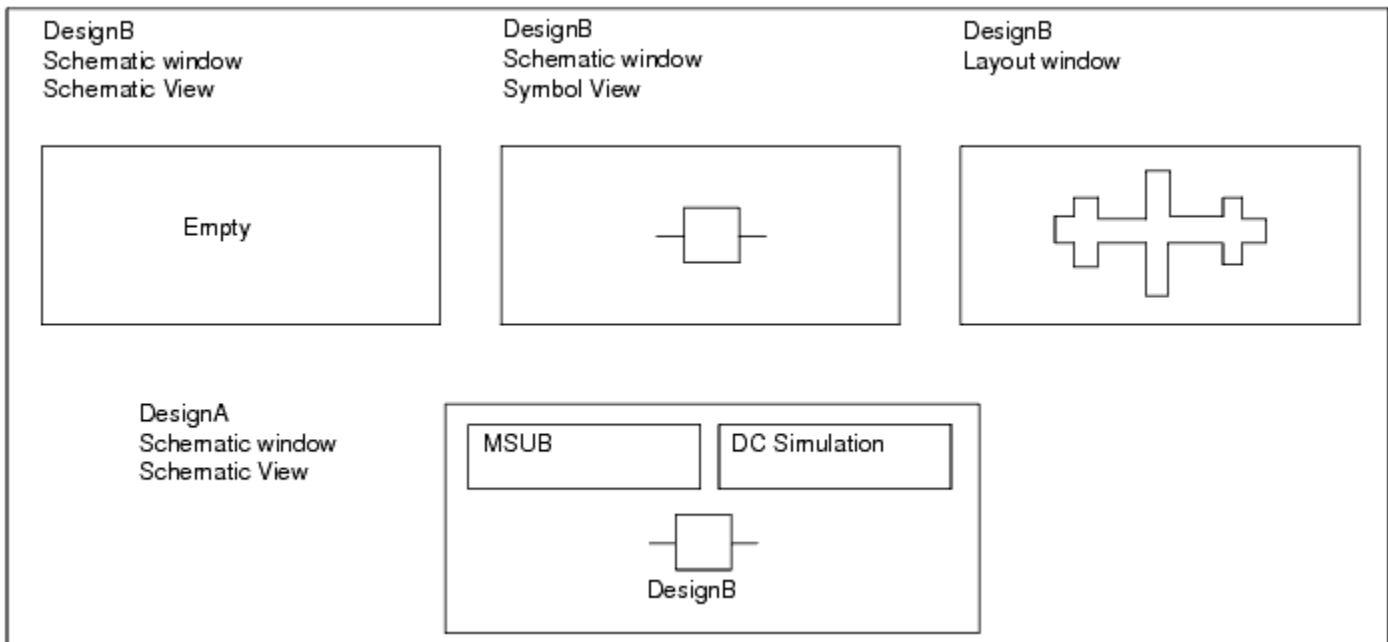
- [Windows Installation](#)
- [UNIX and Linux Installation](#)

Simulating from a Layout

Simulating a layout cannot be done directly from the Layout window; it involves a few steps in the Schematic window. Essentially, you must treat the design as though it were a subnetwork and place it in a higher-level design. To do this, you must create a symbol for it (in the Schematic window).

To simulate a layout:

1. From the Layout window containing the design you want to simulate (in this example, DesignB), choose Window > Schematic.
2. In the Schematic window for DesignB, create a symbol (View > Create/Edit Schematic Symbol). The number of pins must equal the number of ports on the design.
3. Switch back to the Schematic view (View > Create/Edit Schematic) and choose File > Design Parameters and select the option labeled Simulate from Layout (SimLay).
4. Save the design.
5. From the Schematic window, create a new design (in this example, DesignA).
6. In the Schematic window for DesignA, open the Component Library and select and place an instance of DesignB.
7. Place the desired simulation control items, as well as any substrate or equation definitions, in DesignA and run the simulation from DesignA.



Viewing DC Solutions

After a simulation is finished, you can display DC node voltages and branch/pin currents on the schematic. Because a DC simulation is part of most other types of simulations as well, this feature is available for most simulations.

- To view DC solutions, from the Schematic window, choose Simulate > Annotate DC Solution. All node voltages and branch/pin currents of the last DC solution – which was obtained from either the last explicit or implicit DC analysis – are displayed on the schematic.
- To erase the solutions from the schematic, choose Simulate > Clear DC Annotation.

Viewing Device Operating Point Data

Any simulation that includes a DC analysis produces DC operating point information for most active and some passive devices in the circuit. This data includes currents, power, voltages, and linearized device parameters of the selected device. An explanation of the displayed parameters, if available, is under the model documentation in the Circuit Component manual.

To view device operating point data:

- From the Schematic window, choose Simulate > Detailed Device Operating Point. Crosshairs appear. Click the component of interest. The details appear in a separate window.

- To view a condensed list of details, choose Simulate > Brief Device Operating Point instead.
- You can save device operating point data to a dataset for viewing in the Data Display. Under the Parameters tab of most simulators, set the Device operating point level as desired.

Displaying Simulation Results

Most of the simulation results are viewed in the Data Display. You can set the Data Display so that it automatically opens when a simulation is finished:

1. Choose Simulate > Simulation Setup.
2. Select the option Open Data Display when simulation completes.
3. Specify a data display file (.dds) and it will be opened in the Data Display window when the simulation is finished.

The simulation templates include DisplayTemplate components. If your design includes a simulation template and the automatic display option is enabled, then the Data Display window will open with the corresponding display template.

If neither a data display file (.dds) is specified nor a template used, a blank Data Display window is opened.

Subsequent simulations of the same schematic will not open a new window, but rather bring the existing one to the foreground.

For information on how to work with the items in a Data Display window, see [Data Display](#).

There are also ways to view DC data directly from the schematic. You can view:

- DC node solutions
- DC operating point data

Tuning

ADS tuning capability enables you to change one or more design parameter values and quickly see the effect on the output without resimulating the entire design. Multiple traces generated from various tuning trials can be overlaid in the Data Display window. This can help you find the best results and the most sensitive components or parameters more easily.

Basic tuning consists of the following steps:

1. Build the design you want to tune.
2. Set up your simulation.
3. Simulate your design and verify that your simulation operates as expected.

4. Set up, display, and analyze your results in the Data Display window.
5. Choose Simulate > Tuning or click the Tune Parameters icon (tuning fork) on the toolbar.



When the initial analysis is complete, the Tune Parameters dialog box appears.

6. Select each parameter you want to tune by clicking it on the schematic. The Tune Parameters dialog box is updated with a new slider for each parameter selected.
7. Change the tunable parameter(s) by moving the slider(s), or clicking the up/down arrows.
8. Update your schematic with the changes.

For complete details about tuning your design, see [Tuning, Optimization, and Statistical Design](#).

Reusing Simulation Solutions

For some types of simulations, you can save a simulation solution, reusing it as an initial guess in a later simulation. This can save time by avoiding repeating the same, or a very similar simulation, on a design. Using a simulation solution as an initial guess can help the subsequent simulation to reach a final answer faster. If you have made minor changes to the design or simulation setup, reusing solutions can reduce CPU time.

You can save and reuse these types of simulation data:

- DC solutions-You can save the complete DC solution and reuse it for any type of simulation that either performs or relies on a DC solution. For example, once a DC solution is obtained by running an AC simulation, then future AC simulations at different frequencies or linear noise simulations do not have to resimulate to get the same DC solution again. This feature is available from the DC Solutions tab of the Options component. For details, see [Using the Simulator Options Component](#).
- Harmonic Balance solutions-You can save a harmonic balance solution and use it as an initial guess for another harmonic balance simulation, large-signal S-parameter, gain compression, or Circuit Envelope simulation. With the saved harmonic balance solution, you can later perform a nonlinear noise simulation and use the saved solution as the initial guess, removing the time required to recompute the nonlinear harmonic balance simulation. Another use would be to use the initial harmonic balance solution, then sweep a parameter to see the changes. For details, see [Reusing Simulation Solutions](#).
- Harmonic Balance guess from a Transient simulation-Transient simulations can be set to generate a harmonic balance solution that can then be used as an initial guess for a harmonic balance simulation. For example, in circuits such as dividers, harmonic balance usually cannot directly converge on a solution since multiple mathematical, but useless, solutions exist. By first running a transient simulation and generating a harmonic balance solution file that can then be used as an initial guess, the harmonic balance simulation can converge to the desired solution. This feature is available when setting up the Harmonic Balance or Transient controller. For details, see the following documentation:
 - In [Harmonic Balance Simulation](#), see [Transient Assisted Harmonic Balance](#).
 - In [Transient and Convolution Simulation](#), see [Using the Steady State Detector and Transient Assisted Harmonic Balance](#).

Analog/RF Simulation Computations and Convergence Criteria

Analog/RF simulation computes the response of a circuit to a particular stimulus by formulating a system of circuit equations and then solving them numerically. Each simulation technology accomplishes this analysis as follows.

DC analysis

- Solves a system of nonlinear ordinary differential equations (ODEs).
- Solves for an equilibrium point.
- All time-derivatives are constant (zero).
- System of nonlinear algebraic equations.

Transient analysis

- Solves a system of nonlinear ordinary differential equations (ODEs).
- Time-derivatives replaced with a finite-difference approximation (integration method).
- Sequence of systems of nonlinear algebraic equations (one system at each timepoint).

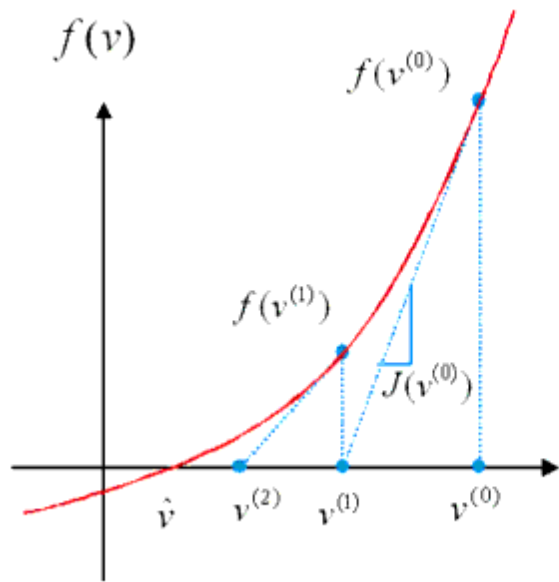
Harmonic Balance (HB)

- Solves a system of nonlinear ordinary differential equations (ODEs).
- Steady-state method.
- Solution approximated by truncated Fourier series.
- System of nonlinear ODEs becomes a system of nonlinear algebraic equations in the frequency domain.

Solving Nonlinear Algebraic Equations

Nonlinear algebraic equations are solved using the Newton-Raphson algorithm (Newton's method) as follows.

1. Convert the problem to a sequence of systems of linear equations.
2. Quadratic convergence near the solution (error squared at each iteration).



- **Newton's method on**

$$f(\hat{v}) = 0$$

- **guess** $v^{(0)}$
- **linearize about** $v^{(0)}$
- **solve for next guess** $v^{(1)}$

- **If all goes well:**

$$v^{(k)} \rightarrow \hat{v} \quad \text{as} \quad k \rightarrow \infty$$

Common Circuit Simulation Methods

Backward Euler

- First order method that assumes the solution waveform is linear over one time step
- One-step method (needs one previous time point solution only)
- Adapts faster to abrupt signal changes
- Stable on all stable differential equations and some unstable ones.
- Exhibits heavy numerical damping, increases loss
- Require smaller time step to maintain accuracy

Trapezoidal Rule

- Second-order method, assumes the solution waveform is quadratic over one time step
- One-step method
- May exhibit point-to-point ringing on circuits that have very small time constant comparing to time step (stiff circuit)
- Stable only on stable differential equations
- Exhibits no artificial numerical damping

Backward Difference Formulas (Gear's methods)

- Multiple order polynomial over one time step
- Only the first six orders are available in ADS
- First order method is identical to backward Euler
- Higher-order polynomials allow a larger time step without sacrificing accuracy, are efficient for smooth waveforms
- Higher order methods (order > 2) may exhibit stability problems on lightly damped circuits
- Second-order backward difference formula (Gear 2)
- Two-step method
- Stable on all stable differential equations and some unstable ones.
- Exhibit some numerical damping

Truncation Error

The error made by replacing the time derivatives with a discrete-time approximation. This error is difficult to estimate and depends on the type of circuits and the time steps.

Local Truncation Error (LTE)

The truncation error made on a single step

Global Truncation Error (GTE)

- Maximum accumulated truncation error
- The circuit with long time constant is sensitive to these errors
- Logic and bias circuits are not sensitive to these errors

Convergence Criteria

Newton's iteration is converged if the approximate solution first satisfies the Residue criteria at the end of each Newton iteration and the Update criteria once the residue criteria are satisfied.

Residue criterion

KCL satisfied to a given tolerance. This is enforced at each node and is important when impedance at a node is small.

Update criteria

Difference between the last two iterations must be small. This is important when impedance at a node is large.

Using Continuation Methods

Use continuation methods to provide a sequence of initial guesses that are sufficiently close to the solution to assure Newton's method convergence.

- Choose a natural or contrived continuation parameter which controls a modification of the circuit
- Step the continuation parameter from 0 to 1 (the original circuit configuration), using the solution from the previous step as the starting point.

As long as the solution changes continuously as a function of the continuation parameter and the steps are small enough, Newton's method will converge. Keep in mind though that the first two methods, Source and gmin stepping, will fail if the continuation path contains a limit point.

Source Stepping

Uses a fraction of the source voltages and currents applied to the circuit as the continuation parameter.

- Turn off all sources when the continuation parameter equals 0.
- Raise source levels to their final levels slowly, generating a sequence of circuit configurations.
- Use the solution from the previous configuration as an initial guess for the current configuration.

Gmin Stepping

Uses the continuation parameter to control the value of the gmin resistors

Advanced Design System 2008

- Start with a large gmin for an easy to compute solution because nonlinear device behavior is muted by the presence of the small resistors.
- End with very small gmins for resistors that are so large that they no longer affect the circuit.
- Remove the gmins to compute the final solution.

Arc-Length Continuation

Works best for complicated continuation paths and limit points using a continuation parameter that is a function of the arc-length parameter

- Travel same distance at each step, as specified by the arc-length.
- Increase or decrease the continuation parameter along the path in each step.

Preventing Convergence Problems

Convergence problems usually arise as a result of errors in circuit connectivity or unreasonable (out of range) model or component values. Some of the steps you can take are as follows.

- Turn on the topology checker (default is on). In ADS, set the topology checker mode on the Options component's Misc tab.
- Turn on warnings.(default is no). In ADS, set warnings on the Option component's Output tab.
- Act upon the messages in the Status window.
- Eliminate small floating resistors (or increase I_AbsTol). Any error in computed voltages for nodes with small resistors results in large error currents.
- Avoid very large and very small resistances connected to a node. Large resistances are lost during Jacobian construction due to numerical round-offs.

Clearing Highlights from Items Causing Simulation Errors

When an error occurs during simulation, a box is drawn around each item causing an error. To clear all highlights, choose the Clear Highlighting command from the View menu in the appropriate window.



Hint

The color of this identifying box is the Highlight color defined through Options > Preferences > Display.

Using Circuit Simulators for RF System Analysis

The steady-state simulation of RF/IF subsystems in the frequency domain is achieved in Advanced Design System through the use of various circuit and system simulation components, as well as through a variety of measurement functions that can be applied to simulation data. Budget, spur (spurious signal), noise, and group delay data are typical objectives obtainable with these components, and a sweep analysis is a common way to obtain system response at a variety of frequencies, power levels, and other operating conditions.

For details refer to the following related topics, including a variety of examples:

- [Applicable Simulation Components](#) lists the simulators that are best suited for RF system analysis.
- [Applicable Measurements](#) lists the types of measurements that work well with RF system analysis.
- [Fundamentals of Using Circuit Simulators for System Analysis](#) illustrates a typical RF system setup that can be used with circuit simulation.
- [Budget Analysis](#) gives an overview of budget analysis, describes several examples, and discusses limitations in [Budget Analysis Capabilities](#).
- [Using IMT-Based Mixer Models in Spurious Signal Analysis](#) describes how to use the IM tables and behavioral mixer models for generating spurious signals and the approach to simulating spurious signals.
- [System Noise Analysis](#) is an overview of contributors to system noise and how they are treated in a simulation.



Note

The appropriate simulator licenses are required to run simulations examples, i.e. a Harmonic Balance simulation requires the Harmonic Balance simulator license (included in all Circuit Design suites except the RF Designer suite.) You may build the examples without the appropriate license, but will simply be unable to run the simulations.

Applicable Simulation Components

The circuit simulation components that can be used for system analysis are the AC, S-Parameter, Harmonic Balance, LSSP (including the use of the P2D Simulation component to generate power-dependent S-parameters), and XDB components. This topics assumes that you are familiar with how to use those simulators to analyze circuits and display data. This section presents just a few of the RF system objectives that can be obtained.

By selecting the Harmonic Balance Simulation (HB) component in the Simulation-HB palette, you can achieve the following:

- Perform a budget analysis to determine the signal and noise performance for elements in an RF system network. This includes measuring system performance at an element's input or output, and therefore finding the degree to which an element contributes to the degradation of system performance.
- Perform a sweep analysis to determine the network's port-to-port performance with respect to a swept parameter such as frequency or power.
- Perform a spurious-signal analysis to determine the network's spurious spectral tones, where all intermodulation

products are due not only to mixer signal input and local oscillator mixing, but also to the nonlinearities of amplifiers.

- Use the Perform Budget simulation option to obtain currents and voltages at named nodes throughout the system. Use budget measurements to postprocess the resulting data.

By selecting the AC Simulation component (AC) in the Simulation-AC palette, you can achieve the following:

- Use the Enable AC Frequency Conversion option to do a small-signal analysis in systems containing freq.
- Perform a small-signal noise analysis at the IF at a variety of nodes.
- Use the Perform Budget simulation option to obtain currents and voltages at all pins of each element in the system. Use budget measurements to postprocess the resulting data.

By selecting the S_Param (SP) component, in the Simulation-S_Param palette, you can achieve the following:

- Determine linear scattering parameters (S-parameters).
- Use the Group delay option to analyze the group delay from the input to the output of the system.

By selecting the LSSP simulation component (LSSP), in the Simulation-LSSP palette, you can achieve the following:

Perform a large-signal sweep of, for example, gain versus frequency and power, to determine the effect of gain compression on S-parameters.

By selecting P2D Simulation component (P2D), in the Simulation-LSSP palette, you can achieve the following:

Create a system-level amplifier model by generating a power-dependent S-parameter file for a circuit-level amplifier design.

Applicable Measurements

From among the many measurements that are available, a variety of those specific to system analysis are found in their respective simulation palettes. The following table lists some of the system-applicable measurements that are available on the schematic page, and shows the palettes in which they can be found:

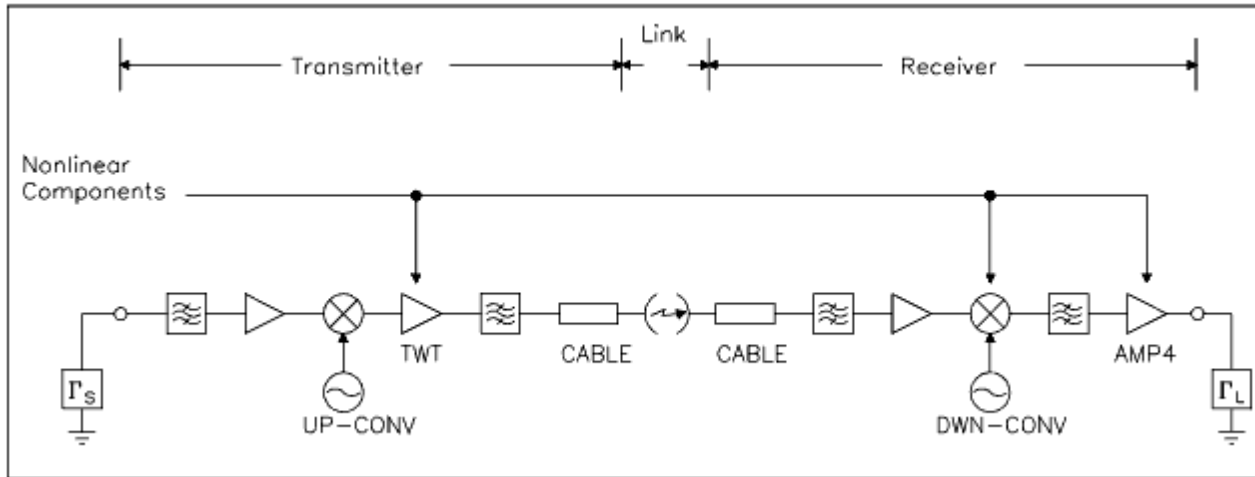
Palette	Measurement	Description
Simulation-HB	BudLinearization	Linear budget simulation
	Pspec	Power frequency spectrum
	Vspec	Voltage frequency spectrum
	Ip3in	Input third-order intercept point
	Ip3out	Output third-order intercept point
	CarrToIM	Ratio of carrier signal power to IMD power
	SFDR	Spurious-free dynamic range
	SNR	Signal-to-noise ratio
Simulation-LSSP	BudLinearization	Linear budget simulation
	GainComp	Gain compression
	PhaseComp	Phase compression
Simulation-S-Param	PwrGain	Power gain
	MaxGain	Maximum available gain
	NsPwrRefBW	Noise power in a reference bandwidth

These measurements and others are placed in the Schematic window and used in conjunction with a simulation.

Fundamentals of Using Circuit Simulators for System Analysis

The following figure shows a block diagram of a typical RF system. Such a system often consists of many interconnected linear and nonlinear "black-box" elements representing amplifiers, filters, mixers, modulators, demodulators, transmission lines, and radio links, as well as source and load matching elements. Each black box can in turn be composed of many circuit elements.

At the system simulation level, the signal transformation property of each black box is known, but the internal circuit characteristics of the elements need not be of concern.



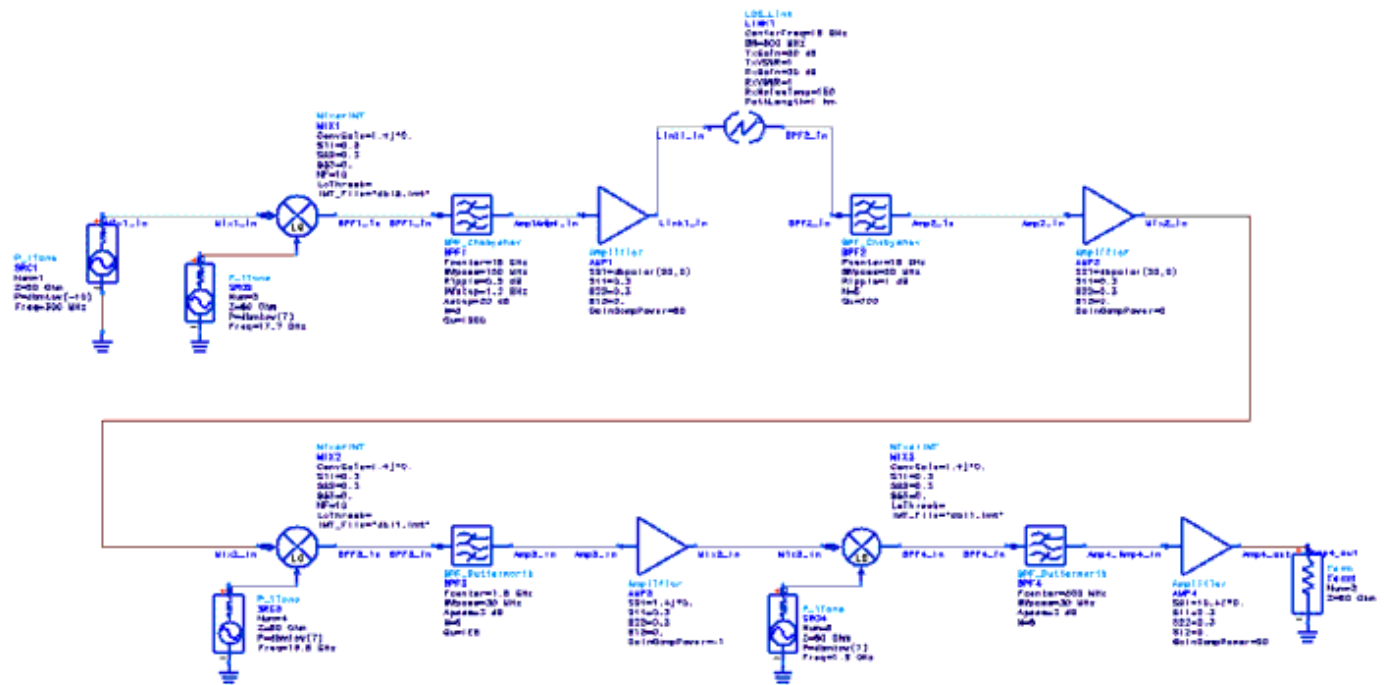
Block diagram of a typical RF system

The following figure shows an example RF system as represented on the schematic page. A variety of applicable simulation components and measurements are made available on the example schematic; turn them on and off as needed.

Note
 This design, RF_SYS1.dsn, can be found in the examples directory under Tutorial/SimModels/networks. The results of the simulation can be found in RF_SYS1_spectra.dds.

Example

The system depicted in the RF_SYS1.dsn uses a single upconversion stage and two downconversion stages. The RF frequency is 300 MHz, as defined in the P_1Tone source PORT1. The RF input power level is -10 dBm. This tone sums with 17.7 GHz from the first LO source, SRC1, to produce (among other tones) an IF of 18 GHz when mixed by the first upconversion mixer, MIX1. MIX1 and the other mixers in this example use the MixerIMT component, which relies on intermodulation tables to produce spurs. The parameter IMT_File references the table used for this component. For a discussion of IMT files as they are used here, refer to [Using IMT-Based Mixer Models in Spurious Signal Analysis](#).



Example RF system

The Chebyshev filter BPF1 (note the passband BWpass and quality factor Qu) then selects and passes the 18-GHz signal to amplifier AMP1, where the signal receives 20 dB of gain at a phase of 0 degrees. LINK1 applies both a transmitter and a receiver gain of 30 dB and adds the loss that would be incurred over a path length of 1 km. BPF2, another Chebyshev filter, receives the signal and passes it through to AMP2, which applies 30 dB of gain at 0 degrees phase.

MIX2 combines the 18-GHz signal with 16.5 GHz from the LO, SRC2, to produce (among other tones) a 1.5-GHz tone at the mixer's output. This time the signal is passed through a much narrower (note the high Q) Butterworth filter to a low-gain amplifier, AMP3. From there the signal proceeds to a final downconversion stage, MIX3.

The signal input to MIX3 combines with 1.2 GHz from the LO at SRC3 to produce (among other tones) 300 MHz, the frequency originally inserted into the system. This is filtered by BPF4 (note the very high Q) and amplified by AMP4. Finally, a 50 ohm load at the output terminates the signal path.

Later we will revisit this design and the results of simulations on it.

Budget Analysis

Budget analysis determines the signal and noise performance for elements in the top-level design. Therefore, it is a key element of system analysis.

Budget measurements are performed upon data generated during a special mode of circuit simulation. AC and HB simulations are used in budget mode depending upon if linear or nonlinear analysis is needed for a system design. These measurements show the performance at the input and output pins of each element of the system at the top-level design. This enables the designer, for example, to adjust the gains or to reduce the nonlinearities of various components. These measurements can also indicate the degree to which a given component can degrade overall system performance.

Budget measurements include power gain, incident and reflected powers, noise figure, VSWR, and a variety of nonlinear measurements, such as SNR and gain compression.

There are various ways to obtain budget data:

- Use the Perform Budget simulation option (available in the HB and AC simulation dialog boxes). This option is required if budget measurements are to be used following a simulation (see below). Alternatively, the flag OutputBudgetIV can be set to Yes.
- Use the budget measurement components, available in the AC and HB simulation palettes. By placing one or more of the budget measurement components on the schematic and by selecting the required options, budget data can be generated.
- Add a budget path to your schematic using Simulate > Generate Budget Path. Budget data will be generated for the specified portion of the circuit. This is used in conjunction with other measurement components.
- Use Measurement Functions, available in Data Display windows as functions that can be input directly into an equation. First, the appropriate data must be referenced in the default dataset.
- Use the BudLinearization Component, available in the Harmonic Balance, LSSP, and XDB simulation palettes. This component, which must be used in conjunction with one of the harmonic balance simulators, provides information regarding the nonlinear effect of circuit elements.

Using the Perform Budget Simulation Option

Two simulators provide a budget simulation option:

- In the Harmonic Balance Simulation component (HB), select the Params tab, then select Perform Budget simulation.
- In the AC Simulation component, select the Parameters tab, then select Perform Budget simulation.

Using Budget Measurement Components

The budget measurement components are available in the AC and HB Simulation palettes, and must be used by selecting the required options. A budget data can be generated by placing one or more of these budget components in schematic.

The budget results at the terminal(s) of each element are sorted in ascending order of the component names. These

component names are attached to the budget data as additional dependent variables.

Adding a Budget Path

You can generate budget results over a specified path in your circuit using Simulate > Generate Budget Path. You use this in conjunction with other budget measurement components, by replacing the pin with the name of the budget path. First specify the path, then modify your budget measurements.

To specify a path:

1. From the menu bar, choose Simulate > Generate Budget Path.
2. The names of the components in the circuit appear in two lists. Set the start of the path by selecting the name of a component in the left list.
3. Set the end of the path by selecting the name of a component in the right list.
4. Click Generate. To verify that the path is correct, click Highlight to highlight the path on the schematic. Click Clear to erase the highlighting.
5. If you are satisfied with the path, click Close.
6. Search the schematic for a measurement called BudPath, which was created when you invoked the command Generate Budget Path. You can change the name of the equation if desired. The default name is budget_path.

To modify your budget measurements:

1. Select a budget measurement and double-click to edit it.
2. Where the pinNumber is specified in the budget expression, replace it with the name of the budget path measurement.

Notes

In budget expressions supporting the alternative syntax using SourceName as the first parameter, don't use the SourceName parameter to specify the budget path. You should still replace the pinNumber variable with the budget path measurement name.

For the bud_gain expression, you can include the budget path measurement name only by using the alternative syntax as shown here:

```
x=bud_gain("SourceName",SrcIndx,Zs,Plan,budgetPath)
```

Using Budget Measurement Functions


The budget functions can also be entered by means of the Eqn component in the Data Display window. The Perform Budget simulation option must be selected prior to a simulation before measurement functions can be used following

the simulation.

The budget function can refer only to the default dataset, that is, the dataset selected in the Data Display window.

You can use a variety of budget-related measurement functions in the Equation entry field in the Data Display window. These include the following:

bud_freq		bud_gain		bud_gain_comp		bud_gamma
bud_ip3_deg		bud_nf		bud_nf_deg		bud_noise_pwr
bud_pwr_inc		bud_pwr_refl		bud_snr		bud_tn
bud_vswr						


 **Note**
For details on the above and other measurement functions, open a MeasEqn component dialog box and click Help.

Using the BudLinearization Component

This component is available in the Simulation-HB, Simulation-LSSP, and Simulation-XDB palettes. It may be used in conjunction with the Harmonic Balance, LSSP, or XDB analysis controllers. The P2D controller currently does not support budget analysis. To use the BudLinearization component, place the component in the schematic and edit it to reference the simulation component to be used, as well as the circuit components that are to be linearized. If no component is specified, all the components in the top-level design are linearized one at a time.

The BudLinearization component first performs a regular harmonic balance simulation, and then looks at the DC operating point for each nonlinear system component in turn, linearizing the response of that component while the responses of the remaining nonlinear system components remain nonlinear. If a system component is in a subnetwork, the entire subnetwork will be linearized.

The results of BudLinearization analysis are sorted in ascending order of the names of the linearized components. LinearizedElementIndex, an integer independent variable, is attached to the data as an additional sweep variable. The first point corresponds to the results of the regular harmonic balance simulation when none of the components is linearized. Another variable, LinearizedElement, that contains the names of the linearized components is also generated.

 **Note**
The inclusion of the BudLinearization component in conjunction with a Harmonic Balance simulation causes N+1 harmonic balance simulations to be run, where N is the number of nonlinear components to be linearized. Consequently, such a simulation will take N+1 times as long.

Budget Examples

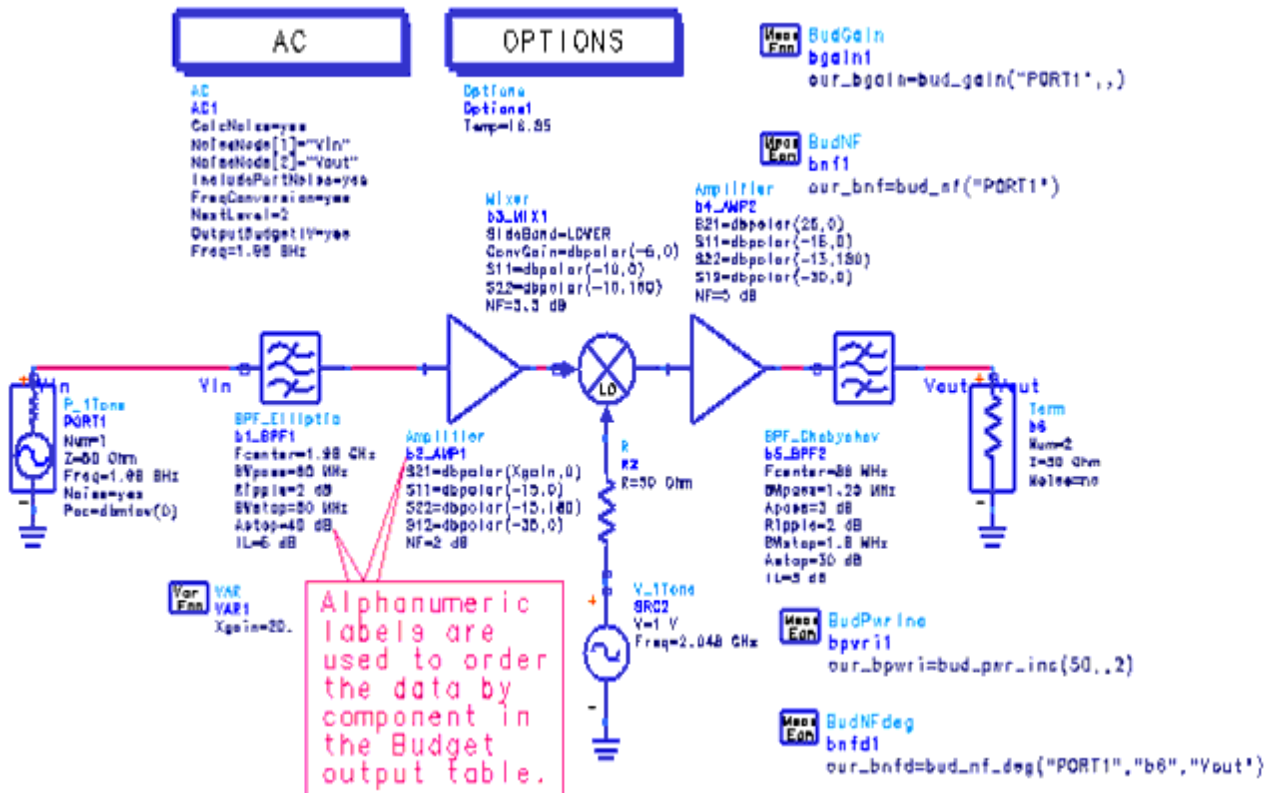
This section includes these budget examples:

- [Calculating Gain and Noise Figure](#) shows how to determine the power gain and noise figure budgets for a typical RF System. It uses the AC simulator and measurement functions.
- [Calculating Spurious Signals and TOI](#) shows how to use the Harmonic Balance simulator to analyze a PCS receiver.
- [Obtaining Budget Incident Power and Gain](#) illustrates how to use budget measurement functions and how to display results.
- [Obtaining Group Delay Data](#) uses the S-parameter simulator and linear sweep to calculate group delay data.

Calculating Gain and Noise Figure

The following figure illustrates one way to obtain budget gain and noise figure data.


Note
The design Linear_Budget.dsn is in the Examples directory under Com_Sys/Linear_Budget_prj. The results are in SchematicMeasurements.dd s.



Using the AC simulator to obtain budget gain and noise figure


Advanced Design System 2008

With its single mixer stage, this example provides a less complex view of the issues raised in RF_SYS1.dsn.

 **Hint**
To make it easy to observe the gains and losses contributed by individual components along the signal path, label them so that a List plot sorts them alphanumerically. In this case they have been labeled b1_BPF1, b2_AMP1, and so on.

This example uses the AC Simulation component, with parameters set as follows:

- Under the Noise tab, Calculate noise has been enabled. The noise nodes (the input of the first filter and output of the final filter) has been labeled "Vin" and "Vout" and have been added to the list of Nodes for noise parameter calculation. The Noise contributors mode is set to Sort by name.
- Under the Parameters tab, both Enable AC Frequency Conversion and Perform Budget simulation have been selected.
- Under the Frequency tab, the frequency is set to 1.960 GHz.

 **Note**
In nonlinear noise analyses, it is recommended that the Options component be used to establish a global simulation temperature of 16.85 degrees Celsius. This can be done by editing Temp=16.85 in the Schematic window, or by selecting the Misc tab and editing Simulation temperature to that value.

Add budget measurements to the schematic:

- From the Simulation-AC palette, add BdGain (budget gain), BudNF (budget noise figure), BudNFd (budget noise figure degradation), and BudPwrI (budget incident power).
- Double-click the budget noise figure degradation component to edit it. Edit the equation by replacing " term2 " with " b6 " and changing " vout " to " Vout". Note that b6 is the name of the output termination component.

To display all of the Budget measurements at once, open the Data Display SchematicMeasurements.dds. This will include gain, incident power, noise figure and noise figure degradation by component, both in tabular and graphic form. Ensure that the default dataset name is set to the name of the design you have simulated. The results of the simulation appear as follows:

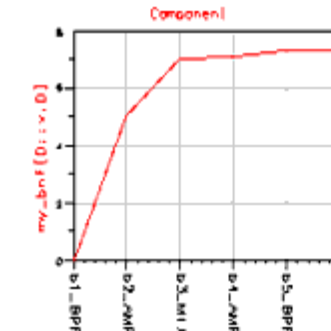
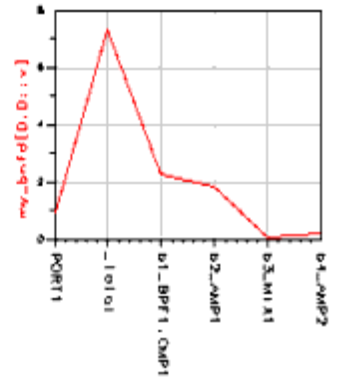
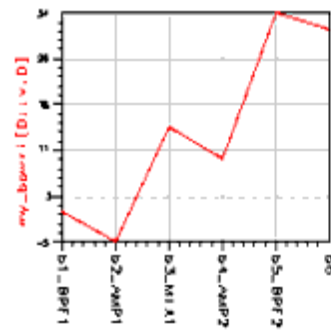
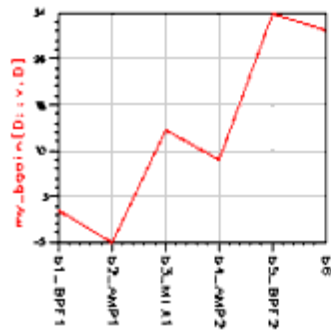
Component	my_bgain
	freq=1960000000.000
b1_BPF1	-0.022
b2_AMP1	-5.239
b3_MIX1	14.068
b4_AMP2	8.908
b5_BPF2	34.049
b6	31.047
PORT1	-0.022
R2	-3040.000
SRC2	-3040.000

Component	my_bwr1
	freq=1960000000.000
b1_BPF1	0.000
b2_AMP1	-5.001
b3_MIX1	14.524
b4_AMP2	9.048
b5_BPF2	34.048
b6	31.047
PORT1	-22.424
R2	-3010.000
SRC2	-3010.000

Component	my_bnf
	freq=1960000000.000
b1_BPF1	0.000
b2_AMP1	5.005
b3_MIX1	7.000
b4_AMP2	7.078
b5_BPF2	7.301
b6	7.301
PORT1	0.000
R2	0.000
SRC2	0.000

Index(my_bnf)	my_bnf
PORT1	0.895
_total	7.301
b1_BPF1.CMP1	2.237
b2_AMP1	1.834
b3_MIX1	0.073
b4_AMP2	0.223
b5_BPF2.CMP1	3.413E-4

```
Eqn >=sweep_size(my_bgain[0])-4
```



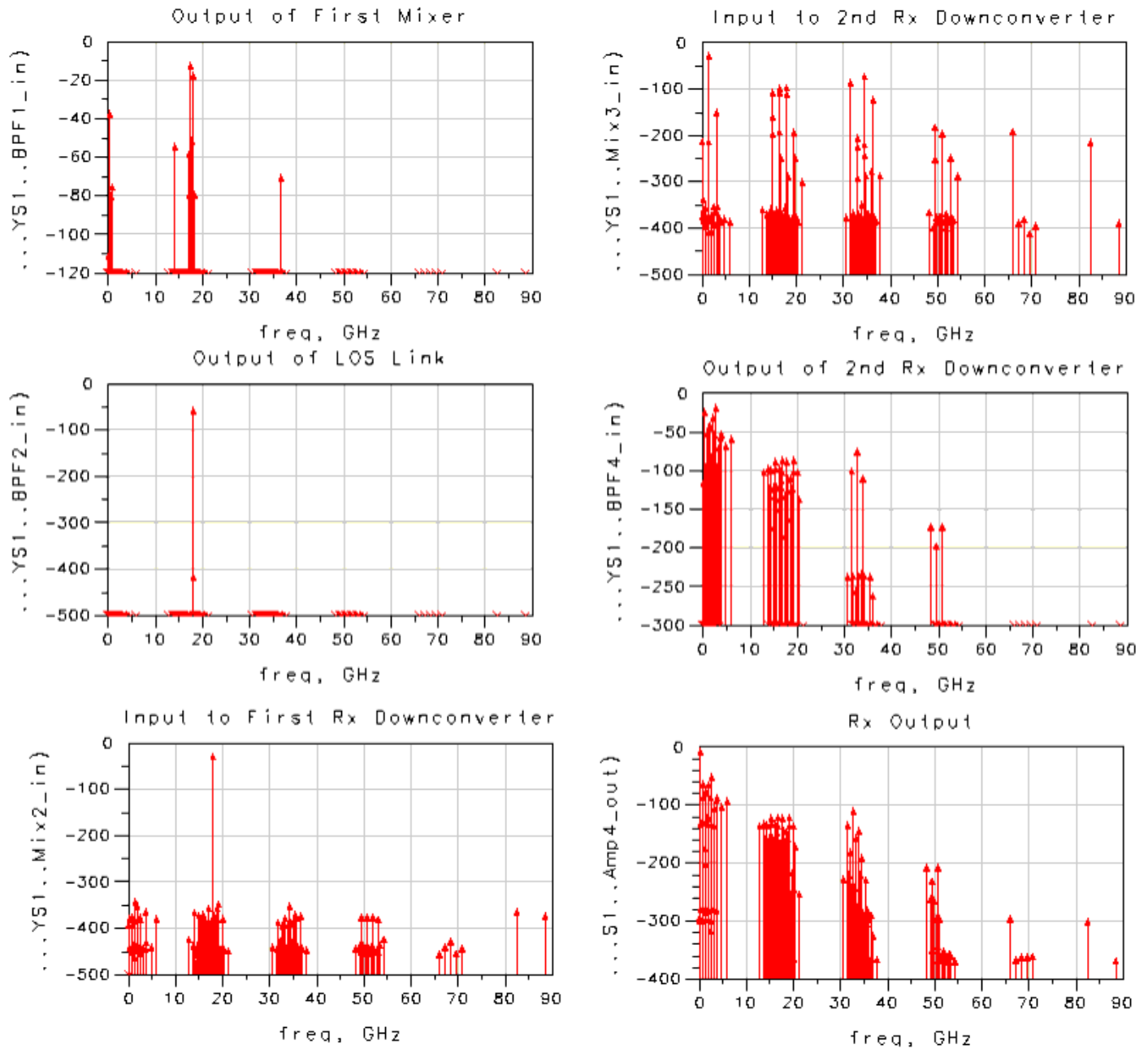
```
index(my_bnf[D::v,x])
```

The first listing shows the losses and gains, in dB, contributed by the various components. For example, the first amplifier has a nominal insertion loss (IL) of 5 dB at 1.960 GHz, but here the figure is -5.239 dB – the result of reflection and thermal loss. Try lowering the reflection parameters S 11 and S 22 to approximate ideal values (simply add zeros) to observe the bud_gain figure approach 5 dB. The second listing shows the incident power at the input of each component. The third and fourth listings show the noise figure and the noise figure degradation at pin 1 of each component, respectively. For more discussion of noise analysis, refer to [System Noise Analysis](#).

Budget measurements may also be performed entirely within Data Display window by adding equations that operate on the current and voltage data provided by the AC simulation. The Data Display DataDisplayMeasurements show an alternative way of using the budget functions, instead of adding measurements to a schematic, after a simulation you can get the same calculations by adding equations to the Data Display page.

Calculating Spurious Signals and TOI

The following figure illustrates traces of the spectral tones (spurious signals) for various nodes of the design RF_SYS1.dsn. For a detailed discussion of mixing products, refer to [Using IMT-Based Mixer Models in Spurious Signal Analysis](#).



Spectral data for RF_SYS1.dsn

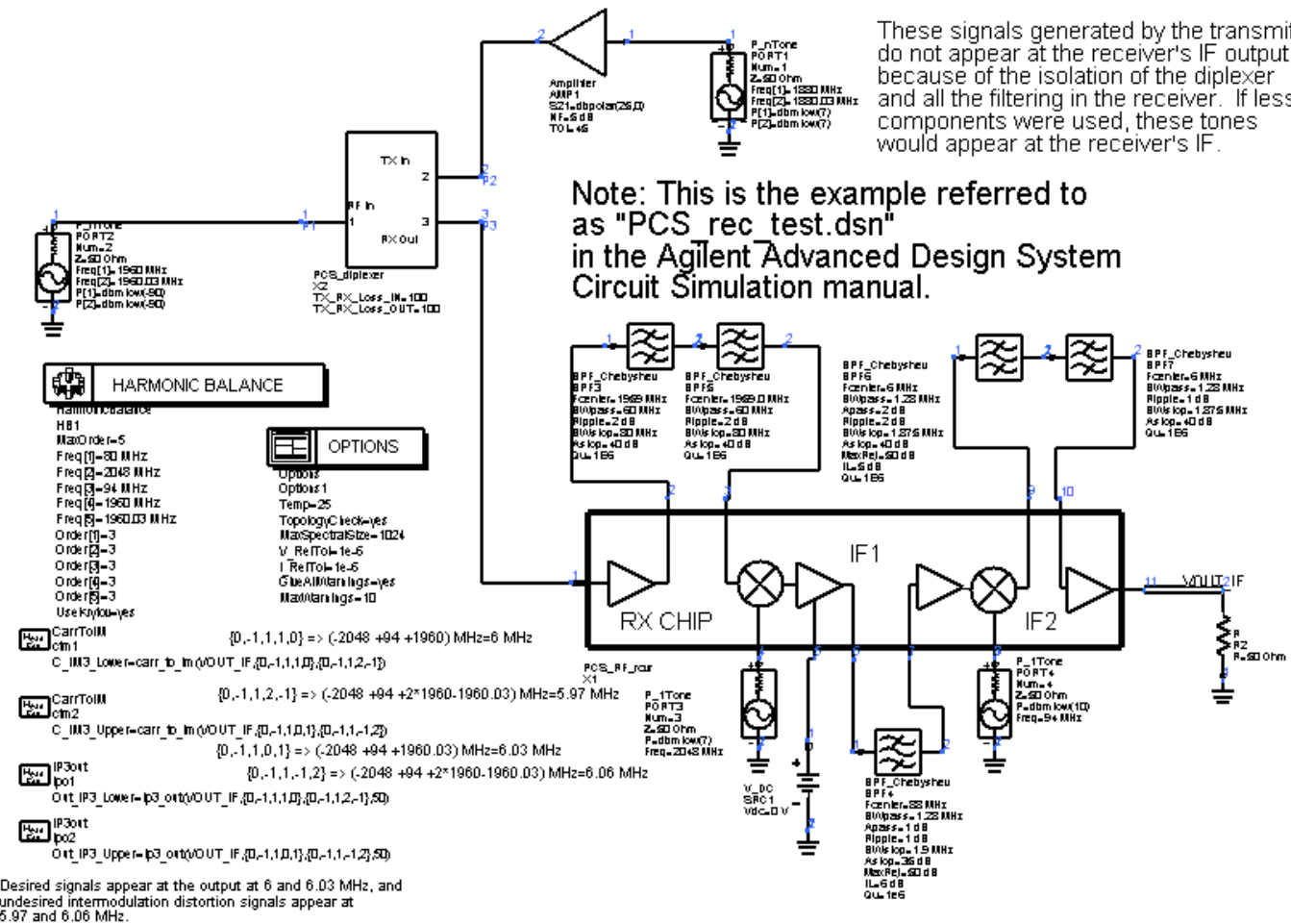
The following figure illustrates a setup for obtaining third-order intercept.

Note
The design PCS_Rx_TOI_test.dsn is in the Examples directory under Com_Sys/RF_System_prj. The results are in PCS_Rx_TOI_test.dds.

Receiver Third-Order Intercept Point and Carrier-to-Intermodulation Distortion Simulations

These signals generated by the transmitter do not appear at the receiver's IF output, because of the isolation of the diplexer and all the filtering in the receiver. If less ideal components were used, these tones would appear at the receiver's IF.

Note: This is the example referred to as "PCS_rec_test.dsn" in the Agilent Advanced Design System Circuit Simulation manual.



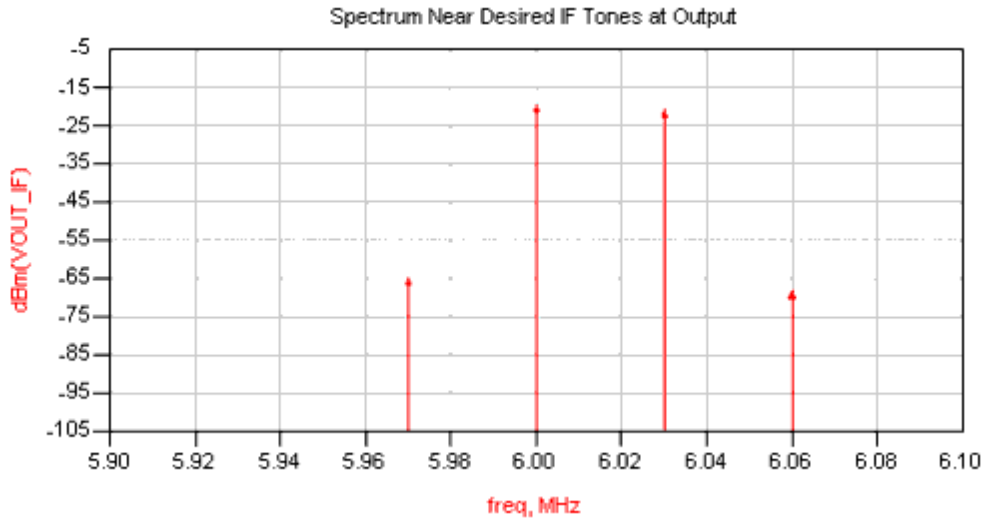
System TOI example

Two sources at the input define two closely adjacent tones, as can be found under typical conditions of communications interference. (A P_nTone component could also be used.) The source at the top of the schematic, identified as PORT1, can be used to test the degree to which the outgoing signal leaks back into the IF stage.

Note
CarrToIM and IP3out measurement components have been used to define those measurements for the lower

and upper intermodulation products (selected by means of the mixing indices vector) that will be passed by the filters. IP3out must reference 50 ohms. Also, although there are six apparent source tones in the design (at 1880, 1880.3, 1960, 1960.3, 2048, and 84 MHz), only five tones are independent.

A plot of the IF output, VOUT_IF, appears as follows:



The mixing products of interest center around 6 MHz, as determined by the final filters. Note that only one of the third-order intermodulation products (at 6.06 MHz) is safely below about -75 dB. The spur at 5.97 MHz indicates that the filter's bandwidth needs to be adjusted.

The listing below is a plot of the Mix data output, which produces mixing indices vectors. Highlighted are the coefficients of those frequency components whose product resulted in 6.030 MHz (0 * 1880 + 1 * 94 - 1 * 88 + 1 * 0.03 + 0 * 1960 = 6.03 MHz).

freq	Mix				
	Mix(1)	Mix(2)	Mix(3)	Mix(4)	Mix(5)
0.0000 Hz	0	0	0	0	0
30.00kHz	0	0	0	-1	1
60.00kHz	0	0	0	-2	2
5.970MHz	0	-1	1	2	-1
6.000MHz	0	-1	1	1	0
6.030MHz	0	-1	1	0	1
6.060MHz	0	-1	1	-1	2
7.940MHz	-1	1	0	1	-2
7.970MHz	-1	1	0	0	-1
8.000MHz	-1	1	0	-1	0
8.030MHz	-1	1	0	-2	1
13.97MHz	-1	0	1	1	-1

The listing below, of the four equations on the schematic, shows the relationships, in dB, of the various products. Note that the difference between the carrier and the lower intermodulation product (the second and first columns, respectively) is approximately 26 dB.

Advanced Design System 2008

C IM3 Lower	C IM3 Upper	Out IP3 Lower	Out IP3 Upper
45.349	47.319	3.043	2.647

The fundamental of the product in the first column is defined by

$$0 * 94 + 1 * 94 - 1 * 88 + 0 * 0.03 + 0 * 1960 = 6 \text{ MHz}$$

and its intermodulation product by

$$0 * 1880 + 1 * 94 - 1 * 88 - 1 * 0.03 + 0 * 1960 = 5.97 \text{ MHz.}$$

The fundamental of the product in the second column is defined by

$$0 * 1880 + 1 * 94 - 1 * 88 + 1 * 0.03 + 0 * 1960 = 6.03 \text{ MHz}$$

and its intermodulation product by

$$0 * 1880 + 1 * 94 - 1 * 88 + 2 * 0.03 + 0 * 1960 = 6.06.$$

Obtaining Budget Incident Power and Gain

The following figure illustrates the use of the bud_gain, bud_snr and bud_pwr_inc measurement functions.

Note
The design IQ_mod_bud.dsn is in the Examples directory under Com_Sys/MultiChan_NL_Budget_prj. The results are in IQ_BudgetSchematic.dds and IQ_Budget.dds.

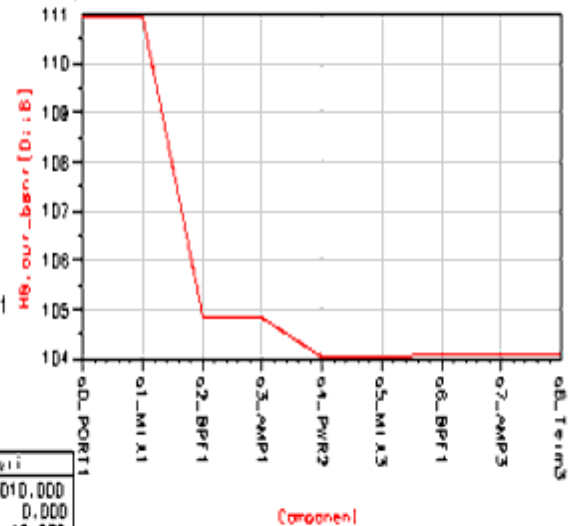
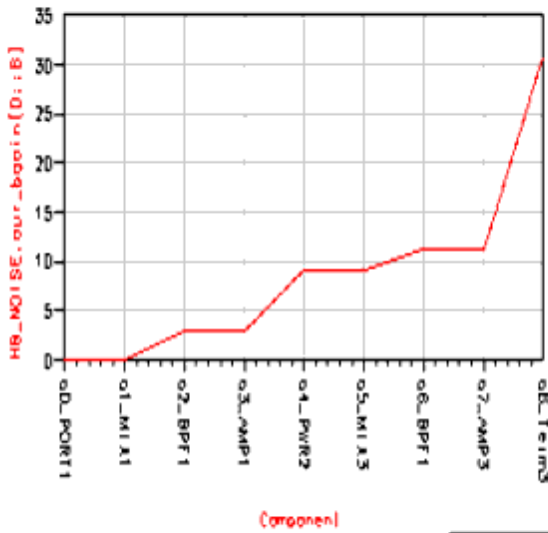
It is similar to the preceding example, with the exceptions that only a single tone is inserted at the input, and the transmitter tone has been replaced by a terminating resistor.

Component	HB_out_bfreq	HB_out_bgain
a0_PORT1	1.000E8	8.843E-18
a1_M1X1	1.000E8	8.843E-18
a2_BPF1	1.100E7	3.027
a3_AMP1	1.100E7	2.871
a4_PWR2	1.100E7	8.222
a5_M1X3	1.100E7	8.222
a6_BPF1	8.010E8	11.222
a7_AMP3	8.010E8	11.222
a8_Term3	8.010E8	30.708

Gain from Port 1 to the Input of each component in the "a" path

Component	HB_NOISE_out_bsnr
a0_PORT1	110.865
a1_M1X1	110.865
a2_BPF1	104.824
a3_AMP1	104.824
a4_PWR2	104.048
a5_M1X3	104.048
a6_BPF1	104.113
a7_AMP3	104.113
a8_Term3	104.087
b0_PORT2	110.865
b1_M1X2	110.865
b2_BPF2	104.824
b3_AMP2	104.824

Signal-to-Noise Ratio from Port 1 to the Input of each component in the "a" path



Note imbalance of power combiner input due to nonlinearity of b3_AMP2. (Data scrolled to top)

Incident Power Budget

Component	HB_out_bpow
a0_PORT1.11	-3010.000
a1_M1X1.11	0.000
a1_M1X1.12	-18.832
a1_M1X1.13	-2.041
a2_BPF1.11	3.000
a2_BPF1.12	-3010.000
a3_AMP1.11	2.871
a3_AMP1.12	-88.660
a4_PWR2.11	-315.112
a4_PWR2.12	11.340
a4_PWR2.13	12.871
a5_M1X3.11	8.222
a5_M1X3.12	-48.704
a5_M1X3.13	3.878
a6_BPF1.11	11.222
a6_BPF1.12	-315.112
a7_AMP3.11	11.222
a7_AMP3.12	-3010.000
a8_Term3.11	30.708

The first listing shows the fundamental frequencies for plan 1 and the power gain from the input port to pin 1 of each component. The second listing (top-left corner) shows the signal-to-noise ratio at pin 1 of each component. The last listing (bottom) shows the incident power at the input of each component at the fundamental frequencies in plan 1 through the system. In this example, note the imbalance at the power combiner input due to the nonlinearity of b3_AMP2.


Note
Certain measurements (such as bud_gain), although they derive data for a single tone, will output results for

all harmonics.


The Data Display IO_Budget shows an alternative way of using the budget functions. The budget measurements are performed within the Data Display window by adding equations that operate on the current and voltage data provided by the HB simulation.

Obtaining Group Delay Data

The following figure illustrates the use of the S-parameter simulator in obtaining group delay data. For more information about group delay, refer to the topics [Calculating Group Delay](#) and [Group Delay](#).

 **Note**
The design Linear_Sweep.dsn is in the Examples directory under Com_Sys/RF_Sys_prj. The results are in Linear_Sweep.dds.

In the S-parameter Simulation component a linear sweep centers closely around the center frequency of the first filter. Under the Parameters tab, the Group delay option has been selected. In addition, Enable AC Frequency Conversion has been selected, and S-parameter freq. conv. port has been set to 1, the input port.

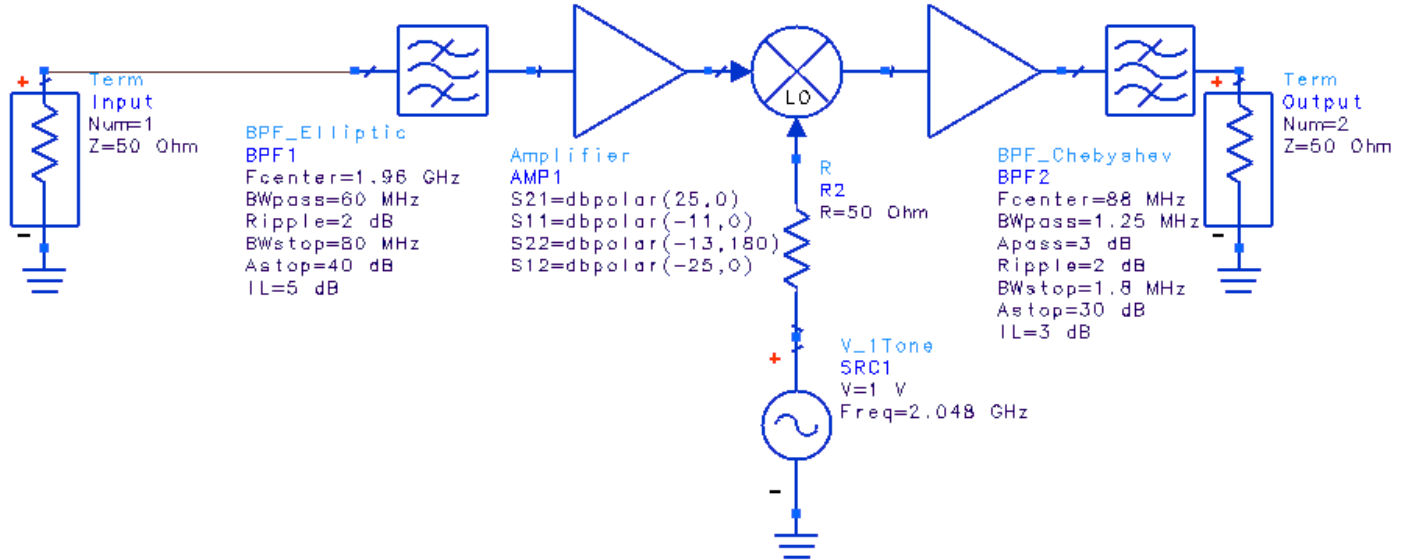
 **Note**
The frequency conversion port must be the number of the input port.

S PARAMETERS

```
S_Param
SP1
Start=1.958 GHz
Stop=1.962 GHz
Step=0.002 MHz
CalcGroupDelay=yes
GroupDelayAperture=1e-4
FreqConversion=yes
FreqConversionPort=1
```

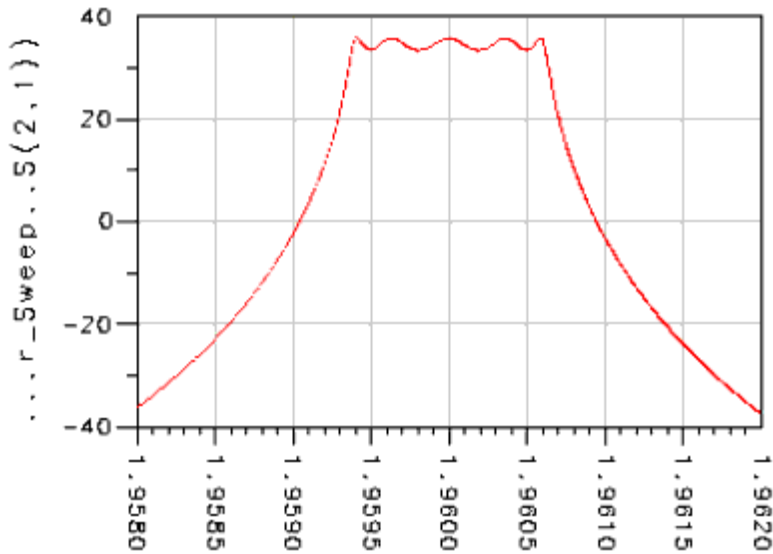
```
Mixer
MIX1
SideBand=LOWER
ConvGain=dbpolar(-5,0)
S11=dbpolar(-10,0)
S22=dbpolar(-10,180)
NF=5.5

Amplifier
AMP2
S21=dbpolar(25,0)
S11=dbpolar(-15,0)
S22=dbpolar(-13,180)
S12=dbpolar(-20,0)
```

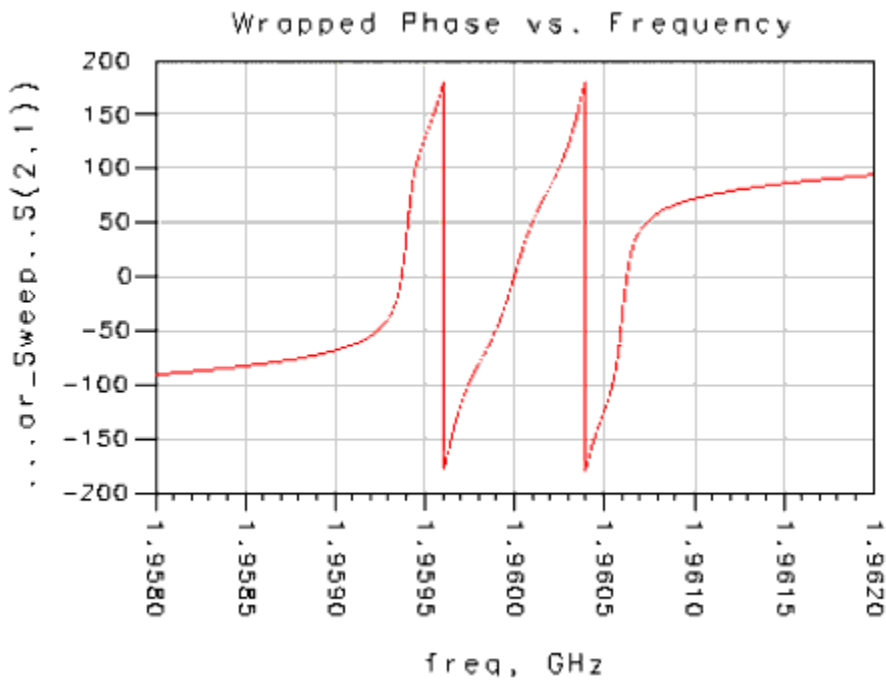


Using the S-parameter simulator to obtain group delay data

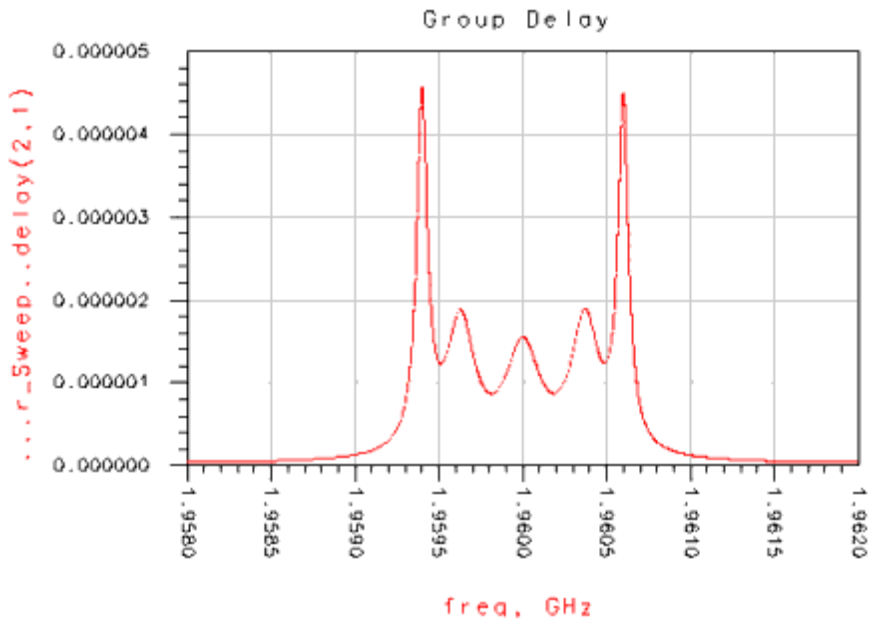
The following is a plot, in dB, of $S(2,1)$, showing the response of BPF2.



Below, a plot of the function phase(S(2,1)) shows wrapped phase versus frequency. This is essentially a compressed view of the response, as the display is normalized to accommodate the ± 180 -degree variations without unnecessary repetition.



Finally, a plot of the function delay(2,1) (the delay at the output port with respect to the signal at the input port) shows the shift in phase in the filter's passband (below).



Budget Analysis Capabilities

The topics in this section provide details about specific areas of the budget analysis feature, including additional instructions about setting up budget measurements.

The existing budget analysis is a collection of 14 predefined measurement functions intended to provide insight into the propagation of circuit characteristics along a set of selected locations throughout a circuit. These measurements are implemented as AEL functions. The following table lists the currently available budget functions with a summary of their parameters and usage.

These AEL functions are available for use from two locations. In the Schematic window you can insert templates for them from a simulation palette. In the Data Display window you must manually enter and edit the functions. Using the Functions Help button in the Equations dialog box helps with pasting a template from the documentation.

Budget Measurement Functions

Function	# of Forms	Used in		# of Vars	Source and Zs in 3 Vars††	Freq Plan #	Pin #	Zref	SimInst
		AC	HB†				Budget Path		
Utility									
bud_freq	2	X	X	3		X	X		X

Advanced Design System 2008

Power Gain									
bud_gain	2	X	X	6	X	X	X		X
bud_gain_comp	2		XS	7	X	X	X		X
Intercept									
bud_ip3_deg	1		XL	5				X	
Noise									
bud_nf	2	X		7	X		X		X
bud_tn	2	X		7	X		X		X
bud_nf_deg	2	X		8	X				
bud_noise_pwr	1	X	X	4		X	X	X	X
bud_snr	1	X	X	3		X	X		X
Basic									
bud_gamma	1	X	X	4		X	X	X	X
bud_pwr††	1	X	X	3		X	X		X
bud_pwr_inc	1	X	X	4		X	X	X	X
bud_pwr_refl	1	X	X	4		X	X	X	X
bud_vswr	1	X	X	4		X	X	X	X
† "S" denotes the use of a sweep. "L" denotes the use of BudLinearization. †† Functions include three variables: either vIn, iIn, Zs, or SourceName, SrcIndx, Zs. ††† Available only in the Data Display window.									

Budget Analysis calculations are based on both voltages and currents

Advanced Design System 2008

The internal evaluation of budget functions such as bud_gain is based on both voltages and currents, and thus is correct. This may not be obvious when setting up the budget analysis tasks.

Budget Analysis setup requires close attention

Setting up budget analysis measurements requires close attention to the following areas:

- The AEL function templates are not always complete and you may need to focus on their arguments to achieve proper use of those functions.
- Contextual meaning of the multi-purpose syntax and the effect on function arguments.
- There is more than one place where the same parameter or flag must be set simultaneously so the budget calculations can be invoked, or properly carried out.

Budget measurements work with only AC or HB simulations

The AC and HB simulations are typical for many traditional budget measurements. The budget analysis methodology is not extended to other types of analyses, and should be used only with AC and HB simulations.

Budget measurements may not work with both AC and HB simulations

The table above shows the currently available budget measurements. Of the 14 functions, all but one (bud_pwr) are available in the Schematic window from the AC or HB palette, or from both. All 14 functions are available in the Data Display window. It is important to verify which simulator a particular function can work with. A common misconception is, for example, that the bud_nf function works with Harmonic Balance. Though that function is not available on the HB simulation palette, it can be added to the Data Display window.

Different AC and HB usage of functions that can work with both simulations

Functions that are designed to work with both AC and HB simulations, such as bud_gain, require different handling of their input arguments for those different simulations. When editing AEL function arguments, it is important that you verify the syntax for the desired simulation; for details see [Measurement Expressions](#).

HB Budget Analysis needs a frequency plan

When different frequencies are present at a specific circuit location, such as at the output of a mixer with both upper and lower side bands, the budget measurements are organized according to frequency plans. A frequency plan specifies the measurement frequency for each location. The plan number is a required input in harmonic balance budget analysis.

Frequency plans are internally generated

Frequency plans determine what frequencies are to be monitored at various locations. ADS tries to determine frequency plans for the user; however, for some circuits, ADS cannot generate a frequency plan. In these situations, ADS displays an error message that a plan cannot be generated which interrupts/disables any budget calculations.

For example, a feedback loop over a mixer in circuits such as AGC leads to conflicting frequency values for the mixer output. As a result the frequency plan is not generated, and, consequently, no budget measurements are available.

In cases when the frequency plans are generated, you may still face a problem since it is not known up-front what frequencies are present in a specific plan. That information becomes available after the simulation is completed. Therefore, if an incorrect plan had been selected in the schematic, the circuit may have to be re-simulated. For more information about frequency plans, see [Budget Measurement Analysis](#).


Common issues about Budget Noise Figure

The Budget Noise Figure function (`bud_nf`) is one of the most popular budget measurements. Though the `bud_nf` function is formally available only for AC simulations, it may still, however, appear as functional for HB simulations. You must carefully interpret any results. For example, there is no argument provided for the frequency plan. As a result the calculations are based on default values. In general, the results reported by `bud_nf` may not be meaningful, and you should not use `bud_nf` for HB simulations.

The second expectation is that the function would calculate partial noise figures. The term partial would mean that, for a specific circuit location, the function would return the standard NF of the two-port defined between the source and the location, or the two-port defined between the location and the output. In order to carry out such calculations, ADS would have to break the connections and create the corresponding two-port. Such an interpretation might be possible for cascaded circuits; ADS, however, is a general circuit topology tool and no such assumption can be made.

Budget Noise Figure is really a reformulated signal-to-noise ratio

The `bud_nf` function calculates quantities that could be termed as internal noise figures. What is presented as a noise figure at a location is actually the signal-to-noise ratio decrement from the location to the output that is normalized with respect to (subtracted from) the overall (from the input to the output) noise figure.

 Example

Consider a circuit with four components selected for budget path, with the last one being the load. If the function `bud_snr` reported 60, 54, 50, and 45 signal-to-noise ratios and if the overall $NF = 17$, then the function `bud_nf` would report 2, 8, 12, and 17 (all in dB).

As such, the function `bud_nf` does not provide any additional useful information with respect to the function `bud_snr`.

The calculated signal-to-noise ratio for an internal location is an in-circuit measurement. It combines the noise contributions coming from noise sources in the entire circuit, for example in a cascade from both to the left and to the right. Therefore, such internal noise figures do not represent the standard notion of NF, that of noise added by a stage.

Using the `pinNumber` argument in budget functions

The `pinNumber` argument can assume one of the following values:

1	selects pin 1 of the component (all components)
2	selects all the pins of the component (all components)
<code>budget_path_name</code>	typically selects pin 2 (pin 1 for the source or the Term components) and only path components are reported

The default value of 1 leads to a budget measurement reporting scheme that is not obvious, and requires closer examination. A measurement, such as gain, reported for a component refers to signals before the component. As such, it may not include the full contribution of that component. For example, if that component is an amplifier whose input impedance is infinite, the corresponding current (and power) at pin 1 is zero. If the results were reported for pin 2 then the amplifier gain would be included in the budget power gain reported for that amplifier, but for pin 1 it is not. This is correct operation.

Using the `SrcIndx` argument in budget functions

The `SrcIndx` is described as "the frequency index that corresponds to the source frequency to determine which frequency to use from a multi-tone source as the reference signal". You may find it difficult to figure out how to determine a proper value for that argument.

The actual meaning of this argument is the index of the desired input (reference) frequency as determined internally by ADS: all spectral components (harmonics and intermodulation products) are ordered starting from DC to the highest frequency. The DC component as the first one corresponds to zero index.

The following example explains the situation. Let one of the fundamental source frequencies be named `Upper_Freq` in a VAR block and assume a value of 1.98805 GHz. We want the budget gain to be calculated with respect to that input

frequency. If, after simulation, we display the array `freq` in the Data Display window and we find the frequency in question as the 18th entry, then the index, counted from 0, is 17. Thus, we can define the `bud_gain` function with `SrcIndex = 17`, as

```
BG = bud_gain("PORT1",17,....)
```

If we wanted to define the function up-front in the Schematic window, we would not know that value. A solution to such a problem is to use the `find_index` function as

```
BG = bud_gain("PORT1", find_index (freq, 1.98805e9),....)
```

```
BG = bud_gain("PORT1", find_index (freq, Upper_Freq),....)
```

The second approach could be more useful if a change to the actual value is possible. However, an even more flexible way is to indicate which fundamental is of interest as in the following scheme

```
BG = bud_gain("PORT1", find_index (freq, indep(mix(freq, {0, 1, 0}))),....)
```

But, even this approach is not general enough. For instance, it may become invalid or incorrect if the fundamental frequencies in the HB controller are rearranged.

Budget Path - how to use it effectively

Budget path is generated automatically by choosing `Simulate > Generate Budget Path`, then selecting the input and the output ports (components). The name of the function thus generated can then be specified in place of the `pinNumber` variable (, , , `budget_path_name`, , ,) - if one exists.

For a few of the budget functions there exists an undocumented feature of passing the budget path name via the `SourceName` argument, which is, for example, the first argument in the `bud_gain` function. This is not a recommended way of using budget path.

Finally, you can deal with any insufficiencies of the automatically generated budget path by directly editing the budget path measurement equation. The terminal numbers can be changed, components can be dropped or added, as desired, as long as legitimate component instance names at the highest level of the hierarchy are used. Also, more than one budget path, each with a different `budget_path_name`, can be defined using the Copy feature in the Schematic window.

Mixer2 and MixerIMT2 components issues

Do not use these components in conjunction with HB budget analysis.

These two components are SDD-based. Thus, the frequency plan generation process in budget HB has no knowledge that a frequency conversion takes place in either of the two components. This leads to a conflict, and budget calculations are not carried out. The two explicit mixer components Mixer and MixerIMT will not cause the frequency plan generation to fail. Please note that MixerIMT is not available from the palette, but can still be inserted into the circuit by typing its name in the Component History field.

Clipped values in bud_nf and bud_tn

Under some circumstances, the software clips the values returned by bud_nf to 0, and by bud_tn to 290 K. This is usually set for the input port, regardless of the value of the reverse signal to noise ratio. See [Budget Noise Figure is really a reformulated signal-to-noise ratio](#). This may be misleading since it creates an impression of noiseless stages in situations when the designer knows they are not.

Contextual meaning of input parameters: Beware of short syntax form

Some functions are documented with a long and a short syntax form to choose between. You must be very careful when using either form.

The differences between the forms are not only in the meaning of individual parameters, but also in the type of the values entered: strings, real or integer constants, and whether they are enclosed in the quotation marks or not. In general, instance names are strings surrounded by quotes, while variable names are strings entered without quotes.

Most of the functions can be formulated with a truncated set of arguments. The arguments that are not listed will assume default values. For example, for bud_gain, the set of six arguments can be truncated as much as necessary that still leaves any required arguments, which is just the first one in AC, or the first four in HB. Furthermore, if an input parameter such as the frequency plan is required, as it is in bud_gain in HB, the short syntax form does not mean a change in the location of that parameter. If necessary, empty (default) parameters must be entered by means of commas, as in the following example

```
BG = bud_gain("PORT1",17,,1)
```

Additionally, the interpretation of some arguments may be contextual. This is, for example, the case in bud_gain where the user can enter either

vIn	the name of the dataset variable for source voltage
-----	---

iIn	the name of the dataset variable for source current
-----	---

or

SourceName	the instance name of the source, in quotes
SrcIndex	the index of the source frequency (see above)

Take special note of the fact that for `bud_gain` in AC the `SrcIndex` is irrelevant, so that the short syntax form can actually consist of just one parameter `SourceName`. However, if `vIn` and `iIn` are used, both are needed.

Nevertheless, if `SourceName` is used, and you want to use the budget path, then the correct argument count must be preserved, as in the following example

```
BG = bud_gain("PORT1",,,,budget_path_name)
```

Finally, for the `bud_nf_deg` function the short syntax form is indeed different from the regular form: the output port instance name (in quotes) takes the place of the second argument, and the output node name (also in quotes) takes place of the third argument.

Using IMT-Based Mixer Models in Spurious Signal Analysis

Signal generation or frequency translation achieved within non-linear elements such as mixers, non-linear amplifiers, and spectrally impure oscillators results in the creation of spurious signals that can be polynomially related to the fundamental tones involved in the primary frequency conversion process. Spectral response of components that generate such spurious tones are represented as Spur Charts in the RFIC industry. Within ADS, there are three types of intermodulation table (IMT) formats which contain equivalent descriptions for use with behavioral mixer models such as `MixIMT_Data` and `MixerIMT2`. The use of IM tables along with the conversion gain parameter on mixer models is illustrated in this section.

As the name suggests, IM tables are two-dimensional matrices containing information about strengths of spurious tones or intermodulation products for specified reference values of signal and oscillator powers. By convention, each n th column contains mixing products generated by the $(n-1)$ th harmonic of the local oscillator (LO) and the m th row contains mixing products generated by the $(m-1)$ th harmonic of the signal (RF). Simple single-side banded IM tables for single-RF and single-LO mixing start with the DC term (where m and n are both zero) and the value of active RF harmonic implied by the position of the row. More complicated double-side banded tables for multi-RF single-LO mixing require both positive and negative specification of the values of various RF harmonics in the first few columns to distinguish between sum and difference tones.

Interpreting IMT Charts

Advanced Design System 2008

The simplest of IM tables, corresponding to Spur Chart specification of mixer hardware is the O-type IMT file. As shown in the following code example, it contains specification of reference signal power PRF = -10 dBm and oscillator power PLO = +7 dBm. In this specific table, all entries are non-negative and the IF fundamental power value is zero. In order for it to be representative of a physical system, this table should be interpreted as a relative spur chart where a value of x at location (m, n) indicates that the spurs at $(m-1)*FRF + (n-1)*FLO$ and at $|(m-1)*FRF - (n-1)*FLO|$ are both x dB below the value of the IF fundamental (at m=n=1). For O-type IM tables only, missing elements are assumed to be x=99, that is 99 dB lower in power than the IF fundamental. The first element of the first column indicates that the DC spur is 79 dB below the IF fundamental. Likewise the element at (3,2) indicates that $2*FRF + FLO$ and $|2*FRF - FLO|$ are both 84 dB below the IF fundamental.

```
BEGIN IMTDATA
# IMT ( -10 7 )
%   0   1   2   3   4   5
    79  56  67  74  72  83
    24   0  68  69  92
    72  84  56  63  90
END IMTDATA
```

O-type IMT data for second degree RF nonlinearity mixed with fifth degree LO nonlinearity at PRF = -10 dBm and PLO = +7 dBm

Additional information can be appended to the structure shown above by including complex spur information where both strength and phase are specified for each table entry. Also, sum and difference tone behaviors may be distinguished by including an RF-side multiplier term as shown for the dual-RF and single-LO IM table below. All elements must be specified in (dBm,degree) pairs in a double-side banded table in A- and B-type IMT files. The B-type table shown in the following code example also contains exact RF and LO frequencies used on some mixer subcircuit to generate the following multi-RF spur chart. Notice in the table that each RF tone has positive and negative mixing indices enabling the registration of separate entries for sum and difference tones.

```
BEGIN IMTDATA
! Frequency units of GHz, absolute spur values in \ (dBm, degree
! Reference resistance = port impedance = 50 Ohms
# IMT ( GHZ S DBM R 50.0 )
! The RF fundamentals are at 2.0 GHz and 1.4 GHz.
%   FRF1   FRF2
    2.0    1.4
! The LO fundamental is at 1.7 GHz.
%   FLO
    1.7
! The reference powers of the two RF tones are -10 and -15 dBm
! respectively.
%   PRF1           PRF2
    -10            -15
! The reference powers of the LO tones is \+7 dBm.
%   PLO
    7
! The first two columns indicate mixing multipliers on RF side.
! The last 3 columns indicate mixing multipliers on LO side.
%   M1   M2   0           1           2           3
    -1   -1   -49   39   -29   -49   -52   -76   -32   -76
    -1   0   -64   -73   -59   23   -84   -15   -44   21
```

-1	1	-43	22	-76	23	-55	63	-48	22
0	-1	-33	39	-52	-76	-74	29	-49	12
0	0	-79	0	-39	-33	-83	72	-98	6
0	1	-33	-39	-53	-55	-63	-48	-42	-10
1	-1	-43	-22	-43	-42	-76	-23	-55	-63
1	0	-64	73	-55	-39	-33	-83	72	-98
1	1	-49	-39	-49	39	-29	-49	-52	-76
END IMTDATA									

B-type IM table showing complex spur information for dual RF and single LO mixing

Note that when double-side banded IM tables are presented along with explicit frequency information, spurs involving differences of one or more tones should always be interpreted based on the positive value of IF spectral frequency. In the example immediately above, $-1*FRF1 + 0*FRF2 + 1*FLO$ is numerically -300 MHz. In the table, the (-1 0, 1) entry is -59 dBm, 23 degrees. The measurable physical manifestation of this spur is at +300 MHz with the same strength of -59 dBm but phase inverted to -23 degrees. Also, note how such a multi-RF IM table allows separate classification of colliding tones at the same IF frequency. The two RF fundamentals captured to the above table are mutual image frequencies with respect to the LO frequency. The two difference fundamentals of the IF spectrum both occur at 300 MHz. However, the one due to $| -FRF1 + FLO |$ is captured at the location (-1 0, 1) whereas the one due to $-FRF2 + FLO$ is captured at (0 -1, 1). The former should be conjugated to derive physical value. Such accuracy of capturing colliding tones can be achieved in the ADS Simulation environment. See the design BehavioralModels > MixIMT_prj > CKT_IMTB_extraction.dsn in the Examples directory for details.

For details of various IMT formats, see [IMT Format](#).

Interpolation of IMT Data

Since each data point contained within IM tables is not only relevant for the reference power and frequencies, they are also related to each other across the mixing grid, making interpolation and extrapolation of IM tables across signal powers and frequencies a non-trivial proposition. This situation is further complicated by the movement of high-side LO to low-side LO as one increases RF frequency or lowers LO frequency or performs a combination of both. Add to that the issue of intermodulation products that collide at the same spectral point for certain combinations of input frequencies.

For O-type IM tables, where no explicit frequency information is available, no frequency domain interpolation is performed by data models such as MixIMT_Data. Power domain interpolation is done by scaling IM table element up in proportion to its mixing multipliers as demonstrated below:

Q1. Given an IF spur value of x at location (m,n) of an O-type table for reference powers of PRF dBm and PLO dBm, what is the projected value of the (m,n) spur at PRF' and PLO'?

A1. Assuming that the mixer is operating in saturation where roll-off of the strength of the IF fundamental is 1 dB for 1 dB increase in signal or LO power, the (m,n) position of the O-type table corresponding to the IF tones $(m-1)*FRF + (n-1)*FLO$ and $|(m-1)*FRF - (n-1)*FLO|$ will undergo a change of y dB where $y = (|m|-1)*(PRF'-PRF) + (|n|-1)*(PLO'-PLO)$. The interpolated value of the spur at (m, n), will then be $x + y$. This behavior is exhibited by MixIMT_Data for all variation of RF and LO powers. For earlier models, e.g. MixerIMT2, it is the responsibility of the user to maintain $PRF < PRF' < PRF + 3$ dBm and $PLO -10 < PLO' < PLO + 3$ dBm to obtain comparable results.

For A- and B-type IM tables, where explicit frequency domain data is available and multiplicity of IM tables is allowed within the same data file, some amount of linear interpolation across corresponding mixing products may be allowed for very small deviations along FRFn and FLO axes, where relative spacing of RF and LO tones remain unchanged and no LO-crossover occurs. For values of FRFn and FLO beyond those on file, it is safe to enforce the nearest neighboring IM spur chart. Note that frequency domain interpolation of IF strengths across (m,n) th location of one IM table and the exact same location of another may not be an accurate representation of circuit level behavior but the values obtained are rarely non-physical. See example design BehavioralModels > MixIMT_prj > BEH_IMT_2R_4L.dsn for an interpretation of frequency domain interpolation using the MixIMT_Data component. Note that MixerIMT and MixerIMT2 components cannot operate on A- and B-type IMT data.

Power domain interpolation of A- and B-type IM data is done in a manner similar to that of O-type files explained above.

Modeling IMT with Conversion Gain

Mixer models that rely on IM tables often possess an additional parameter called ConvGain or conversion gain. This parameter specifies voltage gain from RF port to IF port and affects the entire IF spectrum without having any direct impact on RF or LO spectra. Internally, the voltage value sensed at RF port is amplified by the complex conversion gain and the amplified RF value used to scale the IF spectrum as specified for IM power domain interpolation. Thus, given $ConvGain=dbpolar(a,b)$, the MixIMT_Data model generates $(m-1)*a$ more power at all IM tones of the m th row of the IM table. It simultaneously adds a phase lead of $(m-1)*b$ to that IF vector. See example design BehavioralModels > MixIMT_prj > BEH_IMT_2R_4L.dsn for an interpretation of frequency domain interpolation using the MixIMT_Data component.

System Noise Analysis

There are three major contributors to system noise: passive-element thermal noise, active-element noise, and oscillator phase noise. (For a discussion of oscillator phase noise, see [Harmonic Balance for Oscillator Simulation](#).) The system noise response is simulated by the program under small-signal conditions. A linear analysis of system noise gives a reasonable representation even when the signal is well into compression, provided the signal-to-noise ratio is not too low.

System thermal and active noise simulation uses a noise-wave model that accounts for the effects of element mismatches.

System phase noise is described as the phase-noise level versus the oscillator offset frequency. System phase noise is simulated by sequentially combining the phase-noise characteristics of consecutive oscillators from the system input port to the system output port.

The combined active noise, thermal noise, and phase noise at the system outputs can be observed in terms of noise power in dBm versus frequency. This total noise power can be combined with the output signal so that the system output signals can be observed with the system noise added.

Note

In a noise analysis the following assumptions are made with respect to the terminating resistances that are implicitly connected to the system network's input and output ports:

- The signal source (assumed to be connected to the system network input, or Port 1) is assumed to have a resistance of 50 ohms at a standard physical temperature, T_0 , of 290 K. This source resistance provides noise power at a noise density of -174 dBm/Hz into the system. This source must be a power source.
- The system network outputs (any port other than Port 1) are assumed to be terminated with 50 ohms at absolute zero physical temperature, 0 K. This termination resistance does not contribute any noise to the noise measured from the system. This termination must be a Term component.

When a noise analysis is requested, it is recommended that you use an Options component and set the global temperature to 16.85°C.

Parameter Sweeps and Sweep Plans

Generally, sweeps of individual parameters can be performed most efficiently from within many of the simulator dialog boxes themselves. The ability to step through a series of values automatically is incorporated into all the standard simulation controllers. Sweeps can be performed at both the circuit and the system level.

The following parameters are typical candidates for sweeping:

- Signal frequency, amplitude, or power
- Bias voltage or current
- Resistance
- Signal path attenuation

- Impedance
- Ambient temperature
- Most component parameters

However, it is possible to combine sweeps of several parameters into a hierarchical sweep plan. By using parameter sweeps, you can do the following:

- Find the bias voltage that yields the best mixer conversion gain.
- Find the load impedance that yields the lowest harmonic distortion.
- Simulate a load-pull measurement.
- Simulate the effects of process variations and temperature on circuit performance.

By selecting the ParamSweep and SweepPlan controllers from any of the simulator palettes, you can sweep a variety of parameters and construct a series of sweep plans for special purposes. See the sections [SweepPlan Controller](#) and [Parameter Sweep Controller](#).

Conducting Sweeps

Sweeps of frequencies can be conducted from within many of the simulation controllers themselves, using options available under the Sweep tab where applicable. To sweep a parameter such as power, for example, select a ParamSweep controller from a simulator palette and edit it. It is necessary to ensure that frequency and power variables to be swept are defined appropriately in a source component, such as a P_1Tone component (available in the Sources-Freq Domain library).

If using the Load Sharing Facility (LSF) utility, you can break up a sweep and run the simulation on multiple machines, in parallel, by selecting Parallel Hosts as the Simulation Mode (Simulate > Simulation Setup). Individual sweep points are run on each machine and results combined into a single dataset on the local machine. For details on setting up remote and local machines for remote processing, see "Using Remote Simulation" in the installation documentation for your platform:

- [Windows Installation](#)
- [UNIX and Linux Installation](#)

Using Options under the Simulator Sweep Tab

A variety of simulation options enable you to conduct a simulation for only a single parameter at a single value, or to sweep a parameter over a defined range, either linear or logarithmic. They also enable you to select a named sweep plan that you can define.

Select the Sweep tab in various simulation controllers to do the following:

Advanced Design System 2008

- Define a parameter to sweep (can be a variable or component parameter)
- Select a sweep type
- Select start, stop, and step sizes
- Select a sweep plan

A parameter entered into the Parameter to sweep field will appear on the schematic in quotes. To display a parameter to sweep so you can edit it directly on the schematic, do the following:

1. Select the Display tab.
2. Select SweepVar.
3. Select Display parameter on schematic, then click OK.


You can then define that parameter directly on the schematic, taking care to place the definition in double quotes.

Using Sweep Controllers

In addition to the sweeps that are provided within various simulators, ParamSweep and SweepPlan controllers are available in each simulation palette. To use these controllers, place and edit them in a schematic as you would other controllers. You must also define the swept parameter in a VAR item. While both of these sweep controllers expand the sweep capability beyond what a simulator provides, they do have differences.

The ParamSweep can be used by itself to sweep a parameter that has been defined. This controller can sweep only in equal increments using single point, linear, or log sweeps. The ParamSweep must also identify the simulator to be used. If a design happens to use additional simulators, you can enter their names in the ParamSweep in the order they should be invoked.

The SweepPlan controller provides more flexibility. It supports more ranges, and a sweep plan can include a single point along with sweeps across a range of values. To use a SweepPlan controller that you have defined, it must be referenced by a ParamSweep or simulation controller. To reference a SweepPlan controller, select Use sweep plan on the ParamSweep or simulation controller, then enter the SweepPlan controller's name. Some simulation controllers do not include access to a SweepPlan. When a SweepPlan is used, its settings override any sweep setup information in a simulator or ParamSweep. Multiple SweepPlans can be chained together by selecting Next Sweep Plan on the SweepPlan controller.

 **Note**
The placement of sweep controllers within a circuit or system design does not affect the order in which parameters are swept. Similarly, the order in which the sweeps are automatically numbered does not determine the order in which they are executed. The order of execution is determined by the order in which one sweep calls another, as determined by the value of the parameter SweepPlan. The simulation or ParamSweep controller calls the first sweep plan to be conducted, whatever it is named.

Basic Procedures

This section presents the following example sweep scenarios:

- [Using ParamSweep to Sweep Two Parameters](#)
- [Using SweepPlans to Perform Fine and Coarse Sweeps](#)

Note
The appropriate simulators are required to run the following simulations, i.e. a Harmonic Balance simulation requires the Harmonic Balance simulator (included with all Circuit Design suites except the RF Designer suite.). You may build the examples with the appropriate license, you will simply be unable to run the simulations.

Using ParamSweep to Sweep Two Parameters

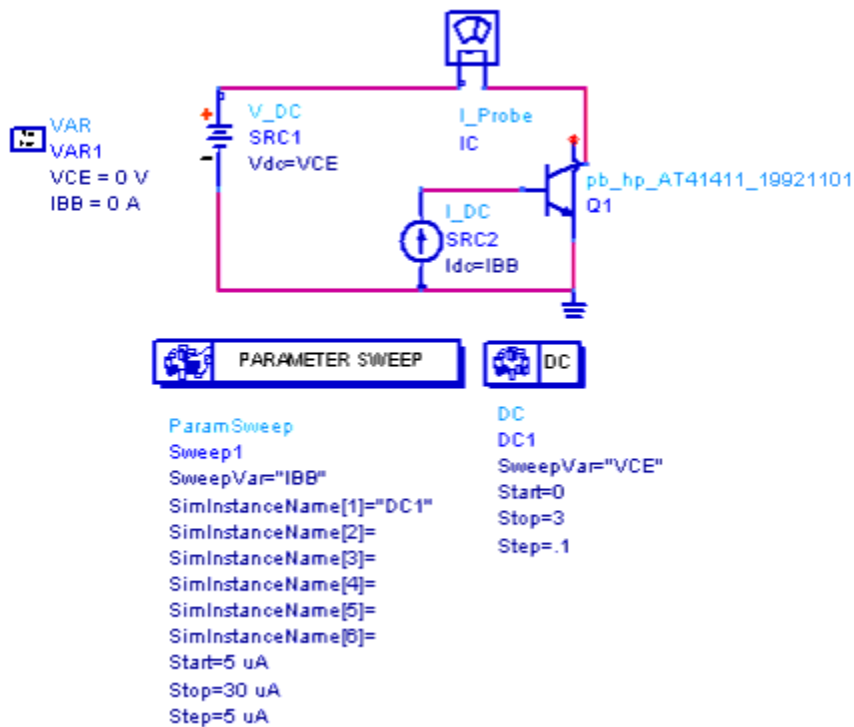
The next figure below illustrates an example setup that uses a ParamSweep controller (available in all the simulation palettes) to sweep two parameters. In this case, the result is a curve-tracer display of collector current versus V_{ce} for different values of I_{bb} .

Note
This design, Curve_Tracer.dsn, is in the Examples directory under MW_Ckts/LNA_prj. The results re in Curve_Tracer.dds.

Although it is not necessary for the current design, such a simulation can be used to characterize a device whose I-V relationships are unknown. SRC1 establishes collector-to-emitter voltage V_{ce} . SRC2 establishes base current I_{bb} . The current probe, Probe1, measures collector current I_{cc} . Note that Probe1 has been named I_{cc} and that I_{cc} , the current at this point, will be plotted later.

The procedure for using the ParamSweep controller, in conjunction with a simulator sweep and an equation, is outlined as follows.

Note
The following steps describe the design under discussion. Modify the details to suit your particular needs.



Example setup for a hierarchical sweep

To use the ParamSweep controller to sweep two parameters:

1. Use a VarEqn component (available from Component Palette List > Data Items > Var eqn) to define two variables—an "inner" and an "outer" variable. The inner variable is swept over its full range each time the outer variable is stepped.
2. Use a DC Simulation controller to define the parameter to sweep, the inner variable.
3. Use a ParamSweep controller to establish a sweep plan for the outer variable.

The following illustrates this procedure in detail.

Define Inner and Outer Variables

1. Edit the VarEqn component on the schematic.
2. In the Select Parameter field, ensure that the following variables are defined by an equation:
 - VCE = 0 V
 - IBB = 0 A

Note
The simulator requires that all variables be initialized. The above voltage values would be used if no sweep were in effect.


Advanced Design System 2008

These equations are written in the field on the right of the box, with Variable or Equation Entry Mode set to Name=Value.

3. Click OK.

Define the Inner Variable in the Simulation Controller

1. Edit the DC Simulation controller.
2. In the Parameter to sweep field, ensure that VCE is entered. VCE is the inner variable, which has been established in the Var/Eqn component.

 **Note**
Variables entered into this field will appear in quotes on the schematic. If you enter a variable directly on the schematic (in this case, as the right-hand side of the SweepVar statement in the DC controller), you must surround the variable with double quotes.

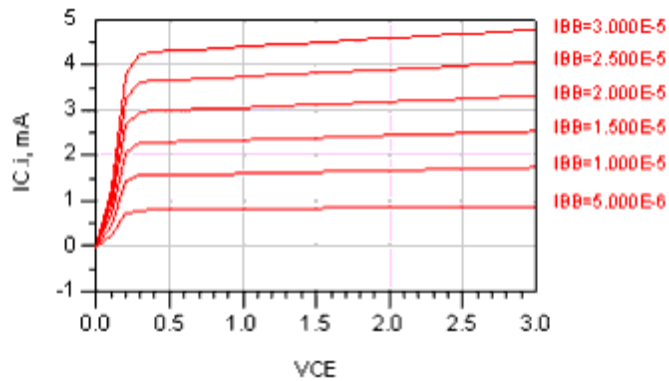
3. Ensure that the following parameters are set:
 - Sweep Type = Linear
 - Start/Stop is selected.
4. Click Apply, then OK.

Establish a Sweep Plan for the Outer Variable

1. Edit the ParamSweep controller.
2. Ensure that the following parameters are set, and make them visible on the schematic:
 - Parameter to sweep = IBB. This establishes Idc at SRC2.
 - Sweep Type = Linear
 - Start/Stop is selected
 - Start = 20 μ A
 - Stop = 100 μ A
 - Step-size = 10 μ A
3. Select the Simulations tab, and ensure that DC1 is entered in the Simulation 1 field.

Launch the Simulation and Display Data

1. Launch the simulation. The following is a plot of Icc.i:



Information such as that depicted above is useful in deciding whether a device is suitable for a given application where the limiting factors are available voltage and current.

Nesting Parameter Sweeps with Multiple Items

To nest parameter sweeps with multiple parameter sweep items, assign the instance name of the Parameter sweep item associated with the inner sweep to the SimInstance name [1] parameter of the Parameter sweep item associated with the outer sweep.

Using SweepPlans to Perform Fine and Coarse Sweeps

The next figure below illustrates an example setup for using two SweepPlan controllers (available in all the simulation palettes) to perform a fine sweep of noise figure versus RF frequency, followed by a coarse sweep.

Note
This design, SweptRF_NF.dsn, is in the Examples directory under RFIC/Mixers_prj. The results are in SweptRF_NF.dds.

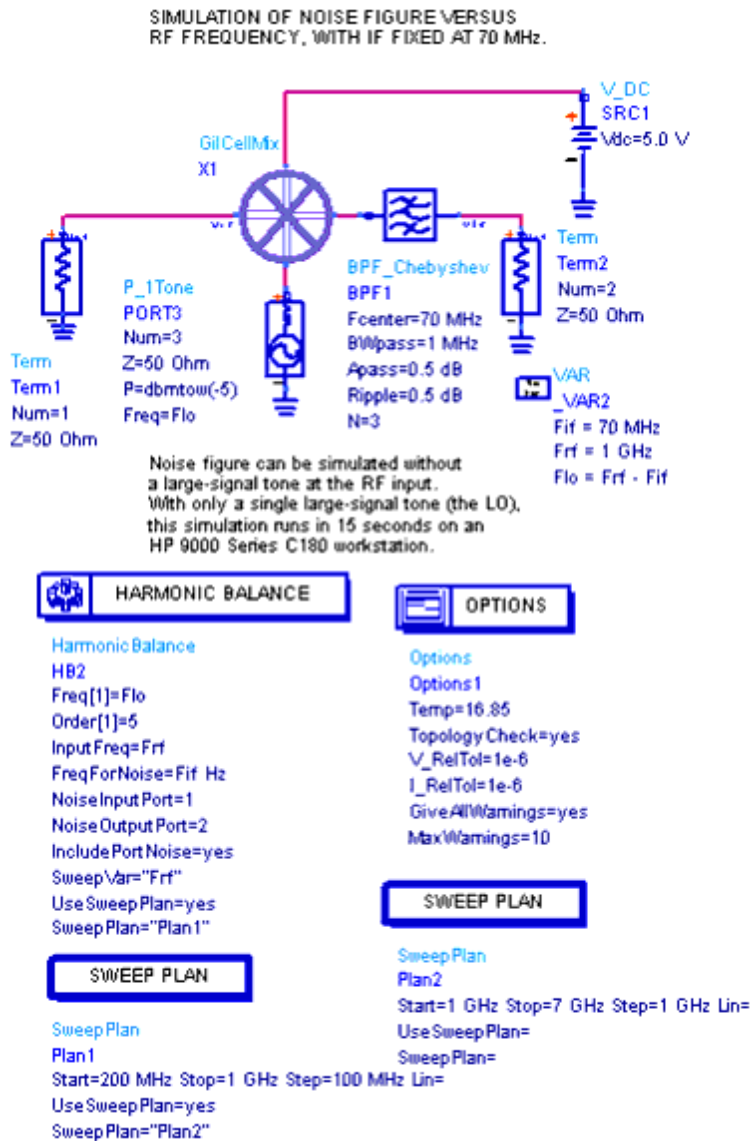
This simulation uses a Harmonic Balance Simulation controller (HB) to call a sweep plan, and that sweep plan, in turn, calls a second sweep plan. A simulation controller cannot call more than one sweep plan.

Note
The following steps describe the design under discussion. Alter the details to suit your particular needs.

To use two SweepPlan controllers to perform a fine and a coarse sweep:

1. Use a VarEqn component (available from Component Palette List > Data Items > Var eqn) to define all frequencies.

- Using two SweepPlan controllers (available in all the simulation palettes), establish two sweep plans—one for a fine sweep and one for a coarse sweep.
- Use a Harmonic Balance Simulation controller (HB) to define the parameter to sweep and reference a sweep plan.




Example setup for using two sweep plans

The following topics illustrate this procedure in detail.

Define Frequencies

1. Edit the VarEqn component on the schematic.
2. In the Select Parameter field, confirm that the following variables are defined by an equation:
 - F_{if} (intermediate frequency) = 70 MHz. This is the noise frequency defined in the Harmonic Balance Simulation controller (HB) (see below).
 - F_{rf} (RF input frequency) = 1 GHz. This is the input frequency defined in the Harmonic Balance Simulation controller (HB) (see below).
 - $F_{lo} = F_{rf} - F_{if}$. F_{lo} is the frequency assigned to the P_1Tone component at the LO input of the mixer.

 **Note**
The simulator requires that all variables be initialized. Although a value of 0 MHz or 0 GHz is sufficient to establish units for a local variable, leaving realistic units on the schematic allows a meaningful simulation to be conducted even when those variables are not swept.

These equations are written in the field on the right of the box, with Variable or Equation Entry Mode set to Name=Value.


3. Click Apply, then OK.

Establish Two Sweep Plans

1. From any simulation palette, select the SweepPlan controller and place two of these in the Schematic window.
2. Edit the first SweepPlan (Plan1) as follows:
 - Sweep Type = Linear
 - Start = 300 MHz
 - Stop = 1 GHz
 - Step = 350 MHz
 - Select Next Sweep Plan, and enter Plan2 (to be established in the next step) in the field below the option.
3. Edit the second SweepPlan (Plan2) as follows:
 - Sweep Type = Linear
 - Start = 1 GHz
 - Stop = 7 GHz
 - Step = 3 GHz
 - Because this is the last SweepPlan in this series, do not select Next Sweep Plan or enter a plan name.

Define Sweep Frequencies and Sweep Plan in the Simulation Controller

1. Under the Sweep tab of the Harmonic Balance Simulation controller (HB), establish the following:
 - Parameter to sweep = Frf

 **Note**
Variables entered into this field will appear in quotes on the schematic. If you enter a variable directly on the schematic (in this case, as the right-hand side of the SweepVar statement in the DC controller), you must surround the variable with double quotes.

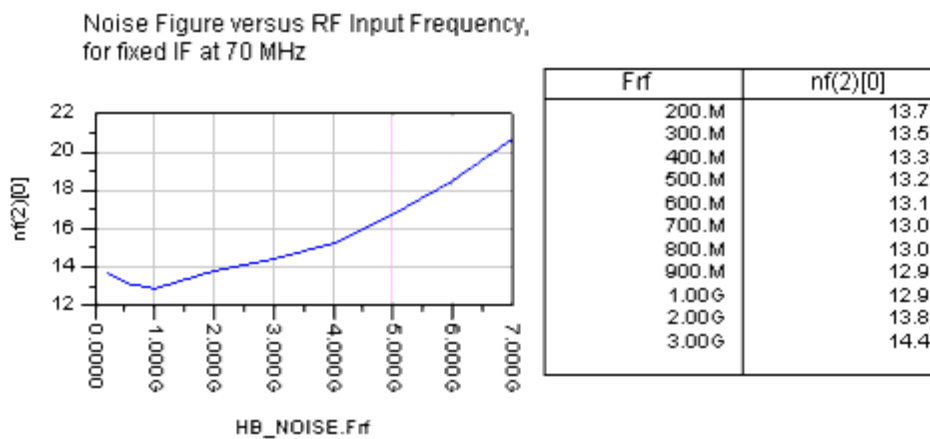
- Sweep Type = Linear

Advanced Design System 2008

- Use sweep plan is selected, and the entry in the field is Plan1. You can use the pull-down menu at the right of the field to select the name of a SweepPlan controller. Other sweep parameter entries will be overridden by values in the SweepPlan controller.
 - Finally, because this is a noise analysis, also select Nonlinear noise to enable the analysis.
2. Click Apply, then OK.

Launch the Simulation and Display Data

1. Launch the simulation. The following are Rectangular and List plots of noise figure:



Additional Examples

An additional example of a swept-parameter simulation is found in the examples directory, in RFIC/Mixers_prj/networks. IMDLOSwp.dsn uses a ParamSweep controller to sweep LO power in conjunction with harmonic balance simulation.

Recommendations and Tips

This section presents suggestions for using sweeps and improving the accuracy of results.

Ensuring that Sweep Results are Displayed Correctly

Parameter sweeps in the time domain are remarkable only in the way that they affect the adaptive time-step algorithm, which may have possible negative effects when results are displayed in the Data Display window. If the

Transient simulator is allowed to use its adaptive time-step algorithm and Max time step is not specified by the simulation controller, the simulator will probably produce results that have irregular data intervals. This does not matter when you are simulating only versus time and not sweeping – that is, when you are attempting to produce only a single trace for each nodal waveform, not a family of traces.


The potential difficulty arises when the Data Display server attempts to display a family of traces, each having different numbers of trace points and irregular spacings. Simply stated, the data cannot be displayed. The Data Display server must have a rectangular array of data. This means that all subtraces must have the same number of points, and that the spacing between the points must be the same as that between the corresponding points in other subtraces. The spacing can be irregular, as long as the distribution of spacing along the x-axis is the same for all traces. The only way to be sure that the Data Display server receives a neatly formatted, rectangular array of data is to specify a value for Max time step. This is especially true of Monte Carlo simulations, which are essentially statistical parameter sweeps.

Controlling the Amount of Data Sent to the Dataset

Time-domain simulations generally take more time and produce more nodal data than do comparable frequency-domain simulations. Hundreds (or even thousands) of time points may be required to simulate the behavior of the circuit accurately. This makes it especially important to minimize the number of time points and swept parameter values. You can control the amount and kind of data sent to the dataset using the Output tab of each analysis controller.

Using Sweeps in Monte Carlo Analysis

It is possible to use sweeps in Monte Carlo statistical analyses with the transient simulator, to the extent that histograms can be generated. Yield percentage analyses are not possible except through the following indirect method.

 **Note**
Monte Carlo analyses are enabled through the use of the Yield and YieldOptim components in the Optim/Stat/Yield palette (under their Parameters tab, enable Random variables to dataset, and establish seed values as appropriate).

Yield percentages can be estimated if the pass/fail yield criterion can be expressed in terms of node voltages and currents at individual time points.

Instead of plotting a histogram, plot the cumulative distribution function (CDF) using the `cdf()` operator and equations. The CDF increases monotonically from 0 to 1. Insert two markers onto the CDF trace at the limits of the pass/fail criterion. Then use the delta marker mode to find the difference between them. The yield percentage is the difference between the two markers, multiplied by 100. For example, suppose the response of a circuit to a 1.0-V step with a 100-psec risetime is being plotted. The maximum risetime desired from the circuit is 300 psec.

Ask for 100 Monte Carlo trials. Select the time point where the output voltage must be at least 90% (perhaps $t = 350$ psec). Then plot the CDF multiplied by 100. Insert a marker on the 90% value and another on the maximum value, then examine the difference. This difference is approximately the yield.

Monte Carlo statistical analyses can be regarded as a special type of parameter sweep in which the parameter values are assigned randomly, rather than as a succession of ascending values.

SweepPlan Controller

The options listed in the following table describe the SweepPlan controller's parameters. This controller sweeps a parameter which may be called by a ParamSweep controller or a simulator. It is unitless, as the parameter it sweeps can be any parameter. In the table, names used in netlists and schematics appear under Parameter Name.

Sweep Plan Options

Setup Dialog Name	Parameter Name	Description
SweepPlan Instance Name		Enter the name of the SweepPlan controller. The default is SwpPlan1.
Parameter		Use this area in conjunction with the Add button to add Start, Stop, and Step parameters to the schematic. Use the Cut button to remove a parameter set, and Paste to copy one that has been selected.
Sweep Type		

	Single point	Pt	Enables simulation at a single frequency point. Specify the desired value in the Parameter field.
	Linear		Enables sweeping a range of values based on a linear increment. Click the Start/Stop option to select start and stop values for the sweep, or Center/Span to set the center value and a span of the sweep.
	Log		Enables sweeping a range of values based on a logarithmic increment. Click Start/Stop to set start and stop values for the sweep, or Center/Span to select a center value and a span of the sweep.
Start/Stop Start, Stop, Step-size, Pts./decade, Num. of pts.		Start Stop Step Dec Lin	Select the Start/Stop option to sweep based on start, stop, step-size or pts./decade, and number of points. Linear sweep uses Step-size; Log sweep uses Pts./decade. <ul style="list-style-type: none"> - Start-the start point of a sweep - Stop-the stop point of a sweep - Step-size-the increments at which the sweep is conducted - Pts./decade-number of points per decade - Num. of pts.-the number of points over which sweep is conducted
Center/Span Center, Span, Step-size, Pts./decade, Num. of pts.		Center Span Step Dec Lin	Select the Center/Span option to sweep based on center and span, step-size or pts./decade, and number of points. Linear sweep uses Step-size; Log sweep uses Pts./decade. <ul style="list-style-type: none"> - Center-the center point of

		a sweep - Span-the span of a sweep - Step-size-the increments at which the sweep is conducted - Pts./decade-number of points per decade - Num. of pts.-the number of points over which sweep is conducted
Note: Changes to any of the Start, Stop, etc. fields causes the remaining fields to be recalculated automatically.		
Increasing Order	Reverse=no	Start and progress through sweep from lower to higher values.
Decreasing Order	Reverse=yes	Start and progress through sweep from higher to lower values.
Next Sweep Plan	UseSweepPlan=yes SweepPlan=	Use this field to enter the name of the sweep plan (SweepPlan) to be performed after the current plan.

Parameter Sweep Controller

This section describes the fields of the Parameter Sweep controller tabs. In the following tables, names used in netlists and schematics appear under Parameter Name.

Parameter Sweep Options

Setup Dialog Name	Parameter Name
ParamSweep Instance Name	
Parameter to sweep	SweepVar
Parameter sweep	

	Sweep Type	
		Single point
		Linear
		Log
	Start/Stop Start, Stop, Step-size, Pts./decade, Num. of pts.	Pt Stop Step Dec Lin
Center/Span Center, Span, Step-size, Pts./decade, Num. of pts.	Center Span Step Dec Lin	
Note: Changes to any of the Start, Stop, etc. fields causes the remaining fields to be recalculated automatically.		
Use sweep plan		SweepPlan

Simulations Tab

Setup Dialog Name	Parameter Name	Description
ParamSweep Instance Name		Enter the name of the sweep controller. The default is Sweep1
Simulations to perform	SimInstanceName[n]	Use this area to enter the name(s) of the simulation(s) you wish to perform. As an example, entering DC1, AC1, and HB3 in the setup dialog box causes the following to appear in the schematic: Simulation 1="DC1", Simulation 2="AC1", and Simulation 3="HB3". The simulation controller instance name(s) entered should include double quotation marks ("name") on the schematic. The quotation marks are inserted automatically when you enter a name using the dialog box. You must include the quotation marks if you enter names directly on the schematic. If the quotation marks are missing, the simulation

		will not work. Simulations will be performed in the sequence listed.
--	--	---

Display Tab

For information on the Display tab, which enables you to control the visibility of simulation parameters on the Schematic, see the topic [Displaying Simulation Parameters on the Schematic](#).

Working with Data Files

Data files are ASCII text representations of circuit responses based on various settings of independent variables. For instance the S-parameter response of an amplifier can be captured against frequency and power variations in a .p2d data file. Some ADS components, such as the AmplifierP2D_Setup and AmplifierS2D_Setup help create data files. Other sources for data files are measurement instruments or other simulation tools which output circuit responses in text form. Thus data files enable you to take data from sources outside of Advanced Design System or RF Design Environment and apply it to projects within these design environments.

The common purpose of all the various applications which require the use of data files, is to generate the behavior of a specific component or a circuit based on simulated or measured data points. Thus, data files allow the transfer of realistic parameter values to simple components and also enable the modeling of components with complex behavior such as black box and gray box models. Some examples of the use of data files are:

- Using S-parameters to define the behavior of a linear black-box component representing an attenuator, a filter, or a small-signal transistor. The S-parameters, which are saved in a file, are used in conjunction with a component like the S2P. This permits the creation of a realistic and customized 2-port network during an ADS simulation.
- Storing sets of transistor model parameters in separate files, and accessing them automatically through the course of a simulation to define the behavior of the transistor.
- Defining the behavior of a complex nonlinear amplifier by using the gray-box AmplifierS2D component and data saved in an S2D file. The small and large signal S-parameters as well as noise parameters contained within the file can be used to define the behavior of the amplifier during a simulation.

Besides data-file driven components, there are other user-defined models such as using SDDs, FDDs, equation-based components, or the Model Builder which are discussed elsewhere. This section focuses on how to use the various types of data files to define the behavior of components and circuits in addition to providing a comprehensive understanding of the classification and formats associated with the various types of data files used by various components.

A data file is simply data in an ASCII text file, but there are several formats to choose from depending on the application. When determining the type of data file to choose please note:

- The format you choose may depend on the type of data you have and where you want to use the data. The table below lists supported file formats and examples of where they are used.
- The DataAccessComponent may be used to access the data from any data file regardless of format and to use it with any component that accepts file-based parameters. In this case, you need to make sure there is a logical relationship between the data and how you intend to use it. For more information, see [Using Data Files, Datasets, and Data Access Components](#).

Supported Data Formats

The following table lists the supported data formats with a brief description and a reference to detailed information:

Available File Format Types

Format	Description	Usage	Details
Touchstone Format			
SnP †,†† (.snp)	Small signal S, H, Y, Z, or G-parameters. May also include optional noise data (2 port data only). Where n is the number of ports from 1 to 99.	n-port S-parameter file (S n P) components in the Data Items Library.	Touchstone SnP Format
MDIF Formats			
Discrete (.dscr)	Discrete (indexed) tabular and possibly statistical density data.	Components that accept file-based parameters, link via the DAC.	Discrete Format
Model MDIF	Nonlinear model parameters.	EE_BJT2_Model, JFET_Model, etc.	Model MDIF Files
PDF ††† (.pdf)	User defined, piece-wise linear probability density function data for arbitrary distributions that are not correlated.	With expressions in the Statistics tab.	PDF Format
S2PMDIF (.s2p)	Multi-dimensional 2-port, S, Y, Z, H, G signal and optional 2-port noise parameter (Fmin, Gopt, Rn) data.	With S2PMDIF, DAC, and components represented by black box statistical characterization.	S2PMDIF Format

Advanced Design System 2008

P2D †,†† (.p2d)	Large-signal, power-dependent, 2-port S, H, Y, Z, or G -parameters.	AmplifierP2D in the System - Amps & Mixers library.	P2D Format
S2D †,†† (.s2d)	2-port S, H, Y, Z, or G-parameters with forward gain compression and optional noise and intermodulation data.	Amplifier2 and AmpSingleCarrier in the System-Amps & Mixers library, AmplifierS2D in the System-Data Models library.	S2D Format
IMT †† (.imt)	Intermodulation product table of mixer intermodulation products between the LO and signal that relates the mixer IM output level to signal input level.	MixerIMT in the System - Amps & Mixers library. MixerIMT Data in the System-Data Models library.	IMT Format
SPW †† (.ascsig text) (.sig binary)	Time-domain voltage data file in Cadence Alta Group SPW text and binary formats.	TimeFile item in Timed Sources and OutFile item in the Sinks library.	SPW Format
TIM †† (.tim)	Time-domain data.	TimeFile item in Timed Sources and OutFile item in Sinks library.	TIM Format
SDF ††,††† (.sdf)	Time-domain voltage data file in 89600 file format.	TimeFile item in Timed Sources and OutFile item in Sinks library.	See software documentation for the Agilent 89600.
GCOMP ††	Gain compression data	Amplifier and Mixer items in the System - Amps & Mixers library.	Understanding GCOMP Data
Generic MDIF (.mdif)	Generalized multi-dimensional tables unifying other MDIF formats. Use in place of any specific MDIF.	AmplifierS2D, AmplifierP2D, or any other MDIF example listed above. Link via the DAC.	Generic MDIF
CITIfile Format			
CITIfile †††	A general data format supported by network analyzers. Capable of storing multiple packages of multi- dimensional data.	n-port S-parameter file (S n P) components in the Data Items Library.	CITIfile Data Format
Agilent IC-CAP Formats			
DUT, MDL, SET †,††	Device under test (DUT),	Once the data is read into a	See Agilent IC-CAP

	<p>model (MDL), and setup (SET) files from the Agilent IC-CAP program. These files can contain Measured, Simulated, and/or Transformed data.</p>	<p>dataset, it can be used with any component (for example, a VtDataset source) that can read data from a dataset.</p>	<p>documentation.</p>
<p>† When writing data from a dataset to a file, the variable names are limited to S,H,Y,Z or G, for example, S[1,1], S[1,2], G[1,1], G[1,2]. The variable name is used to determine the type of data. †† The first set of data in the dataset that matches the data type (name) will be output. It is not possible to arbitrarily select which data will be output. ††† There are some specific problems with the current version in writing and/or reading this data format. On the Agilent EEsof web site, refer to the Release Notes in Product Documentation and to Technical Support for more information and workarounds (www.agilent.com/find/eesof). ‡ The Data File Tool can only read IC-CAP data. ‡‡ Only simple, scaled expressions with numbers or variables and one operator (either +, -, *, or /) are supported for start, stop, step, and number of points parameters, for example, start= 1 GHz or stop=icmax/10. ‡‡‡ This format is not yet fully supported. The COD, FIR, LAS, and SPE formats were obsolete when ADS 1.0 was introduced and are not used by the application. The LIST2 and T2D formats are also obsolete.</p>			

For information about a particular component, refer to the manuals for Analog/RF or Signal Processing components, or click Help when editing component parameters.

Making a Data File

You can create a data file using these methods:

- Manually type the data into a text file using any text editor, being sure to follow the formatting guidelines for the type of data file that you want.
- Use the Data File Tool. The Data File Tool enables you to transfer data between datasets and files that are in the following file formats: Touchstone, Measurement data interchange format (MDIF), CITIfile, and IC-CAP. For learn about the Data File Tool, see [Reading and Writing Data Files](#).
- Perform a P2D controller-based simulation using AmplifierP2D_Setup or AmplifierS2D_Setup. The results of these simulations are saved to files in P2D and S2D formats respectively, which can then be used with the AmplifierP2D or AmplifierS2D. For more information on the P2D controller itself, see [P2D Simulation](#).

Saving a Data File

When saving a data file, save it as an ASCII text file.

- A data file does not require a particular extension, but you may want to use the extension recommended for each format for identification purposes. These extensions are given in the details describing each format. If you choose a different extension, you must provide this information to the component you intend to use through the File parameter of the component.

You can save the data file:

- In the project where it is used under the <prj>/data directory. This is the default location if no path is provided to the component using the file.
- Any location, if you provide the full file path to the component. You provide this information using the File parameter of the component.

For more information about the File parameter, place the component of interest on a schematic, double-click to edit it, and click the Help button at the bottom of the dialog box.

Using Data Files, Datasets, and Data Access Components

Both data files and datasets can contain data that you want to use with a component. You can use either one, depending on your situation:

- You may want the S-parameters from a simulated design. In this case, use the dataset and a DAC to link the data to the component you want to use.
- You may have S-parameter specifications from a component sheet. In this case, type them into a data file.

You also want to consider the method of linking your data with the component of interest:

- You have a data file, such as an .s2d, and you want to use it with a component that can read such a file, such as the AmplifierS2D. No DAC is needed.
- You have a dataset or data file, but the component you want to use doesn't read the data directly. Use a DataAccessComponent to link the data file and component. The component you choose must have the file-based option under the Parameter Entry Mode or the parameter AllParams. For example, the BJTM1 model has the parameter AllParams; the R component has the Parameter Entry Mode. For instructions on how to use a DAC, see [DataAccessComponent \(Data Access Component\)](#) or [Schematic Capture and Layout](#).

Reading and Writing Data Files

Use the Data File Tool to import and export data between datasets and text files that are in the following file formats:

- Touchstone
- Measurement data interchange format (MDIF)

- CITIfile
- IC-CAP

You can transfer data from a file into a dataset, or vice versa. One application is to transfer data from a dataset to an MDIF file, for use with a specific type of component. For example, a file in P2D format (P2D is one of several MDIF formats) containing S-parameters can then be used by the P2D amplifier. Using the Data File Tool, you can write S-parameters from a dataset to a file in P2D format. Another application is reading Agilent IC-CAP data into a dataset to be used in conjunction with a component, such as a source, that can read data from a dataset.

See the table above in [Supported Data Formats](#) for a list of the available file types, a description of the file contents, and the component that uses the data. Be sure to review the notes at the end of the table. The details about each file format are described later in this topic.

Starting and Exiting the Data File Tool

You can start the Data File Tool from a Schematic window or a Data Display Window.

- From a Schematic window, choose Tools > Data File Tool. Or, click the Data File Tool icon.



- From a Data Display window, choose Tools > Data File Tool. Or, click the Data File Tool icon.

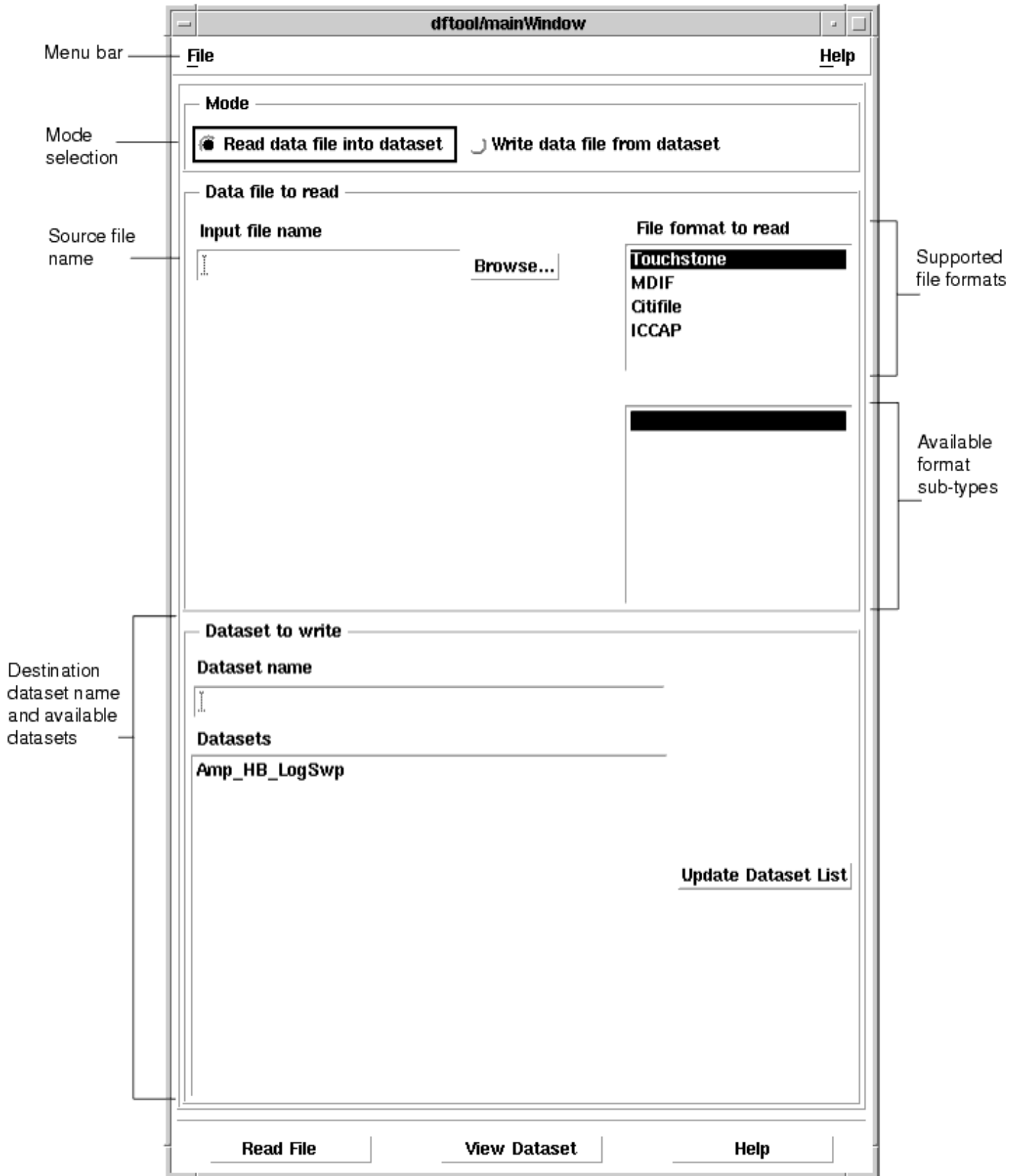


To exit the Data File Tool, choose File > Exit from the menu bar.

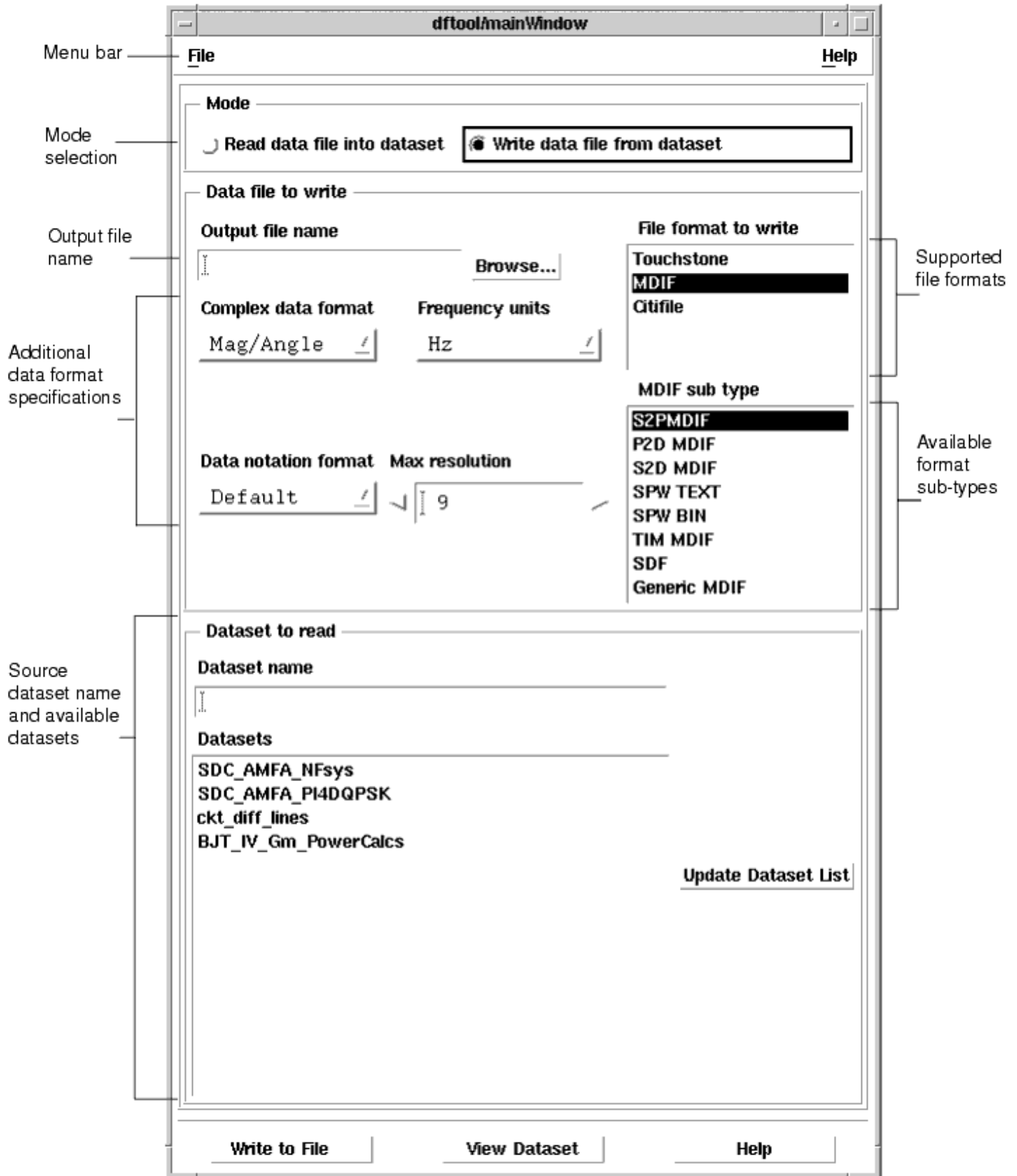
Parts of the Data File Tool

The following illustration shows the default appearance of the Data File Tool user interface for a UNIX-based system when the Data File Tool is started.

The layout of the interface and names of the various elements vary with the task being performed (read or write) and can also vary with the file format selected. Examples of this variation in the appearance of the Data File Tool user interface is shown in the two following figures.



Data File Tool in read mode




These are the more frequently used elements of the interface:

- The Menu bar displays the menus that are available in the Data File Tool window.
- The Dataset field lists the datasets in the current project. The selected dataset or the name of a new dataset is displayed in the Dataset name field.

Reading a File

To read the contents of a file into a dataset:

1. From an open Data File Tool window, click Read data file into dataset.
2. Under File format to read, select one of the following file formats:
 - Touchstone
 - MDIF
 - Citifile
 - ICCAP
3. For the MDIF format, choose the appropriate sub-format from MDIF sub type.
4. Under Input file name, type in the file name if the file is in the project. If it is not, click Browse to locate and select the file.
5. The data from the selected file will be written to a dataset. Enter a name in the Dataset name field or select from the existing datasets in the Datasets list. If you choose a dataset from the list, any data that is already stored in the dataset will not be saved and will be overwritten with new data.
6. Click Read File to send the file contents to the dataset.


 **Note**
The source file must not use any ADS reserved variables or non-ASCII characters. Use of such a file can produce misleading results.

Writing to a File

To write data to a file:

1. From an open Data File Tool window, click Write data file from dataset.
2. Under File format to write, select one of the following file formats:
 - Touchstone
 - MDIF
 - Citifile
3. For the Touchstone and MDIF formats, choose the appropriate sub-format from Touchstone data type, or MDIF sub type.
4. Under Output file name, type in the file name you want to write to. It will be saved in the project directory. If

- you want to save the file in a different location, click Browse to select a location.
5. Under Complex data format, select the complex data format to be used in the file.
 6. For Touchstone and MDIF files, under Frequency units, select the frequency units to be used in the file.
 7. Under Data notation format, select the data notation format to be used in the file.
 8. Under Max resolution, select the maximum resolution to be used in the file.
 9. The source of the data can be any project dataset. It should contain data matching the file format selected.
Select an existing dataset from the Datasets list. Click View Dataset to view the contents of the selected dataset.
 10. Click Write to File to send the data to the file.

 **Note**
If a dataset has just been created by reading a file, it might be necessary to click Update Dataset List to see it appear in the list.

Examples

You can find designs that use different data files in the Examples directory under Data_comp_prj and DataAccess_prj.

Instructions for using a particular type of file with a component that is designed to read the file (like an .snp file and SnP component) can be found in the remaining reference sections.

Touchstone SnP Format

These files contain small-signal G-, H-, S-, Y-, or Z-network parameters described by frequency-dependent linear network parameters for 1- to 99-port components. The 2-port component files can also contain frequency-dependent noise parameters. This data file format is also known as Touchstone format.

An .snp file can be used with an SnP component to model the behavior of a linear model using S-parameters. The file contains the S-parameters, the component is placed within the schematic.

This section describes:

- Choosing an .snp file for use with an SnP component
- An overview of the SnP file
- The basic SnP format
- Adding noise to a 2-port Snp file
- The basic SnP format applied to G-, H-, S-, Y-, and Z-parameters, plus examples of each

Linking an .snp File to an SnP Component

To link a file to the component:

1. Add an SnP component to your schematic. It can be found in the Data Items library.
2. Select the File parameter. Ensure that the Parameter Entry Mode is set to Network Parameter File Name .
3. In the File Name field, enter the name of the file you want to use:
 - You can type the name directly in the field.
 - Click Data files list to locate a file in the current project (or any files located based on the setting of the DATAFILES variable in de_sim.cfg).
 - Click Browse to locate a file outside the current project.
 - Click Copy template to select an example file that you can customize.
4. After you select a file, click Edit if you want to view the file or change its contents.

For instructions on how to set the remaining parameters, click Help in the open component dialog box.

Overview

SnP data files are ASCII text files in which data appears line by line, one line per data point, in increasing order of frequency. Each line of data consists of a frequency value and one or more pairs of values for the magnitude and phase of each S-parameter at that frequency. Values are separated by one or more spaces, tabs or commands. Comments are preceded by an exclamation mark (!). Comments can appear on separate lines, or after the data on any line or lines. Extra spaces are ignored. Recommendations for filenames are:


1-port: filename.s1p

2-port: filename.s2p

Up to 99 ports can be defined.

You can specify the following parameters in an .snp file:

S	=	Scattering parameters
Y	=	Admittance parameters
Z	=	Impedance parameters
H	=	Hybrid-h parameters
G	=	Hybrid-g parameters

 **Note**
The mismatched port impedance is not supported by the ADS simulator. If a Touchstone file has the

input/output mismatch information in the header, it is ignored by the DAC and the SnP components, and a default matching 50 ohm port impedance is used.

The following sections discuss the content and format of network parameter files as input for circuit analysis.

Basic SnP File Format

The following example shows the general format for component data files. It consists of:

- An option line
- Data lines
- Comments

The Option Line

The option line, specifying the frequency units and the normalizing impedance, precedes the data lines.

(freq_units parameter format R n)

<data line>

...

<data line>

where:

#	=	The delimiter that tells the program you are specifying these parameters
freq_units	=	Sets the units. Options are GHz, MHz, KHz, or Hz.
parameter	=	Sets the desired parameter. Options are:
format	=	The format desired. Options are:
R n	=	The reference resistance in ohms, where n is a positive number of ohms; which is the real impedance to which the parameters are normalized.

In summary, the option line should read:

For .s1p files:	#	[HZ/KHZ/MHZ/GHZ][S/Y/Z]	[MA/DB/RI]	[R n]
For .s2p files:	#	[HZ/KHZ/MHZ/GHZ][S/Y/Z/G/H]	[MA/DB/RI]	[R n]
For .s3p/.s4p	#	[HZ/KHZ/MHZ/GHZ][S]	[MA/DB/RI]	[R n]

files:					
--------	--	--	--	--	--

where square brackets [...] indicate optional information; .../.../.../ indicates that you select one of the choices; and, n is replaced by a positive number.

Default Option Line

The default option line for component data files is:

```
# GHz S MA R 50
```

Option Line Examples

Frequency in GHz, S-parameters in real-imaginary format, normalized 100 ohms:

```
# GHz S RI R 100
```

Frequency in KHz, Y-parameters in real-imaginary format, normalized 100 ohms:

```
# KHz Y RI R 100
```

Frequency in Hz, Z-parameters in magnitude-degree format, normalized to 1 ohm:

```
# Hz Z MA R 1
```

Frequency in KHz, H-parameters in real-imaginary format normalized to 1 ohm:

```
# KHz H RI R 1
```

Frequency in Hz, G-parameters in magnitude-degree, format normalized to 1 ohm:

```
# Hz G MA R 1
```

Data Lines

Data lines contain the data of interest. A special format is used for 2-port data files where all of the network parameter data for a single frequency is listed on one line. The order of the network parameters is:

N11, N21, N12, N22

For 3-port or higher data files, the network parameters appear in the file in a matrix form, each row starting on a separate line. A maximum of four network parameters (with 2 real numbers for each) appear on any line. The remaining network parameters are continued on as many additional lines as are needed.

The following sections describe the data-line format for single and multi-port components.

Advanced Design System 2008

Data-line Formats

When you type the data below the option line, the columns need not line up precisely like those shown. The syntax for entering data is as follows:

1-port Component

Magnitude-Angle format:

(Columns:	f	Mag	Ang)
	f	S11	

2-port Component

Magnitude-Angle format:

f	S11
---	-----

Real-Imaginary format:

f	Re{S11} Im{S11} Re{S21} Im{S21} Re{S12} Im{S12} Re{S22} Im{S22}
---	--

dB-Angle format:

f	20log10 x11
---	-------------

where

x	=	S/Y/Z/H/G
f	=	Frequency

3-port Component

Magnitude-Angle format:

(Columns :	f	Mag	Ang	Mag	Ang	Mag	Ang)
	f	S11	<S11	S12	<S12	S13	<S13
		S21	<S21	S22	<S22	S23	<S23
		S31	<S31	S32	<S32	S33	<S33

4-port Component

Magnitude-Angle format:

(Columns	f	Mag	Ang	Mag	Ang	Mag	Ang	Mag	Ang)
:									
	f	S11	<S11	S12	<S12	S13	<S13	S14	<S14
		S21	<S21	S22	<S22	S23	<S23	S24	<S24
		S31	<S31	S32	<S32	S33	<S33	S34	<S34
		S41	<S41	S42	<S42	S43	<S43	S44	<S44

where:

f	=	Frequency
Mag	=	Magnitude of S-parameter Sij
Ang	=	Angle of S-parameter Sij

Adding Comments to Data Files

You can document your data files by preceding a comment with the exclamation mark (!) on any line. A comment can be the only entry on a line or can follow the data on any line.

Adding Noise Parameters to an SnP File

Noise parameters can be included in SnP 2-port data files. Noise data can follow G-, H-, S-, Y-, or Z-parameters described for each frequency. The x values are data.

Each line of a noise parameter has the following five entries:

x1 x2 x3 x4 x5


where:

x1	=	Frequency in units. The first point of noise data must have a frequency less than the frequency of the last S-parameter frequency
x2	=	Minimum noise figure in dB
x3	=	Source reflection coefficient to realize minimum noise figure (MA)

x4	=	Phase in degrees of the reflection coefficient (MA)
x5	=	Normalized effective noise resistance. The system simulator requires this parameter to meet physical requirements. If the user-supplied x5 value is less than allowed for this requirement, then the system simulator will force this x5 value to the lowest physical limit.

Sopt noise data in an SnP file must be expressed in a Magnitude/Angle (MA) format. The simulator assumes that the Sopt noise data in an SnP file is specified in a MA format. This is the only supported format for Sopt data in an SnP file. The complex format type specified in the format line does not apply to the Sopt data specified in the Noise block.

The data file reader cannot determine if the numbers representing the Sopt data in the SnP data file are expressed in MA format and not in dB or Real/Imag formats. It reads in whatever numbers appear on the data line and processes them as if they are MA, without issuing an error or warning message. If the Sopt data in the SnP data file is expressed in a format other than MA, this can produce simulation data that look incorrect.

 **Note**
 The frequencies for noise parameters and network parameters need not match. The only requirement is that the lowest noise-parameter frequency be less than or equal to the highest network-parameter frequency. This allows the file processor to determine where network parameters end and noise parameters begin.

The source reflection coefficient and effective noise resistance are normalized to the same resistance as specified for the network parameters.

Example File Containing Noise Data

This is an example of a data file with noise data:

```
! NEC710
# GHZ S MA R 50
2 .95 -26 3.57 157 .04 76 .66 -14
22 .60 -144 1.30 40 .14 40 .56 -85
! NOISE PARAMETERS
4 .7 .64 69 .38
18 2.7 .46 -33 .40
```

Applying the SnP Format, and Examples

In this section are formatting references and examples for:

- G-parameter files
- H-parameter files
- S-parameter files
- Y- and Z-parameter files

Guidelines

- The optimum source reflection coefficient and the normalized effective noise resistance are assumed to be with respect to the normalizing resistance value (appearing after the R keyword) on the header line.
- The frequencies for noise parameters and G-, H-, S-, Y-, or Z-parameters (network parameters) do not have to match. The only requirement is that the lowest noise parameter frequency be less than or equal to the highest network parameter frequency. This allows the file processor to determine where G-, H-, S-, Y-, or Z-parameters end and noise parameters begin.

G-Parameter Files

G-parameter files (Hybrid-g parameters) use MA or RI format. They are strictly 2-port files. G-parameter measurements are:

G11	input admittance (port 2 open)
G22	output impedance (port 1 shorted)
G21	forward voltage gain (port 2 open)
G12	reverse current gain (port 1 shorted)

G-Parameter MA and RI File Formats

```
#      frequency_unit      G      MA      R      impedance
freq  magG11  angG11  magG21  angG21  magG12  angG12  magG22  angG22
#      frequency_unit      G      RI      R      impedance
freq  reG11  imG11  reG21  imG21  reG12  imG12  reG22  imG22
```

G-Parameter File Example

```
! symbol freq-unit parameter-type data-format keyword impedance-ohms
#      KHZ      G      MA      R      1
! freq  magG11  angG11  magG21  angG21  magG12  angG12  magG22  angG22
```

Advanced Design System 2008

2	.95	-26	3.57	157	.04	76	.66	-14
3	.93	-40	3.53	147	.05	69	.65	-20
4	.89	-52	3.23	136	.06	62	.63	-26

H-Parameter Files

H-parameter files (Hybrid-h parameters) use MA or RI format. They are strictly 2-port files. H-parameter measurements are:

H11	input impedance (port 2 shorted)
H22	output admittance (port 1 open)
H21	forward current gain (port 2 shorted)
H12	reverse voltage gain (port 1 open)

H-Parameter File Example

```

! symbol      freq-unit      parameter-type      data-format      keyword      impedance-ohms
#   KHZ       H           MA           R           I
!  freq      magH11      angH11      magH21      angH21      magH12      angH12      magH22      angH22
2     .95      -26         3.57        157         .04         76         .66         -14
3     .93      -40         3.53        147         .05         69         .65         -20
4     .89      -52         3.23        136         .06         62         .63         -26
    
```

S-Parameter Files

S-parameter files (scattering parameters) can have MA, RI, or DB format for files with 1 to 99 ports.

S-Parameter 1-Port MA, RI, and DB File Formats

```

#      frequency_unit      S      MA      R      impedance
freq      magS11      angS11
#      frequency_unit      S      RI      R      impedance
freq      reS11      imS11
#      frequency_unit      S      DB      R      impedance
freq      dbS11      angS11
    
```

S-Parameter 2-Port MA, RI, and DB File Formats

Advanced Design System 2008

```

# frequency_unit S MA R impedance
freq magS11 angS11 magS21 angS21 magS12 angS12 magS22 angS22
# frequency_unit S RI R impedance
freq reS11 imS11 reS21 imS21 reS12 imS12 reS22 imS22
# frequency_unit S DB R impedance
freq dbS11 angS11 dbS21 angS21 dbS12 angS12 dbS22 angS22
    
```

S-Parameter 3-Port MA, RI, and DB File Formats

```

# frequency_unit S MA R impedance
freq magS11 angS11 magS12 angS12 magS13 angS13 ! 1st row
      magS21 angS21 magS22 angS22 magS23 angS23 ! 2nd row
      magS31 angS31 magS32 angS32 magS33 angS33 ! 3rd row
# frequency_unit S RI R impedance
freq reS11 imS11 reS12 imS12 reS13 imS13 ! 1st row
      reS21 imS21 reS22 imS22 reS23 imS23 ! 2nd row
      reS31 imS31 reS32 imS32 reS33 imS33 ! 3rd row
# frequency_unit S DB R impedance
freq dbS11 angS11 dbS12 angS12 dbS13 angS13 ! 1st row
      dbS21 angS21 dbS22 angS22 dbS23 angS23 ! 2nd row
      dbS31 angS31 dbS32 angS32 dbS33 angS33 ! 3rd row
    
```

S-Parameter 4-Port MA, RI, and DB File Formats

```

# frequency_unit S MA R impedance
! 1st row
freq magS11 angS11 magS12 angS12 magS13 angS13 magS14 angS14
      magS21 angS21 magS22 angS22 magS23 angS23 magS24 angS24 ! 2nd row
      magS31 angS31 magS32 angS32 magS33 angS33 magS34 angS34 ! 3rd row
      magS41 angS41 magS42 angS42 magS43 angS43 magS44 angS44 ! 4th row
# frequency_unit S RI R impedance
freq reS11 imS11 reS12 imS12 reS13 imS13 reS14 imS14 ! 1st row
      reS21 imS21 reS22 imS22 reS23 imS23 reS24 imS24 ! 2nd row
      reS31 imS31 reS32 imS32 reS33 imS33 reS34 imS34 ! 3rd row
      reS41 imS41 reS42 imS42 reS43 imS43 reS44 imS44 ! 4th row
# frequency_unit S DB R impedance
freq dbS11 angS11 dbS12 angS12 dbS13 angS13 dbS14 angS14 ! 1st row
      dbS21 angS21 dbS22 angS22 dbS23 angS23 dbS24 angS24 ! 2nd row
      dbS31 angS31 dbS32 angS32 dbS33 angS33 dbS34 angS34 ! 3rd row
      dbS41 angS41 dbS42 angS42 dbS43 angS43 dbS44 angS44 ! 4th row
    
```

S-Parameter 1-Port File Example

```

! symbol freq-unit parameter-type data-format keyword impedance-ohms
# MHZ S MA R 50
! freq magS11 angS11 (commented header line)
  2.000 0.894 -12.136
  3.000 0.893 -18.179
  4.000 0.891 -24.193
    
```

Advanced Design System 2008

S-Parameter 5- to 99-Port File Formats

These file formats appear in a matrix form similar to the 3- and 4-port files, except that only four S-parameters (with 2 real numbers for each) can appear on a given line. Therefore, the remaining S-parameters in that row of the S-matrix continue on the next line of the file.

Each row of the S-matrix must begin on a new line of the file. The first line of the first row of the S-matrix begins with the frequency value.

S-Parameter 10-Port File Example (at One Frequency)

```
#      frequency_unit      S      MA      R      impedance
freq magS11 angS11 magS12 angS12 magS13 angS13 magS14 angS14 ! 1st row
magS15 angS15 magS16 angS16 magS17 angS17 magS18 angS18
magS19 angS19 magS1,10 angS1,10
magS21 angS21 magS22 angS22 magS23 angS23 magS24 angS24 ! 2nd row
magS25 angS25 magS26 angS26 magS27 angS27 magS28 angS28
magS29 angS29 magS2,10 angS2,10
magS31 angS31 magS32 angS32 magS33 angS33 magS34 angS34 ! 3rd row
magS35 angS35 magS36 angS36 magS37 angS37 magS38 angS38
magS39 angS39 magS3,10 angS3,10
magS41 angS41 magS42 angS42 magS43 angS43 magS44 angS44 ! 4th row
magS45 angS45 magS46 angS46 magS47 angS47 magS48 angS48
magS49 angS49 magS4,10 angS4,10
magS51 angS51 magS52 angS52 magS53 angS53 magS54 angS54 ! 5th row
magS55 angS55 magS56 angS56 magS57 angS57 magS58 angS58
magS59 angS59 magS5,10 angS5,10
magS61 angS61 magS62 angS62 magS63 angS63 magS64 angS64 ! 6th row
magS65 angS65 magS66 angS66 magS67 angS67 magS68 angS68
magS69 angS69 magS6,10 angS6,10
magS71 angS71 magS72 angS72 magS73 angS73 magS74 angS74 ! 7th row
magS75 angS75 magS76 angS76 magS77 angS77 magS78 angS78
magS79 angS79 magS7,10 angS7,10
magS81 angS81 magS82 angS82 magS83 angS83 magS84 angS84 ! 8th row
magS85 angS85 magS86 angS86 magS87 angS87 magS88 angS88
magS89 angS89 magS8,10 angS8,10
magS91 angS91 magS92 angS92 magS93 angS93 magS94 angS94 ! 9th row
magS95 angS95 magS96 angS96 magS97 angS97 magS98 angS98
magS99 angS99 magS9,10 angS9,10
!10th row
magS10,1 angS10,1 magS10,2 angS10,2 magS10,3 angS10,3 magS10,4 angS10,4
magS10,5 angS10,5 magS10,6 angS10,6 magS10,7 angS10,7 magS10,8 angS10,8
magS10,9 angS10,9 magS10,10 angS10,10
```

Linear 1-Port (.s1p) File Example

```
# GHZ      S      RI      R      50.0
1.00000000 0.9488  -0.2017
1.50000000 0.9077  -0.3125
2.00000000 0.8539  -0.4165
2.50000000 0.7884  -0.5120
3.00000000 0.7124  -0.5978
3.50000000 0.6321  -0.6546
4.00000000 0.5479  -0.7013
```

Advanced Design System 2008

4.50000000	0.4701	-0.7380
5.00000000	0.3904	-0.7663
5.50000000	0.3302	-0.7778
6.00000000	0.2702	-0.7848
6.50000000	0.2041	-0.7890
7.00000000	0.1389	-0.7878
7.50000000	0.0894	-0.7849
8.00000000	0.0408	-0.7789
8.50000000	0.0134	-0.7649
9.50000000	0.0654	-0.7471
9.00000000	0.1094	-0.7319
10.00000000	0.1518	-0.7140

Linear 2-Port (.s2p) File Example

```

# GHZ      S      RI      R      50.0
1.0000    0.3926   -0.1211  -0.0003  -0.0021  -0.0003  -0.0021  0.3926  -0.1211
2.0000    0.3517   -0.3054  -0.0096  -0.0298  -0.0096  -0.0298  0.3517  -0.3054
10.000    0.3419    0.3336  -0.0134   0.0379  -0.0134   0.0379  0.3419  0.3336
!Noise params
1.0000    2.0000   -0.1211  -0.0003   .4
2.0000    2.5000   -0.3054  -0.0096   .45
3.0000    3.0000   -0.6916  -0.6933   .5
4.0000    3.5000   -0.3756   0.4617   .55
5.0000    4.0000    0.3880   0.6848   .6
6.0000    4.5000    0.0343   0.0383   .65
7.0000    5.0000    0.6916   0.6933   .7
8.0000    5.5000    0.5659   0.1000   .75
9.0000    6.0000    0.4145   0.0307   .8
10.0000   6.5000    0.3336   0.0134   .85
    
```

Linear 3-Port (.s3p) File Example

```

# GHZ      S      MA      R      50.0
! POWER DIVIDER, 3-PORT
5.00000    0.24254   136.711  0.68599  -43.3139  0.68599  -43.3139
           0.68599  -43.3139  0.08081  66.1846  0.28009  -59.1165
           0.68599  -43.3139  0.28009  -59.1165  0.08081  66.1846
6.00000    0.20347   127.652  0.69232  -52.3816  0.69232  -52.3816
           0.69232  -52.3816  0.05057   52.0604  0.22159  -65.1817
           0.69232  -52.3816  0.22159  -65.1817  0.05057   52.0604
7.00000    0.15848   118.436  0.69817  -61.6117  0.69817  -61.6117
           0.69817  -61.6117  0.02804   38.6500  0.16581  -71.2358
           0.69817  -61.6117  0.16581  -71.2358  0.02804   38.6500
    
```

Linear 4-Port (.s4p) File Example

```

# GHZ      S      MA      R      50
5.00000    0.60262   161.240  0.40611  -42.2029  0.42918  -66.5876  0.53640  -79.3473
           0.40611  -42.2029  0.60262  161.240  0.53640  -79.3473  0.42918  -66.5876
           0.42918  -66.5876  0.53640  -79.3473  0.60262  161.240  0.40611  -42.2029
    
```

Advanced Design System 2008

	0.53640	-79.3473	0.42918	-66.5876	0.40611	-42.2029	0.60262	161.240
6.00000	0.57701	150.379	0.40942	-44.3428	0.41011	-81.2449	0.57554	-95.7731
	0.40942	-44.3428	0.57701	150.379	0.57554	-95.7731	0.41011	-81.2449
	0.41011	-81.2449	0.57554	-95.7731	0.57701	150.379	0.40942	-44.3428
	0.57554	-95.7731	0.41011	-81.2449	0.40942	-44.3428	0.57701	150.379
7.00000	0.50641	136.693	0.45378	-46.4151	0.37845	-99.0918	0.62802	-114.196
	0.45378	-46.4151	0.50641	136.693	0.62802	-114.196	0.37845	-99.0918
	0.37845	-99.0918	0.62802	-114.196	0.50641	136.693	0.45378	-46.4151
	0.62802	-114.196	0.37845	-99.0918	0.45378	-46.4151	0.50641	136.693

Y- and Z-Parameter Files

Immittance parameters are specified in MA or RI format, where the # line has Y for admittance and Z for impedance. Both are normalized to the reference resistance.

Y- (Z-) Parameter 1-Port MA and RI File Formats

```
# frequency_unit Y MA R impedance
freq magY11 angY11
# frequency_unit Y RI R impedance
freq reY11 imY11
```

Y- (Z-) Parameter 2-Port MA and RI File Formats

```
# frequency_unit Y MA R impedance
freq magY11 angY11 magY21 angY21 magY12 angY12 magY22 angY22
# frequency_unit Y RI R impedance
freq reY11 imY11 reY21 imY21 reY12 imY12 reY22 imY22
```

Y- (Z-) Parameter 3-Port MA and RI File Formats

```
freq magY11 angY11 magY12 angY12 magY13 angY13 ! 1st row
magY21 angY21 magY22 angY22 magY23 angY23 ! 2nd row
magY31 angY31 magY32 angY32 magY33 angY33 ! 3rd row
# frequency_unit Y RI R impedance
freq reY11 imY11 reY12 imY12 reY13 imY13 ! 1st row
reY21 imY21 reY22 imY22 reY23 imY23 ! 2nd row
reY31 imY31 reY32 imY32 reY33 imY33 ! 3rd row
```

Y- (Z-) Parameter 4-Port MA and RI File Formats

Advanced Design System 2008

```

# frequency_unit Y MA R impedance
freq magY11 angY11 magY12 angY12 magY13 angY13 magY14 angY14 ! 1st row
    magY21 angY21 magY22 angY22 magY23 angY23 magY24 angY24 ! 2nd row
    magY31 angY31 magY32 angY32 magY33 angY33 magY34 angY34 ! 3rd row
    magY41 angY41 magY42 angY42 magY43 angY43 magY44 angY44 ! 4th row
# frequency_unit Y RI R impedance
freq reY11 imY11 reY12 imY12 reY13 imY13 reY14 imY14 ! 1st row
    reY21 imY21 reY22 imY22 reY23 imY23 reY24 imY24 ! 2nd row
    reY31 imY31 reY32 imY32 reY33 imY33 reY34 imY34 ! 3rd row
    reY41 imY41 reY42 imY42 reY43 imY43 reY44 imY44 ! 4th row

```

Y- (Z-) Parameter 3-Port File Example

```

! symbol      freq-unit      parameter-type      data-format      keyword      impedance-ohms
#            GHz            Y            MA            R            1
! freq      magY11      angY11      magY12      angY12      magY13      angY13      ! 1st line
!           magY21      angY21      magY22      angY22      magY23      angY23      ! 2nd line
!           magY31      angY31      magY32      angY32      magY33      angY33      ! 3rd line
4           0.008      83.122      8.5e-04     -86.740     0.007      -98.037
           0.046      -12.740     0.005       36.580     0.049      171.554
           0.046      177.588     0.004      -152.638    0.050       0.134
8           0.016      79.068     0.002      -84.015     0.014     -102.924
           0.049     -23.015     0.006       52.828     0.051      164.123
           0.048     175.827     0.005     -139.640    0.052      -0.004
12          0.025      73.501     0.003      -81.736     0.023     -109.374
           0.058     -36.736     0.009       58.596     0.058      152.007
           0.055     169.129     0.007     -136.047    0.059      -5.349
18          0.036      65.138     0.004     -71.761     0.033     -119.900
           0.059     -54.761     0.013       72.274     0.052      137.118
           0.052     162.979     0.010     -121.976    0.055      -6.677

```

Y- (Z-) Parameter 5- to 99-Port File Formats

These file formats appear in a matrix form similar to 3- and 4-port files. Only four Y-, or Z-parameters (with 2 real numbers for each) can appear on a given line; therefore, the remaining parameters in that row of the matrix continue on the next line of the file. Each row of the Y-matrix must begin on a new line of the file. The first line of the first row of the Y-matrix begins with the frequency value. The actual Y- (Z-) parameter value is obtained by dividing (multiplying) the file entry with the reference resistance.

Y- (Z-) Parameter 10-Port File Example (at One Frequency)

```

# frequency_unit Y MA R impedance
freq magY11 angY11 magY12 angY12 magY13 angY13 magY14 angY14 ! 1st row
magY15 angY15 magY16 angY16 magY17 angY17 magY18 angY18
magY19 angY19 magY1,10 angY1,10
magY21 angY21 magY22 angY22 magY23 angY23 magY24 angY24 ! 2nd row
magY25 angY25 magY26 angY26 magY27 angY27 magY28 angY28
magY29 angY29 magY2,10 angY2,10

```

```

magY31  angY31  magY32  angY32  magY33  angY33  magY34  angY34  ! 3rd row
magY35  angY35  magY36  angY36  magY37  angY37  magY38  angY38
magY39  angY39  magY3,10  angY3,10
magY41  angY41  magY42  angY42  magY43  angY43  magY44  angY44  ! 4th row
magY45  angY45  magY46  angY46  magY47  angY47  magY48  angY48
magY49  angY49  magY4,10  angY4,10
magY51  angY51  magY52  angY52  magY53  angY53  magY54  angY54  ! 5th row
magY55  angY55  magY56  angY56  magY57  angY57  magY58  angY58
magY59  angY59  magY5,10  angY5,10
magY61  angY61  magY62  angY62  magY63  angY63  magY64  angY64  ! 6th row
magY65  angY65  magY66  angY66  magY67  angY67  magY68  angY68
magY69  angY69  magY6,10  angY6,10
magY71  angY71  magY72  angY72  magY73  angY73  magY74  angY74  ! 7th row
magY75  angY75  magY76  angY76  magY77  angY77  magY78  angY78
magY79  angY79  magY7,10  angY7,10
magY81  angY81  magY82  angY82  magY83  angY83  magY84  angY84  ! 8th row
magY85  angY85  magY86  angY86  magY87  angY87  magY88  angY88
magY89  angY89  magY8,10  angY8,10
magY91  angY91  magY92  angY92  magY93  angY93  magY94  angY94  ! 9th row
magY95  angY95  magY96  angY96  magY97  angY97  magY98  angY98
magY99  angY99  magY9,10  angY9,10
!10th row
magY10,1  angY10,1  magY10,2  angY10,2  magY10,3  angY10,3  magY10,4  angY10,4
magY10,5  angY10,5  magY10,6  angY10,6  magY10,7  angY10,7  magY10,8  angY10,8
magY10,9  angY10,9  magY10,10  angY10,10

```

Discrete Format

The discrete data file consists of an array of data arranged in rows and columns. The values available for each parameter are arranged in columns. Following the BEGIN DSCRDATA line is the % format line which specifies the names of dependent variables. The first column is always treated as a string; other columns are real, integer or string, depending on the first row of data.

The first column, under the heading Index in the example below, contains entries used to identify each row in the file. These entries can be either an integer or an alphanumeric identifier, and can be thought of as a list of specification numbers (or part numbers). For example, the data file data/stdvalues15.dscr is arranged as follows:

```

REM      stdvalues15.dscr
BEGIN DSCRDATA
%  INDEX      A12      A13
   1          1000     1000
   2          1000     1200
   3          1000     2200
   4          1200     1000
   5          1200     1200
   6          1200     2200
END DSCRDATA

```

Selecting a Row

A row of data can be selected by specifying its row index (starting from 0). In this example, the file lists two columns

of values labeled A12 and A13. By specifying 2 as the row number, the values 1000 and 2200 are selected for A12 and A13, respectively.

Using the File with a DAC

To use the data within the file, you must link the file to the component of interest. You reference a discrete data file in this way by using a DAC:

1. Place a DataAccessComponent data item in your design. The DAC is located in the Data Items palette. Double-click the DAC to edit it.
2. On the File tab, in the File Name field, specify the name of the discrete data file, and accept the default setting for File Type , which is Discrete .
3. On the Interpolation tab, accept the defaults for Interpolation Method (Index Lookup) and for Interpolation Domain (Rectangular).
4. On the Independent Variable tab, set the names and values for the independent variables. This is necessary since data in a discrete data file can be accessed only by using an index lookup value. This means looking up data by row number.

To set up an independent variable, enter the name in the Variable Name field. For a discrete data file, the innermost independent variable is the dimension number which should be used as the name. Next, enter the row number in the Value field, which can be a variable assigned a value on the schematic. Then click Add to insert the name and value in the table at the left. Repeat this process for each independent variable in the data file. Values entered for Variable Name are treated as strings, and quotation marks are inserted with these values automatically when added to the table's Name column. However, the innermost independent variable of a discrete data file must be specified as a cardinal integer instead of a string name. Assuming you are working with a one-dimensional data file, enter @1 to enter the integer (@ suppresses the quotation marks). For example, here is a portion of a one-dimensional discrete data file:

```
Begin dsrdata
% index mydata
0 12
1 34
2 56
....
end
```

If you define a variable called MyIndex in a schematic whose value represents the index of the row of data to be accessed, the table of independent variables should be constructed this way to read the one-dimensional file:

```
Name Value
1 MyIndex
```

The value assigned to MyIndex in the schematic determines which row of data is read. So if MyIndex = 0 , the first row is read.

5. Place the component whose parameter values should come from the data file. Double-click the component.
6. Under Pa rameter Entry Mode, select File Based.

7. Under Data Access Component Instance , enter the ID of the desired DAC data item.
8. Under Dependent Parameter Name , enter the name of a dependent variable in the discrete data file (In the sample above, the dependent variable names are A12 and A13.).

Example

For an example of using a discrete data file, refer to amp1.dsn in the Examples directory, under Tutorials/DataAccess_prj .

Model MDIF Files

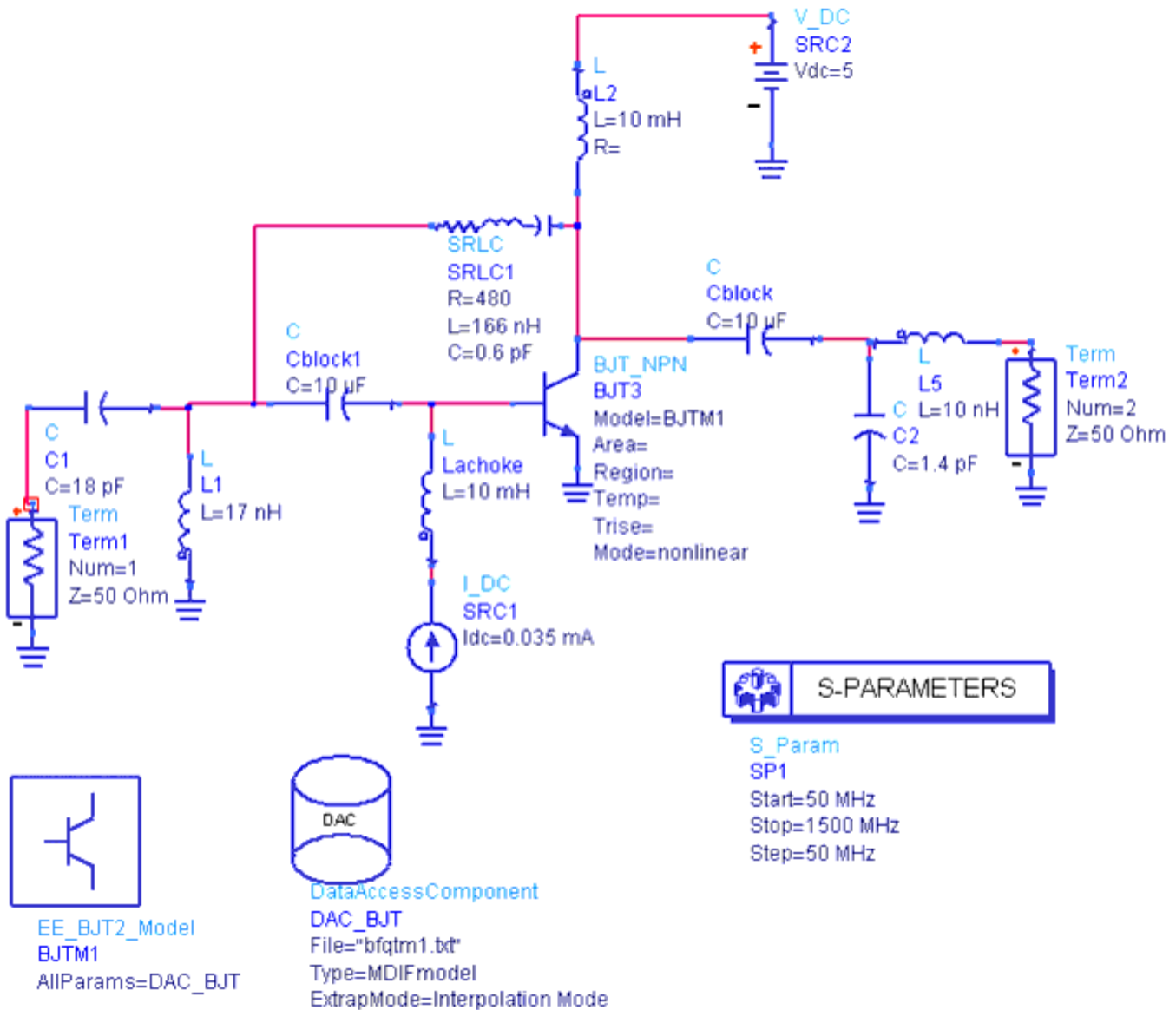
Nonlinear devices obtain their model parameters either from a model item or a file. For those devices that use a file, such as the EE_BJT2_Model , this section discusses the appropriate format for a model file, and how to read the file.

Model files are text files that contain model parameter names and values. A sample model file for the EE_BJT2_Model is shown below. Comments can be placed in the file by starting the line with REM . The model parameters are placed between BEGIN BDTA and END BDTA . One or more parameter names are placed on a line beginning with the percent symbol (%); corresponding values are placed in the same order on the next line. Parameter names can be in any order and are not case sensitive. Any parameters that are not present in the file take on their documented default values. Parameter names for each device are listed in the documentation for that device.

```

REM any line that starts with REM is ignored
BEGIN BDTA
% Rb          Rc          Re          Tamb          Ibir          Nbr
 0.637326    5.17646         0.695         25          3.72528e-15   1.0537
% Isc         Nc          Ibif         Nbf          Ise          Ne
 0           2           5.70565e-17  1.10843     4.63077e-14   1.83578
% Var         Nr          Isr          Ikr          Vaf          Nf
 1.1858      1.02201        1.12936e-14  100         54.9731       0.977768
% Isf         Ikf         Cjc          Vjc          Mjc          Cje
 5.37696e-16 0.857731       5.87976e-13  0.313492    0.0650757     9.81026e-13
% Vje         Mje         Tf           Tr           Fc
 1.05535     0.475724       2.09636e-12  0           0.9
% Xtf         Vtf         Itf
 10          4.32912         1e-09
END BDTA
    
```

The following figure shows how to obtain parameter values from a model MDIF file using a DataAccessComponent. The DAC refers to the model file by name. In this example, the file name is bfqtm1.txt . Additionally, Type is set to MDIFmodel and ExtrapMode set to InterpolationMode . On the device model component, the AllParams parameter is set to the name of the DAC, which is DAC_BJT .



Reading a Model MDIF file using a DataAccessComponent

PDF Format

This format is for user-specified Probability Density Functions (PDF). PDF is used for arbitrary distributions that are not correlated. The two methods available accommodate:

- Situations where the spread of statistical data is proportional to the nominal value (as in a percent tolerance).

Advanced Design System 2008

- Situations where the spread is independent of the nominal value (an absolute tolerance).

Probability density functions are represented as vertices of piecewise linear segments. These value/ordinate pairs are stored in a textual data file in a prescribed format. The nominal value is also stored.

User-specified probability density files have an extension .pdf and use an MDIF file format. Only a single distribution definition is allowed in each .pdf file.

Guidelines for .pdf


- In addition to the nominal value, there must be a minimum of three value/ordinate data pairs.
- For tolerance representation, all value data and the nominal value must have the same algebraic sign.
- The ordinate data associated with the most negative and most positive value data must be zero.
- All ordinate data must be non-negative.
- At least one non-end ordinate data must be non-zero.

Example PDF File

The following example shows how to create a user-defined PDF file. The technique involves the use of a discrete file which contains the nominal value and the statistical data (in piece-wise linear form.) In this example, the first block contains the allowable nominal values, and the second block contains the statistical information. The VALUE/ORDINATE pairs describe the user-defined PDF.

```
BEGIN DSCRDATA
% INDEX my_index
1          50
2          60
3          70
4          100
END DSCRDATA
REM
REM VAR INDEX value below must correlate with the row in the above data
REM block, as selected by the iVall parameter of the DAC.
REM For example, if iVall=3, the 4th row of data from above is chosen
REM (INDEX=4), and VAR INDEX = 4 must be specified below
REM
REM By experiment, the NOMINAL value does not have to correlate with the
REM row number in the DSCRDATA block.
REM
VAR INDEX = 4
VAR PARNAME = my_index
BEGIN TOLERANCE
%NOMINAL
100
%VALUE ORDINATE
90      0.0
95      1.0
98      0.0
100     0.0
102     0.0
```

```
105      1.0
110      0.0
END TOLERANCE
```

 **Note**
The total area under your PDF does not have to equal one as in the strict definition of a PDF-the simulator will automatically scale your PDF to meet this condition.

Interpretation of PDF data

Interpretation of user-supplied PDF data is piece-wise linear with respect to value/ordinate pairs. The data preparation previously described enables the program to supply a properly qualified variate.

To realize a statistical variable which obeys the user-supplied PDF, a uniform variate on the interval 0 to 1 is used as an input to a function which is the inverse of the cumulative distribution function (CDF). The CDF is formed by integrating the PDF from its most negative value to its most positive value, with the following conditions:

- $F_x(-\infty)$, the CDF at minus infinity = 0
- $F_x(\infty)$, the CDF at infinity = 1

Applying this uniform [0,1] variate to the inverse of the CDF results in a statistical variable having the user-specified probability density function.

PDFs may be used with:

- Yield analysis, with or without post-production tuning
- Yield optimization (design centering)

S2PMDIF Format

The S2PMDIF data format file (.s2p) can contain multiple two-port small-signal measurement data and associated noise measurement data in a single file. S2P indicates that the data used is typically S-parameters, though other small-signal parameters (Y, Z, H, G) are supported. These files are a natural extension of two-port S-parameter Touchstone files. For information about Touchstone files, see [Touchstone SnP Format](#). MDIF refers to the fact that these files use the format and syntax rules associated with the Measurement Data Interchange Format (MDIF).

The most typical application of the S2PMDIF format is the creation of a file-based statistical representation for one or more devices in the fabrication process. Due to process variations, S-parameters for the same device will vary naturally. Using the S2PMDIF format, which captures all S-parameter data, in conjunction with statistical and yield analysis tools, which can randomly select a part, statistical characterization of a device (known as a truthmodel) for yield analysis is achieved.

General File Structure

The file structure can repeat for as many small-signal data/noise data pairs as needed. Noise data is optional and the file structure shown here may only have ACDATA blocks if desired.

```

! Comment Line
VAR <_Your_variable_name#1_> = <Your_Value>_
VAR <_Your_variable_name#2_> = <Your_Value>_
VAR <_Your_variable_name#n_> = <Your_Value>_
BEGIN ACDATA
! Option line
% F n11x n11y n21x n21y n12x n12y ! signal format line
! < Your small data consistent with above format line >
END
BEGIN NDATA
! Option line
% F nfmin n11x n11y rn ! noise format line
! < Your noise data >
END
! Repeat entire ACDATA and NDATA blocks above if necessary
! preceded by different VAR values to distinguish measurements.
    
```

Guidelines

The details presented in this section are demonstrated in [Example S2PMDIF File](#). You are encouraged to review the example, then refer back to these guidelines for the detailed information.

VAR items VAR items are used to declare variables that distinguish different small signal/noise parameter pairs. The format is,

```
VAR <name> = <value>
```

Examples:

```
VAR Part_XYZ_sample = 1
VAR SAMPLE = 0
```



Note

VAR is a reserved keyword for the MDIF file. SAMPLE is a reserved variable for statistical analysis applications.

General information S2PMDIF supports

multiple small-signal and/or noise data pairs
 or
 one small-signal and/or noise data pair.

If the latter is used, no VAR declaration is required.

Comments Comments in the S2PMDIF are supported using ! or the REM statement. The ! may appear at the beginning of a line, or as a trailing comment at the end of a line. REM, however, may only serve as a leading comment on the beginning of a line. Examples:

```
REM VAR Lot = 1
! VAR Lot = 1
VAR Lot = 1 ! This is the wafer lot number
```

S2PMDIF data blocks S2PMDIF contains two main data blocks framed by BEGIN and END statements:

ACDATA	Lists small-signal parameters vs. frequency. Framed by BEGIN ACDATA ... END. The ACDATA block is required in the S2PMDIF file.
NDATA	Lists noise parameters vs. frequency. Framed by BEGIN NDATA ... END. The NDATA block is optional in the S2PMDIF file.

Supported small-signal parameters The ACDATA (small-signal parameter) block supports the small-signal parameter types S, Y, Z, H, or G.

Supported noise data The NDATA (noise data) block supports parameters NFMIN (minimum noise figure), Gamma Opt (optimal source reflection coefficient), and RN (noise resistance). This is supported for use with all small-signal parameter types.

Option line The option line declares data contained in the SFC m2PMDIF file. These are

- The frequency units
- The type of small-signal parameter
- The format used to express the small signal parameters (ACDATA block) or optimum source reflection (NDATA) (Magnitude & Angle, Real & Imaginary, dB & Angle).

There are two option line formats:

#AC (freq_unit SS_ParmType SS_Parm_Format R Scaling/system impedance)

Advanced Design System 2008

freq_unit SS_ParmType SS_Parm_Format R Scaling/system

where:

freq_unit	=	Sets frequency units. Options are Hz, KHz, MHz, or GHz.
SS_ParmType	=	Sets small-signal parameter type. Options are: S, Y, Z, H, or G (default is S).
SS_Parm_format	=	Small signal parameter format. Options are RI, MA, or DB (default is RI), where: RI declares Real Imaginary Format MA declares Magnitude Angle DB is $20 * \log(\text{Parameter_Magnitude})$ Angle format
R Scaling/system	=	Declares scaling/system impedance (default is 50 ohms)

Important Option Line Information

- While both option line formats are supported, #AC(...) is recommended.
- Option lines are required in the file. If any are omitted, unpredictable results will occur.
- Option lines may have different frequency units, small-signal parameter formats (e.g., MA, RI, and DB), and system/scaling impedance values in the same S2PMDIF file. However, the small-signal parameter type (e.g., S, Y, Z, H, G) must be the same for all option lines in the S2PMDIF file. Agilent recommends that the same option line be used throughout the same S2PMDIF file except where scaling differences require changes (e.g., noise data is not scaled, whereas S-parameter data has a normalizing system impedance).
- If default SS_ParamType, SS_Param_format, and R Scaling/System are used, at a minimum, the freq_unit must appear in the option line. For example:
#AC (GHz) - preferred format
GHz - alternative format (Touchstone based)
- Option line syntax is case insensitive.
- When entering parameters other than S-parameter data, typically R 1 is used, since some users who are accustomed to using S-parameters make the common mistake of entering R 50 Z-parameters. In that event, the Z-parameters would be scaled to an undesirable level.
- If an option line is not specified within a data block in the file, the following default option line is used for that data: # hz R ri R 50 .

Signal Format Line The syntax for the signal format line is

% F n11x n11y n21x n21y n12x n12y n22x n22y

The signal format line comprises nine columns of data in the ACDATA block. The first column is frequency, the remaining columns pertain to small-signal parameters. In S2PMDIF, each two-port small-signal parameter is

Advanced Design System 2008

represented in two parts. The format being either Magnitude(x) and Angle(y), or Real(x) and Imaginary(y), or dB(x) and Angle(y). (See guideline for Option line above.) Given this, the small signal parameters are entered as eight columns of data. The format line has the following meaning where n is S, Y, Z, H, or G:

F	=	Frequency in Hz, kHz, MHz, or GHz.
n11x	=	Magnitude, real part, or dB value (depending which option is used) for small signal parameter n11.
n11y	=	Angle, or imaginary part (depending which option is used) for small signal parameter n11.
n21x	=	Magnitude, real part, or dB value (depending which option is used) for small signal parameter n21.
n21y	=	Angle, or imaginary part (depending which option is used) for small signal parameter n21.
n12x	=	Magnitude, real part, or dB value (depending which option is used) for small signal parameter n12.
n12y	=	Angle, or imaginary part (depending which option is used) for small signal parameter n12.
n22x	=	Magnitude, real part, or dB value (depending which option is used) for small signal parameter n22.
n22y	=	Angle, or imaginary part (depending which option is used) for small signal parameter n22.

When the small-signal parameter format is declared (Magnitude Angle, Real Imaginary, or dB Angle), all small-signal parameters assume that format. For example, if Magnitude Angle is declared, all small signal parameters assume magnitude and angle format. If Real Imaginary is declared, all parameters assume real and imaginary format. If MA, RI, or DB are not specified at all, the default format is RI.

Noise Format Line The syntax for the noise format line is

```
% F nfmin n11x n11y n
```

The noise format line comprises five columns of data in the NDATA block.

Advanced Design System 2008

F	=	Frequency in Hz, kHz, MHz, or GHz.
nfmin	=	Minimum noise figure.
n11x n11y	=	The third and forth columns are optimum source reflection coefficient described as either Magnitude (n11x) and Angle (n11y), Real (n11x) and Imaginary (n11y), or DB (n11x) and Angle (n11y). (See guideline for Option line above.)
rn	=	Equivalent normalized noise resistance.

In the option line, when MA, RI, or DB is declared, it only pertains to the optimum source reflection coefficient data. If MA, RI, or DB are not specified at all, the default format is RI.

Example S2PMDIF File

The following example is annotated using single small signal, noise parameter data pair.

```

! File using #AC ( ... ) Optionline
! Created Mon Jan 26 15:02:05 2004
VAR Wafer_Lot = 0
BEGIN ACDATA
#AC( hz S ma R 50 )
! Choice of units: Hz, KHz, MHz, or GHz
! Optional Selection: S,Z,Y,H,or G {default S}
! Optional Selection: MA, DB, RI {default RI}
!                               DB = 20*log(mag(value))
! Optional Selection: R XY; where XY=reference res {default R 50}
% F n11x n11y n21x n21y n12x n12y n22x n22y
!F      Mag_S11  Ang_S11   Mag_S21  Ang_S21   Mag_S12  Ang_S12   Mag_S22  Ang_S22
1e9    0.2416   -89.666   0.9138   -22.252   0.9138   -22.252   0.2366   -107.202
2e9    0.4456   -108.425  0.8336   -43.447   0.8336   -43.447   0.4394   -143.149
3e9    0.6068   -121.516  0.7234   -62.672   0.7234   -62.672   0.6044   -171.881
4e9    0.7203   -131.748  0.6068   -79.382   0.6068   -79.382   0.7259   -164.03
5e9    0.7945   -139.741  0.5001   -93.494   0.5001   -93.494   0.8097   143.997
END
BEGIN NDATA
! Noise Data block optional
#AC( hz S ma R 50 )
! Choice of units: Hz, KHz, MHz, or GHz
! Optional Selection: S,Z,Y,H,or G {default S}
! Optional Selection: MA, DB, RI {default RI}
!                               MA, DB, RI pertains to Gamma Opt data only
!                               DB = 20*log(mag(value))
! Optional Selection: R XY; where XY=reference res {default R 50}
% F NFMIN N11X N11Y RN
! Freq      nfmin_in_dB  GammaOpt_Mag  GammaOpt_Ang  Normalized R
1.0000000E9  0.1221             0.8026        29.711        0.1200
2.0000000E9  0.2440             0.6919        57.344        0.1200
3.0000000E9  0.3659             0.6515        80.598        0.1201
4.0000000E9  0.4878             0.6507        98.564        0.1201
5.0000000E9  0.6097             0.6671        111.954       0.1202
END

```

Advanced Design System 2008

```

! File using # freq_unit ... Optionline
! Created Mon Jan 26 15:02:05 2004
VAR Wafer_Lot = 0
BEGIN ACDATA
# hz S ma R 50
! Choice of units: Hz, KHz, MHz, or GHz
! Optional Selection: S,Z,Y,H,or G {default S}
! Optional Selection: MA, DB, RI {default RI}
!
!           DB = 20*log(mag(value))
! Optional Selection: R XY; where XY=reference res {default R 50}
% F n1lx n1ly n2lx n2ly n12x n12y n22x n22y
!F      Mag_S11  Ang_S11   Mag_S21  Ang_S21   Mag_S12  Ang_S12   Mag_S22  Ang_S22
1e9    0.2416   -89.666   0.9138   -22.252   0.9138   -22.252   0.2366   -107.202
2e9    0.4456   -108.425  0.8336   -43.447   0.8336   -43.447   0.4394   -143.149
3e9    0.6068   -121.516  0.7234   -62.672   0.7234   -62.672   0.6044   -171.881
4e9    0.7203   -131.748  0.6068   -79.382   0.6068   -79.382   0.7259   -164.03
5e9    0.7945   -139.741  0.5001   -93.494   0.5001   -93.494   0.8097   143.997
END
BEGIN NDATA
! Noise Data block optional
# hz S ma R 50
! Choice of units: Hz, KHz, MHz, or GHz
! Optional Selection: S,Z,Y,H,or G {default S}
! Optional Selection: MA, DB, RI {default RI}
!
!           MA, DB, RI pertains to Gamma Opt data only
!           DB = 20*log(mag(value))
! Optional Selection: R XY; where XY=reference res {default R 50}
% F NFMIN N11X N11Y RN
! Freq      nfmin_in_dB   GammaOpt_Mag   GammaOpt_Ang   Normalized R
1.0000000E9  0.1221                0.8026          29.711          0.1200
2.0000000E9  0.2440                0.6919          57.344          0.1200
3.0000000E9  0.3659                0.6515          80.598          0.1201
4.0000000E9  0.4878                0.6507          98.564          0.1201
5.0000000E9  0.6097                0.6671          111.954         0.1202
END

```

Additional Examples: ACDATA and NDATA Blocks

```

! Created Mon Jan 26 15:02:18 2004
VAR Wafer_Lot = 0
BEGIN ACDATA
#AC( hz S ri R 50 )
% F n1lx n1ly n2lx n2ly n12x n12y n22x n22y
      1e+009    0.00140798345    -0.241631115    0.845829604
-0.346084206    0.845829604    -0.346084206    -0.0699873389
-0.226051765
      2e+009    -0.140908488    -0.422961056    0.6052091
-0.573277416    0.6052091    -0.573277416    -0.351668311
-0.263572469
      3e+009    -0.317221301    -0.51732628    0.332133361
-0.642749288    0.332133361    -0.642749288    -0.598403728
-0.0853663052
      4e+009    -0.479677672    -0.537464874    0.111818135
-0.596499788    0.111818135    -0.596499788    -0.697916575
0.199718707
      5e+009    -0.606360921    -0.513473    -0.0304812178
-0.499151696    -0.0304812178    -0.499151696    -0.655090631
0.475990476
END
BEGIN NDATA
#AC( hz S ri R 1 )
% F NFMIN N11X N11Y RN
1.000000000000000000E9    1.22029516577356920E-1    6.97160538558048340E-1

```

Advanced Design System 2008

```

3.97731417445914650E-1    6.00019739208800920
2.00000000000000000E9    2.44034955394409670E-1    3.73350985510129000E-1
5.82545299649481410E-1    6.00078956835208110
3.00000000000000000E9    3.65992281414391130E-1    1.06430454557825050E-1
6.42814879525177040E-1    6.00177652879219800
4.00000000000000000E9    4.87877544837279590E-1    -9.69155972949636890E-2
6.43519718192936190E-1    6.00315827340834660
5.00000000000000000E9    6.09666923194363260E-1    -2.49450600814350180E-1
6.18811927554552810E-1    6.00493480220054730
END
VAR Wafer_Lot = 1
BEGIN ACDATA
#AC( hz S ri R 50 )
% F n1lx n1ly n2lx n2ly n12x n12y n22x n22y
1e+009    0.01333409179    -0.230137244    0.840070938
-0.340315654    0.840070938    -0.340315654    -0.0617617486
-0.229382577
2e+009    -0.120955165    -0.40456472    0.604746033
-0.565976614    0.604746033    -0.565976614    -0.345621191
-0.270746041
3e+009    -0.289098146    -0.497222626    0.335340188
-0.637454262    0.335340188    -0.637454262    -0.596941273
-0.0943544823
4e+009    -0.445587328    -0.518605287    0.116142331
-0.593841291    0.116142331    -0.593841291    -0.70077353
0.192167183
5e+009    -0.568522019    -0.496734037    -0.026397422
-0.498328865    -0.026397422    -0.498328865    -0.660073231
0.471463342
END
BEGIN NDATA
#AC( hz S ri R 1 )
% F NFMIN N11X N11Y RN
1.00000000000000000E9    1.71762793604862220E-1    6.97308317021682810E-1
3.02305313263392340E-1    6.9596391614503332
2.00000000000000000E9    3.43458465714362450E-1    4.15378270949377270E-1
4.48199090821574850E-1    6.9623162008013688
3.00000000000000000E9    5.15020130532548670E-1    1.85487069735727930E-1
5.06615547307641110E-1    6.9667779330530832
4.00000000000000000E9    6.86381371368581040E-1    5.35645741807940560E-3
5.20374575111159030E-1    6.9730243582054969
5.00000000000000000E9    8.57476469498637610E-1    -1.35072700985538610E-1
5.12375200485824140E-1    6.981055476258586
END
VAR Wafer_Lot = 2
BEGIN ACDATA
#AC( hz S ri R 50 )
% F n1lx n1ly n2lx n2ly n12x n12y n22x n22y
1e+009    0.0261985778    -0.23292849    0.822455256
-0.341254054    0.822455256    -0.341254054    -0.0505770288
-0.23686219
2e+009    -0.112333318    -0.406095322    0.583604607
-0.563040028    0.583604607    -0.563040028    -0.340993754
-0.278957576
3e+009    -0.282374986    -0.494226554    0.314667444
-0.627999778    0.314667444    -0.627999778    -0.594911193
-0.0996720908
4e+009    -0.437476982    -0.511079142    0.0996827436
-0.579649532    0.0996827436    -0.579649532    -0.698293478
0.188345791
5e+009    -0.557310676    -0.486460159    -0.0377810523
-0.482462248    -0.0377810523    -0.482462248    -0.657181646
0.467271981
END
BEGIN NDATA
#AC( hz S ri R 1 )
% F NFMIN N11X N11Y RN
1.00000000000000000E9    2.20723603091905400E-1    6.88702175738749210E-1
3.10193703295093660E-1    8.7252393655197622

```

Advanced Design System 2008

```

2.0000000000000000E9      4.41304833276760710E-1      4.00100485663632010E-1
4.56013841018826490E-1      8.7307131145790162
3.0000000000000000E9      6.61602142031938460E-1      1.66985293710630910E-1
5.11697368308757120E-1      8.7398360296778144
4.0000000000000000E9      8.81475616395049320E-1      -1.39839699776407360E-2
5.22456153286954630E-1      8.75260811081613
5.0000000000000000E9      1.10078776413520350E0      -1.53929731170780970E-1
5.11924800755568120E-1      8.7690293579939471
END
VAR Wafer_Lot = 3
BEGIN ACDATA
#AC( hz S ri R 50 )
% F n1lx n1ly n2lx n2ly n12x n12y n22x n22y
      1e+009      -0.111624701      -0.405714686      0.66636886
-0.469619205      0.66636886      -0.469619205      -0.191986725
-0.329432821
      2e+009      -0.41908962      -0.512855503      0.292032657
-0.563375528      0.292032657      -0.563375528      -0.569338122
-0.206257358
      3e+009      -0.620672222      -0.472453429      0.0607572371
-0.47694961      0.0607572371      -0.47694961      -0.718017693
0.101776232
      4e+009      -0.732902391      -0.404898014      -0.0522793425
-0.36498764      -0.0522793425      -0.36498764      -0.69113052
0.390569198
      5e+009      -0.795560017      -0.34381194      -0.101065286
-0.269885315      -0.101065286      -0.269885315      -0.576309143
0.608680065
END
BEGIN NDATA
#AC( hz S ri R 1 )
% F NFMIN N11X N11Y RN
1.0000000000000000E9      5.18703037517405720E-1      2.56297603514494640E-1
5.77469208317883620E-1      9.5231179096495424
2.0000000000000000E9      1.03556771684315740E0      -2.06911574434137120E-1
5.86373914380050380E-1      9.5447677385981624
3.0000000000000000E9      1.54881350043704070E0      -4.46640278565243860E-1
5.11094686752464700E-1      9.5808507868458594
4.0000000000000000E9      2.05677054388418230E0      -5.81970283235218930E-1
4.38893368283120290E-1      9.6313670543926495
5.0000000000000000E9      2.55792347993451760E0      -6.65702281631675600E-1
3.80285542816346120E-1      9.6963165412384953
END
VAR Wafer_Lot = 4
BEGIN ACDATA
#AC( hz S ri R 50 )
% F n1lx n1ly n2lx n2ly n12x n12y n22x n22y
      1e+009      -0.0997555774      -0.381004445      0.705619581
-0.459619141      0.705619581      -0.459619141      -0.188617047
-0.322113995
      2e+009      -0.387429707      -0.50652345      0.338299265
-0.580037701      0.338299265      -0.580037701      -0.568533379
-0.21853959
      3e+009      -0.59300033      -0.48258943      0.0913926446
-0.508488094      0.0913926446      -0.508488094      -0.734058749
0.0900874159
      4e+009      -0.71385299      -0.421239785      -0.0364444847
-0.397584211      -0.0364444847      -0.397584211      -0.713634507
0.390775305
      5e+009      -0.783429944      -0.361124669      -0.0944553209
-0.298630566      -0.0944553209      -0.298630566      -0.595965263
0.61920336
END
BEGIN NDATA
#AC( hz S ri R 1 )
% F NFMIN N11X N11Y RN
1.0000000000000000E9      4.07278952069115260E-1      3.20981863953611680E-1
5.26706311842366580E-1      7.6643738931871832
2.0000000000000000E9      8.13665870677917270E-1      -1.24308690147576310E-1

```

```

5.64368849214357040E-1    7.6773474702487121
3.0000000000000000E9      1.21828615365247780E0    -3.71333282289684650E-1
5.08578990984615050E-1    7.6989700986845886
4.0000000000000000E9      1.62029912735584050E0    -5.17462235668050940E-1
4.46062394082844540E-1    7.7292417784948277
5.0000000000000000E9      2.01891277231332160E0    -6.10837572225724480E-1
3.92061347849876940E-1    7.7681625096794225
END
VAR Wafer_Lot = 5
BEGIN ACDATA
#AC( hz S ri R 50 )
% F n1lx n1ly n2lx n2ly n12x n12y n22x n22y
      1e+009      -0.148708863      -0.411149613      0.661371247
-0.487106054    0.661371247      -0.487106054      -0.245385025
-0.343281312
      2e+009      -0.459669272      -0.498617399      0.27661003
-0.561103773    0.27661003      -0.561103773      -0.633354256
-0.183147263
      3e+009      -0.65008453      -0.447693986      0.054820678
-0.464075002    0.054820678      -0.464075002      -0.761781857
0.147962113
      4e+009      -0.751951738      -0.378576048      -0.0489188832
-0.35214127     -0.0489188832     -0.35214127     -0.712725662
0.441632279
      5e+009      -0.807697362      -0.319231466      -0.0924561924
-0.260818351    -0.0924561924    -0.260818351    -0.581417298
0.655785284
END
BEGIN NDATA
#AC( hz S ri R 1 )
% F NFMIN N11X N11Y RN
1.0000000000000000E9      4.82467789743262190E-1    2.28415357401098710E-1
5.09316970466515430E-1    7.1489835897047316
2.0000000000000000E9      9.63454968428843020E-1    -2.06744364689974120E-1
5.11947374976623950E-1    7.1696884038188884
3.0000000000000000E9      1.44152132366022330E0      -4.32828356211392770E-1
4.47558248165678220E-1    7.2041964273425005
4.0000000000000000E9      1.91530443304156290E0      -5.63042210953368820E-1
3.86155387977519470E-1    7.2525076602755627
5.0000000000000000E9      2.38355146947575140E0      -6.45263220258679840E-1
3.36048596462367750E-1    7.3146221026180545
END

```

P2D Format

The large-signal or power-dependent S-parameter (.p2d) file is a system input file you create from an S-parameter file, inserting the MDIF format. You use the .p2d file to characterize the component by a complete set of 2-port large-signal S-parameters, accounting for power dependence of S₂₁, S₁₁, S₂₂, and S₁₂, plus optional noise data, and optional intermodulation data.

A .p2d file can also be created via simulation by placing a P2D simulation component from the LSSP Simulation Library in the design.

It is possible to have multi-dimensional P2D files with VAR statements separating individual basic P2D sections. Such files may be automatically generated using the AmplifierP2D_Setup component from the System-Data Models library.

Advanced Design System 2008

The format for a basic P2D section is shown here. Required keywords appear in UPPERCASE ITALIC characters.

```
!!! Begin basic P2D syntax
BEGIN ACDATA !!! required 2-port S-parameter data block
..... ( Required small signal section identical to ACDATA section of S2D format )
..... ( Optional large signal section - shown in the next section )
END ACDATA
BEGIN NDATA !!! optional 2-port noise data block
....
END NDATA
BEGIN IMTDATA !!! optional intermodulation table
....
END IMTDATA
!!! End basic P2D syntax
```

This format can be expanded to form a multi-dimensional P2D file using VAR statements. For instance a single P2D file containing two temperature points over three bias points contains a total of six basic sections as shown. Note that each sequence of VAR statements should preserve the order of the sweep, in this case temperature over bias and each sweep variable, for example, bias has its values arranged in monotonically increasing order. In this example $B1 < B2 < B3$ and $T1 < T2$. Required keywords appear in UPPERCASE ITALIC characters.

```
VAR temp=T1
VAR bias=B1
..... ( 1st Basic P2D section )
VAR temp=T1
VAR bias=B2
..... ( 2nd Basic P2D section )
VAR temp=T1
VAR bias=B3
..... ( 3rd Basic P2D section )
VAR temp=T2
VAR bias=B1
..... ( 4th Basic P2D section )
VAR temp=T2
VAR bias=B2
..... ( 5th Basic P2D section )
VAR temp=T2
VAR bias=B3
..... ( 6th Basic P2D section )
```

Guidelines for .p2d

Advanced Design System 2008

- Basic MDIF syntax contains four reserved words. VAR begins an independent variable definition line, in the form VAR <name> = <value>. BEGIN <blockname> signals the beginning of a data block, and END signals the conclusion of a data block. A line beginning with REM or the comment symbol (!) will be assumed as comments.
- VAR statements are used to specify multidimensional data, that is, two or more independent variables. The value of a VAR statement can be a number.
- Multiple sets of data (ACDATA, NDATA, IMTDATA) can be used with VAR statements used before or after any data set.
- The file data set is made up of data blocks, each separated by BEGIN and END statements. Three different types of data blocks are allowed:

ACDATA	=	Lists small and large-signal parameters vs. frequency and power (required)
NDATA	=	Lists noise parameters vs. frequency (optional)
IMTDATA	=	Used for a 2-port frequency converter to give the single-tone intermodulation table (optional)

The ACDATA Block

The ACDATA block allows you to specify the small- and large-signal characteristics of the 2-port.

- General format:
 BEGIN ACDATA
 # AC (.....) ! this is the option line
 % ! this is a format line ___
 small-signal data goes here
 % F ! this is a format line
 ... first large-signal data frequency
 % P1 P2 ! this is a format line
 ... large-signal data at first frequency
 % F ! next large-signal frequency
 ... second large-signal data frequency
 % P1 P2
 ... large-signal data at second frequency
 ...
 ... additional sets of large- signal frequency and data
 ...
 END
- Option line:
 # AC(unit parm_type parm_format R xx FC m b)
 where:

unit	=	Sets the frequency unit. Options are HZ, KHZ, MHZ, or GHZ.
------	---	---

parm_type	=	Only S-parameters are allowed; use S only.
parm_format	=	MA, DB, RI, VDB sets the format for the four network parameters, where: MA declares magnitude and angle (degrees) DB is for $20 \cdot \log_{10}(\text{MA})$ and angle (degrees) RI declares real and imaginary VDB declares n11 and n22 as VSWR and angle (degrees) and n21 and n12 as DB and angle (degrees)
R xx	=	Declares resistance, where xx = normalization resistance.
FC m b	=	Declares input to output frequency conversion where $F_{out} = m \cdot F_{in} + b$; where: m is for frequency multiplication b is for frequency translation

- Small-signal format line:

% F n11x n11y n21x n21y n12x n12y n22x n22y

This line gives the order for the data in the lines to follow. All of the keywords shown must be given in the format line. The order of these keywords is arbitrary. The order shown is preferred.

F	=	For frequency data
n11x, n11y	=	The 11 data pair for the 2 port data matrix
n21x, n21y	=	The 21 data pair
n12x, n12y	=	The 12 data pair
n22x, n22y	=	The 22 data pair

$$\begin{bmatrix} n11 & n12 \\ n21 & n22 \end{bmatrix}$$

If the parameter type = S, and the parameter format = DB, then
n11x = |S11| in dB, and n11y = angle of S11 in degrees.

- Large-signal format line:

% F

This line precedes the frequency for the following large-signal data.

% P1 P2 n11x n11y n21x n21y n12x n12y n22x n22y

This line precedes the large-signal data. The large-signal data may have up to 101 sets of power data per frequency.

All of the keywords shown must be given in the format line. While the order of keywords is arbitrary, the order shown is preferred.

Advanced Design System 2008

P1	=	The power (dBm) incident at port 1 with port 2 terminated in R ohms, for the measurement of S11 and S21
P2	=	The power (dBm) incident at port 2 with port 1 terminated in R ohms, for the measurement of S22 and S12

In the following data, a value of 1000 (1.e3) for the P2 data indicates that the S22 and S12 data are for small-signal measurement.

n11x, n11y	=	The 11 data pair for the 2 port data matrix
n21x, n21y	=	The 21 data pair
n12x, n12y	=	The 12 data pair
n22x, n22y	=	The 22 data pair

$$\begin{bmatrix} n11 & n12 \\ n21 & n22 \end{bmatrix}$$

If the parameter type = S, and the parameter format = DB, then n11x = |S11| in dB, and n11y = angle of S11 in degrees.

ACDATA Block Examples

2-port with 50-ohm S-parameters:

```

BEGIN ACDATA
# AC( GHZ S DB R 50 FC 1.0 0.0 )
% F n11x n11y n21x n21y n12x n12y n22x n22y
! RF-freq S11-db S11-deg S21-dB S21-deg S12-dB S12-deg S22-db S22-deg
1.0000 -15 45 8 25 -20 -15 -12 10
2.0000 -16 25 9 30 -20 -15 -12 20
3.0000 -17 -10 10 35 -20 -15 -11 30
% F
1.00
% P1 P2 n11x n11y n21x n21y n12x n12y n22x n22y
-15.000 -5.000 ...
-5.000 5.000 ... large-signal S-parameter data here ....
5.020 15.000 ...
% F
2.00
% P1 P2 n11x n11y n21x n21y n12x n12y n22x n22y
-15.000 -5.000 ...
-5.000 5.000 ... large-signal S-parameter data here ...
5.020 15.000 ...
END
    
```

2-port frequency converter with variable RF freq, variable LO freq, fixed IF freq, fixed LO power, and RF-to-IF 2-port 50-ohm S-parameters:

```

BEGIN ACDATA
# AC( GHZ VDB R 50 FC 0 0.2 ) ! this gives a constant IF of 0.2 GHz
% F n11x n11y n21x n21y n12x n12y n22x n22y
! RF-freq S11-vswr S11-deg S21-dB S21-deg S12-dB S12-deg S22-vswr S22-deg
1.0000 1.2 0 -8 0 -20 0 1.3 0
2.0000 1.3 0 -9 0 -20 0 1.2 0
3.0000 1.4 0 -10 0 -20 0 1.3 0
% F
1.00
% P1 P2 n11x n11y n21x n21y n12x n12y n22x n22y
-15.000 -5.00 ...
-5.000 5.00 ... large-signal S-parameter data here ....
5.020 15.00 ...
% F
2.00
% P1 P2 n11x n11y n21x n21y n12x n12y n22x n22y
-15.000 -5.00 ...
-5.000 5.00 ... large-signal S-parameter data here ...
5.020 15.00 ...
END
    
```

2-port frequency converter with variable RF freq, fixed LO freq, variable IF freq, fixed LO power, and RF-to-IF 2 port 50-ohm S-parameters:

```

BEGIN ACDATA
# AC( GHZ DB R 50 FC -1 4 ) ! this gives an IF = 4 - RF
% F n11x n11y n21x n21y n12x n12y n22x n22y
! RF-freq S11-dB S11-deg S21-dB S21-deg S12-dB S12-deg S22-dB S22-deg
1.0000 -12 0 -8 0 -20 0 -13 0
2.0000 -13 0 -9 0 -20 0 -12 0
3.0000 -14 0 -10 0 -20 0 -13 0
% F
1000.00
% P1 P2 n11x n11y n21x n21y n12x n12y n22x n22y
-15.000 -5.00 ...
-5.000 5.00 ... large-signal S-parameter data here ....
5.020 15.00 ...
% F
2000.00
% P1 P2 n11x n11y n21x n21y n12x n12y n22x n22y
-15.000 -5.00 ...
-5.000 5.00 ... large-signal S-parameter data here ...
5.020 15.00 ...
END
    
```

The NDATA Block

The NDATA block allows the user to specify the small-signal noise characteristics of the 2-port.

- General format:
BEGIN NDATA
AC (.....) ! this is the option line

% ! this is the format line
 data goes here
 END

- Option line:
 # AC(freq_unit parm_type parm_format R xx)
 where:

freq_unit	=	Sets the frequency unit. Options are HZ, KHZ, MHZ, or GHZ.
parm_type	=	S sets the source noise match type to optimum source reflection coefficient.
parm_format	=	MA, DB, RI sets the format for the optimum source match, where: MA declares magnitude and angle (degrees) DB declares $20 \cdot \log_{10}(MA)$ and angle (degrees) RI declares real and imaginary
R xx	=	Declares resistance, where xx = normalization resistance for the source match and rn.

- Format line:
 % F nfmin n11x n11y rn
 This line gives the order for the data in the lines to follow. All of the keywords shown must be given in the format line. While the order of keywords is arbitrary, the order shown is preferred

F	=	For frequency data
nfmin	=	For minimum noise figure data
n11x, n11y	=	For the optimum source impedance for minimum noise figure
rn	=	For the equivalent input normalized noise resistance. The system simulator requires this parameter to meet physical requirements. If the user-supplied rn value is less than allowed for this requirement, then the system simulator will force this rn value to the lowest physical limit.

The IMTDATA Block

The IMTDATA data block allows you to specify for a 2-port frequency converter the single tone output intermodulation

levels with respect to the fundamental output tone.

- General format:

```
BEGIN
reference_signal_power reference_LO_power
..... IMT data goes here
END
```

This data is given for specific LO and RF input power levels. However, during a system simulator spurious signal analysis, the data is adjusted for the actual converter input signal and LO power levels.

- Example

```
BEGIN IMTDATA
! Intermodulation table for double balanced mixer #1
! Reference Signal Level (dBm) Reference LO Level (dBm)
# -10 7
! M x LO (Horizontal) N x Signal (Vertical)
% 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
!
99 26 35 39 50 41 53 49 51 45 65 55 75 65 85 99
24 0 35 13 40 24 45 28 49 35 55 45 65 55 99
73 73 74 70 71 64 69 64 69 65 75 75 85 99
67 64 69 50 77 47 74 44 74 45 75 55 99
86 90 86 88 88 85 86 85 90 85 85 99
90 80 90 71 90 68 90 65 88 65 99
90 90 90 90 90 90 90 90 90 99
90 90 90 90 90 87 90 90 99
99 95 99 95 99 95 99 99
90 95 90 95 90 99 99
99 99 99 99 99 99
90 99 99 99 99
99 99 99 99
99 99
99
END
```

In the IMT table:

- The vertical row number, N, (0, 1, to 15) indicates the harmonic of the signal used in deriving the spurious output signal.
- The horizontal column number, M, (0, 1, to 15) indicates the harmonic of the local oscillator used in deriving the spurious output signal.
- In row 2, column 4, the data is 13. This means that for an input signal at -10 dBm input, with an LO drive of +7dBm, an output spurious signal will occur at 3*LO + 1*signal, with a level that is 13 dB below the fundamental output signal.
- If the input signal differs from the -10 dBm reference power level listed at the top of the table by X dB, then the number in the table is adjusted by adding (N-1)*X dB to it. This manner of adjustment is good for input power levels up to 5 dB greater than the reference signal power.
- If the local oscillator signal differs from the +7 dBm reference power level listed at the top of the table by X dB, then the number in the table is adjusted by adding it by M*X dB to it. This manner of adjustment is good for local oscillator power levels from the reference level minus 10 dB to the reference level plus 3 dB.
- The data values must fall in the range of 0 to 99. Numbers outside this range will cause an error.
- The "#IMT ()" is optional. Signal and LO reference power levels can be listed without this.
- IMT data can be in square or triangular format.

Example .p2d File

```

! -----
! ampp2d.p2d
! -----
! This is a sample P2D data file containing an ACDATA block, an NDATA block,
! and an IMTDATA block.
! -----
!
BEGIN ACDATA
! This line and all lines starting with the comment (!) character are ignored
! Do not have a blank line or comments as the first line in this file
! In a P2D file ACDATA is a required block of data. It may have one or two
! sections.
! (a) Required - Full 2-port small signal S-parameters vs frequency
! (b) Optional - Full 2-port large signal S-parameters vs (frequency X power)
! The small signal data must precede the large signal data in this block.
# AC( GHZ  S  RI  R 50.0  FC  1.  0. )
! Optional Selection: HZ, KHZ, MKZ, or GHZ; Default is GHZ
! Optional Selection: S, Z, Y, H, or G; Default is S
! Optional Selection: MA, DB, or RI; Default is RI
! Optional Selection: R xx; where xx = reference resistance; Default R 50.0
! Optional Selection: FC x1 x2 is for frequency conversion: Fout=x1*Fin + x2
! Default is Fout = Fin
% F n1lx n1ly n2lx n2ly n12x n12y n22x n22y
! The above line is the format line showing the order in the following data
! lines
! The columns of data can be in any order
1.0000  0.3926  -0.1211  -0.0003  -0.0021  -0.0003  -0.0021  0.3926  -0.1211
2.0000  0.3517  -0.3054  -0.0096  -0.0298  -0.0096  -0.0298  0.3517  -0.3054
3.0000  0.0430  -0.5916  -2.6933  -0.1433  -0.5933  -0.1433  0.0430  -0.5916
4.0000  0.4071  -0.2756  2.4617  0.6234  0.3617  0.4234  0.4071  -0.2756
5.0000  0.2041  0.2880  2.6848  -0.5367  0.3848  -0.4367  0.2041  0.2880
6.0000  0.5666  0.0343  2.0383  -0.7437  0.0383  -0.7437  0.5666  0.0343
7.0000  0.0430  0.6916  -2.6933  0.1433  -0.6933  0.1433  0.0430  0.6916
8.0000  0.3059  0.5659  -0.1000  0.1424  -0.1000  0.1424  0.3059  0.5659
9.0000  0.3071  0.4145  -0.0307  0.0673  -0.0307  0.0673  0.3071  0.4145
10.0000 0.3419  0.3336  -0.0134  0.0379  -0.0134  0.0379  0.3419  0.3336
! This is the end of the small signal ACDATA section
!
! This is the beginning of the large signal ACDATA section
% F
1.00
% P1 P2 n1lx n1ly n2lx n2ly n12x n12y n22x n22y
-15.00 -25.00 0.3926 -0.1211 -0.0003 -0.0021 -0.0003 -0.0021 0.3926 -0.1211
-10.00 -20.00 0.3826 -0.1221 -0.0004 -0.0022 -0.0003 -0.0021 0.3926 -0.1211
-5.00 -15.00 0.3726 -0.1231 -0.0007 -0.0029 -0.0003 -0.0021 0.3926 -0.1211
0.00 -10.00 0.3856 -0.1245 -0.0010 -0.0129 -0.0003 -0.0021 0.3926 -0.1211
% F
2.00
% P1 P2 n1lx n1ly n2lx n2ly n12x n12y n22x n22y
-15.00 -25.00 0.3517 -0.3054 -0.0096 -0.0298 -0.0096 -0.0298 0.3517 -0.3054
-10.00 -20.00 0.3517 -0.3154 -0.0098 -0.0298 -0.0096 -0.0298 0.3517 -0.3054
-5.00 -15.00 0.3517 -0.3254 -0.0104 -0.0298 -0.0096 -0.0298 0.3517 -0.3054
0.00 -10.00 0.3517 -0.3354 -0.0106 -0.0298 -0.0096 -0.0298 0.3517 -0.3054
! ... (more frequencies and power sweeps under each frequency may be added)
! This is the end of the small signal ACDATA section
END
BEGIN NDATA
! This is an optional block of data
# AC( GHZ  RI  S  R 50.0 )
! Optional Selection: HZ, KHZ, MKZ, or GHZ; Default is GHZ
! Optional Selection: S, Z, Y, H, or G; Default is S
! Optional Selection: MA, DB, or RI; Default is RI
! Optional Selection: R xx; where xx = reference resistance; Default R 50.0
% F nfmin n1lx n1ly rn
! The above line is the format line showing the order in the following data
! lines

```

```

! The columns of data can be in any order
1.0000 2.0000 0.3926 -0.1211 .4
2.0000 2.5000 0.3517 -0.3054 .45
3.0000 3.0000 0.0430 -0.5916 .5
4.0000 3.5000 0.4071 -0.2756 .55
5.0000 4.0000 0.2041 0.2880 .6
6.0000 4.5000 0.5666 0.0343 .65
7.0000 5.0000 0.0430 0.6916 .7
8.0000 5.5000 0.3059 0.5659 .75
9.0000 6.0000 0.3071 0.4145 .8
10.0000 6.5000 0.3419 0.3336 .85
END
BEGIN IMTDATA
! This is an optional block and may be used some mixer component if the file
! format is supported.
! Intermodulation table for double balanced mixer #1
! Signal Level (dBm) LO Level (dBm)
# -10 7
! M x LO ( Horizontal ) N x Signal (Vertical )
! 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
!
99 26 35 39 50 41 53 49 51 42 62 51 60 47 77 50
24 0 35 13 40 24 45 28 49 33 53 42 60 47 63
73 73 74 70 71 64 69 64 69 62 74 62 72 60
67 64 69 50 77 47 74 44 74 47 75 44 70
86 90 86 88 88 85 86 85 90 85 85 85
90 80 90 71 90 68 90 65 88 65 85
90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 87 90 90 90
99 95 99 95 99 95 99 95
90 95 90 90 90 99 90
99 99 99 99 99 99
90 99 90 95 90
99 99 99 99
90 99 90
99 99
99
END
!
! -----

```

S2D Format

The S-parameter Data (.s2d) file is a system input file you create from an S-parameter file, inserting the MDIF format. In the .s2d file, you describe small-signal data, optional noise data, optional nonlinear data, and optional intermodulation data. You can describe nonlinearity as a function of drive power or nonlinear parameter consisting of some combination of third-order intercept point, 1dB gain compression, saturated output power, and gain compression at saturation. In ADS, it possible to have multidimensional S2D files with VAR statements separating individual basic S2D sections. Such files may be automatically generated using the AmplifierS2D_Setup component from the System-Data Models library.

The format for a basic S2D section is shown here. Required keywords appear in UPPERCASE ITALIC characters.

```

!!! Begin basic S2D syntax
BEGIN ACDATA !!! optional small-signal data block
....

```

```
END ACDATA
BEGIN GCOMPx !!! required compression information x=(1,...,7)
!!! only one type of GCOMPx supported in one S2D file
....
END GCOMPx
BEGIN NDATA !!! optional 2-port noise data block
....
END NDATA
BEGIN IMTDATA !!! optional intermodulation table
....
END IMTDATA
!!! End basic S2D syntax
```

This format can be expanded to form a multi-dimensional S2D file using VAR statements in ADS. For instance a single S2D file containing two temperature points over three bias points contains a total of six basic sections as shown. Note that each sequence of VAR statements should preserve the order of the sweep, in this case, temperature over bias and each sweep variable. For example, bias has its values arranged in monotonically increasing order. In this example $B1 < B2 < B3$ and $T1 < T2$. Required keywords appear in UPPERCASE ITALIC characters:

```
VAR temp=T1
VAR bias=B1
..... ( 1st basic S2D section )
VAR temp=T1
VAR bias=B2
..... ( 2nd basic S2D section )
VAR temp=T1
VAR bias=B3
..... ( 3rd basic S2D section )
VAR temp=T2
VAR bias=B1
..... ( 4th basic S2D section )
VAR temp=T2
VAR bias=B2
..... ( 5th basic S2D section )
VAR temp=T2
VAR bias=B3
..... ( 6th basic S2D section )
```

Guidelines for .s2d

- Basic MDIF syntax contains four reserved words:
 - VAR begins an independent variable definition line, in the form VAR <name> = <value>.
 - BEGIN <blockname> signals the beginning of a data block.

Advanced Design System 2008

- END signals the conclusion of a data block.
- REM or the comment symbol (!) at the beginning of a line signifies a comment.
- VAR statements are used to specify multidimensional data, that is, two or more independent variables. The value of a VAR statement can be a number.
- Comments can be inserted on any line within the file and must be preceded by a comment symbol (!).
- Multiple sets of data (ACDATA, NDATA, GCOMP, IMTDATA) can be used with VAR statements before or after any data set. Note that only one type of GCOMP block can be used in a single S2D file, for instance do not create an S2D file containing one GCOMP3 and one GCOMP5 block.
- Each block of ACDATA, NDATA, GCOMPx can be headed by an option line such as #ACDATA (MHz S RI R 50) to allow scaling and type definition for the network and its parameters.
- The file data set is made up of data blocks, each separated by BEGIN and END statements. The following ten different types of data blocks are allowed:

ACDATA	Lists small-signal network parameters vs. frequency (required)
NDATA	Lists noise parameters vs. frequency (optional)
GCOMP1	Lists output 3rd order intercept (IP3) (optional)
GCOMP2	Lists output 1dB gain compression power (IDBC) (optional)
GCOMP3	Lists output IP3 and 1DBC (optional)
GCOMP4	Lists output IP3 and output saturation power (PS) and gain compression at saturation (GCS) (optional)
GCOMP5	Lists output 1DBC, PS, and GCS (optional)
GCOMP6	Lists output IP3, 1DBC, PS, and GCS (optional)
GCOMP7	Lists S21 as a function of input power at a single frequency (optional)
IMTDATA	Used for a 2-port frequency converter to give the single-tone intermodulation table (optional)

- One or more spaces or tabs separate entries on a line. Names can be mixed case-the file reader will store them as lowercase.
- Multi-dimensional IMT tables are not supported. Only one IMT table may be present in a single S2D if necessary.

The ACDATA Block

The ACDATA block allows you to specify the small-signal characteristics of the 2-port.

- General format:
 BEGIN ACDATA
 # AC (.....) ! this is the option line
 % ! this is the format line
 data goes here

Advanced Design System 2008

END

- Option line syntax:
`# AC(freq-unit parm-type parm-format R xx FC m b)`
 where:

freq-unit	=	Sets the frequency units. Options are: HZ, KHZ, MHZ, GHZ.
parm-type	=	Sets the 2-port network parameter option using S, Z, Y, H, or G.
parm-format	=	Sets the format for the four network parameters using MA, DB, RI, VDB, where: MA declares magnitude and angle (degrees) DB is $20 \cdot \log_{10}(\text{MA})$ and angle (degrees) RI declares real and imaginary VDB declares n11 and n22 as VSWR and angle (degrees) and n12 and n21 as DB and angle (degrees)
R xx	=	Declares resistance, where xx = normalization resistance
FC m b	=	Declares input to output frequency conversion such that $F_{out} = m \cdot F_{in} + b$ where: m is for frequency multiplication b is for frequency translation

- Example:
`#AC (MHZ S MA R 50 FC 0 30)`
 would set the frequency units to MHZ, 2-port parameters to S, 2-port parameter format to magnitude and angle, reference resistance to 50 ohms, frequency conversion to 30 MHZ of constant output frequency.

FC 1 0	Sets the output frequency equal to the input frequency ($F_{out} = 1 \cdot F_{in} + 0$)
FC -1 30	Sets the output frequency to 30 minus the input frequency ($F_{out} = -1 \cdot F_{in} + 30$)

- Format line:
`% F n11x n11y n21x n21y n12x n12y n22x n22y`
 This line gives the order for the data in the lines to follow. All of the keywords shown must be given in the format line. While the order of keywords is arbitrary, the order shown is preferred.

F	=	Frequency data column
n11x, n11y	=	The 11 data pair for the 2 port data matrix

n21x, n21y	=	The 21 data pair
n12x, n12y	=	The 12 data pair
n22x, n22y	=	The 22 data pair

where these data pairs belong to the 2-port characterization matrix:

$$\begin{bmatrix} n11 & n12 \\ n21 & n22 \end{bmatrix}$$

- Examples

The following examples illustrate the ACDATA block with various data options:

2-port with 50-ohm S-parameters:

```
BEGIN ACDATA
# AC ( GHZ S DB R 50 FC 1 0 )
% F n11x n11y n21x n21y n12x n12y n22x n22y
! RF-freq S11-dB S11-deg S21-dB S21-deg S12-dB S12-deg S22-dB S22-deg
1.0000 -15 45 -8 25 -20 -15 -12 10
2.0000 -16 25 -9 30 -20 -15 -12 20
3.0000 -17 -10 -10 35 -20 -15 -11 30
END
```

2-port frequency converter with variable RF freq, fixed LO freq, variable IF freq, fixed LO power, and RF-to-IF 2-port 50-ohm S-parameters:

```
BEGIN ACDATA
# AC( GHZ S DB R 50 FC -1 4 ) ! this gives an
! IF = 4 - RF
% F n11x n11y n21x n21y n12x n12y n22x n22y
! RF-freq S11-dB S11-deg S21-dB S21-deg S12-dB S12-deg S22-dB S22-deg
1.0000 -12 0 -8 0 -20 0 -13 0
2.0000 -13 0 -9 0 -20 0 -12 0
3.0000 -14 0 -10 0 -20 0 -13 0
END
```

The NDATA Block

The NDATA block allows you to specify the small-signal noise characteristics of the 2-port.

- General format:

```
BEGIN NDATA
# AC ( ..... ) ! this is the option line
% .... ! this is the format line
..... data goes here
END
```

- Option line syntax:

AC(freq-unit parm-type parm-format R xx)
 where:

freq-unit	=	HZ, KHZ, MHZ, or GHZ sets the frequency units.
parm-type	=	S only, source reflection coefficient.
parm-format	=	MA, DB, RI sets the format for optimum source match, where: MA declares magnitude and angle (degrees) DB declares $20 \cdot \log_{10}(\text{MA})$ and angle (degrees) RI declares real and imaginary
R xx	=	Declares resistance where xx = normalization resistance for the source match and noise resistance.

- Format line syntax:

% F nfmin n11x n11y rn

This line gives the order for the data in the lines to follow. All of the keywords shown must be given in the format line. While the order of keywords is arbitrary, the order shown is preferred.

F	=	Frequency data column.
nfmin	=	For minimum noise figure data.
n11x, n11y	=	For the optimum source reflection coefficient for minimum noise figure.
rn	=	For the equivalent normalized input noise resistance of the 2-port. The system simulator requires this parameter to meet physical requirements. If the user-supplied rn value is less than allowed for this requirement, then the system simulator will force this rn value to the lowest physical limit.

For more information on noise figure, see [S-Parameter Simulation Noise Analysis](#).

The following is an example of the NDATA block:

```

BEGIN NDATA
# ACDATA ( GHz S RI R 50 )
% F nfmin n11x n11y rn
1.0000 2.0000 -0.1211 -0.0003 .4
2.0000 2.5000 -0.3054 -0.0096 .45
3.0000 3.0000 -0.6916 -0.6933 .5
END
    
```

Understanding GCOMP Data

There are seven mutually exclusive formats for expressing large signal response in an S2D file, each corresponding to one type of GCOMP block. GCOMP stands for gain compression, indicating that only forward transmission behavior of the device under large signal conditions is captured here. Each GCOMP section may optionally contain multiple profiles, each at a different gain compression frequency. Various nonlinear device models refer to these frequencies as GCFreq or GainCompFreq.

GCOMP1 through GCOMP6 use parametric specifications whereas GCOMP7 uses a data-based profile. As such, the first six types of GCOMP blocks are restricted in the modeling capability of nonlinear behavior whereas GCOMP7 can represent any arbitrary response including gain expansion regions.

GCOMP1 through GCOMP6 use various combinations of the following four standard measures of nonlinear behavior, all referenced to the abscissa of an output (dBm) versus input power (dBm) plot in:

- IP3 - Third order intercept point in dBm, usually referenced to output power axis. This is a theoretical point where the power of the fundamental at the output of the nonlinear device would have equalled that of the third harmonic if there were no expansion or compression effects at high input drives.
- 1DBC - 1 dB compression point in dBm, usually referenced to output power axis. This parameter marks the onset of nonlinear behavior and refers to the point where actual output power at the fundamental tone is 1 dB below the predicted linear output power.
- PS - Power at saturation in dBm refers to the maximum possible output power at fundamental frequency under normal operating conditions, that is, prior to breakdown due to high input drive.
- GCS - Gain compression at saturation in dB, refers to the amount of compression with respect to linear behavior at the onset of saturation of the output fundamental frequency.

System level components that can interpret S2D profiles use an odd-order polynomial fitting to emulate narrow-band nonlinear behavior based on GCOMP information. The amount of compression information available for polynomial fitting depends on the GCOMP convention as follows:

- GCOMP1 and GCOMP2 each enable the modeling of 3rd order nonlinear behavior.
- GCOMP3 is modelled using a 5th order polynomial.
- GCOMP4 and GCOMP5 require a 7th order polynomial.
- GCOMP6 includes all four parameters and requires 9th order polynomial fitting.
- GCOMP7 fits to 3rd through 7th (odd) orders if 3-7 data points are specified. If more than seven data points are specified, it performs a 9th order polynomial fitting of the nonlinearity.

The following seven sections show the format for each GCOMP type. Note that each type of block can have an multiple sections each delineated by an optional compression frequency line as shown in the following generic example. Required keywords appear in UPPERCASE ITALIC characters:

```
BEGIN GCOMPx
% F
comp_freq_A
% ( option line for block-type x )
```

```
.... ( data for block-type x at fundamental output frequency A )
% F
comp_freq_B
% ( option line for block-type x )
.... ( data for block-type x at fundamental output frequency B )
.....
% F
comp_freq_F
% ( option line for block-type x )
.... ( data for block-type x at fundamental output frequency F )
END GCOMPx
```

The GCOMP1 Block

The GCOMP1 data block for the .s2d data file allows you to specify the 2-port 3rd order output intercept (IP3). No option line is used.

- General format:
 BEGIN GCOMP1
 % IP3 ! this is the format line, required
 data goes here
 END
- Format line (required):
 % IP3
- Example:

```
BEGIN GCOMP1
% IP3
25 ! this sets the output IP3 to 25 dBm
END
```

The GCOMP2 Block

The GCOMP2 data block for the .s2d data file allows you to specify the 2-port output power at 1 dB gain compression. No option line is used.

- General format:
 BEGIN GCOMP2
 % 1DBC ! this is the format line, required .
 data goes here
 END
- Format line (required):

% 1DBC

- Example:

```
BEGIN GCOMP2
% 1DBC
15 ! this sets the output power for 1 dB gain
! compression at 15 dBm
END
```

The GCOMP3 Block

The GCOMP3 data block for the .s2d data file allows you to specify the 2-port output IP3 and 1DBC simultaneously. No option line is used.

- General format:

```
BEGIN GCOMP3
% 1DBC IP3 ! this is the format line
..... data goes here
END
```

- Format line:

```
% 1DBC IP3
```

This line gives the order for the data in lines to follow. All keywords shown must be given in the format line; keyword order is arbitrary.

- Example:

```
BEGIN GCOMP3 ! includes IP3 and 1DBC
% IP3 1DBC
25 15
END
```

The GCOMP4 Block

The GCOMP4 data block for the .s2d data file allows you to specify the 2-port output IP3, output power at saturation (PS), and the gain compression at saturation (GCS). No option line is used.

- General format:

```
BEGIN GCOMP4
% IP3 PS GCS ! this is the format line
..... data goes here
END
```

- Format line syntax:

```
% IP3 PS GCS
```

This line gives the order for the data in the lines to follow. All keywords shown must be given in the format line;

keyword order is arbitrary.

- Example:

```
BEGIN GCOMP4 ! output 3rd order intercept occurs at 25 dBm
% IP3 PS GCS ! output saturation occurs at 20 dBm
25 20 5 ! with 5 dB of gain compression
END
```

The GCOMP5 Block

The GCOMP5 data block for the .s2d data file allows you to specify the 2-port output 1DBC, output power at saturation (PS), and the gain compression at saturation (GCS). No option line is used.

- General format:

```
BEGIN GCOMP5
% 1DBC PS GCS ! this is the format line
..... data goes here
END
```

- Format line syntax:

```
% 1DBC PS GCS
```

This line gives the order for the data in the lines to follow. All of the keywords shown must be given in the format line; keyword order is arbitrary.

- Example:

```
BEGIN GCOMP5 ! output 1 dB gain compression occurs at 15 dBm
% 1DBC PS GCS ! output saturation occurs at 20
15 20 5 ! dBm with 5 dB of gain compression
END
```

The GCOMP6 Block

The GCOMP6 data block for the .s2d data file allows you to specify the 2-port output IP3, 1DBC, output power at saturation (PS), and the gain compression at saturation (GCS). No option line is used.

- General format:

```
BEGIN GCOMP6
% IP3 1DBC PS GCS ! this is the format line
..... data goes here
END
```

- Format line:

```
% IP3 1DBC PS GCS
```

This line gives the order for the data in the lines to follow. All of the keywords shown must be given in the format line; keyword order is arbitrary.

- Example:

```

BEGIN GCOMP6 ! output 3rd order intercept
! occurs at 25 dBm
% IP3 1DBC PS GCS ! output 1 dB gain compression
! occurs at 15 dBm
25 15 20 5 ! output saturation occurs
! at 20 dBm with 5 dB of gain compression.
END
    
```

The GCOMP7 Block

The GCOMP7 data block for the .s2d data file enables you to specify the input-to-output gain compression characteristic by listing the differential dB gain and differential phase as a function of input power in tabular form.

The S2D file format allows gain compression data at multiple frequencies to be specified in the GCOMP7 block. Note, however, that the AmplifierS2D component – with which many S2D files are later associated – cannot interpolate between gain compression data at different frequencies but uses a fixed frequency specified by the parameter GCfreq . If the compression behavior is to hold true at multiple frequencies, separate GCOMP7 blocks must be defined for each indexing frequency within the same .s2d data file.

It is important to note that the values of S21x and S21y contained in the GCOMP7 section are not the absolute S-parameter responses of the system at the relevant input power. They are differences with respect to the small signal values recorded in the ACDATA section at the same frequency. For instance, if the small signal S21 response at frequency F is polar(ss21mag, ss21deg), and the actual large signal S21 at power PinX at the same frequency F is polar(ls21mag, ls21deg), then the GCOMP7 entry for PinX will register the value of polar(ds21mag, ds21deg) where:

$$ds21mag = 10^{*(20*\log_{10}(ls21mag/ss21mag)/20)} = ls21mag/ss21mag$$

$$ds21deg = ls21deg - ss21deg$$

Please note that regardless of the final format in which the small signal S-parameters [ssijmag, ssijdeg] or the [ds21mag, ds21deg] pairs are expressed in the data file, the dB domain differential definition of the GCOMP7 sections S21 values always holds as mentioned above. Caution must be employed when manually generating or interpreting the contents of the GCOMP7 section and all variables converted to dB domain before numerically adding / subtracting to compute the actual S21values at PinX power.

- General format:
 BEGIN GCOMP7
 # AC (.....) ! this is the option line
 %! this is format line 1
 frequency goes here

%! this is format line 2
 data goes here
 END

- Option line syntax:
 # AC(freq-dim parm-type power-nit parm_format R xx)
 where:

freq-dim	=	Sets the frequency units. Options are HZ, KHZ, MHZ, or GHZ.
parm-type	=	S only.
power_dim	=	DBM only.
parm_format	=	Sets the format for S21 using MA, DB, RI, where: MA declares magnitude and angle (degrees) DB declares $20 \cdot \log_{10}(\text{MA})$ and angle (degrees) RI declares real and imaginary
R xx	=	Declares resistance, where xx = reference resistance for S-parameters.

- Format line:
 There are two format lines:
 % F format line 1
 % PIN n21x, n21y format line 2
 where:

F	=	Indicates that the following data point is the frequency (only one frequency can be specified).
PIN	=	The input power.
n21x, n21y	=	The S21 dB-differential data pair which may be expressed in DB, MA, or RI formats (default is RI).

These lines give the order for the data in the lines to follow. All of the keywords shown must be given in the format line. The order of these keywords is arbitrary.

- Example:

```

BEGIN   GCOMP7
# AC( GHZ S DBM DB R 50.0 )
! Optional Selection: HZ, KHZ, MKZ, or GHZ; Default is GHZ
! Optional Selection: S only; Default is S
! Optional Selection: DBM only; Default is DBM
! Optional Selection: MA, DB, or RI; Default is RI
! Optional Selection: R xx; where xx = reference resistance;
! Default R 50.0
! The S2D file format allows gain compression data at multiple
    
```

```

! frequencies to be specified in the GCOMP7 block. Note, however, that
! the AmplifierS2D component - with which many S2D files are later
! associated - cannot interpolate between gain compression data at
! different frequencies but uses a fixed frequency specified by the
! parameter GCfreq.
%
F
5.
PIN      N21x      N21y
0.0      0.000     0.000
2.0      -0.012     0.173
4.0      -0.027     0.399
6.0      -0.046     0.697
8.0      -0.074     1.162
10.0     -0.116     1.988
12.0     -0.186     2.996
14.0     -0.397     3.754
16.0     -0.904     3.729
18.0     -1.718     3.585
20.0     -2.856     4.337
END

```

Complete .s2d File Example

```

! -----
! amps2d.s2d
! -----
! This is a sample S2D data file containing an activated ACDATA block, an
! activated NDATA block and examples of all seven types of GCOMPx blocks, of
! which, only the GCOMP1 block is activated. A functional S2D file should
! contain only one type of GCOMPx block. An S2D file may also contain an
! IMTDATA block. For details see documentation on IMT data files.
! -----
!
! BEGIN ACDATA
! This line and all lines starting with a comment (!) character are ignored
! Do not have a blank line, or comments as the first line in this file
! ACDATA is an optional block of data for an S2D file. However, some
! components such as AmplifierS2D require the existence of this block.
! The following is the OPTION line
# AC( GHZ  S  RI  R 50.0  FC  1.  0. )
! Optional Selection: HZ, KHZ, MKZ, or GHZ; Default is GHZ
! Optional Selection: S, Z, Y, H, or G; Default is S
! Optional Selection: MA, DB, or RI; Default is RI
! Optional Selection: R xx; where xx = reference resistance; Default R 50.0
! Optional Selection: FC x1 x2 is for frequency conversion: Fout=x1*Fin + x2
! Default is Fout = Fin
% F  n11x  n11y  n21x  n21y  n12x  n12y  n22x  n22y
! The above line is the format line showing the order in the following data
! lines
! The columns of data can be in any order
1.0000  0.3926  -0.1211  -0.0003  -0.0021  -0.0003  -0.0021  0.3926  -0.1211
2.0000  0.3517  -0.3054  -0.0096  -0.0298  -0.0096  -0.0298  0.3517  -0.3054
3.0000  0.0430  -0.5916  -2.6933  -0.1433  -0.5933  -0.1433  0.0430  -0.5916
4.0000  0.4071  -0.2756  2.4617  0.6234  0.3617  0.4234  0.4071  -0.2756
5.0000  0.2041  0.2880  2.6848  -0.5367  0.3848  -0.4367  0.2041  0.2880
6.0000  0.5666  0.0343  2.0383  -0.7437  0.0383  -0.7437  0.5666  0.0343
7.0000  0.0430  0.6916  -2.6933  0.1433  -0.6933  0.1433  0.0430  0.6916
8.0000  0.3059  0.5659  -0.1000  0.1424  -0.1000  0.1424  0.3059  0.5659
9.0000  0.3071  0.4145  -0.0307  0.0673  -0.0307  0.0673  0.3071  0.4145
10.0000 0.3419  0.3336  -0.0134  0.0379  -0.0134  0.0379  0.3419  0.3336
END

```

Advanced Design System 2008

```

BEGIN  NDATA
! This is an optional block of data
# AC( GHZ  RI  S  R  50.0 )
! Optional Selection: HZ, KHZ, MKZ, or GHZ;  Default is GHZ
! Optional Selection: S, Z, Y, H, or G;      Default is S
! Optional Selection: MA, DB, or RI;         Default is RI
! Optional Selection: R xx; where xx = reference resistance; Default R 50.0
% F  nfmin  nllx  nllx  rn
! The above line is the format line showing the order in the following data
! lines
! The columns of data can be in any order
1.0000  2.0000  0.3926  -0.1211  .4
2.0000  2.5000  0.3517  -0.3054  .45
3.0000  3.0000  0.0430  -0.5916  .5
4.0000  3.5000  0.4071  -0.2756  .55
5.0000  4.0000  0.2041  0.2880  .6
6.0000  4.5000  0.5666  0.0343  .65
7.0000  5.0000  0.0430  0.6916  .7
8.0000  5.5000  0.3059  0.5659  .75
9.0000  6.0000  0.3071  0.4145  .8
10.0000 6.5000  0.3419  0.3336  .85
END
! In place of a GCOMP1 block of data there can be either of the following
! blocks: GCOMP2, GCOMP3, GCOMP4, GCOMP5, GCOMP6, GCOMP7
! An example of each of these are given below and only GCOMP1 is
! activated for this sample file.
BEGIN  GCOMP1
% IP3
25
END
! BEGIN  GCOMP2
! % 1DBC
! 15
! END
! BEGIN  GCOMP3
! % IP3 1DBC
! 25 15
! END
! BEGIN  GCOMP4
! % IP3 PS GCS
! 25 25 5
! END
! BEGIN  GCOMP5
! % 1DBC PS GCS
! 15 25 5
! END
! BEGIN  GCOMP6
! % IP3 1DBC PS GCS
! 30 20 25 8
! END
! BEGIN  GCOMP7
! # AC( GHZ  S  DBM  DB  R  50.0 )
! ! Optional Selection: HZ, KHZ, MKZ, or GHZ;  Default is GHZ
! ! Optional Selection: S only;                Default is S
! ! Optional Selection: DBM only;              Default is DBM
! ! Optional Selection: MA, DB, or RI;         Default is RI
! ! Optional Selection: R xx; where xx = reference resistance;
! !                                           Default R 50.0
! ! The S2D file format allows gain compression data at multiple frequencies
! ! to be specified in the GCOMP7 block. Note, however, that the
! ! Amplifier2 and AmplifierS2D components cannot interpolate between
! ! gain compression data at different
! ! frequencies but uses a fixed frequency specified by the parameter
! ! GCfreq.
! % F
! 5.
! % PIN          N21x          N21y
! 0.0           0.000         0.000
! 2.0           -0.012        0.173
! 4.0           -0.027        0.399

```

```

!   6.0      -0.046    0.697
!   8.0      -0.074    1.162
!  10.0     -0.116    1.988
!  12.0     -0.186    2.996
!  14.0     -0.397    3.754
!  16.0     -0.904    3.729
!  18.0     -1.718    3.585
!  20.0     -2.856    4.337
! END
!
! -----

```

IMT Format

Intermodulation table (IMT) data is used to represent the behavior of mixers and frequency translators. Three distinct types of IMT formats are supported in ADS. See the following sections for descriptions and examples:

- [O-Type IMT Format](#)
- [A-Type IMT Format](#)
- [B-Type IMT Format](#)

O-Type IMT Format

Standard spur tables for mixers are defined in this format which allows only single side banded (SSB), single-RF, single-LO mixing specification of IF voltage strengths in dB (relative to IF fundamental) or dBm (absolute value). Such files contain only a single IMT matrix preceded by an option line containing reference values of RF and LO signal strengths. O-type IMT files do not contain any information regarding actual RF or LO frequencies, and therefore, may be used to characterize a generic mixing process. Each column of the table represents mixing due to one of the N harmonics of the LO tone and each row represents mixing one of the M harmonics of the RF tone.

Example O-Type IM Table

In the following O-type IM table, five LO harmonics are mixed with three RF harmonics. All values are non-negative and the IF fundamental strength is 0 dB. This indicates that all the spurs are suppressed with respect to the IF fundamental by the specified value. Thus, the IM products for $3*RFfreq + 2*LOfreq$ and $3*RFfreq - 2*LOfreq$ are both suppressed by 69 dB below the strength of the IF fundamental when reference RF and LO powers at mixer inputs are -10 dBm and +7 dBm respectively. There is no specific distinction between sum and difference products, and the table contains no information about IF phase.

```

! O-type IMT file
BEGIN IMTDATA
! Option line for reference power at: RF = -10 dBm, LO = +7 dBm

```

```

# IMT ( -10 7 )
! Format line for LO-side harmonics
% 0 1 2 3 4 5
  99 26 35 39 50 41
  24 0 35 13 40 24
  73 73 74 70 71 64
  67 64 69 50 77 47
END IMTDATA

```

A-Type IMT Format

IM tables extracted from mixers in simulation environment are defined in this format which allows double side banded (DSB), single-RF, single-LO mixing specification of IF voltage strengths in dBm (absolute value). Such files may contain multiple IMT matrices each captured at specific levels of RF and LO powers and input frequencies. A-type IMT files therefore contain frequency specific information, and may be used for interpolation across modest spans of RF or LO frequency and power variation. Each column of the table represents mixing due to one of the N harmonics of the LO tone, and each row represents mixing one of the M harmonics of the RF tone. Both positive (sum) and negative (difference) rows are represented in the table as shown by the value of the first column.

Example A-Type IM Table

In the following A-type IM table two LO harmonics are mixed with two RF harmonics. The IM product for $2 \cdot \text{RFfreq} + 2 \cdot \text{LOfreq}$ is -40 dBm at 23 degrees phase; whereas, the product for $2 \cdot \text{RFfreq} - 2 \cdot \text{LOfreq}$ is -50 dBm at -41 degrees phase when reference RF and LO powers at mixer inputs are -10 dBm and +7 dBm respectively, and reference frequencies are $\text{RFfreq}=2$ GHz and $\text{LOfreq}=1.7$ GHz. Although, for the difference tone the table shows $-2 \cdot \text{FRF} + 2 \cdot \text{FLO}$, given that $\text{FLO} < \text{FRF}$ in this case, we get the phase as -41 degrees by taking the complex conjugate of the value on file which is +41 degrees. To represent a physically feasible system, the values in the first complex column which contain harmonics of the RF tone, should be complex conjugates when mirrored across the DC spur. Thus the spur at $N=0, M=m$ should be the complex conjugate of the spur at $N=0, M=-m$.

```

! A-type IMT file
BEGIN IMTDATA
! Option line for reference power at: RF = -10 dBm, LO = +7 dBm
# IMT ( GHz S DBM R 50.0 )
! Format line for RF frequency
% FRF
  2.0
! Format line for LO frequency
% FLO
  1.7
! Format line for reference RF power
% PRF
  -10
! Format line for reference LO power
% PLO
  -7
! Format line for LO-side harmonics
% M 0 1 2
  -2 -24 -77 -35 39 -50 41
  -1 -67 -64 -35 13 -40 24

```

```

0   -99   73   -74   70   -71   64
1   -67   64   -69   50   -77   47
2   -24   77   -35  -33   -40  -23
END IMTDATA

```

B-Type IMT Format

IM tables extracted from mixers in the simulation environment are defined in this format which allows double side banded (DSB), multi-RF, single-LO mixing specification of IF voltage strengths in dBm (absolute value). Such files may contain multiple IMT matrices each captured at specific levels of LO powers and frequencies. RF frequencies and powers should remain constant throughout the file. B-type IMT files therefore contain frequency specific information, and may be used for interpolation across modest spans of LO frequency and power variation at nominal RF powers and frequencies. Each column of the table represents mixing due to one of the N harmonics of the LO tone and each row represents mixing the various harmonics of the RF tones. Both positive (sum) and negative (difference) rows are represented in the table as shown by the value of the first r columns where r RF tones are active at the mixer's signal input.

To represent a physically feasible system, the values in the first complex column which contain pure RF on RF mixing, should be complex conjugates when mirrored across the DC spur. Thus the spur at N=0, M1=m1 M2=m2 should be the complex conjugate of the spur at N=0, M1=-m1, M2=-m2.

Example B-Type IM Table

In the following B-type IM table two LO harmonics are mixed with two RF harmonics of the first RF tone and one harmonic of the second RF tone. The IM product for $2 \cdot \text{RFfreq1} - \text{RFfreq2} + 2 \cdot \text{LOfreq}$ is -71 dBm at 64 degrees phase; whereas, the product at $-\text{RFfreq1} + \text{RFfreq2}$ is -45 dBm +66 degrees.

```

! B-type IMT file
BEGIN IMTDATA
! Option line
# IMT ( GHz S DBM R 50.0 )
! Format line for RF frequency
% FRF1   FRF2
   2.0   2.1
! Format line for LO frequency
% FLO
   1.7
! Format line for reference RF power
% PRF1   PRF2
  -10   -15
! Format line for reference LO power
% PLO
   -7
! Format line for LO-side harmonics
% M1     M2     0           1           2
  -2     -1    -24    -77    -35     39    -50    41
  -2     0    -67    -64    -35     13    -40    24
  -2     1    -77     23    -74     70    -71    64
  -1    -1   -45    -32    -69     50    -77    47

```

```

-1      0      -43      -97      -35      -33      -40      -23
-1      1      -24      -77      -35      39      -50      41
0       -1      -67      64      -35      13      -40      24
0        0      -99      73      -74      70      -71      64
0        1      -67      -64      -69      50      -77      47
1       -1      -24      77      -35      -33      -40      -23
1        0      -43      97      -37      -29      -55      -23
1        1      -45      32      -71      82      -50      41
2       -1      -77      -23      -74      70      -71      64
2        0      -67      64      -69      50      -77      47
2        1      -24      77      -35      -33      -40      -23
END IMTDATA

```

SPW Format

These files contain time-domain waveform data and are signal data files in SPW format. Advanced Design System can interface with the Cadence Alta Group SPW format through the use of data files in SPW format. Both ASCII (.ascsig) and binary (.sig) file formats are supported.

The SPW version 3.0 data file format is fully supported for real double data and partially supported for complex double data.

Guidelines for .ascsig

- The SPW version 3.0 data file format must be used.
- Comments can only be included on the one line following the \$USER_COMMENT statement.
- A blank line must be included above the statements \$COMMON_INFO and Sampling Frequency.

Example .ascsig Files

There are two examples, one uses real values and the second uses complex numbers.

File 1: Real double-data format

```

$SIGNAL_FILE 9
$USER_COMMENT
$COMMON_INFO
SPW Version = 3.0
Sampling Frequency = 1
Starting Time = 0
$DATA_INFO
Number of points = 6
Signal Type = Double
$DATA

```

```

1.00000000000000000000000000000000
1.00000000000000000000000000000000
- 1.00000000000000000000000000000000
- 1.00000000000000000000000000000000
1.00000000000000000000000000000000
1.00000000000000000000000000000000

```

File 2: Complex double-data format

```

$SIGNAL_FILE 9
$USER_COMMENT
$COMMON_INFO
SPW Version = 3.0
Sampling Frequency = 1
Starting Time = 0
$DATA_INFO
Number of points = 10
Signal Type = Double
Complex Format = Real_Imag
$DATA
1.00000000000000000000000000000000+j1.00000000000000000000000000000000
1.00000000000000000000000000000000+j1.00000000000000000000000000000000
- 1.00000000000000000000000000000000+j1.00000000000000000000000000000000
- 1.00000000000000000000000000000000+j1.00000000000000000000000000000000
1.00000000000000000000000000000000+j1.00000000000000000000000000000000
1.00000000000000000000000000000000+j1.00000000000000000000000000000000
- 1.00000000000000000000000000000000+j1.00000000000000000000000000000000
- 1.00000000000000000000000000000000+j1.00000000000000000000000000000000
- 1.00000000000000000000000000000000+j1.00000000000000000000000000000000
-1.00000000000000000000000000000000+j1.00000000000000000000000000000000

```

TIM Format

The .tim file is a signal data file in MDIF format. It contains time-domain waveform data for defining the signals associated with certain sources.

The general .tim file format is:

```

BEGIN TIMEDATA
# T ( SEC V R xx )
% time voltage
<data line>
...
<data line>
END


```

The BINTIM Format

The BINTIM format (.bintim) is for binary time-domain waveform data files. In .bintim files, the format is the same as .tim files, except the BEGIN line is preceded by a line indicating the number of data points, n:

NUMBER OF DATA n

The <data line> in a .bintim file is just a binary dump of all the waveform (time, voltage) data. Also, there is no END line.

 **Note**
The .bintim format is not supported in the Data File Tool. However, certain signal processing components can read .bintim files.

Guidelines for .tim Files

- An exclamation point (!) at the beginning of a line makes it a comment line. Characters following the ! are ignored by the program.
- The TIMEDATA data block is required.
- When the file reader reads a file, it renames the independent and dependent variable names regardless of the names specified in the file. The file reader reads the independent variable name as time , and the dependent variable name as voltage .

TIMEDATA Block

- The BEGIN statement:
BEGIN TIMEDATA ! Begin time-domain waveform data
- Option line:
T (time_unit data_unit R xx)
where:

#	=	Delimiter tells the program you are specifying these parameters.
T	=	Time
time_unit	=	Sets time units. Options are SEC, MSEC, USEC, NSEC, PSEC.
data_unit	=	Set the units for the voltage values. Options are: V = volts MV = millivolts
R xx	=	Sets resistance, where xx =

		reference resistance. (default is 50.0)
--	--	---

- Format line
% time voltage
where:

%	=	Delimiter tells the program you are specifying these parameters
time	=	time
voltage	=	voltage

By design of the program, the syntax time and voltage in the Format line are arbitrary. These values can be whatever you prefer. For example, an option line such as:
% t mV
can be used. However, these values are converted to time and voltage by the file reader when the .tim file is imported, and these will be the variables appearing in a dataset (.ds) file.

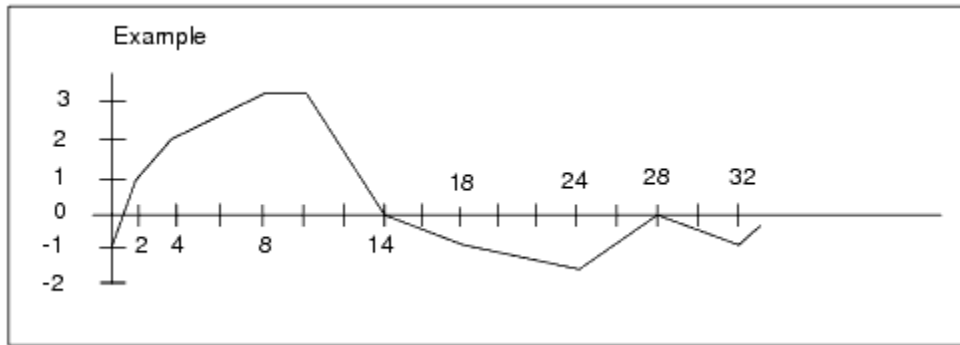
- The TIMEDATA data requirements are as follows:
 - Though a value for time=0 is not required, this can lead to erroneous results when using the DataAccessComponent with its Extrapolation Mode set to Linear. To avoid errors, set a value for time=0, set the DAC's Extrapolation Mode to Constant, or do both.
 - The signal is assumed to be time periodic with the time period equal to maximum time minus minimum time.

Example .tim Files

```

BEGIN TIMEDATA
# T ( USEC V R 50 )
% time voltage
  0.0 -1.0
  2.0 1.0
  4.0 2.0
  8.0 3.0
 10.0 3.0
 14.0 0.0
 18.0 -1.0
 24.0 -2.0
 28.0 0.0
 32.0 -1.0
END
    
```

This example file results in a time periodic voltage versus time with time period 32 μsec, interpreted as a piece-wise linear voltage description.



Time periodic voltage vs. Time with time period 32 μ sec.

The following example shows how to handle the independent and dependent variable names when using a `DataAccessComponent`. This is useful since the file reader reads the independent variable name as time, and the dependent variable name as voltage, regardless of the names specified in the file. The following example data files shows the variable names specified as t and v :

```
BEGIN TIMEDATA
%      t          v
      0          0
1e-011  0.00995017
2e-011  0.0198013
5e-011  0.0487706
1.4e-010 0.130642
4.1e-010 0.33635
1e-009   0.632121
END
```

Though the variable names are t and v , the file reader changes the names to time and voltage, requiring the following syntax for the `DataAccessComponent`:

```
DataAccessComponent
Type=Time Domain Waveform (TIM MDIF)
iVar1="time"
iVal1=time
VAR
X=file{DAC1,"voltage"}
```

Generic MDIF

The generic MDIF provides a generalized MDIF format for unifying the various specific MDIF formats, and overcoming some limitations of other formats. The generic format enables diverse applications to use a common data I/O interface, so long as the intent is to access/save multidimensional (multiple independent vs dependent variables) data.

The general format is as follows:

```

VAR var1Name(var1Type) = var1
ValueVAR var2Name(var2Type) = var2Value
..
VAR varNName(varNType) = varNValue
BEGIN blockName
% bVar1Name(bVar1Type) bVar2Name(bVar2Type) ....
% bVarLName(bVarLType) ...
% ...
% bVarQName(bVarQType) ... bVarPName(bVarPType)
bVar1Value bVar2Value ...
bVarLValue ..
..
bVarQValue ... bVarPValue
bVar1Value bVar2Value ...
bVarLValue ..
..
bVarQValue ... bVarPValue
...
END

```

where var*Type can be the token:

- 0 or int
- 1 or real
- 2 or string

Type bVar*Type can be one of the above as well as:

- 3 or complex
- 4 or boolean
- 5 or binary
- 6 or octal
- 7 or hexadecimal
- 8 or byte16

The variable names above constitute a name-space uniquely identified by the string blockName which is either:

- alphanumeric: all bVar*Name block variables are dependent, except bVar1Name, which is usually the most rapidly changing (innermost) independent variable.
- or
- DSCR(blockName): all bVar*Name block variables are dependent, and there is an indexing implicit independent variable.

A string type variable's value must be surrounded by "".

If there are multiple blocks, the outermost independent variables (e.g., VAR var1Name(var1Type) = var1) apply only to the block immediately following the variable definitions, and not to any other blocks.

Advanced Design System 2008

The block data (bVar*Value) lines must follow the pattern (order, number of values per line, and number of lines) of the format (%) lines. If the number of values in any data line does not match the number of dependent variables specified in the corresponding format (%) line, incorrect results will occur. A variable's value cannot be split across lines. Although there is no line length limit specified, MDIF file readers may choose to truncate at some finite length. This may result in a file read error, or, if the file was carefully crafted, truncated names and/or string-type values.

Scale factors, which can be applied only to real numbers, may be case-insensitive suffixes as follows:

f = 1e-15, p = 1e-12, n = 1e-9, u = 1e-6, mil = 2.54e-5, m = 1e-3,

k = 1e3, g = 1e9, t = 1e12

E.g.: 15mA = 15e-3, 30KHz = 30e3

There should be no space between the number and the suffix, and extra characters are ignored. Unrecognized suffixes result in 1.0. The above is not totally consistent with the rest of ADS.

```
!=====
! Example 1
REM This has 3 indepVars: v1, v2, v3(innermost) and
REM 4 depVars: dv1(integer), dv2(real), dv3(string) and
REM dv4(hexadecimal), but is read in as a string.
REM The outermost indepVars: v1, v2 apply only to the block
REM immediately following them, and not to any other block.
! There are 2 data nodes
VAR v1(0) = 1
VAR v2(1) = 2.2
BEGIN blk1
% v3(1) dv1(1) dv2(1) dv3(2) dv4(hexadecimal)
7.7 8 9.9999 "line 1" 0xabc
8.8 9 1.11 "line 2 " 0x123
END
VAR v1(0) = 2
VAR v2(1) = 3.2
BEGIN blk1
% v3(1) dv1(1) dv2(1) dv3(2) dv4(hexadecimal)
8.7 9uF 10.9999mA "line 1" 0xff
9.8 10uF 11.11mA "line 2 " 0xdef
END
!=====
! Example 2
! Created Tue Mar 9 13:39:19 1999
! Data Acquired Tue Mar 9 13:38:34 1999
BEGIN NDATA_noise
% freq(real)  Sopt(complex)  NFmin(real)  Rn(real)  PortZ[1](real)
1e+09         0.098481        0.017365        1         5         50
2e+09         0.18794         0.068404        2         10        50
3e+09         0.25981          0.15            3         15        50
4e+09         0.30642         0.25712         4         20        50
5e+09         0.32139         0.38302         5         25        50
6e+09         0.3             0.51962         6         30        50
7e+09         0.23941         0.65778         7         35        50
```

```
      8e+09      0.13892      0.78785      8      40      50
9.543e+09     -0.014122      0.911     9.5445     46.166     50
END
```

CITIfile Data Format

This section describes the CITIfile format definitions of key terms, and file examples. It also includes:

- Keyword reference
- File guidelines
- Instructions for converting between disk formats
- Device-specific definitions
- File name requirements

Overview

CITIfile is a standardized data format that is used for exchanging data between different computers and instruments. CITIfile stands for Common Instrumentation Transfer and Interchange file format.

This standard is a group effort between instrument and computer-aided design program designers. As much as possible, CITIfile meets current needs for data transfer, and it is designed to be expandable so it can meet future needs.

CITIfile defines how the data inside an ASCII package is formatted. Since it is not tied to any particular disk or transfer format, it can be used with any operating system, such as DOS or UNIX, with any disk format, such as DOS or HFS, or with any transfer mechanism, such as by disk, LAN, or GPIB.

By careful implementation of the standard, instruments and software packages using CITIfile are able to load and work with data created on another instrument or computer. It is possible, for example, for a network analyzer to directly load and display data measured on a scalar analyzer, or for a software package running on a computer to read data measured on the network analyzer.

Data Formats

There are two main types of data formats: binary and ASCII. CITIfile uses the ASCII text format. Although this format requires more space than binary format, ASCII data is a transportable, standard type of format which is supported by all operating systems. In addition, the ASCII format is accepted by most text editors. This allows files to be created, examined, and edited easily, making CITIfile easier to test and debug.

File and Operating System Formats

CITIfile is a data storage convention designed to be independent of the operating system, and therefore may be implemented by any file system. However, transfer between file systems may sometimes be necessary. You can use any software that has the ability to transfer ASCII files between systems to transfer CITIfile data. Refer to [Converting Between Disk Formats](#) for more information.

The descriptions and examples shown here demonstrate how CITIfile may be used to store and transfer both measurement information and data. The use of a single, common format allows data to be easily moved between instruments and computers.

CITIfile Definitions

This section defines: package , header , data array , and keyword .

Package

A typical CITIfile package is divided into two parts:

- The header is made up of keywords and setup information.
- The data usually consists of one or more arrays of data.

The following example shows the basic structure of a CITIfile package:

Header	CITIFILE A.01.00 NAME MEMORY VAR FREQ MAG 3 DATA S RI
Data	BEGIN -3.54545E-2, -1.38601E-3 0.23491E-3, -1.39883E-3 2.00382E-3, -1.40022E-3 END

When stored in a file there may be more than one CITIfile package. With the Agilent 8510 network analyzer, for example, storing a memory all will save all eight of the memories held in the instrument. This results in a single file that contains eight CITIfile packages .

Header

Advanced Design System 2008

The header section contains information about the data that will follow. It may also include information about the setup of the instrument that measured the data. The CITIfile header shown in the first example has the minimum of information necessary; no instrument setup information was included.

Data Array

An array is numeric data that is arranged with one data element per line. A CITIfile package may contain more than one array of data. Arrays of data start after the `BEGIN` keyword, and the `END` keyword follows the last data element in an array.

A CITIfile package does not necessarily need to include data arrays. For instance, CITIfile could be used to store the current state of an instrument. In that case the keywords `VAR`, `BEGIN`, and `END` would not be required.

When accessing arrays via the DAC (DataAccessComponent), the simulator requires array elements to be listed completely and in order.

Example: `S[1,1]`, `S[1,2]`, `S[2,1]`, `S[2,2]`

Keywords

Keywords are always the first word on a new line. They are always one continuous word without embedded spaces. A listing of all the keywords used in version A.01.00 of CITIfile is shown in [CITIfile Keyword Reference](#).

CITIfile Examples

The following are examples of CITIfile packages.

Display Memory File

This example shows an Agilent 8510 display memory file. The file contains no frequency information. Some instruments do not keep frequency information for display memory data, so this information is not included in the CITIfile package.

Note that instrument-specific information (`#NA` = network analyzer information) is also stored in this file.

```

CITIFILE A.01.00
#NA VERSION HP8510B.05.00
NAME MEMORY
#NA REGISTER 1
VAR FREQ MAG 5
DATA S RI
BEGIN
-1.31189E-3,-1.47980E-3
-3.67867E-3,-0.67782E-3
-3.43990E-3,0.58746E-3
-2.70664E-4,-9.76175E-4
0.65892E-4,-9.61571E-4
END
    
```

Agilent 8510 Data File

This example shows an 8510 data file, a package created from the data register of an Agilent 8510 network analyzer. In this case, 10 points of real and imaginary data was stored, and frequency information was recorded in a segment list table.

```

CITIFILE A.01.00
#NA VERSION 8510B.05.00
NAME DATA
#NA REGISTER 1
VAR FREQ MAG 10
DATA S[1,1] RI
SEG_LIST_BEGIN
SEG 1000000000 4000000000 10
SEG_LIST_END
BEGIN
0.86303E-1,-8.98651E-1
8.97491E-1,3.06915E-1
-4.96887E-1,7.87323E-1
-5.65338E-1,-7.05291E-1
8.94287E-1,-4.25537E-1
1.77551E-1,8.96606E-1
-9.35028E-1,-1.10504E-1
3.69079E-1,-9.13787E-1
7.80120E-1,5.37841E-1
-7.78350E-1,5.72082E-1
END
    
```

Agilent 8510 3-Term Frequency List Cal Set File

This example shows an 8510 3-term frequency list cal set file. It shows how CITIfile may be used to store instrument setup information. In the case of an 8510 cal set, a limited instrument state is needed to return the instrument to the same state that it was in when the calibration was done.

Three arrays of error correction data are defined by using three `DATA` statements. Some instruments require these arrays be in the proper order, from `E[1]` to `E[3]`. In general, CITIfile implementations should strive to handle data arrays that are arranged in any order.

```

CITIFILE A.01.00
#NA VERSION 8510B.05.00
NAME CAL_SET
#NA REGISTER 1
VAR FREQ MAG 4
DATA E[1] RI
DATA E[2] RI
DATA E[3] RI
#NA SWEEP_TIME 9.999987E-2
#NA POWER1 1.0E1
#NA POWER2 1.0E1
#NA PARAMS 2
#NA CAL_TYPE 3
#NA POWER_SLOPE 0.0E0
#NA SLOPE_MODE 0
#NA TRIM_SWEEP 0
#NA SWEEP_MODE 4
#NA LOWPASS_FLAG -1
#NA FREQ_INFO 1
#NA SPAN 1000000000 3000000000 4
#NA DUPLICATES 0
#NA ARB_SEG 1000000000 1000000000 1
#NA ARB_SEG 2000000000 3000000000 3
VAR_LIST_BEGIN
1000000000
2000000000
2500000000
3000000000
VAR_LIST_END
BEGIN
1.12134E-3,1.73103E-3
4.23145E-3,-5.36775E-3
-0.56815E-3,5.32650E-3
-1.85942E-3,-4.07981E-3
END
BEGIN
2.03895E-2,-0.82674E-2
-4.21371E-2,-0.24871E-2
0.21038E-2,-3.06778E-2
1.20315E-2,5.99861E-2
END
BEGIN
4.45404E-1,4.31518E-1
8.34777E-1,-1.33056E-1
-7.09137E-1,5.58410E-1
4.84252E-1,-8.07098E-1
END

```

When an instrument's frequency list mode is used, as it was in this example, a list of frequencies is stored in the file after the VAR_LIST_BEGIN statement. The unsorted frequency list segments used by this instrument to create the VAR_LIST_BEGIN data are defined in the #NA ARB_SEG statements.

2-Port S-Parameter Data File

This example shows how a CITIfile can store 2-port S-parameter data. The independent variable name FREQ has two values located in the VAR_LIST_BEGIN section. The four DATA name definitions indicate there are four data arrays in the CITIfile package located in the BEGIN...END sections. The data must be in the correct order to ensure values are assigned to the intended ports. The order in this example results in data assigned to the ports as shown in the table that follows:

```

CITIFILE A.01.00
NAME BAF1
VAR_FREQ MAG 2
DATA S[1,1] MAGANGLE
DATA S[1,2] MAGANGLE
DATA S[2,1] MAGANGLE
DATA S[2,2] MAGANGLE
VAR_LIST_BEGIN
1E9
2E9
VAR_LIST_END
BEGIN
0.1, 2
0.2, 3
END
BEGIN
0.3, 4
0.4, 5
END
BEGIN
0.5, 6
0.6, 7
END
BEGIN
0.7, 8
0.8, 9
END
    
```

DATA	FREQ = 1E9	FREQ = 2E9
s[1,1]	s[0.1,2]	s[0.2,3]
s[1,2]	s[0.3,4]	s[0.4,5]
s[2,1]	s[0.5,6]	s[0.6,7]
s[2,2]	s[0.7,8]	s[0.8,9]

CITIfile Keyword Reference

The following table lists keywords, definitions, and examples.

CITIfile Keywords and Definitions

Keyword	Example and Explanation
CITIFILE	<p>Example: CITIFILE A.01.00</p> <p>Identifies the file as a CITIfile and indicates the revision level of the file. The CITIFILE keyword and revision code must precede any other keywords.</p> <p>The CITIFILE keyword at the beginning of the package assures the device reading the file that the data that follows is in the CITIfile format. The revision number allows for future extensions of the CITIfile standard. The revision code shown here following the CITIFILE</p>

Advanced Design System 2008

	keyword indicates that the machine writing this file is using the A.01.00 version of CITIfile as defined here. Any future extensions of CITIfile will increment the revision code.
NAME	<p>Example: NAME CAL_SET</p> <p>Sets the current CITIfile package name. The package name should be a single word with no embedded spaces. Some standard package names:</p> <p>RAW_DATA : Uncorrected data.</p> <p>DATA: Data that has been error corrected. When only a single data array exists, it should be named DATA .</p> <p>CAL_SET: Coefficients used for error correction.</p> <p>CAL_KIT: Description of the standards used.</p> <p>DELAY_TABLE: Delay coefficients for calibration.</p>
VAR	<p>Example: VAR FREQ MAG 201</p> <p>Defines the name of the independent variable (FREQ); the format of values in a VAR_LIST_BEGIN table (MAG) if used; and the number of data points (201).</p>
CONSTANT	<p>Example: CONSTANT name value</p> <p>Lets you record values that do not change when the independent variable changes.</p>
#	<p>Example: #NA POWER1 1.0E1</p> <p>Lets you define variables specific to a particular type of device. The pound sign (#) tells the device reading the file that the following variable is for a particular device. The device identifier shown here (NA) indicates that the information is for a network analyzer. This convention lets you define new devices without fear of conflict with keywords for previously defined devices. The device identifier can be any number of characters.</p>
SEG_LIST_BEGIN	<p>Indicates that a list of segments for the independent variable follows.</p> <p>Segment Format: segment type start stop number of points</p> <p>The current implementation supports only a signal segment. If you use more than one segment, use the VAR_LIST_BEGIN construct. CITIfile revision A.01.00 supports only the SEG (linear segment) segment type.</p>
SEG_LIST_END	Sets the end of a list of independent variable segments.
VAR_LIST_BEGIN	Indicates that a list of the values for the independent variable (declared in the VAR statement) follows. Only the MAG format is supported in revision A.01.00.
VAR_LIST_END	Sets the end of a list of values for the independent variable.

DATA	<p>Example: DATA S[1,1] RI</p> <p>Defines the name of an array of data that will be read later in the current CITIfile package , and the format that the data will be in. Multiple arrays of data are supported by using standard array indexing as shown above. CITIfile revision A.01.00 supports only the RI (real and imaginary) format, and a maximum of two array indexes.</p> <p>Commonly used array names include:</p> <ul style="list-style-type: none">S – S parameterE – Error TermVoltage – VoltageVOLTAGE_RATIO – a ratio of two voltages (A/R)
------	---

CITIfile Guidelines

The following general guidelines aid in making CITIfiles universally transportable:

Line Length. The length of a line within a CITIfile package should not exceed 80 characters. This allows instruments which may have limited RAM to define a reasonable input buffer length.

Keywords. Keywords are always at the beginning of a new line. The end of a line is as defined by the file system or transfer mechanism being used.

Unrecognized Keywords. When reading a CITIfile, unrecognized keywords should be ignored. There are two reasons for this:

- Ignoring unknown keywords allows new keywords to be added, without affecting an older program or instrument that might not use the new keywords. The older instrument or program can still use the rest of the data in the CITIfile as it did before. Ignoring unknown keywords allows "backwards compatibility" to be maintained.
- Keywords intended for other instruments or devices can be added to the same file without affecting the reading of the data.

Adding New Devices. Individual users are allowed to create their own device keywords through the # (user-defined device) mechanism. (Refer to the table immediately above for more information.) Individual users should not add keywords to CITIfiles without using the # notation, as this could make their files incompatible with current or future CITIfile implementations.

File Names. Some instruments or programs identify a particular type of file by characters that are added before or after the file name. Creating a file with a particular prefix or ending is not a problem. However in general an instrument or program should not require any such characters when reading a file. This allows any file, no matter what the filename, to be read into the instrument or computer. Requiring special filename prefixes and endings makes the exchange of data between different instruments and computers much more difficult.

Converting Between Disk Formats

Most current Agilent Technologies instruments use disks formatted in the Logical Interchange Format (LIF). Some instruments also use DOS-formatted disks. CITIfiles created on one file system (LIF, DOS, HFS, etc.) may be transferred to other file systems. This is useful for designers using test equipment in addition to ADS to read/write CITIfiles.

HFS

Several LIF and DOS utilities are available for HP-UX workstations. The HP-UX utilities `lifcp` and `doscp` can transfer CITIfiles to and from LIF and DOS disks. Using `lifcp` and `doscp` are similar; using `lifcp` is described below. Several other LIF and DOS utilities are also available. Consult the manuals for these utilities for more detailed information. Listing the contents of a LIF disk when using HP-UX would be similar to the following example:

```
lifls /dev/rdisk/1s0.0
```

The device name used will depend on how your system was configured. Copying a CITIfile named `DD_FILED1` from a LIF disk to HFS would be similar to the following example:

```
lifcp /dev/rdisk/1s0.0: DD_FILED1 DD_FILED1
```

To copy a standard HFS ASCII file to a LIF disk:

```
lifcp DD_FILED1 /dev/rdisk/1s0.0: DD_FILED1
```

When used on an HFS disk, The HP-UX program `RMB/UX` (Rocky Mountain BASIC for HP-UX) has the ability to write a CITIfile in either as a standard HFS ASCII file, or as a LIF volume file. The LIF volume file is the default. This type of file is not directly readable when using the HP-UX operating system, and the copy commands listed above will not work correctly.

BASIC program writers are encouraged to detect when writing to an HFS disk, and to use the standard HFS format. The program examples `CITIWRITE` and `CITIDOALL` show how this can be done. However CITIfiles stored in the LIF volume format can still be transferred to LIF disks, or converted to standard HFS files. To copy a LIF volume file named `DD_FILED` stored on an HFS disk and move it to a LIF disk:

```
lifcp DD_FILED1:WS_FILE /dev/rdisk/1s0.0: DD_FILED1
```

To copy the LIF volume file DD_FILED1 to a standard HFS file named NEWFILE :

```
lifcp DD_FILED1:WS_FILE NEWFILE
```

DOS

Utilities are available for DOS machines that enable them to transfer files to and from a LIF formatted disk. Many of these programs are menu-driven, and are available from the following companies: Agilent, Oswego, Meadow Soft Works, and Innovative Software Systems.

CITIfile Device-Specific Definitions

CITIfile is a generic definition of a data storage format for any type of computer or instrument. However each type of device may need to define certain conventions for itself. This section describes the device-specific keywords and conventions for current implementations.

Network Analyzer (#NA) Definitions

Data Grouping. Data arrays of the same type, obtained during a single measurement operation, are stored in a single CITIfile package. For example, all error correction arrays are stored in the same CITIfile package, and all parameters acquired during an s-parameter measurement operation are stored in the same CITIfile package.

A CITIfile package is as described in the main CITIfile documentation: the CITIFILE keyword, followed by a header section, usually followed by one or more arrays of data.

 **Note**
There are some specific problems with the current version in reading and/or writing this data format. On the Agilent EEsof web site, refer to the Release Notes in Product Documentation, and to Technical Support for more information and workarounds (<http://www.agilent.com/find/eesof>).

Network Analyzer Keywords. The definition of CITIfile allows for statements that are specific to a certain type of device. the following table lists the currently defined commands for the #NA (network analyzer) keyword.

Network Analyzer Keyword Commands

Statement	Explanation
#NA ARB_SEG x y p	A list segment, as entered by the user.

Advanced Design System 2008

	<p>xx = start value y = stop value p = number of points.</p>
#NA REGISTER nn	<p>Register in instrument that the current data package was stored in. nn = number of register.</p>
#NA SWEEP_TIME tt	<p>The sweep time of the analyzer. tt = time in seconds.</p>
#NA POWER1 pp	<p>Power level of signal source #1. pp = power in dBm.</p>
#NA POWER2 pp	<p>Power level of signal source #2. pp = power in dBm.</p>
#NA PARAMS aa	<p>Bitmap of valid parameters for a calibration. Bit positions 1-8 represent the following: Bit #1 = S 11 Bit #2 = S 21 Bit #3 = S 12 Bit #4 = S 22 Bit #5 = user1 Bit #6 = user2 Bit #7 = user3 Bit #8 = user4 A bit equal to one means that the calibration is valid for that parameter; a zero means that the calibration is not valid for that parameter. Bit #0 is the least significant bit.</p>
NA# CAL_TYPE CC	<p>The type of calibration used: 1 = response calibration. 2 = response and isolation calibration. 3 = one-port calibration on port 1. 4 = one-port calibration on port 2. 5 = two-port calibration (includes one-path full & TRL)</p>
NA# POWER_SLOPE SS	<p>Change in power versus frequency. ss = dBm/GHz</p>
NA# SLOPE_MODE mm	<p>On/off flag for power slope. mm = 0 = off mm = 1 = on</p>
NA# TRIM_SWEEP tt	<p>Linearity adjustment value for swept sources.</p>
NA# SWEEP_MODE SS	<p>Type of sweep done to make measurement. 0 = swept 1 = stepped 2 = single-point 3 = fast CW 4 = list</p>

Advanced Design System 2008

NA# LOWPASS_FLAG ff	Low-pass time domain flag. ff = 0 = low-pass time domain enabled . ff = 1 = low-pass time domain disabled .
NA# FREQ_INFO ii	The frequency information flag. ii = 0 = frequency information displayed on instrument screen. ii = 1 = frequency information not displayed on instrument screen.
NA# DUPLICATES dd	Delete duplicates flag. Determines if points listed more than once should be measured more than once. dd = 0 = points listed more than once are measured as many times as they are listed. dd = 1 = points are measured only once.
NA# SPAN xx yy pp	The sweep parameters: xx = start value yy = stop value pp = number of points
NA# IF_BW gg	The IF bandwidth setting of the receiver. gg = IF bandwidth in Hertz.

Error Array Numbering

Current network analyzer implementations use between one and twelve error coefficient arrays to perform error correction. The `CAL_TYPE` keyword description in [Network Analyzer \(#NA\) Definitions](#) lists the currently defined calibration types. The following table defines the meanings of each coefficient array with respect to the error model used.

Network Analyzer Error Coefficient Arrays

Error Array Name	Frequency Response	Response & Isolation	All 1-Port	All 2-Port
E1	Er or Et	Ed or Ex	Ed	Edf
E2	-	Er or Et	Es	Esf
E3	-	-	Er	Erf
E4	-	-	-	Exf
E5	-	-	-	Elf
E6	-	-	-	Etf
E7	-	-	-	Edr
E8	-	-	-	Esr
E9	-	-	-	Err

Advanced Design System 2008

E10	-	-	-	Exr
E11	-	-	-	Elr
E12	-	-	-	Etr

Disk Filename Requirements

Some instruments or programs identify a particular type of file by characters that are added before or after the file name. In general an instrument or program should not require any such characters when reading a file.

There exist CITIfile implementations which do have file naming restrictions. This section explains how to work around these restrictions.

Agilent 8510 Series CITIfile

The 8510 checks the first 3 letters of the filename to determine what is stored in the file. The file prefixes for an 8510 CITIfile are listed in the following table.

Agilent 8510 CITIfile Prefixes

File Prefix	File Contents	Notes
RD_	Raw Data	Raw (uncorrected data array(s).
DD_	Data Data	Error corrected data array(s)
FD_	Formatted Data	Corrected & formatted data array.
DM_	Display Memory	File holds one memory.
MA_	Display Memory All	Holds all memories in 8510.
CS_	Cal Set	One set of calibration data.
CA	Cal Set All	All sets of calibration data.
DT_	Delay Table	One delay table.

DD_MYDATA is an example of a file name for a file that contains one array of corrected data.

The current 8510 CITIfile implementation is unable to read files unless they have the prefixes above. It is expected that a future 8510 revision will remove this restriction.

Agilent 8700 Series CITIfile

Storing a data file from an 8700-series analyzer in CITIfile format requires that you choose the SAVE USING ASCII option.

The 8700 series of instruments check the last two characters of the filename to determine what is stored in the file. The file endings for an 8700 CITIfile are listed in the following table.

Agilent 8700 CITIfile Filenames

Last Two Chars	File Contents	Notes
Rx	Raw Data	x = 1 = channel 1. x = 5 for channel 2.
Dx	Data Data	x = channel number.
Fx	Formatted Data	x = channel number.
Mx	Display Memory	x = channel number.
xy	Cal Set	x = channel number. y = number of error coefficient arrays in the file. y is displayed in hexadecimal.

FILE1D1 is an example of a file name for a file that contains corrected data for channel #1. FILE1R5 is a filename for raw data arrays from channel #2. MYFILE2C is an example of a name for a file that contains a cal set used by channel #2, with 12 arrays of data (hexadecimal C).

To load data from disk into an 8700-series instrument, there must be a matching instrument state file to go with the data that is being loaded. Consult the 8700-series documentation for more information.

ADS Simulator Input Syntax

This topic provides information related to Advanced Design System's simulator (ADSSim). While this is not an all inclusive document with regards to ADSSim, the information provided here should help you accomplish tasks related to using the ADSSim in your development environment. The ADSSim is supported on the platforms specified in the installation documentation for your system, in the section "Check the System Requirements".

The simulator can be run from within the design environment, as well as from a command line. Before running the simulator, ensure that your system is ready to run the simulator by reviewing these topics:

- [Setting Environment Variables](#)
- [Codewording and Security](#)
- [Running a Simulation from the Command Line](#)

Setting Environment Variables

Before running the ADSsim, the following environment variables must be set:

Environment Variables Required for the ADS Simulator (ADSsim)

Variable	PC Setting	UNIX/Linux Setting
HPEESOF_DIR	<ADS_install_dir>	<ADS_install_dir>
COMPL_DIR	%HPEESOF_DIR%	\$HPEESOF_DIR
PATH	%PATH%;%HPEESOF_DIR%\bin	\$PATH:\$HPEESOF_DIR/bin

These environment variables tell your system the location of the ADS shared libraries/DLLs and device libraries. COMPL_DIR defines the location of component libraries. The COMPL_DIR variable typically uses the same value as HPEESOF_DIR unless the component libraries are located elsewhere. However, the majority of users should be able to set COMPL_DIR to the same value as HPEESOF_DIR.

To set the PC environment variables, use the following commands:

```
set HPEESOF_DIR = <ADS_install_dir>
set COMPL_DIR = %HPEESOF_DIR%
set PATH=%PATH%;%HPEESOF_DIR%\bin
```

To set the UNIX environment variables using the Korn or Bourne Shells, add the following to your ~/.profile :

```
export HPEESOF_DIR = <ADS_install_dir>
export COMPL_DIR = $HPEESOF_DIR
export PATH=$PATH:$HPEESOF_DIR/bin
```

To set the UNIX environment variables using the C Shell, add the following to your ~/.cshrc :

```
setenv HPEESOF_DIR <ADS_install_dir>
setenv COMPL_DIR $HPEESOF_DIR
setenv PATH $PATH:$HPEESOF_DIR/bin
```

Platform-Specific Variables

A platform-specific variable must also be set before running the ADS simulator. Be sure to set the variable to support the simulation mode you are using: 32-bit or 64-bit.

Note
The following commands for UNIX and Linux systems are for those using the Korn or Borne Shells. Use the appropriate equivalent command if using the C Shell.

HP-UX

32-bit (64-bit not supported):

```
export SHLIB_PATH="$HPEESOF_DIR/adsptolemy/lib.hpux11:$SHLIB_PATH"  
export SHLIB_PATH="$HPEESOF_DIR/lib/hpux11:$SHLIB_PATH"
```

Red Hat Linux

32-bit:

```
export LD_LIBRARY_PATH="$HPEESOF_DIR/adsptolemy/lib.linux_x86:$LD_LIBRARY_PATH"  
export LD_LIBRARY_PATH="$HPEESOF_DIR/lib/linux_x86:$LD_LIBRARY_PATH"
```

64-bit:

```
export LD_LIBRARY_PATH="$HPEESOF_DIR/adsptolemy/lib.linux_x86_64:$LD_LIBRARY_PATH"  
export LD_LIBRARY_PATH="$HPEESOF_DIR/lib/linux_x86_64:$LD_LIBRARY_PATH"
```

Solaris 8, 9, and 10

32-bit:

```
export LD_LIBRARY_PATH="$HPEESOF_DIR/adsptolemy/lib.sun58:$LD_LIBRARY_PATH"  
export LD_LIBRARY_PATH="$HPEESOF_DIR/lib/sun58:$LD_LIBRARY_PATH"
```

64-bit:

```
export LD_LIBRARY_PATH="$HPEESOF_DIR/adsptolemy/lib.sun58_64:$LD_LIBRARY_PATH"  
export LD_LIBRARY_PATH="$HPEESOF_DIR/lib/sun58_64:$LD_LIBRARY_PATH"
```

MS Windows

HPEESOF_DIR and PATH can be set using Control Panel > System > Advanced > Environment Variables, or by using a DOS batch file.

32-bit:

```
path %HPEESOF_DIR%\adsptolemy\lib.win32;%PATH%
path %HPEESOF_DIR%\lib\win32;%PATH%
```

64-bit:

```
path %HPEESOF_DIR%\adsptolemy\lib.win32_64;%PATH%
path %HPEESOF_DIR%\lib\win32_64;%PATH%
```

Codewording and Security

The ADSsim is a secured program that requires, at a minimum, a license for the E8881 Linear Simulator to run. Depending on the type of simulation, additional licenses may be required. Also, the license file location may require defining the variable AGILEESOFD_LICENSE_FILE. For more information on codewording and security, see the topic on setting up licenses in the installation documentation for your platform.

Running a Simulation from the Command Line

Besides using the design environment's user interface to run a simulation, you can also run a simulation from a command line using the hpeesofsim command. Before using this command, ensure your system is ready to run the simulator by reviewing these topics:

- [Setting Environment Variables](#)
- [Codewording and Security](#)

The ADSsim can be invoked using the following syntax:

```
hpeesofsim [-r output_rawfile_name] [netlist_inputfile_name]
```

A list of available options can be generated using the following command:

```
hpeesofsim -o
```

The netlist that ADSsim generates can be reviewed and modified. The following example is a netlist generated for a resistive PI pad:

```
Options ResourceUsage=yes UseNutmegFormat=no
TopDesignName="C:\apps\ads2005a\arf_0705_prj\networks\PI_pad"

R:R3 _net28 _net27 R=16 Ohm Noise=yes
R:R2 _net28 0 R=80 Ohm Noise=yes
R:R1 _net27 0 R=80 Ohm Noise=yes

S_Param:SP1 CalcS=yes CalcY=no CalcZ=no GroupDelayAperture=1e-4 FreqConversion=no
FreqConversionPort=1 StatusLevel=2 CalcNoise=no SortNoise=0 BandwidthForNoise=1.0 Hz

DevOpPtLevel=0 \
SweepVar="freq" SweepPlan="SP1_stim" OutputPlan="SP1_Output"
SweepPlan: SP1_stim Start=1.0 GHz Stop=10.0 GHz Step=0.1 GHz
OutputPlan:SP1_Output \
Type="Output" \
UseEquationNestLevel=yes \
EquationNestLevel=2 \
UseSavedEquationNestLevel=yes \
SavedEquationNestLevel=2

Port:Term1 _net27 0 Num=1 Z=50 Ohm Noise=yes
Port:Term2 _net28 0 Num=2 Z=50 Ohm Noise=yes
```

If the option `-r output_rawfile_name` is not given in the command, simulation results will be written to the `spectra.raw` file. Simulation results can be written to both a readable raw file and a binary dataset file. To create a readable raw file, you may need to modify the options listed at the beginning of the netlist. For example, if a netlist contains the options shown in this example:

```
Options ResourceUsage=yes UseNutmegFormat=no
TopDesignName="C:\my_projects\DataAccess_prj\networks\test.ds"
```

change the options line to:

```
Options ResourceUsage=yes UseNutmegFormat=no ASCII_Rawfile=yes
TopDesignName="C:\my_projects\DataAccess_prj\networks\test.ds"
```

`TopDesignName` is the name of the dataset file to be written, which is a binary file. You can use the `dsdump` command (located in `$HPEESOF_DIR/bin`) to view the dataset file as shown in this example:

```
dsdump test.ds
```

General Syntax

This topic uses the following typographical conventions:

Typographic Conventions

Type Style	Used For
[. . .]	Data or character fields enclosed in brackets are optional.
<i>italics</i>	Names and values in italics must be supplied
bold	Words in bold are ADS simulator keywords and are also required.

The ADS Simulator Syntax

The following sections outline the basic language rules. For details about restrictions concerning reserved names, see [Reserved Names and Name Spaces](#).

Field Separators

A delimiter is one or more blanks or tabs.

Continuation Characters

A statement may be continued on the next line by ending the current line with a backslash and continuing on the next line.

Name Fields

A name may have any number of letters or digits in it but must not contain any delimiters or non alphanumeric characters. The name must begin with a letter or an underscore (_).

Parameter Fields

A parameter field takes the form name = value, where name is a parameter keyword and value is either a numeric expression, the name of a device instance, the name of a model or a character string surrounded by double quotes. Some parameters can be indexed, in which case the name is followed by [i] , [i,j] , or [i,j,k] . i , j , and k must be integer constants or variables.

Node Names

A node name may have any number of letters or digits in it but must not contain any delimiters or non alphanumeric characters. If a node name begins with a digit, then it must consist only of digits.

Lower/Upper Case

The ADS Simulator is case sensitive.

Units and Scale Factors

The fundamental units for the ADS Simulator are shown in the following table. A parameter with a given dimension assumes its value has the corresponding units. For example, for a resistance, $R=10$ its assumed to be 10 ohms.

Fundamental Units in ADS

Dimension	Fundamental Unit
Frequency	Hertz
Resistance	Ohms
Conductance	Siemens
Capacitance	Farads
Inductance	Henries
Length	meters
Time	seconds
Voltage	Volts
Current	Amperes

Advanced Design System 2008

Power	Watts
Distance	meters
Temperature	Celsius

Recognizing Scale Factors

Variations on the fundamental units in ADS are referred to as scale factors. A scale factor is a single word that begins with a letter or an underscore character (_). The remaining characters, if any, consist of letters, digits, and underscores. The value of a scale factor is resolved using the following rules in the order shown:

1. If the scale factor exactly matches one of the predefined scale-factor words (see the following table), then use the numerical equivalent; otherwise, go to rule 2.

Predefined Scale Factor Words

Scale Factor Word	Numerical Equivalent	Meaning
mil	2.54×10^{-5}	mils
mils	2.54×10^{-5}	mils
in	2.54×10^{-2}	inches
ft	$12 \times 2.54 \times 10^{-2}$	feet
mi	$5280 \times 12 \times 2.54 \times 10^{-2}$	miles
cm	1.0×10^{-2}	centimeters
PHz	1.0×10^{15}	
dB	1.0	decibels
nmi	1852	nautical miles

2. If the scale factor exactly matches one of the scale-factor units (see following table) except for m, then use the numerical equivalent; otherwise, go to rule 3.

Scale Factor Units

Scale Factor Unit	Numerical Equivalent	Meaning
A	1.0	Amperes
F	1.0	Farads
H	1.0	Henries
Hz	1.0	Hertz
meter meters	1.0	meters

metre metres		
Ohm Ohms	1.0	Ohms
S	1.0	Siemens
sec	1.0	seconds
V	1.0	Volts
W	1.0	Watts

3. If the first character of the scale factor is one of the legal scale-factor prefixes (see the following table), then use the numerical equivalent; otherwise, go to rule 4.

Scale Factor Prefixes

Prefix	Numerical Equivalent	Meaning
T	10 ¹²	Tera
G	10 ⁹	Giga
M	10 ⁶	Mega
K	10 ³	kilo
k	10 ³	kilo
_ (underscore)	1	(no scale)
m	10 ⁻³	milli
u	10 ⁻⁶	micro
n	10 ⁻⁹	nano
p	10 ⁻¹²	pico
f	10 ⁻¹⁵	femto
a	10 ⁻¹⁸	atto

4. The scale factor is not recognized.

Important considerations include:

- Scale factors are case sensitive.
- A single m means milli , not meters .
- A lower case f by itself means femto . An upper case F by itself means Farad .
- A lower case a by itself means atto . An upper case A by itself means Ampere .
- The imperial units (mils , in , ft , mi , nmi) do not accept prefixes.
- The ADS simulator will report a warning if an unrecognized scale factor is encountered, and use a scale-factor value of 1.0.
- It is not required that the characters following a scale-factor prefix match one of the scale-factor units.
- There are no scale factors for dBm , dBW , or temperature. ADS functions are provided to convert these values to the corresponding fundamental units (Watts and Celsius).

Booleans

Many devices, models, and analyses have parameters that are boolean valued. Zero is used to represent false or no, whereas any number besides zero represents true or yes. The keywords yes and no can also be used.

Global Nodes

Global nodes are user-defined nodes which exist throughout the hierarchy. The global nodes must be defined on the first lines in the netlist. They must be defined before they are used.

General Form:

```
globalnode nodename1 [nodename2 ] [... nodenameN ]
```

Example:

```
globalnode sumnode my_internal_node
```

Device Pin Names

To support the schematic device pin names in a dataset, the device pin names are passed to the simulator through the netlist. For each unique device type in the design, the pin-mapping information is passed in the following format:

```
mapping {
```

```
    pinMapping {
```

```
        device-name pinMap pinMap
```

```
        device-name pinMap pinMap
```

```
        device-name pinMap pinMap
```

```
}
```

```
}
```

where pinMap is:

```
< integer >:"< pinName >"
```

Example:

```
mapping {  
  pinMapping {  
    R 1:"PLUS" 2:"MINUS"  
    V_Source 1:"PLUS" 2:"MINUS"  
    BJTM1 1:"c" 2:"b" 3:"e"  
    C 1:"PLUS" 2:"MINUS"  
  }  
}
```

Additional information about mapping device pin names and the simulator syntax includes:

- The keywords mapping , pinCurrent , and pinMapping are reserved. They cannot be used as names for instances, nodes, variables, etc.
- The mapping block appears at the top level of the netlist. A mapping block cannot exist at the sub-circuit level. The mapping block can exist anywhere within the top level, preferably at the bottom or top of the netlist.
- Currently only one mapping and one pinMapping block can exist in a circuit.

Comments

Comments are introduced into an ADS Simulator file with a semicolon; they terminate at the end of the line. Any text on a line that follows a semicolon is ignored. Also, all blank lines are ignored.

Statement Order

Models can appear anywhere in the netlist. They do not have to be defined before a model instance is defined.

Some parameters expect a device instance name as the parameter value. In these cases, the device instance must already have been defined before it is referenced. If not, the device instance name can be entered as a quoted string

using double quotes (").

Naming Conventions

The full name for an instance parameter is of the form:

```
[pathName].instanceName.parameterName[index]
```

where pathName is a hierarchical name of the form

```
[pathName].subnetworkInstanceName
```

The same naming convention is used to reference nodes, variables, expressions, functions, device terminals, and device ports.

For device terminals, the terminal name can be either the terminal name given in the device description, or tn where n is the terminal number (the first terminal in the description is terminal 1, etc.). Device ports are referenced by using the name pm , where m is the port number (the first pair of terminals in the device description is port 1, etc.).

Note that t1 and p1 both correspond to the current flowing into the first terminal of a device, and that t2 corresponds to the current flowing into the second terminal. If terminals one and two define a port, then the current specified by t2 is equal and opposite to the current specified by t1 and p1 .

Currents

The only currents that can be accessed for simulation, optimization, or output purposes are the state currents.

State currents

Most devices are voltage controlled, that is, their terminal currents can be calculated given their terminal voltages. Circuits that contain only voltage-controlled devices can be solved using node analysis. Some devices, however, such as voltage sources, are not voltage controlled. Since the only unknowns in node analysis are the node voltages, circuits that contain non-voltage-controlled devices cannot be solved using node analysis. Instead, modified node analysis is used. In modified node analysis, the unknown vector is enlarged. It contains not only the node voltages but the branch currents of the non-voltage-controlled devices as well. The branch currents that appear in the vector of

unknowns are called state currents. Since the ADS Simulator uses modified node analysis, the values of the state currents are available for output.

If the value of a particular current is desired but the current is not a state current, insert a short in series with the desired terminal. The short does not affect the behavior of the circuit but does create a state current corresponding to the desired current.

To reference a state current, use the device instance name followed by either a terminal or port name. If the terminal or port name is not specified, the state current defaults to the first state current of the specified device. Note that this does not correspond to the current through the first port of the device whenever the current through the first port is not a state current. For some applications, the positive state current must be referenced, so a terminal name of t1 or t3 is acceptable but not t2 . Using port names avoids this problem. The convention for current polarity is that positive current flows into the positive terminal.

Integer Formats for Simulator Input

There are notation rules for simulator input of hex, octal and binary integer constants.

- Integers starting with "0x"/"0X" are read in hexadecimal format.
- Integers starting with "0b"/"0B" are read in binary format.
- Integers starting with "0" and not followed by any of [X,x,B,b] are read in octal format.
- Integers starting with [1-9] are read in decimal format.

Real numbers are always read in decimal format.

Instance Statements

General Form:

```
type [ :name ] node1 ... nodeN [ [ param=value ] ... ]  
type [ :name ] [ [ param=value] ... ]
```

Examples:

```
ua741:OpAmp in out out  
C:C1 2 3 C=10pf  
HB: Distortion1 Freq=10GHz
```

The instance statement is used to define to the ADS Simulator the information unique to a particular instance of a device or an analysis. The instance statement consists of the instance type descriptor and an optional name preceded by a colon. If it is a device instance with terminals, the nodes to which the terminals of the instance are connected come next. Then the parameter fields for the instance are defined. The parameters can be in any order. The nodes, though, must appear in the same order as in the device or subnetwork definition.

The type field may contain either the ADS Simulator instance type name, or a user-supplied model or subnetwork name. The name can be any valid name, which means it must begin with a letter, can contain any number of letters and digits, must not contain any delimiters or non alphanumeric characters, and must not conflict with other names including node names.

Model Statements

General Form:

```
model name type [ [param = value ] ... ]
```

Examples:

```
model NPNbjt bjt NPN=yes Bf=100 Js=0.1fa
```

Often characteristics of a particular type of element are common to a large number of instances. For example, the saturation current of a diode is a function of the process used to construct the diode and also of the area of the diode. Rather than describing the process on each diode instantiation, that description is done once in a model statement and many diode instances refer to it. The area, which may be different for each device, is included on each instance statement. Though it is possible to have several model statements for a particular type of device, each instance may only reference at most one model. Not all device types support model statements.

The name in the model statement becomes the type in the instance statement. The type field is the ADS Simulator-defined model name. Any parameter value not supplied will be set to the model's default value.

Most models, such as the diode or bjt models, can be instantiated with an instance statement. There are exceptions. For instance, the Substrate model cannot be instantiated. Its name, though, can be used as a parameter value for the Subst parameter of certain transmission line devices.

Subnetwork Definitions

General Form:

```

define subnetworkName ( node1 ... nodeN )

[parameters name1 = [value1] ... name n = [value n ] ]

.
.
.

elementStatements

.
.
.

end [subnetworkName ]
    
```

Examples:

```

define DoubleTuner (top bottom left right)
parameters vel=0.95 r=1.0 l1=.25 l2=.25
tline:tuner1 top bottom left left len=l1 vel=vel r=r
tline:tuner2 top bottom right right len=l2 vel=2*vel r=r
end DoubleTuner
DoubleTuner:InputTuner t1 b2 3 4 l1=0.5
    
```

A subnetwork is a named collection of instances connected in a particular way that can be instantiated as a group any number of times by subnetwork calls. The subnetwork call is in effect and form, an instance statement. Subnetwork definitions are simply circuit macros that can be expanded anywhere in the circuit any number of times. When an instance in the input file refers to a subnetwork definition, the instances specified within the subnetwork are inserted into the circuit. Subnetworks may be nested. Thus a subnetwork definition may contain other subnetworks. However, a subnetwork definition cannot contain another subnetwork definition. All the definitions must occur at the top level.

An instance statement that instantiates a subnetwork definition is referred to as a subnetwork call. The node names (or numbers) specified in the subnetwork call are substituted, in order, for the node names given in the subnetwork definition. All instances that refer to a subnetwork definition must have the same number of nodes as are specified in the subnetwork definition and in the same order. Node names inside the subnetwork definition are strictly local unless they are global nodes defined with a globalnode statement. A subnetwork definition with no nodes must still include the parentheses () .

Parameter specification in subnetwork definitions is optional. Any parameters that are specified are referred to by name followed by an equals sign and then an optional default value. If, when making a subnetwork call in your input

file, you do not specify a particular parameter, then this default value is used in that instance. Subnetwork parameters can be used in expressions within the subnetwork just as any other variable.

Subnetworks are a flexible and powerful way of developing and maintaining hierarchical circuits. Parameters can be used to modify one instance of a subnetwork from another. Names within a subnetwork can be assigned without worrying about conflicting with the same name in another subnetwork definition. The full name for a node or instance include its path name in addition to its instance name. For example, if the above subnetwork is included in `subckt2` which is itself included in `subckt1`, then the full path name of the length of the first transmission line is `subckt1.subckt2.tuner1.len`.

Only enough of the path name has to be specified to unambiguously identify the parameter. For example, an analysis inside `subckt1` can reference the length by `subckt2.tuner1.len` since the name search starts from the current level in the hierarchy. If a reference to a name cannot be resolved in the local level of hierarchy, then the parent is searched for the name, and so on until the top level is searched. In this way, a sibling can either inherit its parent's attributes or define its own.

Expression Capability

The ADS Simulator has a powerful and flexible symbolic expression capability which enables you to define variables, functions, and expressions in the netlist. These can then be used to define other functions and expressions to specify device parameters and optimization goals, etc.

Variables are your basic building blocks. For example, declaring `"var1 = 2.5"` assigns the value "2.5" to the variable "var1." Functions are simply parameterized variables such as $F(x) = x + x^2$. You can combine variables, constants, functions, and operators to form expressions. An expression can be a simple or complex series of variables, operators, and functions that evaluates to a single value. For example, `y = abs(0.3-j*0.3)` returns a value of 3.015.

The names for variables, expressions, and functions follow the same hierarchy rules that instance and node names do. Thus, local variables in a subnetwork definition can assume values that differ from one instance of the subnetwork to the next.

Functions and expressions can be defined either globally or locally anywhere in the hierarchy. All variables are local by default. Local variables are known in the subnetwork in which they are defined, and all lower subnetworks; they are not known at higher levels. Variables defined at the root (the top level) are known everywhere within the circuit. To specify a global variable, the global keyword must precede the variable name. The global keyword causes the variable to be defined at the root of the hierarchy tree regardless of the lexical location.

Examples:

```
global var1 = 2.718
```

The expression capability includes the standard math operations of + - / * ^ in addition to parenthesis grouping. Scale factors are also allowed in general expressions and have higher precedence than any of the math operators. For more information about units and scale factors, see [Units and Scale Factors](#).

For complete information about variables, constants, available functions, and using them to define simulator expressions, see the [Simulator Expressions](#) documentation.

C-Preprocessor

Before being interpreted by the ADS Simulator, all input files are run through a built-in preprocessor based upon a C preprocessor. This brings several useful features to the ADS Simulator, such as the ability to define macro constants and functions, to include the contents of another file, and to conditionally remove statements from the input. All C preprocessor statements begin with # as the first character.

Unfortunately, for reasons of backward compatibility, there is no way to specify include directories. The standard C preprocessor ``` -I "` option is not supported; instead, ``` -I "` is used to specify a file for inclusion into the netlist.

File Inclusion

Any source line of the form

```
#include "filename"
```

is replaced by the contents of the file `filename`. The file must be specified with an absolute path or must reside in either the current working directory or in `/$HPPEESOF_DIR/circuit/components/`.

Library Inclusion

The C preprocessor automatically includes a library file if the `-N` command line option is not specified and if such a file exists. The first file found in the following list is included as the library:

```
$HPPEESOF_DIR/circuit/components/gemlib  
$EESOF_DIR/circuit/components/gemlib  
$GEMLIB  
.gemlib  
~/gemlib
```

```
~/gemini/gemlib
```

A library file is specified by the user using the `-I filename` command line option. More than one library may be specified. Specifying a library file prevents the ADS Simulator from including any of the above library files.

Macro Definitions

A macro definition has the form;

```
#define name replacement-text
```

It defines a macro substitution of the simplest kind--subsequent occurrences of the token name are replaced by replacement-text . The name consists of alphanumeric characters and underscores, but must not begin with a numeric character; the replacement text is arbitrary. Normally the replacement text is the rest of the line, but a long definition may be continued by placing a `"\"` at the end of each line to be continued. Substitutions do not occur within quoted strings. Names may be undefined with

```
#undef name
```

It is also possible to define macros with parameters. For example,

```
#define to_celcius(t) (((t)-32)/1.8)
```

is a macro with the formal parameter `t` that is replaced with the corresponding actual parameters when invoked. Thus the line

```
options temp=to_celcius(77)
```

is replaced by the line

```
options temp=(((77)-32)/1.8)
```

Macro functions may have more than one parameter, but the number of formal and actual parameters must match.


Macros may also be defined using the `-D` command line option.

Conditional Inclusion

It is possible to conditionally discard portions of the source file. The `#if` line evaluates a constant integer expression, and if the expression is non-zero, subsequent lines are retained until an `#else` or `#endif` line is found. If an `#else` line is found, any lines between it and the corresponding `#endif` are discarded. If the expression evaluates to zero, lines between the `#if` and `#else` are discarded, while those between the `#else` and `#endif` are retained. The conditional inclusion statements nest to an arbitrary level of hierarchy. The following operators and functions can be used in the constant expression;

<code>!</code>	Logical negation.
<code> </code>	Logical or.
<code>&&</code>	Logical and.
<code>==</code>	Equal to.
<code>!=</code>	Not equal to.
<code>></code>	Greater than.
<code><</code>	Less than.
<code>>=</code>	Greater than or equal to.
<code><=</code>	Less than or equal to.
<code>+</code>	Addition.
<code>defined(x)</code>	1 if x defined, 0 otherwise.

The `#ifdef` and `#ifndef` lines are specialized forms of `#if` that test whether a name is defined.

 **Caution**
 Execution of preprocessor instructions depend on the order in which they appear on the netlist. When using preprocessor statements make sure that they are in the proper order. For example, if an `#ifdef` statement is used to conditionally include part of a netlist, the corresponding `#define` statement is contained in a separate file and `#include` is used to include the content of the file into the netlist, the `#include` statement will have to appear before the `#ifdef` statement for the expression to evaluate correctly.

Data Access Component

Advanced Design System 2008

The Data Access Component provides a clean, unified way to access tabular data from within a simulation. The data may reside in either a text file of a supported, documented format (e.g. discrete MDIF, model MDIF, Touchstone, CITIfile), or a dataset. It provides a variety of access methods, including lookup by index/value, as well as linear, cubic spline and cubic interpolation modes, with support for derivatives.

The Data Access Component provides a "handle" with which one may access data from either a text file or dataset for use in a simulation. The DAC is implemented as a cktlib subnetwork fragment with internally known expressions names (e.g. DAC, _TREE) that are assigned via _VarEqn calls such as `read_data()` and `access_all_data()`. The accessed data can be used by other components (including models, devices, variables, subnetwork calls and other DAC instances) in the netlist, either by the specific file syntax or via the VarEqn function `dep_data()`.

The DAC can also be used to supply parameters to device and model components from text files and datasets. In this case, the `AllParams` device/model parameter is used to refer to a DAC component. The component's parameters will then be accessed from the DAC and supplied to the instance. Care is taken to ensure that only matching (between parameter names in the component definition and DAC dependent column names) data is used. Also, parameter data can be assigned "inline" - as is usually done - in which case the inline data takes precedence over the DAC data.

As the DAC component is composed of just a parameterized subnetwork, it allows alterations (sweep, tune, optimize, yield) of its parameters. Consequently any component that uses DAC data via file, `dep_data()` or `AllParams` will automatically be updated when a DAC parameter is altered. A caveat with sweeping over files using `AllParams` is that all the files must contain the same number of dependent columns of data.

Below is an example definition of a simple DAC component that accesses discrete values from a text file:

```
#uselib "ckt" , "DAC"
DAC: DAC1 File="C:\jeffm\ADS_testing\ADS13_test_prj/.data\SweptData.ds"
Type="dataset" Block="S" InterpMode="linear" InterpDom="ri" iVar1="X"
iVal1=X iVar2="freq" iVal2=freq
S_Port:S2P1 _net1 0 _net6 0 S[1,1]=file{DAC1, "S[1,1]"}
S[1,2]=file{DAC1, "S[1,2]"} S[2,1]=1 S[2,2]=0 Recip=no

dindex = 1
DAC:atc1 File="vdcr.mdf" Type="dscr" \
InterpMode="index_lookup" iVar1=1 iVal1=dindex
```

And its use to provide the resistance value to a pair of circuit components:

```
R:R1 n1 0 R=file{atc1, "R"} kOhm
R:R2 n1 0 R=dep_data(atc1, "R") kOhm
Here, it provides the value to a variable:
V1 = file{atc1, "Vdc"}
```

V1 could be used elsewhere in the circuit, as expected.

In this example, a scaling factor applied to the result of a DAC access is shown:

```
File = "atc.mdf"  
Type = "dscr"  
Mode="index_lookup"  
Cnom = "Cnom"  
DAC:atc_s File=File Type=Type InterpMode=Mode iVar1=1 iVal1 = Cs_row  
C:Cs n1 n2 C=file{atc_s, Cnom} Pf}}
```

In this example, a use of `AllParams` is shown to enter model parameters from a text file:

```
File = "c:\gemini\vdcr.mdf"  
Type = "dscr"  
Mode="index_lookup"  
DAC:dac1 File=File Type=Type InterpMode=Mode iVar1=1 iVal1 = ix  
model rml R_Model R=0 AllParams = dac1._DAC  
rml:rml1 n3 0
```

Reserved Names and Name Spaces

When developing a netlist using the ADS simulator syntax, there are several different places in a netlist where names must be supplied, such as instance names and variable names. When selecting names, be sure that you do not use a reserved name (keyword).

There are six name categories: design/subnetwork, instance, parameter, model, variable, and node. These are known to the simulator as name spaces. Additionally, there are five reserved name groups. Each name space has one or more reserved name groups associated with it. This means that when choosing a name for a category such as a parameter name, you cannot use any of the names in the reserved name groups associated with the parameter name space.

When a user-entered name is parsed, it is checked to see that it is not in the group of reserved names associated with the name space in which it is used. If the user-entered name matches a reserved name, the simulator will issue an error message and terminate. The simulator is case-sensitive, so the case of characters in a name must match, as well as the characters themselves.



Note

Please be aware that names associated with AEL expressions (for example, node names, which are used to name items in the dataset) may have other restrictions that are not noted here. These restrictions are outside the domain of the simulator itself, and follow from the design of the AEL and/or the dataset codes. Only those name restrictions that are imposed by the ADS simulator parser are shown here.

The following tables provide details about the six name spaces, their associated reserved name groups, and lists of the individual reserved names. At the end of the section is a complete alphabetical listing of all reserved names. In addition to these lists, any built-in component name should not be used as a design/subnetwork name. Refer to the

component catalogs for the built-in component names.

Note
 You can use the hpeesofsim command to view all keywords. Enter the following command from the command line:

```
hpeesofsim -h keywords
```

- For a list of the name spaces with descriptions, examples, and their associated reserved name groups, see the table [ADS Simulator Namespaces and Associated Reserved Name Groups](#).
- For a list of reserved names composing each reserved name group, see these tables:
 - [Parser Reserved Names Group](#)
 - [Reserved Names Group](#)
 - [Predefined Expression Reserved Names Group](#)
 - [Predefined Variable Reserved Names Group](#)
 - [Predefined Function Reserved Names Group](#)
- For an alphabetical listing of reserved names, see the table [ADS Reserved Words - Alphabetical List](#).

ADS Simulator Namespaces and Associated Reserved Name Groups

Name Space	Reserved Name Group	Description and Examples
Design/subnetwork	Parser Reserved Names see table Parser Reserved Names Group Reserved Names see table Reserved Names Group	These are the names given to the designs that contain the top-level circuit and any subnetwork definitions. Examples: a top-level design named MyLowNoiseAmp and a subnetwork design named MyBandpassFilter .
Device/subnetwork instance	Parser Reserved Names see table Parser Reserved Names Group Reserved Names see table Reserved Names Group Predefined Expression Names see table Predefined Expression Reserved Names Group Predefined Variable Names see table Predefined Variable Reserved Names Group Predefined Function Names see table Predefined Function Reserved Names Group	These are the labels given to components that are placed in the design. Examples: an R component labelled R1 and a MyBandpassFilter component labelled Filter1 .
Subnetwork parameter	Parser Reserved Names see table Parser Reserved Names Group Reserved Names see table Reserved Names Group	These are the names given to parameters defined for a subnetwork or user-defined model. Example: a parameter named CenterFreq defined for the

Advanced Design System 2008

		subnetwork MyBandpassFilter .
Model	<p>Parser Reserved Names see table Parser Reserved Names Group</p> <p>Reserved Names see table Reserved Names Group</p>	<p>These are the instance names associated with the model definition and model instance. Example: a BJT_Model with the InstanceName BJTM1 .</p>
Variable/expression	<p>Parser Reserved Names see table Parser Reserved Names Group</p> <p>Reserved Names see table Reserved Names Group</p> <p>Predefined Expression Names see table Predefined Expression Reserved Names Group</p> <p>Predefined Variable Names see table Predefined Variable Reserved Names Group</p> <p>Predefined Function Names see table Predefined Function Reserved Names Group</p>	<p>These are the names given to VarEqn items. Example: a variable named Rnominal.</p>
Node	<p>Parser Reserved Names see table Parser Reserved Names Group</p>	<p>These are wire/pin labels. Example: a wire labelled Vout.</p>

Parser Reserved Names Group

Parser Reserved Names (a – m)	(n – _v)
aele AllParams Allparams allParams allparams All_Params All_params all_Params all_params and by define discrete distcompname doe else elseif end	nested no nodoe noopt nostat not notequals notune opt or parameters pinMapping ppt stat static then to tune

Advanced Design System 2008

endif equals file gauss global globalnode ground if inline local lognormal logScale mapping model	unconst uniform yes _M _VER _VEr _VeR _Ver _vER _vEr _veR _ver
--	---

Reserved Names Group

Reserved Names
delay icor j nfmin noise pi portz rn sopt

Predefined Expression Reserved Names Group

Predefined Expression Names
gaussian omega tempkelvin

Predefined Variable Reserved Names Group

Predefined Variable Names (b - s)	(t - __z)
boltzmann c0 CostIndex dcSourceLevel	temp time timestep tinyreal

Advanced Design System 2008

DefaultValue DeviceIndex DF_DefaultInt DF_Value DF_ZERO_OHMS doeindex doeIter e e0 freq hugereal LinearizedElementIndex ln10 logNodesetScale logRforce logRshunt mcindex mcTrial noisefreq Nsample optIter planck qelectron scale ScheduleCycle sourceLevel ssfreq	tnom tranorder u0 version_check _ABM_Phase _ABM_SourceLevel _ac_state _dc_state _default _fc _freq1 _freq10 _freq11 _freq12 _freq2 _freq3 _freq4 _freq5 _freq6 _freq7 _freq8 _freq9 _harm _hb_state _p2dInputPower _sigproc_state _sm_state _sp_state _tr_state __fdd __fdd_v __s __y __z
--	--

Predefined Function Reserved Names Group

Predefined Function Names (a - k)	(l - _x)
abs access_all_data access_all_data_new access_data access_data_new acos acosh amp_harm_coef arcsinh	length lfsr limit_warn list ln log log10 log_amp log_amp_cas

arctan	lookup
asin	mag
asinh	makearray
atan	max
atan2	min
atanh	miximt_coef
attenuator_warn	miximt_poly
avg_dia	mp_fetchS21
bin	mp_poly_gain
bitseq	multivar_access
ceil	multivar_tree
check_indep_limits	multi_freq
coef_count	names
complex	nf
compute_mp_poly_coef	norm
compute_poly_coef	phase
conj	phasedeg
cos	phaserad
cosh	phase_noise_pwl
cos_pulse	polar
cot	polarcpv
coth	pow
cpx_gain_poly	PrintErrorMessage
ctof	pulse
ctok	pwl
cxform	pwlr
damped_sin	pwlr_tr
db	qinterp
dbm	rad
dbmtoa	ramp
dbmtov	rawtoarray
dbmtow	readdata
dbpolar	readlib
dbwtow	readraw
deembed	read_data
deg	read_lib
delay	read_SSpower
dep_data	real
deriv	rect
dphase	rem
dsexpr	repeat
dstoarray	ripple
d_atan2	rms
echo	rpsmooth
embedded_ptolemy_exec	scalearray
ensure_ext	sens
erf_pulse	setDT
eval_miso_poly	sffm
eval_poly	sgn

exp	sin
exp_pulse	sinc
fetch_envband	sinh
floor	spectrum
fmod	sprintf
fread	sqrt
freq_mult_coef	status_print
freq_mult_poly	step
ftoc	strcat
ftok	stypexform
gcdata_to_poly	sym_set
generate_gmsk_iq_spectra	system
generate_gmsk_pulse_spectra	tan
generate_pi_qpsk_spectra	tanh
generate_pulse_train_spectra	thd
generate_qam16_spectra	toi
generate_qpsk_pulse_spectra	transform
get_array_size	v
get_attribute	value
get_block	vlsb
get_fund_freq	vnoise
get_indep_limits	vss
get_LSfreqs	vswrpolar
get_LSpowrs	vusb
get_max_points	WarnTimeDomainDeembed
get_S2D_attribute	window
hpvar_to_vs	wtodbm
hypot	_bitwise_and
i	_bitwise_asl
ilsb	_bitwise_asr
imag	_bitwise_not
impulse	_bitwise_or
imt_hbdata_to_array	_bitwise_xnor
imt_hpvar_to_array	_bitwise_xor
include_SSpower	_discrete_density
index	_divn
innerprod	_gaussian
inoise	_gaussian_tol
int	_get_fnom_freq
internal_generate_gmsk_iq_spectra	_get_fund_freq_for_fdd
internal_generate_gmsk_pulse_spectra	_lfsr
internal_generate_pi_qpsk_spectra	_mvgaussian
internal_generate_pulse_train_spectra	_mvgaussian_cov
internal_generate_qam16_spectra	_nfmin
internal_generate_qpsk_pulse_spectra	_n_state
internal_get_fund_freq	_phase_freq
internal_window	_pwl_density
interp	_pwl_distribution
interp1	_randvar

Advanced Design System 2008

interp2	_rn
interp3	_shift_reg
interp4	_si
iss	_si_bb
issue_message_set_value	_si_d
itob	_si_e
iusb	_sopt
jn	_sv
ktoc	_sv_bb
ktof	_sv_d
	_sv_e
	_tn
	_to
	_tt
	_uniform
	_uniform_tol
	_xcross

ADS Reserved Words - Alphabetical List

A		C
All_Params All_params AllParams Allparams		CostIndex
D		L
DF_DefaultInt DF_Value DF_ZERO_OHMS DefaultValue DeviceIndex		LinearizedElementIndex
N		P
Nsample		PrintErrorMessage
S		W
ScheduleCycle		WarnTimeDomainDeembed
a		b
abs access_all_data access_all_data_new access_data access_data_new acos	amp_harm_coef and arcsinh arctan asin asinh	bin bitseq boltzmann by

Advanced Design System 2008

acosh aele all_Params all_params allParams allparams	atan atan2 atanh attenuator_warn awg_dia		
c		d	
c0 ceil check_indep_limits coef_count complex compute_mp_poly_coef compute_poly_coef conj	cos cos_pulse cosh cot coth cpx_gain_poly ctof ctok cxform	d_atan2 damped_sin db dbm dbmtoa dbmtov dbmtow dbpolar dbwtow dcSourceLevel deembed define	deg delay dep_data deriv discrete distcompname doe doeindex doeIter dphase dsexpr dstoarray
e		f	
e e0 echo else elseif embedded_prolemy_exec end	endif ensure_ext equals erf_pulse eval_miso_poly eval_poly exp exp_pulse	fetch_envband file floor fmod fread	freq freq_mult_coef freq_mult_poly ftoc ftok
g		h	
gauss gaussian gcdata_to_poly generate_gmsk_iq_spectra generate_gmsk_pulse_spectra generate_pi_qpsk_spectra generate_pulse_train_spectra generate_qam16_spectra generate_qpsk_pulse_spectra	get_array_size get_attribute get_block get_fund_freq get_indep_limits get_LSfreqs get_LSpowrs get_max_points get_S2D_attribute global globalnode ground	hpvar_to_vs hugereal hypot	
i		j	
i icor		j jn	

if ilsb imag impulse imt_hpvar_to_array imt_hbdata_to_array include_SSpower inline index innerprod inoise int internal_generate_gmsk_iq_spectra internal_generate_gmsk_pulse_spectra internal_generate_pi_qpsk_spectra internal_generate_pulse_train_spectra internal_generate_qam16_spectra internal_generate_qpsk_pulse_spectra internal_get_fund_freq internal_window interp interp1 interp2 interp3 interp4 iss issue_message_set_value itob iusb			
k		l	
ktoc ktof		length lfsr limit_warn list ln ln10 local log	log10 logNodesetScale lognormal logRforce logRshunt logScale log_amp log_amp_cas lookup
m		n	
mag makearray mapping max mcTrial mcindex min	miximt_coef miximt_poly model mp_fetchS21 mp_poly_gain multi_freq multivar_access	names nested nf nfmin no nodoe noise	noisefreq noopt norm nostat not notequals notune

Advanced Design System 2008

	multivar_tree		
o		p	
omega opt optIter or		parameters phase phase_noise_pwl phasedeg phaserad pi pinCurrent pinMapping planck	polar polarcpX portz pow ppt pulse pwl pwlr pwlr_tr
q		r	
qelectron qinterp		rad ramp rawtoarray read_data read_lib read_SSpower readdata readlib readraw	real rect rem repeat ripple rms rn rpsmooth
s		t	
scale scalearray sens setDT sffm sgn sin sinc sinh sopt sourceLevel	spectrum sprintf sqrt ssfreq stat static status_print step strcat stypexform sym_set system	tan tanh temp tempkelvin thd then time	timestep tinyreal tnom to toi tranorder transform tune
u		v	
u0 unconst	uniform	v value version_check vlsb	vnoise vss vswrpolar vusb
w		y	
window wtodbm		yes	

—			
_ABM_Phase _ABM_SourceLevel _ac_state _bitwise_and _bitwise_asl _bitwise_asr _bitwise_not _bitwise_or _bitwise_xnor _bitwise_xor _dc_state _default _discrete_density _divn _fc _freq1 _freq2 _freq3 _freq4 _freq5 _freq6 _freq7 _freq8 _freq9 _freq10 _freq11 _freq12	_gaussian _gaussian_tol _get_fnom_freq _get_fund_freq_for_fdd _harm _hb_state _lfsr _M _mvgaussian _mvgaussian_cov _n_state _nfmin _p2dInputPower _phase_freq _pwl_density _pwl_distribution	_randvar _rn _shift_reg _si _si_bb _si_d _si_e _sigproc_state _sm_state _sopt _sp_state _sv _sv_bb _sv_d _sv_e	_tn _to _tr_state _tt _uniform _uniform_tol _VER _VEr _VeR _Ver _vER _vEr _veR _ver _xcross
—			
__fdd __fdd_v __randseed	__s __y __z		

Preparing a Circuit for Simulation in ADS

This topic describes a variety of items that can be added to an ADS schematic to prepare it for circuit simulation. You should be familiar with this process and with working in projects before continuing here.

The process for creating a schematic—selecting and placing components, editing component parameters, and wiring—is described in the [Schematic Capture and Layout](#) documentation and in the [Advanced Design System Quick Start](#).


Refer to the following topics for details on using simulation-specific items:

- [Using Current Probes](#) describes how to specify the points in a circuit where you can measure and save current values.
- [Naming Nodes](#) describes how to specify the circuit nodes where you can measure and save voltages.
- [Using NodeSet and NodeSetByName Components](#) describes how to apply best guess voltage and resistance values at points in a circuit to set starting DC values.
- [Highlighting Nodes](#) describes how to highlight nodes to quickly locate a point in a circuit.
- [Using Constants, Variables, and Functions](#) shows how to use variables and equations to assign values to parameters.
- [Applying Measurements](#) shows how to use pre-defined measurements in a schematic, which are evaluated during a simulation and whose results are saved to view in the Data Display.
- [Using Simulation Templates](#) shows how to use predefined circuit and simulation setups to simplify creating your design.
- [Using Simulation Instrument Components](#) shows how to simplify the simulation process by connecting your design to components that represent instruments and run a simulation.

Using Current Probes

Current probes are added to a schematic to collect current data at that point in the circuit. You can place as many probes as you want in a schematic. Current probes are found on the Probe Components palette.

Current probes have parameters that you may want to edit, but it is not necessary. You may want to rename the probe to something meaningful, since the name is used to name the data collected with the probe.

 Note
Current probes (I_Probe components) must be placed so that the arrow on the probe points in the direction of (positive) current flow. To flip a probe horizontally, choose Edit > Rotate/Mirror > Mirror About Y .

Naming Nodes

To collect voltage data at nodes of interest, you label the nodes on the schematic.

There are two types of nodes: global nodes and named nodes .

By placing a GlobalNode component on a sub-level or top-level schematic of a design (Insert > Global Node), you can select and edit the name of a node that will maintain the same identity throughout the entire hierarchy of designs. This means the nodes with the same name as the global node name in other designs are all electrically connected. This facilitates the interconnection of boards, IC chips, and connectors.


A named node can be applied to any schematic, but it is specific to that schematic only.

To specify a global node:


1. Choose Insert > Global Node and place the component on the schematic. Double-click to display the dialog box for entering a name.
2. Type a name in the Enter global node name field. Click Add . On this design and lower-level designs, it will be considered the same node.
3. To make an existing named node a global node, select it from Node Name List and click Add.

To name a node:

1. Choose Insert > Wire/Pin Label .
2. In the dialog box appears, type the desired name and click the node on the schematic that you want to associate with that name.

 **Note**
In general, voltage and current data is in phasor representation, so the voltage values at named nodes are peak voltage.

3. You can repeat this for other nodes, or click Done to dismiss the dialog box.

 **Note**
By placing an exclamation mark (!) at the end of node name, it becomes a global node for compatibility with Cadence formats. This should be used only if Cadence compatibility is required.

Using NodeSet and NodeSetByName Components

The following sections provide details on the components NodeSet and NodeSetByName, which are available in all simulation palettes.

Using NodeSet or NodeSetByName to Facilitate a Simulation

By placing a NodeSet or NodeSetByName component at strategic places in a circuit, you can instruct the DC simulator to begin its analysis at a given best-guess voltage. It is also possible to enter values for connection resistance.

These node set components can be used in any analysis, but are especially useful for:

- Circuits that are bi-stable, such as flip-flops or ring oscillators, to force it to a known high or low state rather than letting the DC solver find the meta-stable state halfway between high and low.

- Circuits that are isolated from DC by blocking capacitors.

NodeSet and NodeSetByName work in a two-stage process. In the first stage, these elements attach the specified voltage source with a series resistor to the specified node(s) to force a value. A DC solution for the entire circuit is then calculated. In the second stage, the forcing source and resistor are removed and the DC solution is refined, using the previous DC solution as an initial guess.

If you choose to use a NodeSetByName component, you can specify a name to facilitate the retrieval of voltage data in the dataset.

Note that if you specify both a DC Initial Guess File and a nodeset component, the former takes precedence.

NodeSet Fields

Following are details on the fields in the dialog box for the NodeSet component.

Instance Name Displays and edits the name of the component.

Select Parameter Selects a voltage or resistance for editing. V (volts) is an estimated initial node voltage. R is connection resistance.

Add adds a voltage or resistance to the Select Parameter field.

Cut deletes a voltage or resistance from the Select Parameter field.

Paste copies a voltage or resistance that has been cut and places it in the Select Parameter field.

Parameter Entry Mode Select standard or file-based data.

Optimization/Statistics/DOE Setup Opens a dialog box providing for the entry of parameters related to optimization and statistics.

Display parameter on schematic Displays or hides a selected node on the schematic.

NodeSetByName Fields

Advanced Design System 2008

Following are details on the fields in the dialog box for the NodeSetByName component.

Instance Name Displays and edits the name of the component.

Select Parameter Selects a node name for editing. This name is associated with an initial voltage V and a connection resistance R.

Add adds a node name from the Edit Node Name field to the Select Parameter field.

Cut deletes a node name from the Select Parameter field.

Paste copies a node name that has been cut and places it in the Select Parameter field.

Select a Node Name: Selects a node name for editing, or for adding to the Select Parameter field.

Node Name List: Type in a node name.

Volt The initial voltage guess associated with the node name. Use this field to edit the voltage.

Res The connection resistance associated with the node name. Use this field to edit the resistance.

Display parameter on schematic Displays or hides a selected node name on the schematic.

Nodes

Highlighting Nodes

Highlighting nodes can help you identify specific points in a schematic or subnetwork. To do this, choose **Simulate > Highlight Node** . This opens a window that lists all nodes (such as named connections, wires, pins, and ports) in a circuit and in all of its subcircuits. Click a node in the list and it will be highlighted on the schematic.

Highlighting nodes can help in troubleshooting a simulation problem. If problems are encountered at a node during simulation, the error and node name will appear in the Simulation/Synthesis Messages window. By using the highlight node feature you can quickly zoom in on the problem area.

Clearing Highlights

There are two ways to clear all highlights:

- In the Highlight Node window, click Clear . This clears all highlights that have been set.
- In the Schematic window, choose View > Clear Highlighting.



Hint

The highlight color can be changed through Options > Preferences > Display > Highlight .

Using Constants, Variables, and Functions

Advanced Design System contains built-in global constants, variables, and functions that can be used in a schematic. You can use them:

- With the VarEqn component
- With components whose parameters can be defined using equations. (For a selected parameter, the Equation Editor button will appear in the component editing dialog box.)

These can simplify schematic design. For example, you can set a variable named Frequency to a specific value, then use the variable wherever the frequency needs to be specified in the schematic. If you want to change the frequency, you do so in one place.

For more information on how to use VarEqn, refer to the VarEqn component help.



Note

You can use the conditional statement if/then/else/endif in variable definitions and component equations. Be sure to include the endif .

Many of the projects in the Examples directory use variables. One example that includes many variable definitions plus conditional statements is NADC_PA.dsn_ in RF_Board/NADC_PA_prj .

Lists of constants, variables, and functions are next.

Pre-Defined Constants

The pre-defined built-in constants available for use in an equation are:

Constant	Value	Description
e	2.718 282 ...	e
ln10	2.302 585 ...	ln(10)

Advanced Design System 2008

c0	2.997 924 58 e+08 m/s	speed of light
e0	8.854 188 ... e-12 F/m	vacuum permittivity (1/(u0*c0*c0))
u0	1.256 637 ... e-06 H/m	vacuum permeability (4*pi*1e-7)
boltzmann	1.380 658 e-23 J/K	Boltzmann's constant
qelectron	1.602 177 33 e-19 C	charge of an electron
planck	6.626 075 5 e-34 J*s	Planck's constant
pi	3.141 593 ...	pi

Pre-Defined Variables

The pre-defined, built-in variables for use in an equation are:

Variable	Default Value	Description
time	0 s	analysis time
timestep	1 s	analysis time step
freq	1 e+006 Hz	analysis frequency for linear and multi-tone simulations such as Harmonic Balance and Circuit Envelope
temp	25 C	analysis temperature; set by Options Temp
tnom	25 C	default nominal temperature for models; set by Options Tnom
_freq1 through _freq12	1 e+0006 Hz	fundamental frequencies defined for multi-tone simulations such as Harmonic Balance and Circuit Envelope

Pre-Defined Functions

Function arguments have the following designations.

Complex	Real	Strings
x, y	r, r0, r1, rx, ry, lower_bound, upper_bound	s, s1, s2

Advanced Design System 2008

In general, the functions return a complex number, unless it is a string operator as noted. A function that returns a real value effectively has a zero value imaginary term.

Function	Description
cos(x)	cosine function, x is in radians
cot(x)	cotangent function, x is in radians
conj(x)	complex-conjugate function
cosh(x)	hyperbolic cosine function
coth(x)	hyperbolic cotangent function
exp(x)	exponential function
imag(x)	imaginary-part function
log(x)	log base 10 function
ln(x)	natural log function
mag(x)	magnitude function
phase(x)	phase (in degrees) function
phasedeg(x)	phase (in degrees) function
phaserad(x)	phase (in radians) function
real(x)	real-part function
sin(x)	sine function, x is in radians
sinh(x)	hyperbolic sine function
sqrt(x)	square root function
tan(x)	tangent function, x is in radians
tanh(x)	hyperbolic tangent function
abs(rx)	absolute value function
arcsinh(rx)	arcsinh function
arctan(rx)	arctan function, returns radians
atan2(rx, ry)	arctangent function (two real arguments), returns radians
complex(rx, ry)	real-to-complex conversion function
db(rx)	decibel function, $20 \log_{10}(x)$
dbpolar(rx, ry)	(dB,angle)-to-rectangular conversion function, rx=mag in dB, ry=angle, degrees

dbmtow(rx)	convert dBm to watts
deg(rx)	radian-to-degree conversion function
int(rx)	convert-to-integer function
jn(r0, r1)	bessel function
max(rx, ry)	maximum function
min(rx, ry)	minimum function
polar(rx, ry)	polar-to-rectangular conversion function, rx=magnitude, ry=angle, degrees
rad(rx)	degree-to-radian conversion function
sgn(rx)	signum function
sinc(rx)	sin(x)/x function
sprintf(...)	formatted print utility; returns a string
<p>example: x = 2 y = 14 z = sprintf("%i.%i", x, y) results in the string "2.14" sprintf follows standard C programming syntax</p>	
strcat(...)	string concatenation utility; returns a string
<p>example: s1 = "my cat" s2 = " is frisky" s3 = strcat(s1, s2) results in the string "my cat is frisky"</p>	

The Effect of Expressions on Units

Due to the way that the simulator processes expressions, the following expression is considered valid by the ADS simulator: $F = 1.0 \text{ M M}$. This value is interpreted by the simulator as: $F = 1.0 * 1.0e6 * 1.0e6$. This situation can occur when a variable is defined with units and the variable is then used as a component parameter that also has a units field. Although valid, such an expression usually does not specify the intended value.

The behavior of the Edit Component Parameters dialog is designed so that a parameter value, initially specified as a number followed by a scale factor, is changed to a non-numeric value, and the scale factor setting is automatically set to None . This scale factor setting can be changed manually, if desired.

Applying Measurements

Measurements are pre-defined expressions that make it easy to make common calculations such as VSWR or signal-to-noise ratio. Measurements are available from the simulation palettes and have two purposes:

- They can be used on the schematic, in conjunction with simulations, to process the results of a simulation.
- They can be used in Data Display equations to process the results of a simulation and display various relationships graphically.

To create your own measurement, use the MeasEqn component. For details about measurements, refer to [Simulator Expressions](#) and [Measurement Expressions](#) documentation.

To add a measurement to a schematic:

- Select a measurement from the simulation palette and place it on the schematic.
 - You can modify the measurement to customize it or change the name. Click the Help button in the dialog box for details about the measurement.
 - You can select the measurement for output for a specific analysis. This has the effect of restricting evaluation to that analysis only (if more exist), as well as saving the result after each analysis iteration (e.g. each time point, or frequency point), instead of after all iterations, thus using less memory for intermediary data.

To view the results after running the simulation:

1. Open a Data Display window and select a plot and place it in the window.
2. The name of the measurement will appear in the list of variables. Select it to add it to the plot and click OK.

Measurements can also be used in Data Display equations to perform additional processing after a simulation. For information on how to use measurements in Data Display equations, see [Data Display](#).

Quantities Measurements Can Reference

Measurements can reference:

- Any simulation outputs (voltages, currents, S-parameters) from the current circuit level and levels below using full hierarchical names (refer to [Simulation Output Names](#)).
- Other measurements and variable equations. Measurement equations and variable equations follow the same nested scoping rules: measurement equations can reference other measurement and variable equations at the current or higher levels. Note that measurement and variable equations cannot share the same name (see [Naming Conventions](#)).
- Existing data in datasets produced by previous simulations or imported via the Data File Tool. The full circuit path of the saved simulation output is always required. Using the same syntax used for data displays to reference an

Advanced Design System 2008

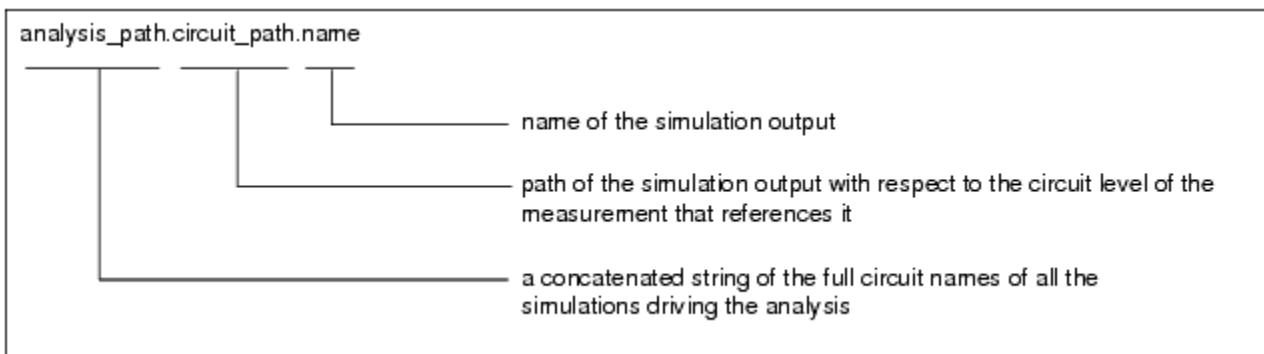
existing dataset entry, preface the measurement name with the dataset name.

Example

MeasEqn1 = Vout	accesses node Vout in the current circuit
MeasEqn2 = saved_dataset.DC1.DC.Vout	accesses node Vout , generated by analysis DC1 in the dataset saved_dataset.ds

Simulation Output Names

To successfully use measurement equations, you must understand the full names associated with simulation outputs. Each simulation output has a unique name. A measurement may refer to such an output by using its unique name, or a condensed version of it. The full unique name of a simulation output is described in the following illustration.



where:

- analysis_path is a concatenated string of the full circuit names of all the simulations driving the analysis. For example, a single top-level DC analysis called DC1 on the design results in the analysis path DC1.DC . If a top-level sweep analysis called Sweep1 drives that DC analysis, then the circuit path is Sweep1.DC1.DC. The .DC suffix is specific to the DC analysis. Major suffixes are as follows:

DC	.DC
AC	.AC
AC noise	.NC
Harmonic Balance, P2D, XDB, Envelope, LSSP	.HB
Harmonic Balance noise	.HB_NOISE
Transient	.TRAN

- circuit_path is the path of the simulation output (node voltage, current, etc.) with respect to the circuit level of the measurement that references it. For example:

MeasEqn1 = Vout

may reference a node voltage Vout at the current level (hence no path), while

MeasEqn2 = X1.Vout

may reference a node voltage Vout in the subcircuit X1 of the current level (hence the circuit path is X1.)

Unlike node voltages and currents, S-, Y-, Z-parameters and the corresponding delays require no circuit path.

- name is the name of the simulation output (e.g. Vout for a node voltage, _I_Probe1.i_ for the current through current probe _I_Probe1_ , S for scattering parameters).

The circuit path and the name are required for proper reference. The analysis path is optional, and may be used in the case where a design contains multiple analyses to differentiate between same-name outputs. The analysis path need not be complete. For example, the node voltage Vout generated through the DC analysis DC1 may be referenced by a same-level measurement as follows:

MeasEqn1 = Vout

MeasEqn2 = DC.Vout

MeasEqn3 = DC1.DC.Vout

but not MeasEqn4 = DC1.Vout

The same resolution rules used for a data display apply here to analysis outputs.

Using Simulation Templates

A number of templates are available to facilitate setting up common simulations. Copy these to a directory where you have write permission.

To use a simulation template:

1. Choose Insert > Template .
2. From the dialog box that appears, select the desired simulation type and click OK . Place the template in the Schematic window and modify it as required.

Simulation Template Descriptions

Template	Description
----------	-------------

Advanced Design System 2008

BJT_curve_tracer	This simulation uses a swept current source for the base current and a swept voltage source for the collector voltage, to simulate the DC collector current versus collector-emitter voltage curves of a BJT.
ConvPulseRespT	This simulation uses nonlinear, time-domain analysis to simulate the pulse response of a network. The pulse response can be the reflection from a network or transmission line or the transmission of the signal through the network or transmission line. Also, coupling of the pulse signal from one line to another can be simulated. If the circuit contains distributed elements, then convolution will be used during the simulation. The reflected and transmitted signals may be shown. Refer to the example file: examples/RF_Board/TDRcrosstalk_prj to see this template in use.
ConvStepRespT	This simulation uses nonlinear, time-domain analysis to simulate the step response of a network. The step response can be the reflection from a network or transmission line or the transmission of the signal through the network or transmission line. Also, coupling of the step signal from one line to another can be simulated. If the circuit contains distributed elements, then convolution will be used during the simulation. The reflected and transmitted signals may be shown. Refer to the example file: examples/RF_Board/TDRcrosstalk_prj to see this template in use.
DC_BJT_T	This generates the same I-V curves as the BJT_curve_tracer, except that the sources and simulation controllers are packaged up into a subcircuit.
DC_FET_T	This generates the same I-V curves as the FET_curve_tracer, except that the sources and simulation controllers are packaged up into a subcircuit.
FET_curve_tracer	This uses swept voltage sources for the gate and drain voltages, to simulate the DC drain current versus drain-source voltage curves of a FET.
HB1Tone	This simulation generates the output power, power gain, harmonic distortion, and the output spectrum, when the test signal is a sinusoid at one power and frequency.
HB1ToneSwptFreq	This simulation generates the frequency-dependent output power, power gain, harmonic distortion, and the output spectrum, when the test signal is a sinusoid at one power and is swept over frequency.
HB1ToneSwptPwr	This simulation generates the output power, power gain,

Advanced Design System 2008

	<p>harmonic distortion, output spectrum, and gain compression, when the test signal is a swept-power sinusoid at one frequency.</p>
HB2Tone	<p>This simulation generates the output power, power gain, output spectrum, and third- and fifth-order intermodulation distortion points (input- and output-referred) when the test signals are two sinusoids of the same power.</p>
HB2ToneSwptPwr	<p>This simulation generates the output power, power gain, output spectrum, and third- and fifth-order intermodulation distortion points (input- and output-referred), as well as the intermodulation distortion levels when the test signals are two sinusoids and their power is swept.</p>
LinearPulseRespT	<p>This simulation uses linear, swept-frequency AC analysis to simulate the time-domain pulse response of a network. The pulse response can be the reflection from a network or transmission line or the transmission of the signal through the network or transmission line. Also, coupling of the pulse signal from one line to another can be simulated. The reflected and transmitted signals may be shown.</p> <p>Refer to the example file: examples/RF_Board/TDRcrosstalk_prj to see this template in use.</p>
LinearStepRespT	<p>This simulation uses linear, swept-frequency AC analysis to simulate the time-domain step response of a network. The step response can be the reflection from a network or transmission line or the transmission of the signal through the network or transmission line. Also, coupling of the step signal from one line to another can be simulated. The reflected and transmitted signals may be shown.</p> <p>Refer to the example file: examples/RF_Board/TDRcrosstalk_prj to see this template in use.</p>
MixConvGainNF	<p>This simulates the conversion gain and noise figure of a mixer.</p>
MixTOI	<p>This simulates the output spectrum, output power, conversion gain, and third-order intercept points of a mixer.</p>
S_Params	<p>This simulates the S-parameters of any two-port network, and generates Smith chart plots for S11 and S22, and polar plots for S21 and S12. The Smith charts include a circle of constant VSWR, whose value you may</p>

Advanced Design System 2008

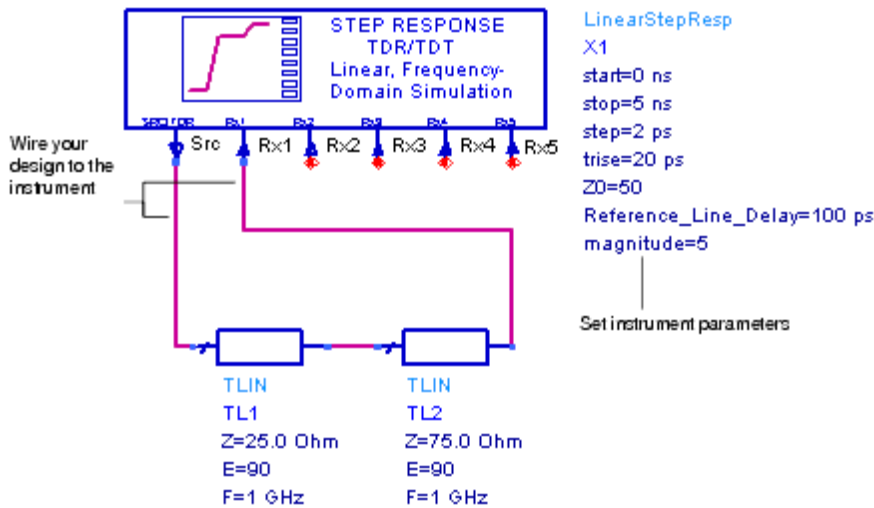
	set.
S_Params_DC	This simulates the S-parameters of any two-port network, and generates Smith chart plots for S11 and S22, and polar plots for S21 and S12. The Smith charts include a circle of constant VSWR, whose value you may set. In addition, it generates zoomed plots of S11 and S21, over a reduced frequency range. A DC simulation is also run.
SP_BJT_T	This is a two-port vector network analyzer equivalent with biasing for a BJT. It sweeps the base current and collector-emitter voltage, and simulates the S-parameters of the device at each bias point, at one analysis frequency.
SP_DiffT	This simulates the S-parameters of any two-port network, but the test ports are ungrounded. This allows the simulation of differential-mode S-parameters.
SP_FET_T	This is a two-port vector network analyzer equivalent with biasing for a FET. It sweeps the gate-source and drain-source voltages, and simulates the S-parameters of the device at each bias point, at one analysis frequency.
SP_NWA_4PortBiasLogT	This simulates the S-parameters of any four-port network, but plots the data assuming you want to compare two sets of two-port S-parameters. It generates Smith chart plots for S11 and S33, and S22 and S44, and polar plots for S21 and S43, and S34 and S12. The Smith chart plots show circles of constant VSWR. It is identical to SP_NWA_4PortT, except that a log frequency sweep is used, and a single DC bias may be set for each of the test ports.
SP_NWA_4PortBiasT	This simulates the S-parameters of any four-port network, but plots the data assuming you want to compare two sets of two-port S-parameters. It generates Smith chart plots for S11 and S33, and S22 and S44, and polar plots for S21 and S43, and S34 and S12. The Smith chart plots show circles of constant VSWR. It is identical to SP_NWA_4PortT, except that a single DC bias may be set for each of the test ports.
SP_NWA_4PortLogT	This simulates the S-parameters of any four-port network, but plots the data assuming you want to compare two sets of two-port S-parameters. It generates Smith chart plots for S11 and S33, and S22 and S44, and polar plots for S21 and S43, and S34 and S12. The Smith chart plots show circles of constant VSWR. It is identical to SP_NWA_4PortT, except that a log

	frequency sweep is used.
SP_NWA_4PortT	This simulates the S-parameters of any four-port network, but plots the data assuming you want to compare two sets of two-port S-parameters. It generates Smith chart plots for S11 and S33, and S22 and S44, and polar plots for S21 and S43, and S34 and S12. The Smith chart plots show circles of constant VSWR.
SP_NWA_LogT	This simulates the S-parameters of any two-port network, and generates Smith chart plots for S11 and S22, and rectangular plots for dB(S21) and dB(S12). It is identical to the SP_NWA_T template, except that a log frequency sweep is used.
SP_NWA_T	This simulates the S-parameters of any two-port network, and generates Smith chart plots for S11 and S22, and rectangular plots for dB(S21) and dB(S12). A two-port vector network analyzer equivalent instrument is used. In addition, it generates zoomed plots of S11 and S21, over a reduced frequency range. Also, plots showing available gain and stability circles may be created.
Sparams_wNoise	This simulates the S-parameters and noise figure of any two-port network, and generates Smith chart plots for S11 and S22, and rectangular plots for dB(S21) and dB(S12). The Smith charts include a circle of constant VSWR, whose value you may set. In addition, it generates zoomed plots of S11 and S21, over a reduced frequency range. Also, plots showing available gain, noise figure, and stability circles are created.
S_ParamsLargeSignal	This simulates the S-parameters of any two-port network, as a function of frequency, and input signal power. The S-parameters are computed as the ratios of the incident and reflected waves at the fundamental frequency. The Rollett stability factor, K and the group delay are also computed from the S-parameters. This template is particularly useful for computing the output reflection coefficient of a device when it is being driven by a large input signal. Note that this template does not use a LSSP simulation controller. Instead it uses harmonic balance combined with small-signal mixer mode.

Using Simulation Instrument Components

Simulation instrument components provide a method for symbolically connecting your circuit to an instrument. You connect your design to components that represent various instruments and run the simulation.

The instruments are set up as curve tracers, TDRs, and network analyzers. There are two or more of each type of instrument—each one is designed for a particular simulation or measurement. They are located on the Component Palette, under Simulation-Instrument. For details on each component, see [Simulation Instruments](#).



To use a simulation instrument:

1. Create your design.
2. From the Component Palette, choose Simulation-Instrument . Select the appropriate instrument and place it on your schematic.
3. Connect the ports of your design to the instrument connectors.
4. Set the instrument parameters.
5. Run the simulation.


S-Parameter Test Labs and Sequencer

S-parameter test labs and Sequencer both enable you to take multiple simulations and combine them into one simulation run. An S-parameter test lab enables you to calculate the S-parameters of multiple N-port networks in a single simulation run. A `Sequencer` controller enables you to sequence multiple simulations into a single simulation run.

An S-parameter test lab is a schematic that contains one S-parameter test lab component and one or more test

benches. A test bench is a schematic that contains an N-port network and terminations for each port of the network. In multiple stage circuit design practices, the designer is interested in viewing the inter-stage circuit behavior of all stages simultaneously. In particular, it is desired for each stage to be terminated not in 50 ohms, but in the applicable input/output impedances of adjacent stages. See [RefNets](#) for more information on using RefNets in conjunction with the S-parameter test lab feature.

There are many reasons why you may want to combine test benches into sequence, using a `Sequencer` controller. These include optimizing a variable across multiple simulations, enabling complex instrument control in Ptolemy and running a series of verifications tests on a design. To sequence these simulations, you will need to create a test bench that includes all the desired simulation controllers and the top-level design file.

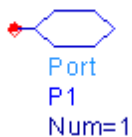
 **Note**
For information on how to create a Test Bench refer to [Creating a Test Bench](#).

Comparison of Test Lab and Sequencer

Sequencer	Test Lab
DC, SP, AC, HB, Tran, ENV, Ptolemy	SP only
Utilizes Test Bench Controllers	Utilizes Test Lab Controller
Different temps per test bench possible	One simulation temp for all
Opt/Stat/ParamSwp at top level	
RefNets supported	

Creating a Test Bench

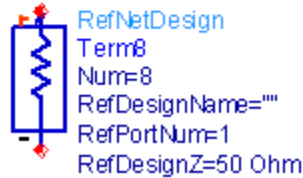
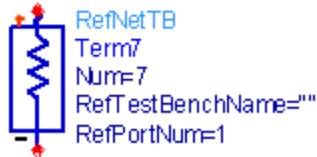
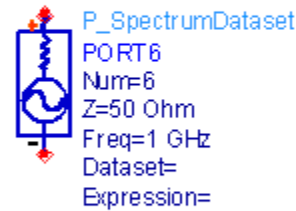
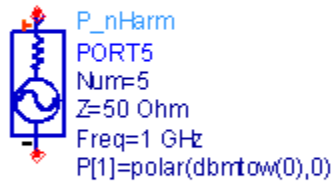
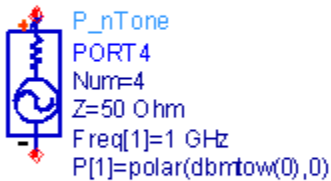
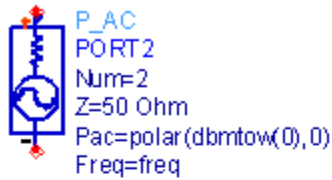
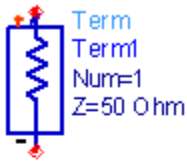
Test Bench	A test bench is simply a top-level design file that a user can run a simulation from. Since the test bench is a top-level design, it should contain no <code>Port</code> components. For simulations that are going to be Sequenced, a test bench must contain a simulation controller. For S-parameter test labs, a simulation controller is optional.
------------	---



Port component

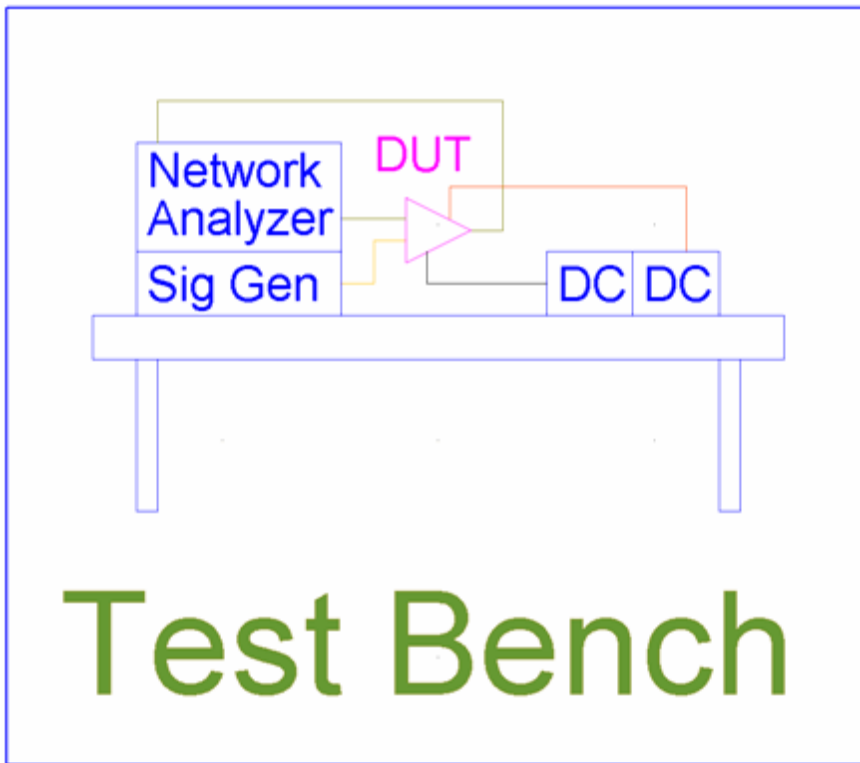
To create a test bench you should declare your schematic (to be tested) as a parametric subnetwork. As with any parameter subnetwork, you can add parameters and set the library location. To learn how to create a parameter subnetwork see [Creating a Parametric Subnetwork](#).

Unlike parametric subnetworks, a test bench schematic should not have any `Port` components. Instead, it should be terminated with proper terminations, power sources with built-in resistors, or `RefNet` component.

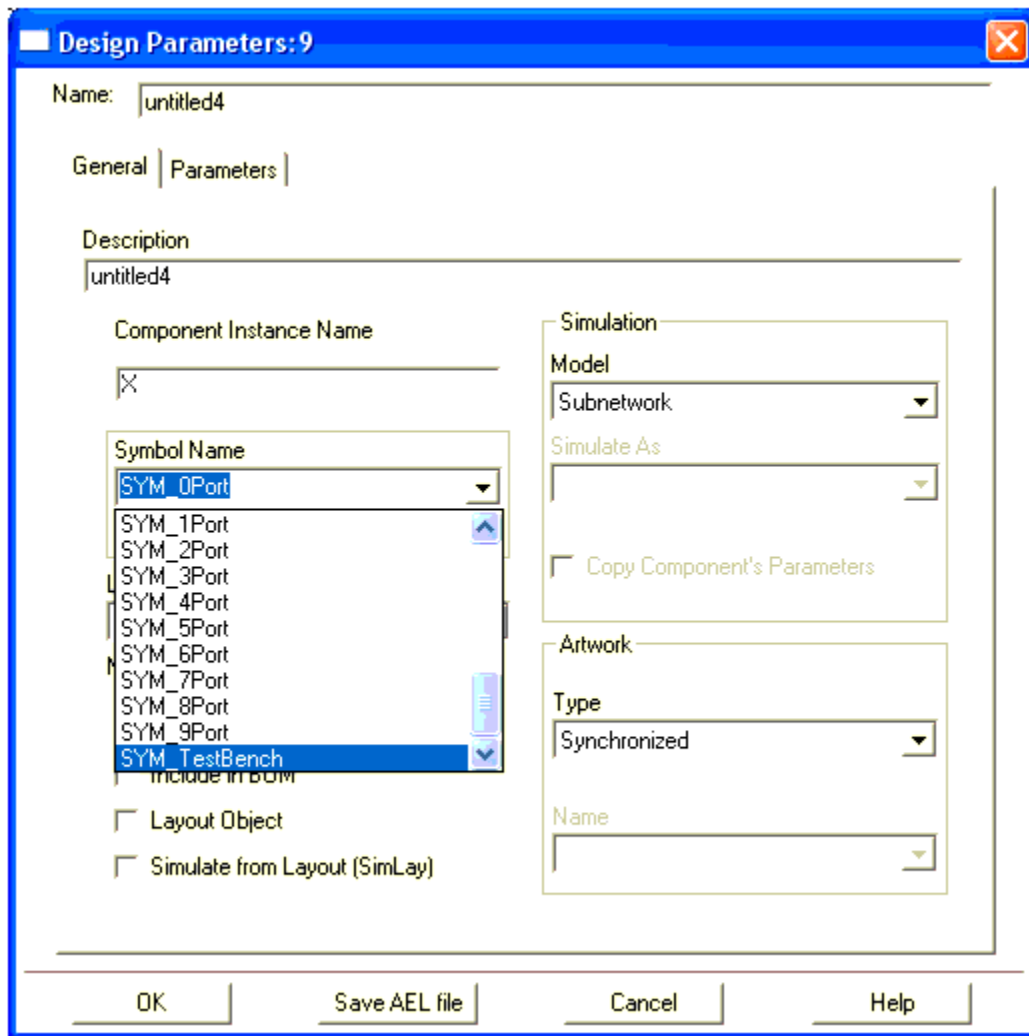


Supported terminal components for S-parameter test lab

Although optional, you should set your symbol for the new subnetwork to be `SYM_TestBench`:



To set this symbol, select File > Design Parameters... and change the symbol name to `SYM_TestBench` as shown in the following figure:



S-Parameter Test Labs

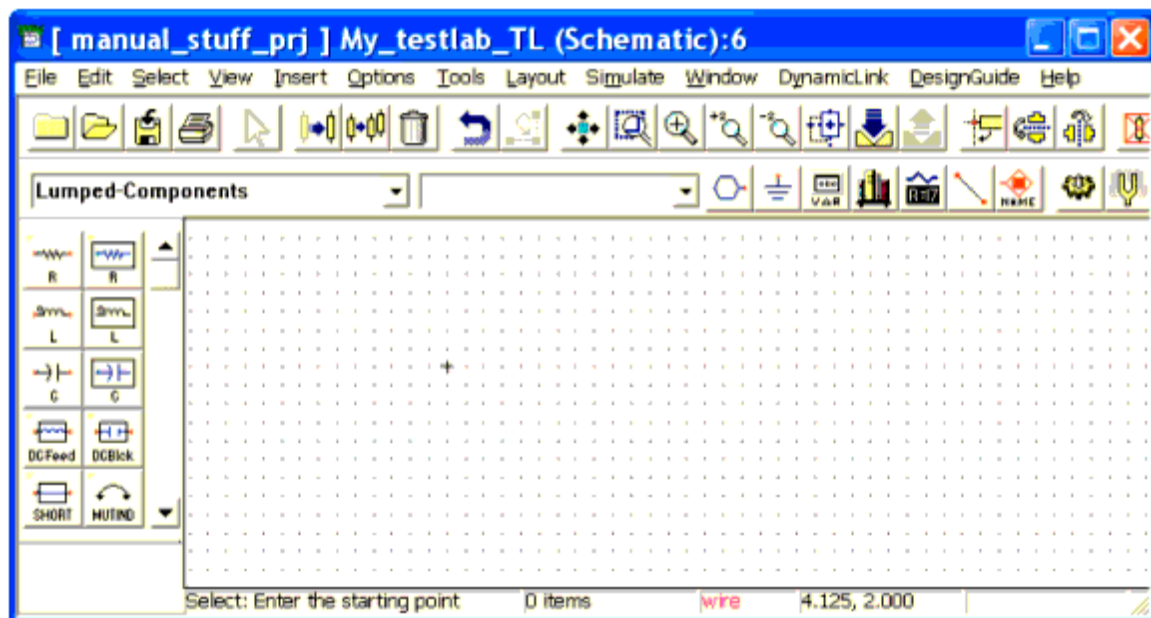
Test Lab Usage Rules

1. As its name implies, the S-parameter test lab is dedicated to S-parameter simulation. As such, a nonlinear design will be linearized about its DC operating point.
2. The S-parameter test lab simulation will ignore any simulation controllers contained in a test bench. It is still useful to have test bench controllers because they can be used to perform a stand-alone simulation of the test bench.
3. The S-parameter test lab should not contain any circuit components other than test benches. Any connectivity (wires) in the S-parameter test lab is ignored. Any component in the S-parameter test lab that has one or more pins is ignored.

4. The minimum requirements for an S-parameter test lab to function are:
 - At least one test bench
 - One `S_ParamTestLab` Simulation Controller
5. By convention, S-parameter test lab names should end in `_TL` . This is not required.
6. Only one S-parameter test lab simulation controller is allowed. A `SweepPlan` can be used to specify multiple continuous or discontinuous frequency combs.
7. S-parameter test labs support the following auxiliary simulation controllers: Options Plan, Sweep Plan, Parameter Sweep, Optimization, Statistical, and DOE controllers.
8. Variable equations (simulator expressions) and measurement expressions are supported. As with any top-level design, variables defined at the top are recognized throughout the hierarchy.
9. Tuning is supported in the S-parameter test lab. Tuning within an S-parameter test lab works identically to tuning for a normal top-level design. Users can push into a test bench and select parameters or variables to tune. Users can also tune variables and item parameters in the schematic window that displays the S-parameter test lab.
10. Global nodes are supported, but they are not global across test benches. They are global within each test bench.
11. Any global expression found in a test bench is available to all other test benches.
12. The `S_ParamTestLab` controller has virtually the same user interface and displayed parameters as the standard S-Parameter controller.

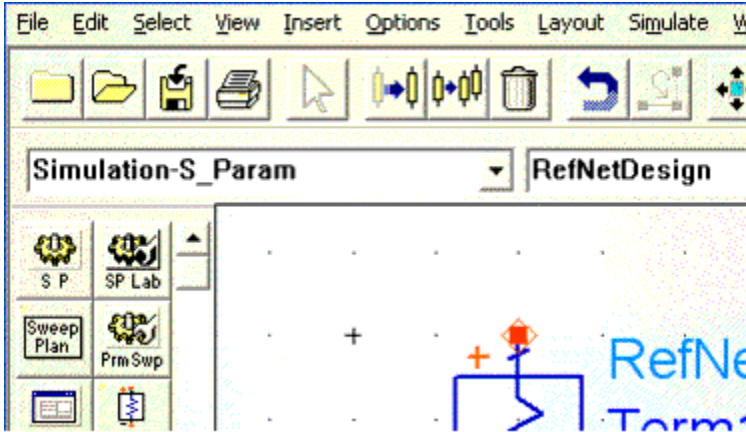
Configuring an S-Parameter Test Lab

1. Determine the design files for which you want to calculate the S-parameters. You will create a test bench for each of these designs.
2. Create the S-Parameter Test Lab.
 - Create a new schematic. This will be the test lab. It is recommended (but not required) that the name of the test lab schematic end in `_TL` . For this example, the name `My_testlab_TL` is used.



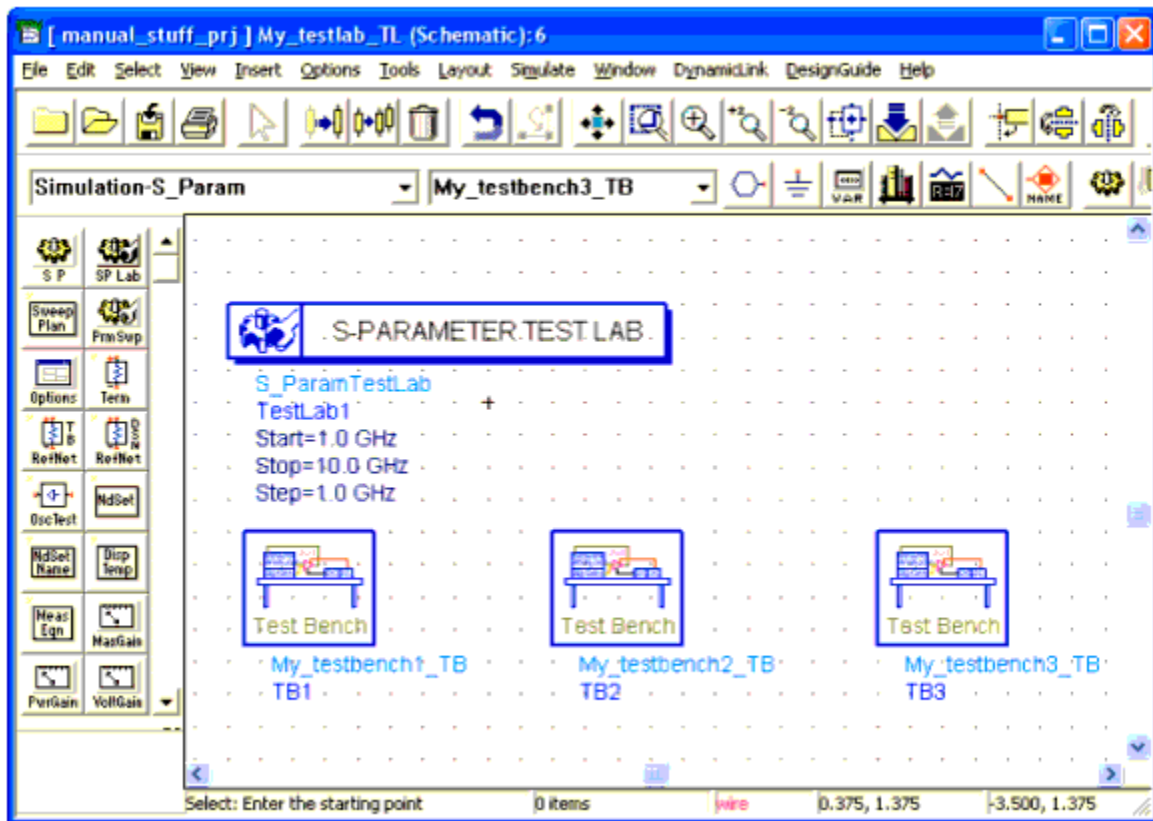
Schematic window with design named using $_TL$ S-parameter test lab naming convention

- In the S-parameter test lab schematic, change to the Simulation-S_Param palette. Place an S_ParamTestLab controller (icon appears as SP Lab). Configure the S-parameter test lab controller the same way you would configure a standard S-parameter analysis.



Test Lab S-parameter controller icon, SP Lab.

- In the S-parameter test lab schematic, place the test bench, My_testbench1_TB , created earlier. For illustration purposes, assume that test benches My_testbench2_TB and My_testbench3_TB were also created. These are placed into the completed test lab.



Completed S-parameter test lab

Notes

1. Both the Sequencer and S-parameter test lab calculate measurement equations that are contained in a test bench. The measurement equation is calculated only for the test bench that contains it. The measurement equation should not use the test bench instance id to refer to data (for example, S-parameters) that is generated for the test bench.
2. If only one test bench is placed, the testbenchID prefix may be omitted. However, this is not recommended.

Data Display Naming Convention for Simulation Results

Standard Results Data

Standard S-parameter simulation output with Noise turned on produces the following standard output to the dataset:

```
S, S(i,j)
PortZ, PortZ(1), PortZ(2), PortZ(n), freq
Icor, Icor(i,j)
nf, nf(i), Nfmin
Rn, Sopt, te, te(n)
```

Where $i = 1,2,\dots$ and $j = 1,2,\dots$ are port indices.

An S-parameter test lab controller will also calculate these items. However, the test bench Instance ID will prefix the names. For example, an S-parameter test lab containing two, two-port test benches will produce the following results:

```
X1.S(1,1)
X1.S(2,1)
X1.S(1,2)
X1.S(2,2)
X2.S(1,1)
X2.S(2,1)
X2.S(1,2)
X2.S(2,2)
```

where X1 and X2 are the test bench instance ID names appearing in the test lab. If only one test bench appears in the S-parameter test lab, then the test bench prefix is not required.

Measurement Equation Results Data

Measurement equations appearing in the S-parameter test lab appear in the dataset as

```
MeasurementEquationName1  
MeasurementEquationName2
```

Measurement equations appearing in test benches in an S-parameter test lab will also appear in the dataset as

```
MeasurementEquationName1  
MeasurementEquationName2  
...
```

However, if the same measurement equation name appears in the S-parameter test lab and a participating test bench, the following nomenclature is used

```
_Testlab1_TL_sp.MeasurementEquationName1  
TestBench1_TB.MeasurementEquationName1
```

Optimization and Statistical Analysis

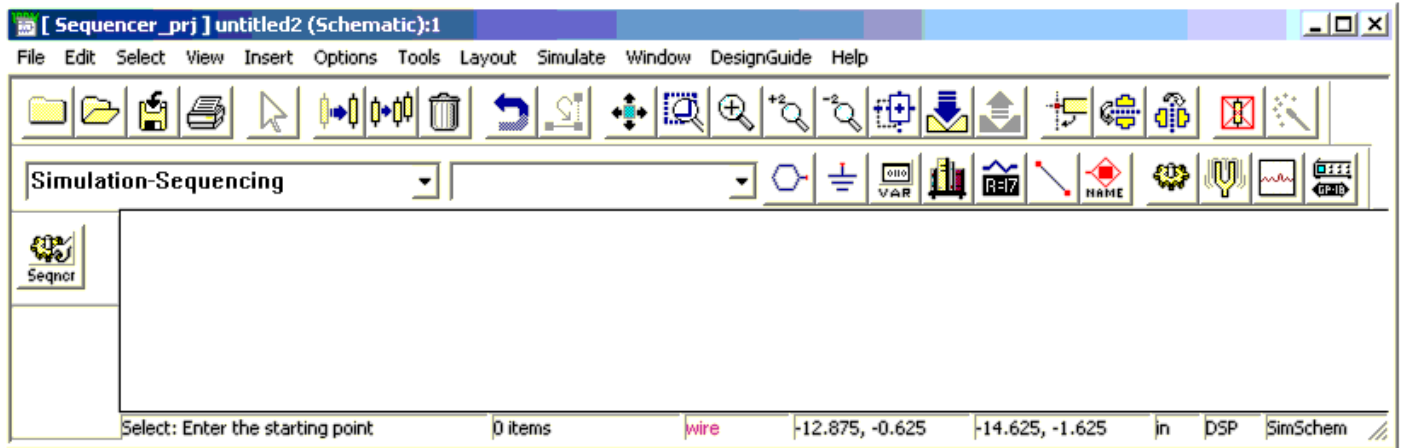
Configuring optimization and statistical analyses in an S-parameter test lab is similar to configuring them for a standard S-parameter analysis. An example project, `TestLab_HOWTO_prj`, is available in the ADS examples directory, `$HPPEESOF_DIR/examples/Tutorial`.

This example illustrates optimization in an S-parameter test lab. The reader is advised to review the example's `Readme.dsn` file, which contains detailed information.

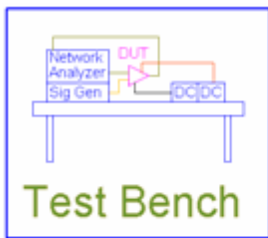
Sequencer

Creating a Sequence

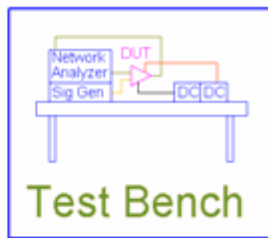
Create a test bench, then you are ready to specify the sequence. To do so, you'll need to create a new top-level design and instantiate each test bench in it. Next, you'll need to add a `Sequencer` controller. This controller is available on the Simulation-Sequencing bitmap palette.



Once you have instantiated the test benches and `Sequencer` controller, your top-level design will look something like:



SineWaveGeneration
TB1

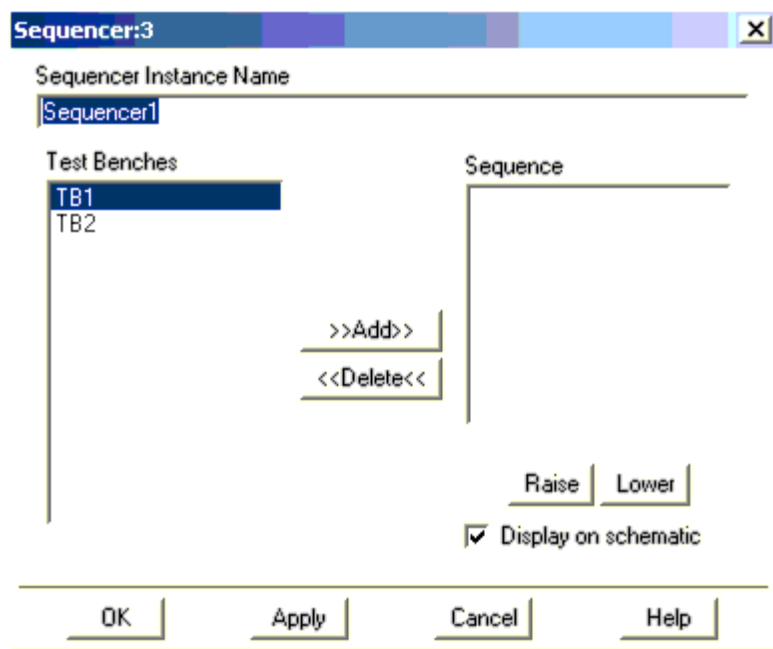


SignalSweep
TB2



Sequencer
Sequencer1
TestBench[1]=""

You are now ready to set the sequence, edit the `Sequencer` parameters and add each test bench in the desired order. When you first bring up the edit parameter box, you'll see the available test benches in the left pane.



Add each test bench by either clicking >>Add>> or double-clicking the desired test bench in the order you wish to run the test benches. This will move the selected test bench from the available list to the sequence list in the right pane. To reorder the sequence, use Raise and Lower located below the right pane.

Now your setup is complete and you can run the simulation.

Note
See [Using Measurement Equations with a Test Lab or Sequencer](#) for information on using Measurement Equations with a Sequencer.

Notes

1. The top-level design should not contain any components other than test benches. Any connectivity (wires) are ignored. Any subnetwork in the top-level design that has one or more pins is ignored.
2. Only one `sequencer` controller is allowed.
3. In addition to the `sequencer` controller you can use auxiliary simulation controllers: Options Plan, Sweep Plan, Parameter Sweep, Optimization, Statistical, and DOE controllers.
4. Variable equations (simulator expressions) and measurement expressions are supported. As with any top-level design, variables defined at the top are recognized throughout the hierarchy.
5. Tuning is supported.
6. Global nodes are supported, but they are not global across test benches. They are global within each test bench.
7. Any global expression found in a test bench is available to all other test benches.

Examples

For reference, a simple Ptolemy sequencer example is included in ADS.

See `/examples/Tutorial/Sequencer_prj` documentation for more details.

Additionally, BER connected solutions examples are also available on Microsoft Windows.

See the following examples documentation located in the ADS Examples Documentation:

- 3GPP Uplink BER Receiver Characteristics Test
- WLAN 802.11a Receiver Input Level Sensitivity Test
- Simple Ptolemy Sequencer

Usage Rules

1. DSP cosimulation with A/RF is not supported.
2. Test benches cannot refer to data saved in a dataset from a previous test bench. You can work around this limitation on the DSP schematic. See the simple example above for more details.
3. A test bench cannot contain a Sequencer or S-parameter test lab controller.

Using Measurement Equations with a Test Lab or Sequencer

In either S-parameter Test Lab or Sequencer simulations, measurement equations are used in the same manner as they are in a standard analysis. However, because of the hierarchy associated with an S-parameter configuration, you need to specify the test bench instance ID when a measurement equation at the top-level refers to data generated for a test bench. The format is,

```
MeasEqnName = TestBenchInstanceID.Sij
```

where $i=1,2,\dots$ and $j=1,2,\dots$ are port indices.

The examples below illustrate this concept:

Example 1: Expressing S11 from test benches X1 and X2

```
S11_testbenchX1 = X1.S11
```

```
S11_testbenchX2 = X2.S11
```

Example 2: Utilizing S21 from test benches TB1 and TB2

```
S21_add = TB1.S21 + TB2.S21
```

```
S21_divide = TB1.S21/TB2.S21
```

Example 3: Taking stability function for test benches TB1 and TB2

```
Stabfact_TB1=stab_fact(TB1.S)
```

```
Stabfact_TB2=stab_fact(TB2.S)
```

Improving Test Lab Simulation Efficiency

Significant time performance gains in test lab simulation can be achieved by minimizing the number of different test benches in the ADS test lab. Using the same test bench repeatedly can result in faster test lab simulation. To take advantage of this, you must have a situation requiring different versions of the same circuit.

To set up a test lab for improved simulation efficiency:

1. In the test circuit, place variables on the component parameters that define the different states.
2. In the test circuit, create passed parameters using the variables created in step 1. Passed parameters are created by choosing File > Design Parameters and selecting the Parameter Tab .
3. Create a test bench containing the test circuit you created previously.
4. In the test bench, create a set of passed parameters. Create one passed parameter in the test bench for each passed parameter created in the test circuit.
5. Create a test lab. Make multiple placements of the test bench. With each placement, assign the passed parameters such that each passed parameter set defines the state of each circuit to be tested.

Doing this, you are effectively placing the same test bench multiple times and each placement defines a different circuit state by the passed parameters used.

RefNets

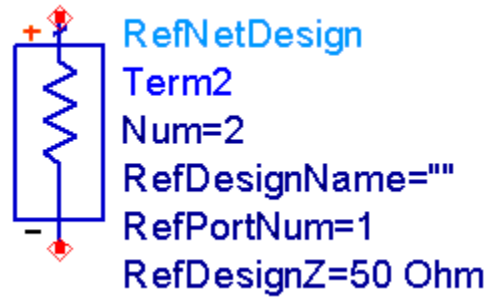
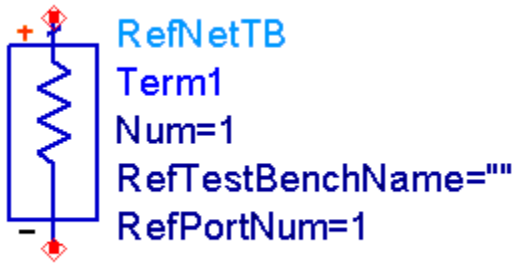
RefNet is short for "reference network". A RefNet is a component that is placed in a design that enables the port impedance from another design file in the system (the referenced network) to be referenced as a terminating impedance for the current design file under test. There are two typical applications for RefNets:

1. Inter-stage circuit analysis and design: In some design applications it is desirable to simultaneously evaluate the performance of individual circuit stages terminated in the input and output impedances of adjacent stages. For example, in transistor matching problem, the transistor in the S-parameter test lab can be terminated in the output impedance of the input matching network and the input impedance of output matching network. Further, it is desired that the matching networks be terminated looking into the appropriate side of the transistor. Simulation of these networks simultaneously is accomplished with the S-parameter test lab, see [S-Parameter Test Labs and Sequencer](#) for more information. To accomplish the termination of an individual stage referenced to a specific port of other stages in the design chain, the RefNet is utilized in the S-parameter test lab.
2. Design specific termination: For some top level DC, AC, or S-Parameter design files, it may be desired to terminate a port whose impedance is characterized by data, from an external file (e.g. S-parameters, Z-parameters, Y-parameters) or some other network.

There are two RefNet components that are available: `RefNetTB` and `RefNetDesign`. Both of these components have the same functionality and are supported under DC, AC and S-Parameter analysis, with two differences:

- `RefNetTB` supports nested network referencing while `RefNetDesign` does not. See [RefNetTB Using an S-Parameter Test Lab](#) for more information on using `RefNetTB`.
- `RefNetTB` uses a test bench as the reference design while `RefNetDesign` uses a standard (non-test-bench) schematic design. See [RefNetDesign - File Based Termination](#) for more information on using `RefNetDesign`.

Note
Nested referencing means that there is a top-level circuit under test that has one or more of its ports terminated with a `RefNetTB`. Further, the reference test bench (specified on the top-level `RefNetTB`) contains a `RefNetTB`, which again references other circuit designs in the system.



RefNetTB and RefNetDesign

The parameters of RefNetTB are as follows:

Num	The port number. This functions identically to the Num parameter found on the Term component and power source components.
RefTestBenchName	The name of the reference test bench (without the .dsn extension) that is used to calculate the reference impedance for this termination.
RefPortNum	Refers to the port number of the reference test bench. The referenced network, for example, may contain several ports. This parameter identifies the port number (the Num parameter) of a termination in the reference test bench.

The parameters of the RefNetDesign component are as follows:

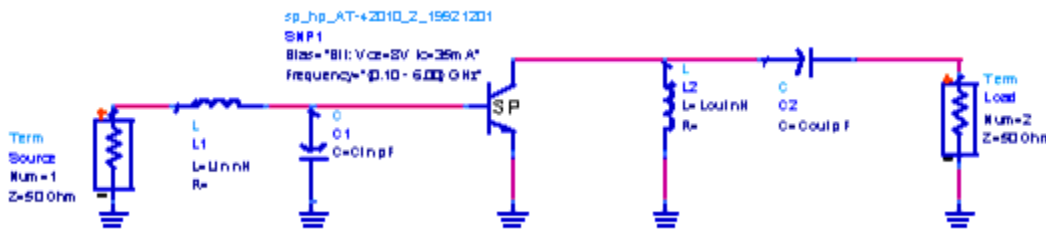
Num	The port number. This functions identically to the Num parameter found on the Term component and power source components.
RefDesignName	The name of the reference design file (without the .dsn extension) that is used to calculate the reference impedance for this termination.
RefPortNum	Parameter that identifies a port number in the reference design. The reference impedance is the impedance looking into this port of the reference design.
RefDesignZ	Describes how all other ports, if any, in the referenced design are terminated. For the case of a one-port referenced network, this parameter is not applicable and is ignored.

Notes:

1. The user cannot push into a RefNet component. If the user wants to view the reference design (test bench), this is accomplished by either toggling design files in the current schematic window or opening a new schematic window and viewing the reference design file there.
2. RefNet components do not support passed parameters.

RefNetTB Using an S-Parameter Test Lab

To best explain this procedure, a tutorial approach will be used. The following example will be considered: The narrow-band amplifier is partitioned out into three subnetworks: input.dsn, device.dsn, and output.dsn.



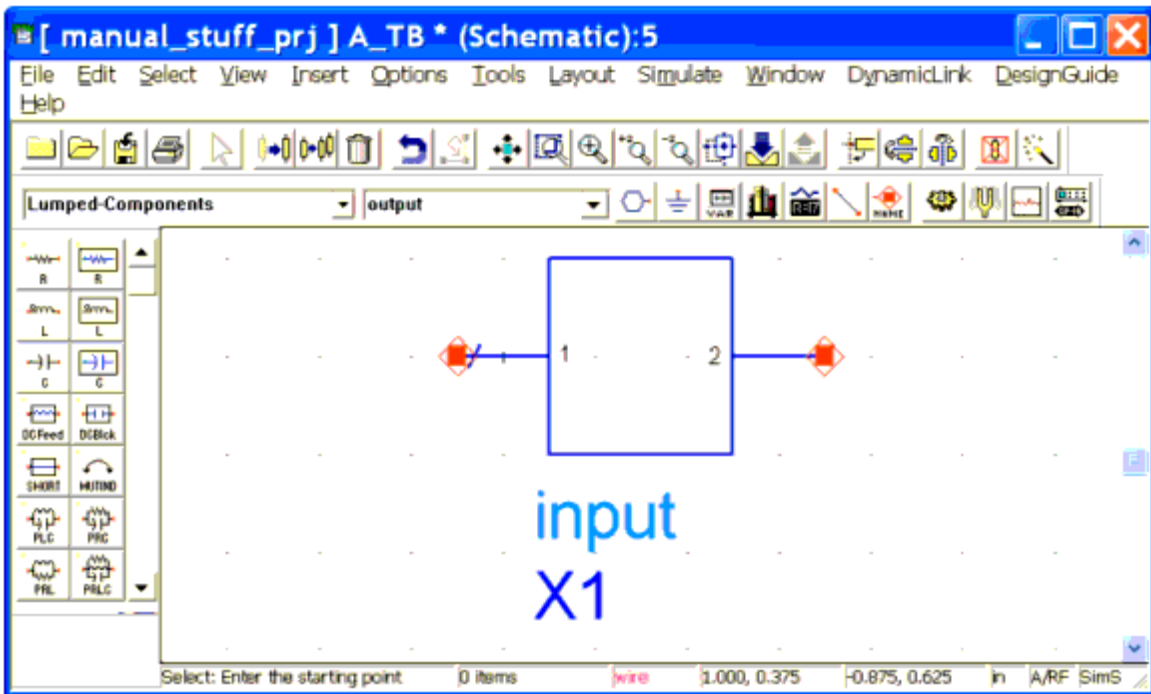
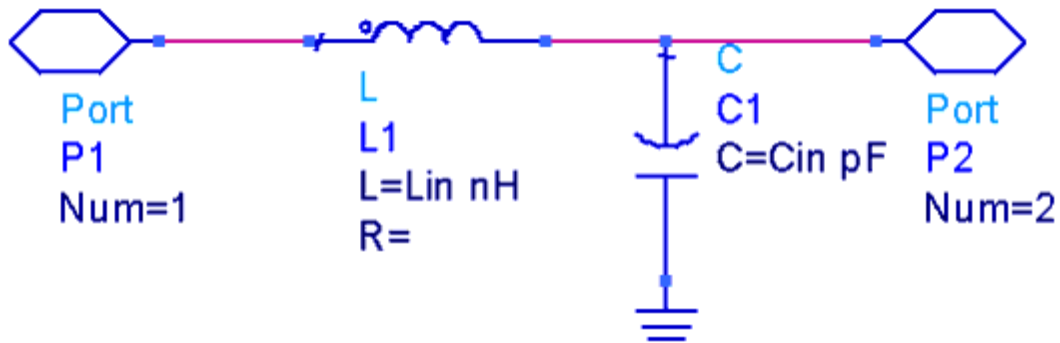
Example amplifier circuit

The objective is to place each of these sub-circuits, via test benches, into an S-parameter test lab such that the input and output ports of each subnetwork are terminated with proper terminations, power sources with built-in resistors, or RefNet component.

Note
In the following description, Port 1 refers to the network's input port, and Port 2 refers to the network's output port.

Procedure

1. Create the input subnetwork from the Example amplifier circuit shown in the previous figure. Save the design as A_TB. Set up the port terminations with:
 - Port 1 of the input network is connected to a 50 ohm source.
 - Port 2 of the input network is terminated into the chain of the device, output network, and its 50 ohm load termination.
 The input subnetwork's schematic and symbol appear in the following figure.

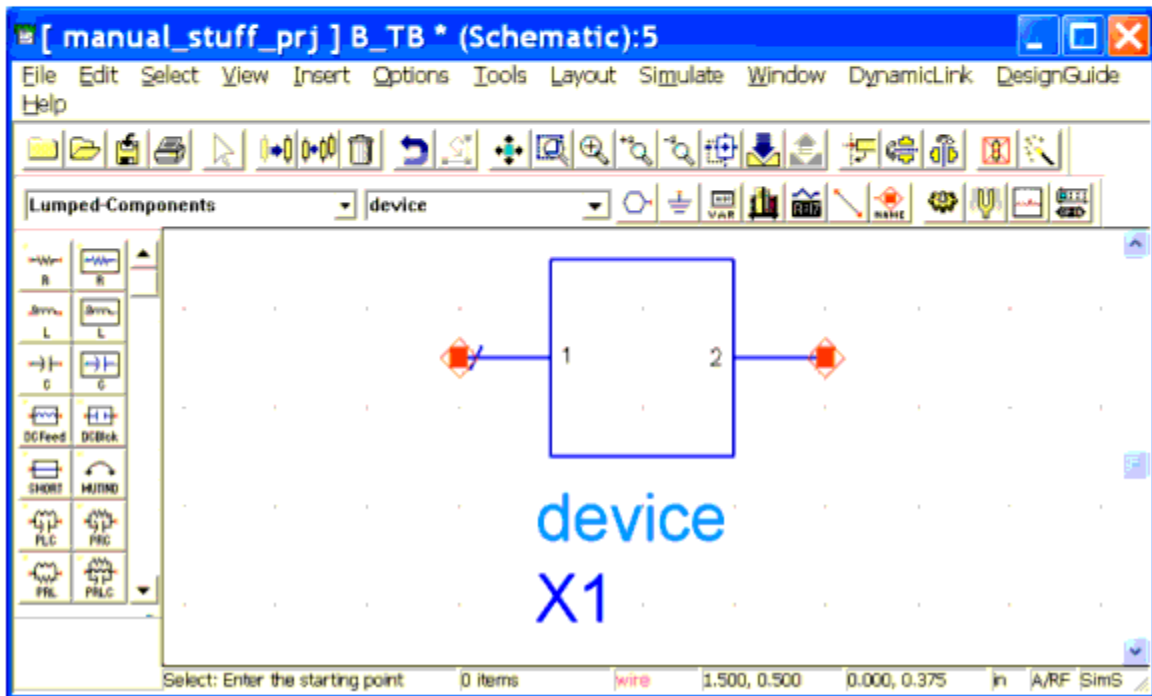
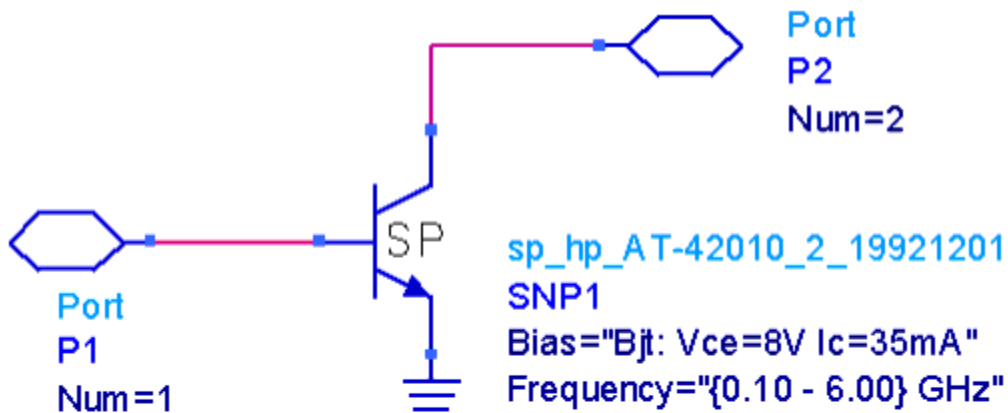


Input subnetwork

2. Create the active device subnetwork from the amplifier circuit shown in the figure above, Example amplifier circuit. Save the design as B_TB. Set up the port terminations with:

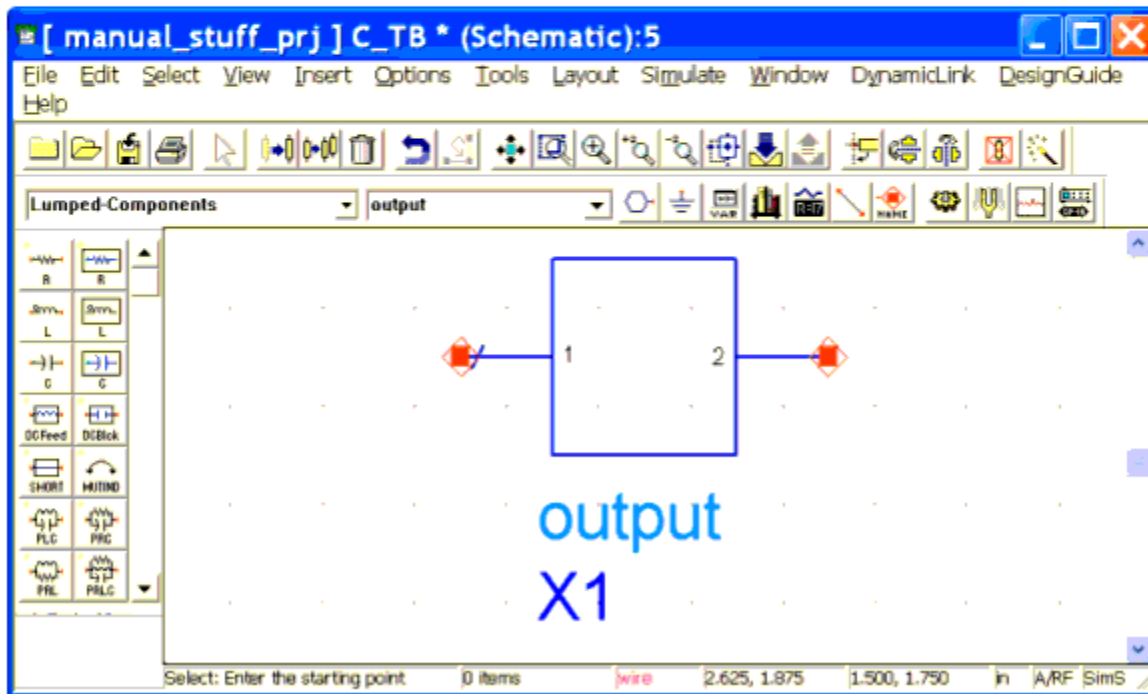
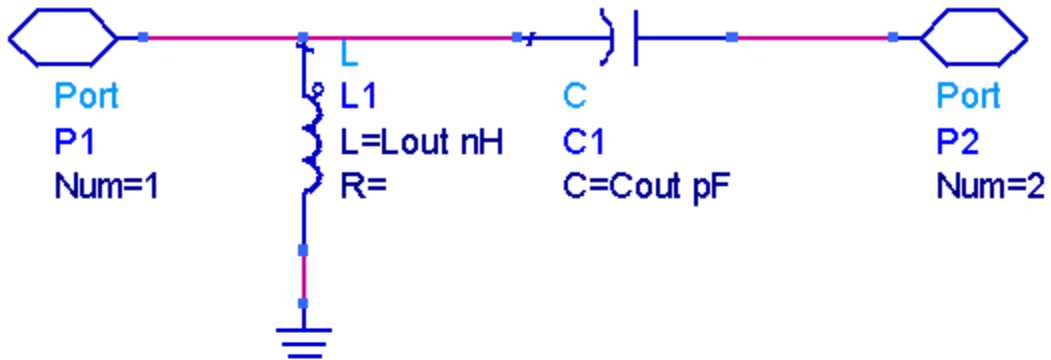
- Port 1 of the device is terminated into port 2 of the input network and its port 1 termination.
- Port 2 of the device is terminated into port 1 of the output network and its port 2 termination.

The active device subnetwork's schematic and symbol appear in the following figure.



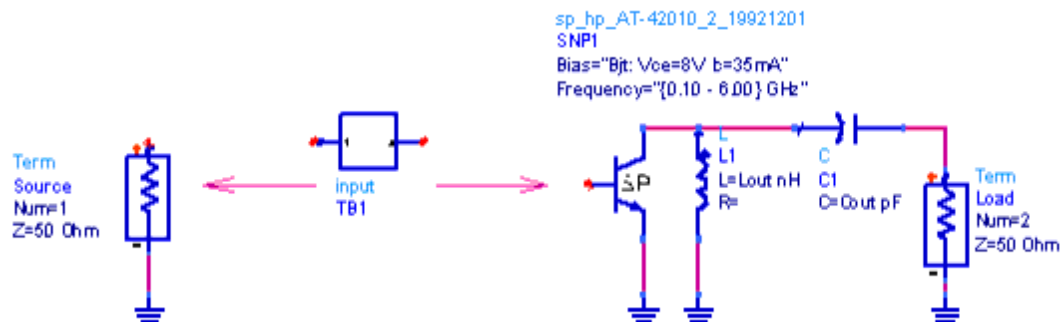
Active device subnetwork

3. Create the output subnetwork from the amplifier circuit shown in the figure above, Example amplifier circuit. Save the design as C_TB. Set up the port terminations with:
 Port 1 of the output network is terminated in the reverse chain from the device output, the input circuit, and its 50 ohm source termination.
 The output subnetwork's schematic and symbol appear in the following figure.

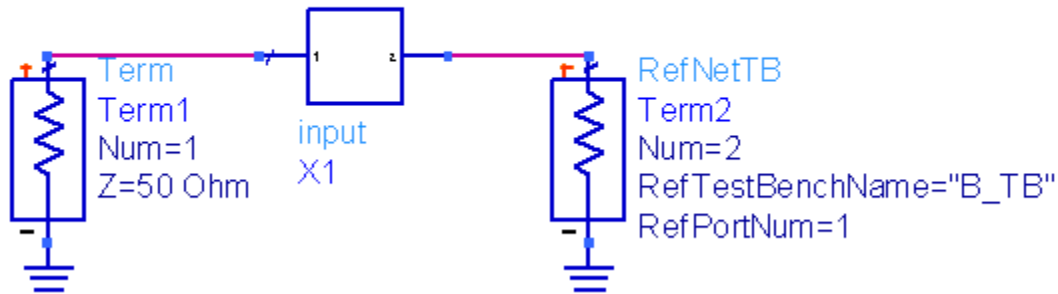


Output subnetwork

4. A Term and a RefNetTB are placed in A_TB to emulate the effective configuration.



Desired effect of test bench A_TB (input)

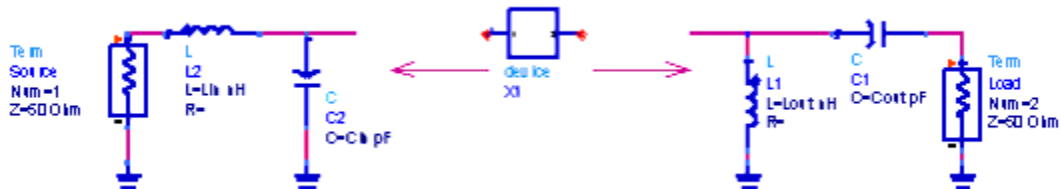


Test bench A_TB completed

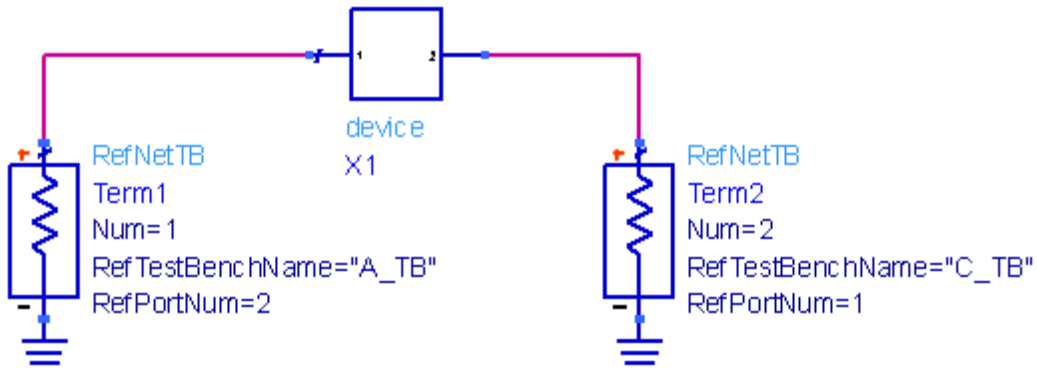
Take note of the RefNetTB parameters as follows:

Num=2	Port 2 of the test bench
RefTestBenchName="B_TB"	Points to the device test bench shown, which in forthcoming steps will be terminated with C_TB.
RefPortNum=1	The port number of B_TB where the impedance is taken.

5. Two RefNetTB components are placed in B_TB (device test bench) to emulate the effective configuration.



Desired effect of test bench B_TB (input) with Term and RefNetTB placed



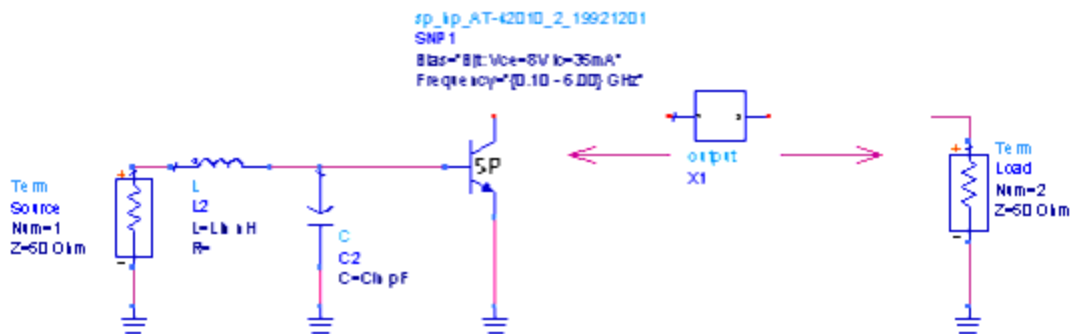
Test bench B_TB completed

Take note of the RefNetTB s placed as follows:

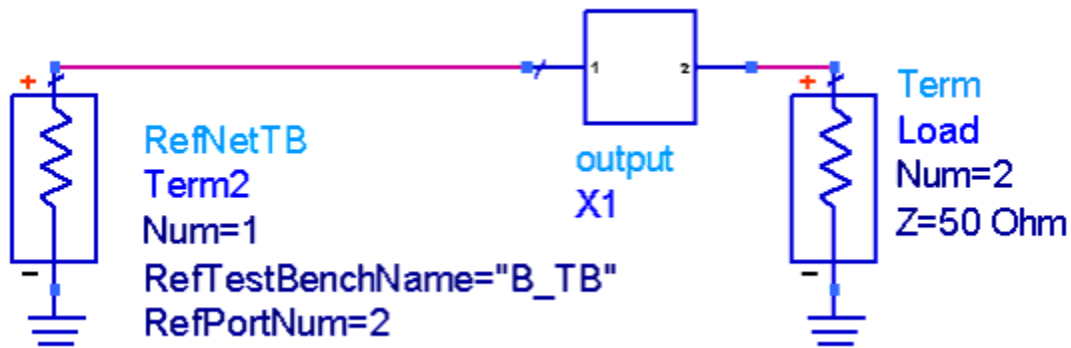
Term1 Parameters	
Num=1	Port 1 of the test bench
RefTestBenchName="A_TB"	Points to the input test bench.
RefPortNum=2	The port number of A_TB where the impedance is taken.

Term2 Parameters	
Num=2	Port 2 of the test bench
RefTestBenchName="C_TB"	Points to the output test bench.
RefPortNum=1	Port number of C_TB where the impedance is taken.

6. A RefNetTB and a Term are placed in C_TB (output test bench) to emulate the effective configuration.



Desired effect of test bench C_TB (output) with RefNetTB and Term placed



Test bench C_TB completed

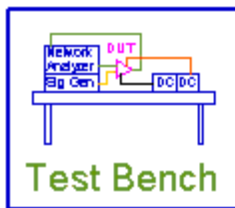
Take note of the RefNetTB parameters as follows:

Num=1	Port 1 of the test bench
RefTestBenchName="B_TB"	Points to the device test bench shown in the figure above, Test bench B_TB completed which in is nested to A_TB via the test bench shown in the figure above, Test bench A_TB completed.
RefPortNum=2	This is the port number of B_TB where the impedance is taken.

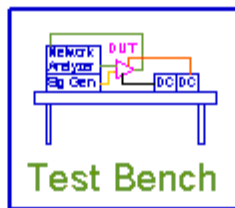
7. With test benches A_TB, B_TB, and C_TB, finished, the S-parameter test lab is created.

S-PARAMETER TEST LAB

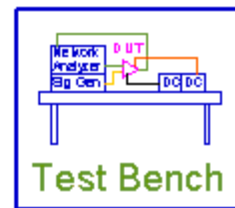
S_ParamTestLab
TestLab1
Start=1.0 GHz
Stop=10.0 GHz
Step=1.0 GHz



A_TB
TB1



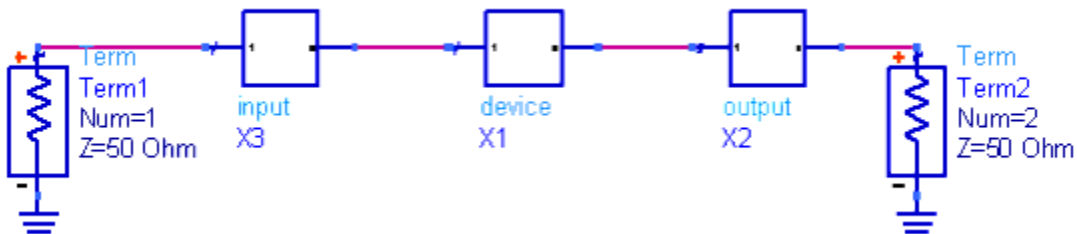
B_TB
TB2



C_TB
TB3

Completed S-parameter test lab incorporating RefNetTB

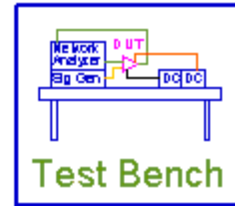
8. There is an optional step that may be desirable. The input, device, and output, can be placed into one test bench for viewing performance of the entire circuit chain.



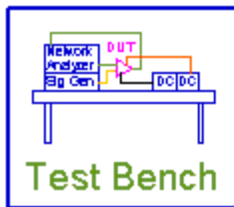
Test bench for entire circuit chain

S-PARAMETER TEST LAB

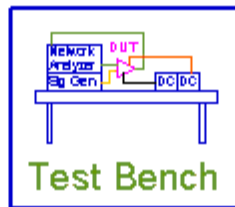
S_ParamTestLab
TestLab1
Start=1.0 GHz
Stop=10.0 GHz
Step=1.0 GHz



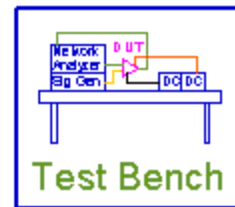
ABC_TB
X1



A_TB
TB1



B_TB
TB2



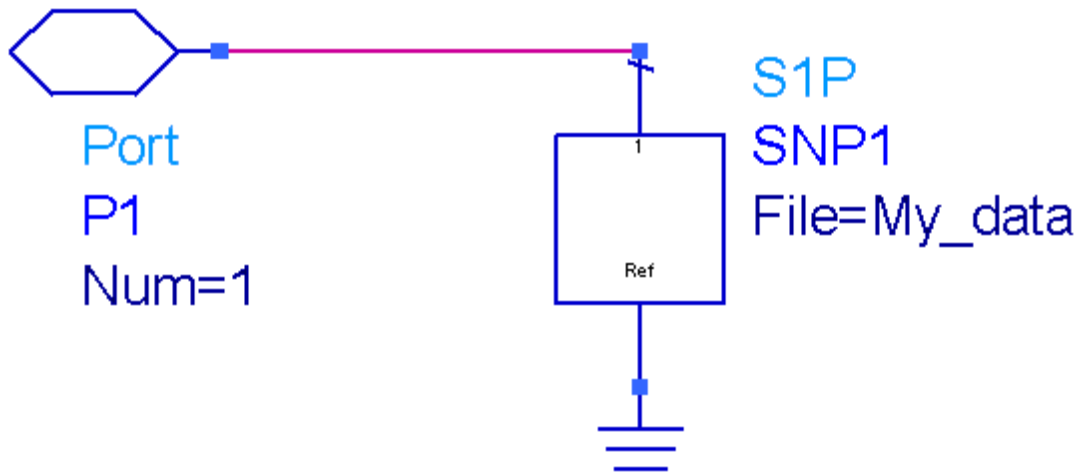
C_TB
TB3

S-parameter test lab incorporating sub-circuit and entire circuit, ABC_TB, test benches

RefNetDesign - File Based Termination

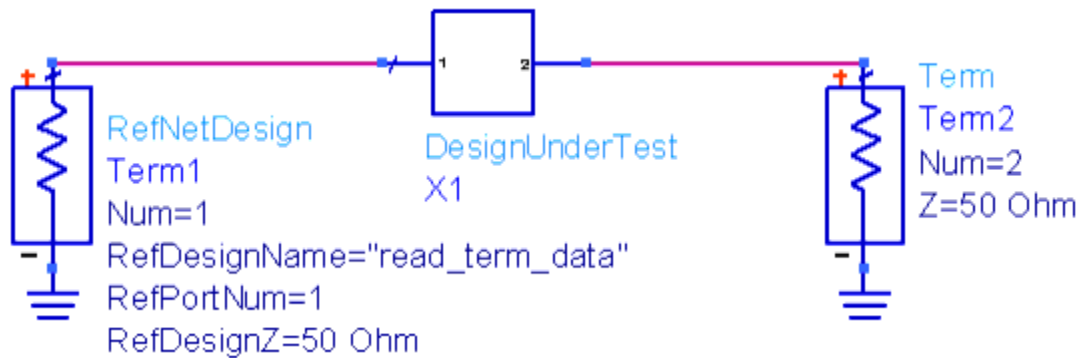
The steps to create a file-based termination are as follows:

1. Create a sub-circuit that reads in data file. For this example, a one-port s-parameter file is used.
2. Save the design file from step 1 as `read_term_data.dsn`.



Top-level one-port design file to read in S-parameter data file

3. In a design file that contains the circuit under test, place `RefNetDesign` at the pin where the S-parameter-based termination is to be applied.



Top-level design that references a file-based one-port network to realize a file-based termination

The parameters of `RefNetDesign` were assigned as follows:

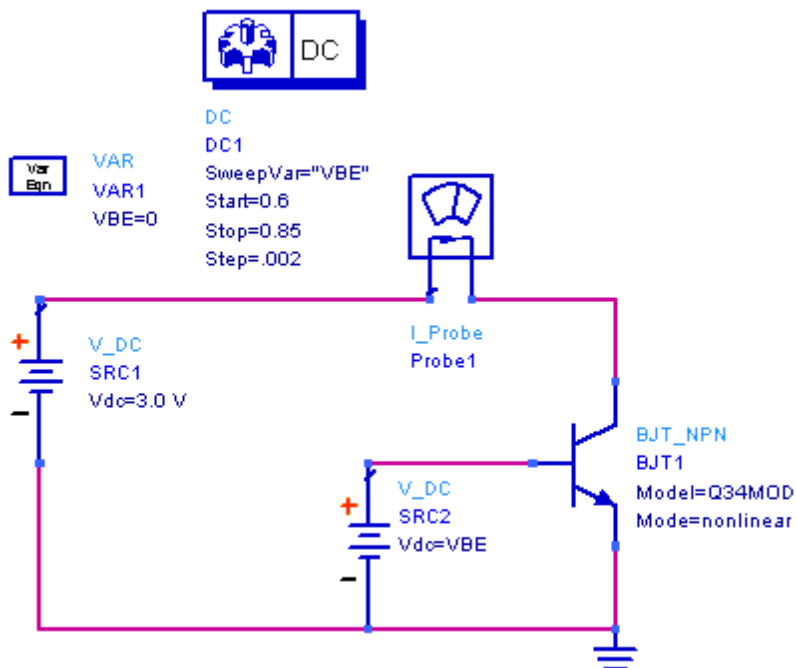
<code>Num=1</code>	Port number for the top level design.
<code>RefDesignName="read_term_data"</code>	Name of the design file for the reference network.
<code>RefPortNum=1</code>	Port number where the impedance is taken for the reference network. Since the reference network is a one-port, this is set to 1.
<code>RefDesignZ=50</code>	For this example, this parameter is not applicable. Had port components been placed in the reference design, this parameter instructs how those ports are to be terminated.

An ADS Simulation Example

This topic is a detailed simulation example for ADS. The circuit is a simple BJT and the simulation is set up to calculate the DC operating point. The simulation process is described here in detail, and this process can be applied to more complex circuits and simulations.

Before continuing with this topic, you should be familiar with the other topics in [Using Circuit Simulators](#), in [Schematic Capture and Layout](#), and in the [Advanced Design System Quick Start](#).

The DC_OP_POINT.dsn design used here is located in the Examples directory under MW_Ckts/LNA_prj . Details on working with example projects can be found in [Schematic Capture and Layout](#).



Working through this example consists of these tasks:

- [Placing Circuit Sources](#)
- [Specifying Points for Collecting Data](#)
- [Selecting a Simulation Type](#)
- [Selecting a Sweep Type and Plan](#)
- [Setting Simulation Options](#)
- [Modifying the Simulation Setup](#)
- [Starting the Simulation](#)
- [Displaying Simulation Data](#)

Placing Circuit Sources

In many cases your circuit will contain sources. In this example, you need to select a voltage source that is appropriate for a DC simulation:

1. From the Component Palette List, choose Sources-Time Domain .
2. Select V_DC (DC Voltage Source) and place this component in the Schematic window.
3. Set the DC voltage (Vdc) to 3.0 V .

Ensuring Sources are Connected Properly

You must ensure that the sources are connected properly to the circuit. You can do this two ways:


- Wire the source directly to your circuit.
- Use node names (Insert > Wire/Pin Label) to define connections. By labeling the end of the source and a point in the circuit, the source then behaves as if connected physically to that point of the circuit. For example, named connections can be used for Vcc, Vb, and Ve.

If you are wiring components, the color of the pin changes when a successful connection is established. Wires that are simply overlaid on intervening nodes will not be connected to those nodes, and clicking on the intervening nodes after the end node is wired does not ensure that the intervening nodes are connected properly.

In addition, the endpoints of dangling wires (wires that are connected at one end only) can be moved but not edited. To move an endpoint, choose Edit > Move > Move Wire Endpoint . When crosshairs appear, select the open endpoint and move it; the wire will move with it. It is also possible, using standard copy and move commands, to move and copy dangling wires and networks of wires.

To label and share a source:

1. Choose Insert > Wire/Pin Label and a dialog box appears.
2. Enter a name for the node (for example, Vcc).
3. Click the pin of the source you want to label (for example, the positive side of V_DC). The label Vcc appears at that node.
4. Next click the pin of a node to which you want Vcc to apply (for example, one end of a resistor). The resistor node is also labeled Vcc. (Note that you must click the component pin, not a wire between components.)

 **Note**
To delete a named connection, choose Edit > Wire/Pin Label > Remove Wire/Pin Label. When crosshairs appear, click the pin whose connection name you want to remove.

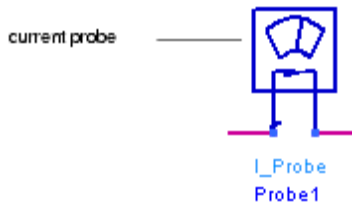
5. Edit any parameters of the source as needed.

Specifying Points for Collecting Data

You must specify the points in the circuit where voltage and current values will be collected and saved in the dataset:

- To identify points where you want voltages to be taken, use node names (Insert > Wire/Pin Label) or global nodes (Insert > Global Node). For circuit nodes that are labeled this way, voltages will be calculated during the simulation and saved to the dataset.
- For collecting current values, insert current probes (I_Probe) in your circuit at points of interest. Current probes are found on the Probe Components palette.
- Some components, such as the DC voltage source (V_DC) used in this example, include a SaveCurrent parameter. If this is set to yes , the current flowing through this component will be saved to the data set.

To help identify the data collected by a probe or source, you might want to change the default Instance Name to something more meaningful. For example, you can name a DC voltage source Vsupply. The current through that source will appear in the dataset as Vsupply.i .



- Note**
Current probes (I_Probe components) must be placed so that the arrow on the probe points in the direction of (positive) current flow. To flip a probe horizontally, choose Edit > Rotate/Mirror > Mirror About Y .
- In general, voltage and current data is in phasor representation, so the voltage values at named nodes are peak voltage.

The process for labeling nodes is described in [Ensuring Sources are Connected Properly](#).

To insert a current probe:

1. Select the Probe Components palette.
2. Select the I_Probe component and place it at the desired point in your schematic. (You may need to move components or rewire to do this.) Edit the probe component and change the Instance Name to something that suggests its purpose in the circuit.

For details on selectively sending data to the dataset, refer to [Selectively Saving and Controlling Simulation Data](#).

Selecting a Simulation Type

There are a variety of simulation methods to choose from. To use a particular simulator, you must have purchased a license for it. If you do not have a license, a message will appear when you attempt to run the simulator. This may also happen if you share licenses that are already in use.

Each simulator has its own palette, which can be selected from the Component Palette List. Details for using each simulator are described in later topics.

To add a DC simulation component (for this example):

1. Select the Simulation-DC palette.
2. Select and place the DC component on your schematic.

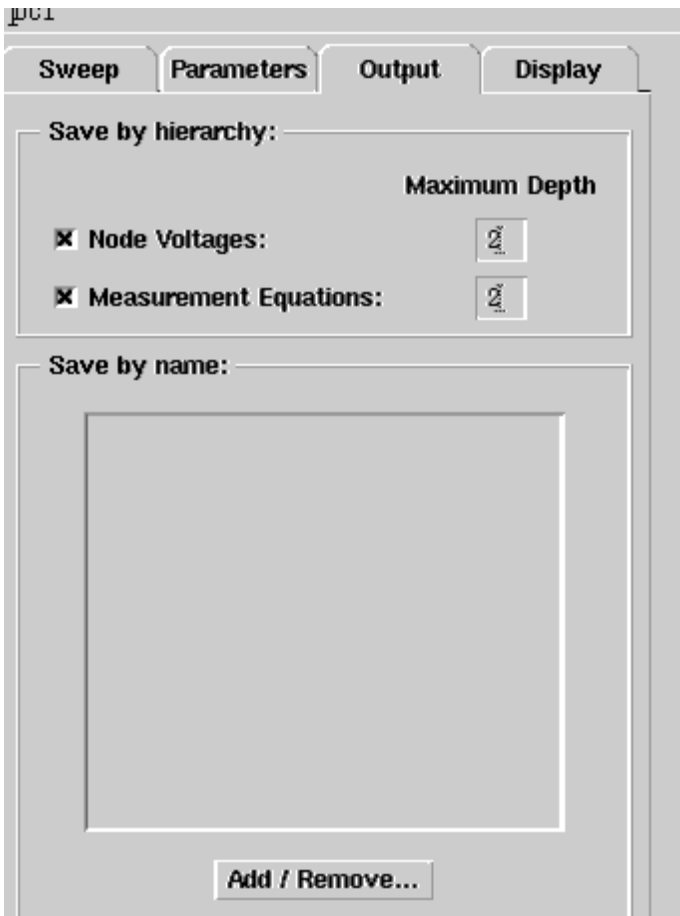
Editing Simulation Parameters

There are two ways to edit simulation parameters:

- You can edit parameters directly on the schematic. Click somewhere within the parameter value to invoke the on-screen editor, and change the value as desired. Press Return to enter data and go to the next entry. Note that by default, not all parameters are displayed on the schematic; to view/edit additional parameters, use the dialog box, as described next.
- You can edit parameters within the dialog box. To open the dialog box, either double-click the simulation component, or select it and choose Edit > Component > Edit Component Parameters . You can also bring up the dialog box using the Edit Component Parameters icon on the toolbar.



i Note
For many simulation options, the default parameters should provide satisfactory results and will not need editing.



You may find it useful to display (on the schematic) the options you have selected, to remind you of the parameters governing a simulation.

To display additional parameters on the schematic:

1. Select the Display tab.
2. Select each parameter you want to appear and click OK.

To exercise certain simulation options, you must first display them on the schematic, then set them equal to the desired value. For example, to send a defined power parameter RF_power to the dataset following a simulation, use the Output tab of the DC Operating Point dialog box. Refer to the section [Selectively Saving and Controlling Simulation Data](#).

Selecting a Sweep Type and Plan

The Sweep tab enables you to identify the parameter that you want to sweep, and then specify the sweep type. You can sweep over:

- A single point
- A linear range
- A logarithmic range

After you select the sweep type, you can then specify the sweep range.

You can also select a sweep plan . A sweep plan enables you to specify a sweep once and then use these settings in other places. For more information, refer to [Parameter Sweeps and Sweep Plans](#). Note that for some simulators, these settings are found on the Freq tab.

To set the sweep values for this example:

1. Double-click the DC component to edit it.
2. Set the following parameters as shown here:
 - Parameter to sweep = VBE
 - Start = 0.6
 - Stop = 0.85
 - Step size = .002
 - Num. of pts. = 126

If using the Load Sharing Facility (LSF) utility, you can break up a sweep and run the simulation on multiple machines, in parallel, by selecting Parallel Hosts as the Simulation Mode (Simulate > Simulation Setup). Individual sweep points are run on each machine and results combined into a single dataset on the local machine. For details on setting up remote and local machines for remote processing, see the topic Using Remote Simulation in the installation documentation for your platform.

Setting Simulation Options

An Options component can be used with any simulation. The most common use of the Options component is to set the simulation temperature, but it also enables you to specify settings for convergence tolerances, warnings and other advanced options. (For this example, an Options component is not used.)

To add an Options component:

1. Select any simulation palette.
2. Select and place a simulation Options component and double-click to edit it. You can set general simulation options such as temperature, DC convergence tolerances, warnings, and other settings.

For more information about each field, click the Help button at the bottom of the Options Component dialog box.

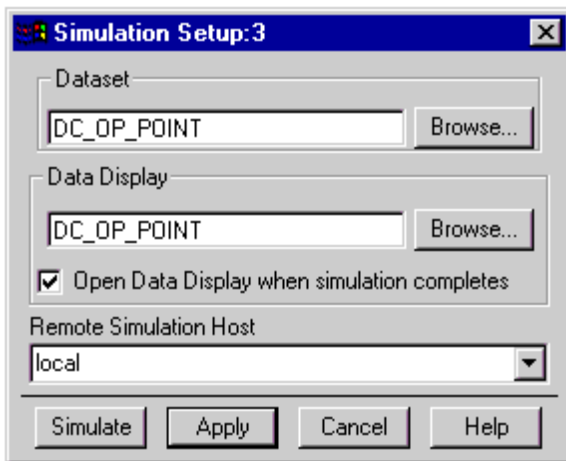
Modifying the Simulation Setup

Additional simulation setup options include:

- Specifying a name for the dataset to which simulation data will be saved
- Automatically displaying data when the simulation is finished
- Selecting a different machine on which to run the simulation

To specify simulation setup options:

1. From the Schematic window, choose Simulate > Setup . A Simulation Setup dialog box appears:



2. The data calculated during a simulation is saved in a dataset. The default dataset name is the same as the project. Change the name as desired.
3. By default, the Data Display will automatically launch when the simulation is complete. If the simulation results were displayed before, the same window will open if you specify its name in the Data Display field.
4. If you use remote simulation hosts, you can specify a machine other than the local one for running the simulation. For more information on how to set up remote machines, see the installation documentation for your platform.

Starting the Simulation

There are several ways to launch a simulation:

- Press F7 on the keyboard.
- Click the Simulate icon on the toolbar.
- Choose Simulate > Simulate from the Schematic window.
- Choose Simulate from the Setup dialog box while it is open.

When the simulation begins, a status and error message window appears. When the simulation is complete, the line

Simulation finished . at the bottom of the window indicates that the simulation has run successfully. The location of the dataset where the simulation data is saved is also noted.

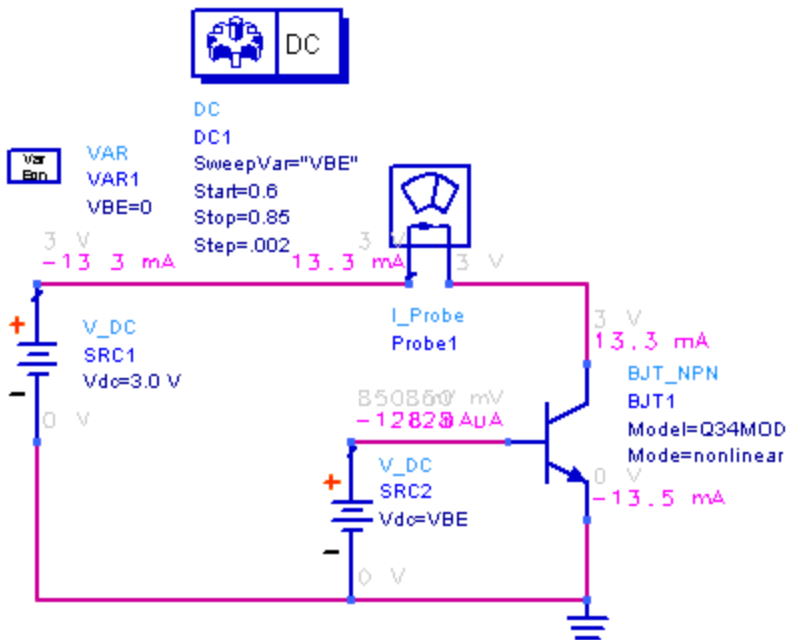
Displaying Simulation Data

You will typically want to view most of your results in a Data Display, however you also have the option to view DC results on the schematic and view lists of device operating point details. While it is not very useful in this example-given that the DC results are reported only for the last point in a sweep – it illustrates an important feature.

Viewing the DC Solution

1. Choose Simulate > Annotate DC Solution . DC voltages and currents appear at the pins of all the active devices and lumped elements, as shown below.

Note
Current is defined as positive if flowing into a device, so +13.5 mA is flowing from the emitter to ground.

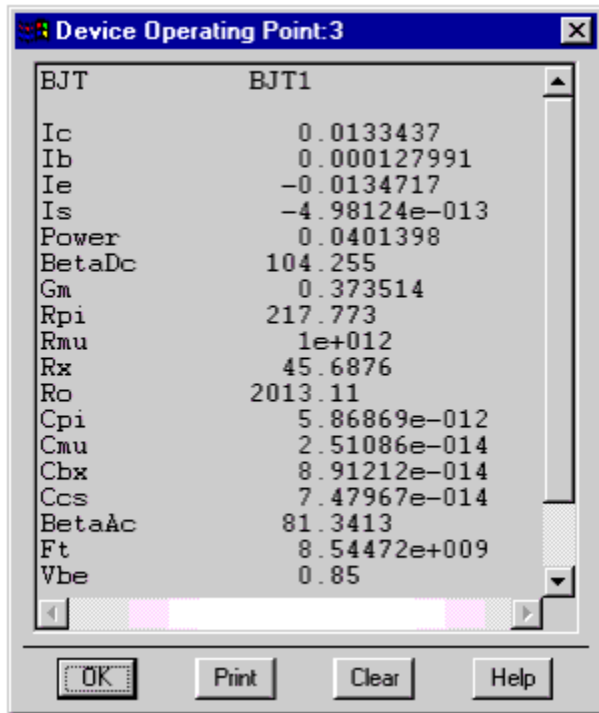


Note
To clear the annotations, choose Simulate > Clear DC Annotation from the main menu. You will have to resimulate to annotate again.

Viewing the Detailed Device Operating Point

1. In the same Schematic window, choose Simulate > Detailed Device Operating Point . Crosshairs appear.
2. Place the crosshairs over a transistor and click. A detailed DC operating point listing appears.
3. As you select other devices in a circuit, the DC operating point data for additional devices is added to the list.

For more information about the parameters that are displayed in the list, refer to the documentation for the specified component.



Viewing the Brief Device Operating Point

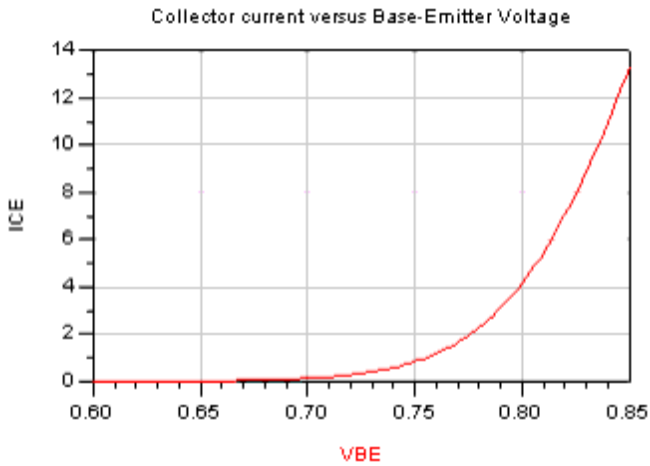
To view a subset of the above information that covers the most common parameters, choose Simulate > Brief Device Operating Point and select a device. The details are similar to those in the detailed list, but this list contains fewer parameters.

For details about the data that appears in the list, refer to the selected component's documentation.

Viewing More Results in the Data Display

The remainder of the simulations results can be viewed using the Data Display. For complete details on how to use the

Data Display, refer to [Data Display](#) and to the [Advanced Design System Quick Start](#). The results of this simulation are in DC_op_point.dds. In this example, collector current versus V be at Probe1 is displayed.



There are various options for plotting and scaling data. To edit general plot characteristics, select the Plot Options tab. This allows you to enter a title and axis label. It also allows you to deselect Auto Scale and enter a scale that zooms in on a range of interest. You can also choose between linear and log scales here, and select grid characteristics.

To edit a trace before placing it (unless you have selected the List plot type), select the parameter under Traces, then click Trace Options . Select the Trace Type , Trace Options , and Trace Expressions tabs to select, for example, trace patterns, trace colors, and fonts for labels, as well as to edit mathematical expressions for display.